DR. BALOGH IMRE

Bevezetés az Üzleti Adatbányászatba SPSS Modeler alkalmazásával

2019.

TARTALOMJEGYZÉK

Bevezetés

A jegyzet bevezetést ad az üzleti adatelemzés piacvezető szoftverének használatába. Az IBM SPSS Modeler-nek jelenleg a 14.1-es változata a legújabb, de a jegyzetben néha a 13.0 változatra és a 12-es változatra is hivatkozunk. A 12-es változat korábbi neve Clementine volt illetve a 13.0-as változat átmenetileg a PASW Modeler néven szerepelt. 2010 eleje óta egységes a névhasználat, azóta ugyanis a korábbi SPSS Inc. beolvadt az IBM-be.

A jegyzethez szervesen hozzátartoznak a példa streamek és példa adatforrások, amelyekre folyamatosan hivatkozik.

Az első változat összeállításában és kipróbálásában nagy segítséget nyújtott az Alkalmazott Informatika és Információmenedzsment Tanszék két korábbi munkatársa Grujber Zoltán és Zsíros Péter, köszönet illeti ezért őket.

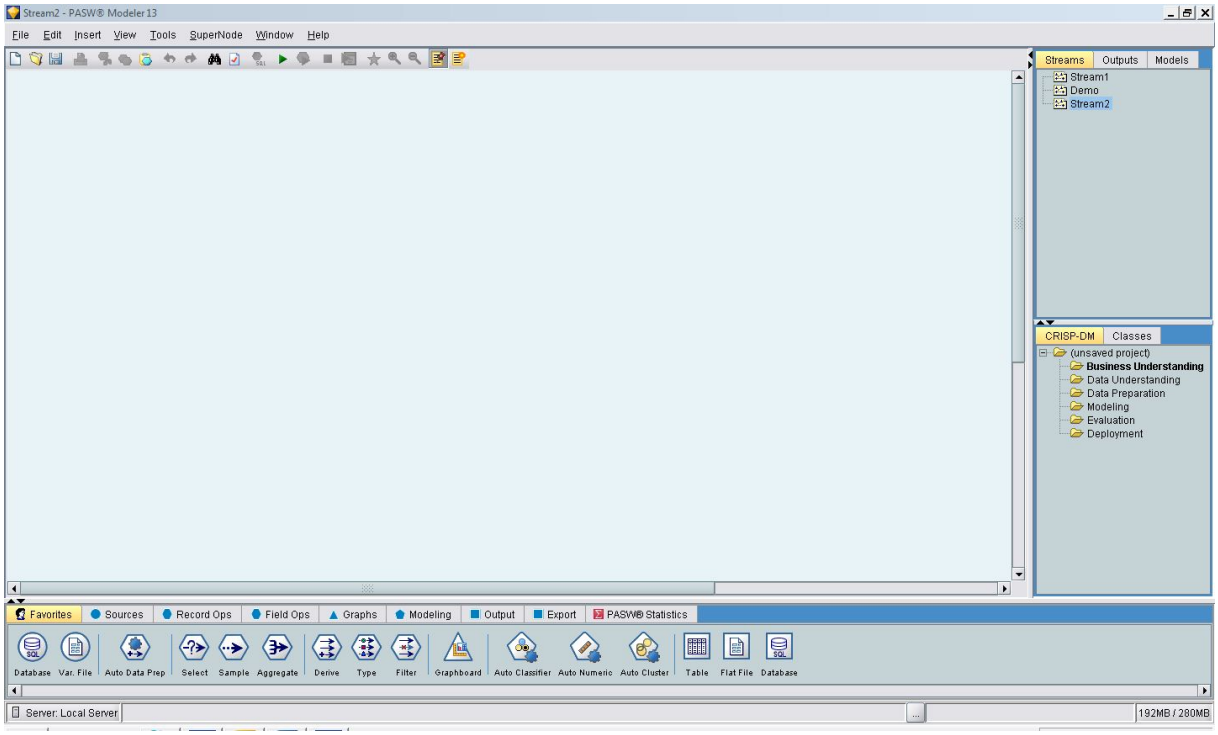
Ugyancsak köszönet illeti Tarr Mártát, aki a kéziratot gondozta.

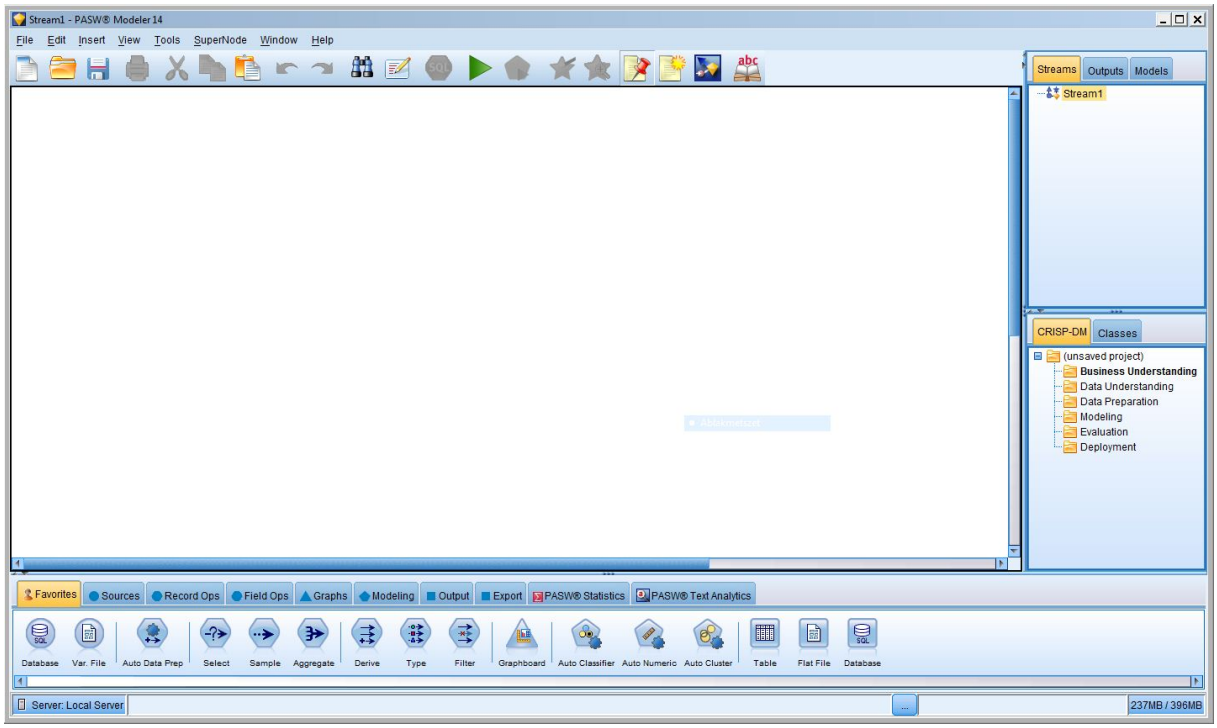
Szombathely-Budapest 2010.

Balogh Imre

1. Ismerkedés a Modelerrel

A program ablak felépítése:





Az ablak részei:

- Menüsor
- Eszköztár
- Munkaterület
- Node paletta
- Managers paletta
- Projekt paletta

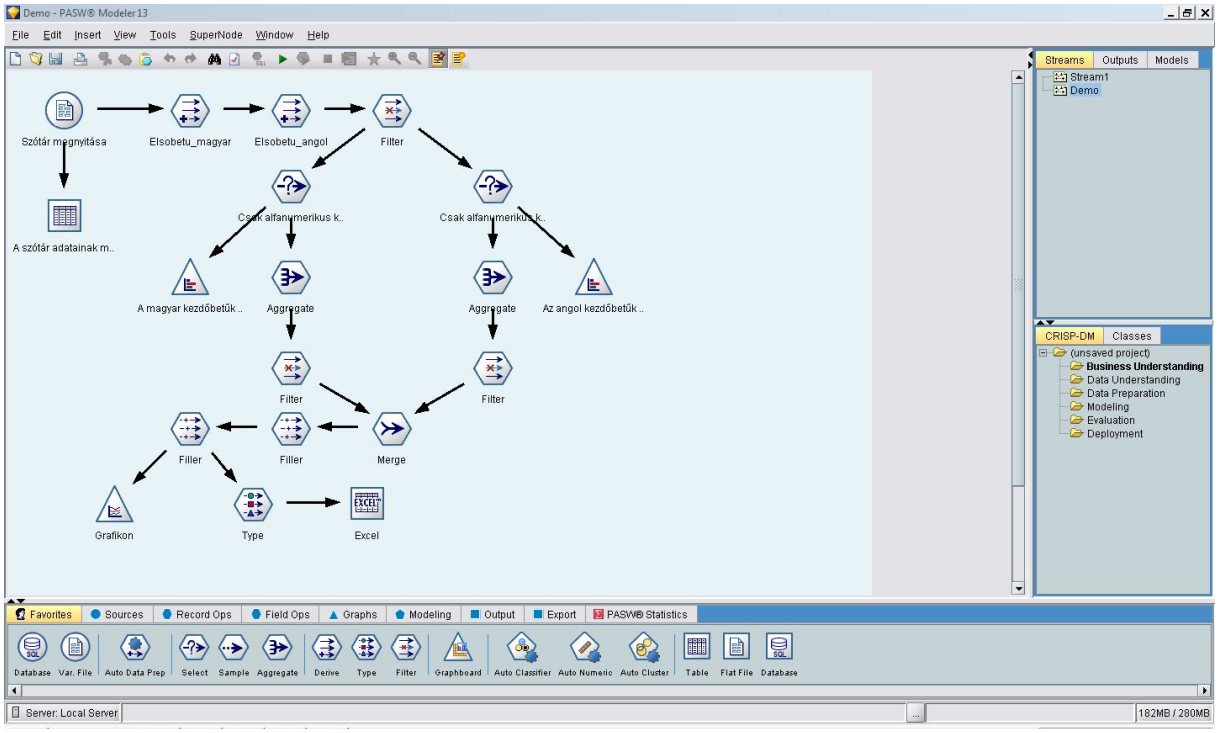
A paletták külön-külön a VIEW menüpont alatt ki-be kapcsolhatók.

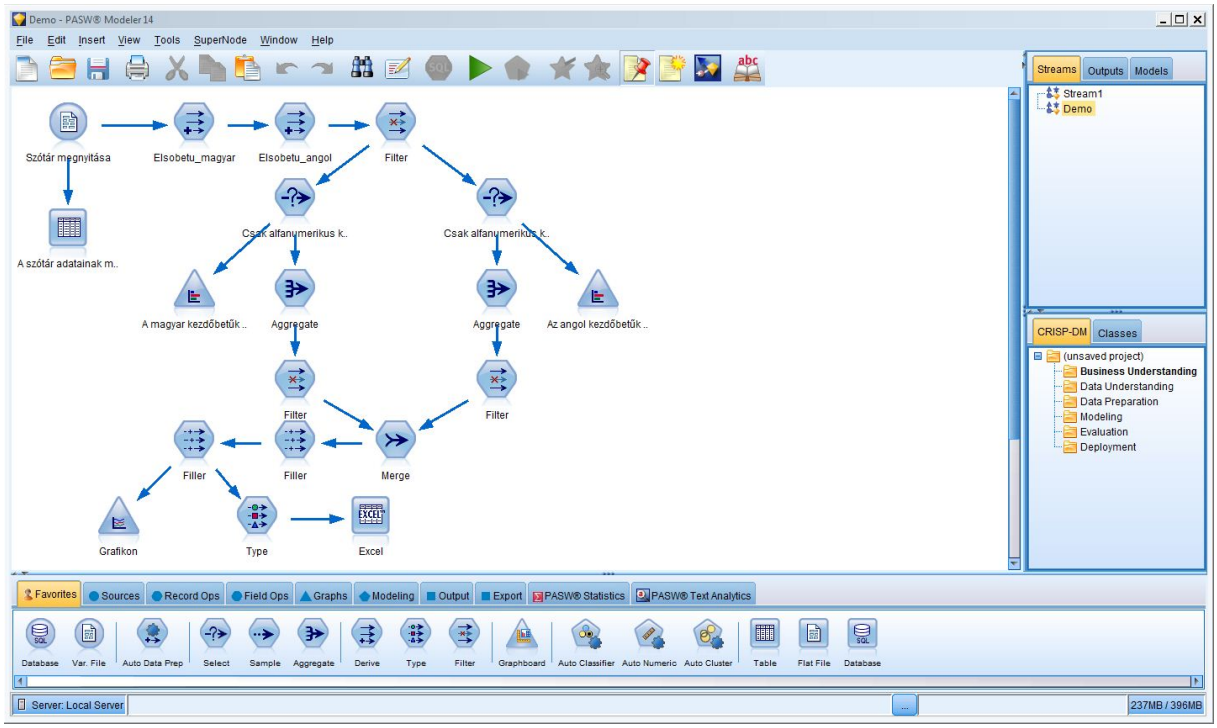
Stream megnyitása

Gyakorlat

Használja a FILE menü OPEN STREAM parancsát, és nyissa meg a Demo.str állományt!

A művelet végrehajtása után az alábbi ábra látható:

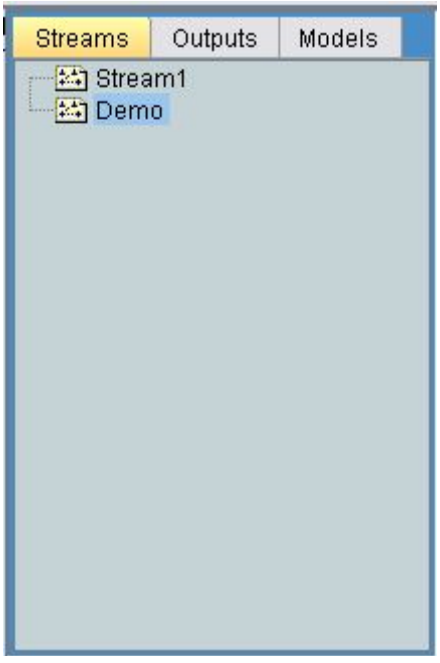


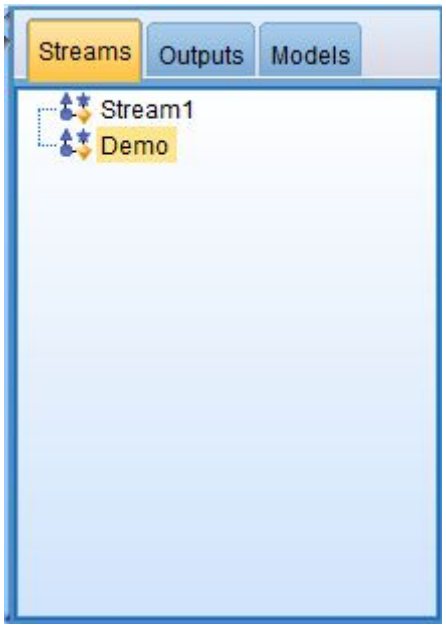


A munkaterületen a megnyitott stream felépítése látható. Vegyük sorra a képen látható elemeket és jelentésüket:

- A hatszög, kör, háromszög és négyzet alakú ikonokat node-nak nevezik.
- A node-okat összekötő fekete nyilak az adatok áramlásának irányát jelzik.

A jobb oldali Managers paletta Streams lapján a megnyitott Stream-ek listája látható:





Gyakorlat:

Váltson át a Stream1 –stream-re, majd vissza a Demo stream-re.

Zárja be a Stream1 stream-et.

Az egérrel jelölje ki az egyes node-okat (egy kattintás).

„Fogd-vidd" művelettel rendezze át a node-okat a munkaterületen.

Bármely node-on történő duplakattintással nyissa meg a node beállításait tartalmazó ablakot.

Node típusok

A node-ok a programablak alján található node palettáról helyezhetők el a stream-ben.

*Ha a Node paletta nem látható, kapcsolja be a **VIEW** menüpont **NODES PALETTE** parancsával!*

A paletta a Favorites fülön kívül hat további fülön csoportosítva tárolja a felhasználható node-okat:

- Sources
- Record Ops
- Field Ops
- Graphs
- Modeling
- Output

Gyakorlat:

Keresse meg, hogy a munkaterületen látható stream-ben lévő egyes node-ok mely fülön találhatók meg! Figyelje meg az egyes node-ok alakját! (hatszög, kör, stb.)

A Sources fülön lévő (kör alakú) node-ok különböző adatforrásokból származó, táblázatos elrendezésű adatok beolvasására szolgálnak.

A Record Ops fülön lévő (hatszög alakú) node-ok alkalmazásával táblázatok soraival végezhetők műveletek.

A Field Ops fülön lévő (hatszög alakú) node-ok alkalmazásával táblázatok oszlopaival végezhetők műveletek.

A Graphs fülön lévő (háromszög alakú) node-okkal az adatok grafikusan ábrázolhatók.

A Modeling fülön lévő (ötszög alakú) node-ok matematikai modellező algoritmusok végrehajtását teszik lehetővé.

Az Output fülön lévő (négyzet alakú) node-ok pedig az adatok különböző formába történő exportálására képesek.

A Demo stream értelmezése, futtatása

Fontos, hogy:

- Kör alakú node-ból csak kiindulhatnak a nyilak.
- Háromszög és négyzet alakú node-ba csak mutathatnak a nyilak.

A hatszög alakú node-ok a beérkező adatokon valamilyen oszlop vagy sor műveletet végeznek el, majd belőlük az adatfolyam tovább kapcsolható más node-ok irányába.

A legtöbb hatszög alakú node csak egy helyről fogadhat adatokat (egy nyíl mutathat rájuk). Kivétel a Record Ops fülön található Merge és Append node.

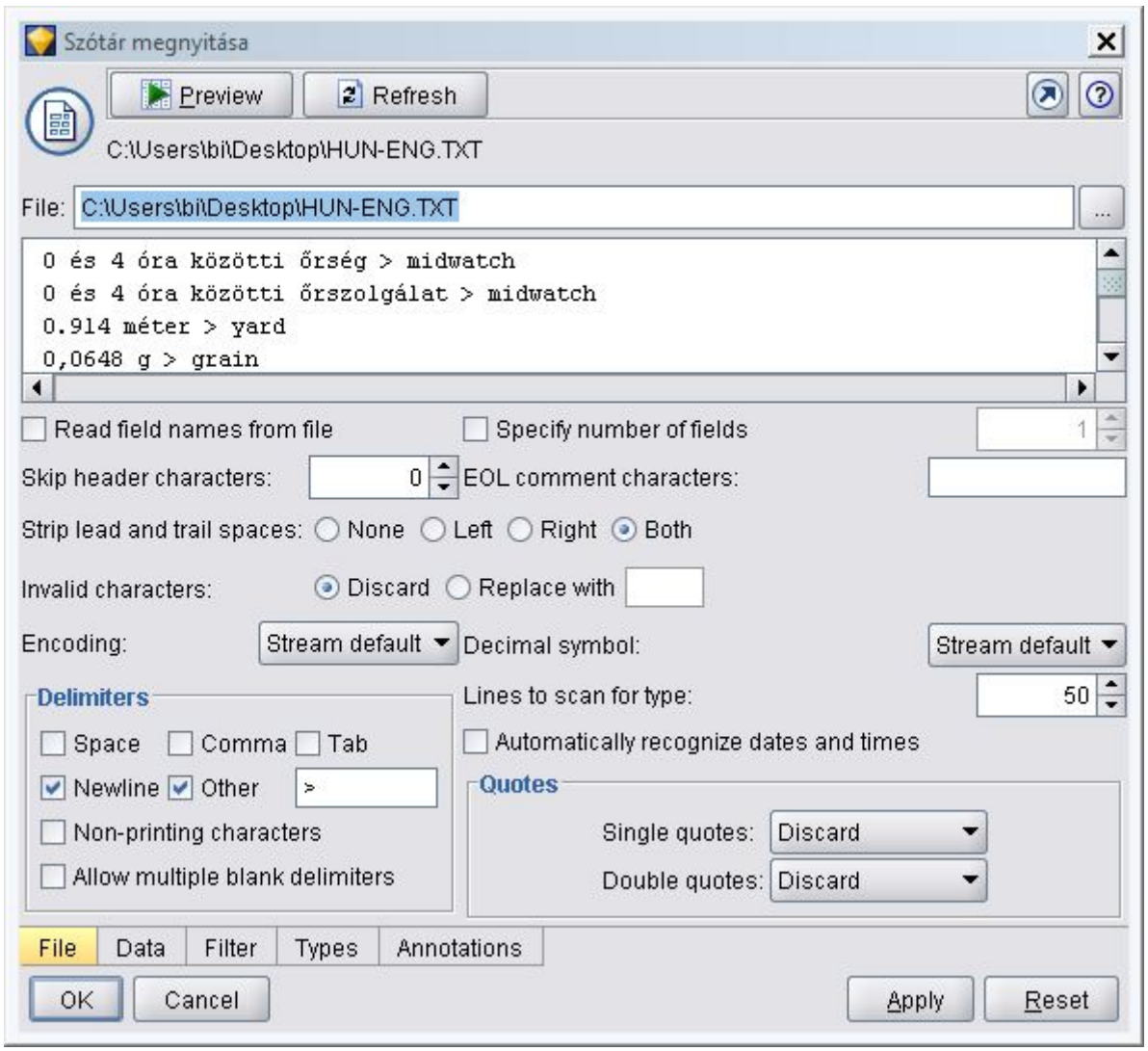
Bármely kör, vagy hatszög alakú node-ból több nyíl is kiindulhat, vagyis több egymástól független ágon is folyhat az adatok további feldolgozása.

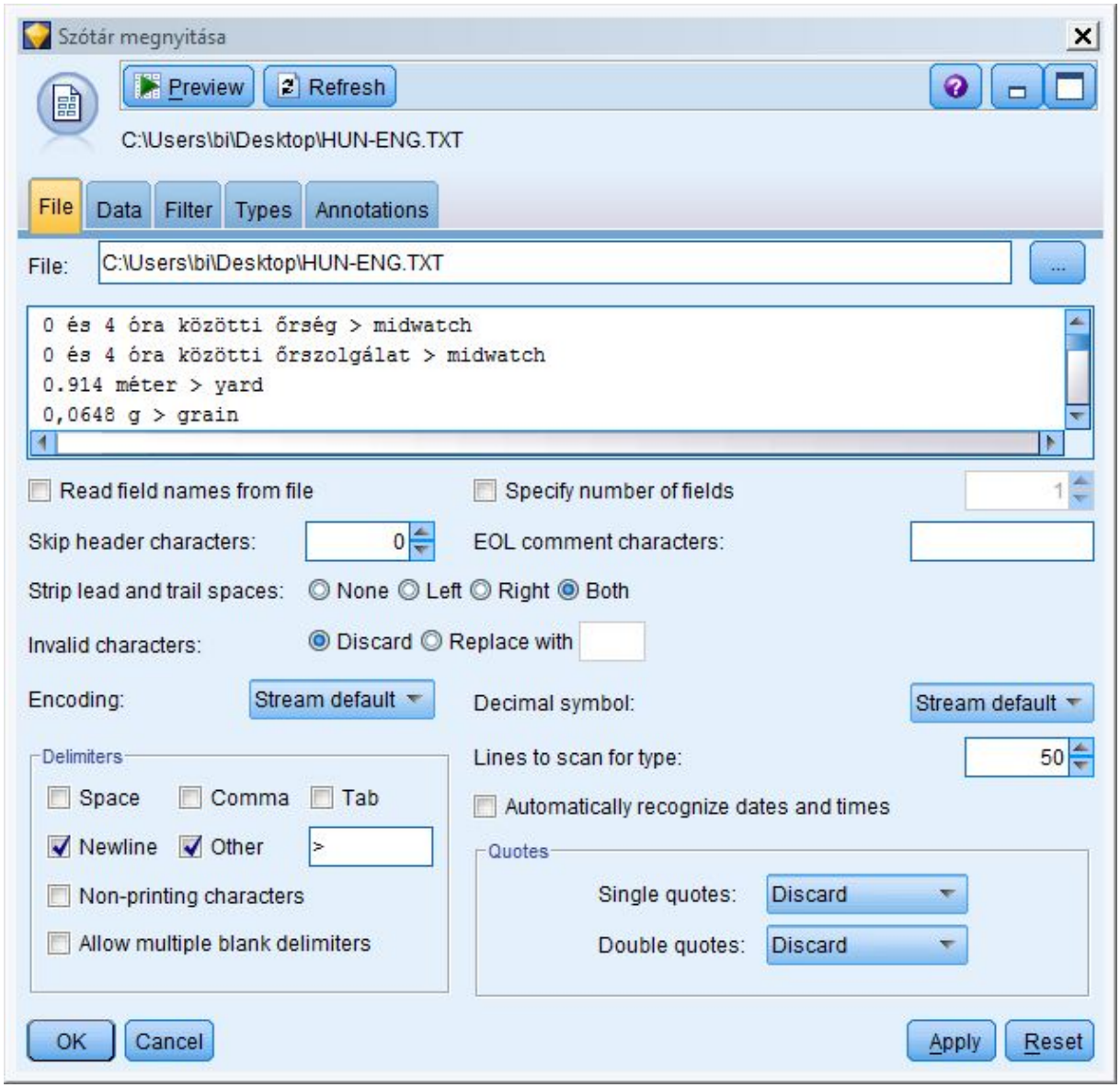
Gyakorlat:

A Demo stream több olyan node-ot is tartalmaz, amely egy egy adatfeldolgozási folyamat végét jelenti. Keresse meg ezeket a node-okat.

Keresse meg, hogy melyik node-ból indulhat ki az adatfeldolgozó folyamat.

A node-on történő dupla kattintással nyissa meg a node beállításait tartalmazó ablakot.





Az ablak alján található fülek az egyes beállításcsoportok közötti válthatásra szolgálnak.

Lehetőségek:

- Ezzel a node-dal olyan szövegfájlokból olvas be adatokat, melyek egy-egy sora megfelel a beolvasandó táblázat egy sorának, és a sorokon belül az oszlopértékek valamilyen határoló karakterrel vannak elválasztva.

Gyakorlat:

Keresse meg az ENG-HUN.txt fájlt, és nyissa meg a jegyzettömbben.

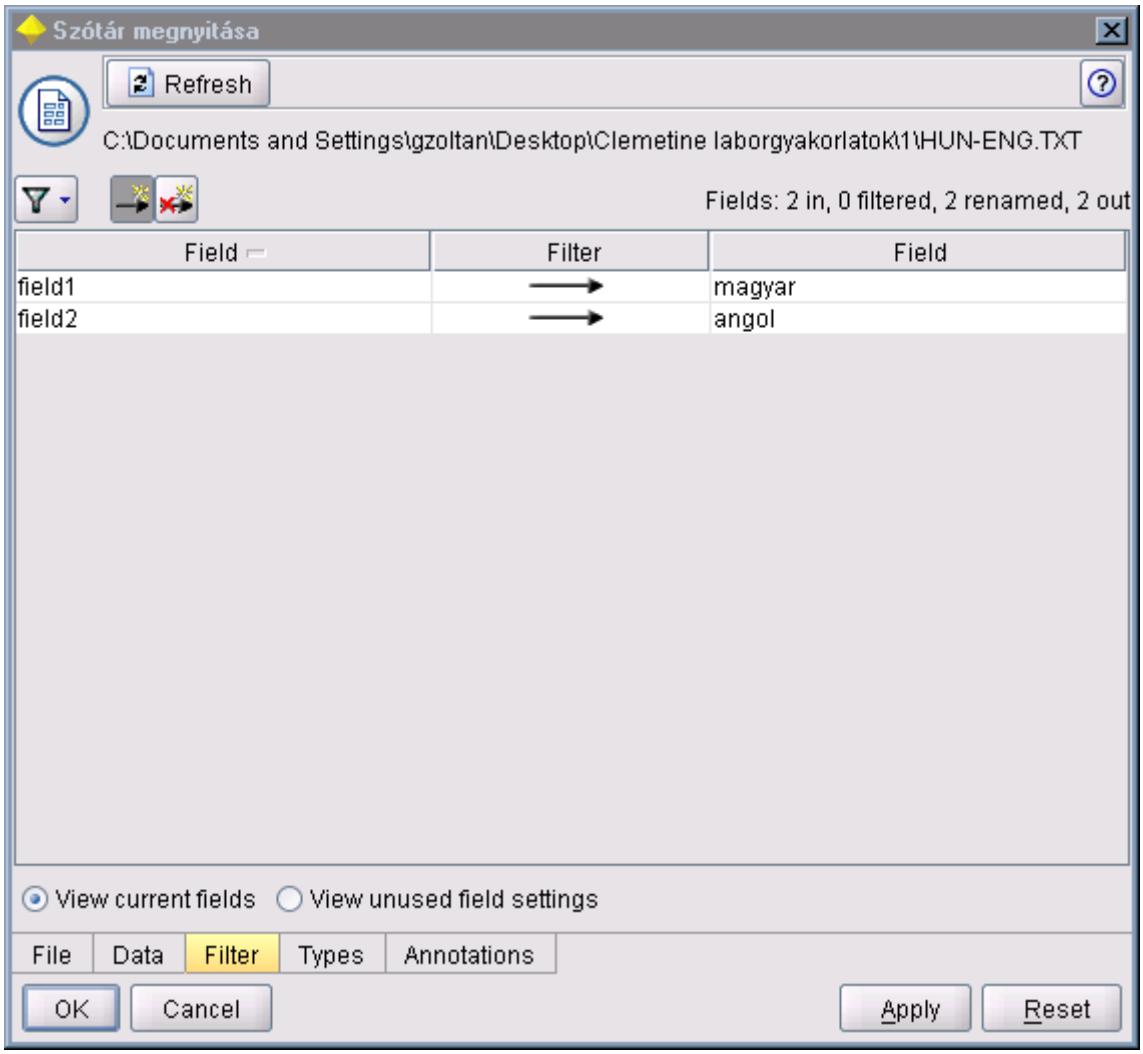
Figyelje meg, hogy mi lehet a határolókarakter a sorokon belül, valamint vannak-e felesleges szóköz karakterek az adatok között.

Ellenőrizze, hogy a File rovatban helyesen van megadva a fájl elérési útja. Ha nem, akkor adja meg pontosan.

- A **DELIMITERS** rovatban adható meg, hogy a szövegfájlon belül az adatok milyen határolókarakterrel vannak elválasztva. Kettő jelölhető be: A **NEWLINE** a sorokat választja el, a **>** pedig a sorokon belül az oszlopokat.

- A **STRIP LEAD AND TRAIL SPACES** rovatban a **BOTH** megadásával a beolvasott adatok mindkét végéről eltávolíthatók a felesleges szóköz karakterek.

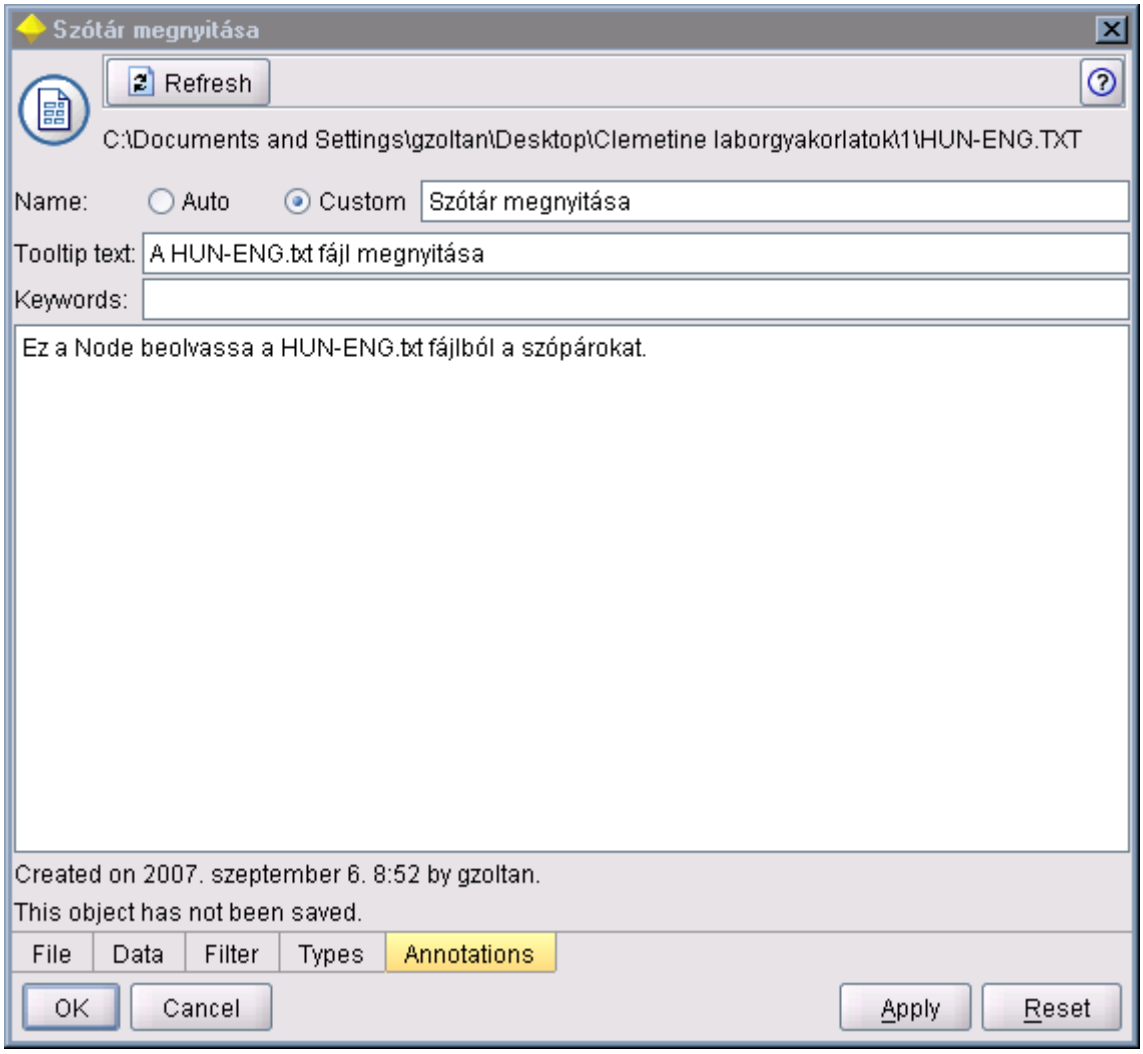
Váltson át a Filter fülre!



- Ezen a fülön azt állítottuk be, hogy a továbbiakban a szövegfájlból beolvasott két oszlopnak (változónak) mi legyen a neve.

Minden egyes node beállításait tartalmazó ablakon megtalálható az **ANNOTATIONS** fül.

Váltson át az Annotations fülre!



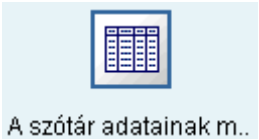
- Ezen a fülön nevet és leírást adhatunk a node-hoz.

A továbbiakban figyeljen az egyes node-oknál erre a lehetőségre. Sok esetben részletes leírást találhat a node céljáról, működéséről.

Zárja be a beállításablakot, az OK megnyomásával, hogy a módosult beállításait elmentse!

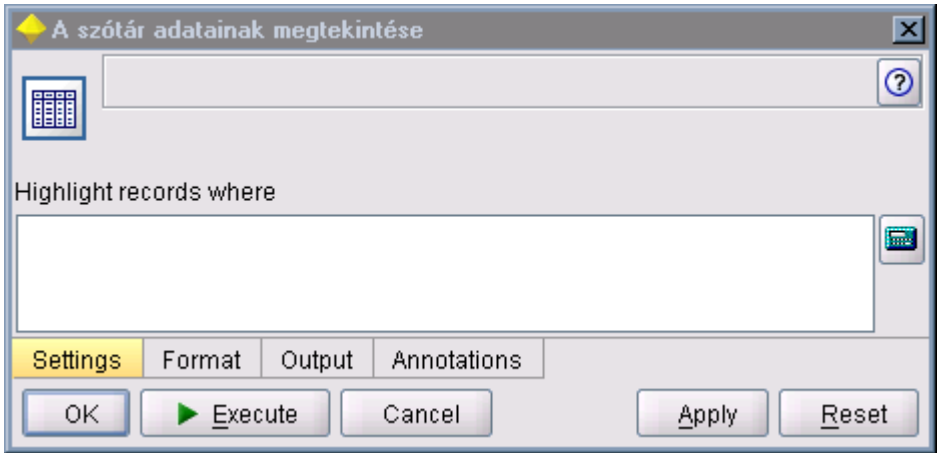
Most, hogy láttuk milyen adatokkal is dolgozik a stream-ünk, ugorjunk egy nagyot, és nézzük meg milyen eredményeket adnak az output node-ok.

Az egyik leggyakrabban használt output node a **TABLE**. Ebből jelenleg egyet tartalmaz a stream, közvetlenül a source nodunkhoz kapcsolva:

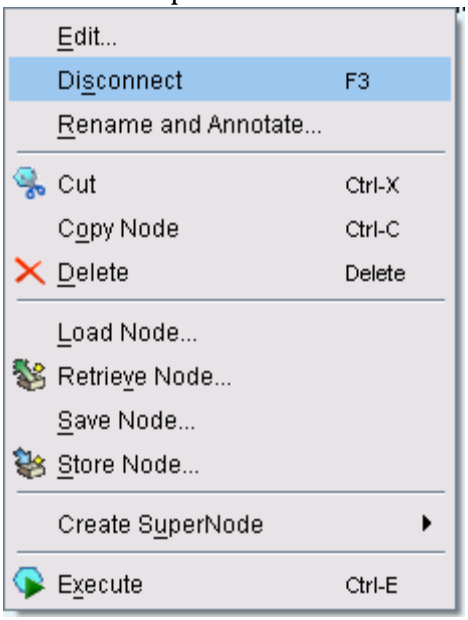


A node eredményeinek megtekintéséhez futtatnunk kell a node-ot. Ezt a következő módokon tehetjük meg:

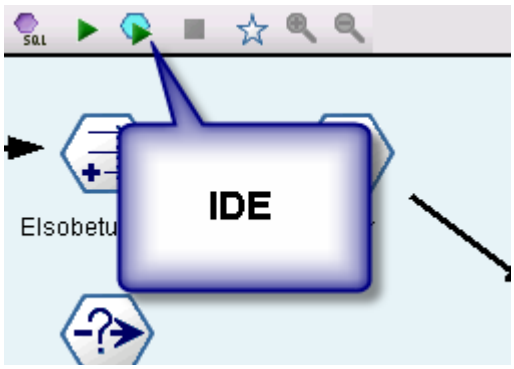
- Megnyitjuk a node beállításait tartalmazó ablakot, ahol is az **EXECUTE** gombra kattintunk:



- A node-on történő jobbklikkes helyi menüből kiválasztjuk az Execute menüpontot:



- Kijelöljük a node-ot, majd leütjük a **CTRL+E** billentyűkombinációt.
- Kijelöljük a node-ot, és az eszköztár **EXECUTE SELECTION** parancsára kattintunk:

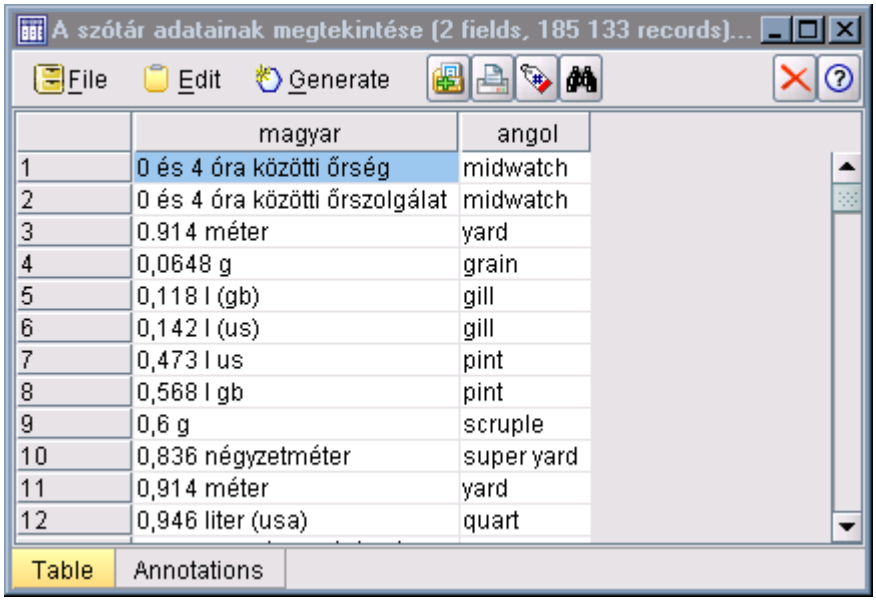


Ha az eszköztáron a sima zöld háromszög ▶ ikonra kattint, akkor az adott stream összes output node-ját lefuttatja egyszerre!

Az output node-ok futtatásának eredménye mindig új ablakban nyílik meg.

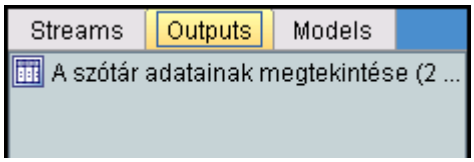
Futtassa a Table output node-ot!

Eredményként valami hasonlót kell látnunk:



	magyar	angol
1	0 és 4 óra közötti őrség	midwatch
2	0 és 4 óra közötti őrszolgálat	midwatch
3	0.914 méter	yard
4	0,0648 g	grain
5	0,118 l (gb)	gill
6	0,142 l (us)	gill
7	0,473 l us	pint
8	0,568 l gb	pint
9	0,6 g	scruple
10	0,836 négyzetméter	super yard
11	0,914 méter	yard
12	0,946 liter (usa)	quart

Figyelje meg, hogy a Managers paletta Outputs fülén is megjelent az eredmény!



Ha az output eredményét tartalmazó ablakot bezárja, az output fülön továbbra is megmarad a bejegyzés, melyre duplán rákattintva ismét megnyithatja azt.

Ha ismételten futtatja a node-ot, akkor az egy újabb eredményhalmazt generál, mely ugyancsak megjelenik az output fülön, az előző futtatás eredményétől függetlenül. Hiszen lehetséges, hogy közben például a source node beállításainak módosítása miatt a két eredményhalmaz nem is egyezik meg.

Az output fülön lévő listából bármelyik elemet törölheti, elmentheti, ha rákattint az egér jobb gombjával, és a megfelelő parancsot kiválasztja!

Ha az output node eredményét tartalmazó ablakot úgy zárja be, hogy az eszköztáron található  *ikonra kattint, akkor egyúttal az output listáról is törli a bejegyzést!*

A Table node-ot leggyakrabban arra használhatjuk, hogy a stream különböző pontjain elhelyezve részeredményeket jeleníthessünk meg, ellenőrizhessük az addig felépített műveletek eredményeit.

Futtassa le a stream többi output node-ját is!

Nézze meg az egyes node-ok leírásait!

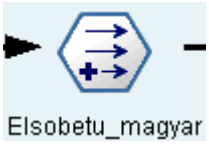
Próbálja értelmezni az egyes node-ok beállításait!

Amint az az eredményekből látható, a stream feladata a két nyelv szókezdő-betűi eloszlásainak a vizsgálata.

A Demo stream működése

A következőkben részletesen végighaladunk a stream-et alkotó node-ok szerepén, beállításain, hogy a stream működését teljes egészében világosan lássuk.

A HUN-ENG.txt fájlt beolvasó source node után közvetlenül egy **DERIVE** node található, a **FIELD OPS** nodepalettáról:

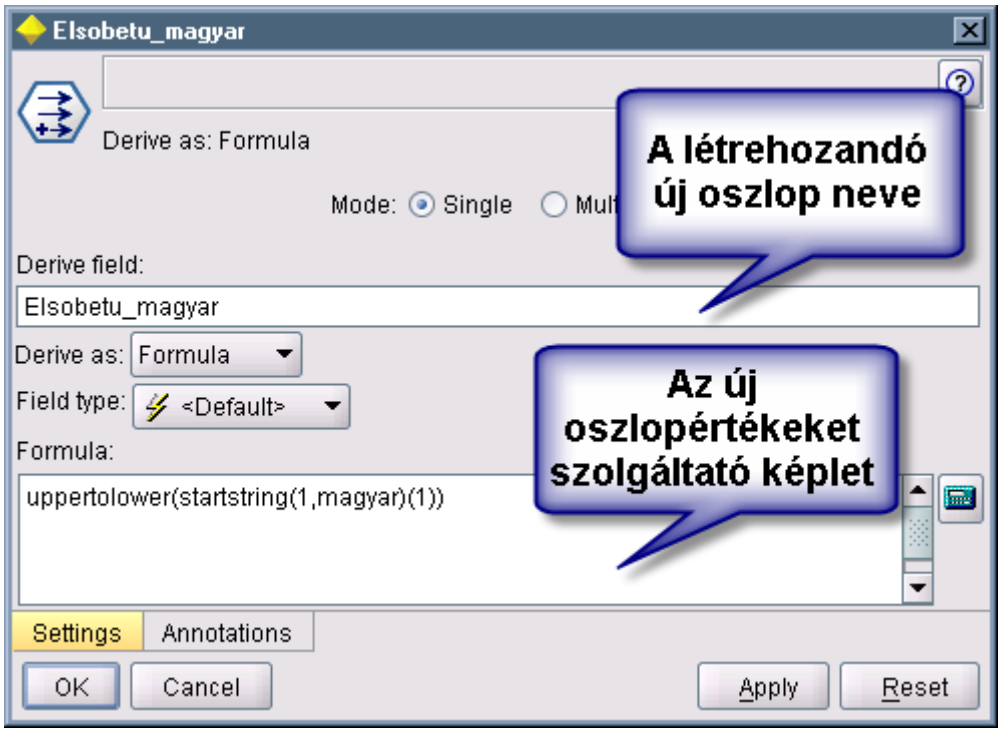


A Derive node-dal új számított oszlopot, vagy oszlopokat helyezhetünk el az adattáblánkban.

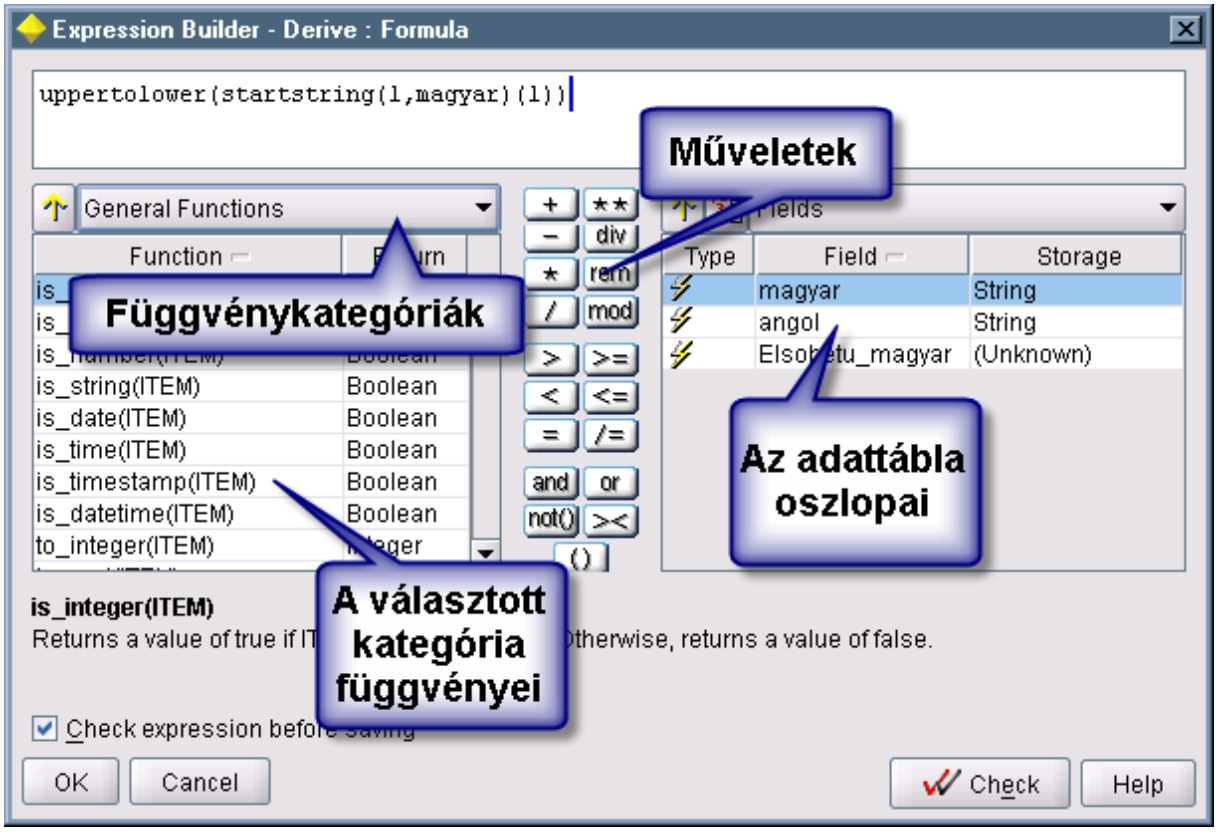
A node feladata, hogy hozzon létre egy új oszlopot **ELSOBETU_MAGYAR** néven, és töltse fel a magyar szavak kezdőkaraktereivel.

A node beállításainak megtekintéséhez kattintson duplán az Elsobetu_magyar node-ra!

Megjelenik a node tulajdonságpanelja:



A képletek összeállításában jelentős segítséget kaphatunk a számológép ikonra történő kattintással:



A kifejezés-szerkesztő használatával kapcsolatos gondolatok:

- Az ablak tetején található beviteli mezőben a képletet „kézzel” is szerkeszthetjük. A bevitt képlet hibáinak a kijavításához, vagy csak egyszerűbb módosítások elvégzéséhez ez az ajánlott módszer.
- Ha egy konkrét feladatra keresünk függvényt, akkor szűkítsük a megjelenített függvénylistát a függvény kategóriájának kiválasztásával.
- A kijelölt függvény leírása elolvasható az ablak alján.
- Duplakattintással a függvényeket és az oszlopneveket elhelyezhetjük a képletünkben.
- Az elkészült képletet a Check gombbal ellenőrizhetjük, hogy szintaktikai hibáktól mentes-e.
- A beírt kifejezésekben a program különbséget tesz a kis és nagybetűk között.

A Clemetine kifejezések a következő elemekből épülhetnek fel:

- Értékek (konstans szövegek, számok, dátumok, stb.),
- Mező nevek (oszlop nevek),
- Operátorok (műveleti jelek),
- Függvénynevek.

Térjünk vissza a node által tartalmazott képletre:

UPPERTOLOWER(STARTSTRING(1,MAGYAR)(1))

A képlet megértéséhez meg kell ismerkednünk két szövegkezelő függvénnyel, valamint a Clemetine különböző adattípusai közül kettővel.

- Szöveg adattípus:

  Az adattáblánkban meglévő két oszlop (magyar, angol) szöveg típusú adatokat tárol. Képletekben, ha konstans szöveget szeretnénk megadni, akkor azt dupla idézőjelek közé zárjuk, mint például: „Alma”.

- Karakter adattípus:

  Gyakorlatilag egyetlen karakter tárolására képes. Nem egyezik meg egy egykarakteres, de szöveges adattípusú értékkel. Képletekben konstans értékként egyszeres idézőjelek közé zárva jelöljük, mint például: `R`.

  FONTOS! Nem ’ jelek közé, hanem ` jelek közé zárjuk. Magyar billentyűzetkiosztás esetén: AltGr+7.

  Ha a képletünkben egy oszlopban letárolt szöveg, vagy bármely kifejezés eredményeképp visszaadott szöveg típusú érték egyetlen karakterére szeretnénk hivatkozni (karakter adattípusként), akkor azt a következő szintaktikával tehetjük meg: **OSZLOPNEV(I)**, ahol i az adott szöveg i-edik karakterét jelenti.

- Uppertolover függvény:

  Karakter adattípusú paramétert vár, és kisbetűsre alakítja.

- Srtartstring függvény:

  Egy tetszőleges szöveg első valahány karakterét adja eredményül.

Mindezek után már nem nehéz megfejteni a képlet eredményét: a magyar szó első karaktere lesz, nagybetű formájában.

Kattintson az annotation fülre a node tulajdonságablakában. Ott is megtalálja a képlet tömör magyarázatát!

Értelmezze a következő node működését! (Elsobetu_angol)

Következik a Filter node:

Ez a node is a Field Ops palettáról származik. Általánosan a következő célokra használjuk:

- Oszlopok átnevezése.
- Oszlopok kikapcsolása a további adatfolyamból.

Ha megnézzük a node beállításait, akkor tisztán láthatjuk, hogy az adattábla eredeti két oszlopát, amelyek a szavak teljes alakját tárolták, kizártuk a további feldolgozásból. Ennek két célja van: egyrészt ne terhelje feleslegesen a rendszert a további számításoknál, másrészt tisztább és áttekinthetőbb a további munka, ha a képletszerkesztőben nem is látjuk már ezeket az oszlopokat.

Ettől a node-tól kezdve az adatfeldolgozó folyamatunkat szétválasztjuk két irányba, amelyek később ismét egyesülnek majd, de ez még ráér…

Az egyik ág a magyar kezdőkarakterek eloszlásával foglalkozik, míg a másik az angol kezdőbetűkkel.

Mindkét ág első node-ja egy Select node, a Record Ops palettáról:

A Select node az adattábla sorainak a szűrésére szolgál. Segítségével bizonyos feltételek alapján sorokat zárhatunk ki a további adatfeldolgozásból.

Miért is merült fel, hogy zárjunk ki sorokat? Könnyen választ kaphatunk, ha megnézzük, hogy a stream-ünk eddigi része milyen eredményeket ad. Ehhez a következőre lenne szükség:

Helyezzünk el egy Table (output) node-ot a 

node-unk után, és futtassuk le az eredmények megtekintéséhez!

Új node-ot a következő módokon helyezhetünk el a munkaterületen szerkesztett stream-ben:

- Duplán rákattintunk:

  Figyeljük meg, hogy noha a munkaterületen lévő node-ok közül akár többet is kijelölhetünk egyszerre, akkor is csak egy aktive node van. Az amelyiket szaggatott keret vesz körül. Ha duplakattintással helyezzük el az új node-ot, akkor az automatikusan kapcsolódni fog az éppen aktuális node-hoz.

- Fogd-vidd művelet:

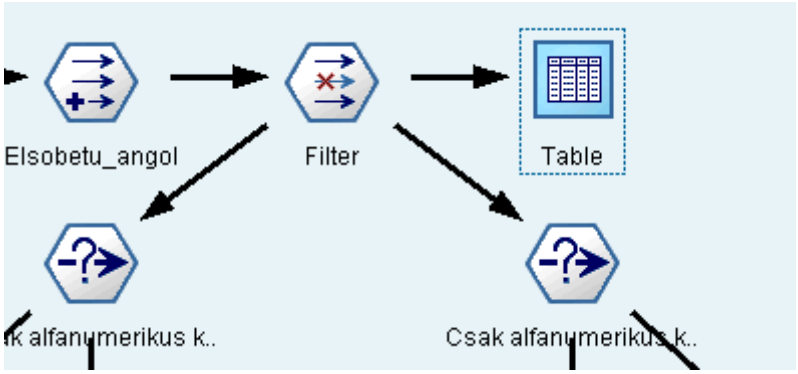
  Ilyenkor utólag kézzel adhatjuk meg, hogy melyik node-hoz kapcsoljuk az újonnan elhelyezett node-ot.

A stream-ben elhelyezett node-okat törölhetjük kijelölésük után, például a DEL billentyű leütésével.

Node-okat „kézzel” egymás után kapcsolni a következő módon lehet:

- Jelöljük ki a kapcsolni kívánt node-ot.
- Üssük le az F2 funkcióbillentyűt, vagy a node helyi menüjéből válasszuk a Connect parancsot.
- Kattintsunk a kapcsolni kívánt következő node-ra.

Ha sikerült a Table node-ot elhelyezni a megfelelő helyen, akkor a következőhöz hasonló képet kell látnunk:



Ha lefuttatjuk a Table node-ot, akkor pedig ez az eredmény kell megjelenjen:

Table (2 fields, 185 133 records)

File Edit Generate

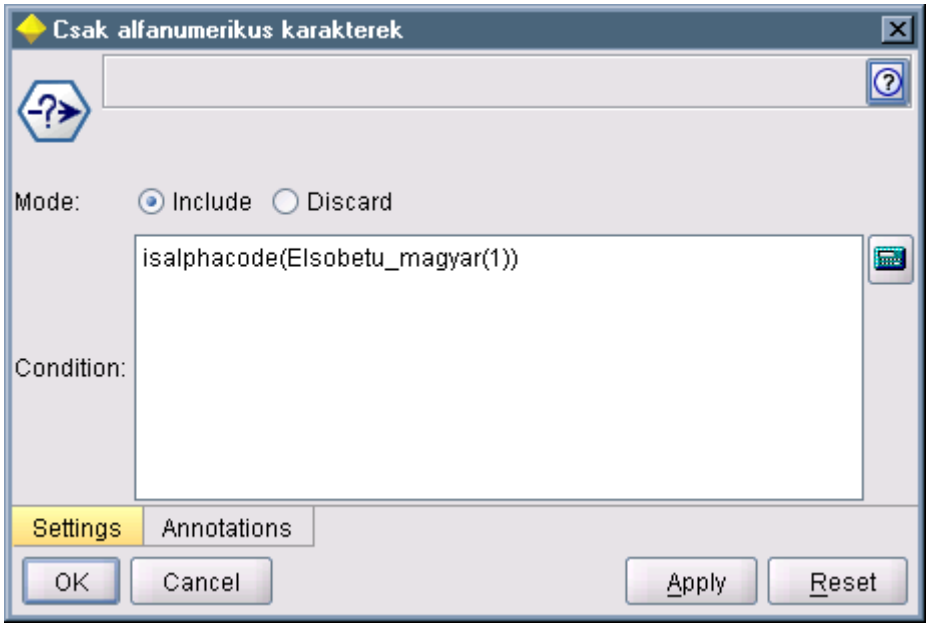
	Elsobetu_magyar	Elsobetu_angol
1	0	m
2	0	m
3	0	y
4	0	g
5	0	g
6	0	g
7	0	p
8	0	p
9	0	s
10	0	s
11	0	y
12	0	q
13	1	k
14	1	g
15	1	l
16	1	c
17	1	m
18	1	t
19	1	m
20	1	p

Table Annotations

Innen már tisztán látható, hogy miért is lenne hasznos szűrni a sorokat. Hiszen egyáltalán nem érdekesek elemzésünk szempontjából a számjegyek.

Bármely node után kapcsolhatunk közvetlenül egy Table output node-ot, hogy az addigi részeredményeket megtekintsük.

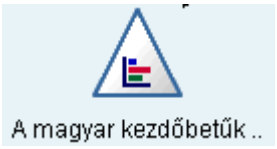
Nézzük akkor a Select node beállításait:



Ahogy az a beállításokon látszik, jelen esetben megtartjuk azokat a sorokat, ahol a feltétel igaz.

A beírt képlet magyarázatához kattintsunk az Annotation fülre!

Innen a stream ismét szétágazik. Pontosabban fogalmazva, az egyik ág nem is igazi ág, csupán egy Distribution node a Graphs palettáról:



Futtassa a node-ot és értelmezze az eredményt!

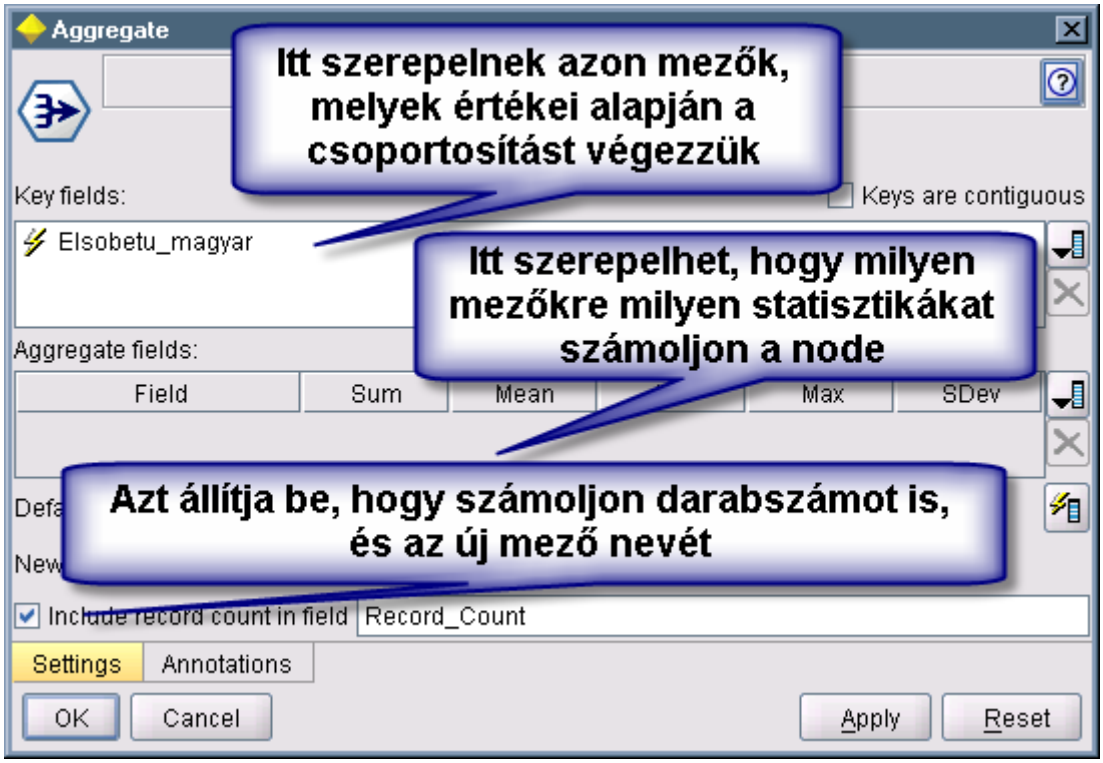
Nézze meg a node beállításait, és értelmezze a látottakat!

Nézzük tovább a másik ág működését. Ezt az ágat az az ötlet hívta életre, hogy a két nyelv kezdőbetűinek az eloszlását egy közös grafikonon jeleníthessük meg.

Azt, amit a Distribution node elvégzett, nevezetesen, hogy betűnként csoportosítva kiszámolta a csoportok elemszámát, elvégezhetjük egy másik node-típussal is úgy, hogy abból további számításokat eszközölhetünk. A stream-ben így egy Aggregate node következik a Record Ops palettáról:

Az Aggregate node fő feladata, hogy a táblában lévő sorokból csoportokat képezzen, és a csoportokra olyan alapstatisztikákat számoljon, mint például: minimum, maximum, átlag, stb. értéke egy adott mezőnek, oszlopnak. A csoportosítás minden esetben a sorok valamely oszlopban, vagy oszlopokban felvett azonos értékén alapul.

Nézze meg az Aggregate node beállításait!



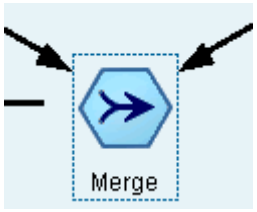
Rakjon be az Aggregate node után egy Table output node-ot is és futtassa a részeredmény megtekintéséhez!

A következő node, mind a magyar, mind az angol ágban már az ismert Filter node. Feladata a munkatábla két oszlopának átnevezése.

Nézze meg a Filter 

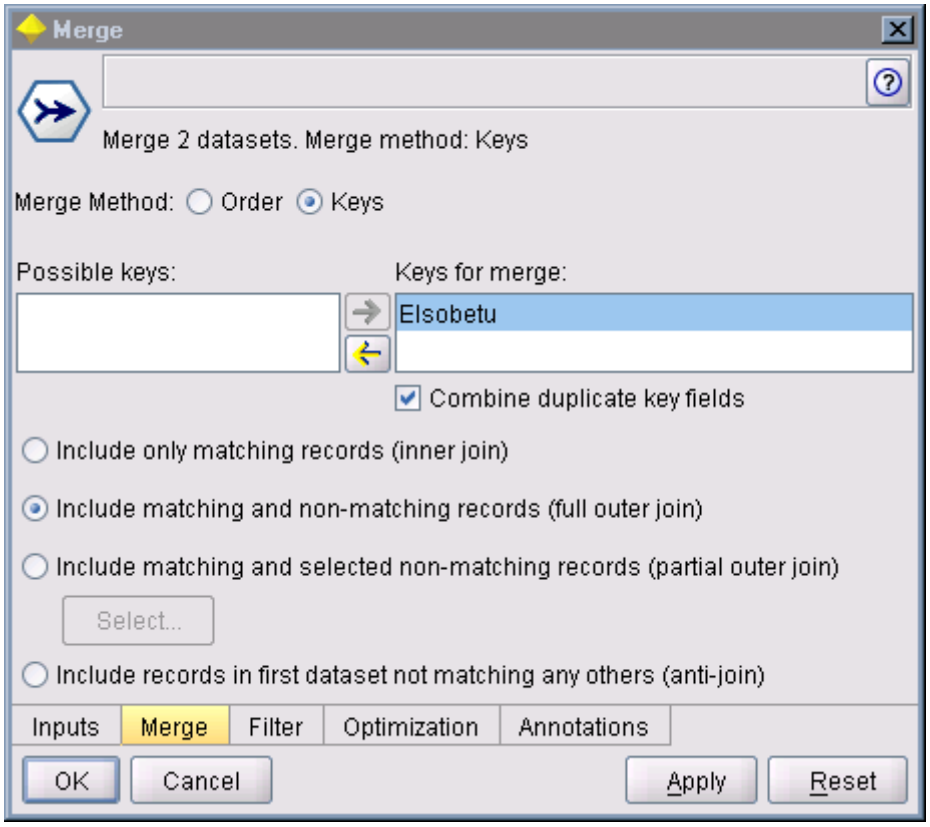
node-ok beállításait.

Megfigyelheti, hogy mindkét ágon az első betűt tartalmazó oszlopot Elsobetu névre neveztük át. Ennek oka, hogy ha a két adathalmazt egyesíteni szeretnénk, akkor ezek az oszlopok értékei alapján lehet a két tábla sorait párosítani. Az egyesítést elvégző Merge node feltételezi, hogy az összekapcsolást végző oszlopok azonos nevűek:



A Merge node az egyik olyan, amely több bemenettel rendelkezhet. A feladata, hogy a bemeneteihez kapcsolt táblákból a sorok egymáshoz rendelésével egy nagy közös kimeneti táblát állítson elő.

Nyissa meg a Merge node beállításait!



A Merge fülén állíthatjuk be a forrástáblák egyesítésének a módját. Jelen esetben az egyesítés kulcs (Keys) értékek egybeesésén alapul. A kulcsként használt oszlop neve: Elsobetu. A full outer join (mindkét irányú laza összekapcsolás) azt eredményezi, hogy mindkét táblából kerüljenek be azok a sorok is az eredménybe, ahol a másik tábla nem rendelkezik az adott kulcsértékkel.

Vegye észre, hogy a panel rendelkezik egy Filter füllel is. Gyakorlatilag ennek a node-nak része egy Filter node is, így egyúttal a kimeneti oszlopok nevei is megadhatók vagy megjelenítésük kikapcsolható.

Kapcsoljon a Merge node után egy Table node-ot, és futtassa!

Az eredmény magáért beszél:

Table (3 fields, 35 records) #1

File Edit Generate

	Elsobetu	Darab_angol	Darab_magyar
1	a	7325	5886
2	á	$null$	3235
3	b	7954	10077
4	c	11265	4558
5	d	6833	3018
6	e	4343	13971
7	é	4	2928
8	f	6328	12731
9	g	3868	4355
10	h	6185	10747
11	i	6237	3641
12	í	$null$	467
13	j	1299	2852
14	k	659	19850
15	l	4392	7569
16	m	5327	14281
17	n	2825	7056
18	o	3472	1996
19	ó	$null$	398
20	ö	$null$	2909

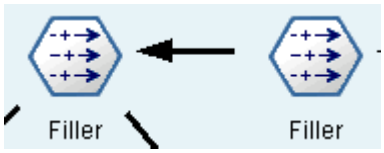
Table Annotations

Jól látható, hogy azon karakterek mellé került **$NULL$** érték, amelyek csak az egyik táblában fordultak elő. A $null$ érték ismeretlen, hiányzó értéket jelöl, és nem egyezik meg a matematikai nullával.

Természetesen szerepelhetne a 0 számérték is egy táblában. Ilyenkor az egyesítés után is szerepelni fog, mint 0 érték. De az nem a $null$ -al egyezik meg, mert az ismert. Tudjuk, hogy nulla.

Mielőtt grafikont készítenénk az adatainkból ezeket a $null$ értékeket szerencsés lenne kicserélni a matematikai 0-ra.

Ezt a feladatot látja el mindkét oszlopra a Filler node:



Nyissa meg a node-ok beállításait, és értelmezze a beállításokat!

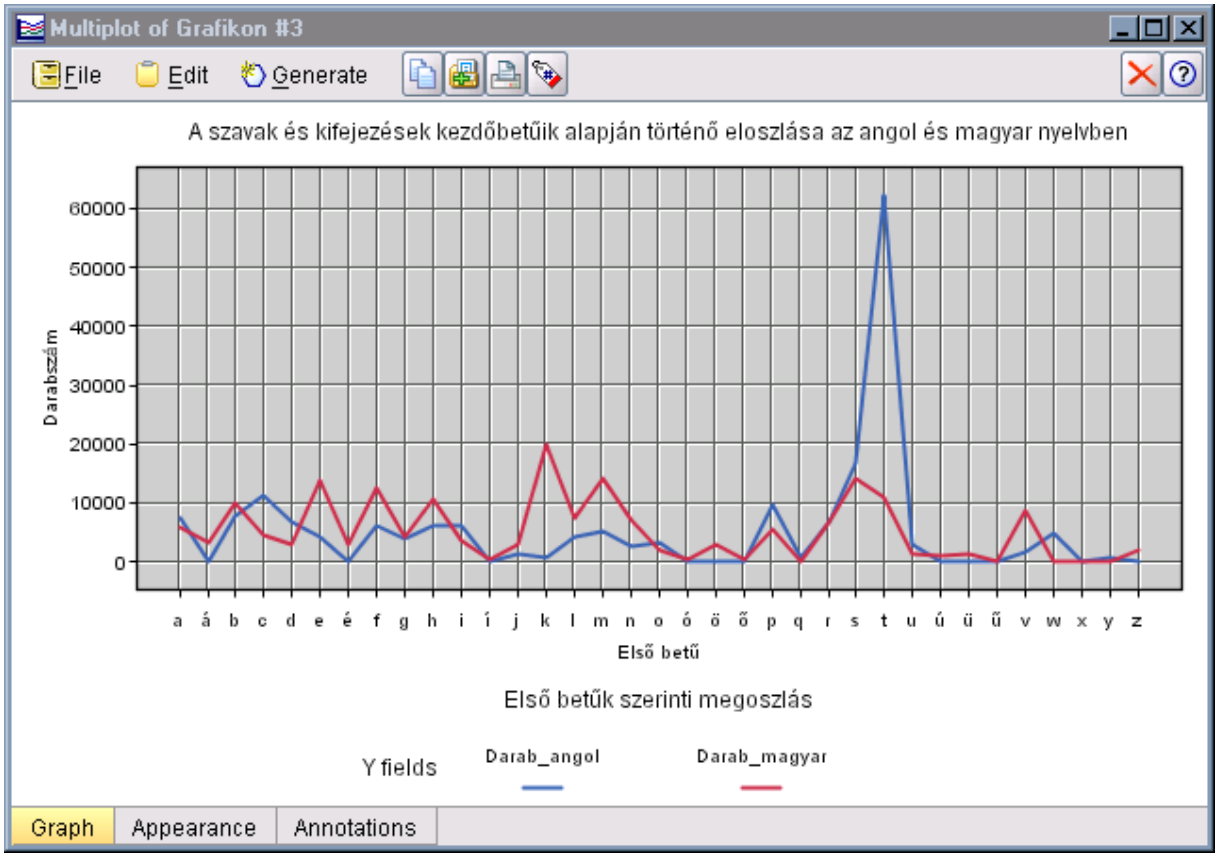
Innen már egyenes az út. Csupán két node van hátra. Az egyik egy grafikonon jeleníti meg a darabszámok alakulását, míg a másik az eddigi eredményeket egy Excel táblába menti el.

A két node beállításai meglehetősen magukért beszélnek! Értelmezze őket önállóan!

Az Excel output node elé beillesztett Type node (Field Ops palettáról) egyetlen célt szolgál. Biztosítja az Excel node számára a mentendő adatok típusainak megadását.(szöveg, szám, stb.)

Önálló feladat

Ha megnézzük az eredményeket grafikonon, akkor az tűnik ki nagyon, hogy az angol szavaknál túlnyomó többséget élveznek a t betűvel kezdődő szavak, míg a magyar esetében meglehetősen egyenletes az eloszlás:

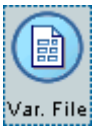


Alakítsuk át a stream-et úgy, hogy ne a szókezdő karakterek alapján számoljon, hanem a szavak második karakterét vegye figyelembe!

Hasonlítsuk össze az így nyert grafikont a jelenlegivel!

2. Adatforrások használata

Az alábbiakban megismerkedünk a Sources Node-ok közül talán a három legáltalánosabban használható node-dal.

Var. File node 

Ezzel a node-dal már találkoztunk az előző gyakorlaton is, amikor az angol-magyar szótárt olvastuk be vele.

A node-dal tagolt szövegfájlokból olvashatunk be adattáblákat. A szövegfájlokon belül a legáltalánosabban használt tagolókarakterek:

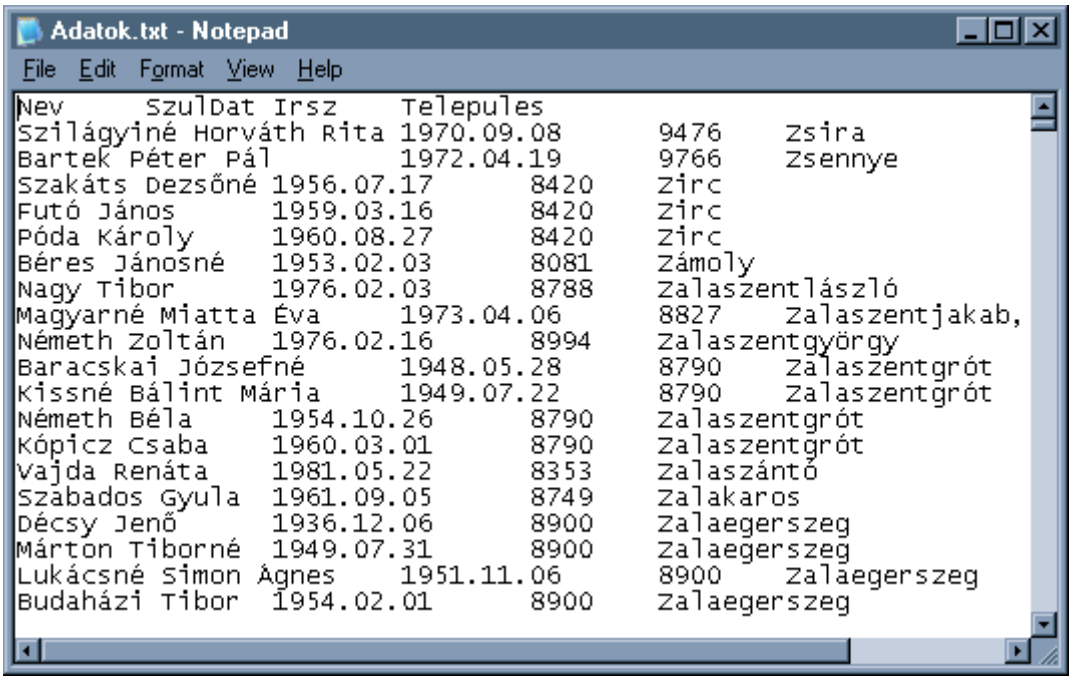
- Vessző
- Pontos vessző
- Tab
- Szóköz

Ezeken túl minden esetben a beolvasandó táblázat sorait a sorvége jel határolja a szövegfájlon belül.

Tipikus fájltípus a txt-n kívül még a csv (Coma Separated Value – Vesszővel Elválasztott Értékek).

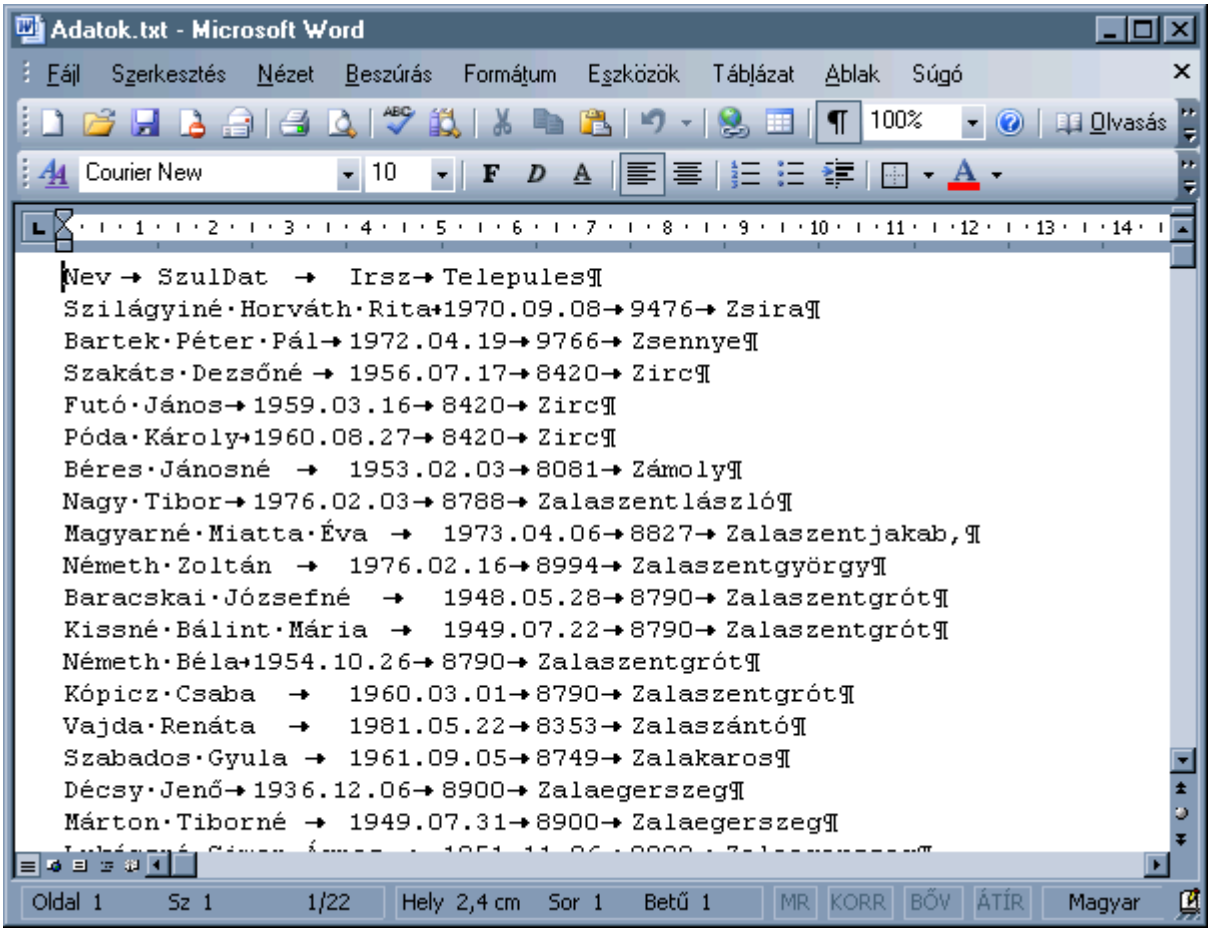
Keresse meg és nyissa meg a jegyzettömbben az ADATOK.TXT fájlt!

Hasonló képet kell látnunk:



Sejthető, hogy adataink három oszlopba vannak tagolva. De vajon mi a határoló karakter az adatok között? Ha a nyíl billentyűkkel mozgunk a sorokon belül, akkor nyilvánvaló,

hogy az oszlopok közötti távolságok nem szóköz karakterekkel vannak kitöltve. Sejthető már, hogy a határolókarakter az értékek között a **TAB**. Hogyan ellenőrizhetnénk hipotézisünk? Nyissuk meg a fájlt egy olyan szövegszerkesztő alkalmazásban, amely képes a láthatatlan vezérlőkarakterek megjelenítésére, például a Microsoft Word-ben:



A nyomtatásban láthatatlan karakterek megjelenítésének bekapcsolásával egyértelműen látszik, hogy valóban a TAB a határoló karakter.

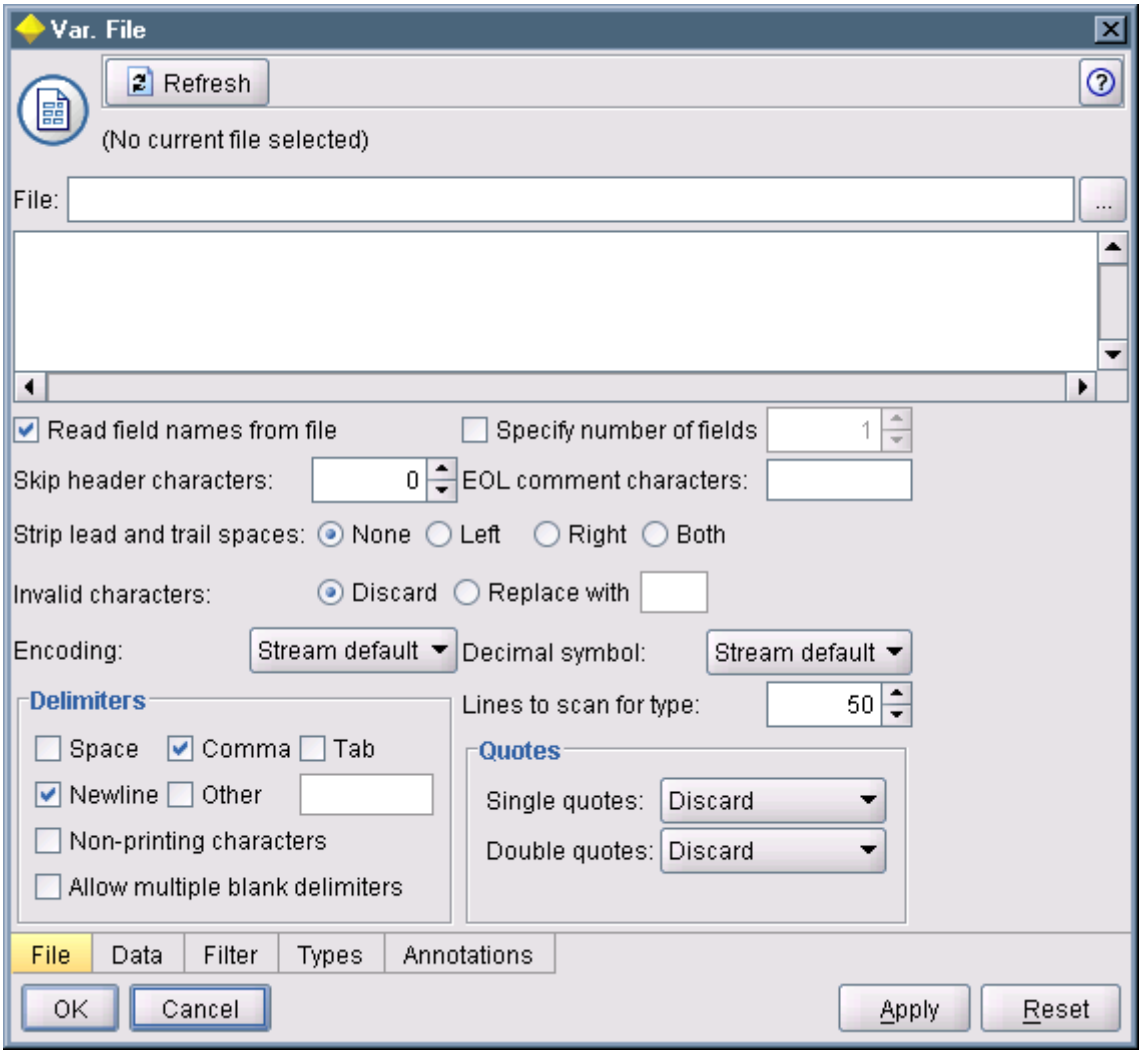
Olvassuk be a Modeler-ben az adatainkat:

A Modeler-en belül nyisson egy új stream-et!

Helyezzen el a stream-ben egy VAR. FILE **node-ot!**

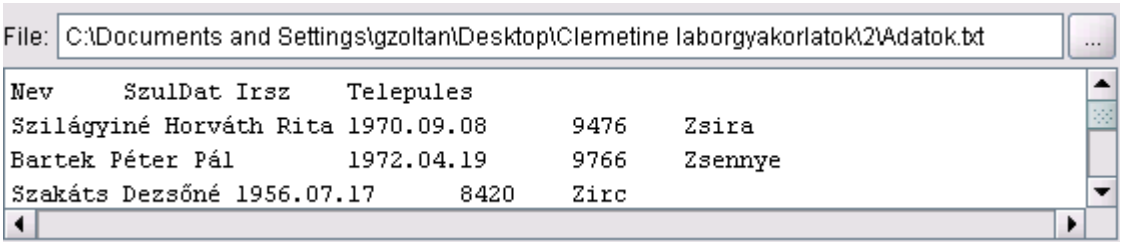
Nyissa meg a node beállításait!

Ha eddig eljutottunk, a következő ablakot láthatjuk a képernyőn:



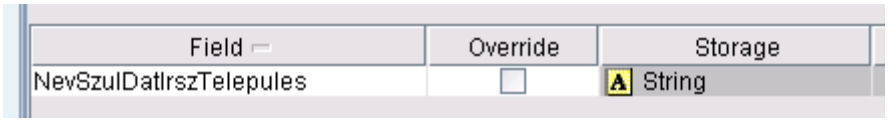
Az ablak tetején látható FILE rovatba tallózzuk be az ADATOK.TXT állományt!

A rovat alatt látható panelban meg is jelenik a fájl tartalma:



A Read field names from file kapcsoló alapállapotát őrizzük meg, hiszen ebben az esetben a szövegfájl első sora a beolvasandó táblázat oszlopainak a neveit tartalmazza.

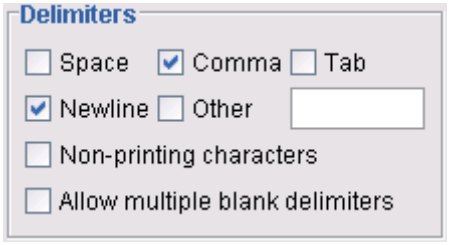
Váltsunk át az ablak alján a DATA fülre, hogy ellenőrizzük a beolvasott adattábla oszlopait!



Láthatóan valami még nem tökéletes. A három oszlop helyett egy oszlopként látja a tartalmat. Sejthető, hogy az oszlopok értékei közötti határoló karakter megadásánál van a probléma.

Váltsunk vissza a FILE fülre!

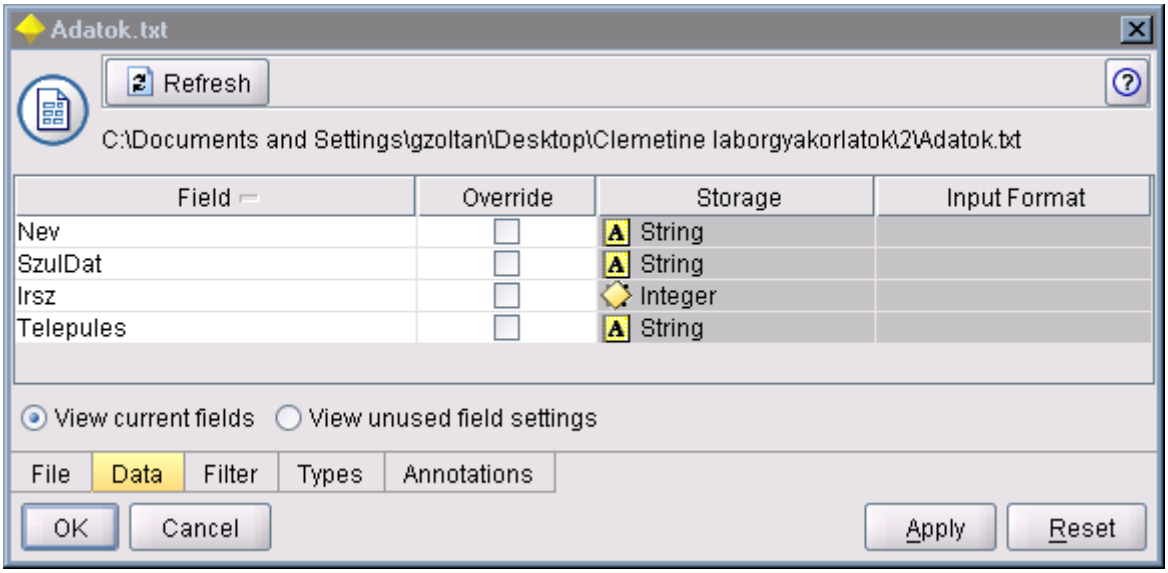
Meg is van a probléma oka:



Amint az látható, Tab helyett a Coma (vessző) van megadva határolónak.

Jelöljük be a COMA helyett a TAB-ot!

Váltsunk vissza a DATA fülre, hogy ellenőrizzük a módosítás hatását!

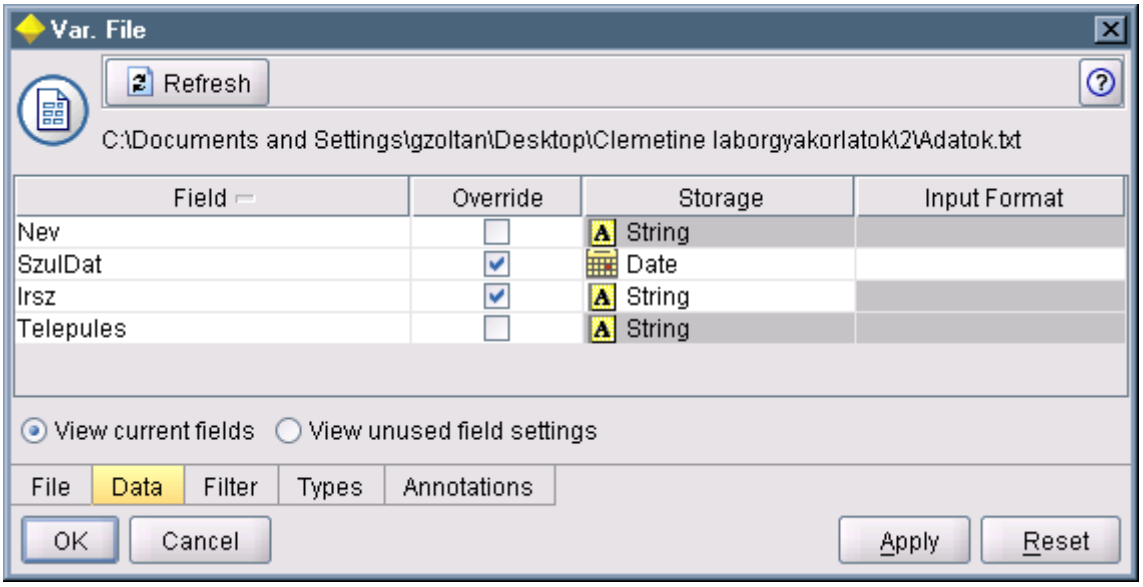


Kezdenek a dolgok a helyükre kerülni! Bár még van egy apró probléma.

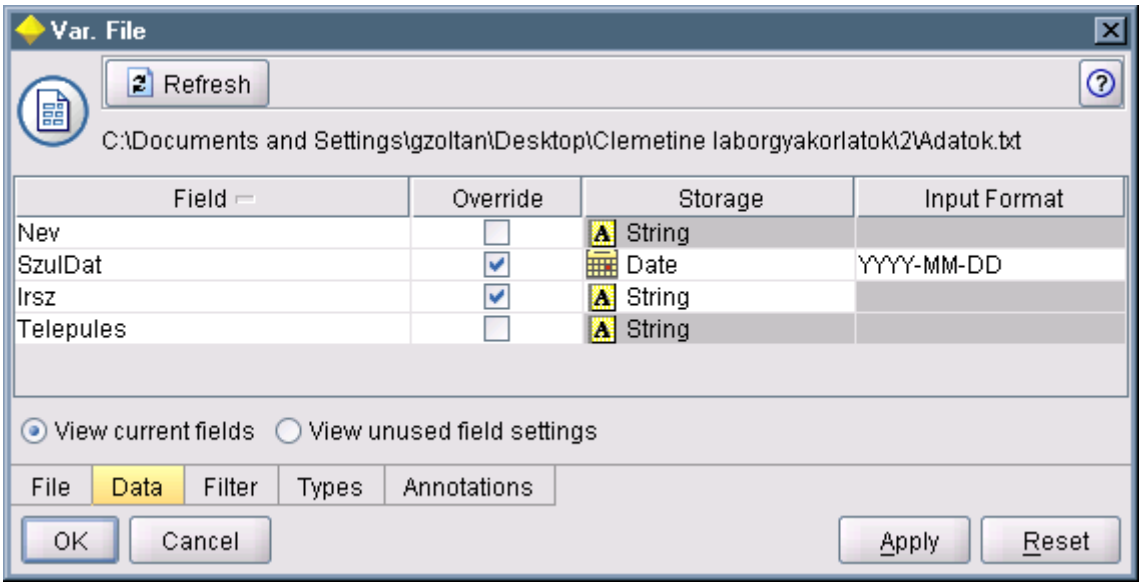
A **DATA** fülön figyelemmel kísérhetjük a beolvasandó tábla oszlopaiban lévő értékek adattípusait. Amint az a képen is látható a Modeler a Nev, a Szuldat és a Telepules oszlopokat szöveges adatként tárolná, míg az Irsz mező értékét egész számként. Probléma a születési dátum szövegként tárolásával, és az irányítószám egész számként történő tárolásával van.

Állítsuk át a SzulDat és az Irsz mezők adattípusát dátumra és szövegre.

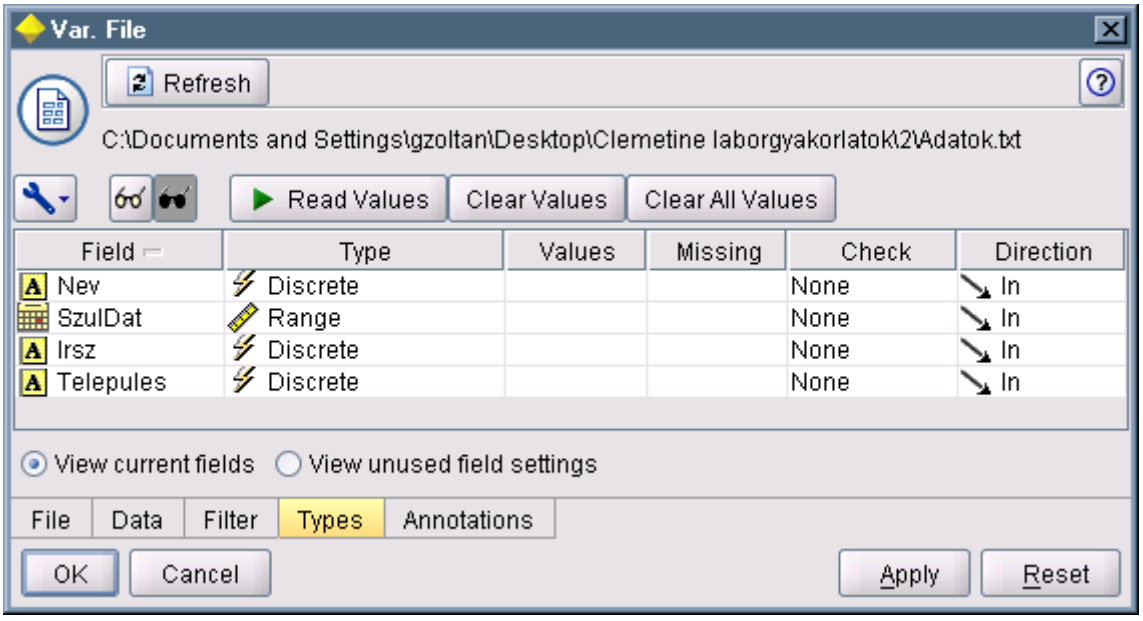
Az Override jelölőnégyzet bepipálásával jelezhetjük, hogy az adott oszlop adattípusát felülbíráljuk:



Az **INPUT FORMAT** oszlopban szükség szerint gondoskodhatunk a beolvasott értékek megjelenítésének módjáról is:



Váltson át a TYPES fülre!



A **FIELD** oszlop mutatja a beolvasandó mezők nevét és tárolási adattípusát. A **TYPE** oszlop pedig a statisztikai változó típusát. A következő értékeket állíthatjuk itt be:

- Range:

  Egész vagy valós számértékeknél, valamint dátumoknál használható.

- Discrete:

  Szöveges típusú értékek esetén használatos, amikor a pontos száma a lehetséges különböző értékeknek nem ismeretes. Igazából egy absztrakt adattípus, ami azt jelzi, hogy még nem ismerünk minden lehetséges információt a tárolt adatainkról. Ha egyszer beolvassuk az adatainkat átalakul Flag, Set, vagy Typeless értékké attól függően, hogy mekkora a maximálisan beállított Set Size (halmazméret) a stream tulajdonságlapján.

- Flag:

  Két különböző értéket tartalmazó változók esetén használjuk. Lehetnek ezek szövegek, számok, de akár dátumok is. (pl.: (1;2), vagy (férfi;nő), stb.)

- Set:

  Több különböző érték leírására szolgál, amelyek mind egy adott halmaz elemei közül kerülnek ki. Például: (kicsi; közepes; nagy).

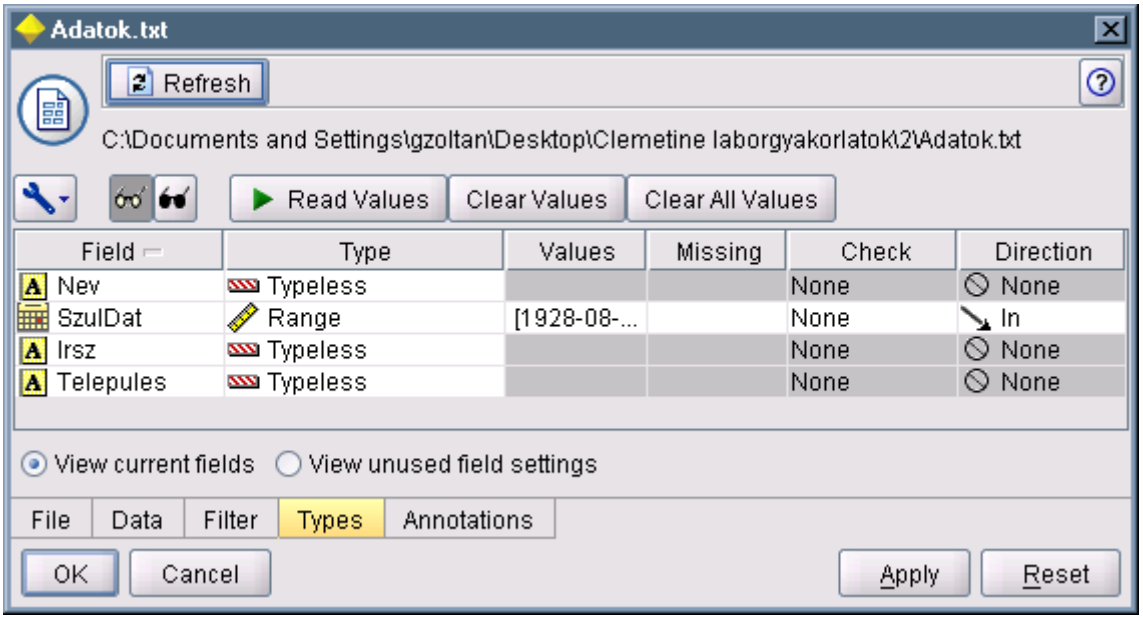
- Ordered Set:

  Több különböző érték leírására szolgál, amelyek mind egy adott halmaz elemei közül kerülnek ki, és van értelme az adatok sorbarendezésének. Például: érdemjegyek.

- Typeless:

  Minden egyéb esetben használható, vagy ha a Set típus esetén a halmaz túl sok elemet tartalmazna.

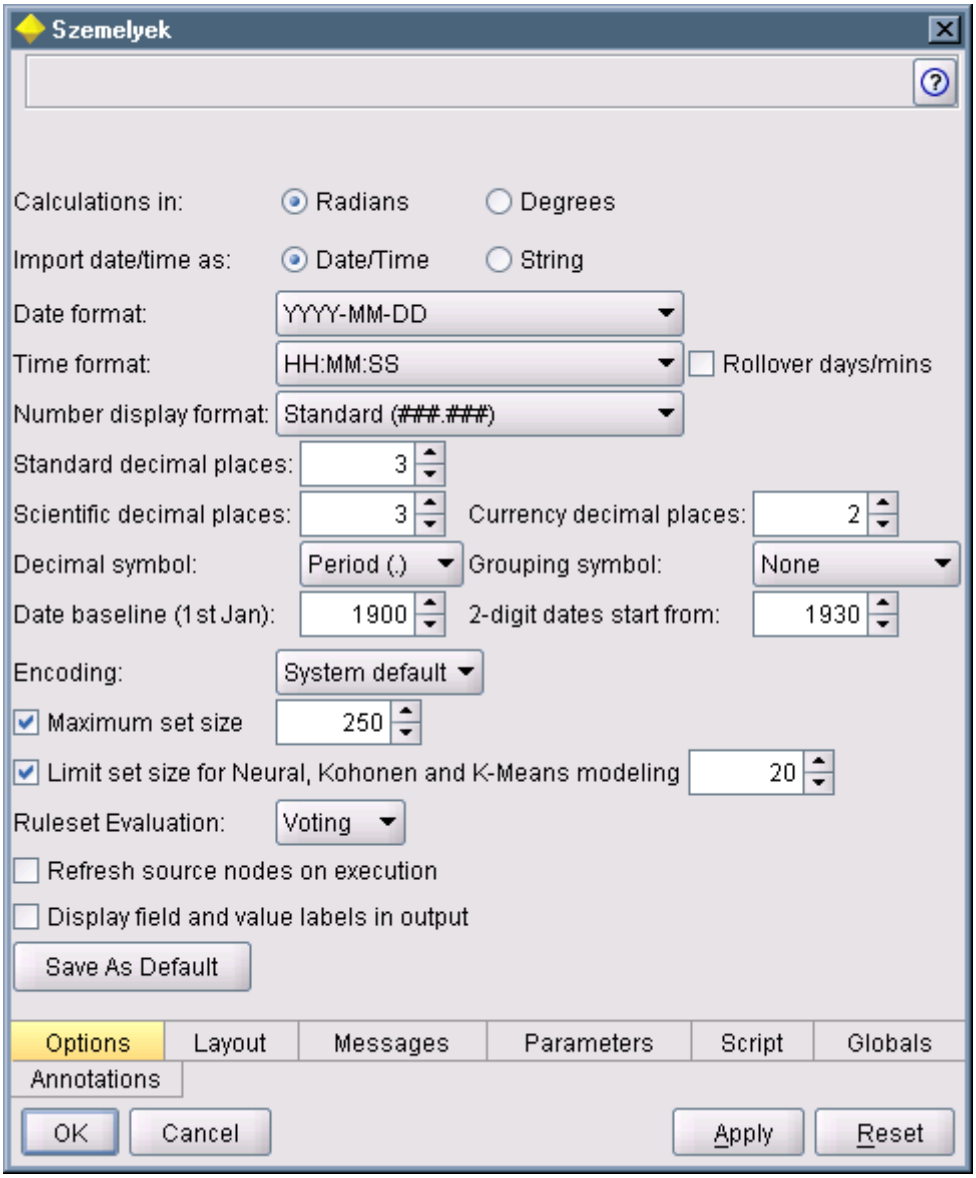
Kattintson a READ VALUES gombra, az adatok beolvasásához és az adattípusok beállításához.



Az eredményen látható, hogy nem tudta halmaz típusúként kezelni sem a nevet, sem az irányítószámot, sem a települést. Mint már olvastuk, ez függhet a stream tulajdonságlapján beállított maximális halmazmérettől.

Az OK –ra kattintva zárja be az ablakot!

A FILE menüből válassza ki a STREAM PROPERTIES... parancsot!



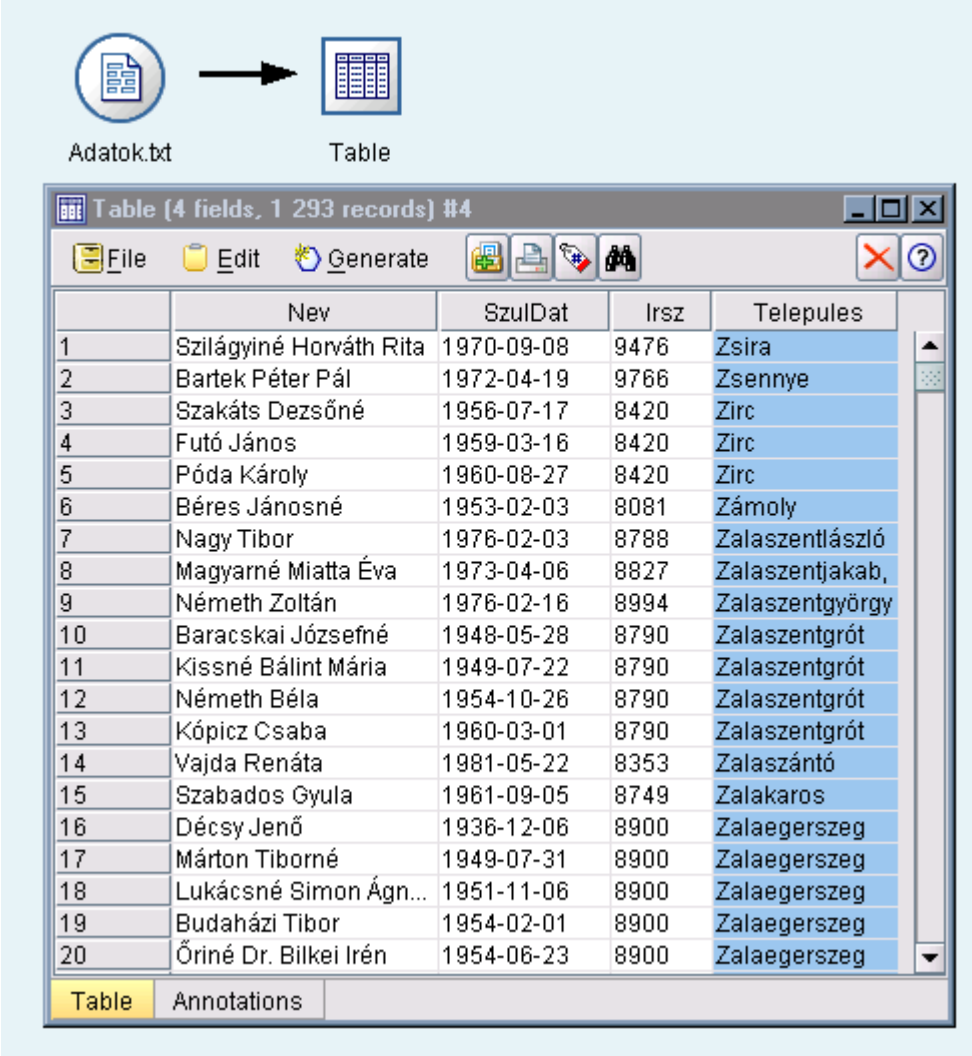
Mint látható a **MAXIMUM SET SIZE** értéke alapból 250-re van állítva. Ezt az értéket szükség esetén növelhetjük, vagy ki is kapcsolhatjuk a maximumot a beállítás előtti jelölőnégyzet kiürítésével.

Ebben az esetben most tekintsünk el a módosítástól, mert amennyiben nem akarjuk adataink például településnév szerinti csoportosítását a későbbiekben kihasználni, nincs rá szükségünk, hogy a Clemetine Set típusúként kezelje az oszlop értékeit.

Zárja be az ablakot a CANCEL-re kattintva!

Teszteljük le a node-unk működését.

Kapcsoljon a node után egy Table output node-ot, és futtassa!



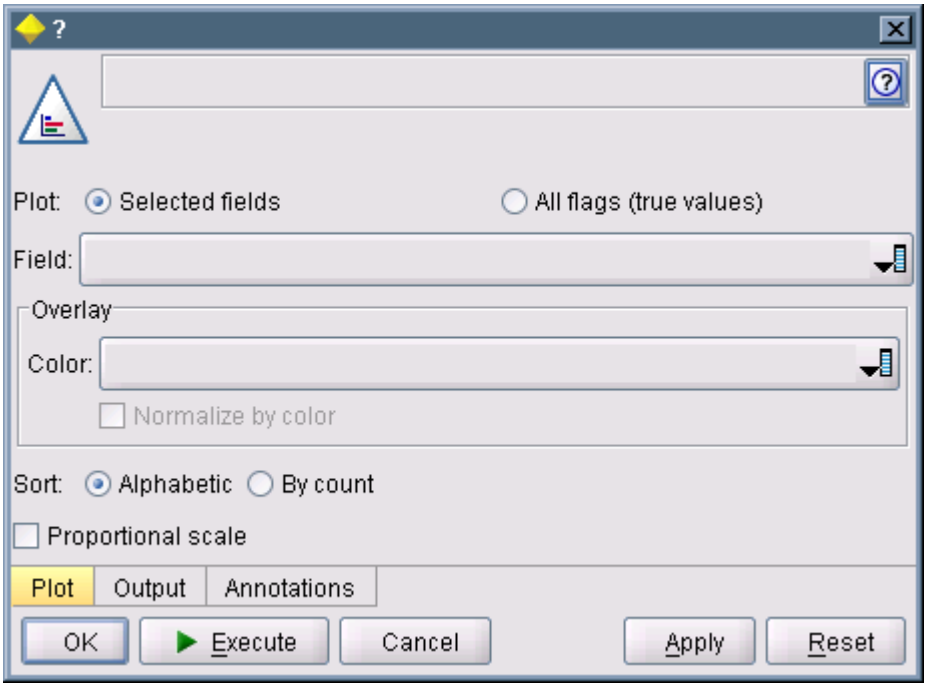
	Nev	SzulDat	Irsz	Telepules
1	Szilágyiné Horváth Rita	1970-09-08	9476	Zsira
2	Bartek Péter Pál	1972-04-19	9766	Zsennye
3	Szakáts Dezsőné	1956-07-17	8420	Zirc
4	Futó János	1959-03-16	8420	Zirc
5	Póda Károly	1960-08-27	8420	Zirc
6	Béres Jánosné	1953-02-03	8081	Zámoly
7	Nagy Tibor	1976-02-03	8788	Zalaszentlászló
8	Magyarné Miatta Éva	1973-04-06	8827	Zalaszentjakab,
9	Németh Zoltán	1976-02-16	8994	Zalaszentgyörgy
10	Baracskai Józsefné	1948-05-28	8790	Zalaszentgrót
11	Kissné Bálint Mária	1949-07-22	8790	Zalaszentgrót
12	Németh Béla	1954-10-26	8790	Zalaszentgrót
13	Kópicz Csaba	1960-03-01	8790	Zalaszentgrót
14	Vajda Renáta	1981-05-22	8353	Zalaszántó
15	Szabados Gyula	1961-09-05	8749	Zalakaros
16	Décsy Jenő	1936-12-06	8900	Zalaegerszeg
17	Márton Tiborné	1949-07-31	8900	Zalaegerszeg
18	Lukácsné Simon Ágn...	1951-11-06	8900	Zalaegerszeg
19	Budaházi Tibor	1954-02-01	8900	Zalaegerszeg
20	Őriné Dr. Bilkei Irén	1954-06-23	8900	Zalaegerszeg

Amint látható eddigi munkánk gyümölcse beérett.

Próbáljuk meg kinyerni az egyes településekre nézve az adataink eloszlását.

Kapcsoljon a source node után egy DISTRIBUTION node-ot a GRAPHS palettáról!

Nyissa meg a node beállításait!



Ha megpróbáljuk kiválasztani a Field rovatban, hogy mely oszlop értékei alapján szeretnénk az eloszlást megkapni, akkor sajnálatos módon üres listával találkozunk.

Mi is lehet ennek az oka? Ha kitalálta, nem is kell tovább olvasni! ☺

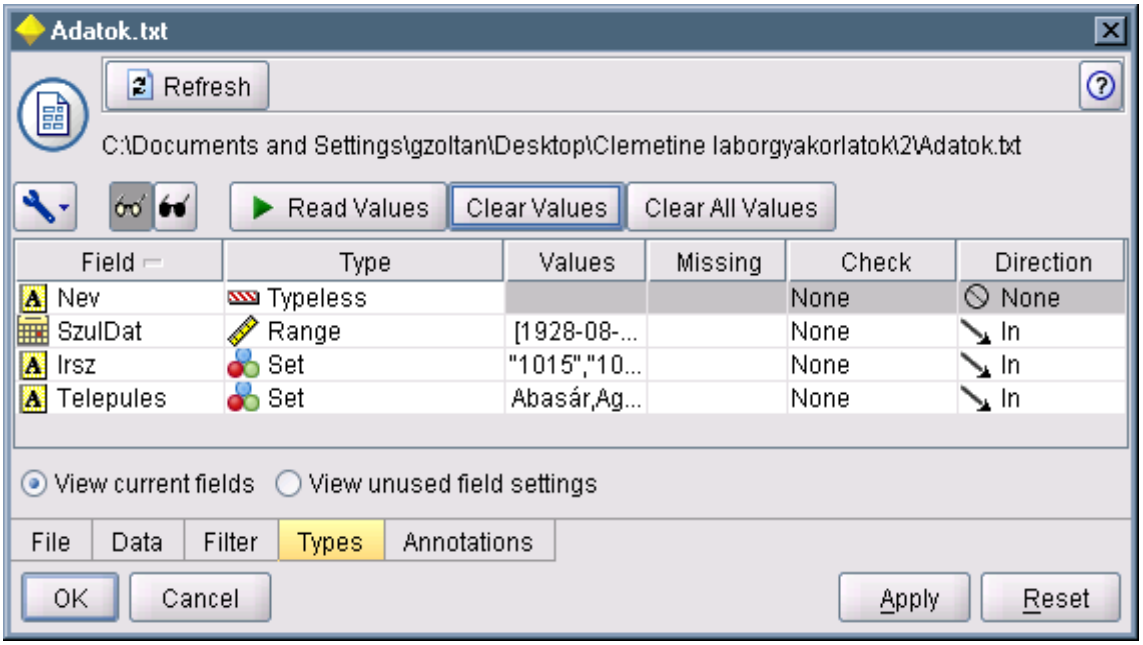
Az eloszlás alapja csakis set, és flag típusú mező lehet!

Nyissa meg a STREAM PROPERTIES ablakot és kapcsolja ki a MAXIMUM SET SIZE opciót!

Nyissa meg ismét a source node beállításait, azon belül is a TYPES Fület.

A típusok ismételt kiértékeléséhez újra kell olvastatni az adatokat.

Kattintson a CLEAR ALL VALUES gombra, majd utána a READ VALUES gombra!

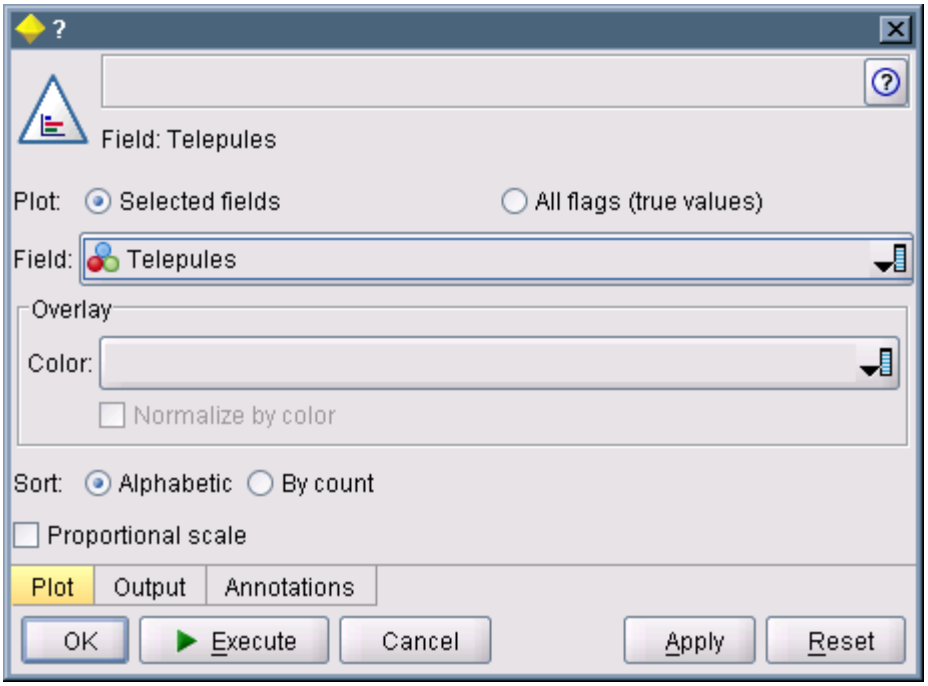


A kapott eredményt látva az a kérdés adódik, hogy a Nev mező miért maradt Typeless?

Ha ismét megnézzük a Stream Properties ablakot,

akkor azt látjuk, hogy még egy beállítás van, amely korlátozza a halmazok méretét. Részletesebb információkért nyissa meg a súgót!

Térjünk vissza a Distribution node-hoz. Nyissa meg a node beállításait!



Amint az látható, már semmi akadálya a node lefuttatásának.

Futtassa a node-ot!

Ha végiggörgeti a településlistát nagyon szembetűnő, hogy mennyire sok hibát is tartalmaz az adatállomány. Ezeken a hibákon a továbbiakban nagyvonalúan átsiklunk.

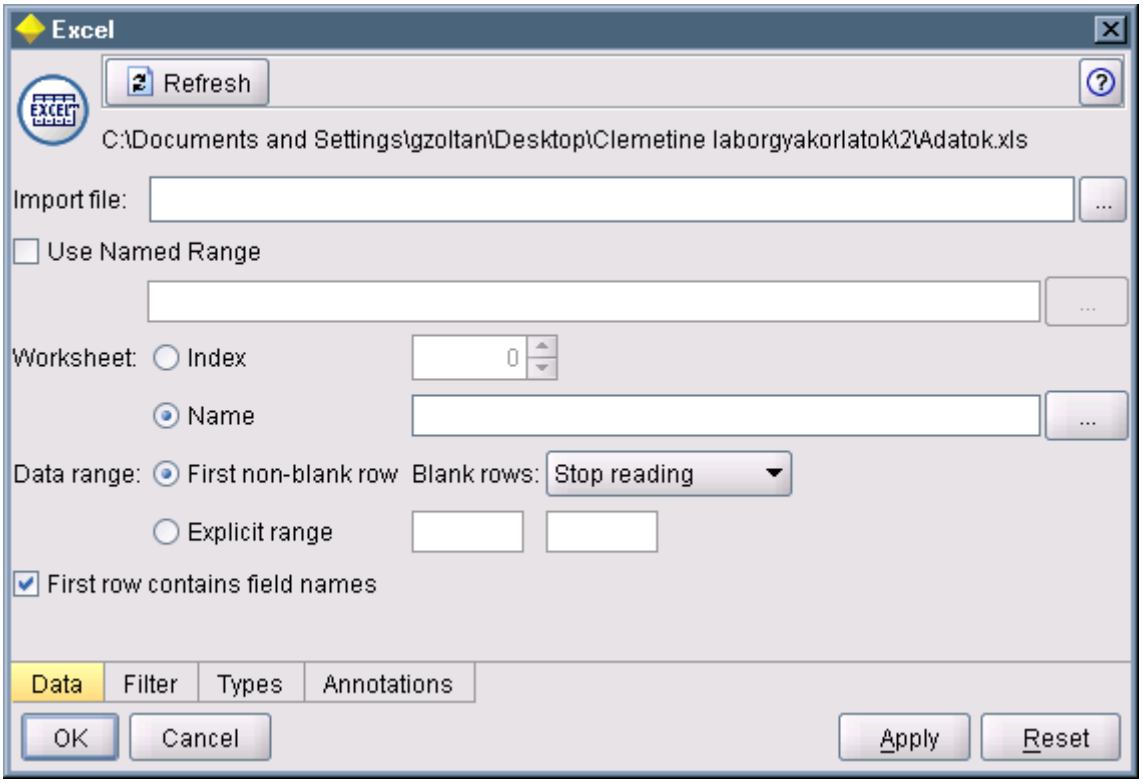
Bár már ismerjük a szerepét, de meg kell említenünk a source node beállításai között megtalálható Filter fület is, ahol átnevezhetjük a beolvasott oszlopokat, valamint kikapcsolhatjuk azokat, amelyekre nem lesz szükségünk.

A következőkben ugyanezt az adatállományt olvassuk be, két további formátumból, két további source node segítségével.

Excel node 

Helyezzen el a stream-ben egy EXCEL node-ot is a SOURCES palettáról!

Nyissa meg a node beállításait tartalmazó ablakot!



A **FIXED FILE** node-hoz képest eltérést csak a **DATA** fül beállításainál tapasztalhatunk (file fül nincs is). Ez logikus is, hiszen az adatforrás fizikai jellemzőin túl már ugyanúgy kell, hogy állíthassuk a típusokat, és a mezők átnevezését, elrejtését.

Az IMPORT FILE rovatban válassza ki az ADATOK.XLS fájlt!

Meg kell adnunk továbbá, hogy az Excel munkafüzet mely munkalapja tartalmazza a beolvasandó adattáblát. Ezt megadhatjuk a munkalap sorszámával **(INDEX)**, vagy a munkalap nevének kiválasztásával is **(NAME)**.

Ha megnyitjuk az Excelben a munkafüzetet, hogy a felépítését megnézzük, ne felejtsük el bezárni, mielőtt a node-ot futtatnánk. Ha meg van nyitva Excelben, akkor az olyan módon zárolja a fájlt, hogy a Modeler nem tud hozzáférni, még olvasásra sem.

A WORKSHEET rovatban jelölje ki a NAME opciót, és tallózza be a munkafüzet SZEMELYEK nevű munkalapját!

A munkalap kiválasztása után még meg kell adni, hogy a munkalap mely tartománya tartalmazza az adatokat.

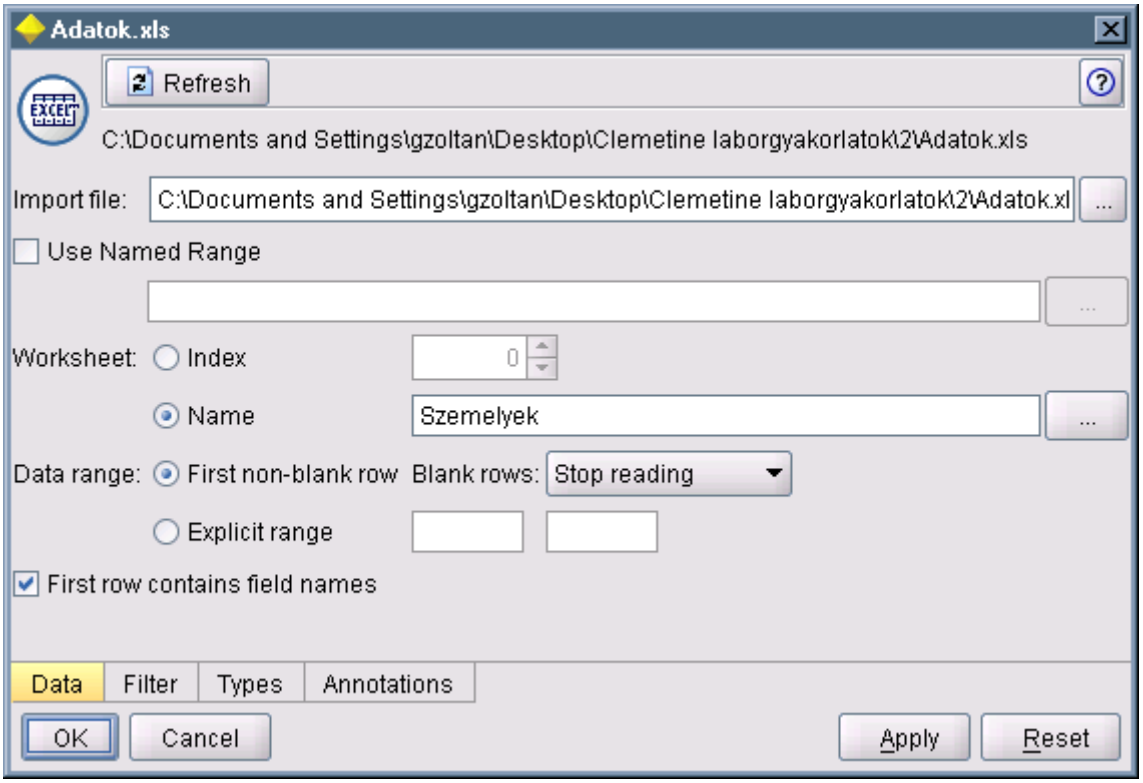
Ha a **DATA RANGE** rovatban a **FIRST NON-BLANK ROW** –ot választjuk, azzal azt jelezzük, hogy adattáblánk a munkalap első nem üres sorában kezdődik. Megadhatjuk ekkor, hogy mi történjen.

Ha további üres sorokat találunk a beolvasáskor:

- **STOP READING:** Befejezi a beolvasást.
- **RETURN BLANK ROWS:** A munkalap összes adatát beolvassa az üres sorokkal együtt.

Ha a **DATA RANGE** rovatban az **EXPLICIT RANGE** opciót választjuk, akkor pontosan megadhatjuk a beolvasandó tartomány méretét, elhelyezkedését. Például: B3:F50.

A beállítások elvégzése után hasonló képet kell látnunk:



Ok –ozza le a node beállításait!

Kapcsoljon egy TABLE output node-ot az EXCEL node után és futtassa az eredmény megtekintéséhez.

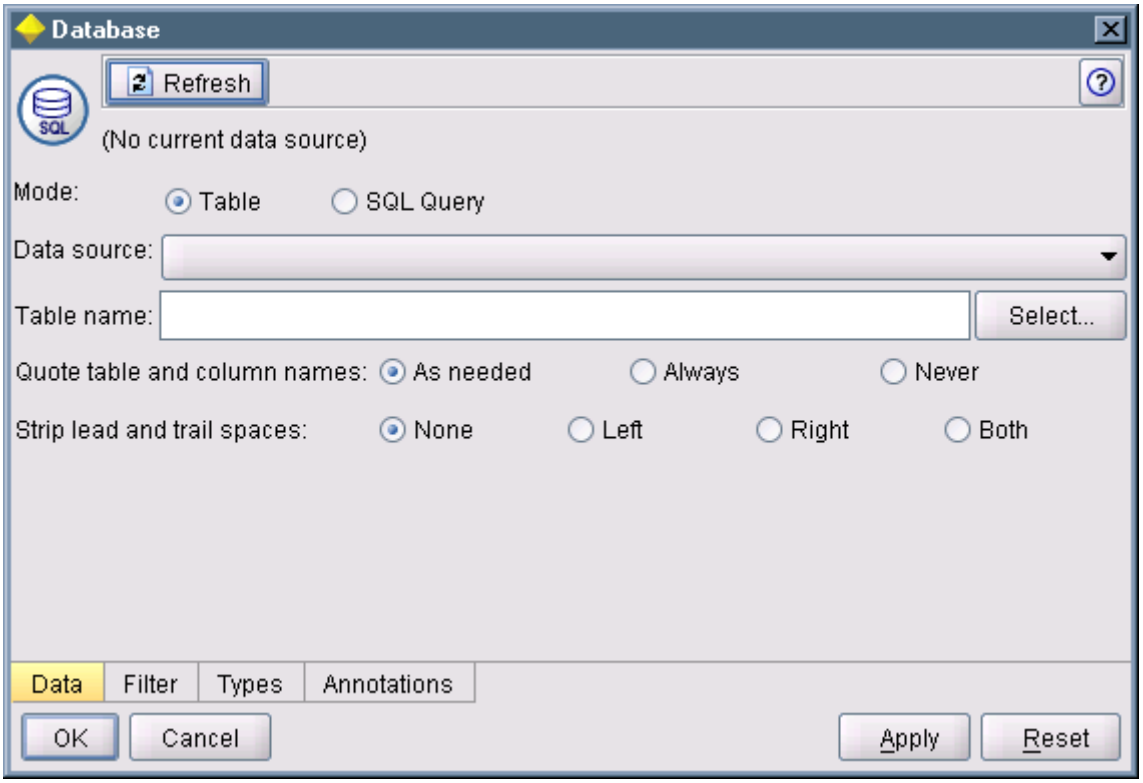
Megfigyelhetjük, hogy ezen node használata esetén nem adhatjuk meg a beolvasott adatok fizikai adattípusát (szám, szöveg, stb.), mert azt az Excel munkafüzetben történő tárolás módja automatikusan meghatározza.

Viszont itt is meg kell adnunk az adatok statisztikai értelemben vett típusát. (Set, Range, stb.)

Database node

Helyezzen el a stream-ben egy DATABASE node-ot is, a SOURCES palettáról!

Nyissa meg a node beállításait!



Ezzel a node-dal relációs adatbázisokban tárolt táblák adatait olvashatjuk be a stream-be. Az **EXCEL** node-hoz képest csak a **DATA** fül beállításaiban találunk eltérést.

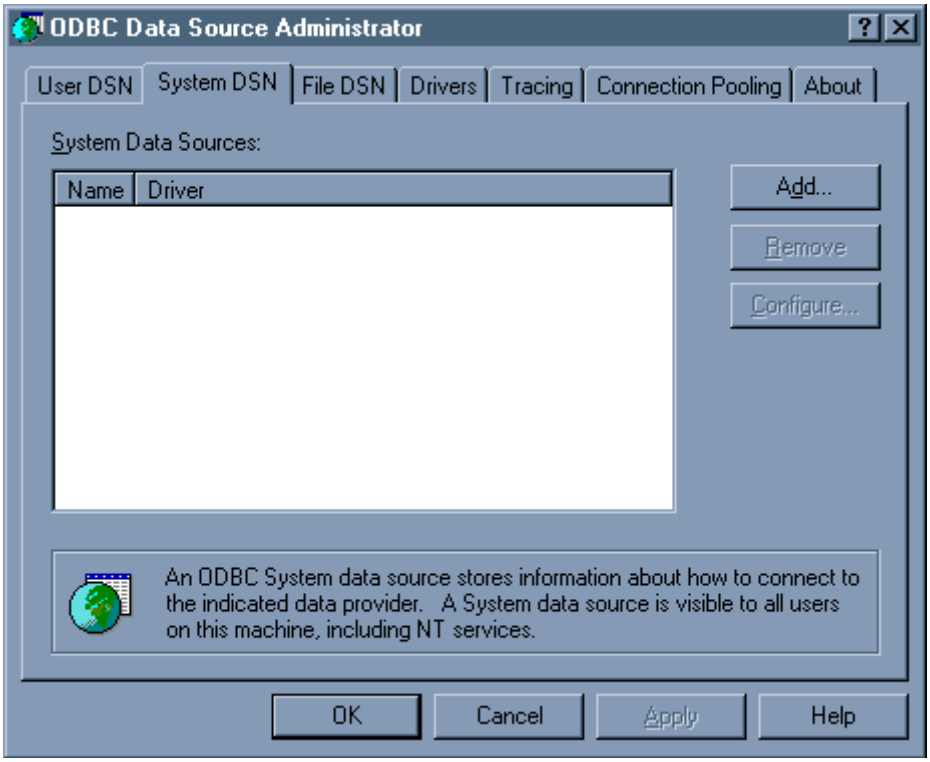
A **DATA SOURCE** rovatban egy már a Windowsban bekonfigurált adatforrást lehet kiválasztani. Tehát ahhoz, hogy a Modeler rá tudjon csatlakozni egy adatbázisra, előtte a Windowsban létre kell hozni egy ODBC bejegyzést.

ODBC = Open DataBase Connectivity (nyílt adatbázis kapcsolat)

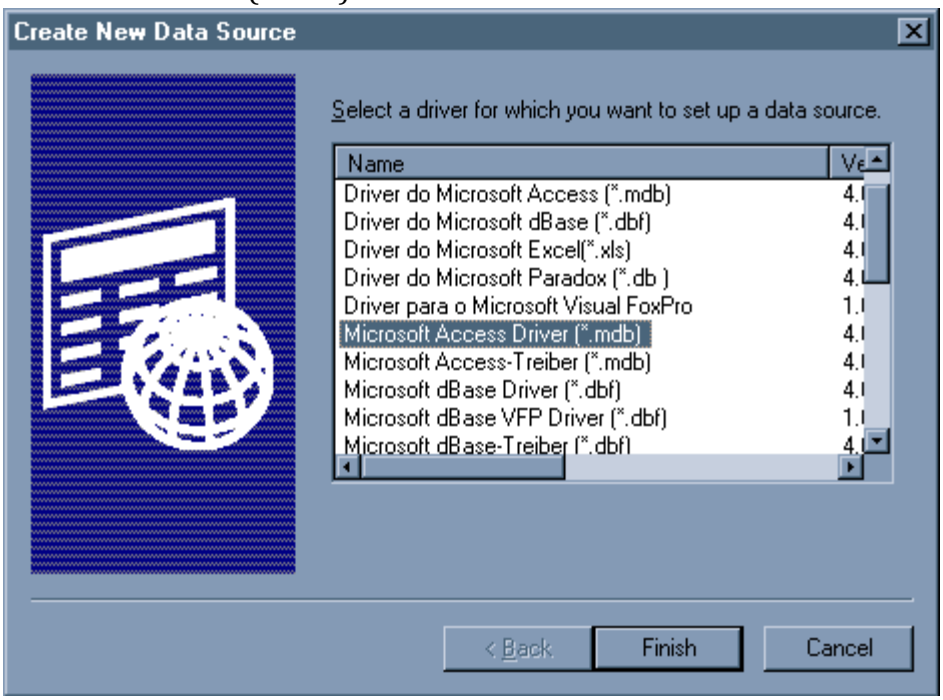
Az ODBC bejegyzés létrehozásához rendszergazdai jog szükséges.

A következő lépéseket kell tenni (Windows XP esetén):

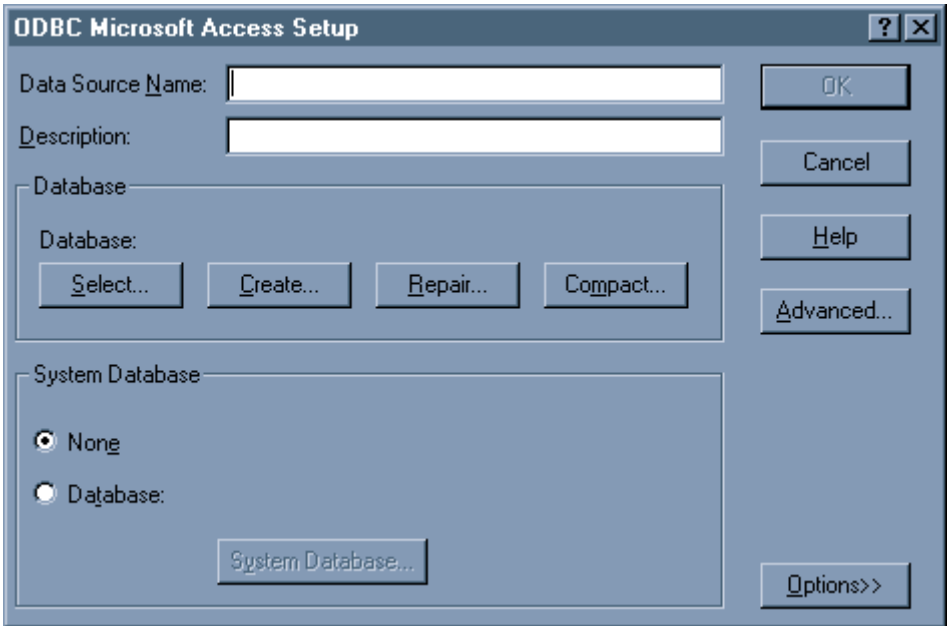
- Nyissuk meg a vezérlőpulton (Control Panel) belül a felügyeleti eszközöket (Administrative Tools).
- Nyissuk meg az Adatforrások (Data Dources) (ODBC) ikont, és váltsunk a Rendszer DSN-re (System Dsn):



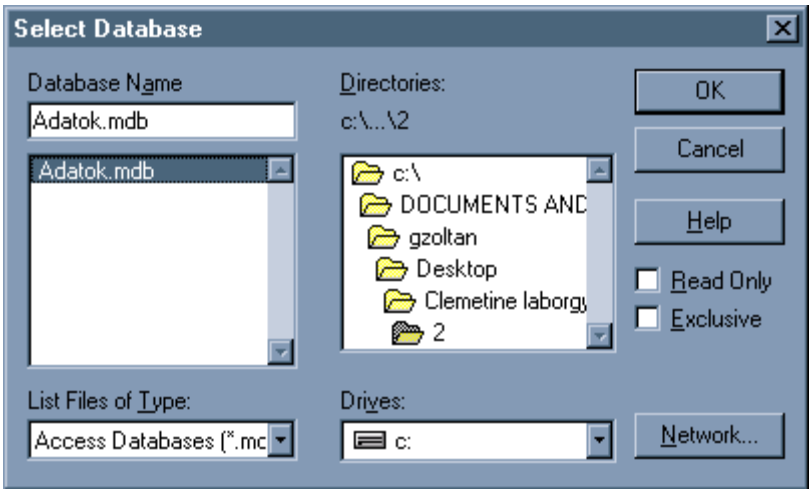
- Hozzáadás (Add...):



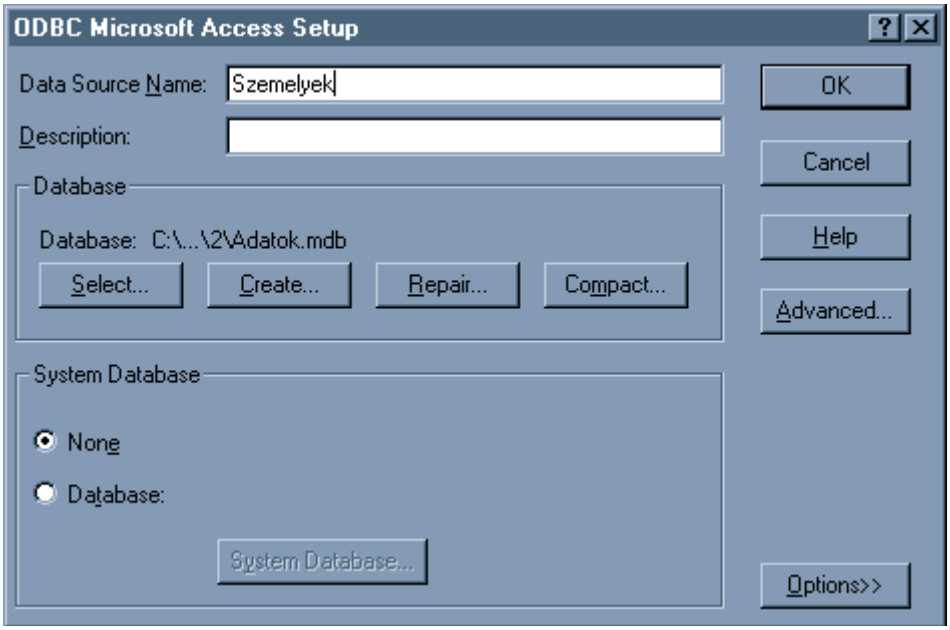
- Jelöljük ki a Microsoft Access Driver (*.mdb) listaelemet, és kattintsunk a Befejezés –re (Finish):



- Írjuk be a létrehozandó ODBC bejegyzés nevét, az ablak tetején lévő beviteli mezőbe: Szemelyek
- A Kijelöl… (Select) gombra kattintva jelölje ki a beolvasni kívánt adatbázist. A mi esetünkben az Adatok.mdb-t:

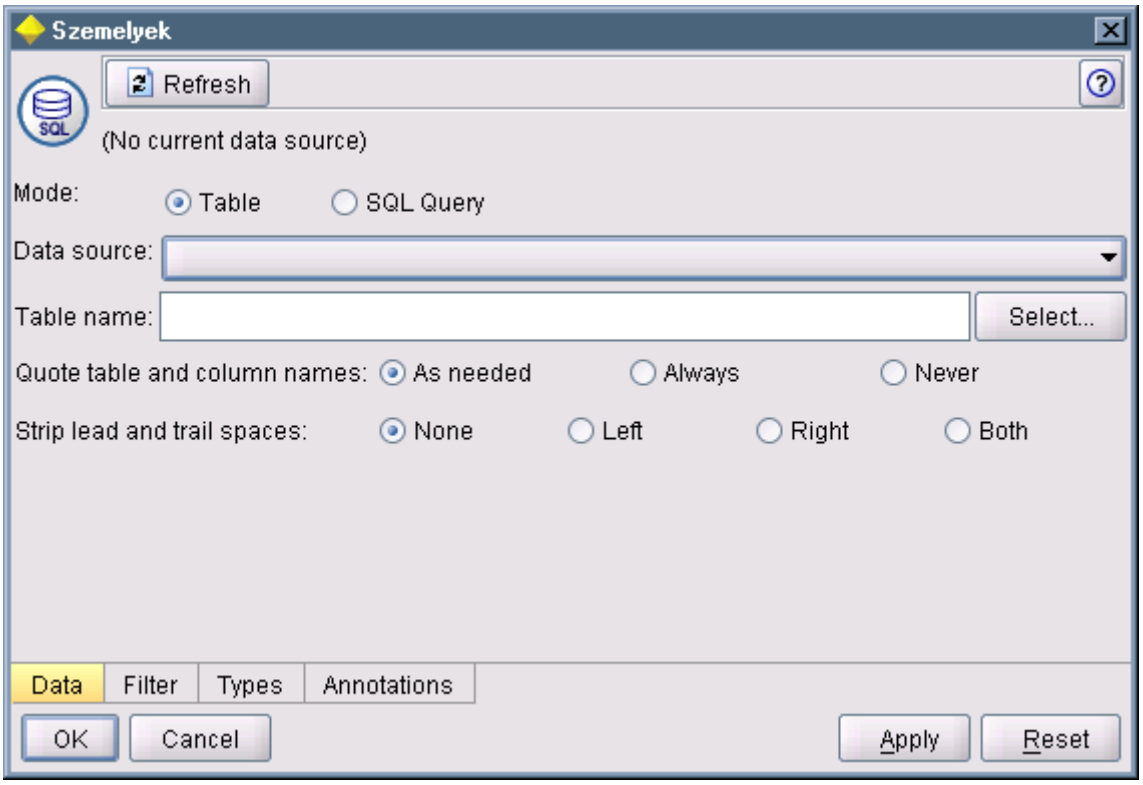


- Kattintsunk az OK-ra:

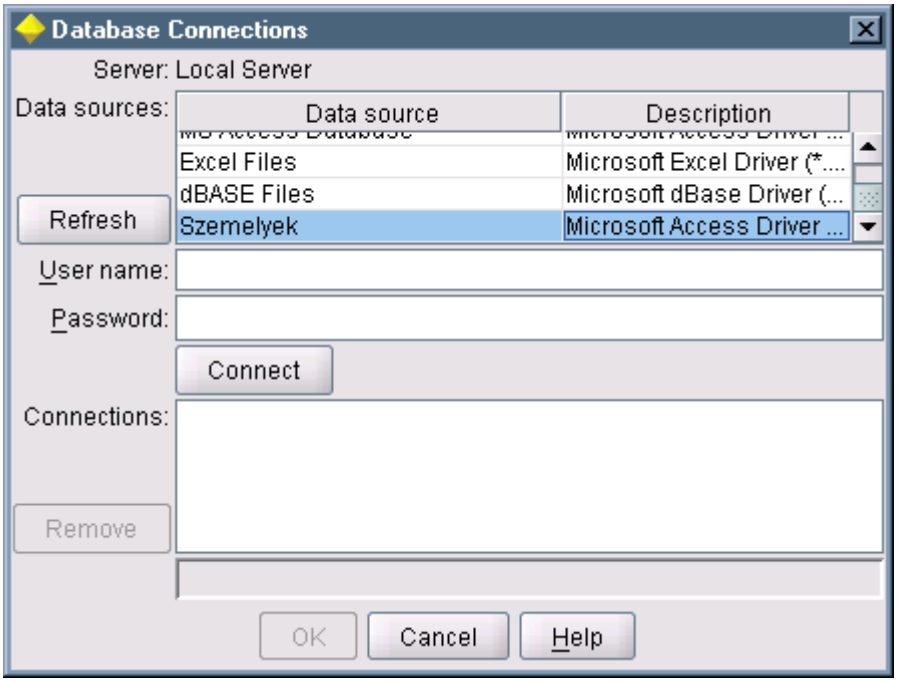


- OK és még egyszer OK az ablakok bezárásához.

Ezzel létre is hoztuk az ODBC bejegyzést. Térjünk vissza a Modeler-hez:

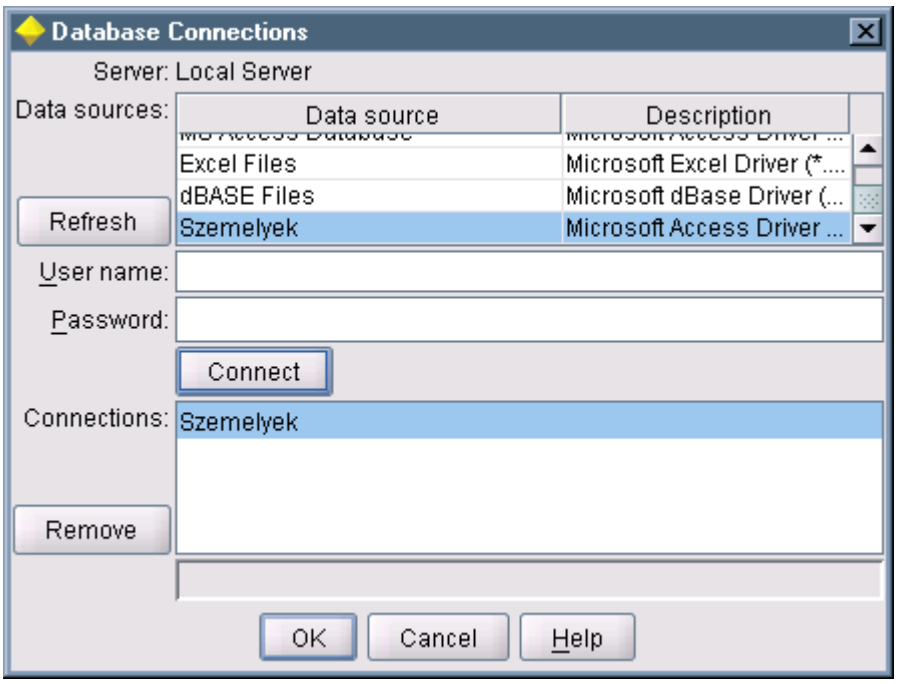


A DATA SOURCE **rovatból válassza az** <ADD NEW DATABASE CONNECTION...> **opciót!**

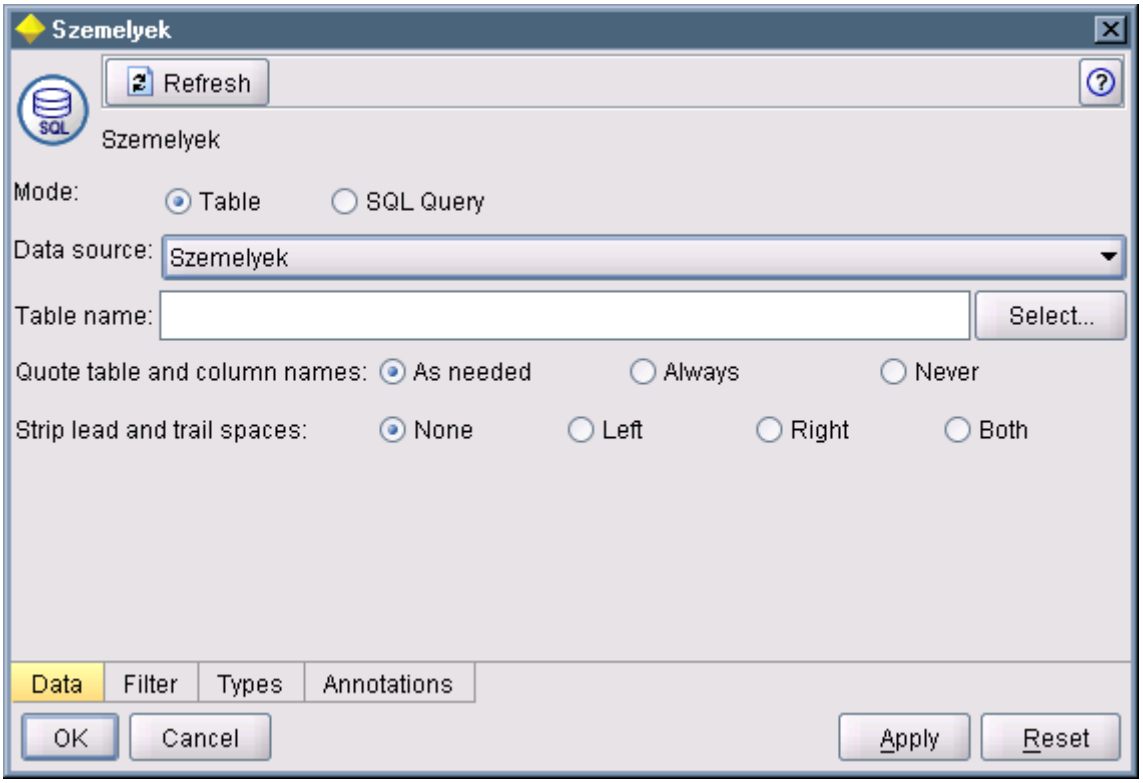


Az ablak tetején látható listában meg kell jelenjen a **SZEMELYEK** bejegyzés is!

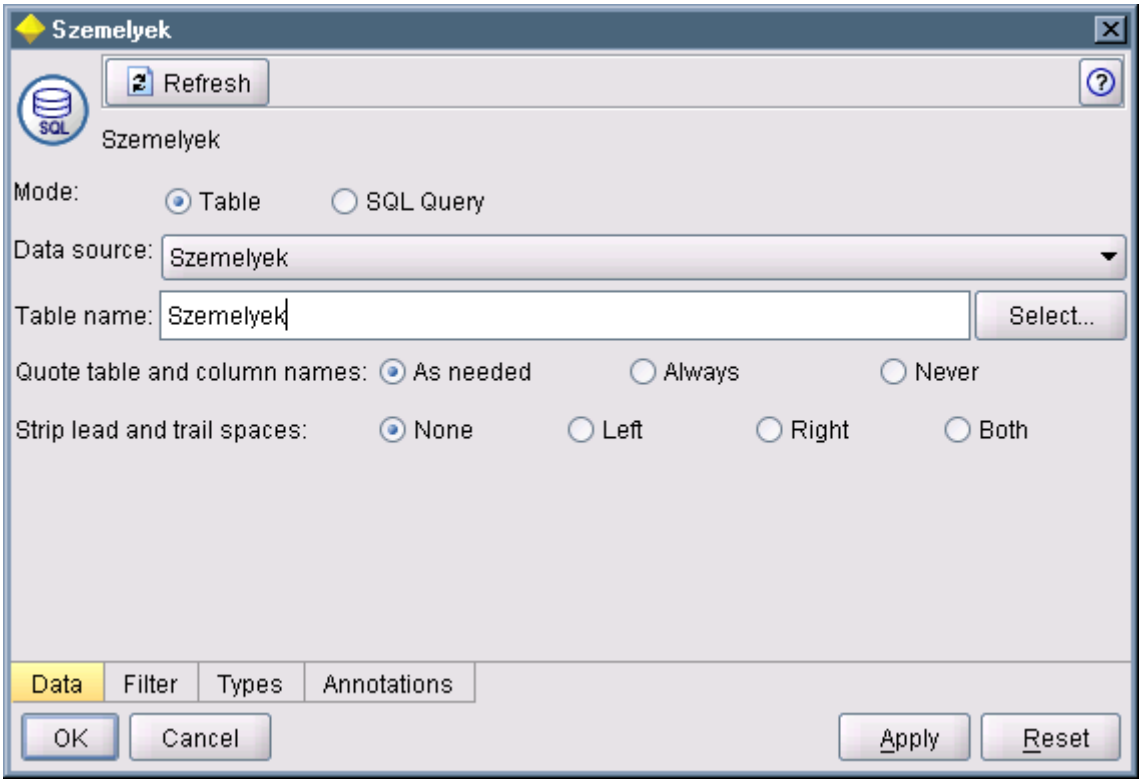
Jelölje ki a SZEMELYEK **bejegyzést a listából, és kattintson a** CONNECT **gombra a kapcsolódáshoz!**



Zárja be az ablakot OK **–kal!**



A TABLE NAME rovatban meg kell adni az adatbázis beolvasandó táblájának a nevét (SZEMELYEK).



Ha átkattintunk a filter fülre, már látni kell az adattábla oszlopainak a nevét. A további beállítások ugyanúgy történnek, mint eddig.

3. Rekord műveletek

Az alábbi gyakorlatok végrehajtásához használjuk az előző gyakorlat adatállományát!

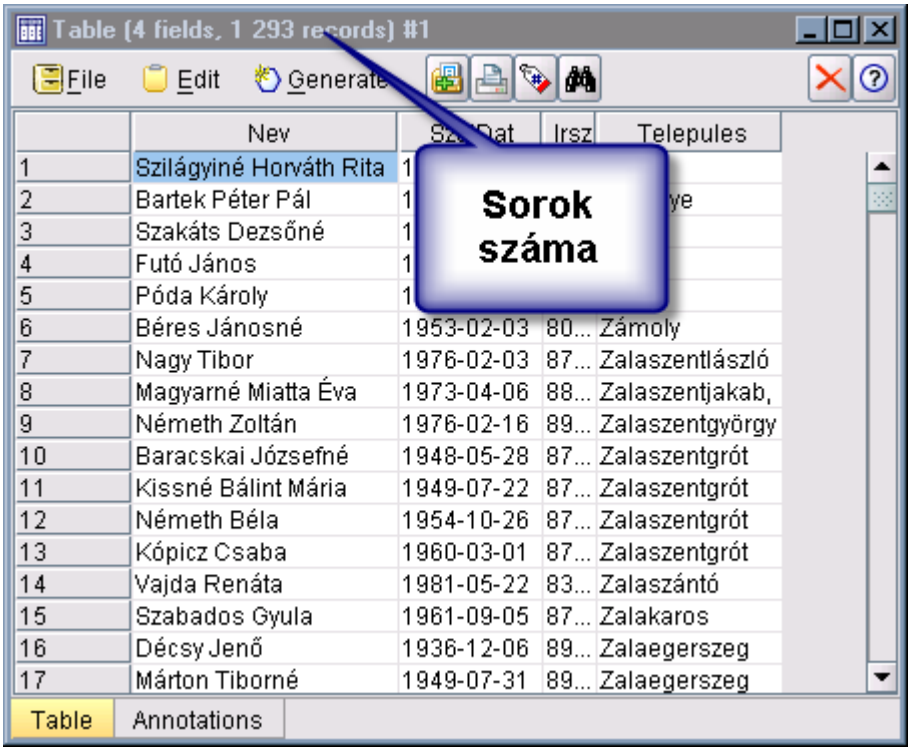
Az alábbiakban előforduló feladatok megoldásai megtalálhatók a Demo.str fájlban.

Adjon a stream-hez egy VAR. FILE node-ot, és olvassa be az ADATOK.TXT állomány tartalmát!

Ellenőrizze a beolvasás sikerességét, egy TABLE output node-dal!

Table node 

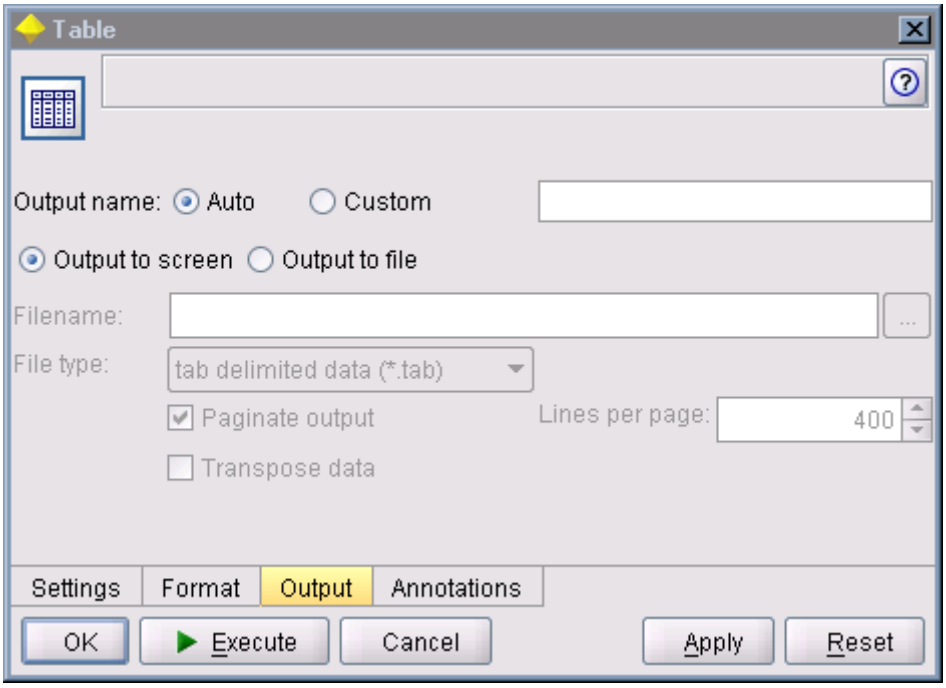
Ismerkedjünk kicsit jobban meg a Table node eredményablakával:



	Nev	Sz...Dat	Irsz	Telepules
1	Szilágyiné Horváth Rita	1		
2	Bartek Péter Pál	1		ye
3	Szakáts Dezsőné	1		
4	Futó János	1		
5	Póda Károly	1		
6	Béres Jánosné	1953-02-03	80...	Zámoly
7	Nagy Tibor	1976-02-03	87...	Zalaszentlászló
8	Magyarné Miatta Éva	1973-04-06	88...	Zalaszentjakab,
9	Németh Zoltán	1976-02-16	89...	Zalaszentgyörgy
10	Baracskai Józsefné	1948-05-28	87...	Zalaszentgrót
11	Kissné Bálint Mária	1949-07-22	87...	Zalaszentgrót
12	Németh Béla	1954-10-26	87...	Zalaszentgrót
13	Kópicz Csaba	1960-03-01	87...	Zalaszentgrót
14	Vajda Renáta	1981-05-22	83...	Zalaszántó
15	Szabados Gyula	1961-09-05	87...	Zalakaros
16	Décsy Jenő	1936-12-06	89...	Zalaegerszeg
17	Márton Tiborné	1949-07-31	89...	Zalaegerszeg

- Elsőre talán a legfontosabb információkat az ablak fejlécéből olvashatjuk ki!
- A File ikonra kattintva az eredménylistát elmenthetjük. Természetesen erre a célra használhatjuk a **FLAT FILE** node-ot is az Output palettáról, ha a mentést automatikusan szeretnénk végrehajtani.

A további lehetőségek megismeréséhez nyissuk meg a Table node beállításait, és váltsunk az OUTPUT fülre!

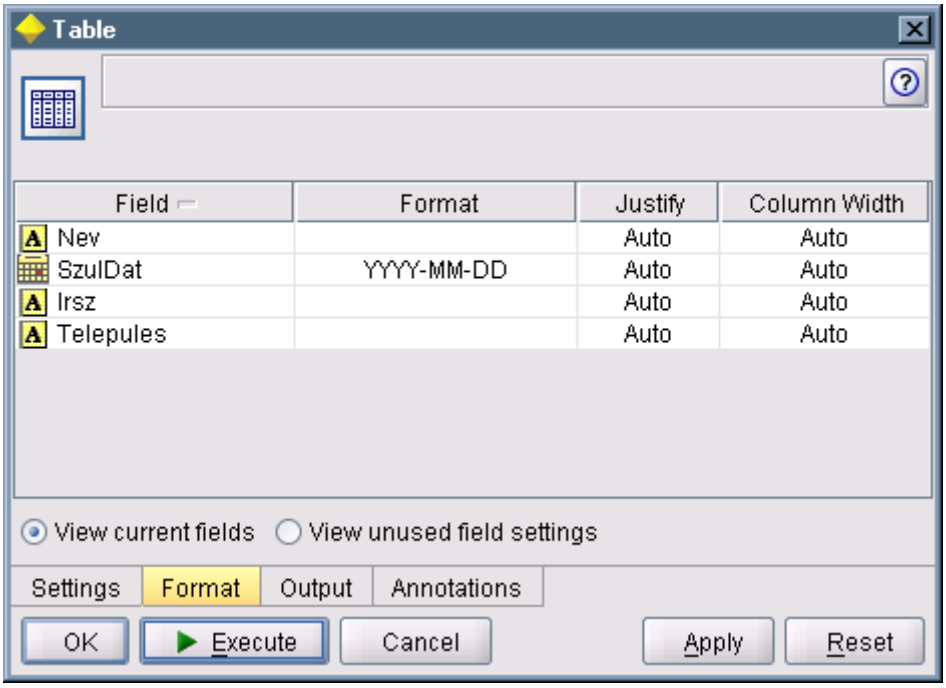


- Az **OUTPUT NAME** rovatban a **CUSTOM** opciót választva egy tetszőleges szöveget adhatunk meg, mely a kimenet „címe" lesz. Megadása esetén az eredményt tartalmazó ablak címsorában a standard (mező és rekordszám) üzenet helyett az itt megadott szöveg lesz látható.
- Ha az **OUTPUT TO SCREEN** opció helyett az **OUTPUT TO FILE** opciót választjuk, akkor a node futtatása esetén az eredményeket nem a képernyőn jeleníti meg, hanem a megadott fájlba menti el.
  A **FILE TYPE** beállításával adatainkat elmenthetjük tab-bal vagy vesszővel szeparált fájlba, weblapba ágyazott táblázatként, vagy a *.cou saját formátumba is.
  A **TRANSPOSE DATA** bejelölésével mentés előtt transzponálja az adattáblát.

---

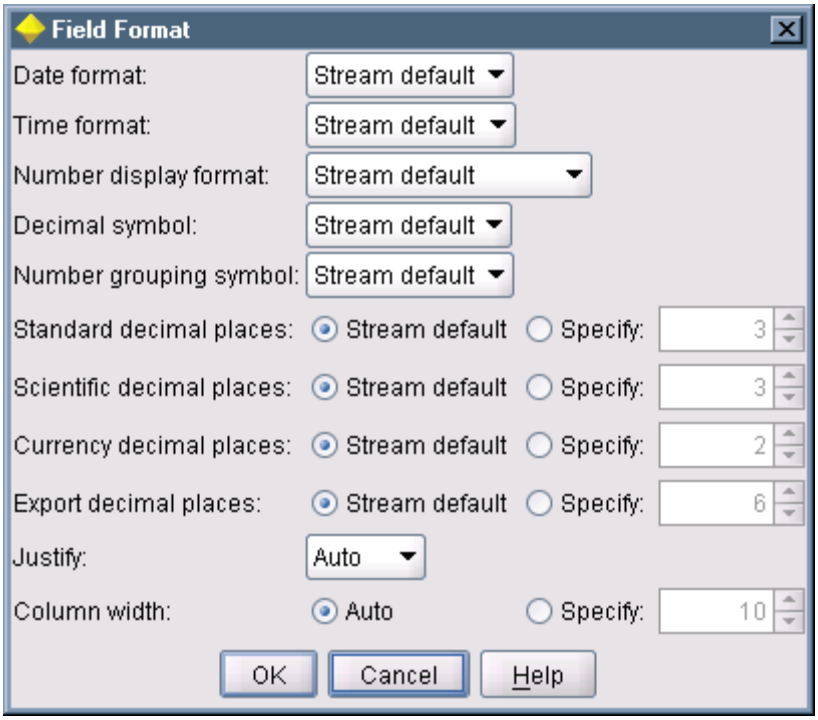
*Ha a table node futtatásának eredményét az Output palettáról kijelölve szeretnénk elmenteni, akkor ott csak a *.cou formátumot választhatjuk.*

Az elmentett, vagy képernyőn megjelenített adatok formai beállításait a **FORMAT** fülön adhatjuk meg:

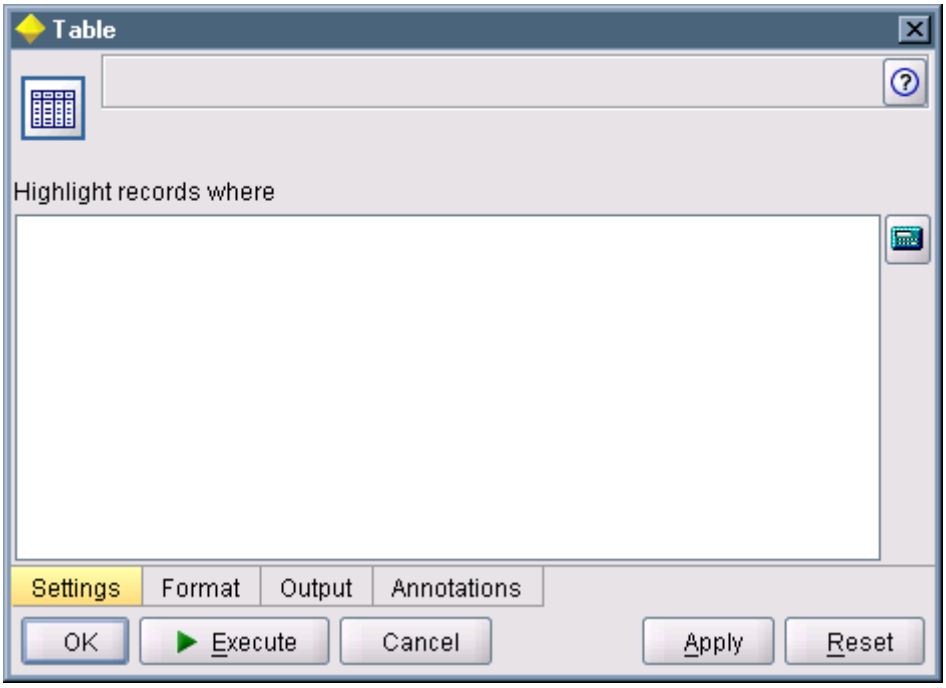


- A **FORMAT** oszlopban a megjelenést szabályozó formátumkódot adhatjuk meg.
- A **JUSTIFY** oszlopban az adatok vízszintes igazítását.
- A **COLUMN WIDTH** oszlopban pedig a megjelenített oszlopok szélességeit szabályozhatjuk.

Ha duplán rákattint valamelyik mező nevére, akkor egy párbeszédablak jelenik meg, melyen könnyebben és részletesebben megadhatja a beállításokat:



Végül lássuk, mit is lehet beállítani a **SETTINGS** fülön:



Ezen a fülön megadhatunk egy logikai kifejezést, melyet a node a tábla minden egyes sorára kiértékel, és ha az eredmény az adott sorra igaz (true) lesz, akkor az adott sort kiemelten jeleníti meg.

A logikai kifejezés megszerkesztésében ismét csak segítségünkre lehet a kifejezés-szerkesztő, melyet a  *ikonra kattintva nyithatunk meg.*

Emelje ki azon rekordokat, ahol az irányítószám értéke 9700!

Ha helyes képletet adott meg, akkor hasonló kell legyen a node futási eredménye:

	Nev	SzulDat	Irsz	Telepules
1	Szilágyiné Horváth Rita	1970-09-08	94...	Zsira
2	Bartek Péter Pál	1972-04-19	97...	Zsennye
3	Szakáts Dezsőné	1956-07-17	84...	Zirc
4	Futó János	1959-03-16	84...	Zirc
5	Póda Károly	1960-08-27	84...	Zirc
6	Béres Jánosné	1953-02-03	80...	Zámoly
7	Nagy Tibor	1976-02-03	87...	Zalaszentlászló
8	Magyarné Miatta Éva	1973-04-06	88...	Zalaszentjakab,
9	Németh Zoltán	1976-02-16	89...	Zalaszentgyörgy
10	Baracskai Józsefné	1948-05-28	87...	Zalaszentgrót
11	Kissné Bálint Mária	1949-07-22	87...	Zalaszentgrót
12	Németh Béla	1954-10-26	87...	Zalaszentgrót
13	Kópicz Csaba	1960-03-01	87...	Zalaszentgrót
14	Vajda Renáta	1981-05-22	83...	Zalaszántó
15	Szabados Gyula	1961-09-05	87...	Zalakaros
16	Décsy Jenő	1936-12-06	89...	Zalaegerszeg
17	Márton Tiborné	1949-07-31	89...	Zalaegerszeg
18	Lukácsné Simon Ágn...	1951-11-06	89...	Zalaegerszeg
19	Budaházi Tibor	1954-02-01	89...	Zalaegerszeg
20	Őriné Dr. Bilkei Irén	1954-06-23	89...	Zalaegerszeg

A feladat megoldását megtalálja a ***DEMOS.STR*** *állományban.*

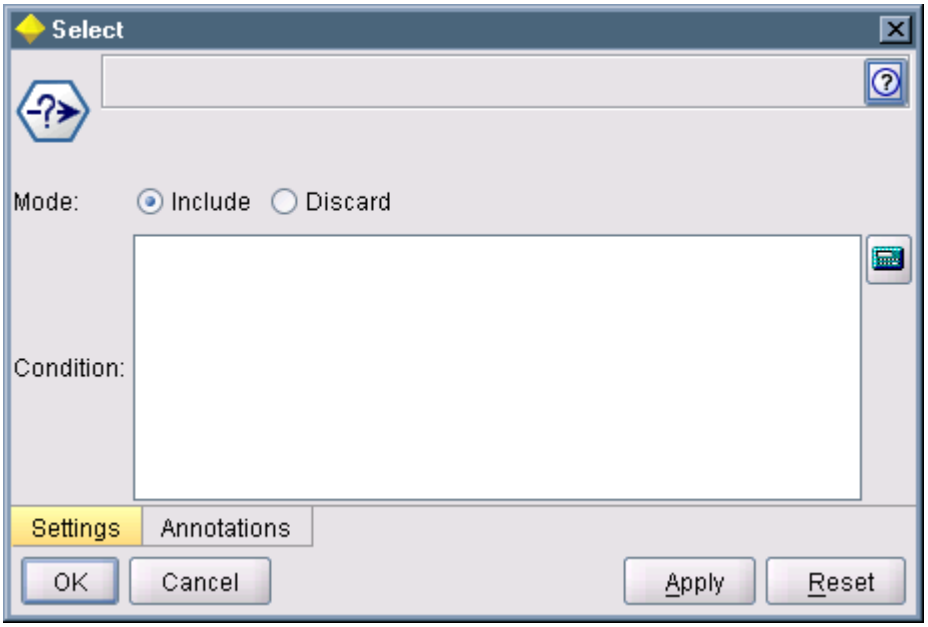
Select node 

Nagyon sokszor nemcsak arra van szükségünk, hogy egyes adatokat kiemelve jelenítsünk meg a többi adat között, hanem a további adatfeldolgozó utasításainkat szeretnénk az adatok egy részére korlátozni. Másképp fogalmazva, a további műveletekből bizonyos adatokat szeretnénk kizárni.

Ha az adattáblánknak csak bizonyos soraira van szükségünk a további műveletekhez, akkor a megfelelő sorok kigyűjtését a Select node segítségével végezhetjük el.

Kapcsoljunk a source node-hoz egy SELECT  **node-ot a** RECORD OPS **palettáról!**

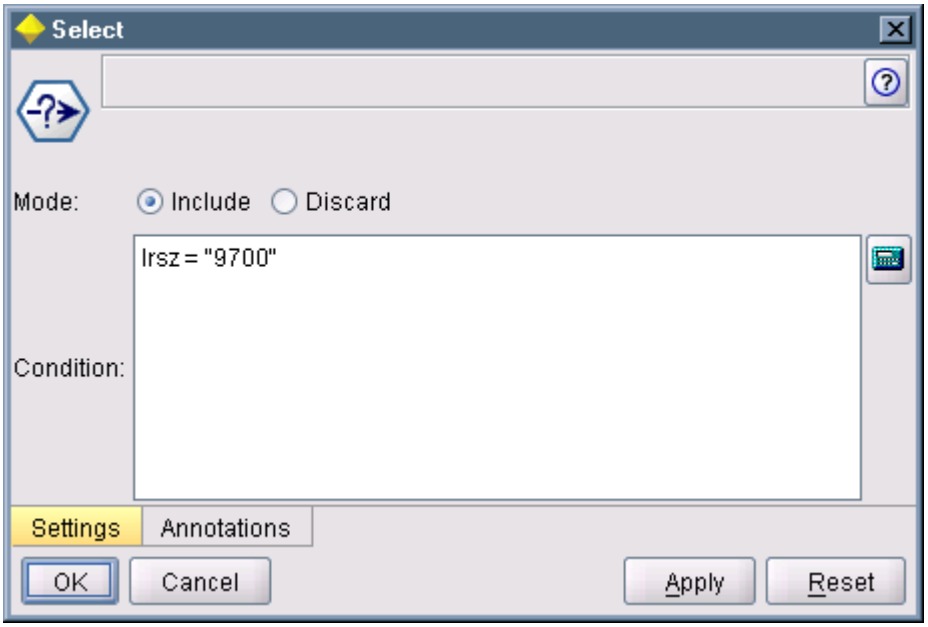
A node beállításait megnyitva a következőket adhatjuk meg:



- A **CONDITION** rovatba egy logikai kifejezést írhatunk, mely minden sorra ki fog értékelődni.
- Ha a **MODE** rovatban az **INCLUDE**-t jelöljük meg, akkor azokat a sorokat engedi tovább a kimenetre, amelyekre igaz a megadott feltétel, ha a **DISCARD**-ot, akkor pedig azokat zárja ki, amelyekre igaz a feltétel.

Csak azokat a sorokat tartsuk meg, ahol az irányítószám értéke 9700!

Tehát a következőket kell beállítanunk:



Ellenőrizze a helyes működést, egy Table node-dal!

A teljesség igénye nélkül álljon itt néhány további példa lehetséges szűrési feltételekre:

- **TELEPULES MATCHES "R*"**

  Ahol a település neve nagy R betűvel kezdődik

- **TELEPULES MATCHES "*R*"**

  Ahol a település neve tartalmaz nagy R betűt

- **TELEPULES MATCHES "??????"**

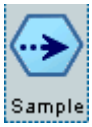
  Ahol a település neve pontosan 6 karakteres

- **DATETIME_YEAR(SZULDAT)=1980**

  1980-ban született

---

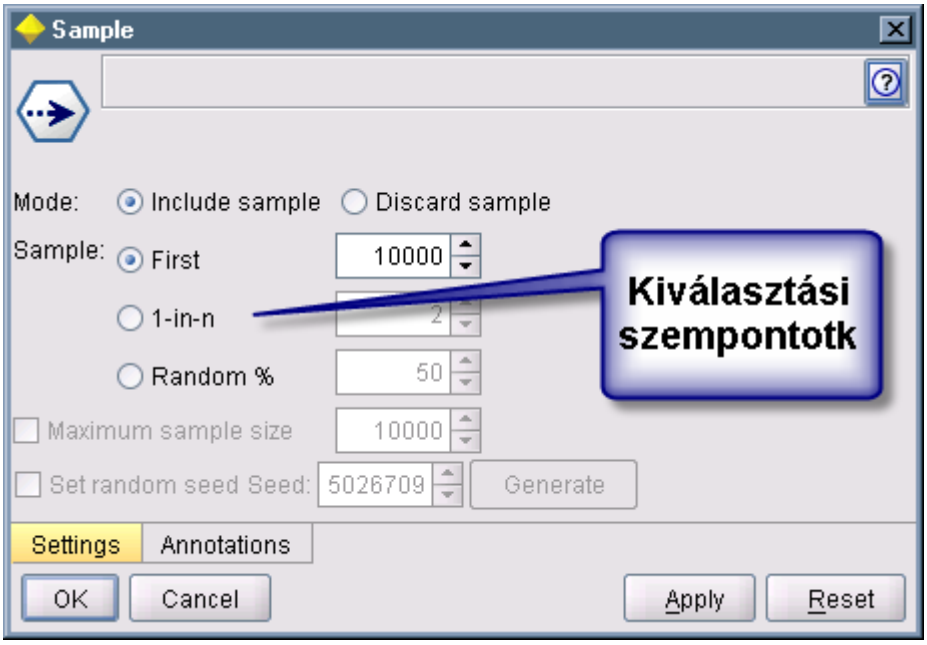
*A matches operátorral a Modeler súgója alapján, csak a * és ? helyettesítő karaktereket lehet használni. ☹ Ha ezek közül bármelyiket kell illeszteni, akkor a \ karaktert kell előtagként használni. Például, ha azokat a szövegeket szeretnénk illeszteni, amelyek * karakterrel kezdődnek, akkor a következő mintát adhatjuk meg: "**".*

Sample node 

Az adattábla sorainak szűrésére a Sample node is használható. Ellentétben a Select node-dal ebben az esetben nem egy logikai feltétel igaz vagy hamis volta dönti el, hogy az adott sor bekerül-e a további adatfeldolgozásba. A sorok kiválasztásának más szempontjait alkalmazhatjuk a node használatával:

- Az első valahány sor
- Minden valahányadik sor
- Véletlen kiválasztott sorok

Helyezzen el egy Sample node-ot a source node után kapcsolva! Nyissa meg a node beállításait!



A **SAMPLE** rovatban állíthatjuk be, hogy mi legyen a rekordok kiválasztásának az alapelve:

- **FIRST:**

  Az első valahány rekord

- **1-IN-N:**

  Minden n-edik rekord

- **RANDOM %:**

  Véletlen rekordok

A **MODE** rovatban, ha az **INCLUDE SAMPLE** helyett a **DISCARD SAMPLE** beállítást választjuk ki, akkor a node pont fordítva működik, és a **SAMPLE** rovat által kijelölt sorokat zárja ki a további feldolgozásból.

A **MAXIMUM SAMPLE SIZE** opciót csak akkor adhatjuk meg, ha **1-IN-N VAGY RANDOM %** módon vesszük a mintát. Használatával korlátozhatjuk a minta méretét.

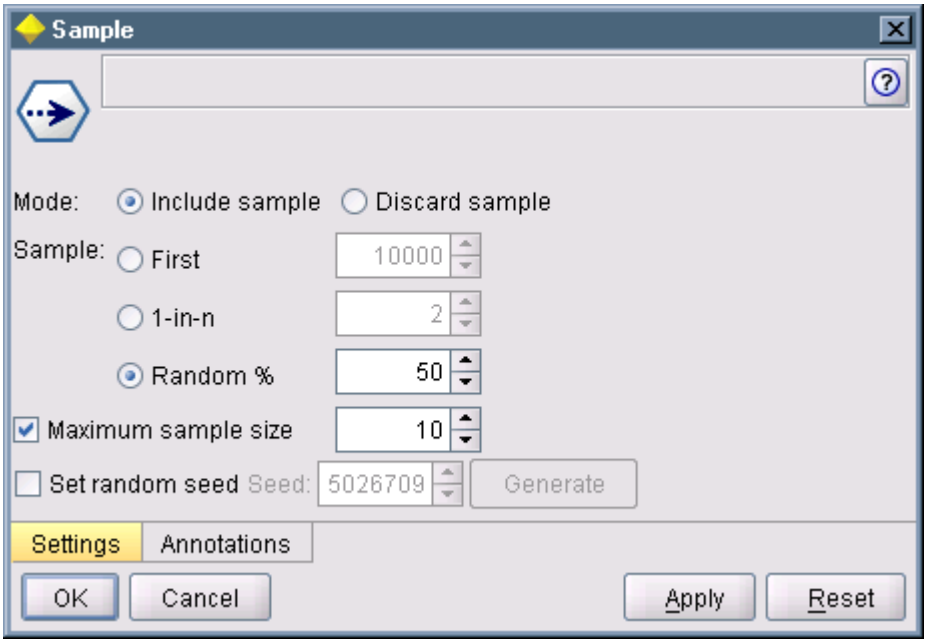
A **SET RANDOM SEED** opció a **RANDOM %** mód esetén jelölhető be. Használatával megismételhetjük a mintavételezést egy másik futtatás esetén. Ha ugyanazt az értéket adjuk meg, akkor ugyanabból a forráshalmazból, ugyanazt a mintát fogja generálni. A **GENERATE** gomb használatával automatikusan megadja a véletlen mintát generáló értéket. (Nem kell nekünk kitalálni, beírni.) Ha az opció ki van kapcsolva, akkor minden egyes futtatás eredménye teljesen véletlenszerű lesz.

---

*Ha adatainkat relációs adatbázisból nyerjük, akkor szükséges lehet egy **SORT** node használata a **SAMPLE** node előtt, ha megismételhető véletlen mintát szeretnénk kapni. Ennek oka, hogy a relációs adatbázisokban, nem meghatározott a sorok sorrendje, ellentétben például egy Excel táblázattal, vagy tagolt szövegfájllal.*

Állítsa be úgy a node-ot, hogy a táblázatból véletlenszerűen listázzon ki 10 sort!

Ha sikerült, akkor a következőket kell látnunk:



A maximum sample size garantálja, hogy az eredményben ne legyen 10-nél több rekord.

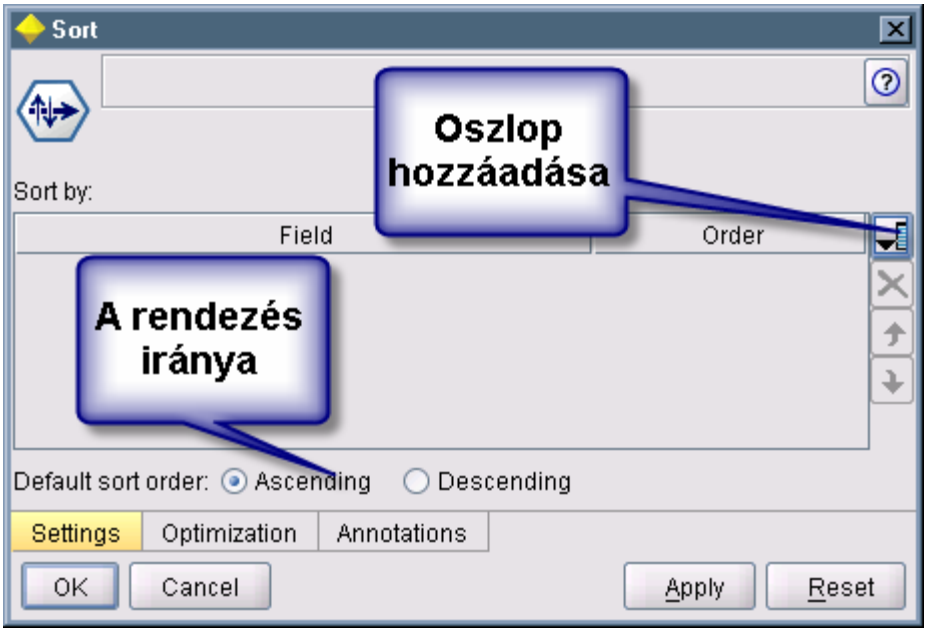
A Random % -nál legalább akkora részét állítsuk be a mintának, hogy az tartalmazzon legalább 10 rekordot.

Kapcsoljon a node után egy Table output node-ot, és futtassa az eredmények ellenőrzéséhez!

Sort node 

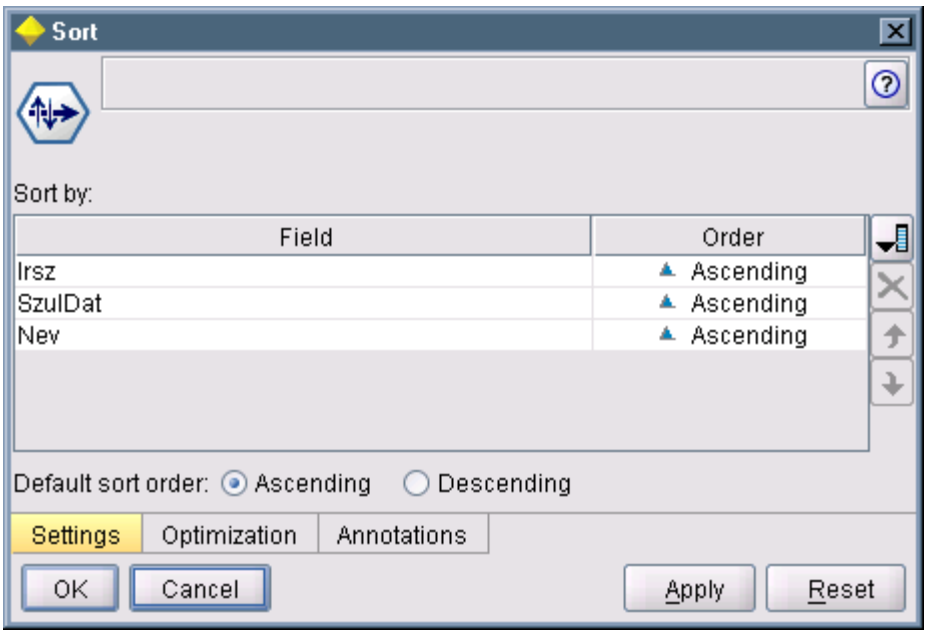
A node segítségével adattáblánk sorait rendezhetjük, valamely oszlop(ok) értékei alapján. Ha több oszlopot is megadunk, mint rendezési szempontot, akkor számít az oszlopok sorrendje. Elsősorban az elsőnek megadott oszlop értékei alapján rendez, ha ez nem lehetséges egyező értékek miatt, akkor a második oszlop értékei alapján, stb.

Kapcsoljon a source node után egy Sort node-ot! Nyissa meg a node beállításait!



Adja meg elsődleges rendezési szempontnak az irányítószámot, másodlagosnak a születési dátumot, és harmadlagos szempontnak a nevet. Mindegyik szempont szerint növekvő rendezést állítson be!

A feladat sikeres végrehajtása után a következőt kell látnunk:



Kapcsoljon a node után egy table node-ot, és ellenőrizze az eredményt a node futtatásával!

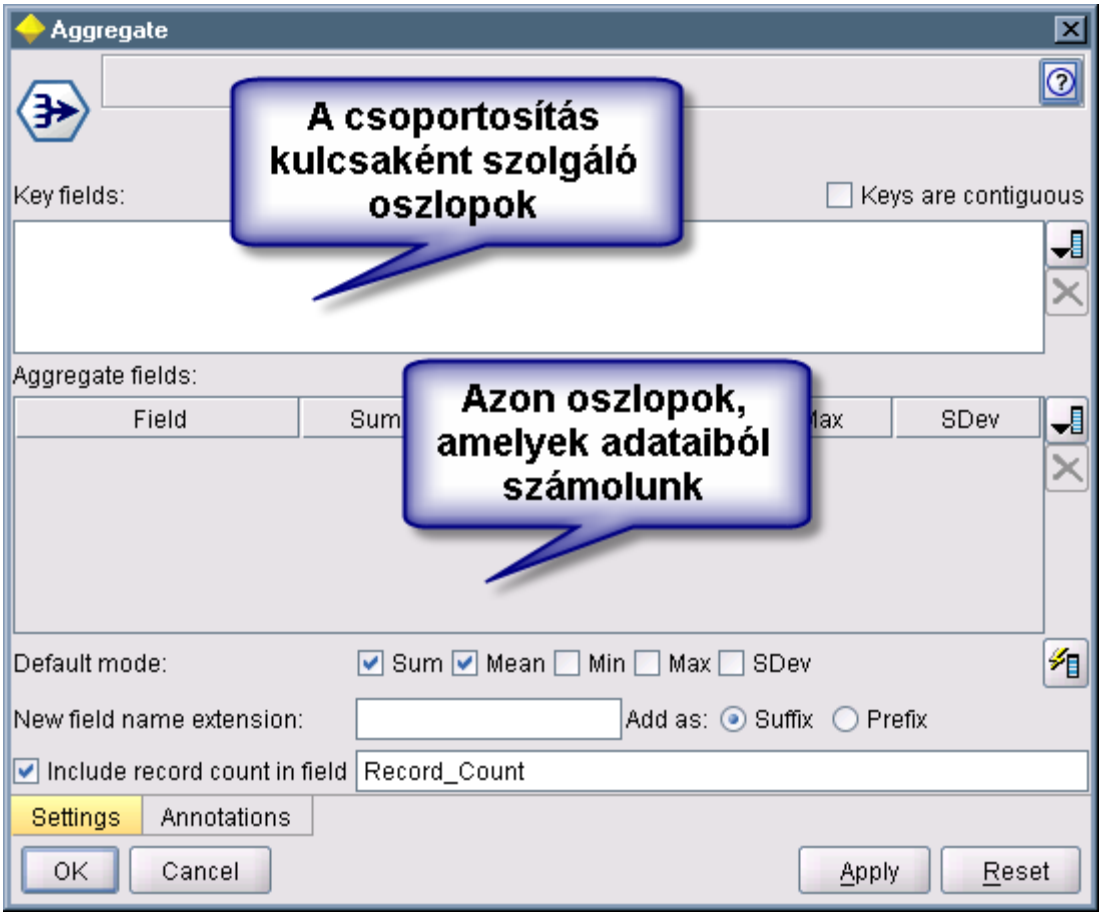
Aggregate node 

A node segítségével az adattábla soraiból csoportok képezhetők, az általunk kiválasztott oszlopok értékeinek egyezősége alapján. Például csoportosíthatjuk a sorokat úgy, hogy egy csoportba az azonos irányítószámú településen lakók kerüljenek, vagy úgy, hogy az egy napon születettek kerüljenek egy csoportba, vagy akár a két szempont közös használatával egy csoportba az azonos napon születettek és azonos településen lakók kerüljenek.

A node lehetőséget biztosít arra, hogy a képződött csoportok adataiból a következő számításokat végezzük:

- Összegzés (Sum)
- Átlagolás (Mean)
- Minimum (Min)
- Maximum (Max)
- Szórás (SDev)
- A csoportban lévő sorok száma

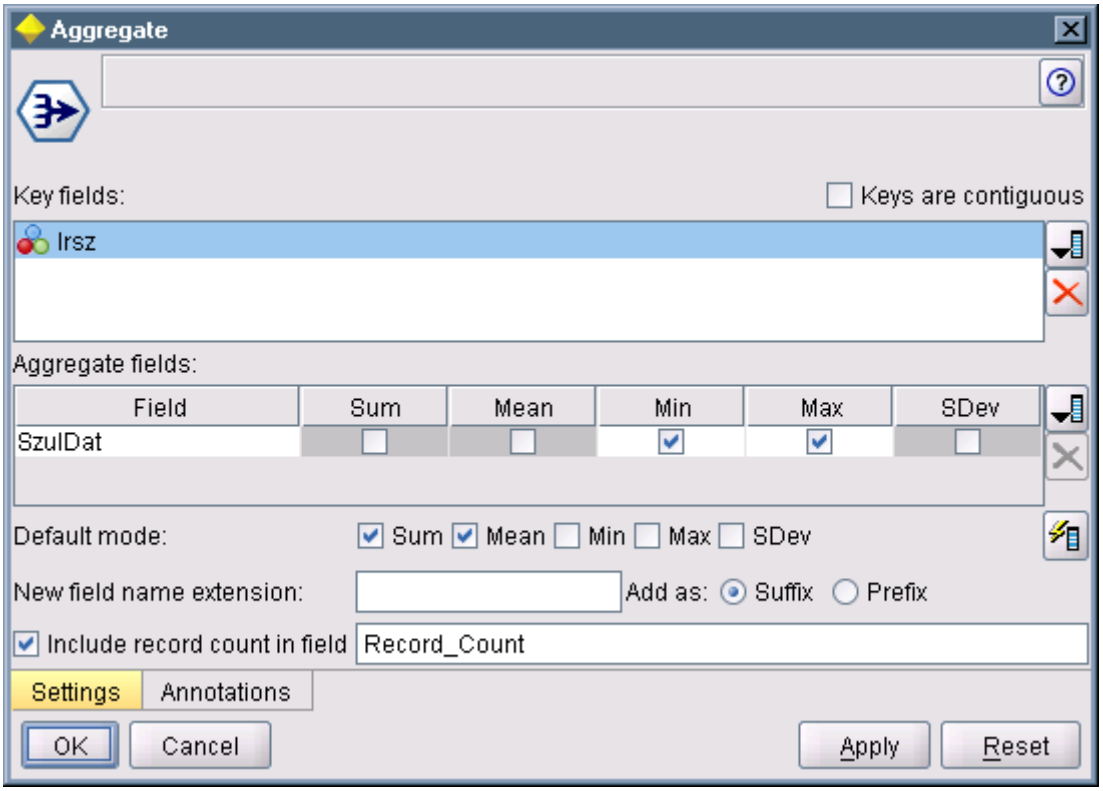
Kapcsoljon a source node után egy Aggregate node-ot! Nyissa meg a node beállításait!



Csoportosítsa a tábla sorait az irányítószám oszlop értékei alapján!

Számolja a csoportokra a legfiatalabb és a legidősebb születési dátumát!

A beállítások elvégzése után ezt látjuk:



Néhány további beállítás:

- Ha a **KEYS ARE CONTIGIOUS** opciót bekapcsoljuk, akkor az azonos kulcsértéket tartalmazó sorok csak akkor fognak egy csoportot alkotni, ha az adott sorok egymás mellett is vannak. Tehát, ha lenne 10 darab 1234 –es irányítószámot tartalmazó sor, de nem egymás mellett, hanem pl. 3-5-2 –es megoszlásban, akkor ebből a 10 sorból nem egy, hanem 3 csoport keletkezne. Viszont ha mind a 10 sor egymás mellett van, akkor az opció bekapcsolása esetén is egy csoportot alkotnának.
- Az **INCLUDE RECORD COUNT IN FIELD** opció bekapcsolásával az eredmények közé beszúr a node egy új oszlopot, mely a csoportokat alkotó sorok darabszámát fogja tartalmazni. Megadhatjuk az oszlop nevét is, ha nem felel meg az alapértelmezett Record_count név.
- A **NEW FIELD NAME** extension kitöltésével a számolt oszlopok nevéhez szabadon meghatározható elő, vagy utótagot rakhatunk.

Kapcsoljon az Aggregate node után egy Table output node-ot, és futtassa az eredmény ellenőrzéséhez!

Önálló feladat

Az Eladasok.xls fájl egy fiktív cég termékeladásait tartalmazza. A táblázat minden egyes sora egy konkrét értékesítést jelent. A táblázat oszlopainak a jelentése:

- Kategórianév:
  Az eladott termék kategóriája
- Terméknév:

Az eladott termék neve
- Egységár:
  Az eladott termék egységára
- Mennyiség:
  A termékből eladott mennyiség
- Cégnév:
  A vevő
- Ország:
  A vevő országa
- Város:
  A vevő városa

Listázza ki annak a 10 terméknek a nevét, amelyekből a legnagyobb mennyiségben adtak el. Ne számítson bele az eredménybe egy olyan sort sem, amikor a vevő a „QUICK-Stop” nevű cég volt.

4. Mező műveletek

A gyakorlat fő célja, hogy megismerkedjünk a Modeler-ben számítások elvégzésére alkalmas utasításokkal és függvényekkel. Képletek megírására számtalan esetben lehet szükségünk. Nemcsak akkor, amikor új számított oszlopot kell létrehoznunk, hanem akkor is, amikor például a Select node-ban meg kell adnunk a sorok kiválasztási feltételét.

Mindenekelőtt szükségünk lesz egy adatforrásra. Erre a célra használjuk fel ismét a már ismert Eladasok.xls állományt.

Helyezzen el egy üres stream-ben egy EXCEL source node-ot, és olvassa be az ELADASOK.XLS adatállományt!

Kapcsoljon a node után egy TABLE nodot, és futtassa!

Table (7 fields, 2 155 records) #2

File Edit Generate

	Kategórianév	Terméknév	Egységár	Mennyiség	Cégnév	Ország	Város
1	Italok	Chai	1800.000	45.000	QUICK-Stop	Németország	Cunewalde
2	Italok	Chai	1800.000	18.000	Rattlesnake Canyon Grocery	USA	Albuquerque
3	Italok	Chai	1800.000	20.000	Lonesome Pine Restaurant	USA	Portland
4	Italok	Chai	1800.000	15.000	Die Wandernde Kuh	Németország	Stuttgart
5	Italok	Chai	1800.000	12.000	Pericles Comidas clásicas	Mexikó	México
6	Italok	Chai	1800.000	15.000	Chop-suey Chinese	Svájc	Bern
7	Italok	Chai	1800.000	10.000	Queen Cozinha	Brazília	Sao Paulo
8	Italok	Chai	1800.000	24.000	La maison d'Asie	Franciaország	Toulouse
9	Italok	Chai	1800.000	15.000	Princesa Isabel Vinhos	Portugália	Lisboa
10	Italok	Chai	2250.000	40.000	Lehmanns Marktstand	Németország	Frankfurt a.M.
11	Italok	Chai	2250.000	8.000	Wartian Herkku	Finnország	Oulu
12	Italok	Chai	2250.000	10.000	Tortuga Restaurante	Mexikó	México
13	Italok	Chai	2250.000	20.000	Mcre Paillarde	Kanada	Montreal
14	Italok	Chai	2250.000	3.000	Du monde entier	Franciaország	Nantes
15	Italok	Chai	2250.000	6.000	Wolski Zajazd	Lengyelország	Warsawa
16	Italok	Chai	2250.000	25.000	Blondel pcre et fils	Franciaország	Strasbourg
17	Italok	Chai	2250.000	15.000	Hungry Owl All-Night Grocers	Írország	Cork
18	Italok	Chai	2250.000	35.000	Berglunds snabbköp	Svédország	Lulel
19	Italok	Chai	2250.000	30.000	QUICK-Stop	Németország	Cunewalde
20	Italok	Chai	2250.000	5.000	Save-a-lot Markets	USA	Boise

Table Annotations

Az adatokat látva több alapvető kérdés is felmerülhet:

Terméktípusonként, termék kategóriánként, évenként, negyedévenként, vevőkként, országonként, stb. mekkora a bevétel?

Ezen kérdések megválaszolásához ki kell tudnunk számolni minden egyes adatsorra az eladás összértékét, vagyis a mennyiség és egységár szorzatát.

Derive node

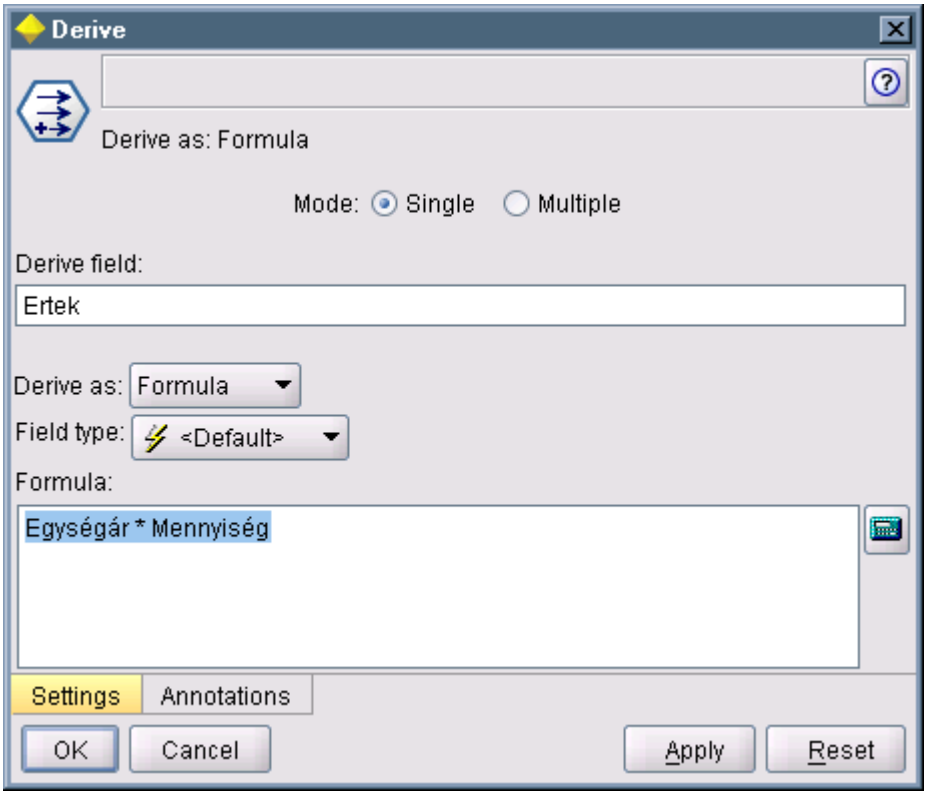
A node feladata, hogy az adattáblába új oszlopot hozzon létre, melynek az adatsorokban felvett értékeit egy általunk megadott képlet határozza meg.

Kapcsoljon a source node után egy DERIVE node-ot a RECORD OPS palettáról, és nyissa meg a node beállításait!

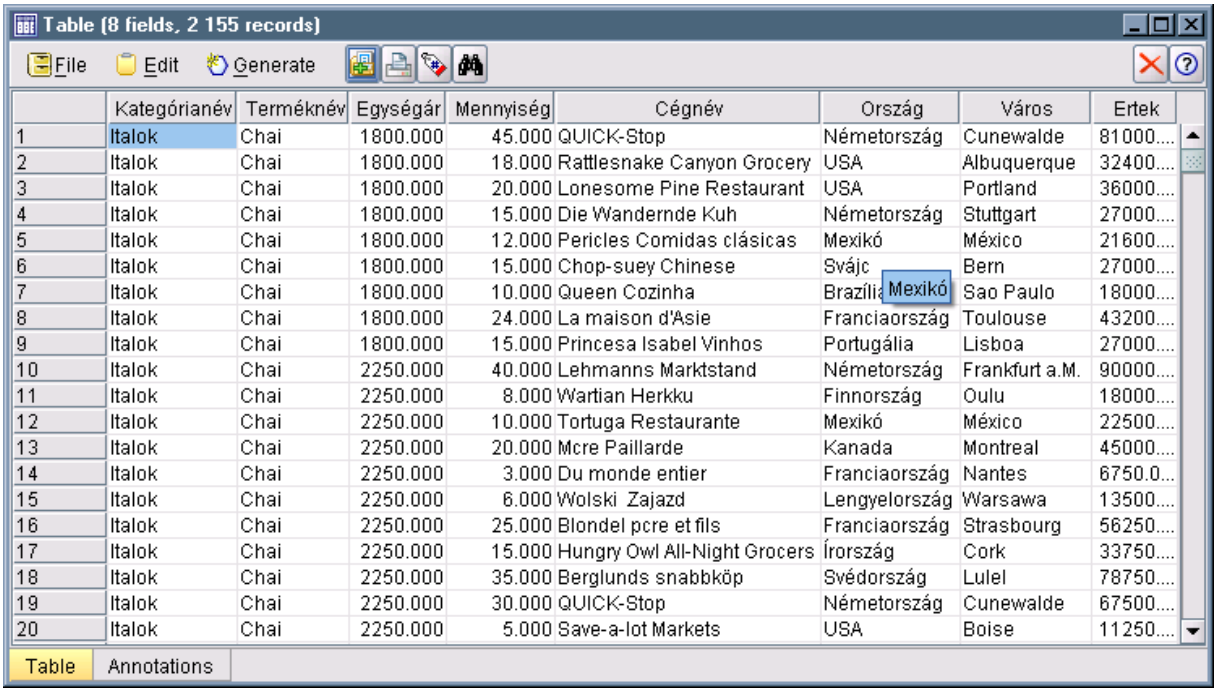
A node rengeteg opciót rejteget. Először a lehető legegyszerűbb módon próbáljuk ki a működését.

Az új oszlop nevének adja meg a következőt: ERTEK

Képletnek pedig ezt: EGYSÉGÁR * MENNYISÉG



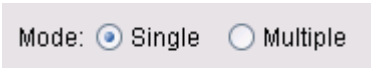
Ok –ozza le a node-ot, és egy Table node-dal ellenőrizze az eredményt!



	Kategórianév	Terméknév	Egységár	Mennyiség	Cégnév	Ország	Város	Ertek
1	Italok	Chai	1800.000	45.000	QUICK-Stop	Németország	Cunewalde	81000....
2	Italok	Chai	1800.000	18.000	Rattlesnake Canyon Grocery	USA	Albuquerque	32400....
3	Italok	Chai	1800.000	20.000	Lonesome Pine Restaurant	USA	Portland	36000....
4	Italok	Chai	1800.000	15.000	Die Wandernde Kuh	Németország	Stuttgart	27000....
5	Italok	Chai	1800.000	12.000	Pericles Comidas clásicas	Mexikó	México	21600....
6	Italok	Chai	1800.000	15.000	Chop-suey Chinese	Svájc	Bern	27000....
7	Italok	Chai	1800.000	10.000	Queen Cozinha	Brazília Mexikó	Sao Paulo	18000....
8	Italok	Chai	1800.000	24.000	La maison d'Asie	Franciaország	Toulouse	43200....
9	Italok	Chai	1800.000	15.000	Princesa Isabel Vinhos	Portugália	Lisboa	27000....
10	Italok	Chai	2250.000	40.000	Lehmanns Marktstand	Németország	Frankfurt a.M.	90000....
11	Italok	Chai	2250.000	8.000	Wartian Herkku	Finnország	Oulu	18000....
12	Italok	Chai	2250.000	10.000	Tortuga Restaurante	Mexikó	México	22500....
13	Italok	Chai	2250.000	20.000	Mcre Paillarde	Kanada	Montreal	45000....
14	Italok	Chai	2250.000	3.000	Du monde entier	Franciaország	Nantes	6750.0...
15	Italok	Chai	2250.000	6.000	Wolski Zajazd	Lengyelország	Warsawa	13500....
16	Italok	Chai	2250.000	25.000	Blondel pcre et fils	Franciaország	Strasbourg	56250....
17	Italok	Chai	2250.000	15.000	Hungry Owl All-Night Grocers	Írország	Cork	33750....
18	Italok	Chai	2250.000	35.000	Berglunds snabbköp	Svédország	Lulel	78750....
19	Italok	Chai	2250.000	30.000	QUICK-Stop	Németország	Cunewalde	67500....
20	Italok	Chai	2250.000	5.000	Save-a-lot Markets	USA	Boise	11250....

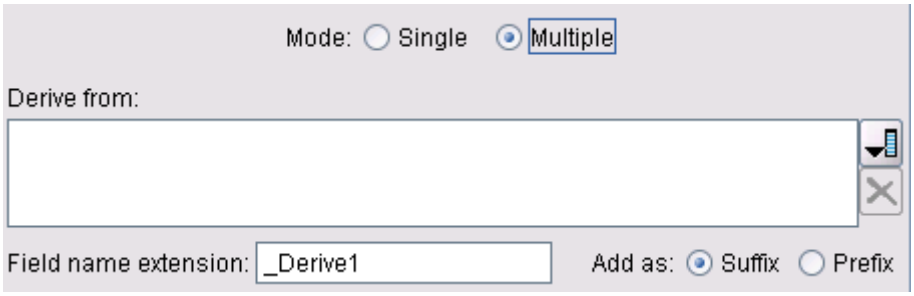
A továbbiakban nézzük át a node részletes beállításait:

Mode



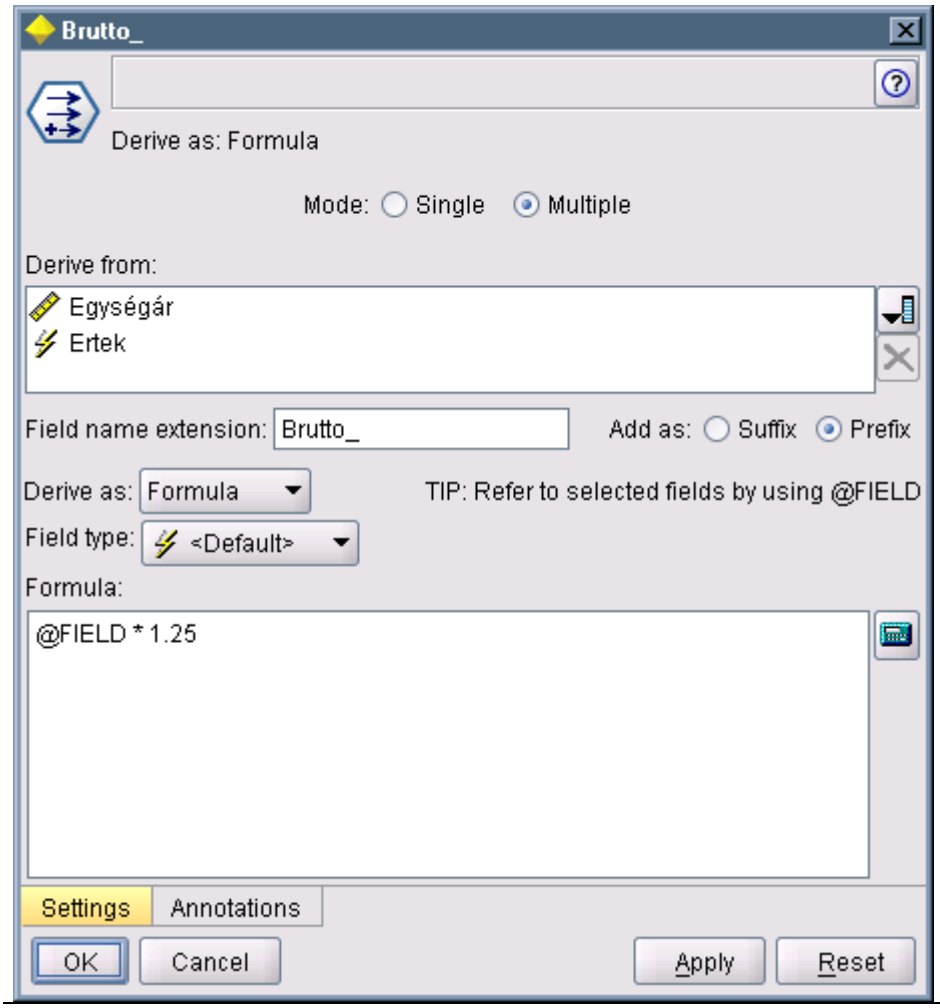
Ha a mode rovat **SINGLE**-re van állítva, akkor a node csak egy új oszlopot hoz létre, melynek értékeit a megadott képlet határozza meg.

Ha a mode értéke **MULTIPLE**, akkor egyszerre több számított oszlopot is létrehozhatunk:



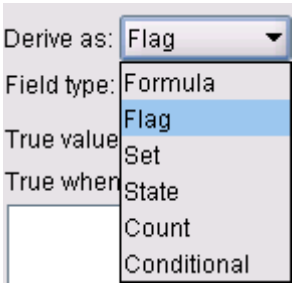
Hogy az új oszlopok mely eredetileg meglévő oszlopok értékeiből legyenek számolva, mi határozhatjuk meg a **DERIVE FROM** rovatban. Az új oszlopok nevei az eredeti oszlopok neveiből és egy általunk meghatározható elő, vagy utótagból tevődnek össze, a **FIELD NAME EXTENSION** rovatban beállítottak szerint.

Mivel minden egyes új oszlop értékeit az eredeti oszlopokból azonos képlettel számolja a node, ezért általában elmondható, hogy azonos adattípusú oszlopok esetében alkalmazható ez a módszer. Például:
Adott két oszlop: Egységár és Érték
Ha szeretnénk a két oszlopból ÁFA-val megnövelt értékeket is számolni, akkor azt a következő beállításokkal tehetjük:



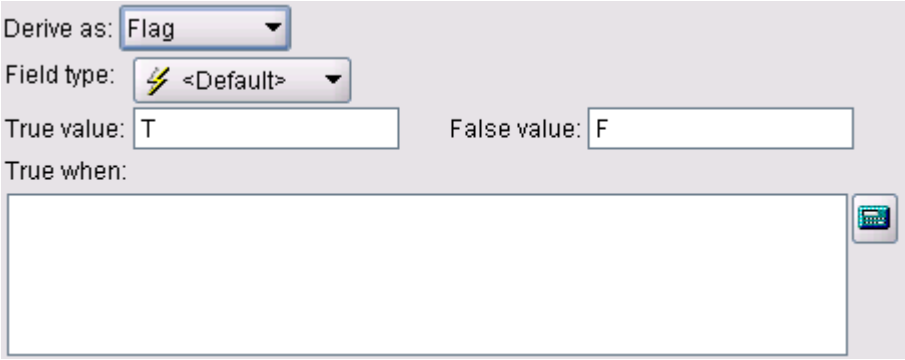
Multiple mód esetén a **@FIELD** függvény segítségével hivatkozhatunk a képlet végrehajtódása alatt éppen aktuális oszlop értékére!

Derive as rovat

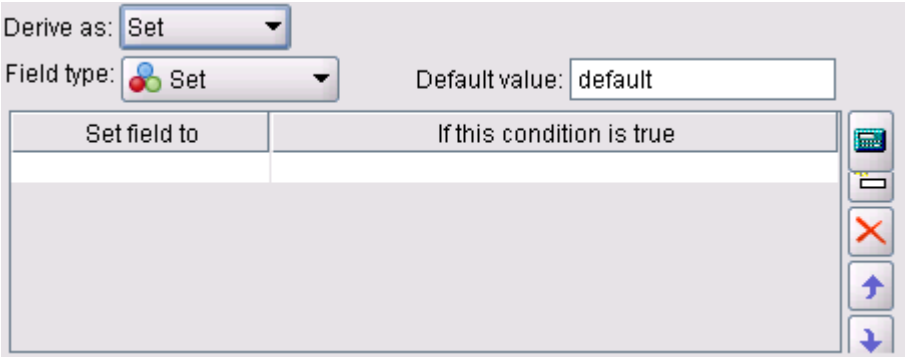


Az új oszlop kiszámítási módját határozza meg:

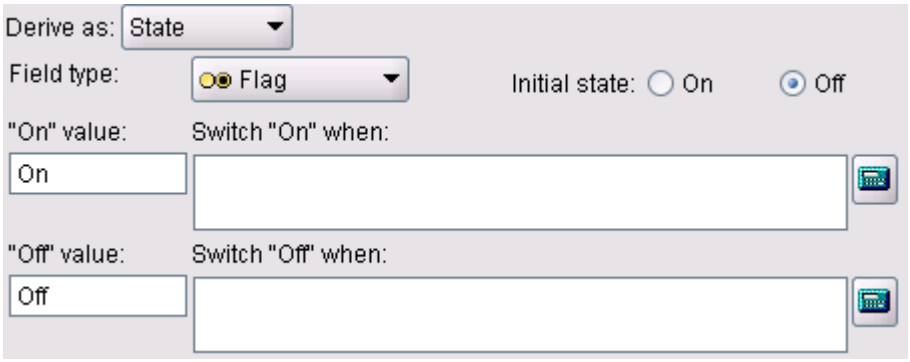
- Formula:
  Általunk megadott képlet alapján
- Flag:
  Az új oszlop kétféle értéket vehet fel:



  Megadhatjuk, hogy mi legyen ez a két érték, és azt a logikai képletet, amelynek igaz visszatérési értékei esetén a **TRUE VALUE** értéket, egyébként a **FALSE VALUE** értéket veszi fel az új oszlop.
- Set:
  Az új oszlop értékei egy halmaz elemei közül kerülnek ki:

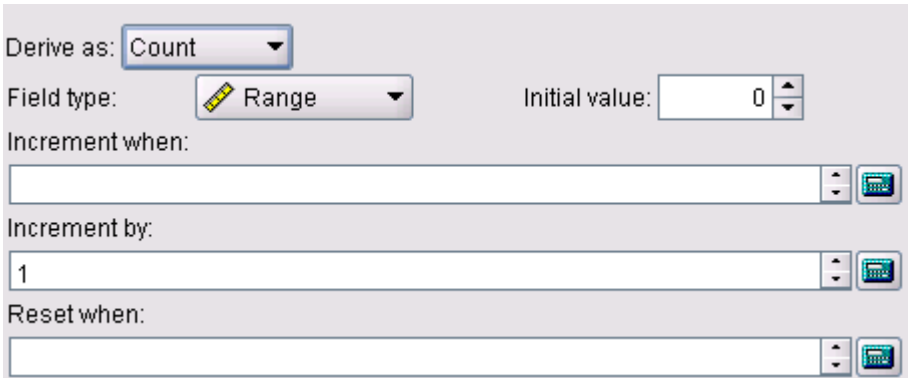


  A táblázat bal oszlopában sorolhatjuk fel a lehetséges értékeket (halmaz elemei), míg a jobboldalon minden egyes értékhez megadhatunk egy logikai kifejezést, melynek igaz visszatérési értéke esetén a mellette lévő értéket veszi fel az új oszlop adott sora. Számít az értékek sorrendje (nyilakkal állítható), mert több feltétel teljesülése esetén a feljebb lévő értéket kapja meg az új oszlop. Ha egyik feltétel sem teljesül, akkor a **DEFAULT VALUE** rovatban megadott értéket veszi fel az új oszlop adott sora.
- State:
  Nagyon hasonló a Flag beállításhoz.



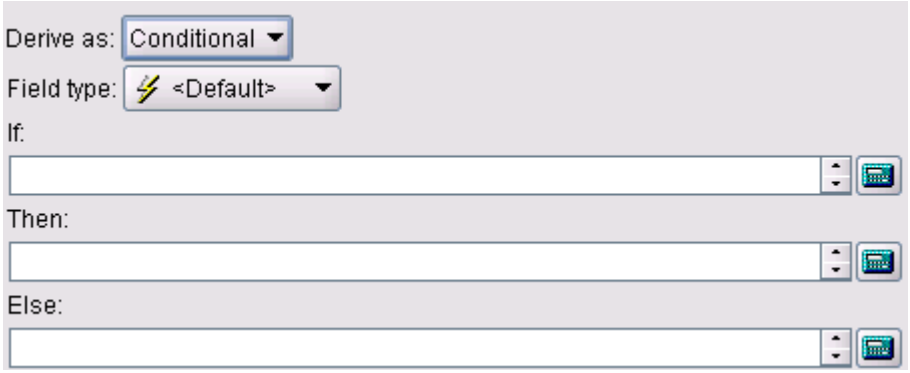
Csak míg a Flag választás esetén egyetlen feltétel igaz, vagy hamis eredménye alapján dől el, hogy melyik értéket veszi fel a két lehetséges közül az oszlop, addig itt mindkét értékhez tartozik egy logikai kifejezés, melyeknek igaz értéke esetén felveszi az adott értéket (átvált rá), hamis értéke esetén viszont az utoljára kiosztott értéket adja a mezőnek. Az első sor kiszámolása esetén az alapértéknek az **INITIAL STATE** rovatban megjelöltet veszi.
Olyan mint egy kapcsoló (state ~ állapot), melyet ha kell átbillenthetünk.

- Count:
  Kiválóan alkalmas sorszámozásra, leszámlálásra.



Az **INITIAL VALUE** rovatban megadhatjuk a kezdőértéket, melyet minden egyes új sor esetén megnövel az **INCREMENT BY** rovatban megadott értékkel, ha az **INCREMENT WHEN** rovatban megadott logikai kifejezés igaz értéket vesz fel.
Ha a **RESET WHEN** rovatban megadunk egy logikai kifejezést, akkor minden egyes olyan sortól kezdve, ahol ez a feltétel igaz értéket ad, újraindítja a számolást.

- Conditional:
  A tipikus HA … AKKOR … EGYÉBKÉNT … formula:

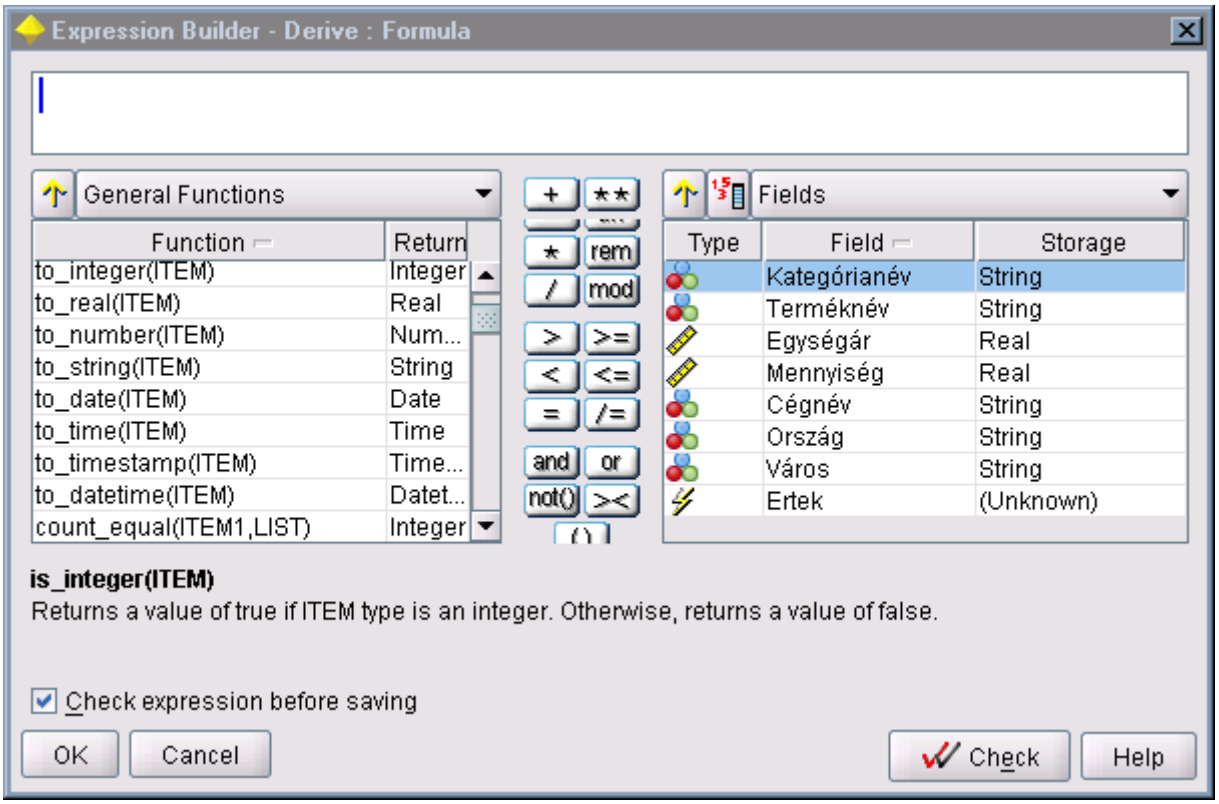


Az **IF** rovatban egy logikai kifejezést adhatunk meg, mely teljesülése esetén a **THEN** rovatot, egyébként pedig az **ELSE** rovatot értékeli ki.

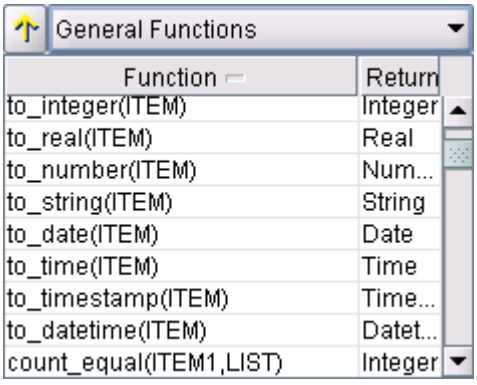
Ha a Derive as rovatban nem létezne, természetesen akkor is megoldhatók lennének a felsorolt esetek a megfelelő képletek megadásával.

Az Expression Builder

Érdemes némi figyelmet szentelnünk a Modeler beépített kifejezés-szerkesztő ablakának is. Minden egyes esetben, ahol a számológép ikon megjelenik, ott arra rákattintva megnyithatjuk ezt az ablakot:



Az ablak tetején lévő beviteli mezőben szerkeszthetjük kifejezéseinket. A kifejezések összeállításában sokat segíthet, hogy az ablak további részeiben megjelenő listák elemein duplát kattintva, vagy a gombokon kattintva az adott tételt beilleszthetjük a képletünkbe és nem kell begépelni azokat.



Az alábbi lista a felhasználható függvények listáját mutatja. A lista tetején lévő legördíthető listából konkrét függvénykategóriát választva leszűkíthetjük a választékot.

A bal oldali oszlop mutatja a függvény nevét és paramétereit, míg a jobboldali a függvény visszatérési értékének típusát adja meg.

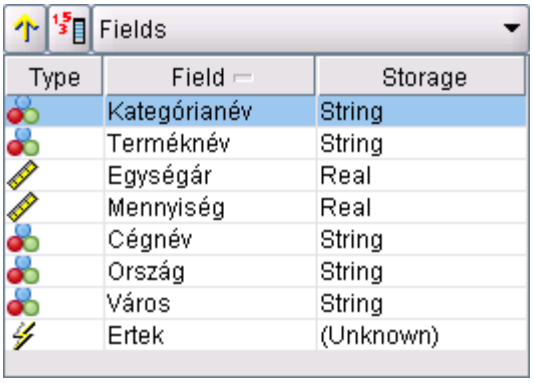
A listából kijelölt függvény leírása elolvasható az ablak alján.

A bal felső sarokban lévő sárga nyílra  kattintva a kijelölt listaelem beilleszthető a

képletünkbe.

A lista tetején kiválaszthatjuk, hogy mit mutasson a lista:

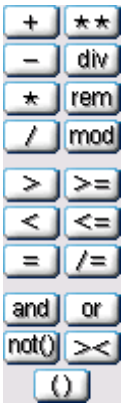
- Fields (oszlopokat)
- Parameters (paramétereket, ha vannak)
- Globals (ha vannak)



A Parameters és a Globals listában elérhető értékekről később lesz szó.

A bal felső sarokban lévő sárga nyílra  kattintva a kijelölt listaelem beilleszthető a képletünkbe.

A sárga nyíl melletti kis ikon  megmutatja a kijelölt mező konkrét értékeit, megkönnyítve a megfelelő mező kiválasztását, ha elfelejtettük az oszlop nevét.



A képleteinkben felhasználható operátorok. A kevésbé egyértelműek magyarázata:

**: hatványozás (3 ** 2 = 9)

div: egész osztás (9 div 4 = 2)

rem: az osztás maradéka (9 rem 4 = 1)

mod: ugyanaz min az előző, használata nem javasolt, mert a jövőben megszüntetik. Csakis visszafele kompatibilitás miatt van még meg

/=: a megegyezik logikai vizsgálat ellentéte: nemegyenlő

><: szövegek összefűzése (”alma” >< ”fa” = ”almafa”)

A Modeler adattípusai

A képletekben, kifejezésekben a következő típusú értékek szerepelhetnek:

- Szöveg (String)
  például: ”c1” ; ”Type 2” ; ”tetszőleges szöveg”
- Egész szám (Integer)
  például: 12 ; 0 ; -134

- Valós szám (Real Number)
  például: 12.34 ; 0.0 ; -0.0023
- Dátum-idő érték (Date/Time)
  például: 5/12/2005
- Karakterkód (Characters)
  például: `a`
- Elemek listája
  például: [1 2 3] ; ['Type 1' 'Type 2']

A karakterkódokat és listákat általában nem használjuk oszlopok értékeiként, viszont gyakran használatosak a Modeler függvények paramétereiként.

Idézőjelezési szabályok:

- Szövegek:
  Szövegeket mindig dupla idézőjelek között adunk meg.
- Karakterek:
  Mindig egyszeres visszafele álló idézőjelet kell használni. (Magyar billentyűkiosztás esetén: AltGr + 7) Például: `a`
  Karakter adattípusú értéket nyerhetünk úgy is, ha egy szövegből kiemeljük valahányadik betűjét: **"SZÖVEG"(4)**
- Mezők (oszlopok):
  A képletekben a mezők neveit általában önmagukban alkalmazhatjuk, ellenben ha a mezőnév szóközt tartalmaz, vagy speciális karakterrel kezdődik, akkor egyszeres idézőjelek közé tesszük.
- Paraméterek:
  Mindig egyszeres idézőjelek közé tesszük őket. Pl.: '$P-threshold'

A Modeler függvényei

A Modeler a következő függvénycsoportokat tartalmazza:

- Information:
  Információkat nyerhetünk ki a mezők értékeiről az itt található függvények használatával.
- Conversion:
  Adattípusok közti átalakításra használható függvényeket tartalmaz.
- Comparison:
  Mezők értékeinek egymással, vagy konstans értékekkel történő összehasonlítására használhatók a kategória függvényei.
- Logical:
  Logikai műveleteket végezhetünk el a függvényekkel.
- Numeric:
  Matematikai számításokhoz lehetnek szükségesek az itt található függvények.
- Trigonometric:
  Trigonometriai számításokhoz használhatók fel a kategória függvényei.
- Probality:
  Valószínűségi próbák számolásához használhatók fel a kategória függvényei.
- Bitwise:
  Egész és kettes számrendszerbeli értékekkel végezhetünk bitműveleteket.
- Random:

Véletlen értékek generálásához, vagy kiválasztásához használhatjuk fel.
- String:
  Szöveg típusú értékek feldolgozásakor nélkülözhetetlenek.
- SoundEx:
  Szövegek fonetikus kiejtésén alapuló hasonlóságának a vizsgálatára alkalmasak a kategória függvényei.
- Date and time:
  Dátum és időértékek manipulációjára használható függvények találhatók itt.
- Sequence:
  A táblák sorainak egymásutániságára épülő műveleteket valósítanak meg.
- Global:
  A Set Globals node-dal létrehozott globális értékek eléréséhez használhatók fel.
- Blanks and null:
  Az üres és hiányzó értékek kezeléséhez használhatjuk fel ezeket a függvényeket.
- Special fields:
  Mezők kiválasztásához használhatók fel. (Például a már használt @FIELD)

A továbbiakban kiemelnénk néhány fontosabb, gyakran használatos függvényt. A függvények részletes szintaktikája, paramétereinek listája, sorrendje megtalálható a súgóban, de a kifejezés-szerkesztő is megmutatja.

Information függvények

- @NULL(ITEM)
  Az vizsgálható vele, hogy egy adott mező értéke meg van-e adva. A Modeler az adattábla soraiban a hiányzó értékeket $null$ értékkel jelöli.
- is_date(ITEM)
  Azt vizsgálja, hogy az adott érték értelmezhető-e dátumként.
- is_integer(ITEM)
  Igaz értékkel tér vissza, ha paramétere egész szám.
- is_number(ITEM)
  Igaz értékkel tér vissza, ha paramétere értelmezhető számként.
- is_real(ITEM)
  Igaz értékkel jelzi, ha paramétere valós szám.
- is_string(ITEM)
  Igaz értéket ad vissza, ha paramétere szöveg.
- is_time(ITEM)
  Igaz lesz a visszatérési értéke, ha a paramétere értelmezhető idő értékként.

Conversion függvények

- to_integer(ITEM)
  A paramétereként megadott érték tárolási típusát egész számmá alakítja.
- to_real(ITEM)
  A paramétereként megadott érték tárolási típusát valós számmá alakítja.
- to_number(ITEM)
  A paramétereként megadott érték tárolási típusát számmá alakítja.
- to_string(ITEM)
  A paramétereként megadott érték tárolási típusát szöveggé alakítja.
- to_time(ITEM)
  A paramétereként megadott érték tárolási típusát időértékké alakítja.

- to_date(ITEM)
  A paramétereként megadott érték tárolási típusát dátumértékké alakítja.
- to_datetime(ITEM)
  A paramétereként megadott érték tárolási típusát dátumidő értékké alakítja.

Comparison függvények

- date_before(DATE1, DATE2)
  Igaz értékkel tér vissza, ha a DATE1 megelőzi a DATE2-t.
- max(ITEM1, ITEM2)
  A nagyobb elemmel tér vissza.
- min(ITEM1, ITEM2)
  A kisebb elemmel tér vissza.
- time_before(TIME1, TIME2)
  Igaz értékkel tér vissza, ha TIME1 megelőzi TIME2-t.

Logical függvények

- if COND then EXPR1 else EXPR2 endif
  Ha a COND feltétel igaz, akkor EXPR1, egyébként az EXPR2 kifejezés értékével tér vissza.

Numeric függvények

- abs(NUM)
  Abszolút értéket számol.
- fracof(NUM)
  A törtrészt adja vissza.
- intof(NUM)
  A csonkolással keletkező egészrészt adja vissza. Nem egyezik meg az egészrésszel, ha a szám negatív.
- log10(NUM)
  10-es alapú logaritmust számol.
- round(NUM)
  Egészre kerekít.
- sign(NUM)
  Előjel függvény.
- sqrt(NUM)
  Négyzetgyököt számol.

Trigonometric függvények

- sin(NUM)
- cos(NUM)
- tan(NUM)
- pi
  Igazából ez nem is függvény, hanem konstans. A pi értékét adja vissza.

Random függvények

- random(NUM)
  Egész vagy valós számmal tér vissza, egy és NUM között, attól függően, hogy a NUM paraméter milyen típusú. Ha NUM egész, akkor egésszel tér vissza, ha NUM

valós érték, akkor valós lesz a visszatérési érték is. A tizedes pontosságot a Stream tulajdonságai határozzák meg.

- random0(NUM)
  Abban különbözik az előzőtől, hogy a véletlen számok 0-tól indulnak, és a NUM értéknél kisebbek, vele egyenlők már nem lehetnek.

String függvények

- allbutfirst(N, STRING)
  A STRING szöveg minden karakterét visszaadja az első N darab karakter kivételével.
- allbutlast(N, STRING)
  A STRING szöveg minden karakterét visszaadja az utolsó N darab karakter kivételével.
- alphabefore(STRING1, STRING2)
  Igaz értékkel tér vissza, ha STRING1 megelőzi STRING2-t az abc-ben.
- endstring(LENGTH, STRING)
  A STRING szöveg utolsó N darab karakterét adja vissza.
- count_substring(STRING, SUBSTRING)
  Megadja, hogy a STRING szövegben hányszor fordul elő a SUBSTRING szöveg.
- isalphacode(CHAR)
  Igaz értékkel tér vissza, ha a paramétere betű.
- isendstring(SUBSTRING, STRING)
  Ha a STRING szöveg vége megegyezik a SUBSTRING szöveggel, akkor azt adja vissza, hogy a STRING szöveg hányadik karakterénél kezdődik a SUBSTRING szöveg, egyébként 0-val tér vissza.
- islowercode(CHAR)
  Igaz értékkel tér vissza, ha paramétere kisbetű.
- ismidstring(SUBSTRING, STRING)
  Ha a SUBSTRING szöveg szerepel a STRING szövegben, de nem a legelején vagy legvégén, akkor a kezdési pozíciójának a sorszámával tér vissza, egyébként 0-val.
- isnumbercode(CHAR)
  Igaz értékkel tér vissza, ha paramétere számjegy.
- isstartstring(SUBSTRING, STRING)
  Ha a STRING szöveg eleje megegyezik a SUBSTRING szöveggel, akkor 1 –et, egyébként 0-t ad vissza.
- issubstring(SUBSTRING, N, STRING)
  A STRING szövegben az N-edik karakterétől kezdve keresi a SUBSTRING szöveget. Ha megtalálja, akkor visszatér a kezdési pozíciójával, egyébként 0-t ad vissza. Ha N-t nem adjuk meg, akkor alapértelmezésben 1-et vesz fel étékül.
- issubstring_count(SUBSTRING, N, STRING)
  Azzal a pozícióval tér vissza, ahol a SUBSTING N-edik előfordulása kezdődik a STRING szövegben. Ha N nagyobb, mint ahányszor előfordul benne, akkor 0-val tér vissza.
- isuppercode(CHAR)
  Igaz értékkel tér vissza, ha paramétere nagybetű.
- last(STRING)
  Visszatér a STRING szöveg utolsó karakterével.
- length(STRING)

Visszatér a STRING szöveg hosszával, karakterek számában mérve.

- replace(SUBSTRING, NEWSUBSTRING, STRING)
  Visszatér azzal a szöveggel, melyet úgy kapunk, hogy a STRING szövegben a SUBSTRING szöveg összes előfordulását helyettesítjük a NEWSUBSTRING szöveggel.
- replicate(COUNT, STRING)
  A STRING szöveget annyiszor sokszorozza meg (fűzi össze önmagával), amennyit a COUNT paraméter megad.
- startstring(LENGTH, STRING)
  A STRING szöveg első N darab karakterét adja vissza.
- substring(N, LEN, STRING)
  A STRING szöveg N-edik karakterétől számolt LEN darab karaktert ad vissza.
- substring_between(N1, N2, STRING)
  A STRING szöveg N1 és N" pozíciója közötti karaktersorozatot adja vissza.
- trim(STRING)
  Eltávolítja a szöveg elejéről és végéről a köz karaktereket.
- unicode_value(CHAR)
  Visszaadja a CHAR karakter unicode értékét.
- uppertolower(CHAR) uppertolower (STRING)
  Kisbetűssé alakít.
- lowertoupper(CHAR) lowertoupper (STRING)
  Nagybetűssé alakít.

Date and time függvények

- @TODAY
  Visszaadja az aktuális dátumot.
- date_before(DATE1, DATE2)
  Igaz értékkel tér vissza, ha DATE1 megelőzi DATE2-t.
- date_days_difference(DATE1, DATE2)
  Napok számában megadja a két dátum különbségét.
- datetime_day(DATE)
  Kiemeli a hónap napjának sorszámát a dátumból. Egész számot ad vissza, 1 és 31 között.
- datetime_hour(TIME)
  Kiemeli az óra értékét az időértékből.
- datetime_minute(TIME)
  Kiemeli a perc értékét az időértékből.
- datetime_month(DATE)
  Kiemeli a hónap sorszámának értékét a dátumból.
- datetime_time(HOUR, MINUTE, SECOND)
  Időértéket állít elő az óra, a perc és a másodperc értékekből.
- datetime_weekday(DATE)
  Visszaadja, hogy a hét hányadik napjára esik az adott dátum.
- datetime_year(DATE)
  Kiemeli az év értékét a dátumból.
- time_before(TIME1, TIME2)
  Igaz értékkel tér vissza, ha TIME1 megelőzi TIME2-t.

Sequence függvények

Sok-sok művelet esetében a rekordok egymásra következősége meghatározó. Ilyen műveletek például:

- A sorok indexelése
- Idősorok
- Mozgóátlagok
- Egymás utáni értékek összehasonlítása
- Stb.

Sok művelet esetében egy sor feldolgozása nem független művelet, hanem nagyon is függ a mellette, előtte, vagy mögötte található rekordok értékeitől. Ezekben az esetekben nagyon fontos odafigyelni a rekordok megfelelő szempont szerinti sorba rendezésére.

A Sequesnce és speciális függvények mindig az alábbi szintaktikát követik:

- @ karakterrel kezdődnek
- A nevük csupa nagybetűvel van írva

Például:

Ha szeretnénk megtudni, hogy mennyi idő telt el egy esemény előfordulása óta, vagy mikor volt utoljára igaz egy feltétel, akkor használhatjuk a @SIENCE függvényt:

@SINCE(INCOME > OUTGOINGS)

Ez a képlet visszatér annak az utolsó sornak az eltolási értékével, amikor a feltétel még igaz volt.

További részletek a súgóban.

Feladatok

1. feladat

Számolja ki, hogy az egyes termékekből összesen milyen értékben adtak el. A listát azzal a termékkel kezdje, amelyből a legnagyobb értékben történt eladás.

2. feladat

Az előző megoldást egészítse ki azzal, hogy az egyes termékekre az összes eladott darabszámot is kiszámolja, valamint az egyes eladási tételekben szereplő legolcsóbb és legdrágább egységárat. Az összesített értékekből számol átlagos egységárat!

3. feladat

Olvassa be az ADATOK.TXT állományt!

A születési dátumok alapján határozza meg minden egyes évre, hogy az adott évben hányan születtek! A listát az évszámok alapján növekvő rendezésben jelenítse meg!

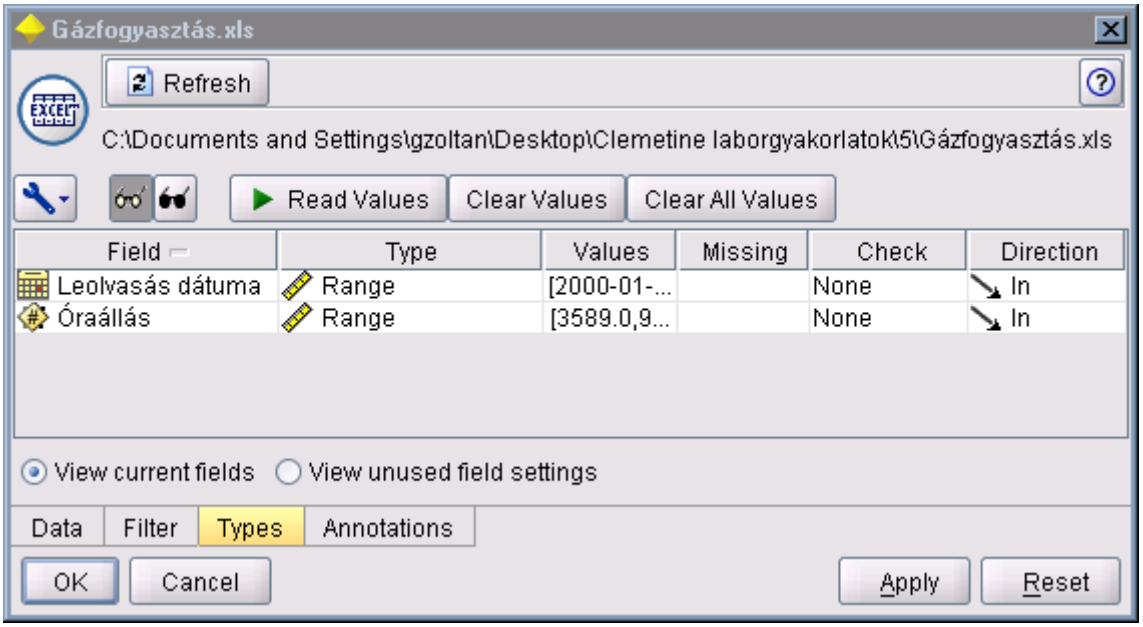
5. Rekord és mező műveletek, vizualizáció

Az alábbi gyakorlat célja, hogy egy összetettebb példán keresztül mélyebbre vezessen bennünket a stream-ek építésében, a képletek alkalmazásában.

A problémakör amit végigjárunk, az egy fiktív háztartás két év alatt összeírt gázfogyasztási adatainak az elemzése. Tegyük fel, hogy XY két éven keresztül minden reggel leolvasta és feljegyezte a gázóra állását.

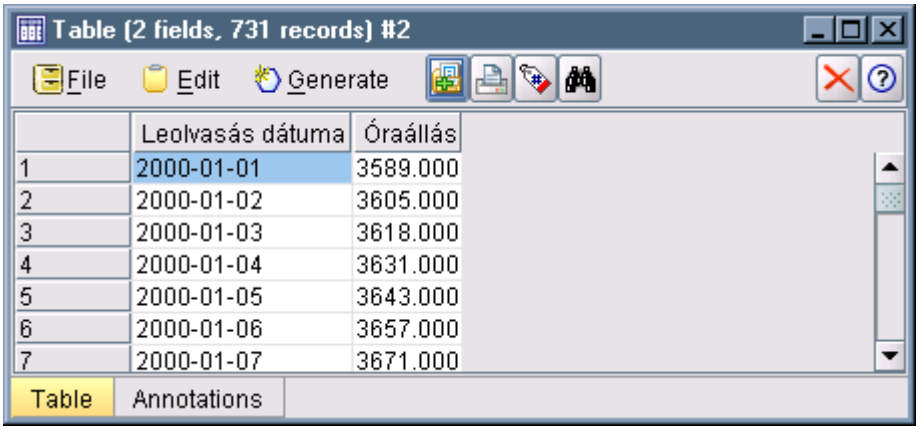
Olvassuk be az adatokat a GÁZFOGYASZTÁS.XLS állományból!

Jelenítse meg a beolvasott adatokat egy Table node-dal!



Figyelje meg, hogy a beolvasott mezők típusa: Range

Az adattábla:



Amint az látható az adattábla a gázóra napi állásait tartalmazza. Nekünk alapvetően a napi fogyasztási értékekre lenne szükségünk, amelyet úgy kaphatunk meg, hogy az

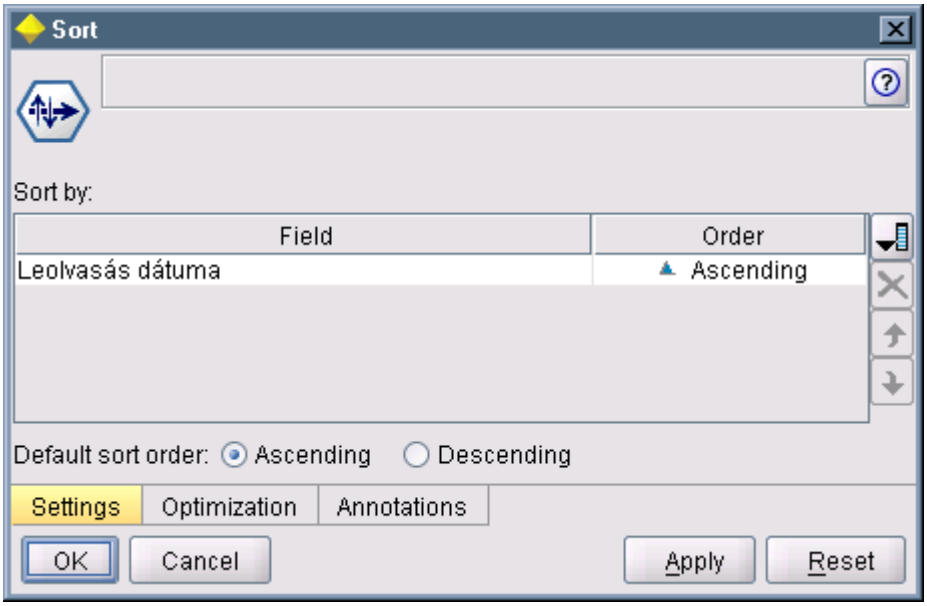
aktuális napi óraállásból kivonjuk az előző napi óraállás értékét. Természetesen így a legelső napra nem lesz érvényes fogyasztási adatunk!

Hozzon létre egy új számított mezőt NAPIFOGYASZTÁS néven (Derive node-dal), mely a napi fogyasztási adatokat tartalmazza!

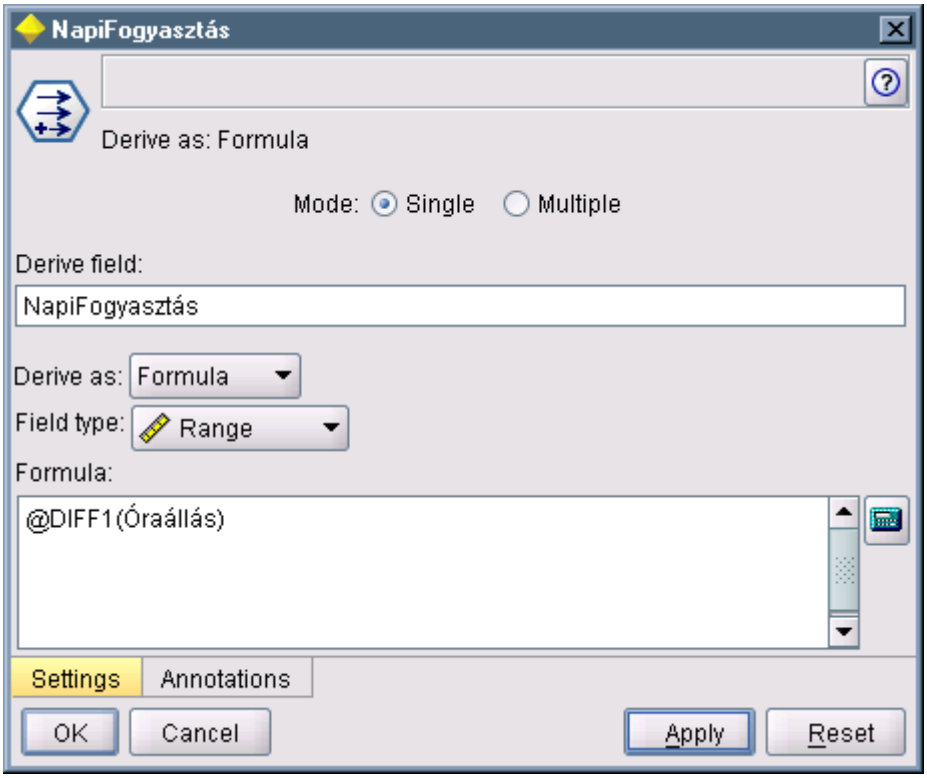
A feladat megoldásához valamely Sequence függvénykategóriába eső függvény segíthet, hiszen olyan műveletet kell megvalósítanunk, amely a tábla sorainak egymásutániságára épül. A jó eredmény biztosításához szükséges, hogy a tábla sorai teljes bizonyossággal a Leolvasás dátuma oszlop szerint legyenek rendezve, növekvő sorrendbe.

Tehát még a Derive node elé fel kell használnunk egy Sort node-ot is:

Ahol a Sort node beállításai a következők:

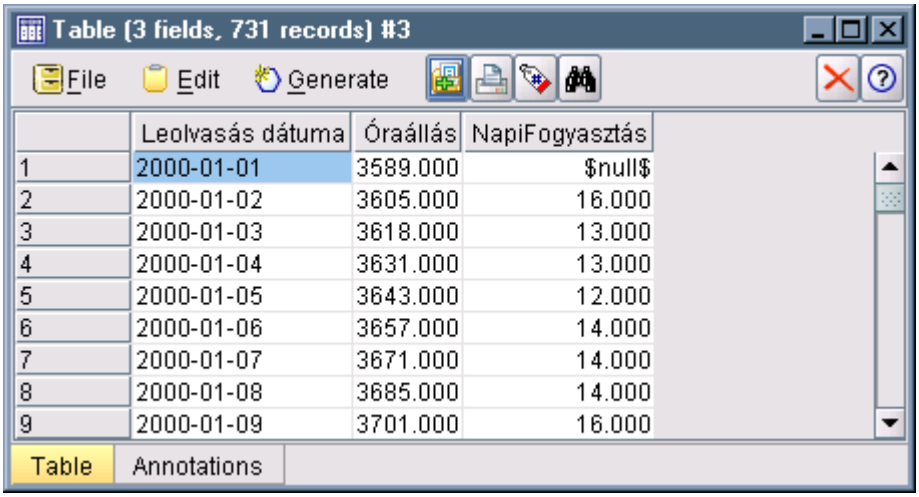


A Derive node beállításai pedig:



A @DIFF1(Óraállás) képlet helyett használhattuk volna a következő képletet is: Óraállás-@OFFSET(Óraállás,1)

Az eredmények:

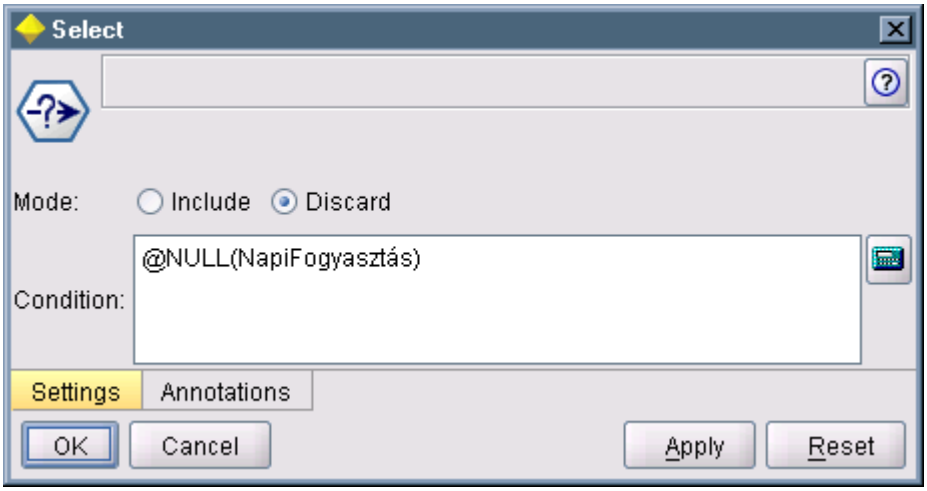


	Leolvasás dátuma	Óraállás	NapiFogyasztás
1	2000-01-01	3589.000	$null$
2	2000-01-02	3605.000	16.000
3	2000-01-03	3618.000	13.000
4	2000-01-04	3631.000	13.000
5	2000-01-05	3643.000	12.000
6	2000-01-06	3657.000	14.000
7	2000-01-07	3671.000	14.000
8	2000-01-08	3685.000	14.000
9	2000-01-09	3701.000	16.000

Amint az látható az első sor esetében $null$ lett a NapiFogyasztás mező értéke.

Zárja ki a további feldolgozásból az első sort!

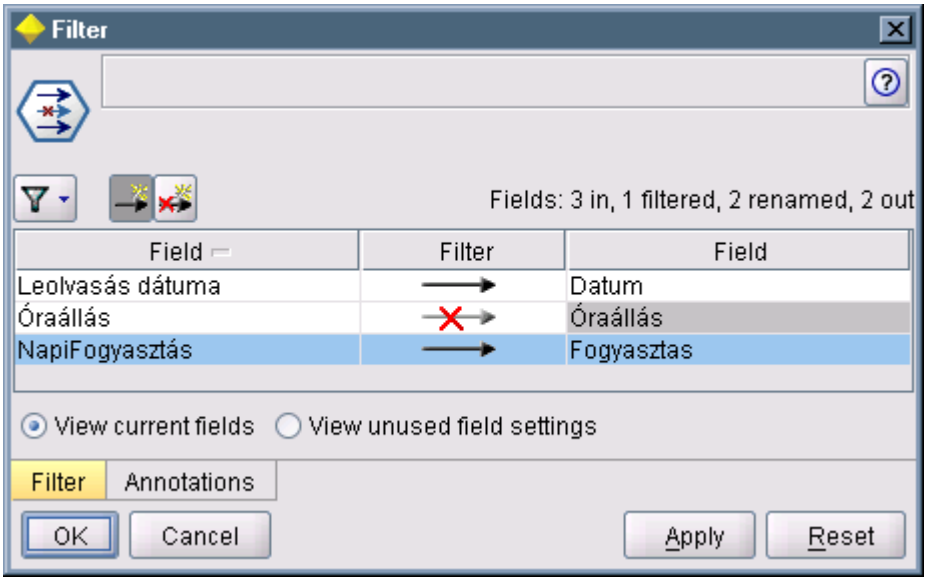
Ezt egy Select node alkalmazásával könnyen megtehetjük:



A további feldolgozáshoz már nem lesz szükségünk az óraállásokat tartalmazó oszlop értékeire, csak a Leolvasás dátumára és a Napi fogyasztásra.

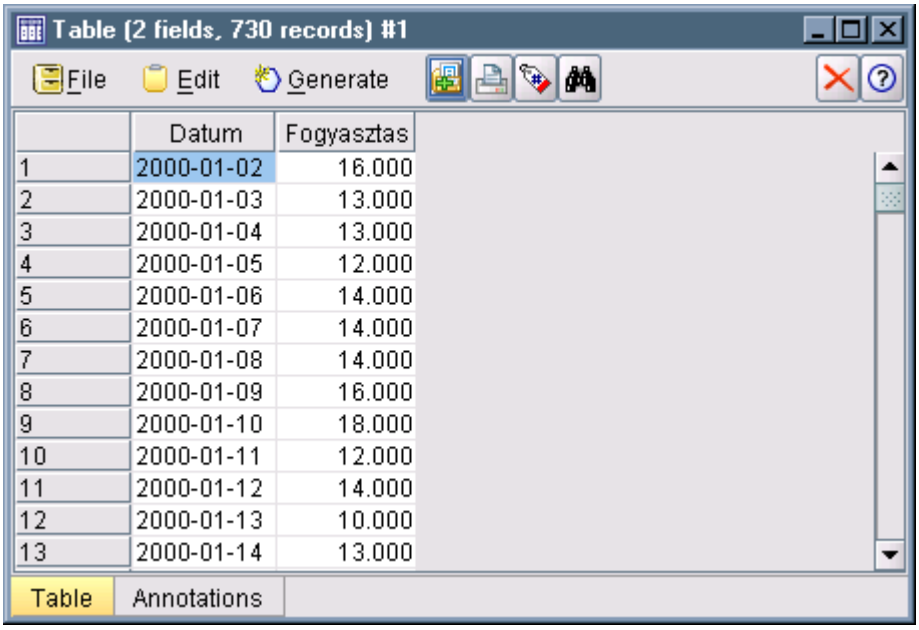
Zárja ki a további műveletekből az Óraállás oszlopot, a másik kettőt pedig nevezze át Datum-ra és Fogyasztas-ra!

Használjuk a Filter Node egy példányát erre a feladatra:



Az eddig elkészült stream:

Ha egy Table node-dal megtekintjük az eddigi eredményeket:

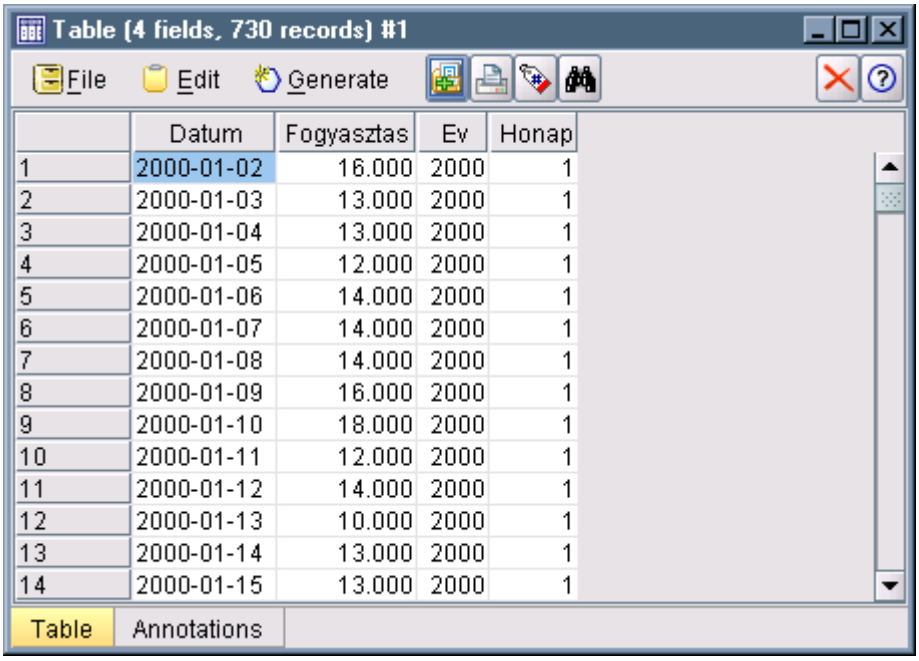


	Datum	Fogyasztas
1	2000-01-02	16.000
2	2000-01-03	13.000
3	2000-01-04	13.000
4	2000-01-05	12.000
5	2000-01-06	14.000
6	2000-01-07	14.000
7	2000-01-08	14.000
8	2000-01-09	16.000
9	2000-01-10	18.000
10	2000-01-11	12.000
11	2000-01-12	14.000
12	2000-01-13	10.000
13	2000-01-14	13.000

A továbbiakban készítsük el a havi összegzéseket. Vagyis a két év összesen 24 hónapjára számoljuk ki az összes fogyasztás értékét. Ehhez csoportosítanunk kell a tábla sorait úgy, hogy egy csoportba az azonos év azonos hónapjára eső sorok kerüljenek.

Hozzon létre két új számított mezőt (Derive node-dal) Ev és Honap néven, ahol az egyik fogyasztási dátum évét, a másik a hónapját tartalmazza!

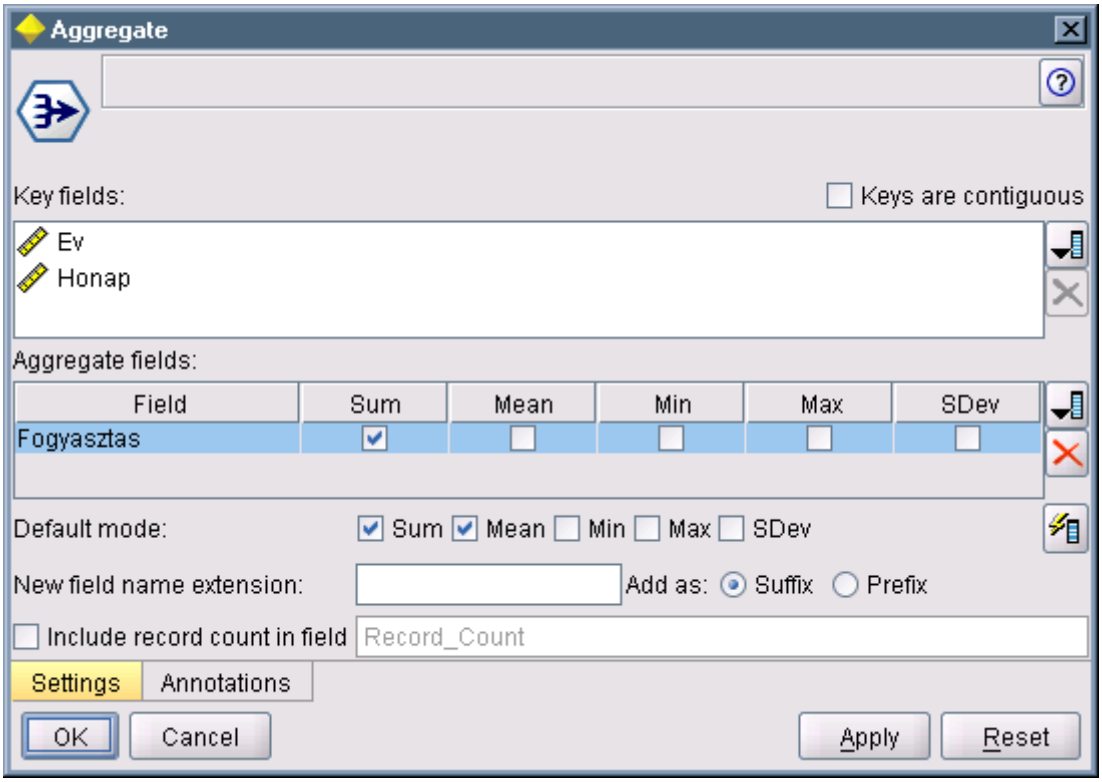
A képletekben a datetime_year és a datetime_mont függvényeket használjuk fel. A futtatás után a következő képen is látható Ev és Honap oszloppal gyarapodtunk:



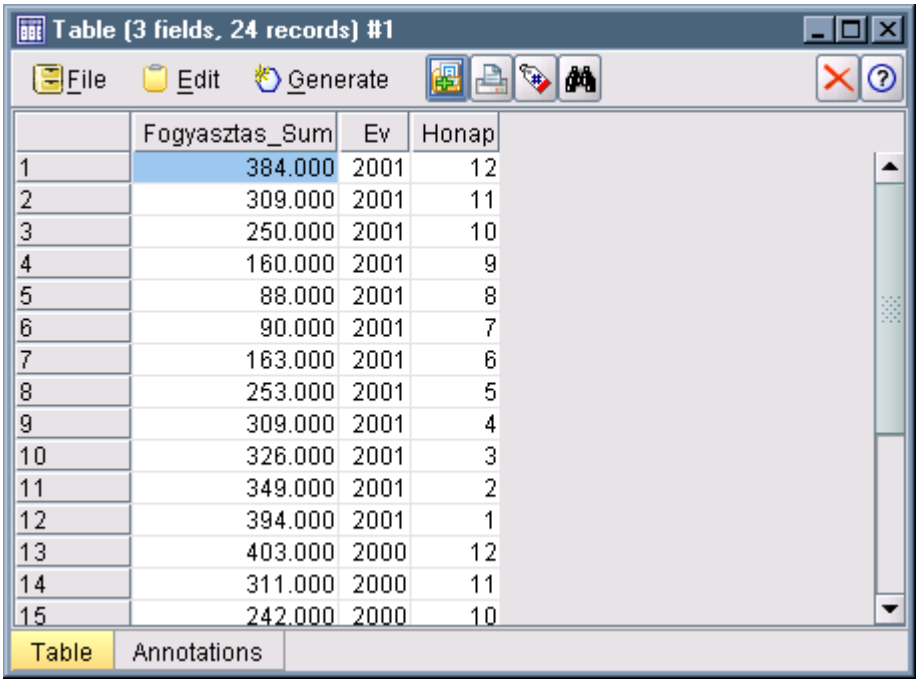
	Datum	Fogyasztas	Ev	Honap
1	2000-01-02	16.000	2000	1
2	2000-01-03	13.000	2000	1
3	2000-01-04	13.000	2000	1
4	2000-01-05	12.000	2000	1
5	2000-01-06	14.000	2000	1
6	2000-01-07	14.000	2000	1
7	2000-01-08	14.000	2000	1
8	2000-01-09	16.000	2000	1
9	2000-01-10	18.000	2000	1
10	2000-01-11	12.000	2000	1
11	2000-01-12	14.000	2000	1
12	2000-01-13	10.000	2000	1
13	2000-01-14	13.000	2000	1
14	2000-01-15	13.000	2000	1

Számolja ki a havi összfogyasztásokat! (Aggregate node)

Az Aggregate node beállításai:



És futásának eredménye:



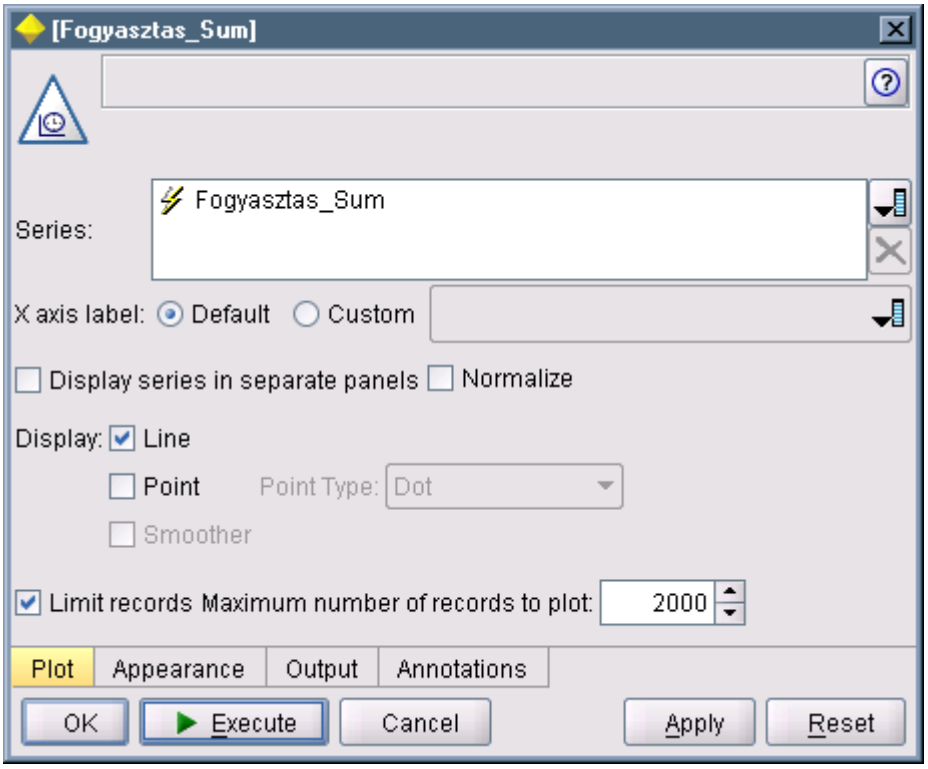
Idősorok ábrázolása

Ábrázoljuk grafikonon a kapott havi fogyasztási adatokat!

Kapcsoljon a stream végére egy TIME PLOT 

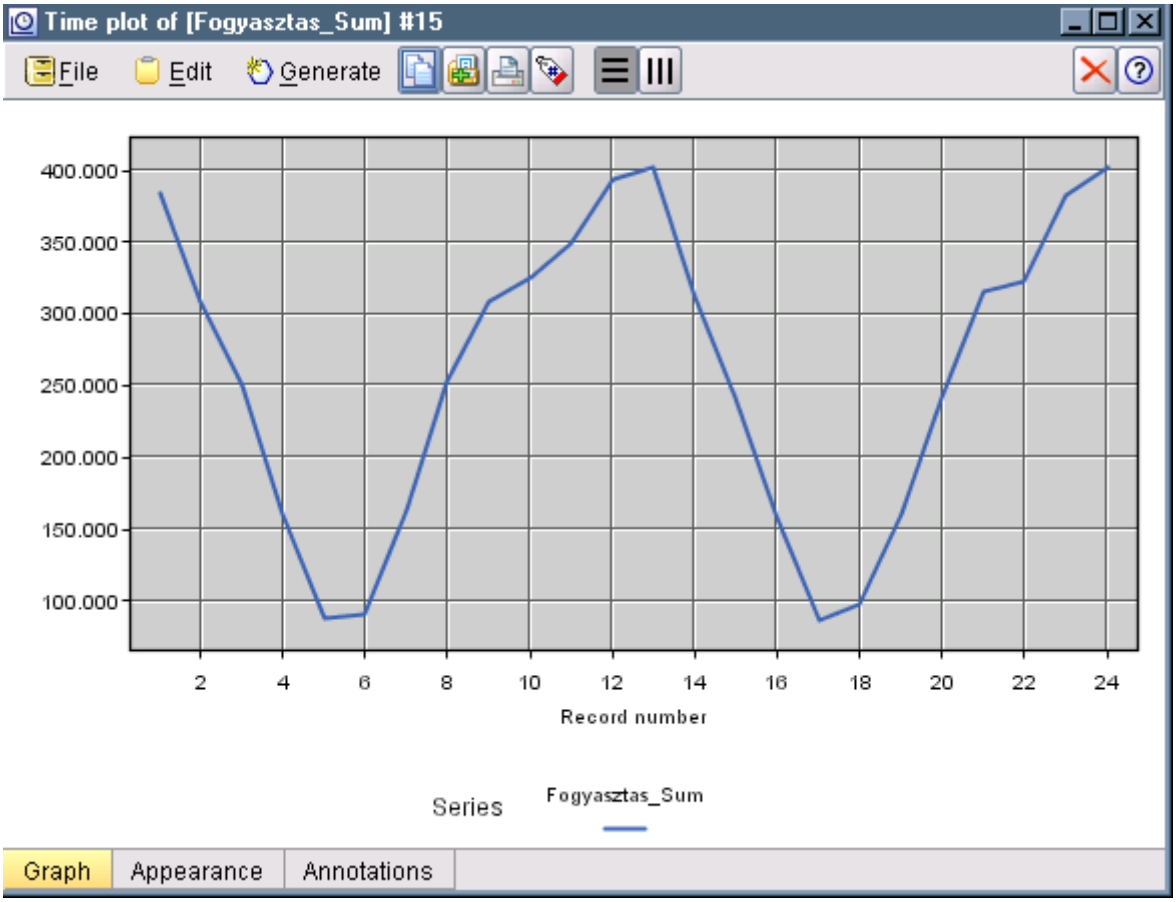
node-ot, a Graphs palettáról!

A node beállításai:



- A Time plot node a **SERIES** rovatban megadott adatsorokat olyan sorrendben fogja ábrázolni az X tengelyen, ahogy azok az adattáblában elhelyezkednek.
- Az X tengely feliratait **(X AXIS LABEL)** alapértelmezésben **(DEFAULT)** besorszámozza. 1-től addig, ahány ábrázolandó adat van. A **CUSTOM** opciót választva megadhatunk egy másik mezőt, melynek értékei kerülnek az X tengely feliratairól megjelenítésre.
- A **DISPLAY** rovatban bekapcsolhatjuk, hogy vonallal, ponttal (alakja megadható) vagy mindkettővel szeretnénk megjeleníteni az ábrázolt adatokat.
- Ha a **NORMALIZE** rovat be van pipálva, akkor az adatokat a 0-1 tartományra méretezi. Bekapcsolása segít több különböző nagyságrendű adatsor esetén az adatsorok közötti összefüggések felderítésében. Ha csak egy adatsorunk van, vagy az adatsorok azonos nagyságrendű értékeket tartalmaznak, akkor felesleges a bekapcsolása.
- A **DISPLAY SERIES IN SEPARATE PANELS** opció bekapcsolása esetén minden egyes adatsort külön grafikonon ábrázol, míg kikapcsolt állapotában az adatsorok közös grafikonon vannak ábrázolva, és színekkel vannak megkülönböztetve.
- A **SMOOTHER** opció csak a külön panelok bekapcsolása esetén érhető el. Egy lágyított átlag görbét illeszt az adatsorokra.

A futtatás eredménye:



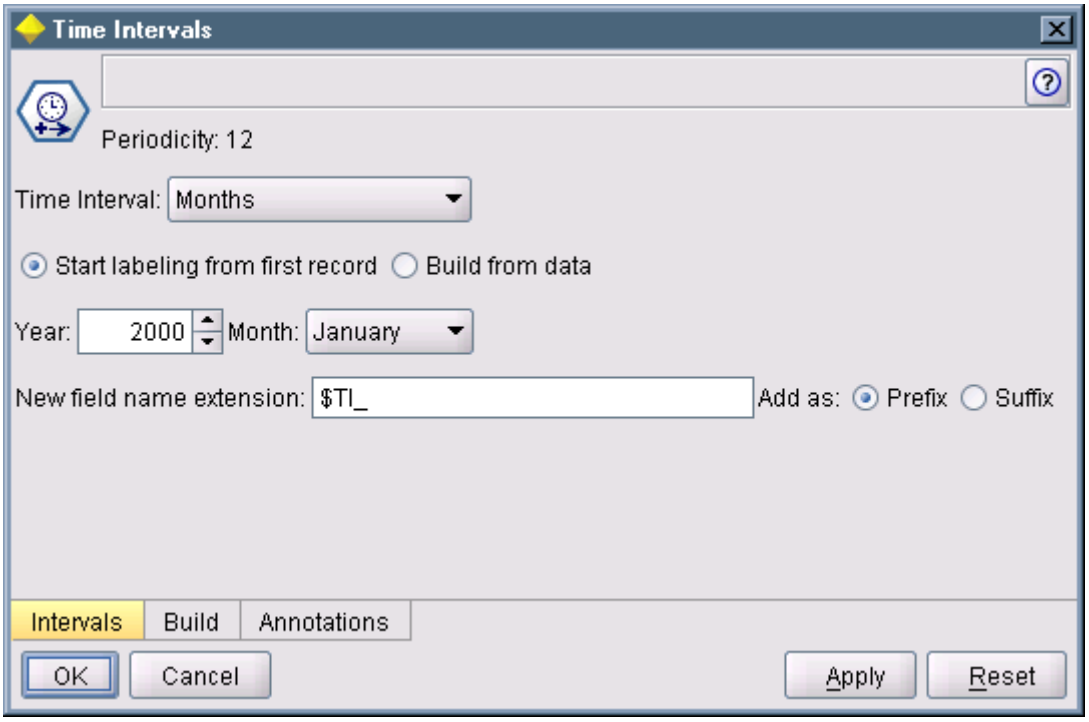
Foglalkozzunk kicsit az X tengely feliratával. Egy alternatíva lehetne, ha a meglévő **EV** és **HONAP** oszlopok kombinálásával létrehoznánk egy új számított oszlopot, és annak értékeit jelenítenénk meg feliratokként.

Egy sokkal jobb megoldás viszont, a Time **INTERVALS NODE** 

használata.

Szúrjunk be egy TIME INTERVALS **node-ot még a** TIME PLOT **node elé!**

A node beállításai:



Amint az a képen látható a node-ot úgy állítottuk be, hogy az első rekordtól kezdje **(START LABELLING FROM FIRST RECORD)** az értékek feltöltését, hónapokban lépkedjen **(TIME INTERVAL** rovat) és 2000 januárja legyen az első érték.

A Time Interval node elé érdemes beszúrni egy Sort node-ot, hogy a sorok felcímkézése garantáltan jó sorrendben történjen!

A node futtatásának eredménye:

Table (7 fields, 24 records) #3

File Edit Generate

	Fogyasztas_Sum	Ev	Honap	$TI_TimeIndex	$TI_TimeLabel	$TI_Year	$TI_Month
1	402.000	2000	1	1	jan. 2000	2000	1
2	383.000	2000	2	2	febr. 2000	2000	2
3	323.000	2000	3	3	márc. 2000	2000	3
4	316.000	2000	4	4	ápr. 2000	2000	4
5	241.000	2000	5	5	máj. 2000	2000	5
6	160.000	2000	6	6	jún. 2000	2000	6
7	97.000	2000	7	7	júl. 2000	2000	7
8	86.000	2000	8	8	aug. 2000	2000	8
9	158.000	2000	9	9	szept. 2000	2000	9
10	242.000	2000	10	10	okt. 2000	2000	10
11	311.000	2000	11	11	nov. 2000	2000	11
12	403.000	2000	12	12	dec. 2000	2000	12
13	394.000	2001	1	13	jan. 2001	2001	1
14	349.000	2001	2	14	febr. 2001	2001	2
15	326.000	2001	3	15	márc. 2001	2001	3
16	309.000	2001	4	16	ápr. 2001	2001	4
17	253.000	2001	5	17	máj. 2001	2001	5
18	163.000	2001	6	18	jún. 2001	2001	6
19	90.000	2001	7	19	júl. 2001	2001	7
20	88.000	2001	8	20	aug. 2001	2001	8

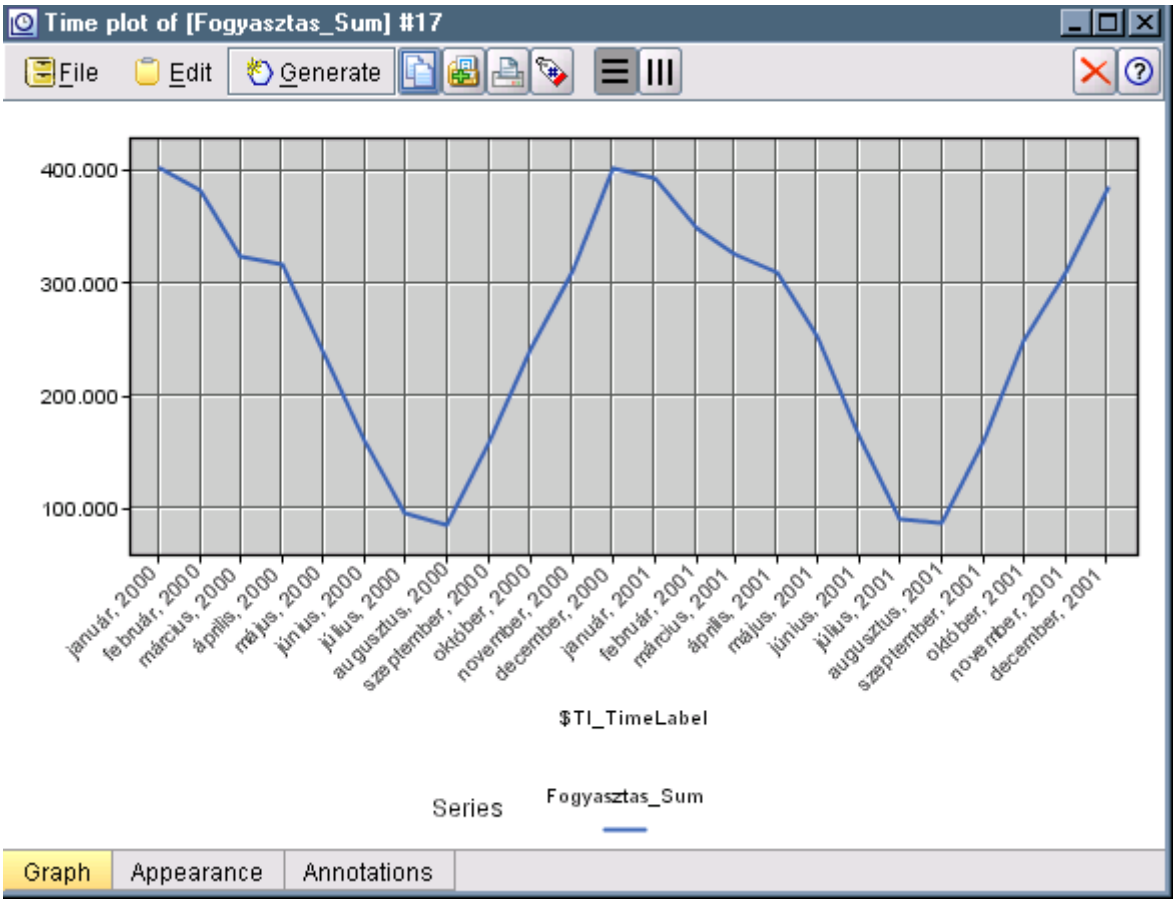
Table Annotations

Amint az látható a Time Interval node három új oszlopot hozott létre.

---

*Az oszlopok neveinek az előtagja a node beállításakor szabadon módosítható (**NEW FIELD NAME EXTENSION**), bár ha nem szükséges, akkor ne tegyük, mert a Time Plot node alapértelmezésben felhasználja ezeket az oszlopokat!*

Futtassuk most a Time Plot node-ot:



További ábrázolások

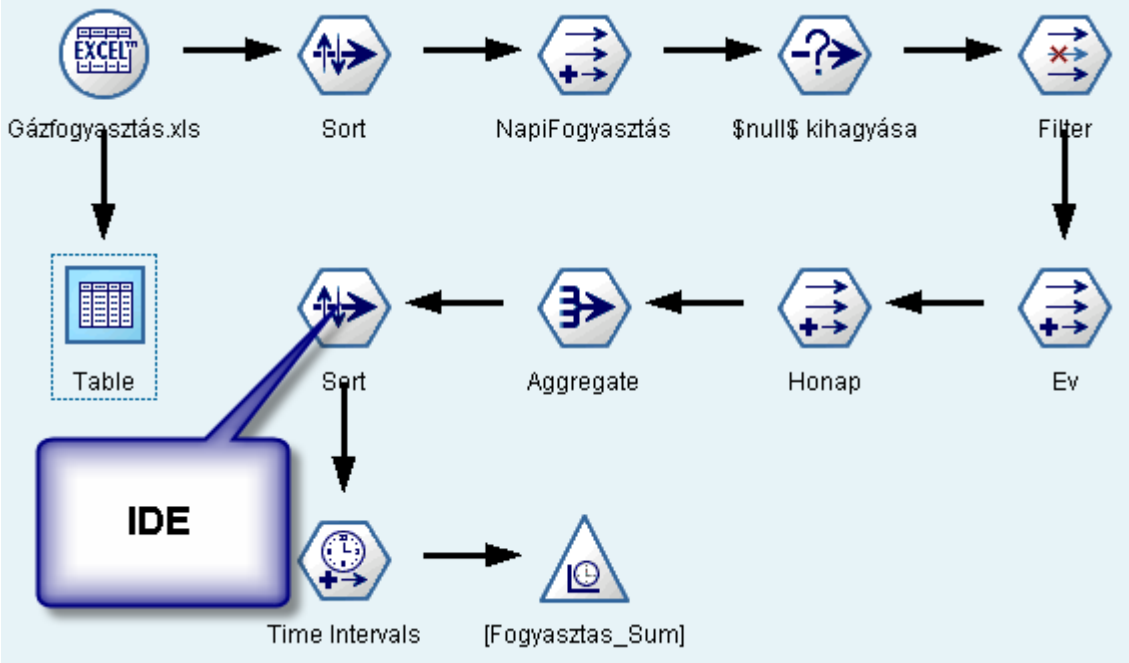
A két fogyasztási év adatait sokkal jobban össze tudnánk hasonlítani, ha egy grafikonban nem egymás mögött, hanem egymás tetején, mint két külön adatsort ábrázolhatnánk.

A probléma megoldását alapvetően két oldalról is megközelíthetjük:

- 24 soros adattáblánkból készítsünk 12 sorosat, ahol viszont külön oszlopok tartalmazzák a 2000-es és 2001-es évi fogyasztási adatokat. Ez az @OFFSET függvénnyel megoldható lenne…
- Használjuk a **PLOT** node-ot a Graphs palettáról.

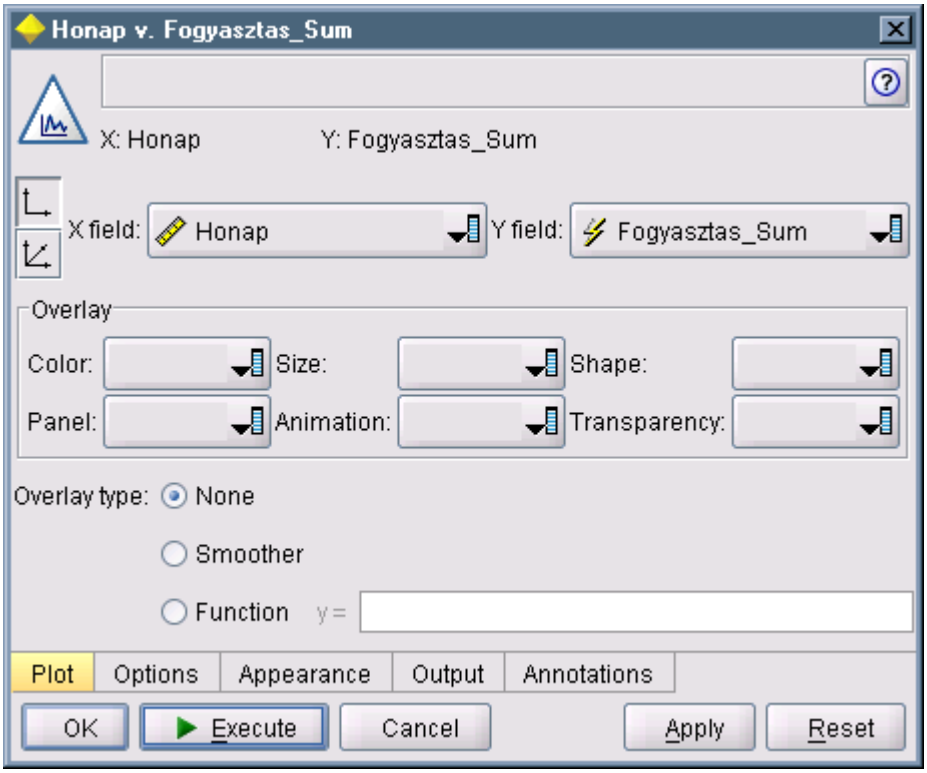
Jelen esetben az utóbbi mellett döntünk!

Stream-ünk eddig az alábbiak szerint néz ki:



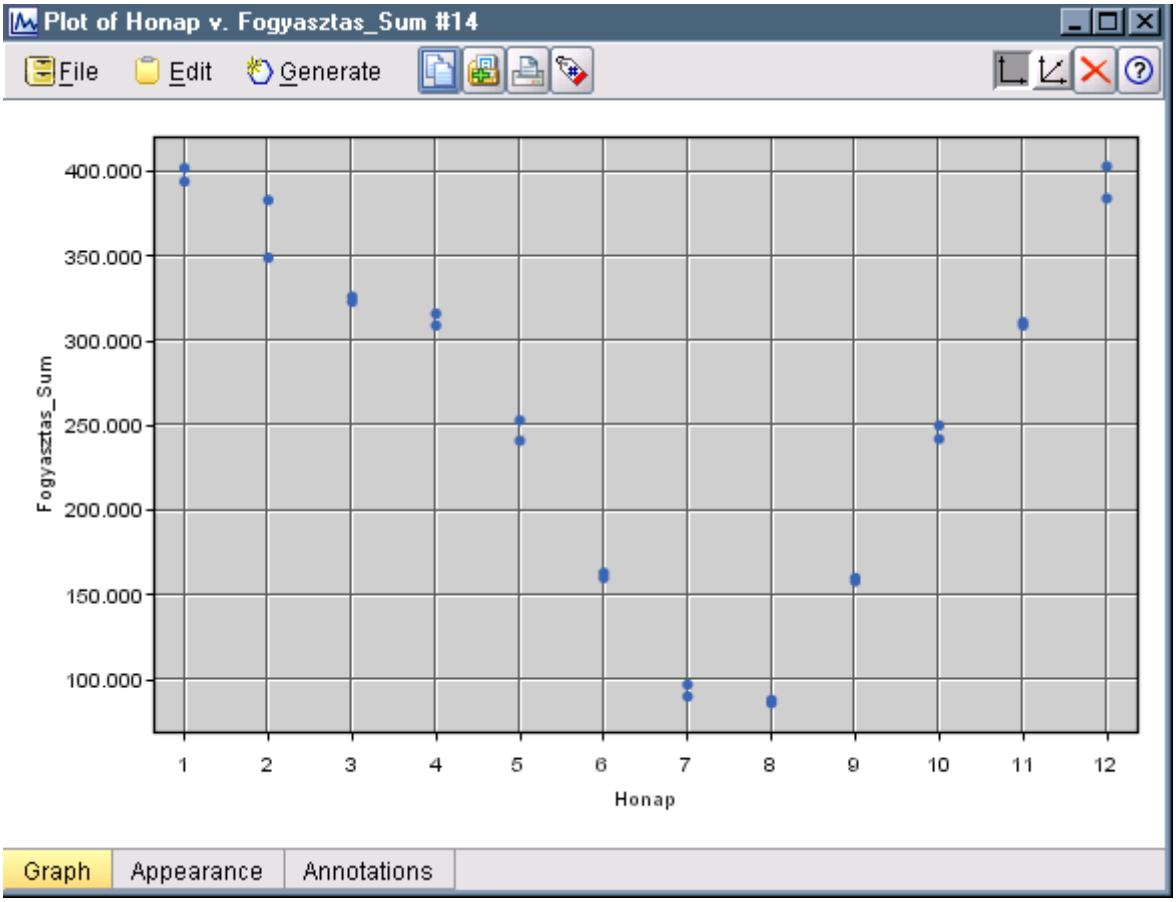
Kapcsoljunk egy PLOT **node-ot** 

az ábrán jelzett Sort node után!



Amint az a képen is látható, az X tengelyen a Honap mező értékeit, míg az Y tengelyen a havi összes fogyasztásokat ábrázoljuk.

A node futtatásának eredménye:

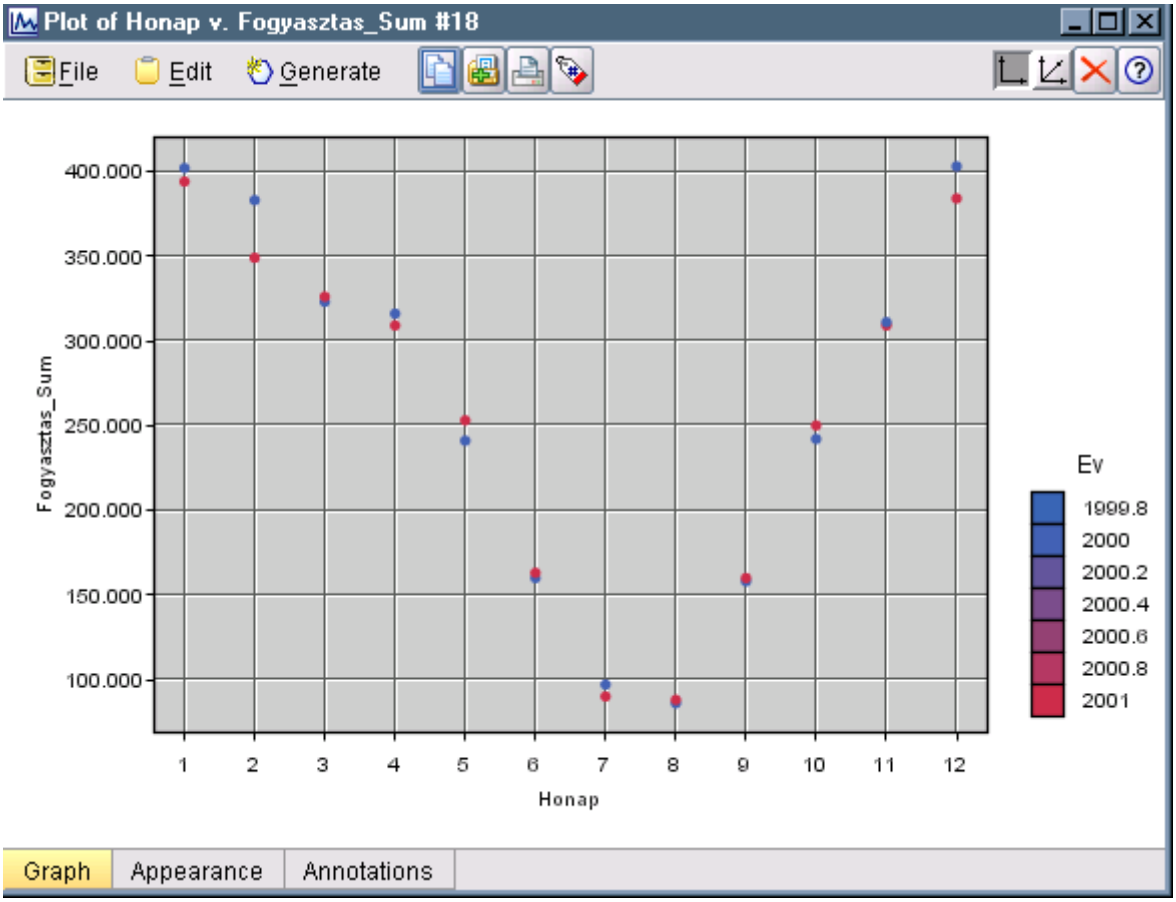


Mint látható minden egyes hónaphoz két adatpont van ábrázolva. Az egyik a 2000-es, a másik a 2001-es fogyasztási érték.

A node beállításainál az **OVERLAY** rovatban különféle vizuális módokon csoportosíthatjuk adatainkat.

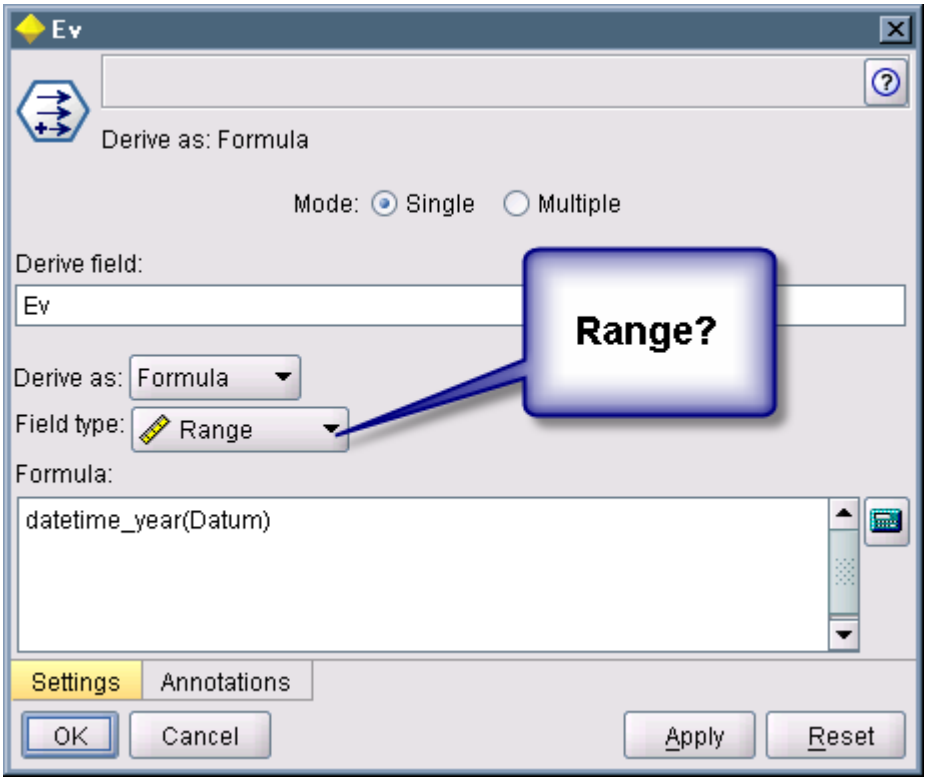
Állítsuk be a Color opciónál az évet!

Eredmény:

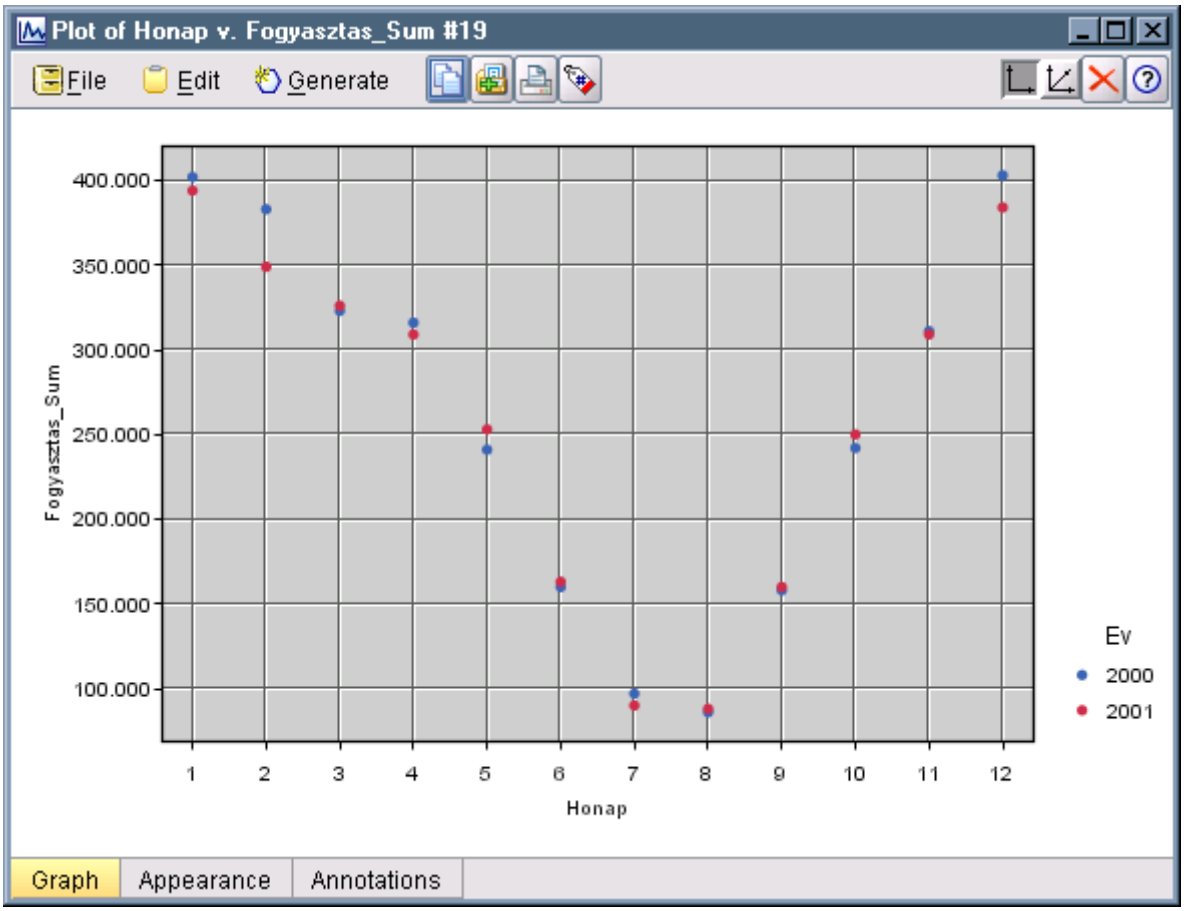


Az ábra magáért beszél.

Meg kell azonban magyaráznunk, hogy miért is olyan furcsa a jelmagyarázat! Emlékezzünk csak vissza arra a Derive node-ra, ahol az Ev mezőt számoltuk ki:

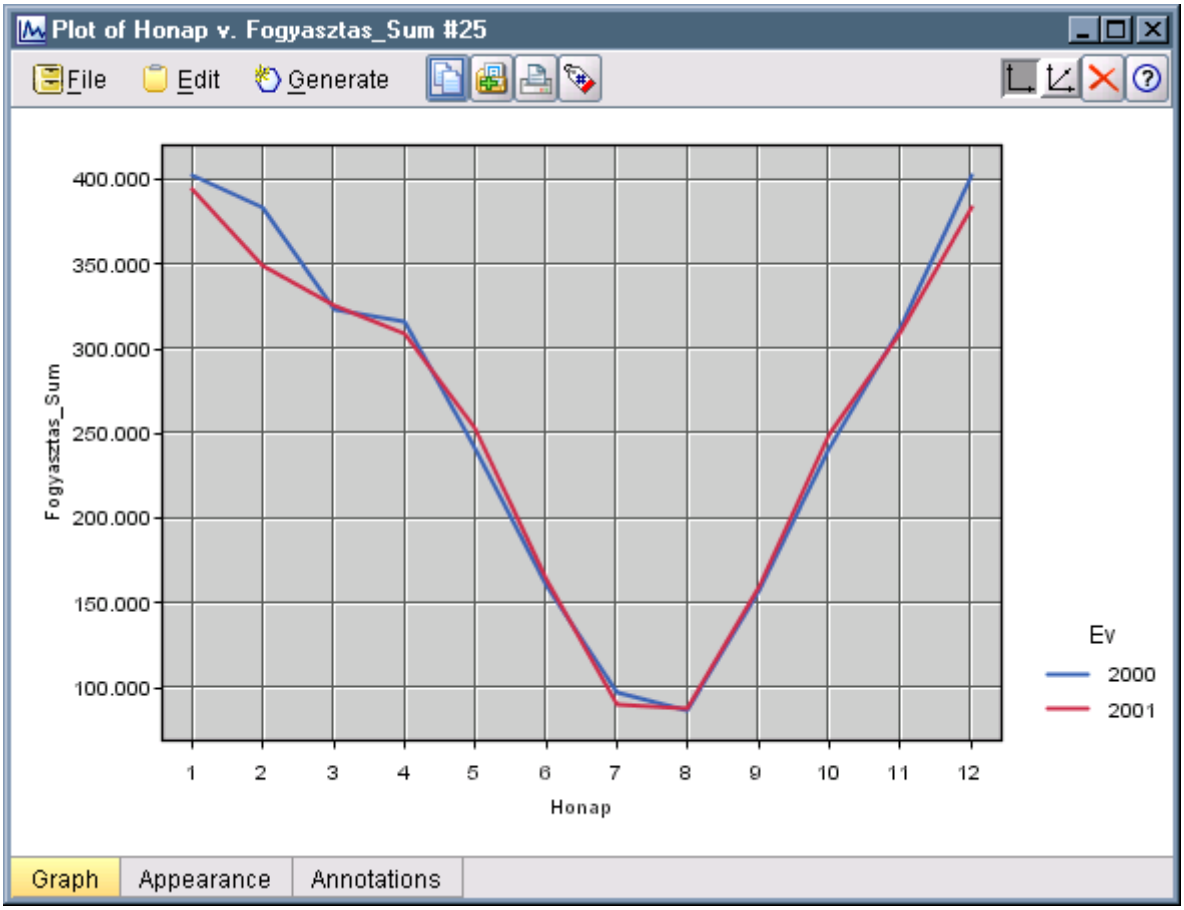


Az EV mező típusát állítsuk át ORDERED SET-re, és futtassuk újra a Plot node-ot!

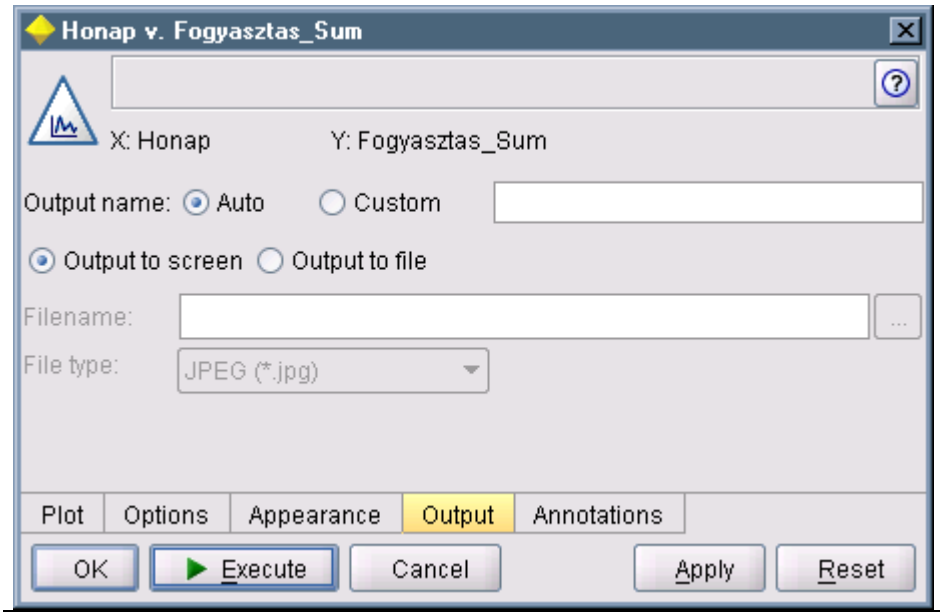


Próbáljuk ki az OVERLAY rovat többi opcióját is!

A node beállításainál az **OPTIONS** fülön például gondoskodhatunk arról is, hogy az adatpontokat kösse össze:



Bármely Graph node esetén a futtatás eredményét automatikusan el is mentheti:



Feladatok

1. feladat:

Van-e az adatokban heti periodicitás? A hipotézis az, hogy hétvégén többet fogyasztanak, mint hétköznapokon, hisz akkor egész nap otthon a család. Ellenőrizze, és igazolja állítását!

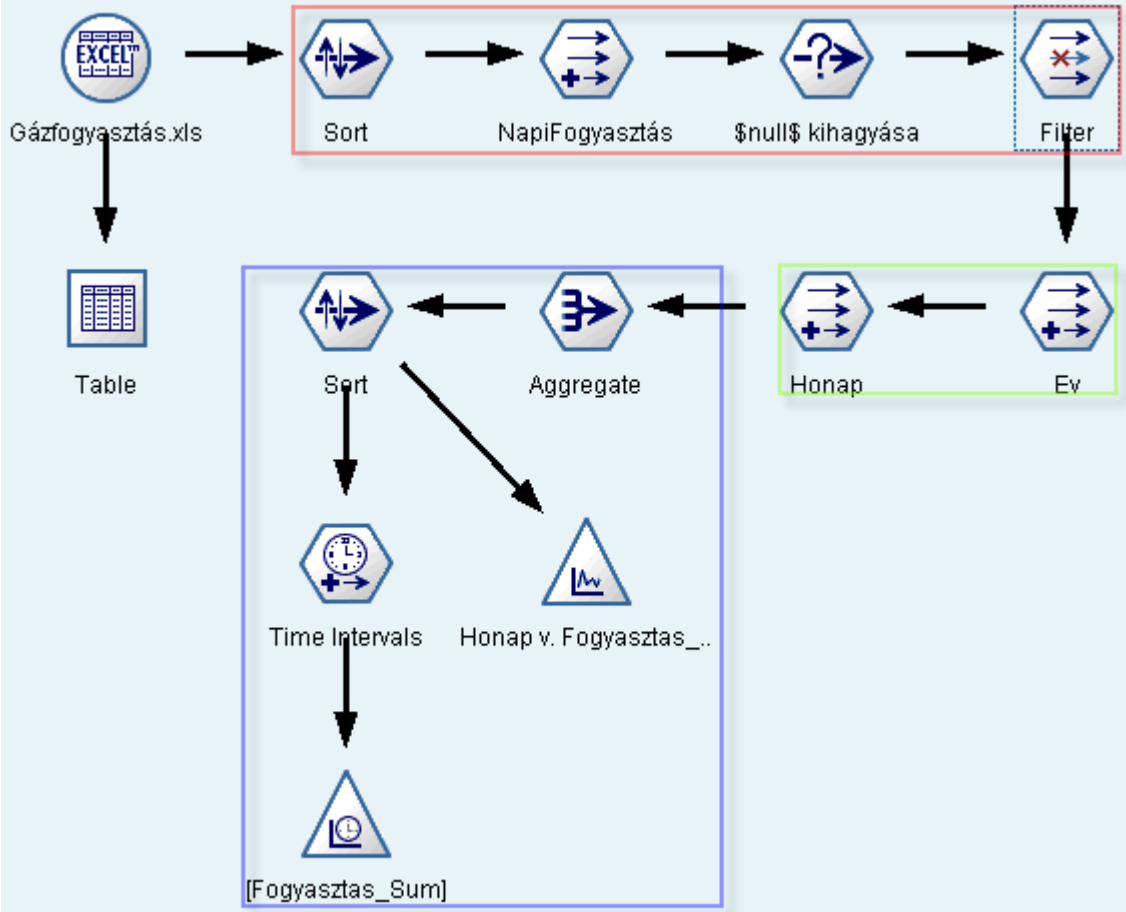
2. feladat

Számolja ki minden egyes hónapra, hogy az éves összes fogyasztás hány százalékát használta el az adott hónapban!

6. Adatforrások egyesítése

A gyakorlat folytatja az előző gyakorlatban megkezdett munkát. Felhasználjuk a már elkészült stream egy részét, és azt kiegészítjük további funkcionalitásokkal.

Az előző gyakorlat témája a gázfogyasztás alakulásának vizsgálata volt. Nyissuk meg a stream-et:



Mit is jelölnek a színes téglalapok?

- A piros színnel jelölt node-ok alapvetően adatelőkészítési műveleteket végeznek. Az adatforrás óraállás oszlopának értékeiből napi fogyasztási adatokat állítanak elő.
- A zöld színnel jelölt node-ok létrehozzák az Év és Hónap számított oszlopokat, melyekre épülnek a havi illetve éves lebontású statisztikák.
- A kék színnel jelölt node-ok végzik el a számításokat, és jelenítik meg azok eredményeit.

Még mielőtt a stream-et tovább építenénk, bővítenénk újabb node-okkal, ismerjük meg egy olyan funkcióját a Modeler-nek, mely segítségével a stream-jeinket átláthatóbbá, strukturáltabbá tehetjük.

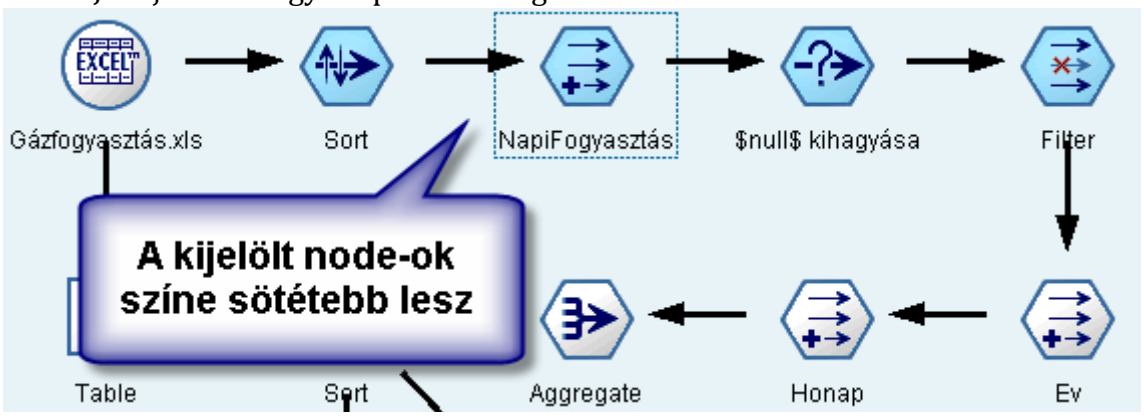
SuperNode 

A SuperNode-ot nem találjuk meg egyik node palettán sem. Igazából nem is végez semmilyen műveletet. Szerepe csupán annyi, hogy más node-okat foglal magába, azokat elrejtve, így átláthatóbbá és könnyebben kezelhetőbbé téve a stream-et.

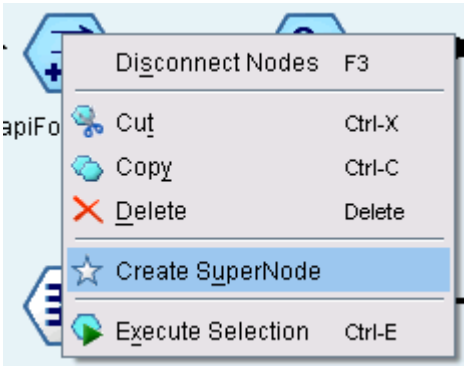
Természetesen célszerű olyan node-okat foglalni egy csoportba és elrejteni egy SuperNode mögé, amelyek meglétének közös értelme, szerepe van. Mint ahogy a képen látható színes téglalapok is ilyen node csoportokat jelölnek.

A SuperNode készítésének lépései:

- Jelöljük ki az egy csoportba befoglalni kívánt node-okat.

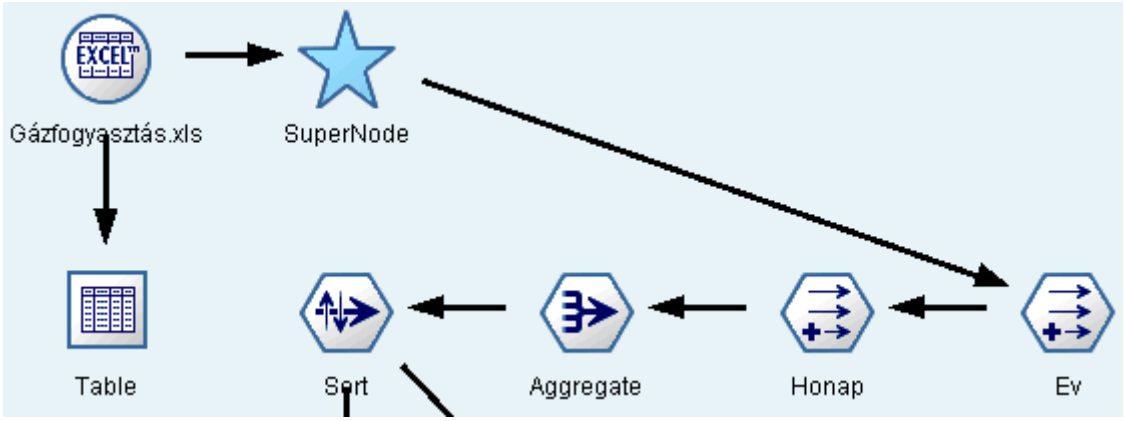


- Kattintsunk a jobb egérgombbal bármelyik kijelölt node-on, és a megjelenő helyi menüből válasszuk a **CREATE SUPERNODE** parancsot.

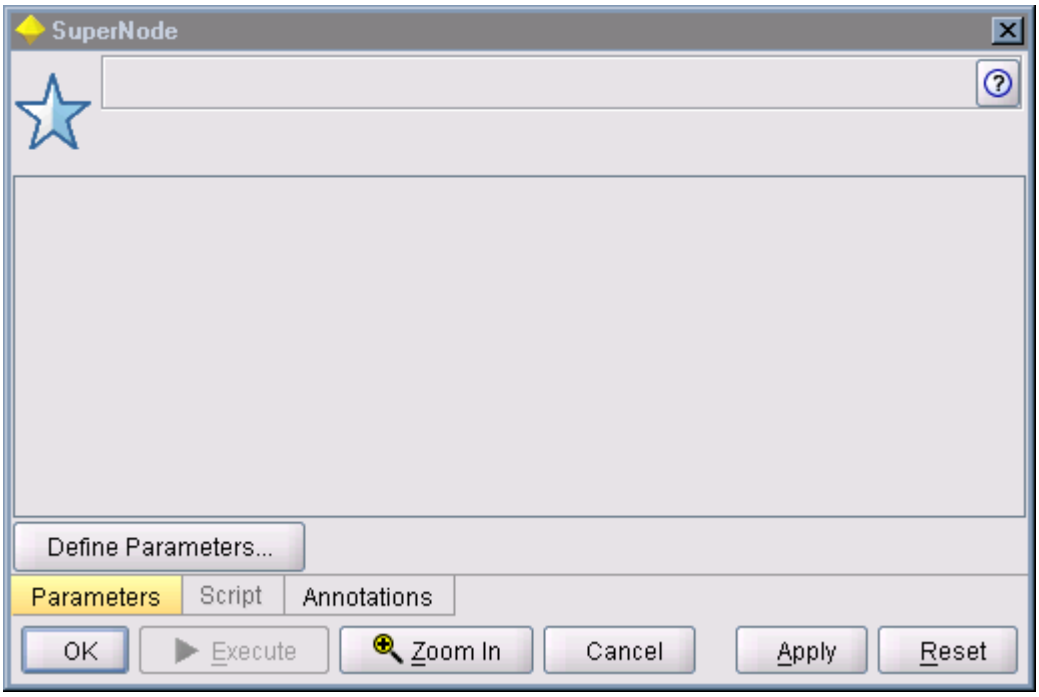


De használhatjuk a **SUPERNODE MENÜPONT CREATE SUPERNODE / FROM SELECTION...** parancsát is.

A parancs kiadása után a következő eredményt látjuk:



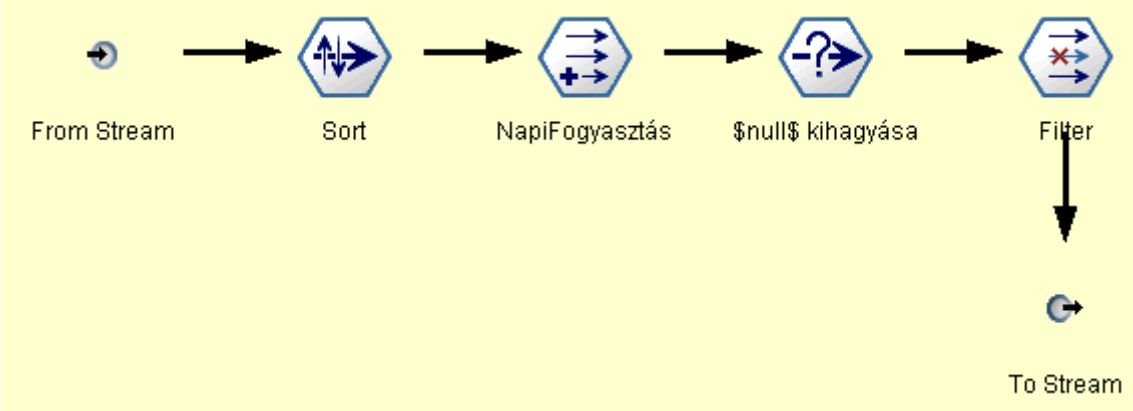
Nyissuk meg a létrejött SuperNode tulajdonságait:



Lehetőségek:

- Az **ANNOTATIONAS** fülön nevet és leírást adhatunk a SuperNode-nak.
- A Zoom In gombra kattintva, pedig „belenagyíthatunk”, megnézhetjük a SuperNode tartalmát.

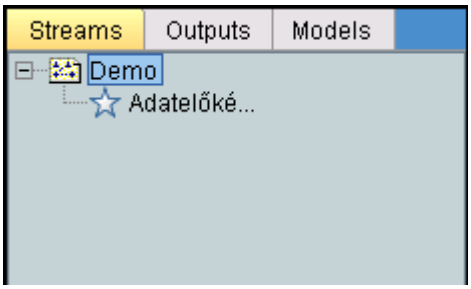
Ha rákattintunk a **ZOOM IN** parancsgombra, megtekinthetjük a SuperNode által elrejtett node-okat:



A SuperNode-on belül, ha szükséges újabb SuperNode-okat helyezhetünk el!

Úgy léphetünk egy szinttel feljebb, hogy a **SUPERNODE** menüpontból kiválasztjuk a **ZOOM OUT** parancsot. Ugyanezt a helyi menüből is megtehetjük.

Vessünk még egy pillantást a **STREAMS** palettára:

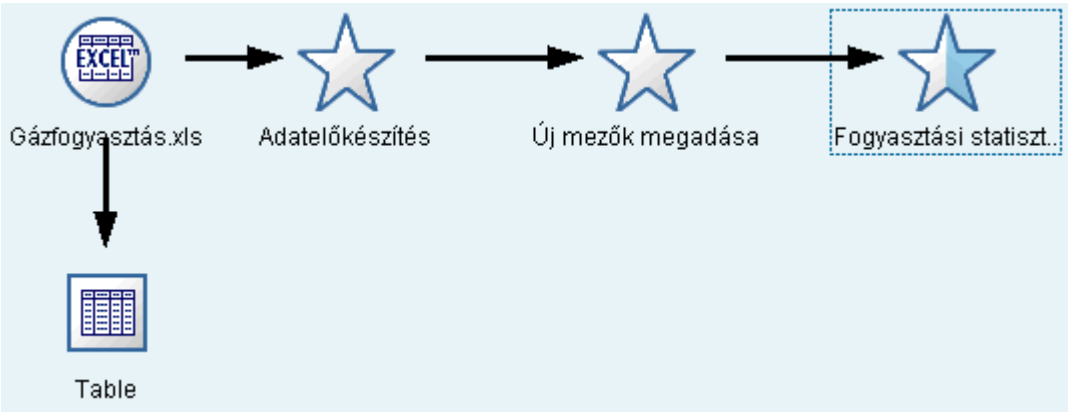


A palettán a stream neve alatt kinyitható/becsukható módon megjelennek a SuperNode-ok. A stream vagy az egyes SuperNode-ok nevére kattintva közvetlenül az adott szintre ugorhatunk.

*Ha szükséges egy meglévő SuperNode-ot fel is számolhatunk, kibontva annak tartalmát egy szinttel feljebbre. Ehhez ki kell jelölni a SuperNode-ot, és **SUPERNODE** menüből az **EXPAND** parancsot választani.*

Készítsen SuperNode-okat a színekkel jelölt node csoportokból. Nevezze is el az elkészült SuperNode-okat.

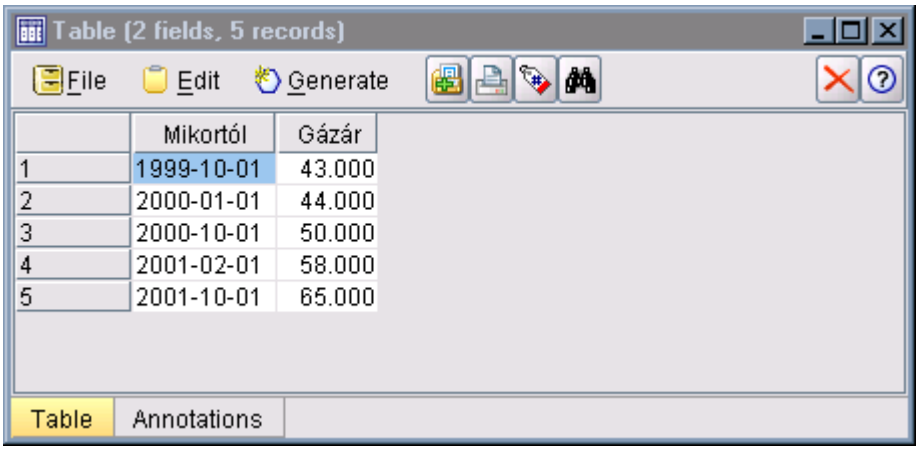
Eredmény:



További adatforrás bevonása

Vegyen fel a stream-be egy újabb Excel source node-ot és olvassa be a GÁZÁRAK.XLS fájl tartalmát!

A beolvasott adattábla két oszlopot tartalmaz:



A **MIKORTÓL** oszlop a gázár változásainak dátumait, míg a **GÁZÁR** oszlop az adott dátumtól érvényes gázár mértékét tartalmazza.

Ha szeretnénk az előző alkalommal elkészült fogyasztási statisztikák mintájára, tehát havi és éves bontásban rezsiköltségeket is számolni, akkor szükség lesz a két adatforrás egyesítésére.

Több tábla adatainak egy táblába történő egyesítésére a **MERGE** és az **APPEND** node használhatók. A táblák oszlopainak egyesítését a Merge, míg két azonos felépítésű tábla sorainak az összesítését az Append node-dal végezhetjük el.

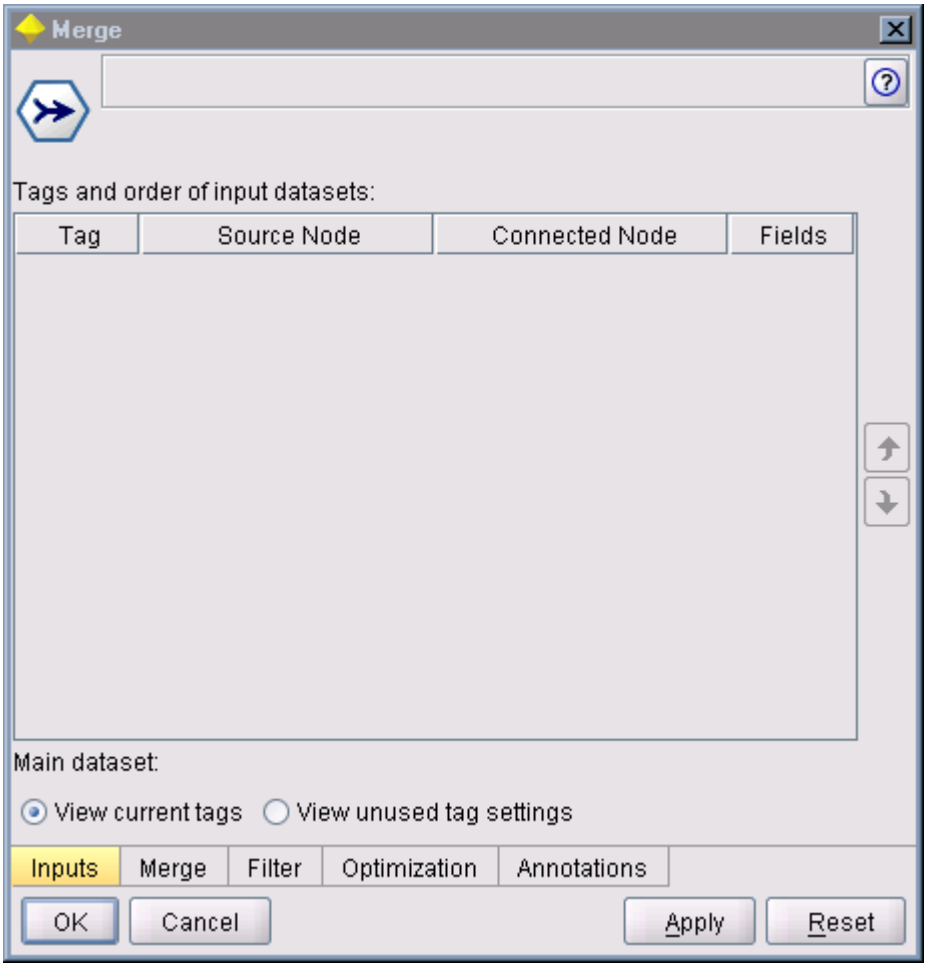
Jelen feladatunk megoldásához a Merge node szükséges, hiszen a két tábla oszlopait szeretnénk egyesíteni: Minden egyes fogyasztási dátumhoz hozzá kell rendelnünk az éppen akkor aktuális gázárat.

A feladatot többféle módon is megoldhatjuk. Éppen ezért mindenekelőtt ismerkedjünk meg a Merge node lehetőségeivel, hogy a számunkra legkényelmesebb módot választhassuk ki!

Merge Node 

A Merge (és az Append is) egy olyan node, amelynek több bemenete is lehet. Tehát a stream-en belül ugyanabba a Merge node-ba több más node-ot is kapcsolhatunk (legalább kettőt, hogy legyen értelme).

A node beállításai:

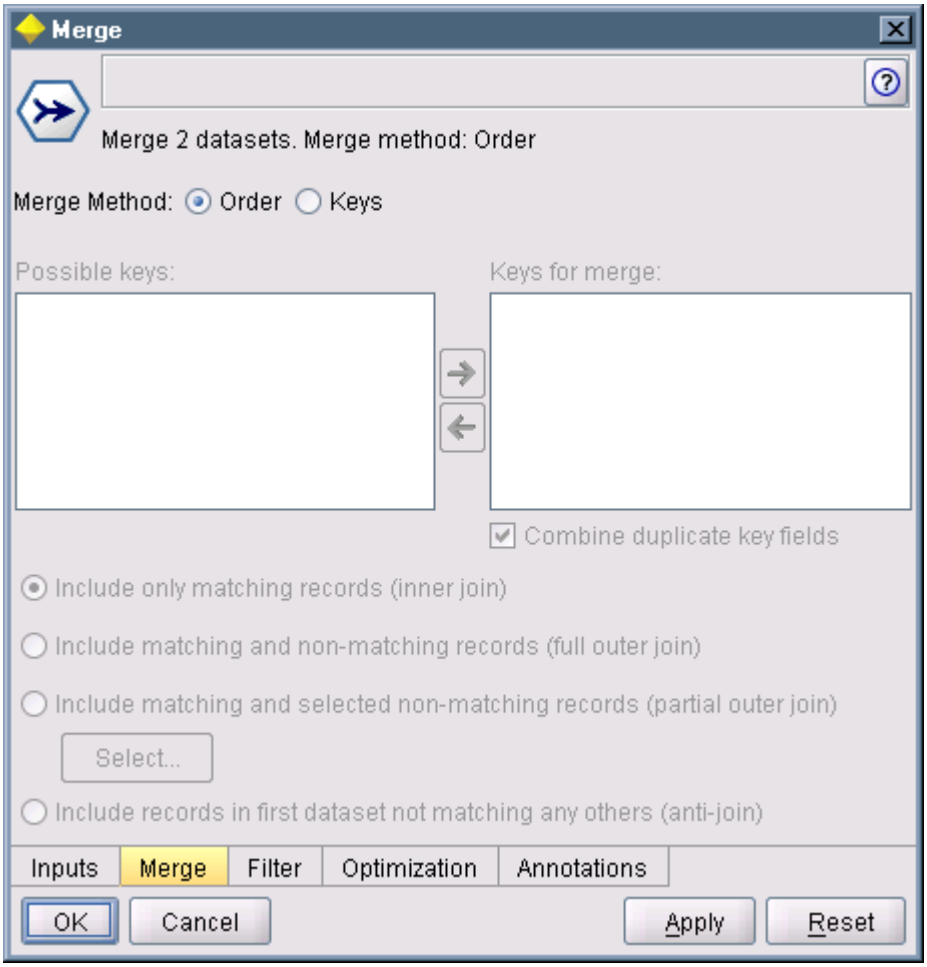


- Az **INPUTS** fülön lévő lista tartalmazza azon adatforrások listáját, amelyek a node bemenetéül szolgálnak.
- A **MERGE** fülön adhatjuk meg a két adatforrás egyesítésének a módját, logikáját.
- A **FILTER** fül már ismert, mezőket nevezhetünk át, és kapcsolhatunk ki.
- Az **OPTIMIZATION** fülön megadhatjuk, hogy mely bemeneti adatok rendezettek már eleve, ezáltal gyorsítva a node futását.

---

*Ha a Merge node valamelyik bemeneti kapcsolatát töröljük, a node akkor is megjegyzi, mint adatforrást. Ha szeretnénk törölni, akkor válasszuk az **INPUTS** fül alján a **VIEW UNUSED TAG SETTINGS** opciót!*

A számunkra legfontosabb beállításokat a Merge fül tartalmazza:



Az oszlopok egyesítésének két lehetséges módja **(MERGE METHOD)** lehet:

- Sorrend alapján **(ORDER)**
- Kulcsértékek alapján **(KEYS)**

Ha az **ORDER**-t választjuk, akkor a táblák oszlopai úgy lesznek egyesítve, hogy minden forrástáblából az azonos sorszámú sorok értékeit illeszti egymás mellé a Modeler. Fontos tudni, hogy ilyenkor az eredménytáblában annyi sor lesz, mint a legkisebb forrástáblában a sorok száma. Oda kell figyelni továbbá a forrástáblákban a sorok megfelelő szempont szerinti rendezésére.

Ha a **KEYS** opciót jelöljük meg, akkor a forrás táblák sorait aszerint rendeli egymáshoz, hogy azok valamilyen általunk kijelölt oszlop, vagy oszlopok értékeiben megegyeznek-e. Ezen oszlopot, vagy oszlopokat hívjuk kulcs mezőnek, mezőknek.

Ezen opció választásakor a Modeler csak azon mezőket (oszlopokat) sorolja fel a lehetséges kulcsok **(POSSIBLE KEYS)** rovatban, amelyek neve megegyezik a forrástáblákban. Emiatt sokszor szükséges lehet egy-egy Filter node használata az egyesítés előtt, hogy átnevezzük a forrástáblák oszlopait.

Ha a kulcsok szerinti (Keys) egyesítést választjuk, akkor négy lehetséges módon hajthatjuk végre a műveletet:

- **INCLUDE ONLY MATCHING RECORDS:**
  Az adatforrások azon sorait kapcsolja össze, ahol a kulcsként megadott oszlopok értékei pontosan megegyeznek. Tehát nem kerül bele az eredménybe egy olyan

sor sem, amelynek kulcs értéke a másik adattáblákban nem szerepel a kulcsértékek között.

- **INCLUDE MATCHING AND NON-MATCHING RECORDS:**
  Mindegyik adatforrás mindegyik sorát beleveszi az eredménytáblába, még azokat is, amelyek kulcsértékei a másik forrástáblákban nem találhatók meg. Amikor egy forrástábla olyan sorát veszi be az eredménytáblába, amelyhez nem rendelhető azonos kulcsértékű sor a többi forrástáblából, akkor a hiányzó oszlopértékeket $null$ értékekkel tölti fel.
- **INCLUDE MATCHING AND SELECTED NON-MATCHING RECORDS:**
  Az előző kettő átmenete. Megadhatjuk, hogy melyik forrástáblákból vegye bele az összes sort az egyesített táblába, és melyekből csak a kapcsolódó sorokat. Ha egyik forrástáblát sem választjuk ki, akkor az első, ha mindegyiket, akkor a második opcióval egyezik meg.
- **INCLUDE RECORD IN FIRST DATASET NOT MATCHING ANY OTHERS:**
  Csak az elsőként megadott forrástáblából veszi be az összes sort, akkor is, ha nincs kapcsolódó kulcsérték a többi táblában. Azt, hogy melyik legyen az első forrástábla az Inputs fülön állíthatjuk be.

Egyesítés

Most, hogy már ismerjük a Merge node lehetőségeit, válasszuk ki az egyesítés számunkra hatékony módját:

- A sorok sorrendje alapján (Order) pillanatnyilag nem tudjuk elvégezni a feladatot. Ehhez arra lenne szükség, hogy a Gázárak táblát kibővítsük minden olyan dátummal, ami még nem szerepel benne, de szerepel a Gázfogyasztás táblában.
- Viszont mindkét tábla tartalmaz dátum típusú adatot, melyeket kulcsnak véve elvégezhetjük az egyesítést. Figyeljünk arra, hogy mindkét tábla tartalmaz olyan dátumot, amelyet a másik tábla nem tartalmaz, tehát, ha nem szeretnénk adatot veszíteni, akkor a full outer join-ot kell választanunk (második opció).

Emlékezzünk: a kulcsként használt oszlopoknak azonos nevűeknek kell lenniük a forrástáblák mindegyikében.

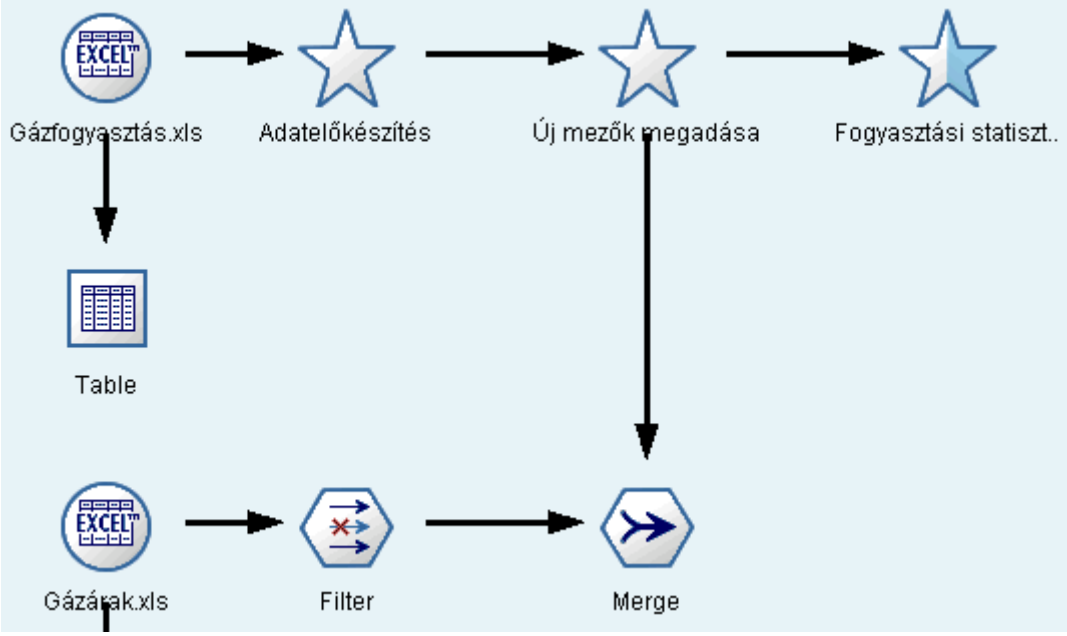
Kapcsoljon egy FILTER node-ot a GÁZÁRAK.XLS source node után! Nevezze át a MIKORTÓL oszlopot DÁTUM–ra!

Helyezzen el egy MERGE node-ot a stream-ben, és kapcsolja hozzá az előbb létrehozott FILTER node-ot!

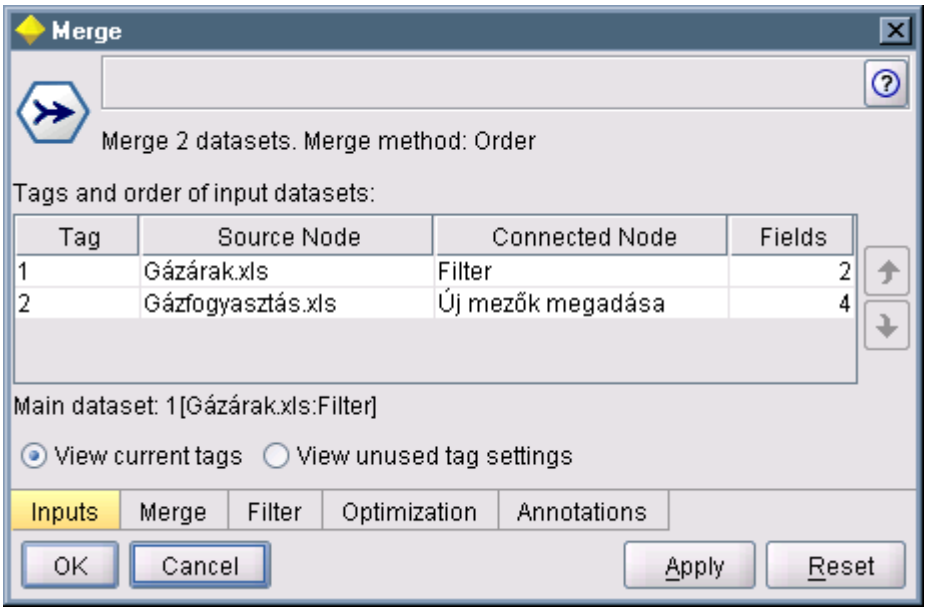
A Merge node-hoz hozzá kell még kapcsolnunk a másik adatforrást. Ne közvetlen a Gázfogyasztás.xls source node-ot kapcsoljuk hozzá, hanem az „Új mezők megadása” SuperNode-ot, hogy ne kelljen később újra számolnunk a napi fogyasztásokat és az Év valamint Hónap mezőket.

Kapcsolja a MERGE node-hoz az ÚJ MEZŐK MEGADÁSA SuperNode-ot is!

A stream eddigi így néz ki:

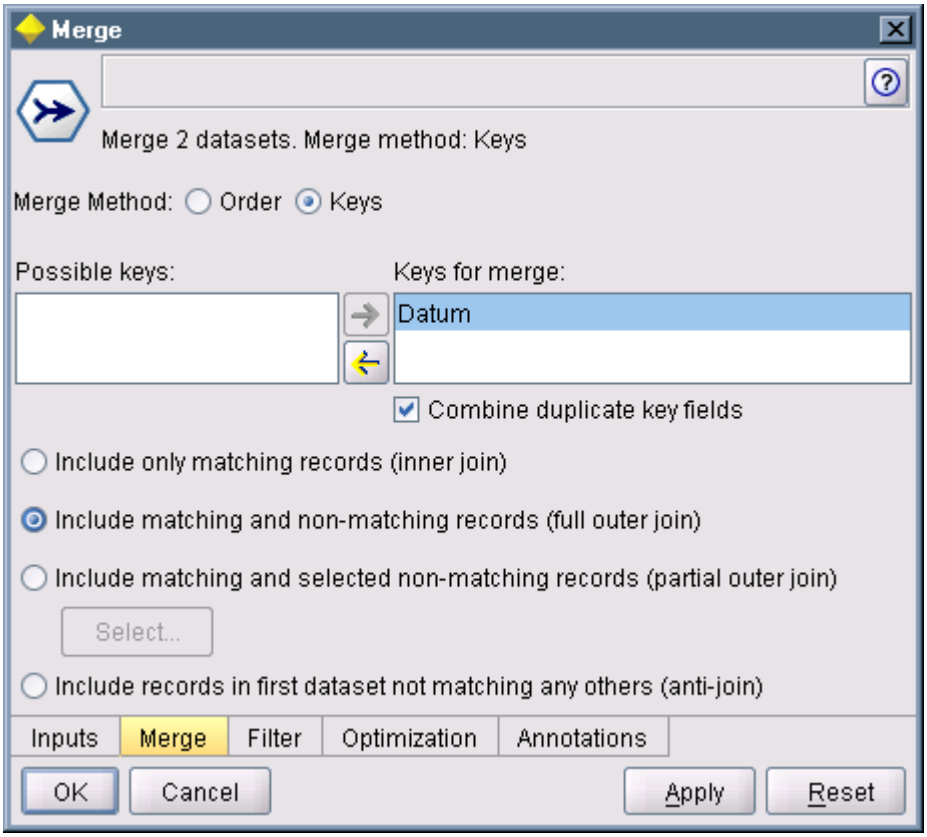


Nyissa meg a MERGE **node beállításait!**

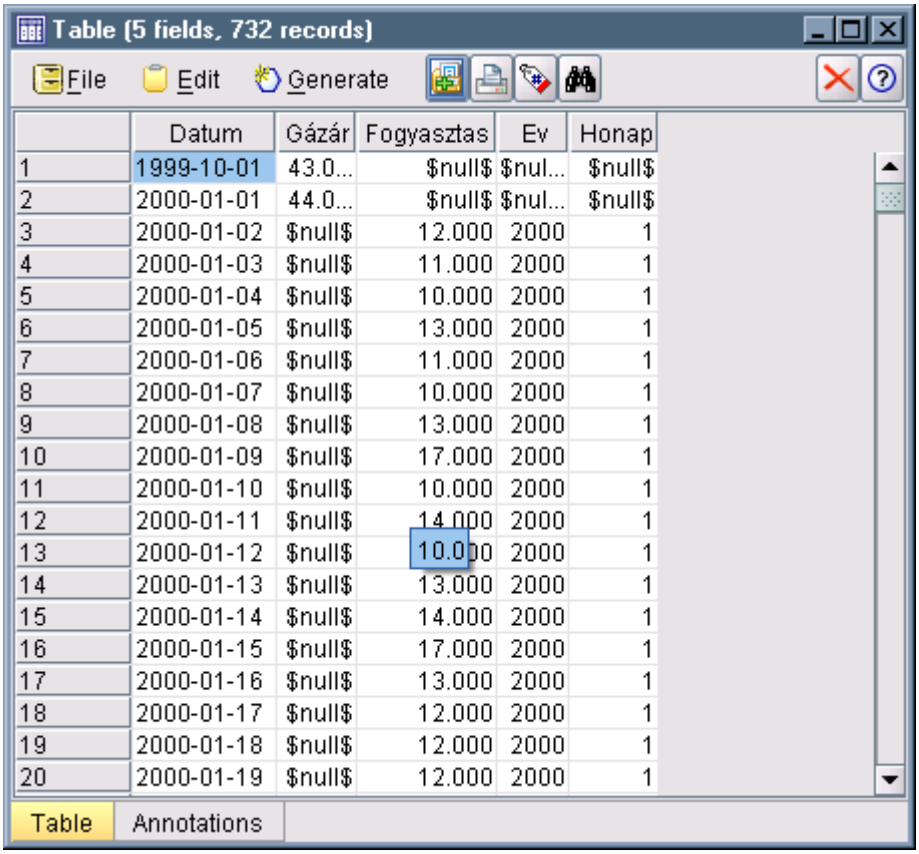


Az Inputs fülön láthatók is az adatforrásaink.

A MERGE **fülön állítsa be, hogy kulcs szerint egyesít, a** DATUM **oszlopot véve kulcsnak, és hogy mindegyik forrástábla minden sorát beleveszi az eredménybe!**



Kapcsoljon a MERGE node után egy TABLE node-ot, és futtassa!



	Datum	Gázár	Fogyasztas	Ev	Honap
1	1999-10-01	43.0...	$null$	$nul...	$null$
2	2000-01-01	44.0...	$null$	$nul...	$null$
3	2000-01-02	$null$	12.000	2000	1
4	2000-01-03	$null$	11.000	2000	1
5	2000-01-04	$null$	10.000	2000	1
6	2000-01-05	$null$	13.000	2000	1
7	2000-01-06	$null$	11.000	2000	1
8	2000-01-07	$null$	10.000	2000	1
9	2000-01-08	$null$	13.000	2000	1
10	2000-01-09	$null$	17.000	2000	1
11	2000-01-10	$null$	10.000	2000	1
12	2000-01-11	$null$	14.000	2000	1
13	2000-01-12	$null$	10.000	2000	1
14	2000-01-13	$null$	13.000	2000	1
15	2000-01-14	$null$	14.000	2000	1
16	2000-01-15	$null$	17.000	2000	1
17	2000-01-16	$null$	13.000	2000	1
18	2000-01-17	$null$	12.000	2000	1
19	2000-01-18	$null$	12.000	2000	1
20	2000-01-19	$null$	12.000	2000	1

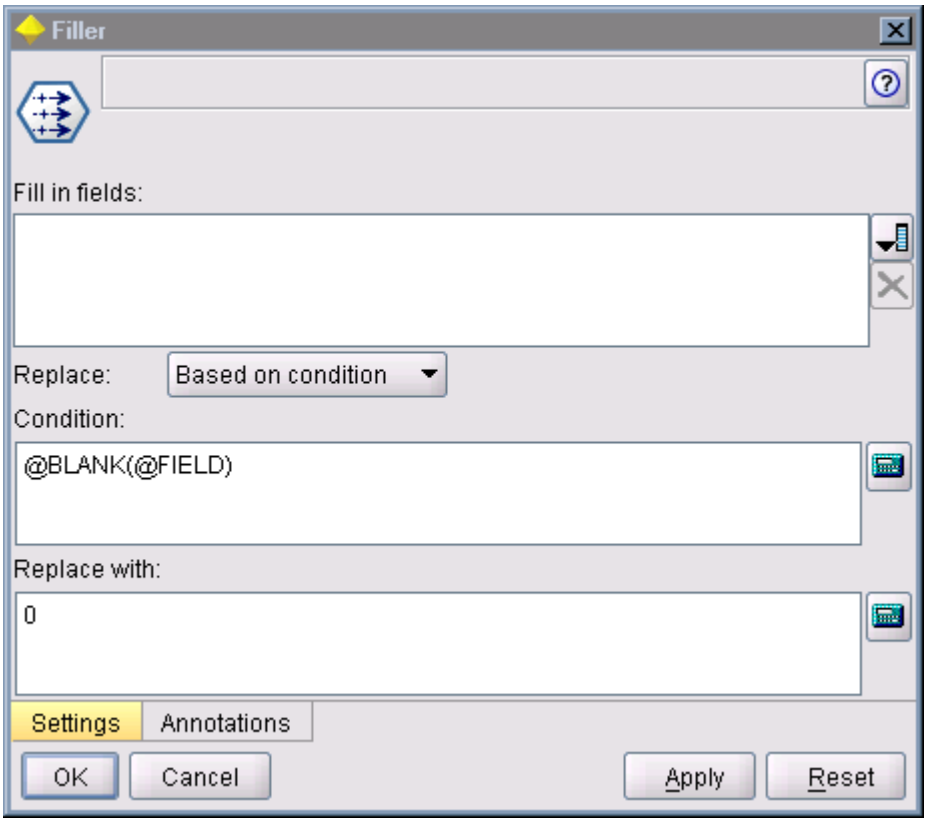
Amint az ábrán is jól látszódik az egyesített táblában mindkét forrás tábla minden sora megtalálható, és ahol nincs kapcsolódó rekord ott **$NULL$** értékekkel van feltöltve a sor.

A feladatunk az, hogy a Gázár oszlop értékeiben a **$NULL$** értékeket kicseréljük az éppen aktuális gázárra. Értsük ezt úgy, hogy feltételezve, hogy a tábla sorai a **DATUM** oszlop szerint vannak rendezve, minden sorban a **$NULL$** értékeket helyettesítsük a felette lévő utolsó nem **$NULL$** értékkel.

Egy már létező mező értékeinek a kitöltésére, újraszámolására a **FILLER** 

node-ot használhatjuk.

Bár a tábla sorai a DATUM oszlop szerint rendezettnek tűnnek, biztos ami biztos, kapcsoljon egy SORT node-ot a MERGE node után, és rendezze a táblát, majd egy FILLER node-ot is kapcsoljon a stream végére!



- A **FILL IN FIELDS** rovatban adhatjuk meg azon mezőket, amelyeknek új értékeket szeretnénk adni.
- A **REPLACE** rovatban annak a módját adhatjuk meg, hogy mely értékeket cseréljük ki:
  - **BASED ON CONDITION:**
    A **CONDITION** rovatban megadott logikai kifejezés adja meg a szabályt. Ha igaz az eredménye, akkor az adott értéket cseréli, egyébként nem.
  - **ALLWAYS:**
    Minden értéket cserél.
  - **BLANK VALUES:**
    A felhasználó által definiált üres értékeket cseréli. Használata megegyezik azzal, mintha feltételként a **@BLANK(@FIELD)** képletet használnánk. A felhasználó az üres értékeket a Type node segítségével határozhatja meg.

- **NULL VALUES:**
  Minden **$NULL$** értéket cserél. Használata ugyanaz, mint a **@NULL(@FIELD)** képlet feltételként történő használata.
- **BLANK AND NULL VALUES:**
  A **$NULL$** és az üres értékek cseréjét is elvégzi.

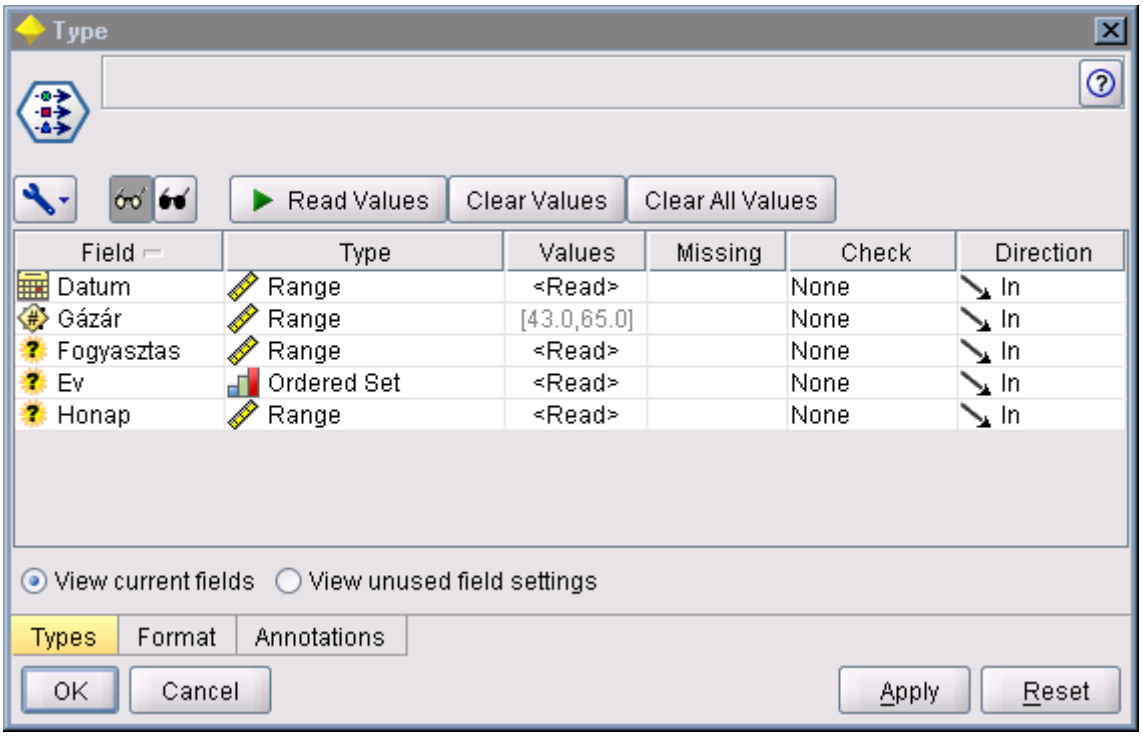
- A **REPLACE WITH** rovatban adhatjuk meg az új értéket, amire cseréli a mezők értékeit.

A Sequence függvények között található @LAST_NON_BLANK függvény használata tűnhetne kézenfekvőnek, hogy segítségével meghatározzuk a Gázár mezőben az új értékeket. Ahhoz, hogy a függvényt használni tudjuk, a $null$ értékeket úgy kell meghatároznunk, mint üres értékeket.

Ehhez egy Type node-ra 

lesz szükségünk.

Még a FILLER node elé fűzzünk be a stream-be egy TYPE node-ot!

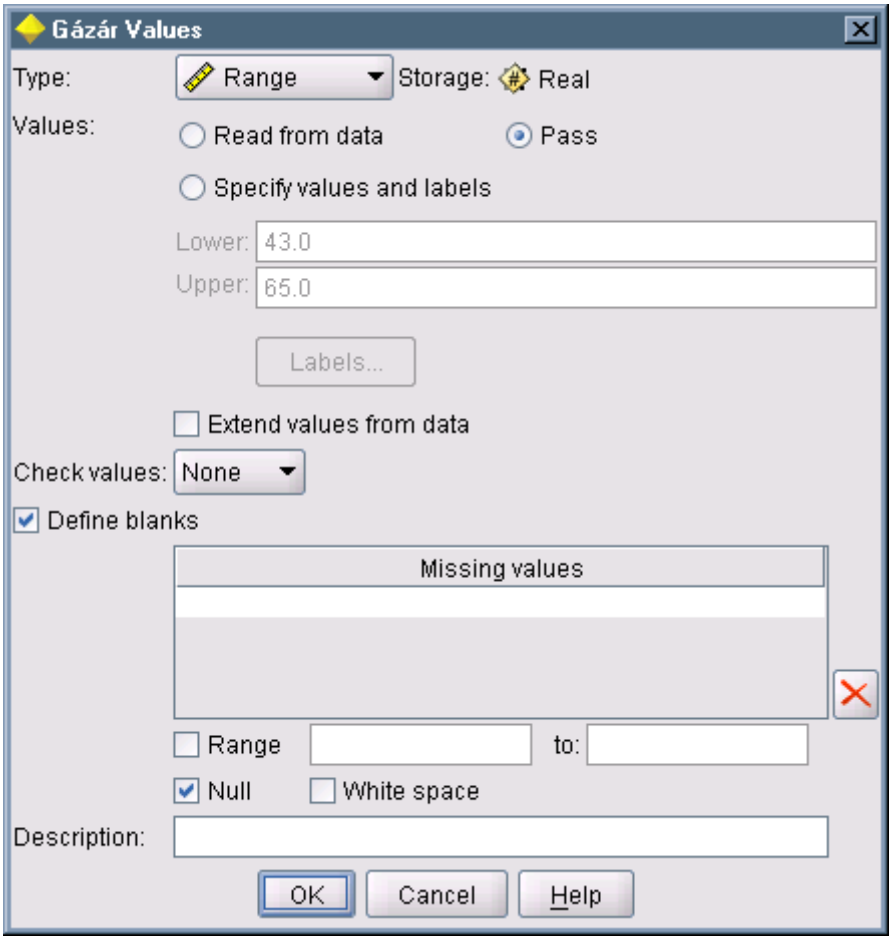


A Type node beépítve megtalálható a már megismert Source node-ban is, csak azoknál eddig még nem ismertük meg az üres értékek meghatározásának a lehetőségét. De természetesen ezt a műveletet, ha szükséges, a Source node-oknál közvetlenül is elvégezhetjük.

A **TYPES** fülön lévő táblázat **MISSING** oszlopában határozhatjuk meg, hogy az egyes mezők mely értékei legyenek a továbbiakban üresként definiálva.

Kattintsunk a GÁZÁR mező MISSING tulajdonságára, és válasszuk az ON(*) opciót.

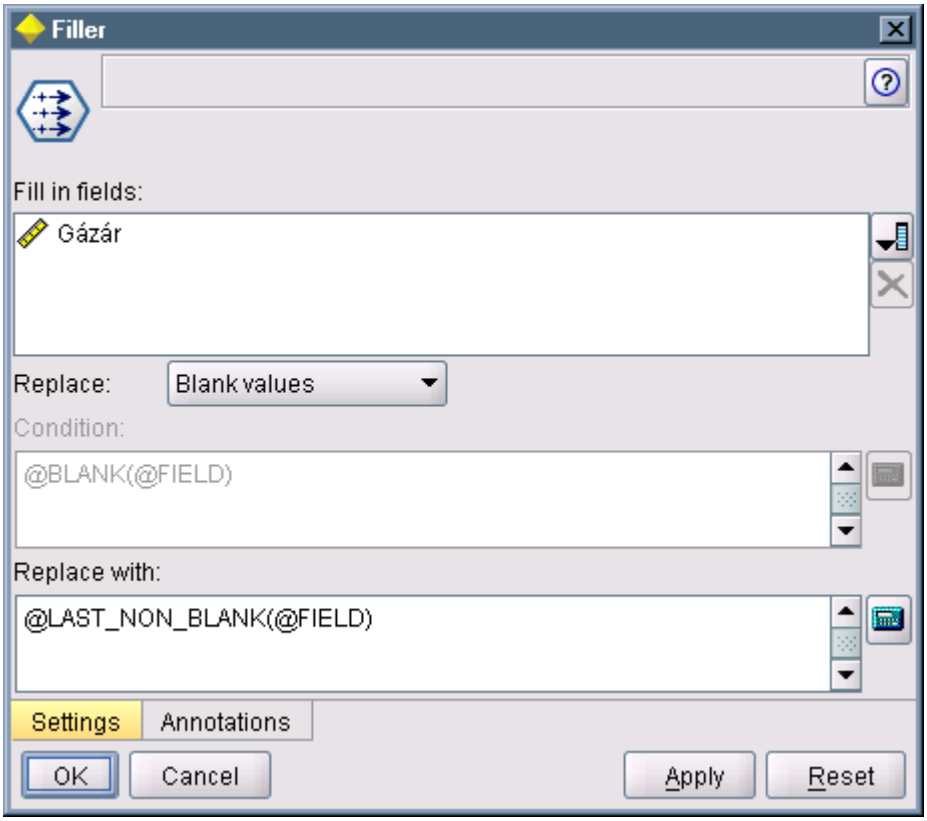
Így bekapcsoltuk egy alapértelmezett módját az üres értékek meghatározásának. Ha szeretnénk részletesebben látni a beállításokat, akkor válasszuk a **SPECIFY...** opciót:



Ami a számunkra fontos most, az a panel alsó része:

- A **DEFINE BLANKS** opció be van kapcsolva:
- Ez szükséges, egyébként nem határoznánk meg az üresnek minősülő értékeket.
- Ha a **DEFINE BLANKS** opció be van kapcsolva, akkor a **MISSING VALUES** listában mi magunk felsorolhatjuk, hogy a node mely értékeket minősítse üresnek. Például egy olyan táblában, ahol a munkahely oszlopban az adatfelvivők a „munkanélküli” szöveget adták meg, ott ezt az értéket megadhatjuk a listában.
- A **RANGE** opció bekapcsolásával megadhatjuk egy teljes tartományát is az értékeknek, amelyeket üresként szeretnénk meghatározni.
- A **NULL** opció bejelölésével a **$NULL$** értékeket is üresnek definiáljuk.
- A **WHITE SPACE** opció megjelölésével a fehér közöket (olyan szövegek, amelyek nem tartalmaznak látható karaktert) tartalmazó értékeket is üresként határozhatjuk meg.

Zárja be az ablakokat, majd nyissa meg a Filler node beállításait, és töltse ki a rovatokat!



*Jelen esetben a **@LAST_NON_BLANK(GÁZÁR)** képlet is megfelelő lett volna.*

Kapcsoljon egy TABLE node-ot a stream végére és ellenőrizze az eredmény helyességét!

Feladatok

1. feladat

Hozzon létre egy új számított mezőt ÉRTÉK néven, melyben a napi fogyasztás és az aznapi gázár szorzatát számolja ki!

2. feladat

Havi lebontásban számolja ki az összes fogyasztást és a gázszámla összegét! A listát év és hónap szerint rendezve jelenítse meg!

3. feladat

Képezzen SuperNode-ot azon node-ok bevonásával, amelyek a költségek beolvasását és egyesítését végzik!

Készítsen SuperNode-ot azon node-okból, amelyek a költségstatisztikák számolását végzik! (1. és 2. feladat)

7. Klaszterezés

Az adatbányászat célja: hagyományos statisztikai számításokkal nem számítható információk kinyerése egy adathalmazból.

A következőkben megismerkedünk néhány adatbányászati eszközzel.

Adatok minőségének vizsgálata: Quality (output) node

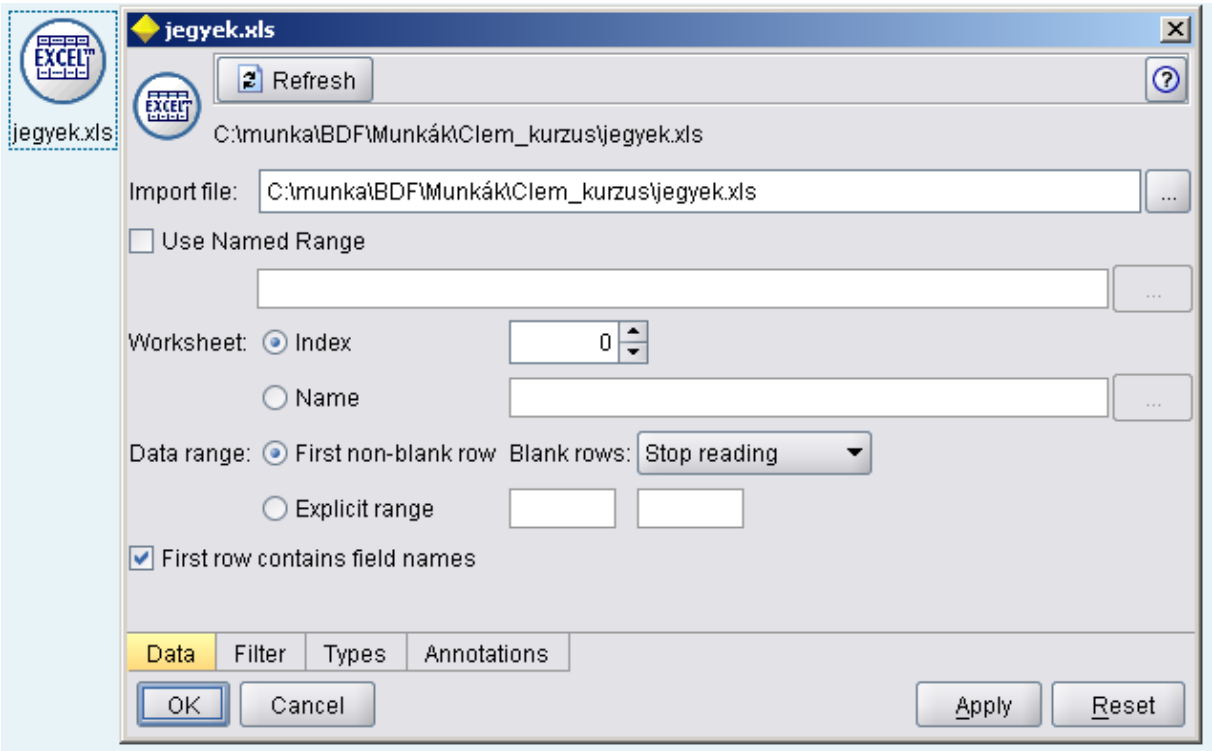
Egy iskola évvégi jegyeit mutatja a **JEGYEK.XLS** fájlban található adatbázis (261 diák adatai):

magyar irod	magyar nyel	történelem	idegen nyelv 1.	idegen nyelv 2.	matemati	fizika	kémia	biológia	földrajz	ének-zene	rajz	testnevelés	informatika	magatartás	szorgalom
4	5	4	5	4	4	4	4	5	4	5	5	5	5	4	4
5	5	5	5	5	5	5	5	5	4	5	5	5	4	5	5
4	4	4	4	4	4	4	4	3	5	4	4	5	4	4	4

Olvassuk be az adatokat a JEGYEK.XLS állományból!

Jelenítse meg a beolvasott adatokat egy Table node-al!

Az Excel (input) node beállításai

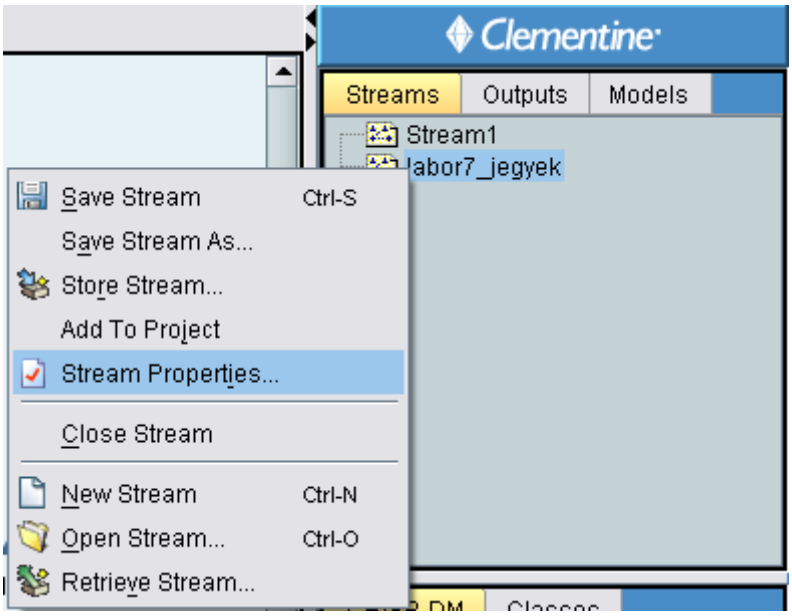


Nevezze át a többszavas mezőket az Excel (input) node Filter fülén!

Field	Filter	Field
magyar irodalom	→	irodalom
magyar nyelv	→	nyelvtan
történelem	→	történelem
idegen nyelv 1.	→	nyelv1
idegen nyelv 2.	→	nyelv2
matematika	→	matematika
fizika	→	fizika
kémia	→	kémia
biológia	→	biológia
földrajz	→	földrajz
ének-zene	→	ének
rajz	→	rajz
testnevelés	→	testnevelés
informatika	→	informatika
magatartás	→	magatartás
szorgalom	→	szorgalom

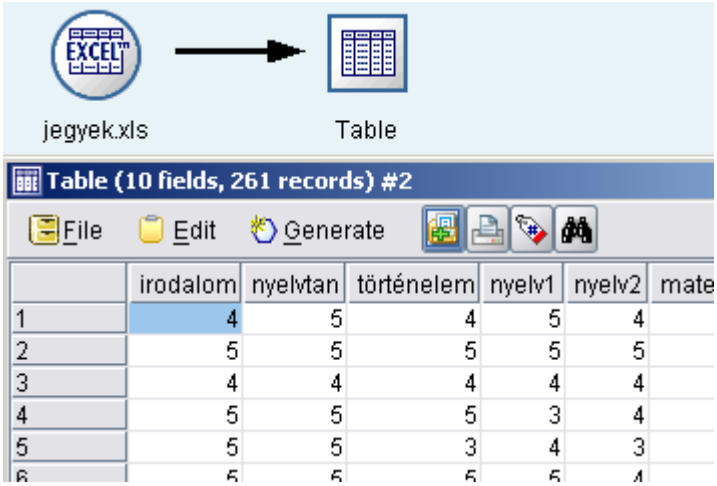
Type fület később állítjuk be.

Állítsa a tizedesjegyeket alapértelmezésként 0 darabra a STREAM PROPERTIES-nél! (Mert osztályzatok lesznek az adatok!)



Az **OPTIONS** fülön ezt a beállítást kell ehhez megtenni:

Egy táblát beillesztve nézze meg a beolvasott adatokat!



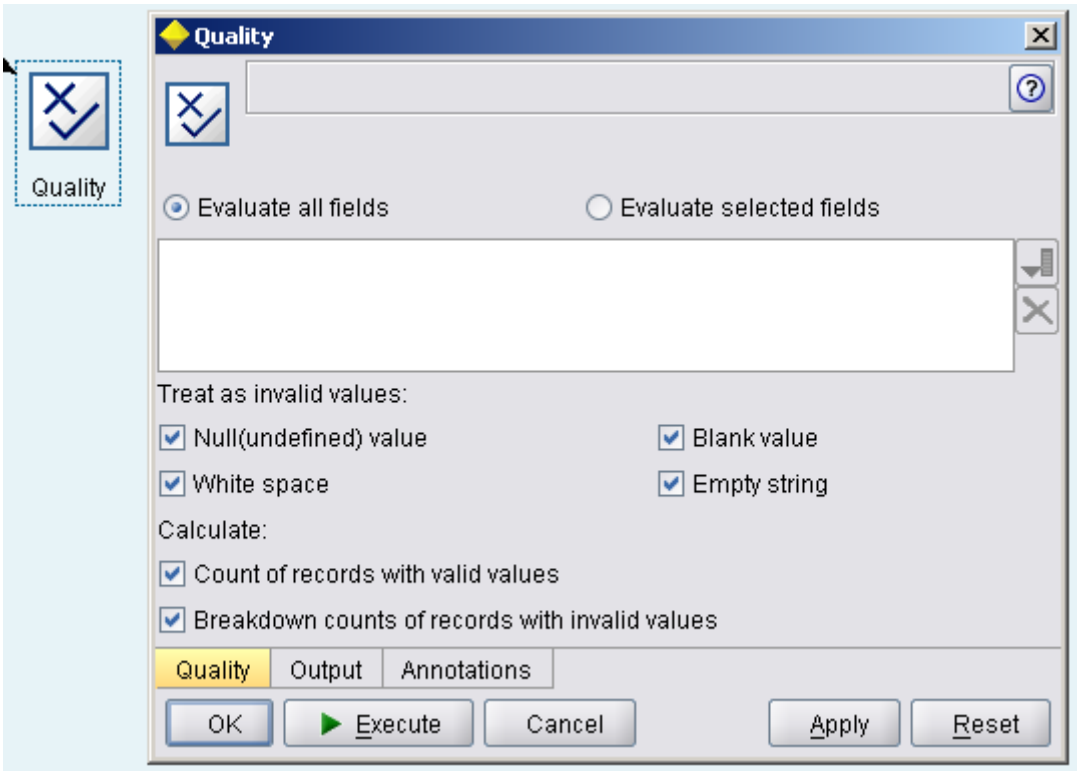
	irodalom	nyelvtan	történelem	nyelv1	nyelv2	mate
1	4	5	4	5	4	
2	5	5	5	5	5	
3	4	4	4	4	4	
4	5	5	5	3	4	
5	5	5	3	4	3	
6	5	5	5	5	4	

Az adatok minőségvizsgálata azt jelenti, hogy melyik változó mennyi használható adatot és mennyi hiányzó adatot tartalmaz. Hiányzó adatként tekinthetők:

- *null* adatok: $null$, a forrásban is hiányoztak
- *blank* adatok: felhasználó által input és type node-nál megadott helyettesítő adatok
- 0 hosszú stringek
- „fehér” stringek: nem látható karakterekből álló stringek

Ilyen adatok előfordulási statisztikáját készíti el a **QUALITY NODE**.

Illesszünk egy Quality node-ot az Excel (source) node után! Próbáljuk ki a rajta levő beállításokat, futtatással meggyőződve az eredményről! A végső beállítások ezek legyenek:



COUNT OF RECORDS WITH VALID VALUES: megadja-e az adott mező szerint érvényes rekordok számát.

BREAKDOWN COUNTS OF RECORDS WITH INVALID VALUES: megadja-e az adott mező szerint érvénytelen rekordok számát a megadott adathiba-típusok szerinti bontásban.

Quality of [11 fields] #1

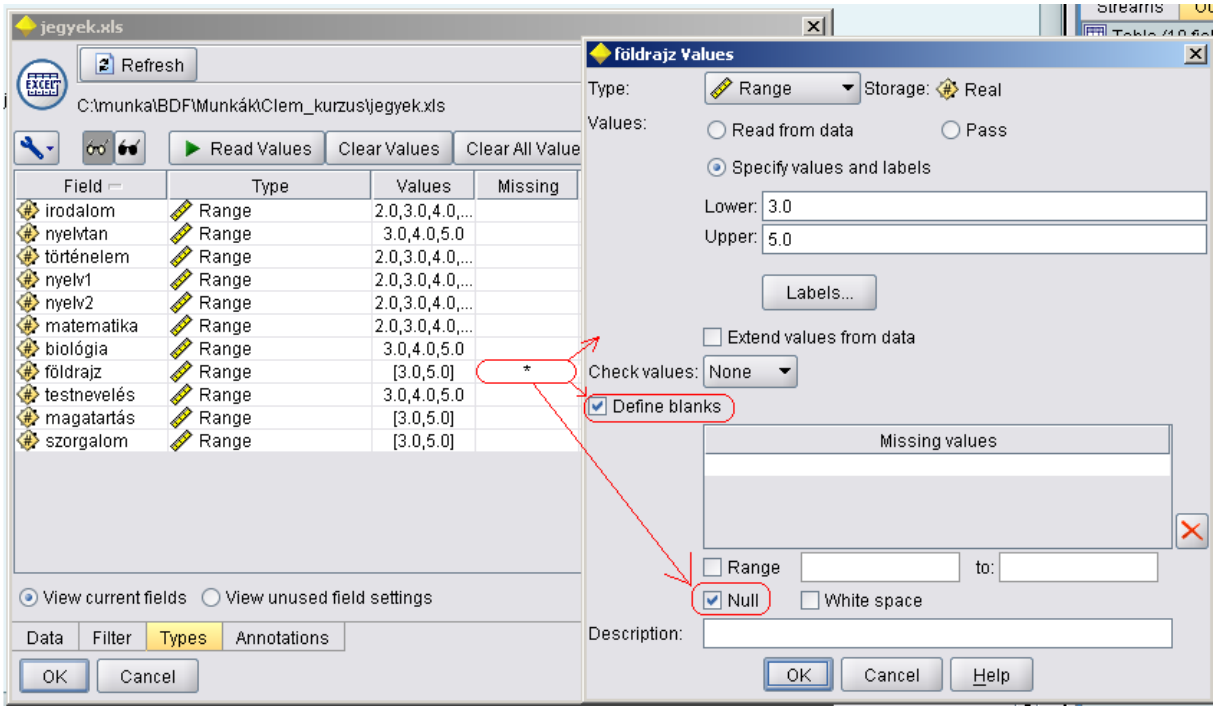
File Edit Generate

Field	% Complete	Valid Records	Null Value	Empty String	White Space	Blank Value
biológia	91,19	238	23	0	0	0
földrajz	65,9	172	89	0	0	0
irodalom	99,62	260	1	0	0	0
magatartás	86,59	226	35	0	0	0
matematika	100	261	0	0	0	0
nyelv1	100	261	0	0	0	0
nyelv2	99,62	260	1	0	0	0
nyelvtan	99,62	260	1	0	0	0
szorgalom	86,59	226	35	0	0	0
testnevelés	96,55	252	9	0	0	0
történelem	99,62	260	1	0	0	0

Nem minden diák esetén szerepel az összes tantárgy (ennek oka lehet: felmentés, fakultációs tárgy, nem minden évfolyamon oktatott tárgyak, gépelési hiba, stb.)

Gördítsük le az Excel (source) node TYPES fülén a *földrajz* mező MISSING rovatát, és a SPECIFY választásával magunk definiáljunk „null" értékű blank adatokat!

Futtassuk ezután újra a beállított Quality node-ot!



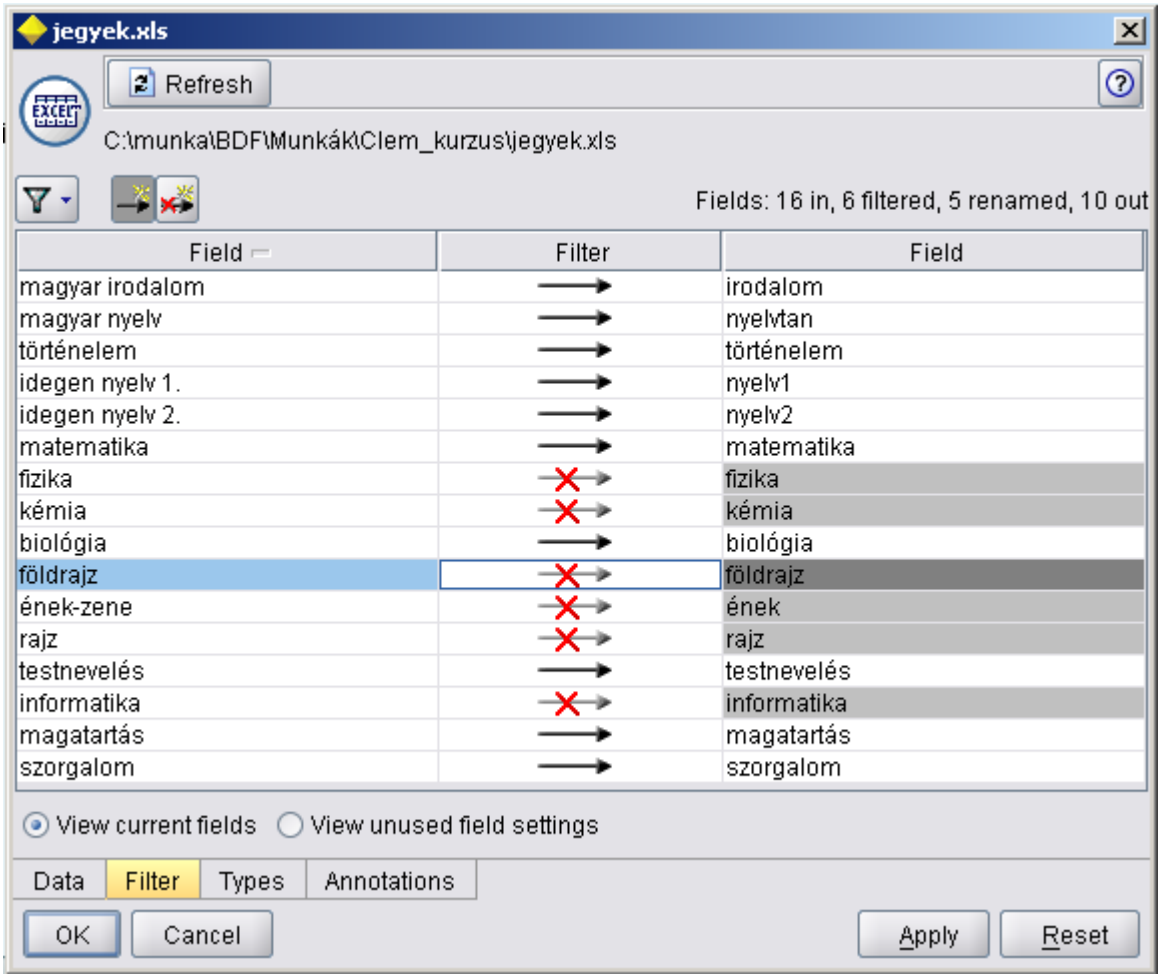
A következő változás látszik a Quality node eredményén, oka ennek, hogy 89 tanuló null adatát blank (felhasználó által megadott) módon helyettesítettük – ugyancsak null értékekkel:

Field	% Complete	Valid Records	Null Value	Empty String	White Space	Blank Value
biológia	91,19	238	23	0	0	0
földrajz	65,9	172	89	0	0	89
irodalom	99,62	260	1	0	0	0

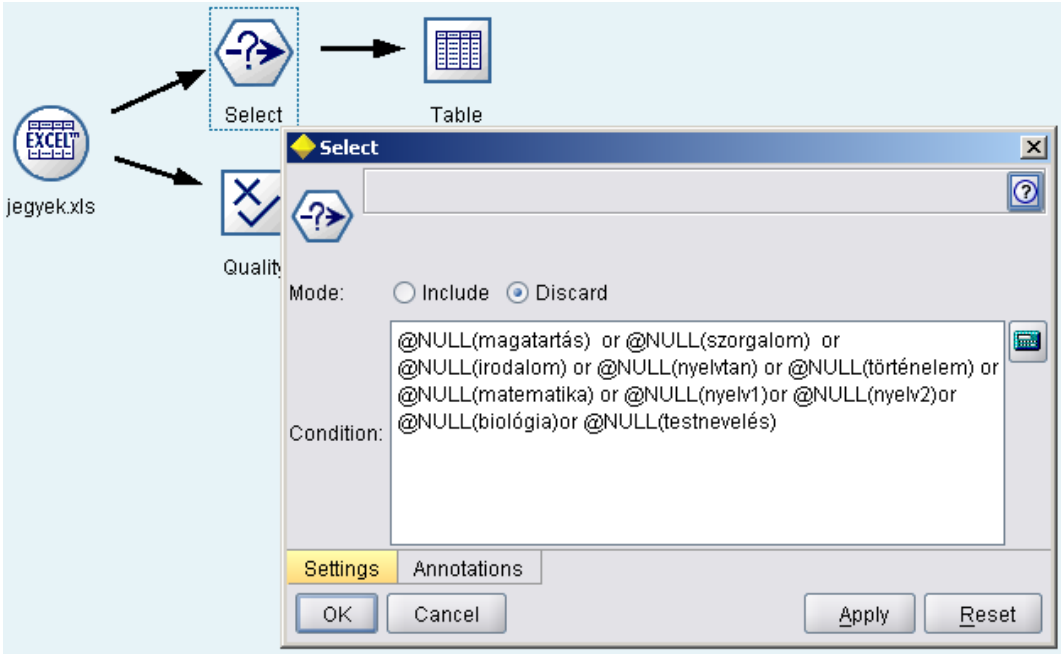
A Quality node eredménye alapján dönthetünk, mely mezőket szeretnénk használni. E példánál hagyjuk meg a 80 % fölötti kitöltöttségű mezőket! A többi mezőre nem célszerű

vizsgálatokat végezni, mert túl sok elhagyott adat van bennük. Cél, hogy legyen elég olyan rekordunk, melynek minden vizsgált mezőjében érvényes adat szerepel.

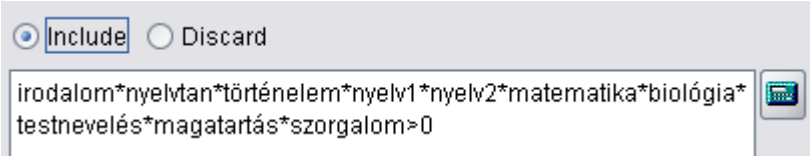
Az Excel (source) node FILTER fülén hagyjuk el a 80% alatti kitöltöttséget mutató mezőket!



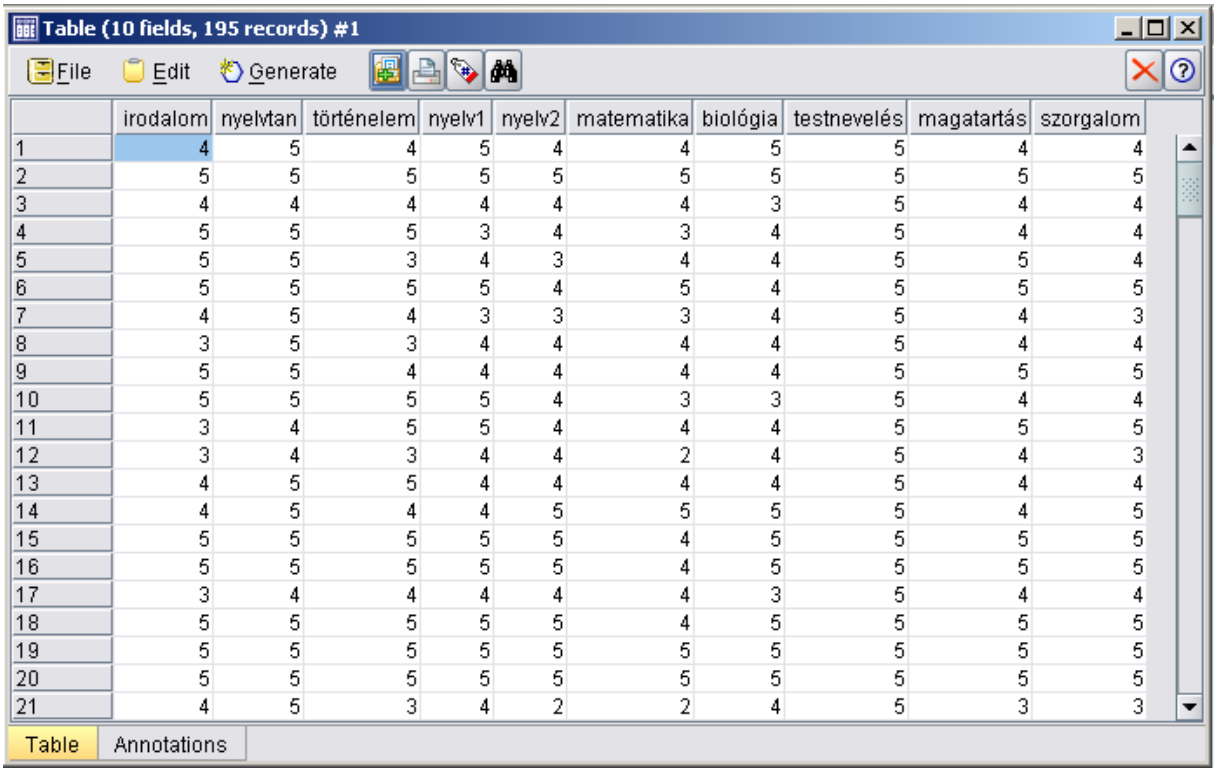
Egy SELECT NODE-dal hagyjuk ki azokat a rekordokat, melyekben hiányzó adat van!



Használhattuk volna a Select node*-ot a következő feltétellel is, kihasználva, hogy $null$ adattal végzett művelet eredménye is $null$, és minden erre vonatkozó logikai feltétel hamis:*



Egy ezt követő eredménytáblázattal nézzük meg a maradék rekordokat!



	irodalom	nyelvtan	történelem	nyelv1	nyelv2	matematika	biológia	testnevelés	magatartás	szorgalom
1	4	5	4	5	4	4	5	5	4	4
2	5	5	5	5	5	5	5	5	5	5
3	4	4	4	4	4	4	3	5	4	4
4	5	5	5	3	4	3	4	5	4	4
5	5	5	3	4	3	4	4	5	5	4
6	5	5	5	5	4	5	4	5	5	5
7	4	5	4	3	3	3	4	5	4	3
8	3	5	3	4	4	4	4	5	4	4
9	5	5	4	4	4	4	4	5	5	5
10	5	5	5	5	4	3	3	5	4	4
11	3	4	5	5	4	4	4	5	5	5
12	3	4	3	4	4	2	4	5	4	3
13	4	5	5	4	4	4	4	5	4	4
14	4	5	4	4	5	5	5	5	4	5
15	5	5	5	5	5	4	5	5	5	5
16	5	5	5	5	5	4	5	5	5	5
17	3	4	4	4	4	4	3	5	4	4
18	5	5	5	5	5	4	5	5	5	5
19	5	5	5	5	5	5	5	5	5	5
20	5	5	5	5	5	5	5	5	5	5
21	4	5	3	4	2	2	4	5	3	3

A végeredményül kapott 195 rekord „tökéletes” minőségű adat, minden mezője osztályzatot tartalmaz. Az adatokat sokféle előzetes vizsgálatnak vethetjük alá, e módszereket előfeldolgozásnak nevezzük. Előfeldolgozás lenne például a nyilvánvaló adathibák kiszűrése is, ennél a példánál a nem 1 és 5 közötti vagy a nem egész értékek kihagyása, manuális javítása.

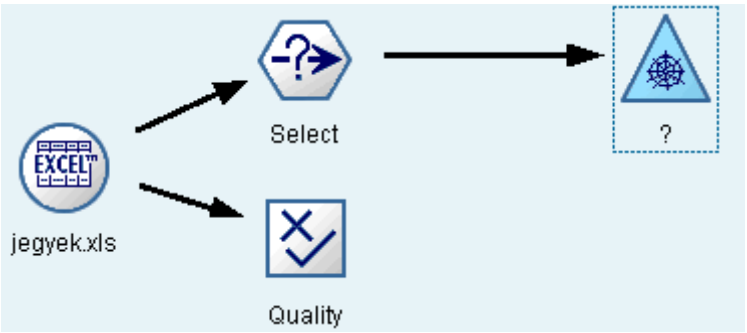
Célszerű minden statisztikai, adatbányászati módszer használata előtt meggyőződni az adatok minőségéről a Quality node segítségével!

Web Node

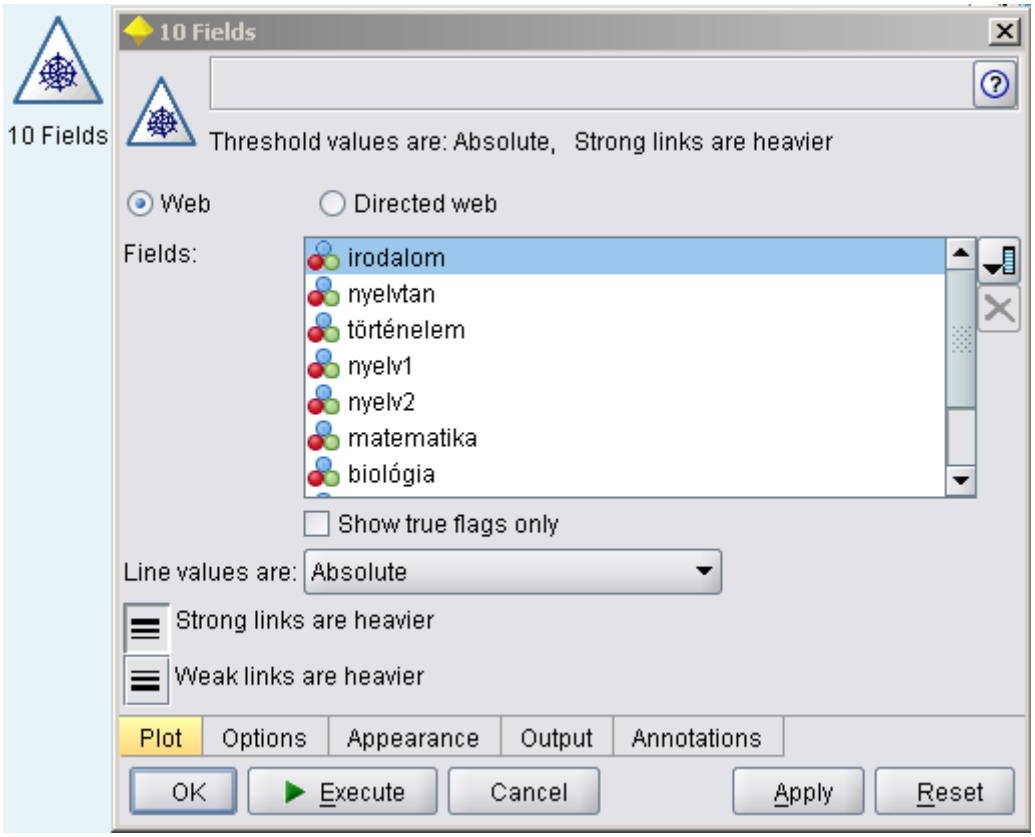
Az adatbányászatban igen fontos a különböző attribútumok (változók, mezők) gyakori együttállását megfigyelni. Halmaz típusú változók leggyakoribb ilyen attribútum-kapcsolatait szemlélteti a Web node. Kapcsolatnak nevezünk ebben az esetben két változó egy-egy adott értékét együttesen tartalmazó rekordot. Pl. a diákok táblázatában 45 rekord esetén van együtt magatartás 5-ös és történelem 4-es. A legtöbb rekordnál meglévő kapcsolatokat mutatja meg a Web (graph típusú) node.

Előkészületként az Excel (source) node TYPE fülén állítsuk mind a tíz használandó változót set típusúra!

Ezután szúrjunk be egy Web (graph) node-ot a streambe a Select node után!

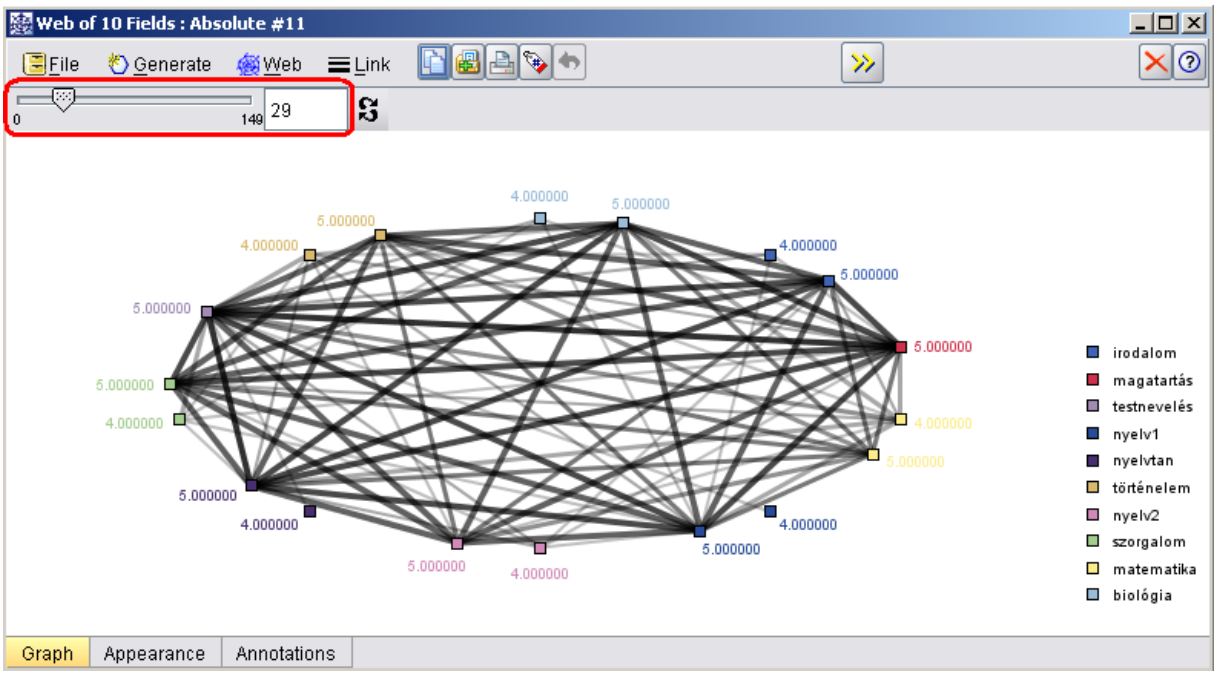


A Web node beállításainál meg kell adni, mely (halmaz típusú!) változók kapcsolatait szeretnénk megjeleníteni.



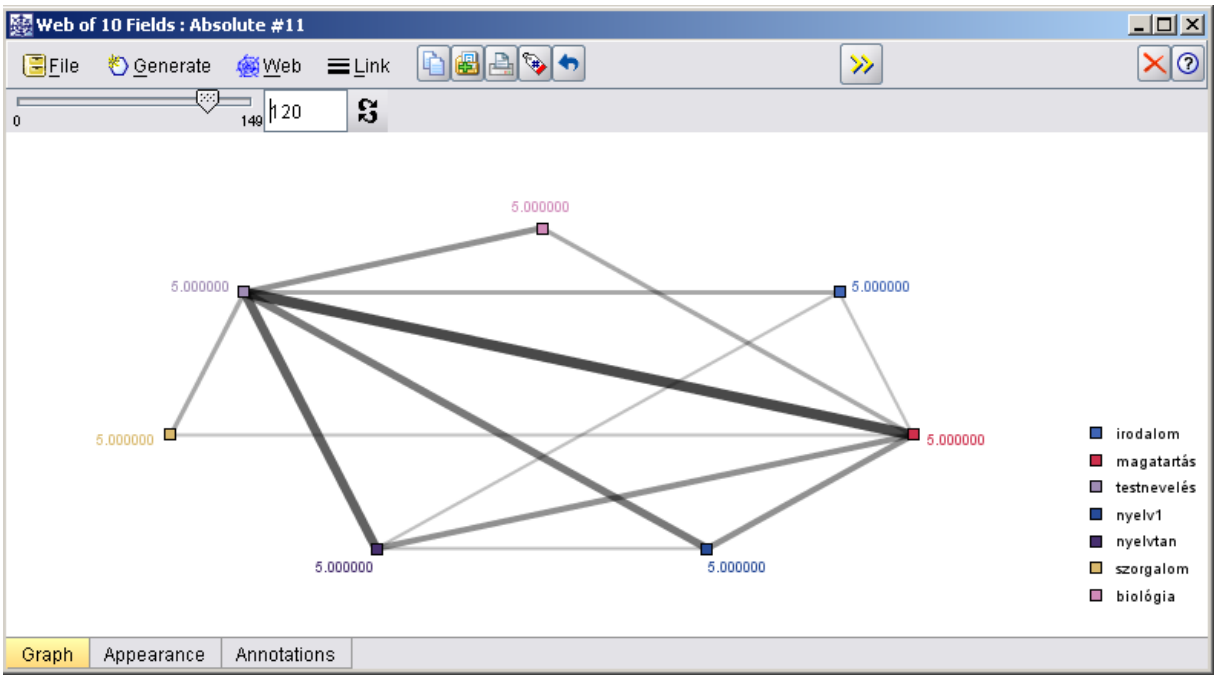
A már ismert módon adjuk meg mind a tíz változót a Fields listán!

Futtassuk le a node-ot a gépi beállításokkal az EXECUTE **gombbal!**



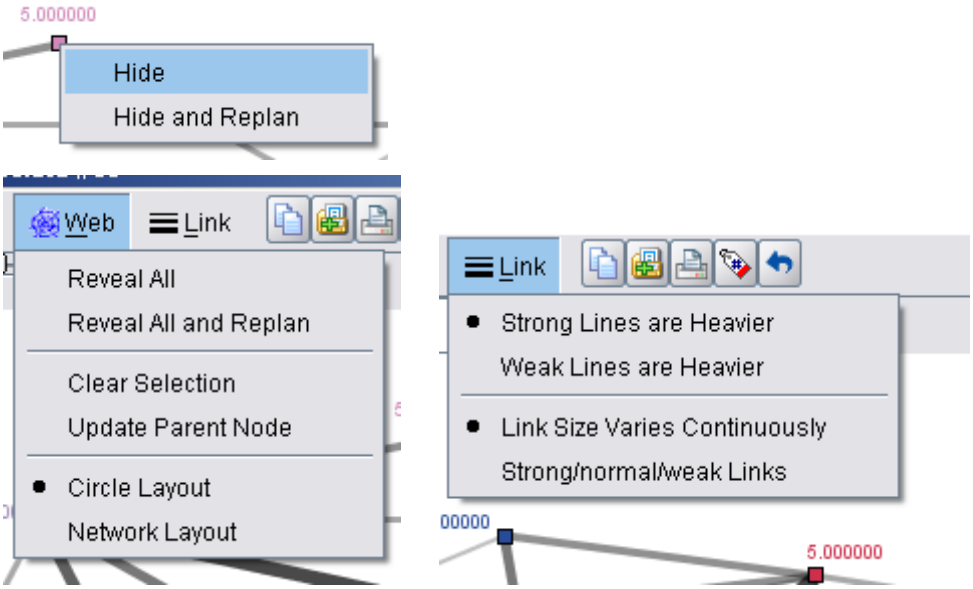
A kapott grafikonon levő gráf csúcsai jelzik az egyes változók bizonyos értékeit, a vonalak vastagságai jelzik két változóérték közti kapcsolatok számát rekord-darabszámban. A kapott grafikonablak interaktív elemeket tartalmaz. A bal felső részen látható (pirossal kiemelt) csúszkával lehet szabályozni, hogy hány rekord fölötti

kapcsolatokat mutasson az ábra, ezt számszerűleg is be lehet írni. Ha 120 értéket adunk meg itt, akkor ezt láthatjuk:



Ha egy csúcson jobbegérrel kattintunk, gyorsmenü jelenik meg, itt a **HIDE** menüponttal elrejthető ez a csúcs, a **HIDE AND REPLAN**-nal pedig a gráf maradék részét újraterveztethetjük e csúcs nélkül.

Próbáljuk elrejteni a biológia 5-ösnek megfelelő csúcsot!

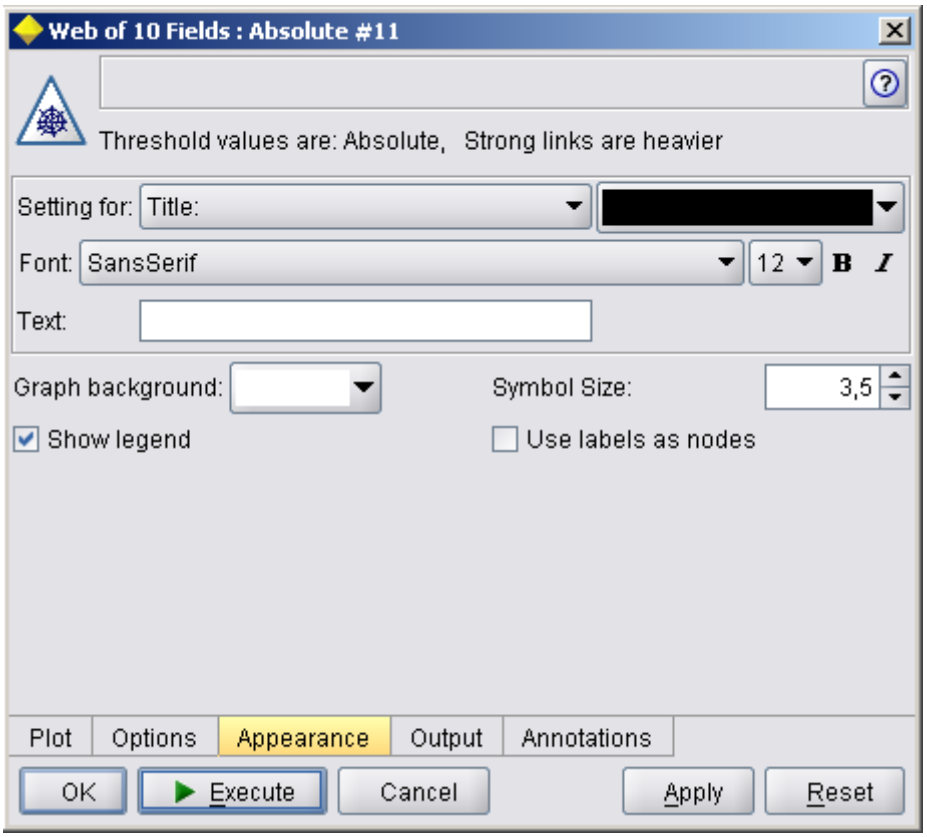


A grafikonablak **WEB** feliratú eszköztárgombjának menüjében a **REVEAL ALL** minden csúcs megjelenítését jelenti (a rejtettek előkerülnek), az alatta levő újra is tervez, a **CLEAR SELECTION** törli a kijelölést a kijelölt részekről, az **UPDATE** frissít, az alsó két menüpont pedig körbe vagy hálózatszerűen a legoptimálisabban helyezi el a csúcsokat.

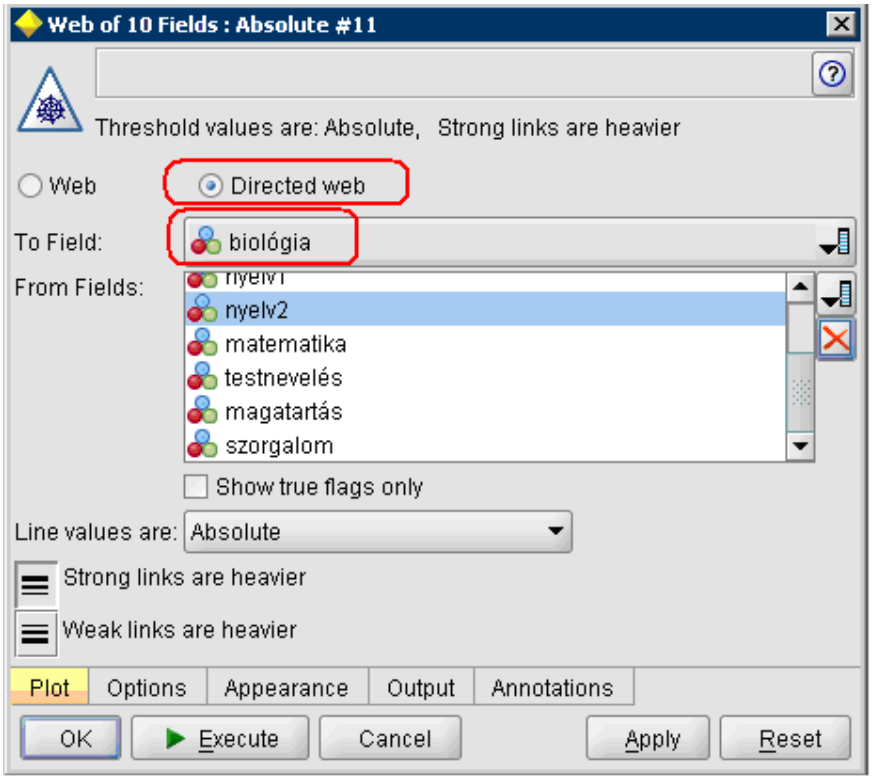
A **LINK** menüben az első két menüpont azt adja meg, hogy a vonalak vastagsága egyenesen vagy fordítottan legyen arányos a rekordszámmal, a második pedig folyamatos és diszkrét (3-féle) vonalvastagság közt vált.

Ugyanezeket a beállításokat megtehettük volna a Web node **OPTIONS** fülén is. Itt a vonalak maximális számát, és megjelenítési korlátokat is megadhatunk. (Ezt lentebb kipróbáljuk, a hozzá tartozó ábra is ott látható.)

A Web node **APPEARANCE** fülén címet, színeket, jelmagyarázatot lehet beállítani.



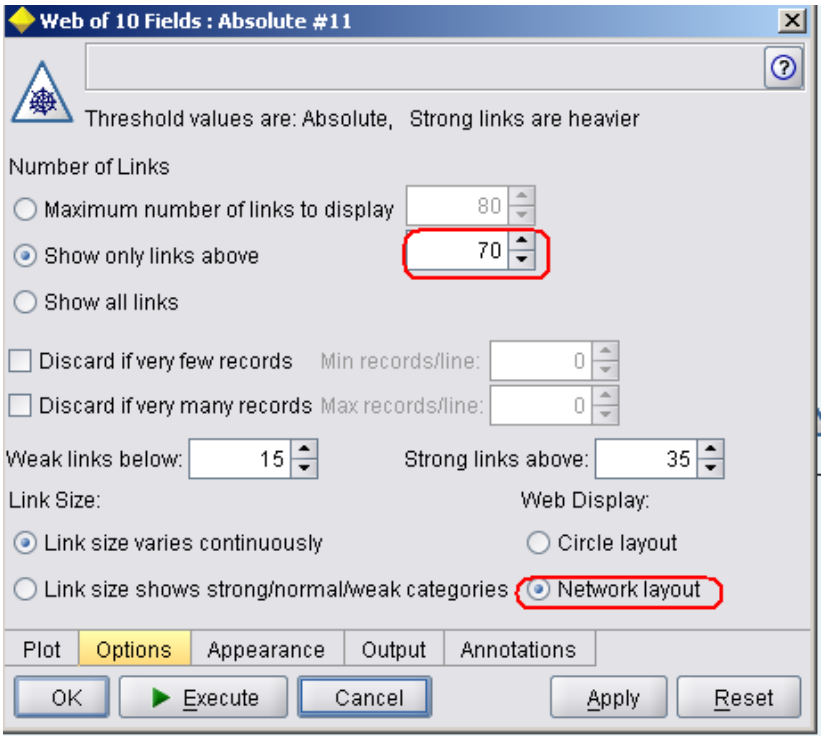
Visszatérve a Web node első **(PLOT)** fülére, ott beállíthatunk a **DIRECTED WEB** lehetőséggel egy célváltozót is.



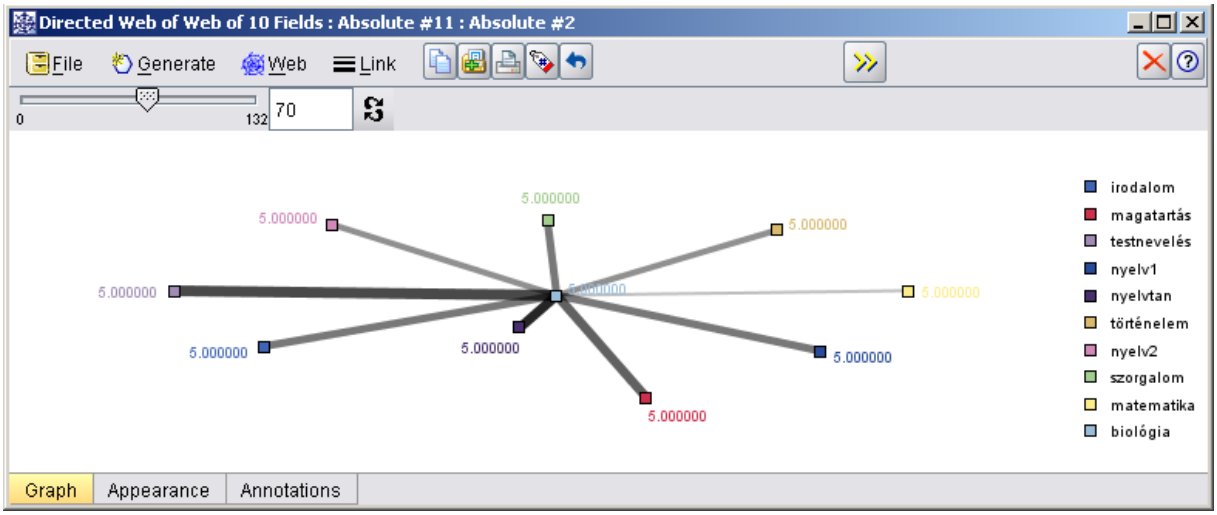
Állítsuk be itt a *biológia* változót, s a másik listáról pedig vegyük el!

Állítsuk az érzékenységet az OPTIONS fül SHOW ONLY LINKS ABOVE részén 70-re! (Így 70 darab rekordnál erősebb kapcsolatokat mutat majd csak a grafikon.)

Kapcsoljuk be a fül alján a NETWORK LAYOUT lehetőséget! (Így a grafikon-gráf csúcsai nem ellipszis alakban lesznek elhelyezve.)



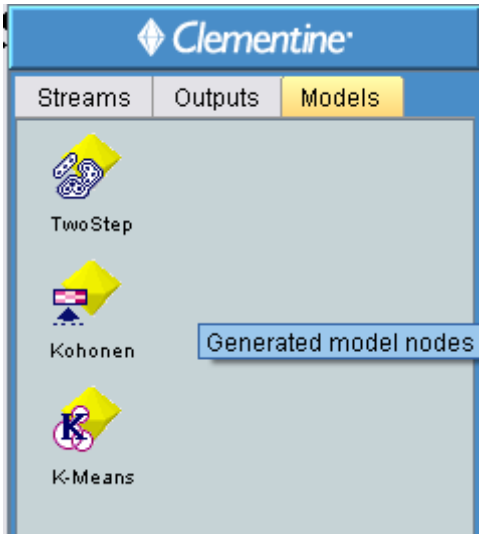
Ez lesz az eredmény futtatás után, középen látható a *biológia* változó 5-ös értéke:



Modellek használata

Adatbányászati modelleket a fenti ötszög alakú node-ok építenek. A modellépítő node-ok egy-egy stream-ág „levelei”, belőlük nyíl nem indulhat tovább. Ha lefuttatjuk ezeket a node-okat, akkor ezek a futtatás hatására többnyire speciális tulajdonságú új node-okat hoznak létre, melyeket később beilleszthetünk a stream-be vagy elmenthetünk.

Az „ötszögek” által generált, kész modelleket jelentő új node-ok a jobb felső munkaablak **MODELS** fülén láthatók, duplakattintással vagy vonszolással illeszthetők be a stream-be. Alakjuk mindig egy sárga „gyémánt”, melynek jobb alsó részén a modell típusára utaló grafika látható:



A kész modelleket vonszolással vagy duplakattintással emelhetjük be a stream-be.

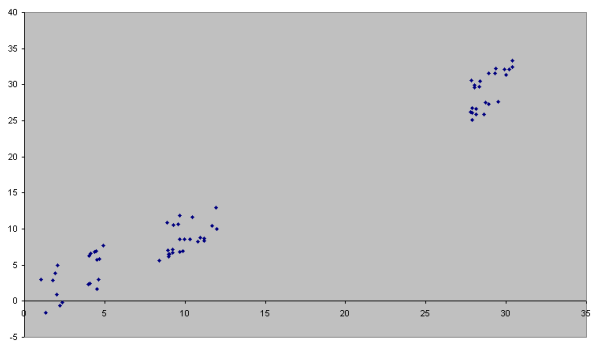
A modellépítés fontos előzménye, hogy megadjuk, milyen bemeneti- és milyen céladatokkal építsen modellt egy-egy ilyen node. A bemenő és kimenő változókat két helyen adhatjuk meg: a source node-ok **TYPE** fülén vagy egy külön **TYPE NODE**-on.

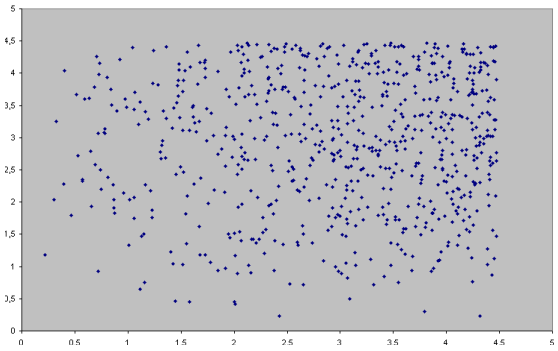
---

*A **MODELS** ablak most még nem tartalmaz „készmodell-node”-okat, mert még nem futtattunk egy modellépítő node-ot sem!*

Klaszterezési módszerek

A klaszterezés nevű adatbányászati eljárással foglalkozunk először, ennek lényege, hogy az adathalmazt olyan csoportokra bontjuk, melyek hasonló tulajdonságú rekordokat tartalmaznak, csak „alig eltérő" attribútumokkal. Például az alábbi kétváltozós adathalmaz közül az elsőnél érdemes, a másodiknál nem érdemes klaszterezni.





Két attribútum (változó) esetén itt szemléletesen látszanak az „adatfelhők". A Celementine képes elvégezni a klaszterezést jóval több változó esetén is.

Leggyakrabban használt modellkészítő node-ok:

Two Step: Kétlépéses módszer. Tömöríti az adatokat az első lépésben, a másodikban hierarchikusan építkezve optimalizálja a belőlük képezhető csoportok számát.

K-means: Megadott számú csoportot képez, úgy, hogy iterációkkal dolgozva kis csoportokat hoz létre, azokat az átlagaikkal helyettesítve készít belőlük újabb és újabb csoportokat, míg a csoportok száma el nem éri az előírt darabszámot. Gyakran ad hasonló eredményt, mint a TwoStep.

Kohonen: kétdimenziós „adatfelhővé" alakítja a sokdimenziós adathalmazt, az X-Y koordinátákkal ábrázolt adatok szemléletesebben mutatják az elkülönülő csoportokat.

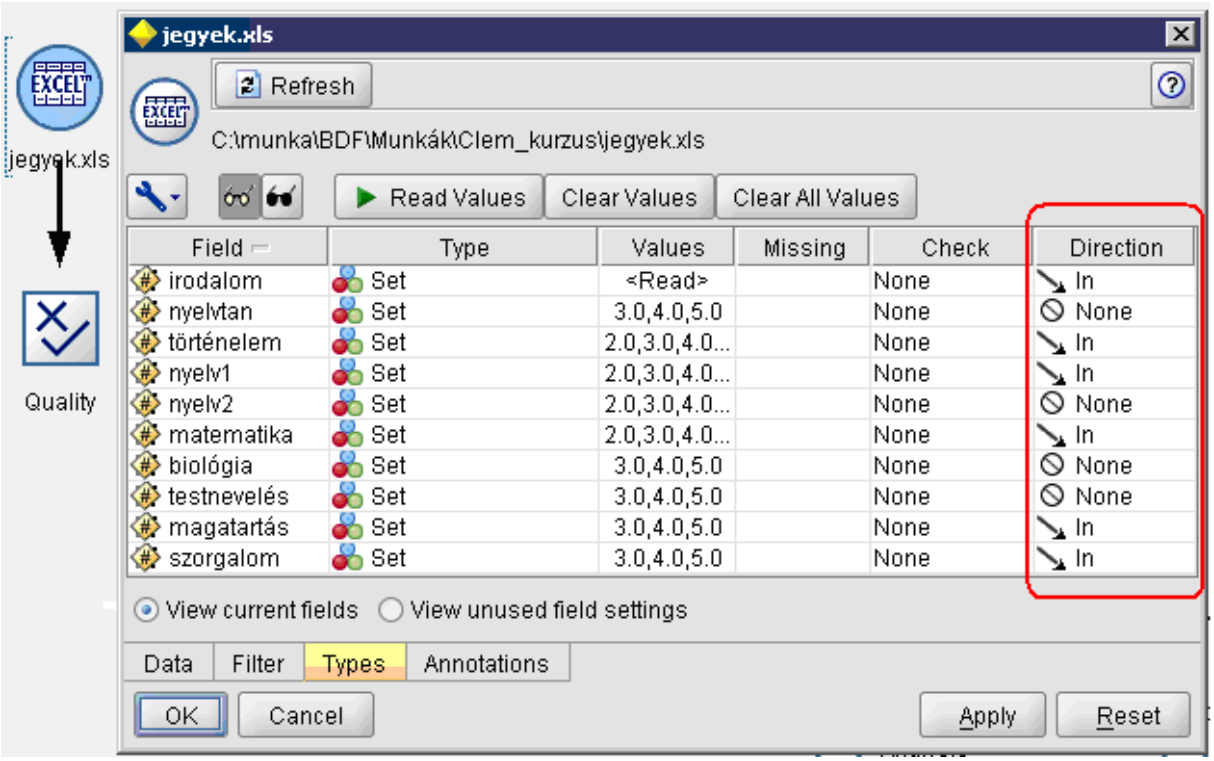
Anomaly: Klaszterezést végez, majd megállapítja, hogy egyes rekordok milyen „távol" esnek saját klaszterüktől, ahol ez az érték magas, azt „rendellenes" adatként tekinthetjük.

Most az első és az utolsó modellel ismerkedünk meg.

Two step modell használata

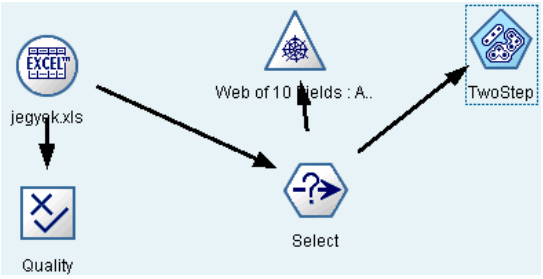
Klaszterezzünk a tanulónyilvántartásban a főtantárgyak (*irodalom, történelem, matematika, nyelv1*), továbbá a *magatartás* és *szorgalom* osztályzatok alapján!

Először az Excel (source) node TYPES fülén a DIRECTION rovatban állítsuk be ezeket a változókat egy leendő modell bemeneteként:



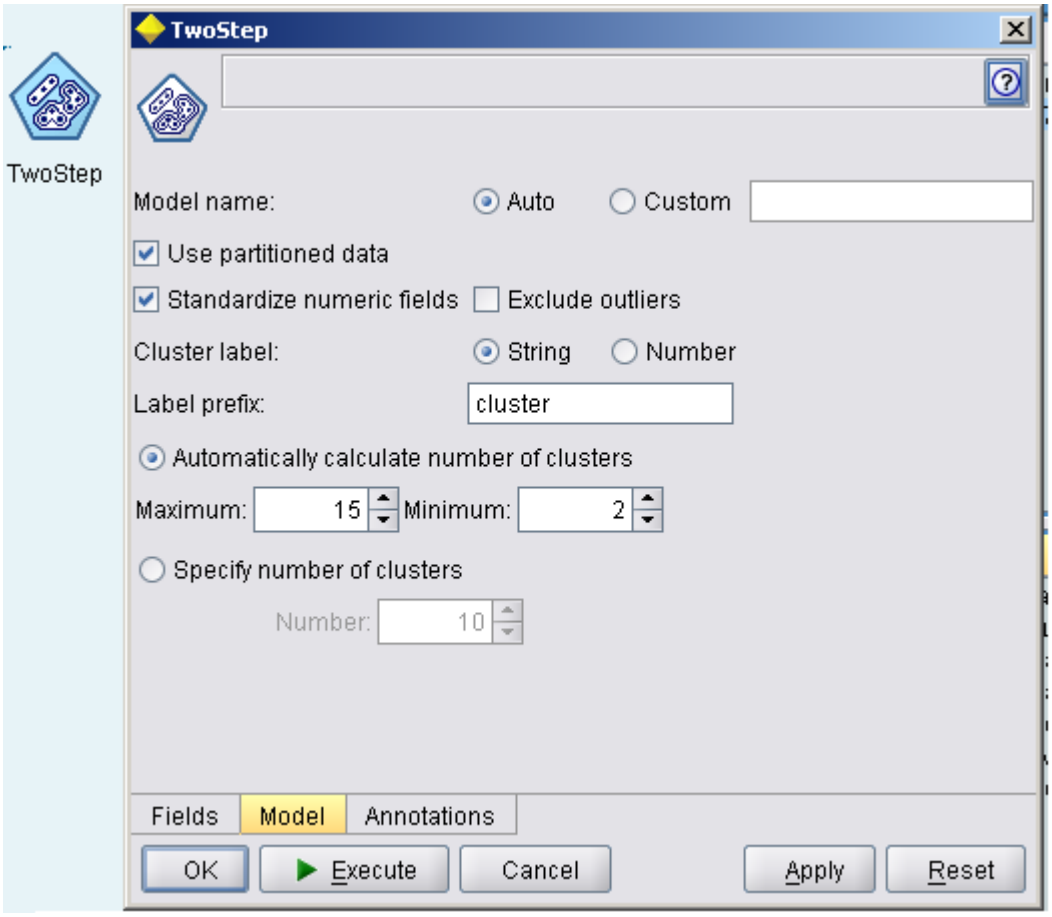
Állítsuk a Types rovatnál mindegyik változót Set típusúvá! Ennek szerepét hamarosan láthatjuk.

Illesszünk be egy TwoStep modellépítő node-ot a Select node után!



Ha majd ezt lefuttatjuk, matematikai módszerek segítségével klaszterekbe sorolja az adatokat. A klaszterbe sorolás szabályait egy beilleszthető „készmodell-node”-ban tárolja. E node-ot a stream-be illesztve és futtatva egy új mezőt fog létrehozni, melynek neve $T-TwoStep, s ebben minden rekordhoz egy jelzés tartozik, hogy ő melyik klaszter tagja.

Ha a modellépítő node beállításait nézzük, ezt láthatjuk:



MODEL NAME: automatikus/user által adott modellnév

USE PARTITIONED DATA: modellépítés, tesztelés vagy érvényesítés céljára megadhatók különböző rekordhalmazok, az előre beállítottakkal dolgozzon-e, vagy az összes adattal.

STANDARDIZE NUMERIC FIELDS: Minden numerikus mezőt először 0 várható értékű 1 szórású változókká transzformáljon-e, vagy adatértékeiknek megfelelő súllyal essenek latba a számításoknál?

EXCLUDE OUTLIERS: A klaszterezés első lépésében a „rendellenes" adatokat kizárja a Modeler a további klaszterépítésből. Ha e rekordok a végén kialakult végleges klaszterekbe nem illenek bele, egy „zaj"-klaszterbe lesznek besorolva, $null$ lesz a klaszterjelzésük. (nem biztos, hogy lesznek ilyen rekordok, de a végső klasztereket befolyásolja az első lépés ilyen módosítása.)

CLUSTER LABEL: A létrehozandó új mező milyen típusú legyen.

LABEL PREFIX: Ha string-adat a klaszternév, akkor milyen előtagja legyen a klaszterkódnak.

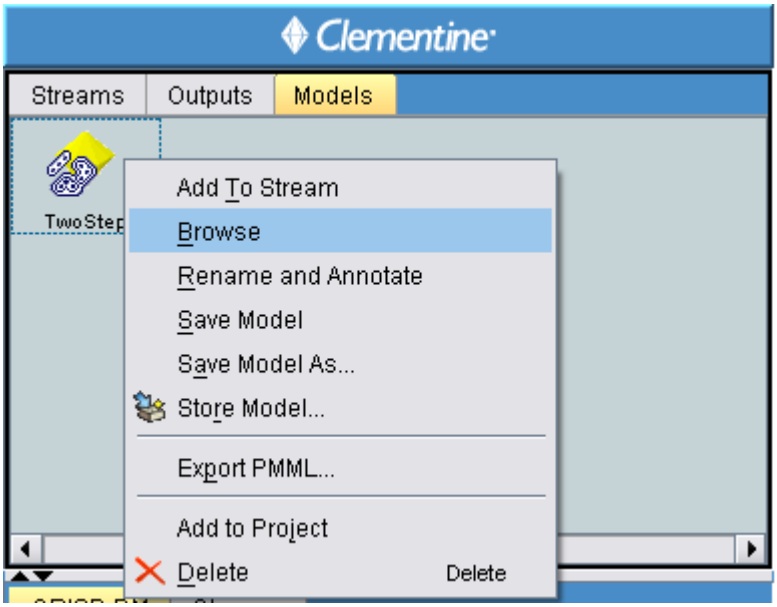
AUTOMATICALLY CALCULATE... : klaszter-darabszám korlátainak megadása

SPECIFY NUMBER... : konkrét klaszter-darabszám megadása

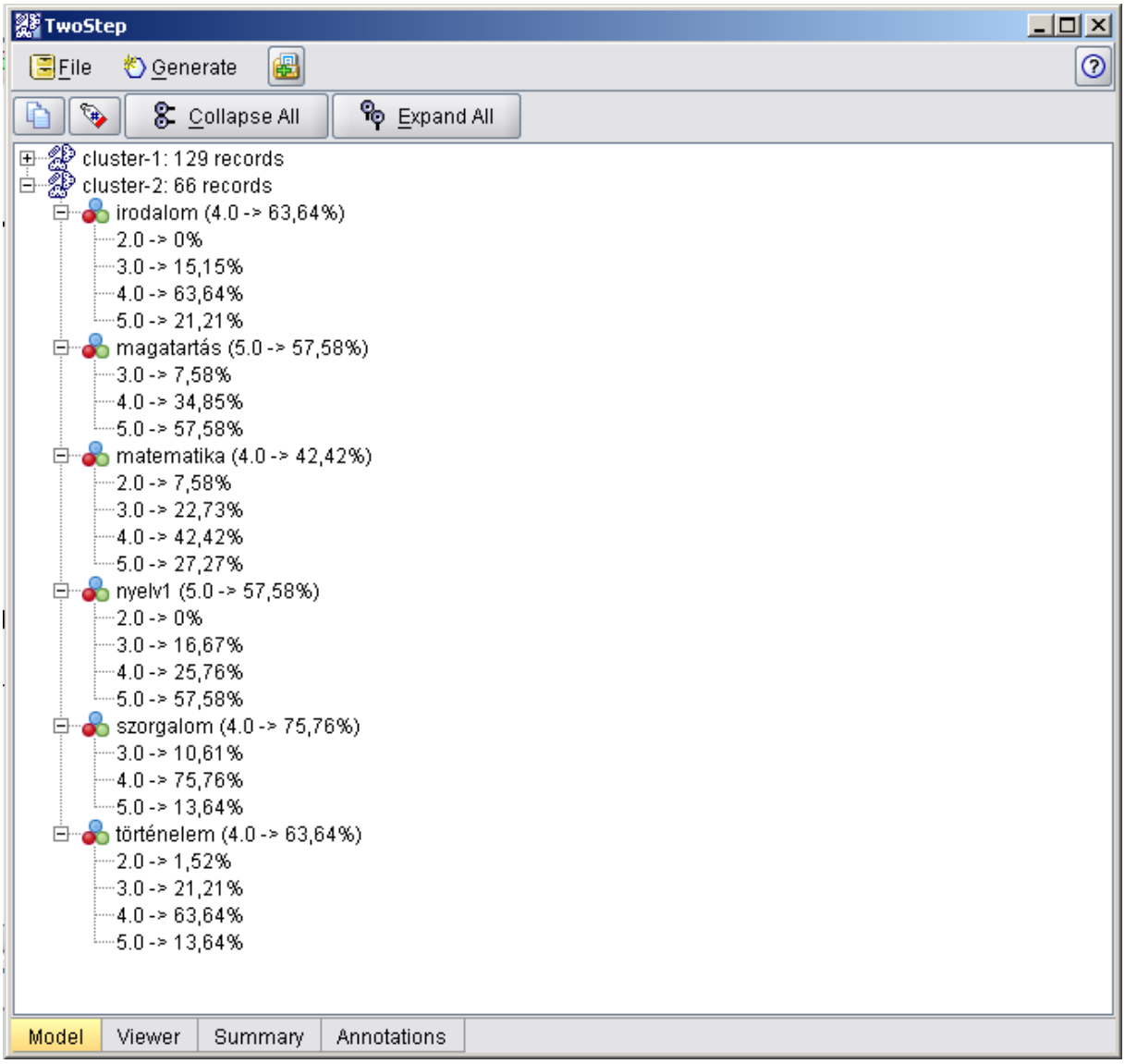
Futtassuk le a modellépítő node-ot a gépi beállításokkal!

Pár másodperc múlva megjelenik egy kész modell ikonja a bal felső ablak **MODELS** fülén.

E gyémánt alakú „készmodell-node"-nak jobbegérmenüjéből válasszuk ki a BROWSE menüpontot!



A felépített modell áttekintő leírása jelenik meg. Információkat kapunk a képzett klaszterekről. Mivel hierarchikusan jeleníti meg az információt az ablak, ezért a + jelekre kattintva nézhetjük meg a részleteket:



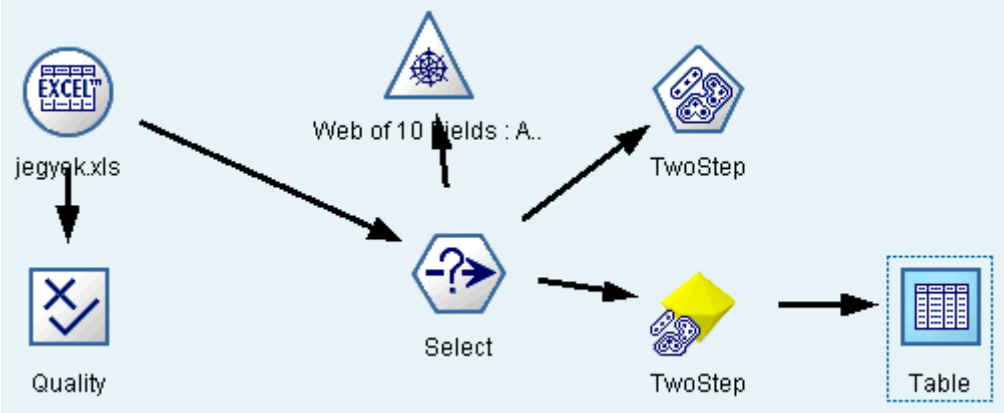
Két klaszter keletkezett, a másodiknak a részleteit kibontva láthatjuk az ábrán. Az első és a második klaszterbeli adatok száma 129, illetve 66, a második klaszter adatainak legnagyobb része irodalomból 4-es (63,64%), az adatok legnagyobb része magatartásból 5-ös (57,58%), stb. minden Set típusú változóra megadja ezt az információt a modell leírása.

Kattintsunk duplán a kész modellre a MODELS ablakban!

Ezzel beemeltük a stream-be egy új node-ként. Gyémánt alakja jelzi, hogy ez nem modellépítő node, hanem kész modell. Bármely olyan streamben alkalmazhatjuk ezt a node-ot, melynek modellben szereplő változói azonosak a jelenlegivel. A kész modell ugyanis szabályokat tartalmaz, milyen bemeneti változók milyen adatai esetén mely klaszterbe tartozik egy-egy adat.

A kész modell (gyémánt alakú) node-ját kössük a Select node-hoz!

Kössünk egy Table (táblázatos megjelenítés) node-ot a „készmodell”-node-hoz!



Futtassuk a Table node-ot!

Table (11 fields, 195 records) #4

File Edit Generate

	irodalom	nyelvtan	történelem	nyelv1	nyelv2	matematika	biológia	testnevelés	magatartás	szorgalom	$T-TwoStep
1	4.0	5.0	4.0	5.0	4.0	4.0	5.0	5.0	4.0	4.0	cluster-2
2	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	cluster-1
3	4.0	4.0	4.0	4.0	4.0	4.0	3.0	5.0	4.0	4.0	cluster-2
4	5.0	5.0	5.0	3.0	4.0	3.0	5.0	5.0	4.0	4.0	cluster-2
5	5.0	5.0	3.0	4.0	3.0	4.0	4.0	5.0	5.0	4.0	cluster-2
6	5.0	5.0	5.0	5.0	4.0	5.0	4.0	5.0	5.0	5.0	cluster-1
7	4.0	5.0	4.0	3.0	3.0	3.0	4.0	5.0	4.0	3.0	cluster-2
8	3.0	5.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	4.0	cluster-2
9	5.0	5.0	4.0	4.0	4.0	4.0	4.0	5.0	5.0	5.0	cluster-1
10	5.0	5.0	5.0	5.0	4.0	3.0	3.0	5.0	4.0	4.0	cluster-2
11	3.0	4.0	5.0	5.0	4.0	4.0	4.0	5.0	5.0	5.0	cluster-1
12	3.0	4.0	3.0	4.0	4.0	2.0	4.0	5.0	4.0	3.0	cluster-2

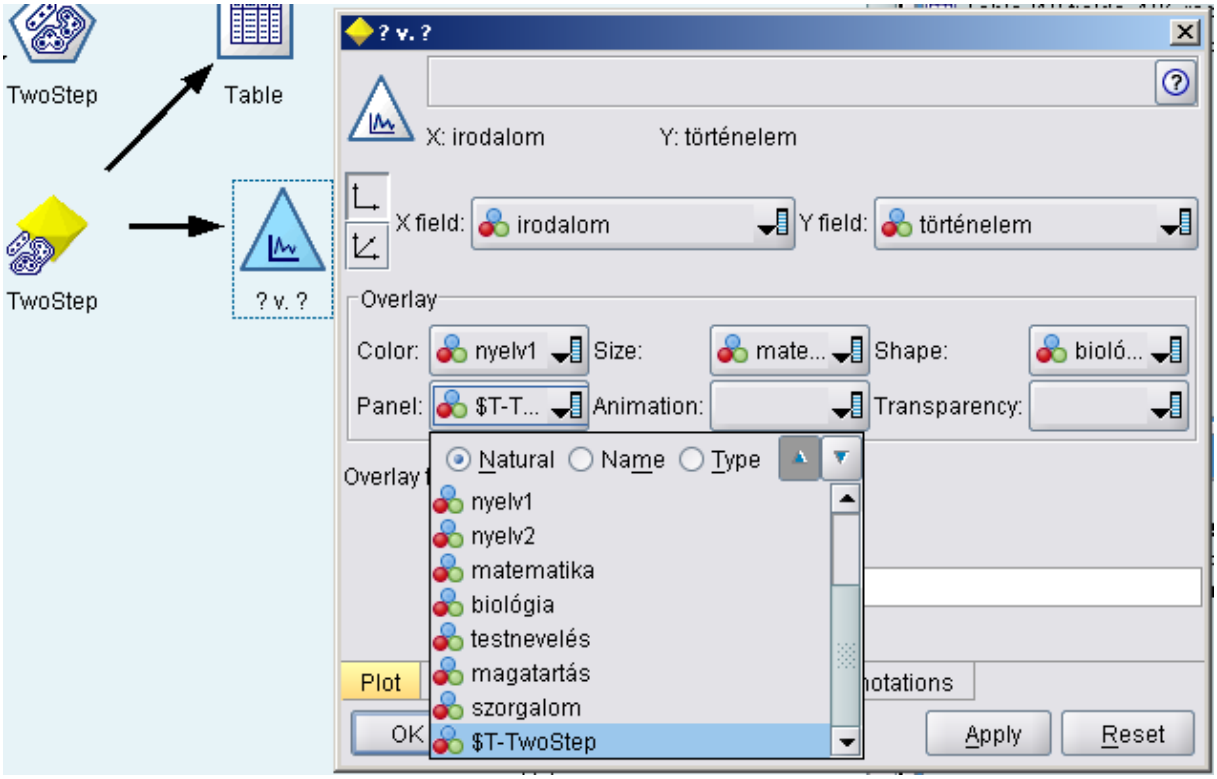
Table Annotations

Most megnézhetjük a klaszterezés eredményét. A kész modell node-ja új mezőt adott az adatokhoz, ebben olvasható, melyik rekord van az 1-es és melyik a 2-es klaszterben.

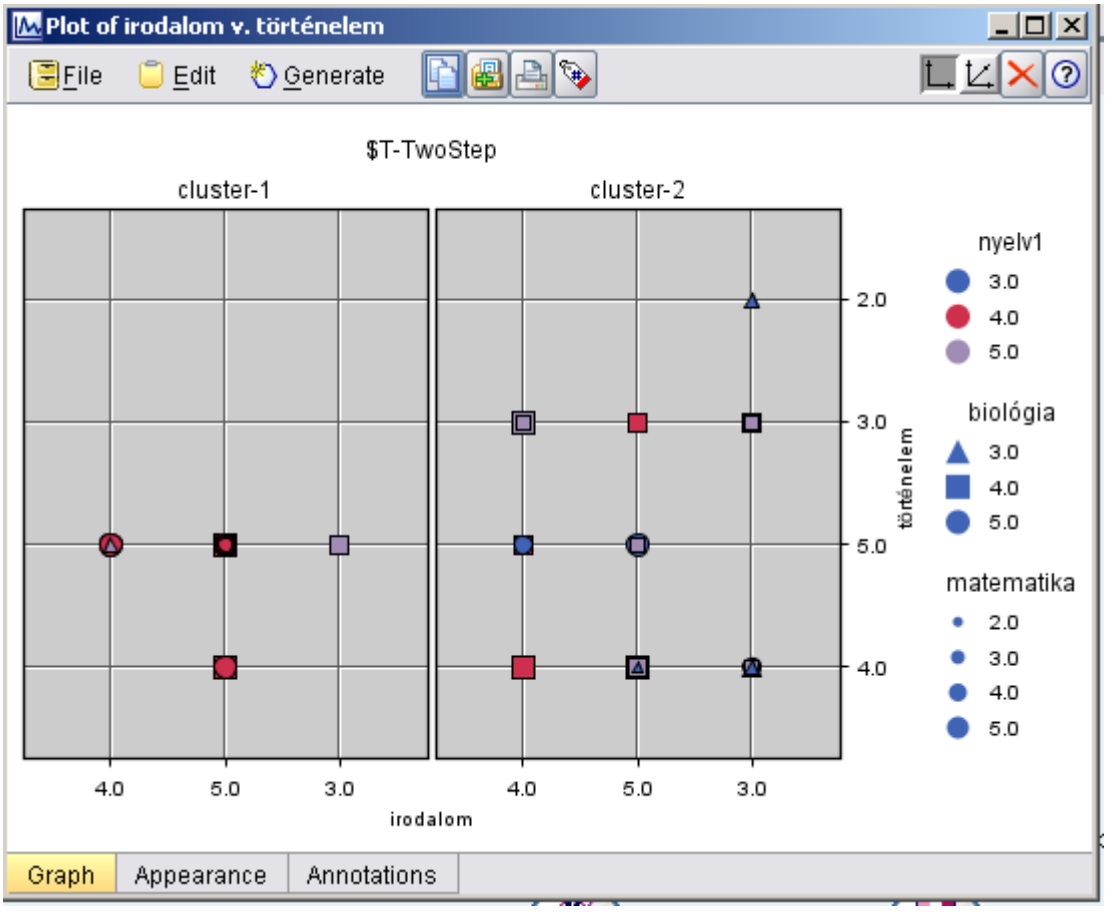
Egy grafikonnal szemléletesebbé tehetjük a klasztereket.

Illesszünk egy Plot node-ot a készmodell-node után!

Állítsuk be az alábbi ábrán látható változókat, az irodalom-történelem tengelyeket, a nyelv1 szerinti színezést, matematika szerinti méretezést és a biológia szerinti alakot, továbbá tegyük külön panelra a klaszterek mezői szerint az adatokat!



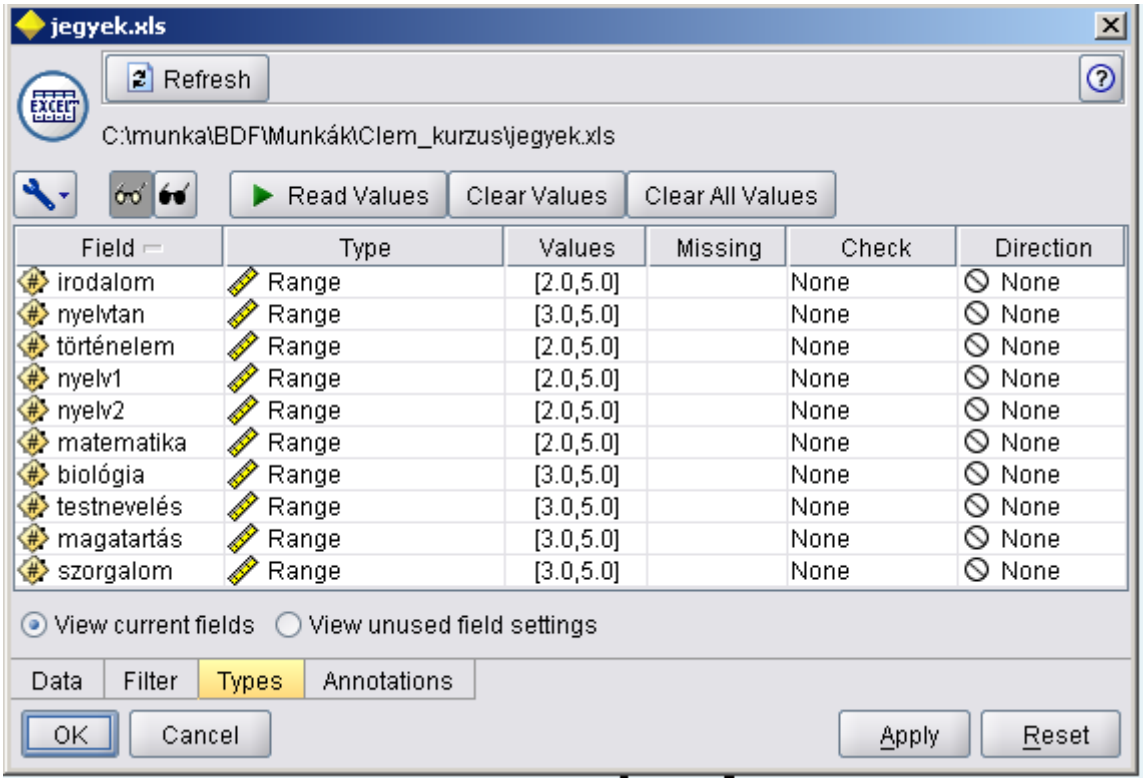
Futtassuk a Plot node-ot!



Rövid vizsgálódás után leolvasható, hogy milyen típusú tanulók kerültek az egyes klaszterekbe.

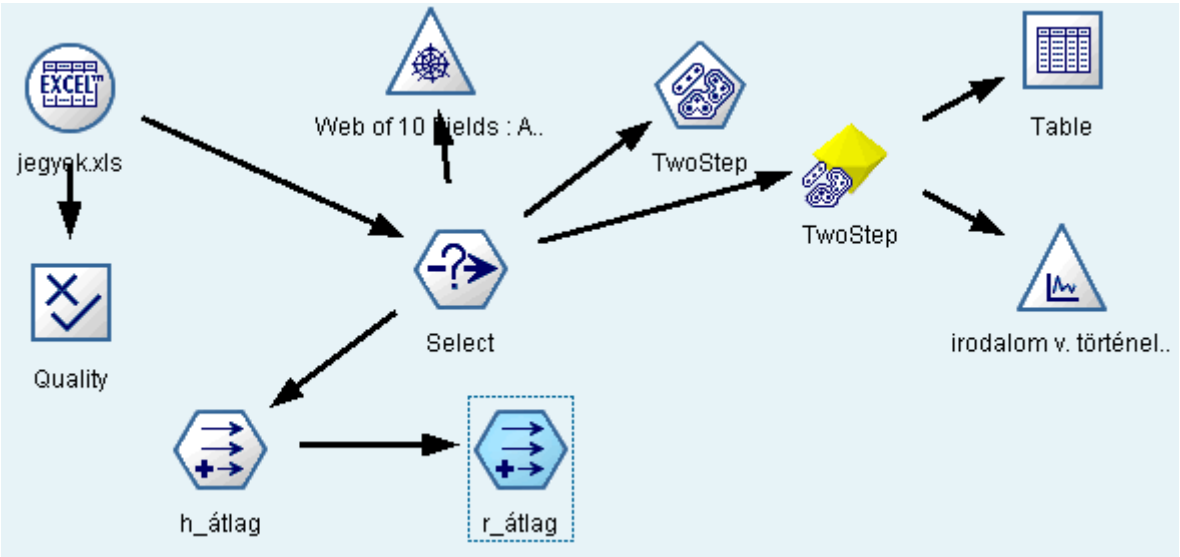
Igaz-e az a mindennapos vélekedés, hogy egy diáknak vagy humán vagy reál beállítottsága van? Ennek megfelelően készítünk új mezőket, s változtatunk a modellépítő node bemenő adatain is.

Az Excel (source) node TYPES **fülén így állítsunk be mindent:**



Itt minden változó *Range* típusú lett, minden változó *None* értékű, modellek számára nem bemenet, nem kimenet.

Két DERIVE NODE**-dal átlagoljuk a humán (*irodalom*, *nyelvtan*, *történelem*, *nyelv1*, *nyelv2*) és a reál (*matematika*, *biológia*) tantárgyakat, az új mezők neve legyen** *h_átlag* **és** *r_átlag*.



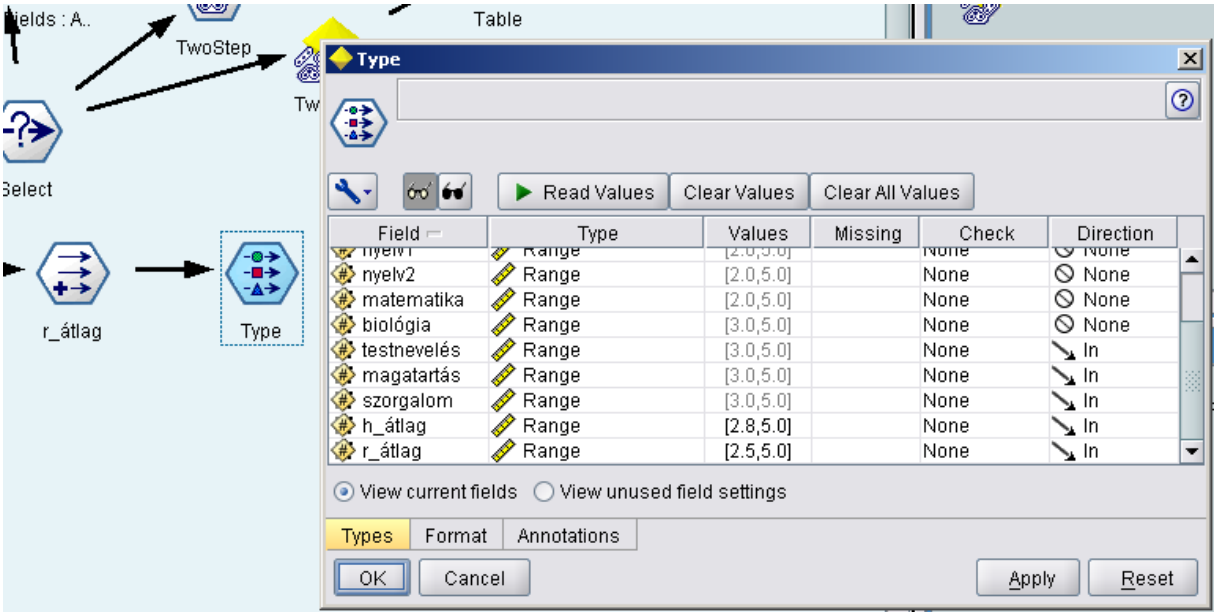
Az átlagokhoz szükséges kifejezések a két Derive node-ban a következők: (*irodalom+nyelvtan+történelem+nyelv1+nyelv2*)/5 illetve (*matematika + biológia*)/2

Mindkét node-ban a **FIELD TYPE** rovathoz **RANGE**-t adjunk meg.

Mivel átlagok szerepelnek, a stream properties-nél kapcsoljunk át egy tizedesjegy értékére!

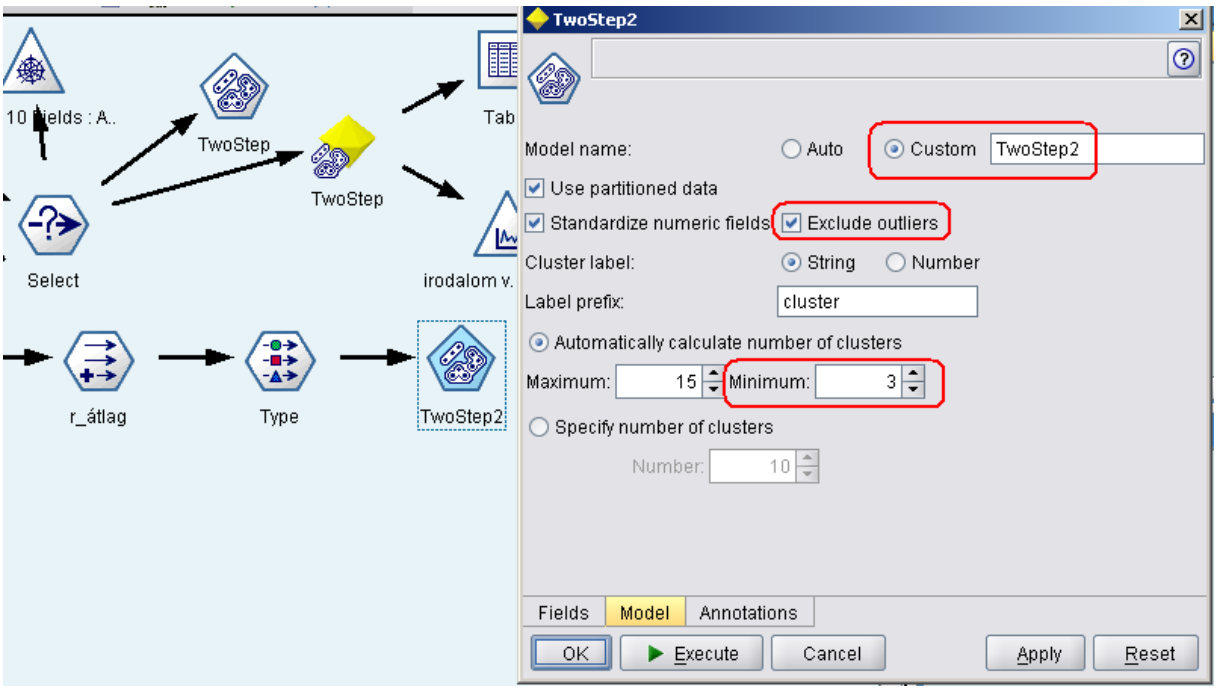
Ez azért kell, mert egészre kerekítve a táblázatokban és a grafikonokon nem jól látjuk majd az átlag szerinti különbségeket.

Szúrjunk be következő node-ként egy Type node-ot és állítsuk be az alábbi módon:



A számított tárgyátlagok egyikében sem szerepel a testnevelés, magatartás és szorgalom tárgy jegye, ezért ezeket is klaszterezési alapként tekintjük. Tehát a modellépítés számára a *testnevelés*, *magatartás*, *szorgalom*, *h_átlag*, *r_átlag* változókat adjuk meg bemenetként.

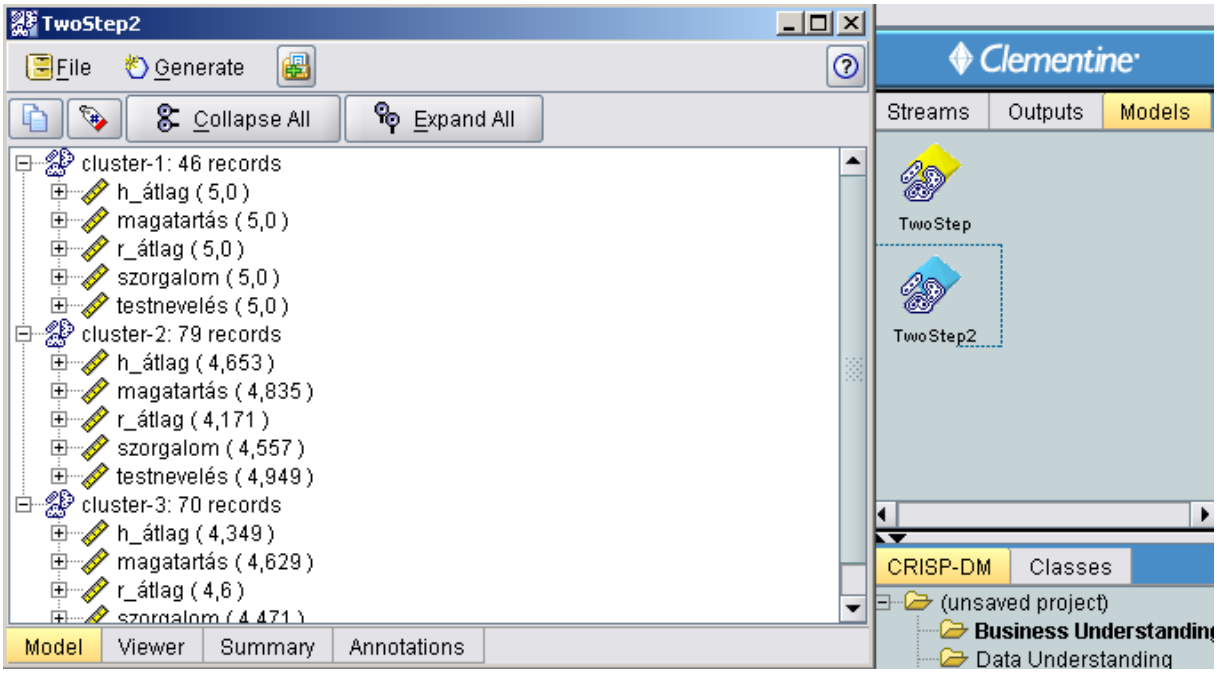
Ezután illesszünk be egy TwoStep modellépítő node-ot, s ebben az alábbi beállításokat tegyük:



Adjunk új nevet (TwoStep2) a modellépítő node-nak, különben lefuttatva felülírja az előző kész modellünket!

Jelöljük be az **EXECUTE OUTLIERS** négyzetet, hogy a „kilógó” esetek kizárásával keletkezzen az első lépés tömörített adathalmaza. A klaszterek számának minimumához írjunk 3-at, lehetőséget adva a humán és reálon kívül más csoportnak is, ha van.

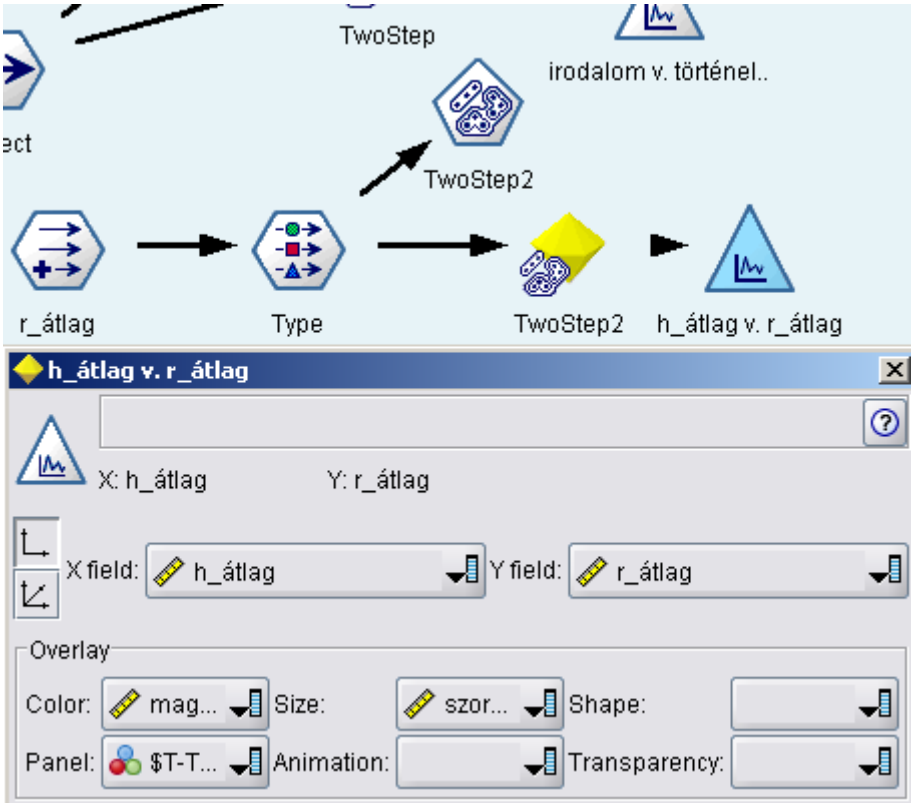
Futtassuk le a modellépítő Twostep2 node-ot!



Az itt látható „készmodellt” kapjuk. Az első klaszterbe a teljes kitűnők tartoznak, a második klaszterben a humán átlagok átlaga a jobb, a harmadikban pedig a reál átlagoké. Megfigyelhető, hogy real típusú változókkal dolgoztunk, s nem a klaszterbe eső rekordok attribútumainak százalékos megoszlását, hanem a klaszterátlagot írja ki minden változóra.

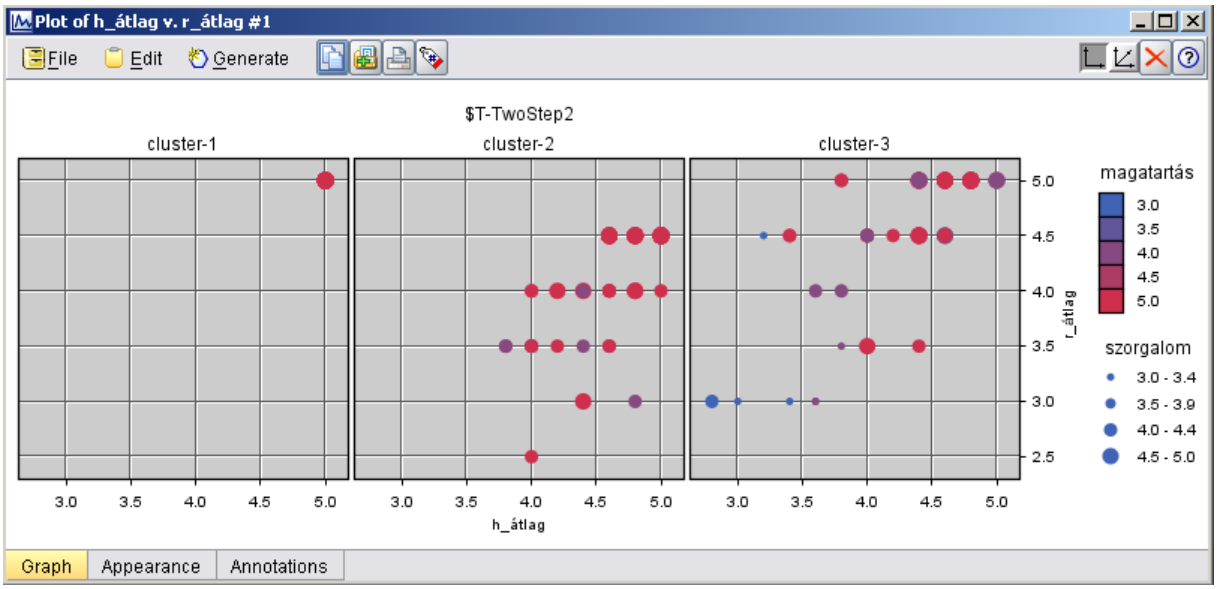
Emeljük be a kész modellt (vonszolással vagy duplaklikkel) a streambe, kössük a Type node után!

Tegyük szemléletessé egy Plot (graph típusú) node-dal az eredményt!



Mivel a jel-alakot csak Set típusú változóra lehet beállítani, ezért egy mezőt nem tudunk megjeleníteni, hagyjuk ki ezért a testnevelést.

Futtassuk a Plot node-ot!

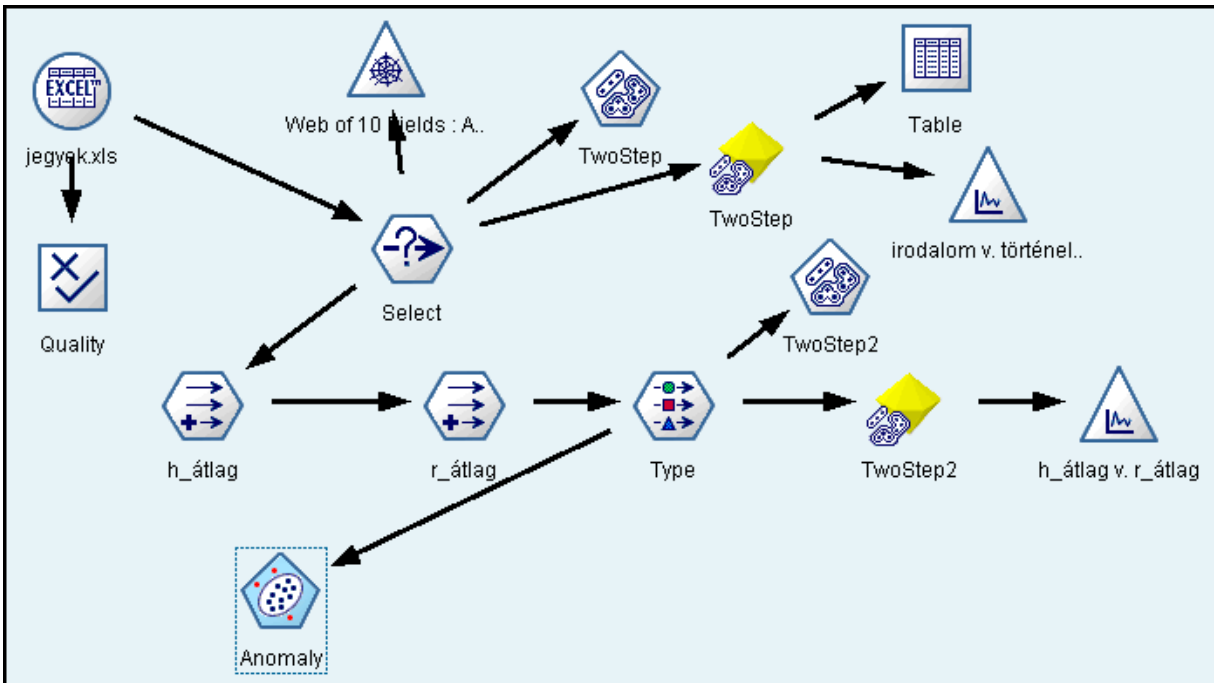


E koordináta-rendszerben a mellékátló feletti pontok olyan rekordokat jelentenek, melyekben a reálátlag jobb a humánnál, a mellékátló alatti pontoknál pedig ez épp fordítva van.

Az Anomaly node használata

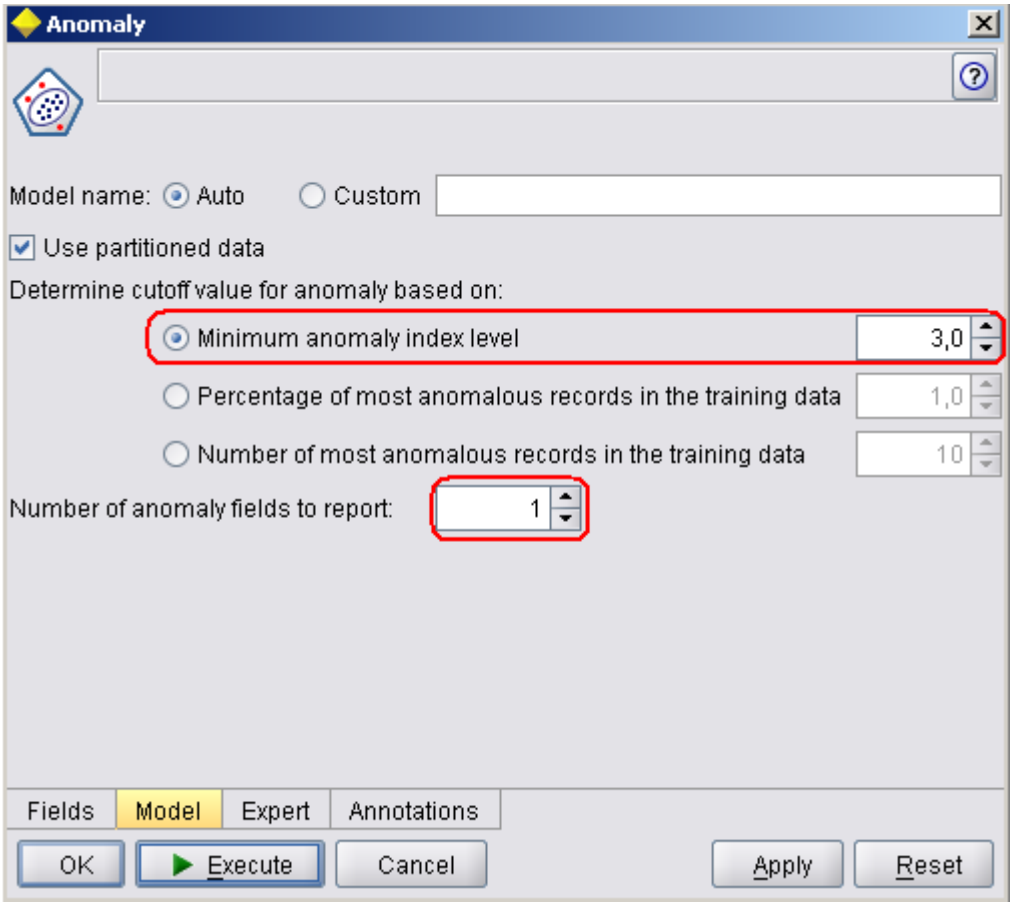
Következő célunk, hogy „szokatlan” rekordokat keressünk az Anomaly modellépítő node segítségével.

Illesszünk eddigi streamünkbe egy Anomaly modellépítő node-ot a meglévő Type node után!



Az Anomaly node bemenetei ugyanazok a mezők, melyeket a TwoStep node számára megadtunk. Ezen mezők adataiban fog „átlagostól eltérő" rekordokat keresni a modellünk.

Tegyük meg az Anomaly (modellépítő) node beállítópaneljének MODEL fülén az alábbi képen látható beállításokat!



MINIMUM ANOMALY INDEX LEVEL: itt megadhatjuk milyen „szabálytalanságindex" fölött tekintsen egy rekordot anomáliának, szabálytalannak. (Ez a jelzőszám a rekord csoportjának csoportátlagtól való átlagos eltéréséhez viszonyítja minden rekord saját csoportátlagától való eltérését, ha 2 fölött van az értéke, az már szabálytalanságra utalhat.)

PERCENTAGE OF...: Az összes rekord százalékában lehet megadni, hány szabálytalan rekordot adjon eredményül az eljárás (a legnagyobb szabálytalanságindexű rekordok).

NUMBER OF MOST ...: Pontos darabszámmal lehet megadni, hány szabálytalan rekordot adjon eredményül az eljárás.

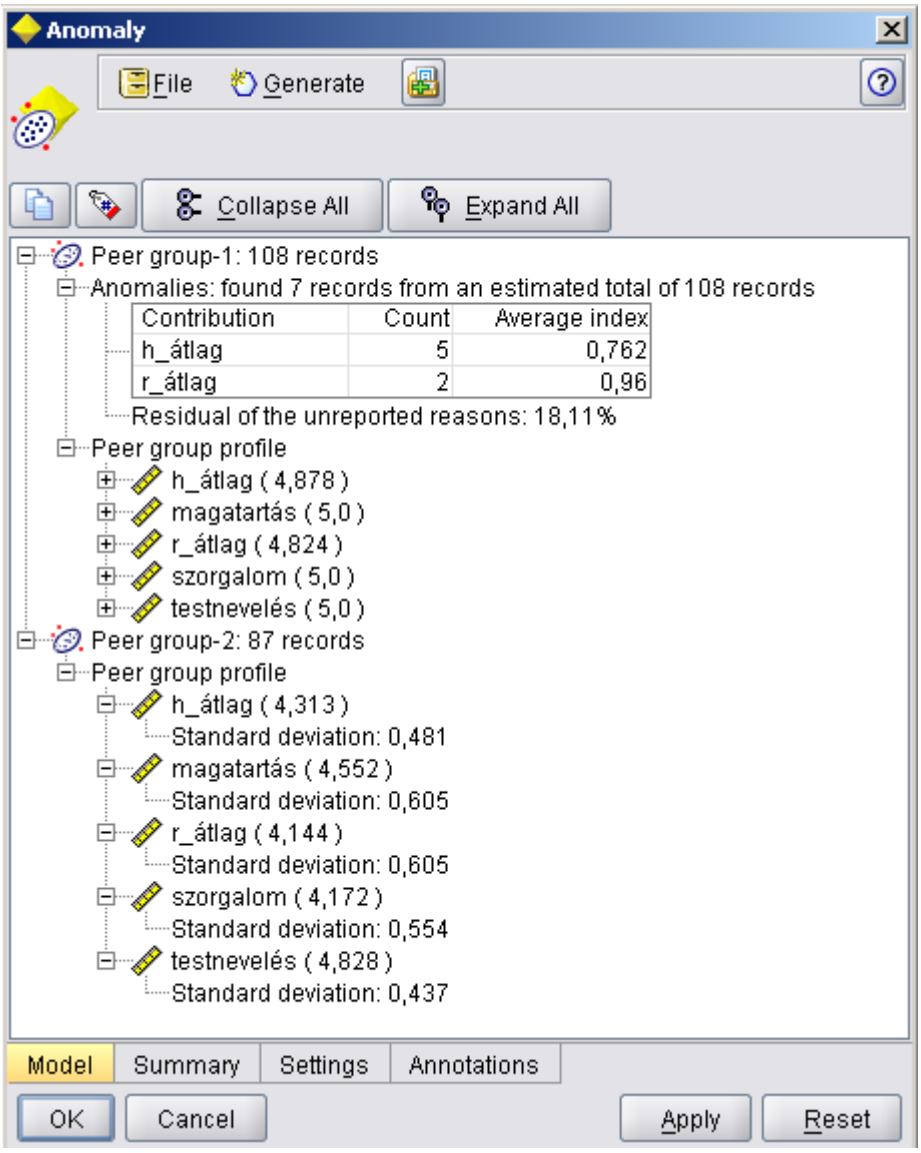
NUMBER OF ANOMALY FIELDS TO REPORT: A klaszterátlagtól legnagyobb eltérést mutató mezőket is jelzi a modell minden rekord esetén, mégpedig az eltérés mértékével együtt. Meg lehet adni, hány ilyen mezőt akarunk kiszámíttatni.

A gép által felkínált beállításokat két helyen kellett átállítani, a 3 fölötti szabálytalanságindex-értékű rekordokat jelöltessük meg anomáliának, illetve egyetlen mezőt adjunk meg csak a legnagyobb eltérés vizsgálatához.

Az adathalmaztól függ, mekkora anomáliaindexet tekintünk „szabálytalanságnak". Kevés változó esetén nem érdemes sok eltérés-változót kiíratni.

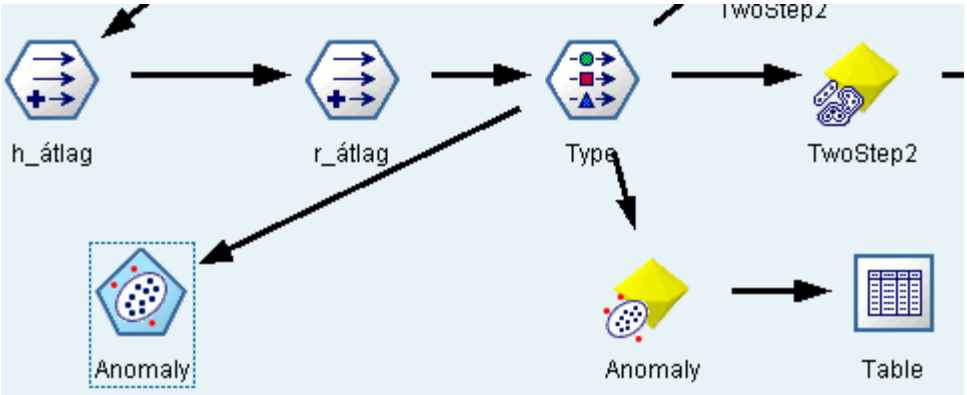
Futtassuk a beállított Anomaly modellépítő node-ot!

A futtatás egy kész modellt (gyémánt alak!) tesz a jobb felső ablakba, a **MODELS** fülre. Ennek jobbegérrel elérhető gyorsmenüjében a **BROWSE** menüpontra kattintva ezt látjuk:

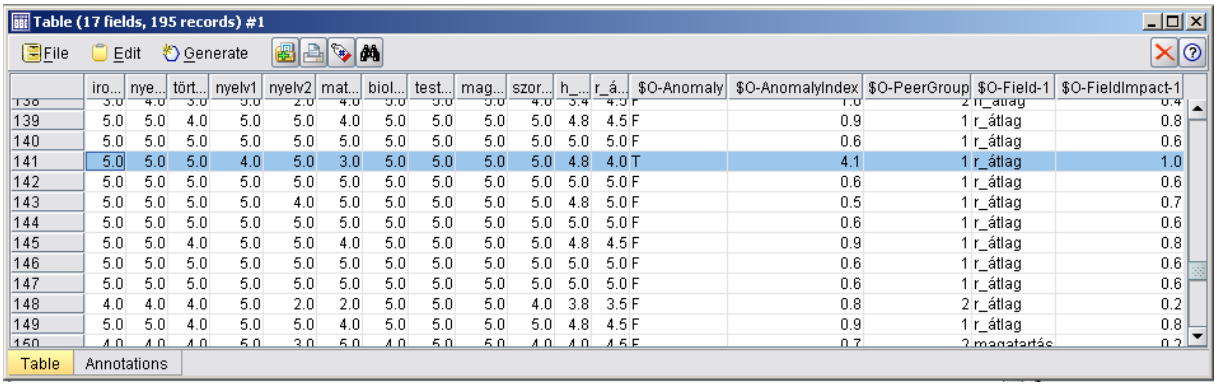


Hét szabálytalan rekordot talált a gép a megadott beállítások szerint. Ezek csoportjuktól való eltérése a *h_átlag* és a *r_átlag* mezőknél a legnagyobb. Láthatjuk a két klaszter (peer group) tulajdonságait is (zárójelben az átlagok, standard deviation = szórás). A további részleteket a hozzáfűzött új mezők tartalmazzák, a szinteket kibontva nézzük meg ezeket!

Szúrjuk be a keletkezett kész modell gyémánt alakú node-ját a Type node után, majd fűzzünk hozzá egy táblázatot outputként!



Futtassuk le a most beszúrt táblát!

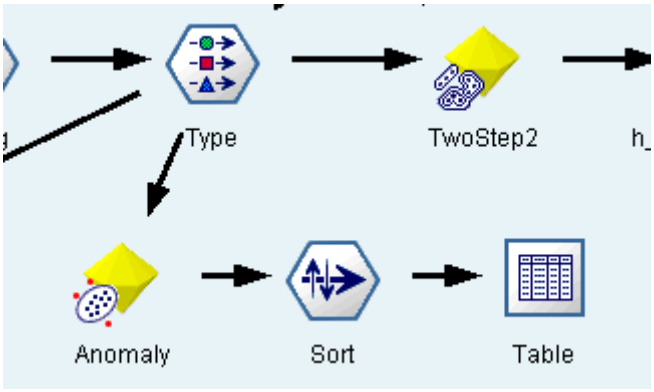


	iro...	nye...	tört...	nyelv1	nyelv2	mat...	biol...	test...	mag...	szor...	h_...	r_á...	$O-Anomaly	$O-AnomalyIndex	$O-PeerGroup	$O-Field-1	$O-FieldImpact-1
139	5.0	5.0	4.0	5.0	5.0	4.0	5.0	5.0	5.0	5.0	4.8	4.5	F	0.9	1	r_átlag	0.8
140	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	F	0.6	1	r_átlag	0.6
141	5.0	5.0	5.0	4.0	5.0	3.0	5.0	5.0	5.0	5.0	4.8	4.0	T	4.1	1	r_átlag	1.0
142	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	F	0.6	1	r_átlag	0.6
143	5.0	5.0	5.0	5.0	4.0	5.0	5.0	5.0	5.0	5.0	4.8	5.0	F	0.5	1	r_átlag	0.7
144	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	F	0.6	1	r_átlag	0.6
145	5.0	5.0	4.0	5.0	5.0	4.0	5.0	5.0	5.0	5.0	4.8	4.5	F	0.9	1	r_átlag	0.8
146	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	F	0.6	1	r_átlag	0.6
147	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	F	0.6	1	r_átlag	0.6
148	4.0	4.0	4.0	5.0	2.0	2.0	5.0	5.0	5.0	4.0	3.8	3.5	F	0.8	2	r_átlag	0.2
149	5.0	5.0	4.0	5.0	5.0	4.0	5.0	5.0	5.0	5.0	4.8	4.5	F	0.9	1	r_átlag	0.8

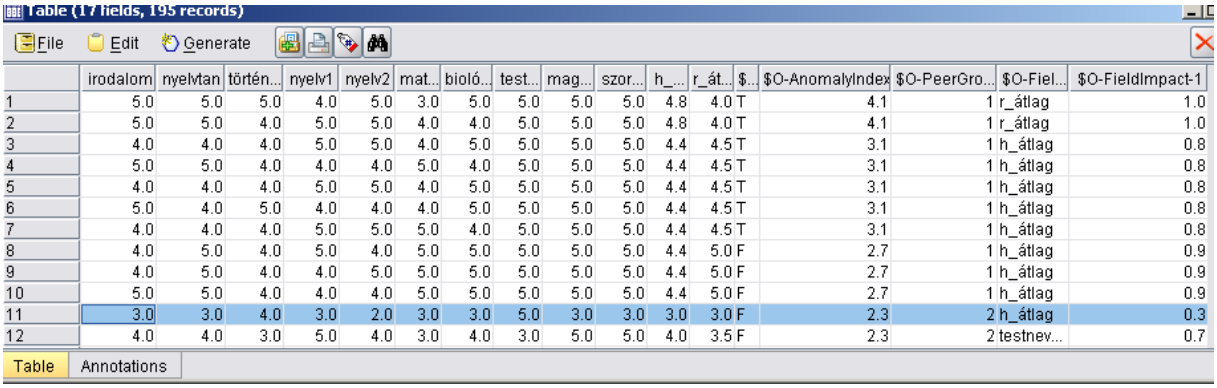
Például a kijelölt 141. rekord szabálytalanságindexe 4,1 , a saját klaszterében, az 1-es számúban a reál átlag rovatban tér el legjobban a saját klaszterétől.

Áttekinthetőbbé válnak a rendellenes rekordok és a csoportosítási logika, ha a $O-Anomalyindex mező alapján sorba rendezünk, és így nézzük át.

Szúrjunk a kész modell és a táblázat közé egy Sort node-ot, mely a $O-Anomalyindex szerint csökkenő sorrendbe rendezi a rekordokat!



Az eredmény:



	irodalom	nyelvtan	történ...	nyelv1	nyelv2	mat...	bioló...	test...	mag...	szor...	h_...	r_át...	$...	$O-AnomalyIndex	$O-PeerGro...	$O-Fiel...	$O-FieldImpact-1
1	5.0	5.0	5.0	4.0	5.0	3.0	5.0	5.0	5.0	5.0	4.8	4.0	T	4.1	1	r_átlag	1.0
2	5.0	5.0	4.0	5.0	5.0	4.0	4.0	5.0	5.0	5.0	4.8	4.0	T	4.1	1	r_átlag	1.0
3	4.0	4.0	4.0	5.0	5.0	4.0	5.0	5.0	5.0	5.0	4.4	4.5	T	3.1	1	h_átlag	0.8
4	5.0	5.0	4.0	4.0	4.0	5.0	4.0	5.0	5.0	5.0	4.4	4.5	T	3.1	1	h_átlag	0.8
5	4.0	4.0	4.0	5.0	5.0	4.0	5.0	5.0	5.0	5.0	4.4	4.5	T	3.1	1	h_átlag	0.8
6	5.0	4.0	5.0	4.0	4.0	4.0	5.0	5.0	5.0	5.0	4.4	4.5	T	3.1	1	h_átlag	0.8
7	4.0	4.0	4.0	5.0	5.0	5.0	4.0	5.0	5.0	5.0	4.4	4.5	T	3.1	1	h_átlag	0.8
8	4.0	5.0	4.0	5.0	4.0	5.0	5.0	5.0	5.0	5.0	4.4	5.0	F	2.7	1	h_átlag	0.9
9	4.0	5.0	4.0	4.0	5.0	5.0	5.0	5.0	5.0	5.0	4.4	5.0	F	2.7	1	h_átlag	0.9
10	5.0	5.0	4.0	4.0	4.0	5.0	5.0	5.0	5.0	5.0	4.4	5.0	F	2.7	1	h_átlag	0.9
11	3.0	3.0	4.0	3.0	2.0	3.0	3.0	5.0	3.0	3.0	3.0	3.0	F	2.3	2	h_átlag	0.3
12	4.0	4.0	3.0	5.0	4.0	3.0	4.0	3.0	5.0	5.0	4.0	3.5	F	2.3	2	testnev...	0.7

Látható, hogy 3 fölötti szabálytalanságindex csak 7 helyen van, érdekesek esetleg a még következő rekordok. Az első 10 anomália az első klaszterből „lóg ki”, míg a 2. számú clusterben a kiemelt 11. rekord a „legszabálytalanabb”.

<u>Önálló feladatok:</u>

Klaszterezzünk Two Step modellel egyszer csak a reál, egyszer pedig csak a humán tantárgyak szerint! Nézzük meg egy Plot grafikonon, mennyiben esnek egybe a kétféle klaszterek!

Keressük meg az öt „legszabálytalanabb” rekordot az összes tantárgy alapján! Figyeljük meg, van-e különbség a végeredményben, ha minden változót Discrete típusúnak állítunk be!

8. Neurális háló

Ez a gyakorlat arról szól, hogy egy bizonyos változó értékeit milyen adatbányászati algoritmusokkal lehet „jósolni” a többi változó értékei alapján.

Adataink egy SPSS fájlban vannak. Ennek mezői egy régebbi 6400 fős amerikai statisztikai felmérésben szereplő változók:

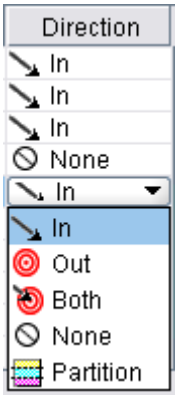
	Name	Type	Width	Decimals	Label	Values
1	kor	Numeric	4	0	Kor években	None
2	házas_e	Numeric	4	0	Házas-e	{0, Nem házas}...
3	lak_idő	Numeric	4	0	Hány éve lakik jelenlegi helyén	None
4	jövedelem	Numeric	8	2	Háztartás bevétele 1000$-ban	None
5	autó	Numeric	8	2	Elsődleges jármű értéke	None
6	autókat	Numeric	8	2	Elsődleges jármű kategóriája	{1,00, Gazdaságos}...
7	végzettség	Numeric	4	0	Végzettségi kategória	{1, Did not complete high school}...
8	munk_idő	Numeric	4	0	Hány éve dolgozik jelenlegi munkahelyén	None
9	nyugdíjas	Numeric	4	0	Nyugdíjas-e	None
10	elégedett	Numeric	4	0	Munkájával való elégedettsége	{1, Nagyon elégedetlen}...
11	neme	String	2	0	Neme	{f, Nő}...
12	együttélők	Numeric	4	0	Háztartásában élők száma	None
13	saját_tv	Numeric	4	0	Van-e saját tv-je	{0, Nem}...
14	saját_video	Numeric	4	0	Van-e saját videólejátszója	{0, Nem}...
15	saját_cd	Numeric	4	0	Van-e saját CD-lejátszója	{0, Nem}...
16	saját_pc	Numeric	4	0	Van-e saját számítógépe	{0, Nem}...
17	saját_fax	Numeric	4	0	Van-e saját faxa	{0, Nem}...

A felmérés adatai:

	kor	házas_e	lak_idő	jövedelem	autó	autó_kat	végzettség	munk_idő	nyugdíjas	elégedett	neme	együttélők	saját_tv	saját_video	saját_cd	saját_pc	saját_fax
1	55	1	12	72,000	36,2	3	1	23	0	5	f	4	1	1	1	0	0
2	56	0	29	153,000	76,9	3	1	35	0	4	m	1	1	1	1	0	0
3	28	1	9	28,000	13,7	1	3	4	0	3	f	3	1	1	1	1	0
4	24	1	4	26,000	12,5	1	4	0	0	1	m	3	1	1	1	1	1
5	25	0	2	23,000	11,3	1	2	5	0	2	m	2	1	1	1	0	0
6	45	1	9	76,000	37,2	3	3	13	0	2	m	2	1	1	1	1	0
7	42	0	19	40,000	19,8	2	3	10	0	2	m	1	1	1	1	0	0
8	35	0	15	57,000	28,2	2	2	1	0	1	f	1	1	1	1	1	0
9	46	0	26	24,000	12,2	1	1	11	0	5	f	2	1	1	1	0	0
10	34	1	0	89,000	46,1	3	3	12	0	4	m	6	1	1	1	0	1
11	55	1	17	72,000	35,5	3	3	2	0	3	f	2	1	1	1	1	0
12	28	0	3	24,000	11,8	1	4	4	0	5	m	1	1	0	0	1	1
13	31	1	9	40,000	21,3	2	4	0	0	2	f	4	1	1	0	0	0
14	42	0	8	137,000	68,9	3	3	3	0	1	f	1	1	1	1	1	0
15	35	0	8	70,000	34,1	3	3	9	0	4	m	3	1	1	1	1	0
16	52	1	24	159,000	78,9	3	4	16	0	5	m	2	1	1	1	1	1
17	21	1	1	37,000	18,6	2	3	0	0	1	m	7	1	1	1	1	1
18	32	0	0	28,000	13,7	1	1	2	0	4	f	2	1	1	1	0	1
19	42	0	9	109,000	54,7	3	3	20	0	3	f	1	1	1	1	1	0
20	40	1	12	117,000	58,3	3	2	19	0	5	f	4	1	1	1	0	0
21	30	0	3	23,000	11,8	1	1	3	0	3	m	1	1	0	0	0	0
22	48	0	14	21,000	9,50	1	3	2	0	3	m	1	1	1	0	1	0
23	39	1	17	17,000	8,50	1	4	2	0	3	m	5	1	1	1	1	1
24	42	1	5	34,000	16,6	2	2	13	0	3	f	4	1	1	1	1	0
25	45	1	12	115,000	57,4	3	1	27	0	4	f	5	1	1	1	0	0

Célváltozóval rendelkező adatbányász (tanuló) algoritmusok

A most következő modellekhez célváltozó(k)ra lesz szükség. Ezt/ezeket a Source node-ok **TYPE** fülén, illetve a Type node-ok beállításainál adhatjuk meg, a **DIRECTION** rovatban. Ha **OUT** típusú egy változó, akkor modellek célváltozója lesz (ha lehetséges), **IN** típusú változó bemenet, **BOTH** típusú mindkettő, **NONE** típusú pedig nem vesz részt a modellépítésben. Be lehet állítani egy **PARTITION** lehetőséget is. Ez azt dönti el, hogy az adott változó értékei a modellépítés és használat bizonyos fázisai számára adhat rekordokat.



A modellek tényleges használata ugyanis többlépcsős eljárás. Három vagy négy lépésben kaphatunk matematikailag megfelelően alátámasztott, használható eredményeket.

A lépések:

- modellépítés: A modellépítő node használata egy kiválasztott rekordhalmaz alapján.
- tesztelés: A kész modell vizsgálata, finomítása egy másik rekordhalmaz alapján, a modell analizálása. Ha nem megfelelőek az eredmények, újra az előző lépéshez kell visszatérni, új modellt kell építeni.
- érvényesítés: A tesztelt, véglegesre alakított modell lefuttatása újabb adathalmazokon, ha kevéssé eltérő eredményeket kapunk, akkor használható a kész modell, ha nagy az eltérés, az első lépéshez térünk vissza.
- tényleges használat: Újabban keletkező adathalmazokra, hasonló adathalmazokban előrejelzésre használható a kész modell, amíg valamilyen új tényező nem befolyásolja a megfigyelt objektumok, személyek viselkedését.

A **PARTITION** lehetőség egy olyan set típusú változót jelez, melynek két vagy három értéke kijelöli, hogy mely rekordok tartoznak az első, a második illetve a harmadik lépés adathalmazához.

Ha egy modellépítő node-nál az **USE PARTITIONED DATA** lehetőséget beállítjuk**,** a modellépítés, tesztelés vagy érvényesítés céljára megadott rekordhalmazokból az előre beállítottakkal fog dolgozni a modell.

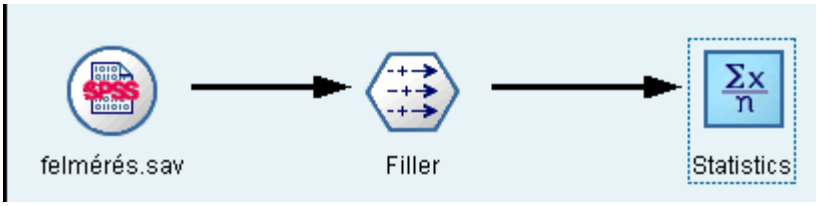
Statistics node használata

Alapvető statisztikákat számít a megadott mezőkből. Az adott felmérésben több változóról gyaníthatjuk, hogy nem lényeges információt hordoz, pl. sejthető, hogy Amerikában saját televízióval szinte mindenki rendelkezik.

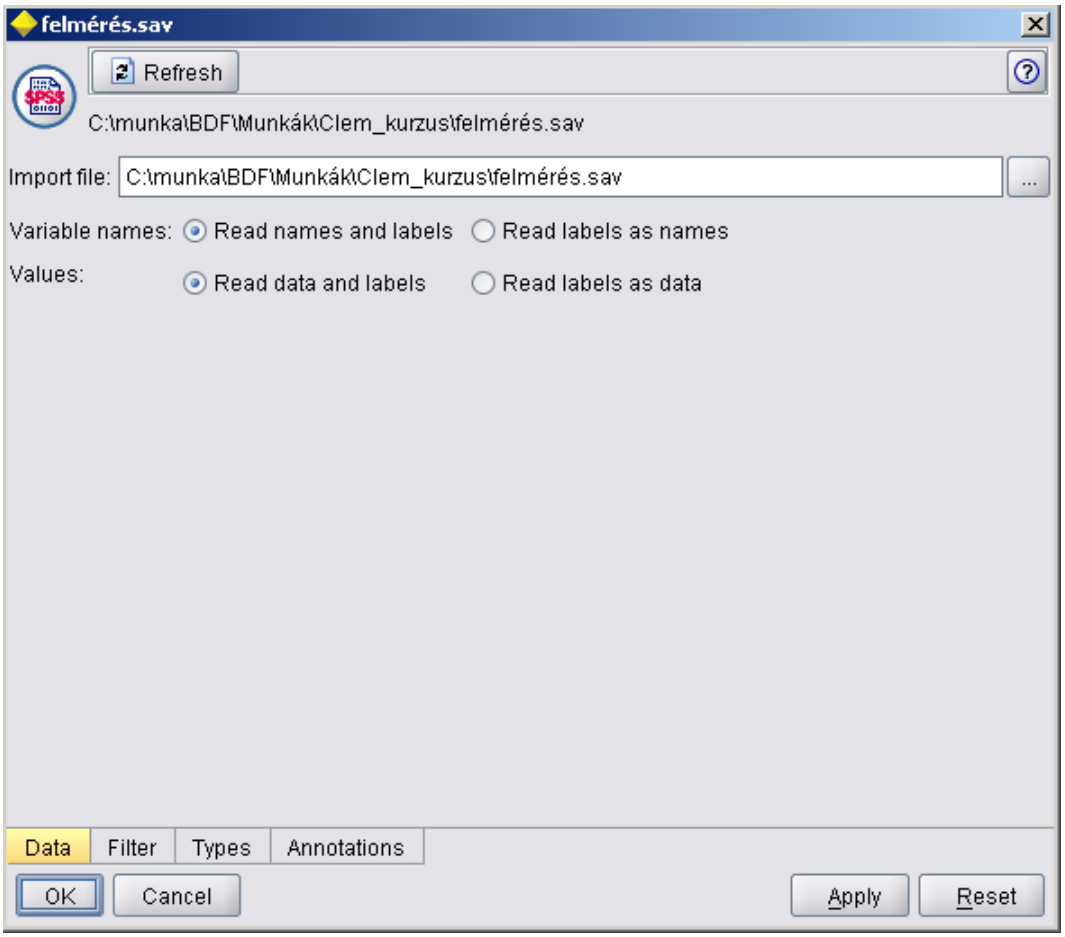
Készítsünk egy Source node-ot a felmérés.sav adathalmaz beolvasásához!

Írjuk át a nemnél található „f” és „m” stringeket a magyar „n” és „f”rövidítésekre!

Illesszünk be egy statistics node-ot ezek után!

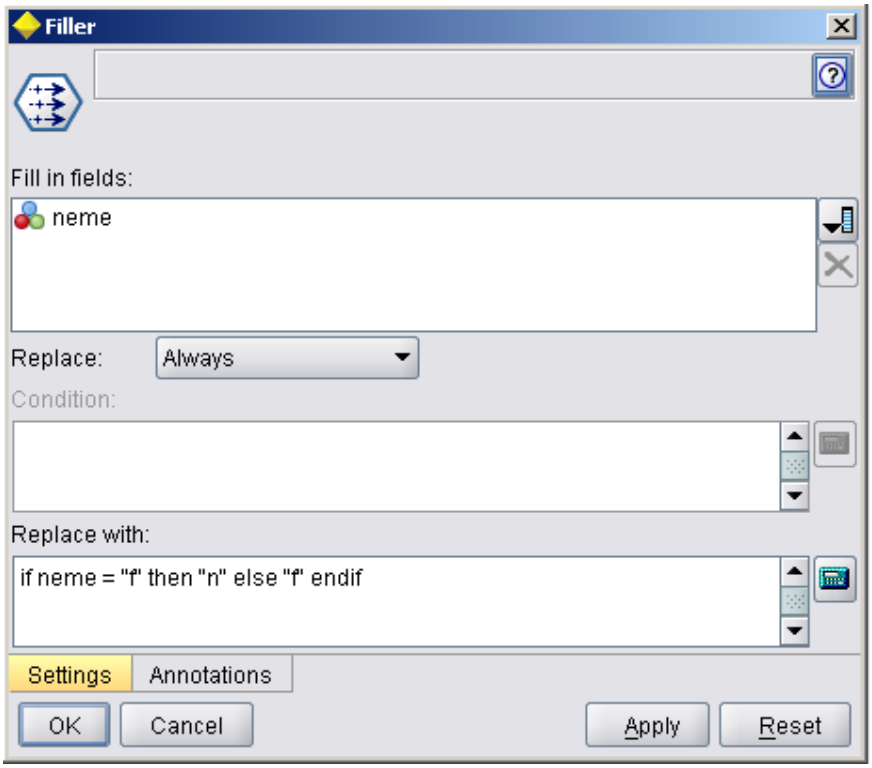


A source node beállításai:



A **FILTER** és a **TYPES** fület később töltjük ki, ha előzetes információk állnak rendelkezésünkre az adatokról.

A Filler node beállításai (az angol nyelvben a *female* és a *male* megfelelői a *férfi* és a *nő*):



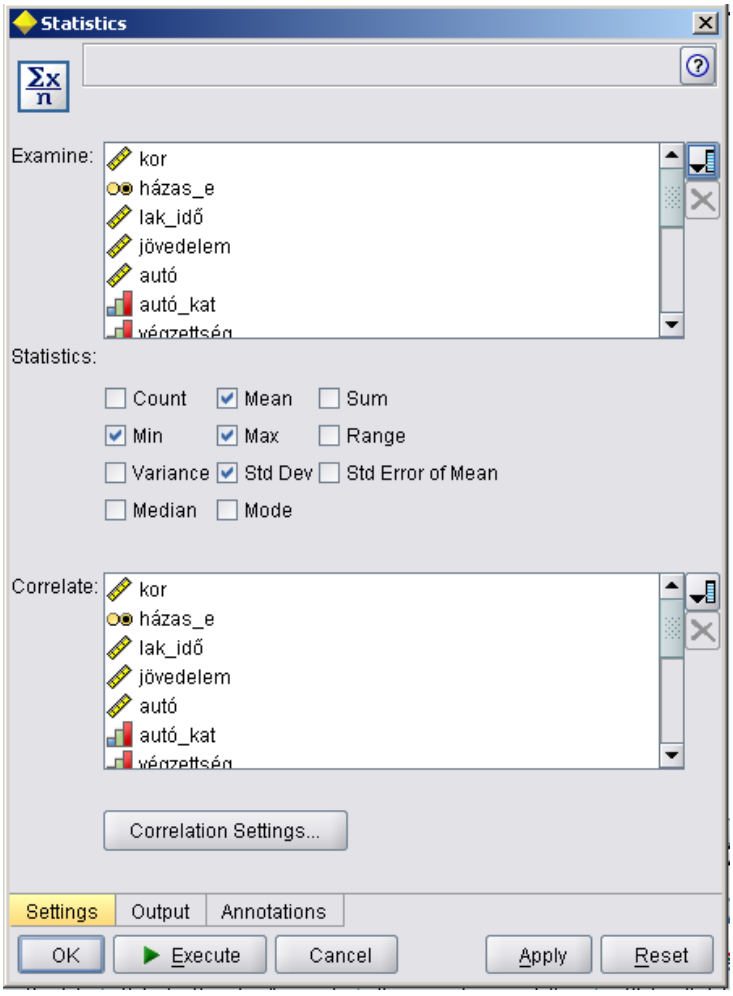
Az előkészületek után végezhetnénk adatminőség-vizsgálatot egy Quality node-dal, de most ez felesleges, mivel tudjuk, hogy minden rekord minden adata ki van töltve.

A Statistics (output) node a változókról szóló statisztikákat és korrelációs együtthatókat mutat – *csak numerikus tárolású adatokra*.

EXAMINE: vizsgált változók

STATISTICS: Milyen fajta statisztikákat (jellemzőket) mutasson az egyes változókról:

- **COUNT:** adatok száma
- **MEAN:** átlag
- **SUM:** adatok összege
- **MIN:** legkisebb adat
- **MAX:** legnagyobb adat
- **RANGE:** az előző kettő különbsége, terjedelem
- **VARIANCE:** szórásnégyzet (variancia)
- **STD DEV:** az előző négyzetgyöke, szórásnégyzet

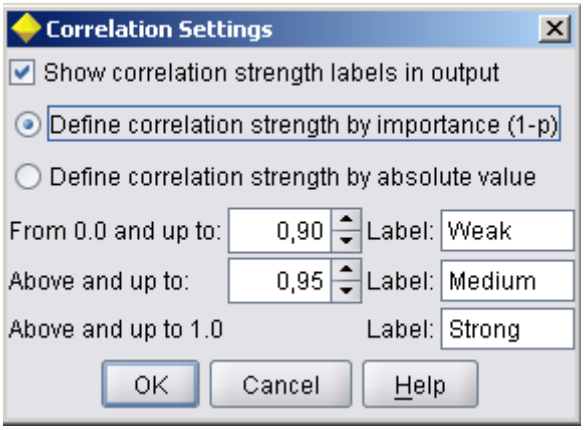


- **STD ERROR OF MEAN:** az átlag standard hibája
- **MEDIAN:** medián
- **MODE:** módusz

CORRELATE: Korrelációszámítás

Itt meg lehet adni, hogy a felső listán felsorolt változókhoz milyen más változókra nézve számítson korrelációs együtthatókat a Modeler.

A **CORRELATION SETTINGS** gombbal a korrelációszámítás beállításai jelennek meg:

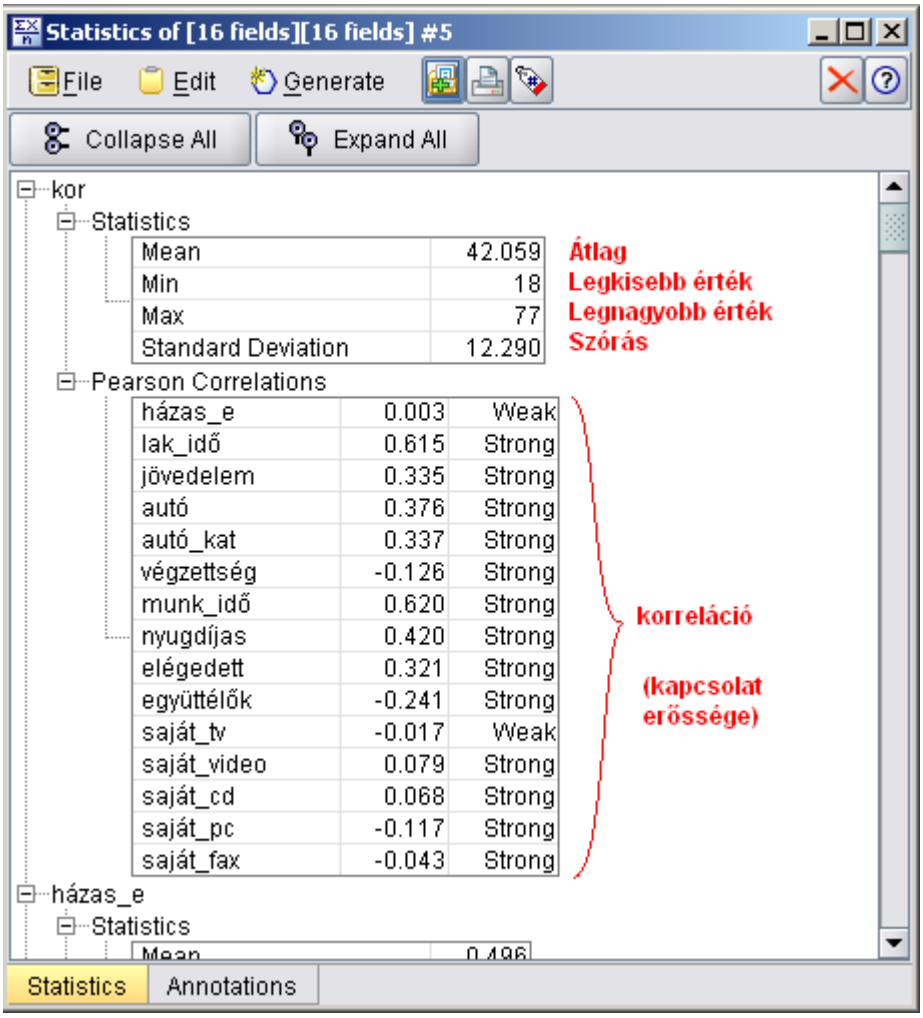


Itt a Pearson-féle korrelációs együtthatók beállításai láthatók. Ez a mérőszám a változók *lineáris kapcsolatának erősségét méri* egy -1 és 1 közé eső számmal. A változók függetlenségének mértékét az mutatja, hogy e szám mennyire közeli a 0-hoz. A (pozitív vagy negatív irányú) lineáris kapcsolat erős, ha abszolútértéke kevéssé tér el az 1-től. Ezt a beállítást lehet bekapcsolni a **DEFINE CORRELATION STRENGTH BY IMPORTANCE** részen, a linearitás előjeles mértékét pedig a **DEFINE CORRELATION STRENGTH BY ABSOLUTE VALUE** részen.

Az alatta látható pörgetőnyilaknál azt lehet szabályozni, hogy milyen mértékű kapcsolat esetén nevezze erősnek (strong), közepesnek (medium) vagy gyengének (weak) a változót.

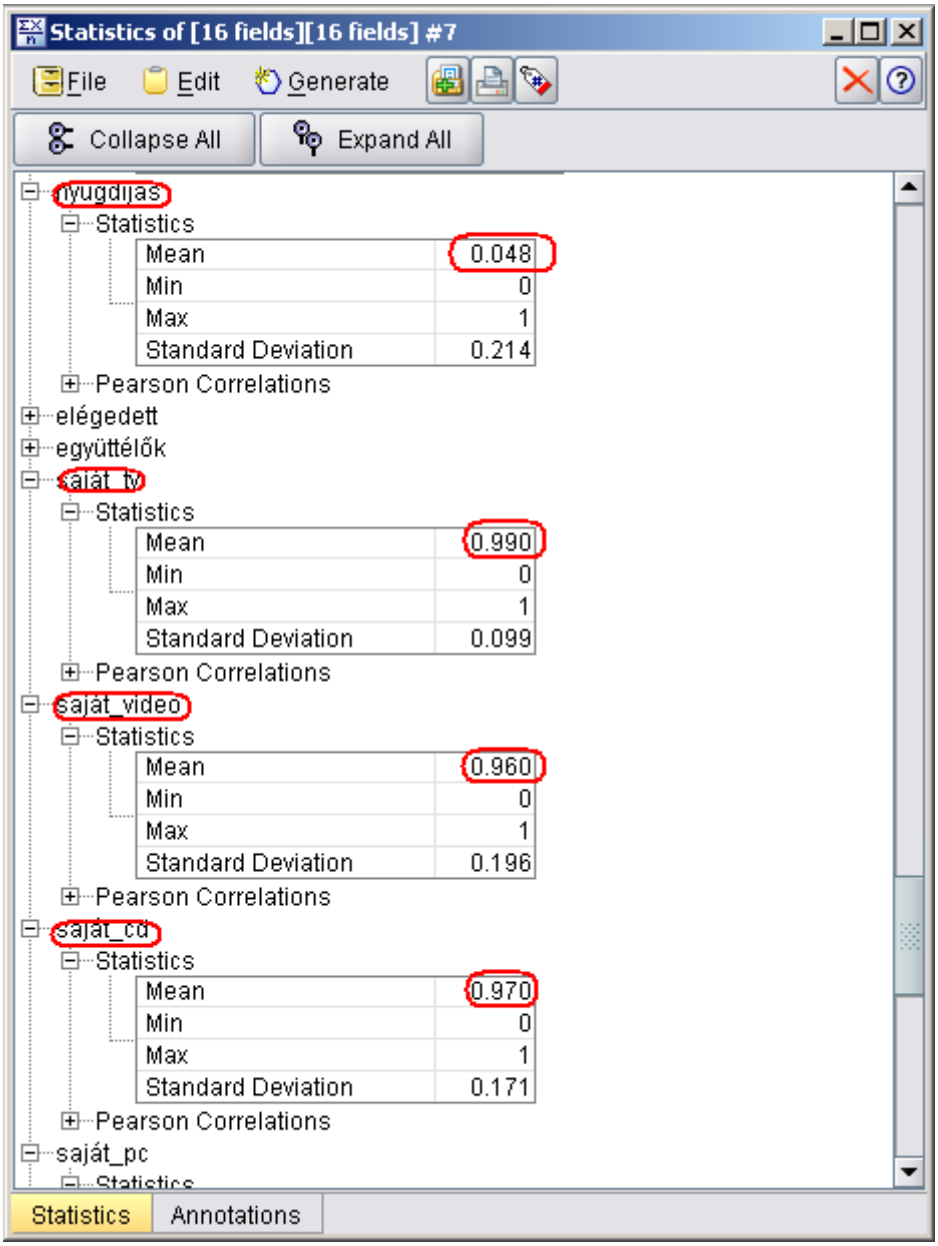
Állítsuk be a Statistics Node fent látható beállításait! (Minden felkínált mezőt adjunk meg a felső és az alsó listán is!)

Futtassuk le és elemezzük a végeredményt!



Alapos tanulmányozás után a következő megfigyeléseket tehetjük:

Mivel a *nyugdíjas* és a *saját_tv*, *saját_video*, *saját_cd* változók értéke csak 0 és 1 lehet, átlaguk értéke arra utal, hogy bennük az adatok több mint 95%-a azonos (az első mezőnél több mint 95% a 0, a többinél pedig az 1-es van túlsúlyban). Így ezek a mezők jelentősen nem különböztethetnek meg adatokat, ezért nem fognak indokot adni más változók értékeinek eltéréseire. Ezeket a mezőket a source node **FILTER** fülén célszerű kikapcsolni.

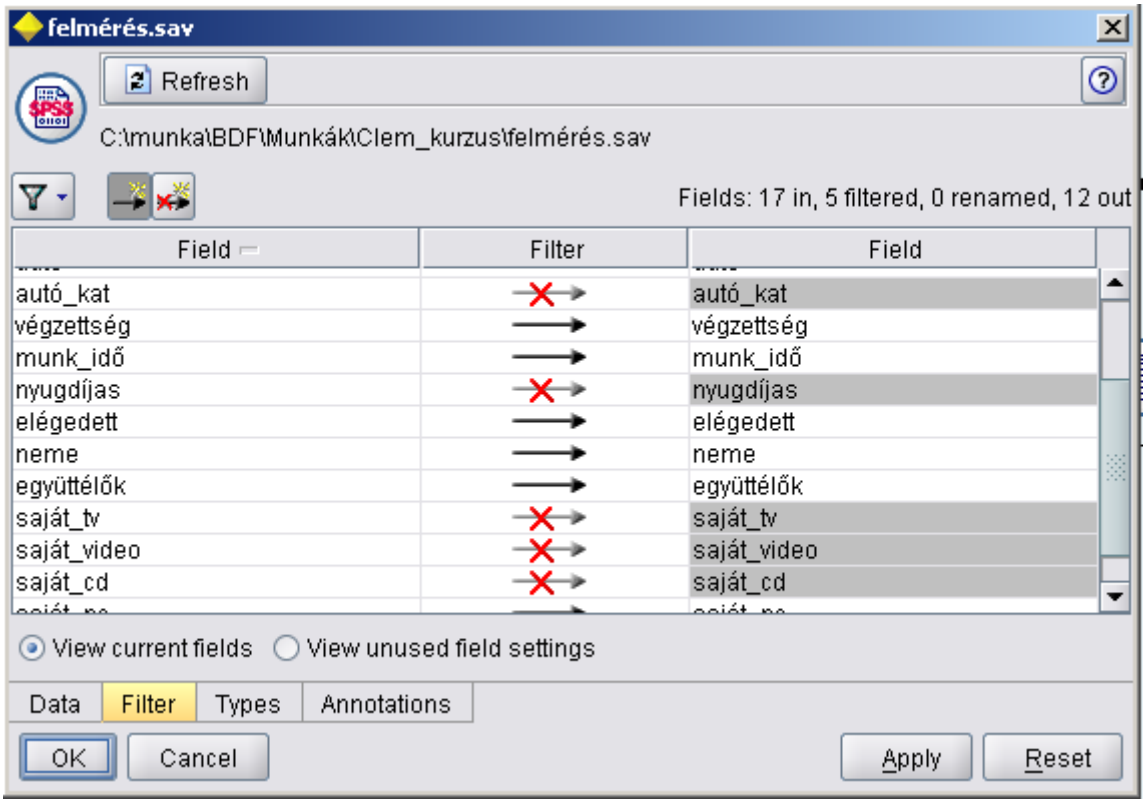


Kapcsoljuk ki a fenti négy mezőt a Source node-ban a Filter fülön!

Az autókategóriák használatát is fontolóra vehetjük, hiszen az autók értékei adják az autókategóriákat, tehát az autókategóriák információja az autók értékének információjából származik. Erre utal a két változó 86% erősségű korrelációja is, azaz majdnem 90%-os a linearitás a két változó között. Ha nem akarunk felesleges (duplikált) információból számolni, akkor el kell hagyni az autókategória mezőt is. (Az információk egyszerűbbé tétele céljából lehetne esetleg csak ezt benn hagyni, s az autóértéket kikapcsolni.)

Kapcsoljuk ki az autókategóriák mezőjét is a Source node Filter fülén!

Ez lesz a végeredmény:



Feature selection node

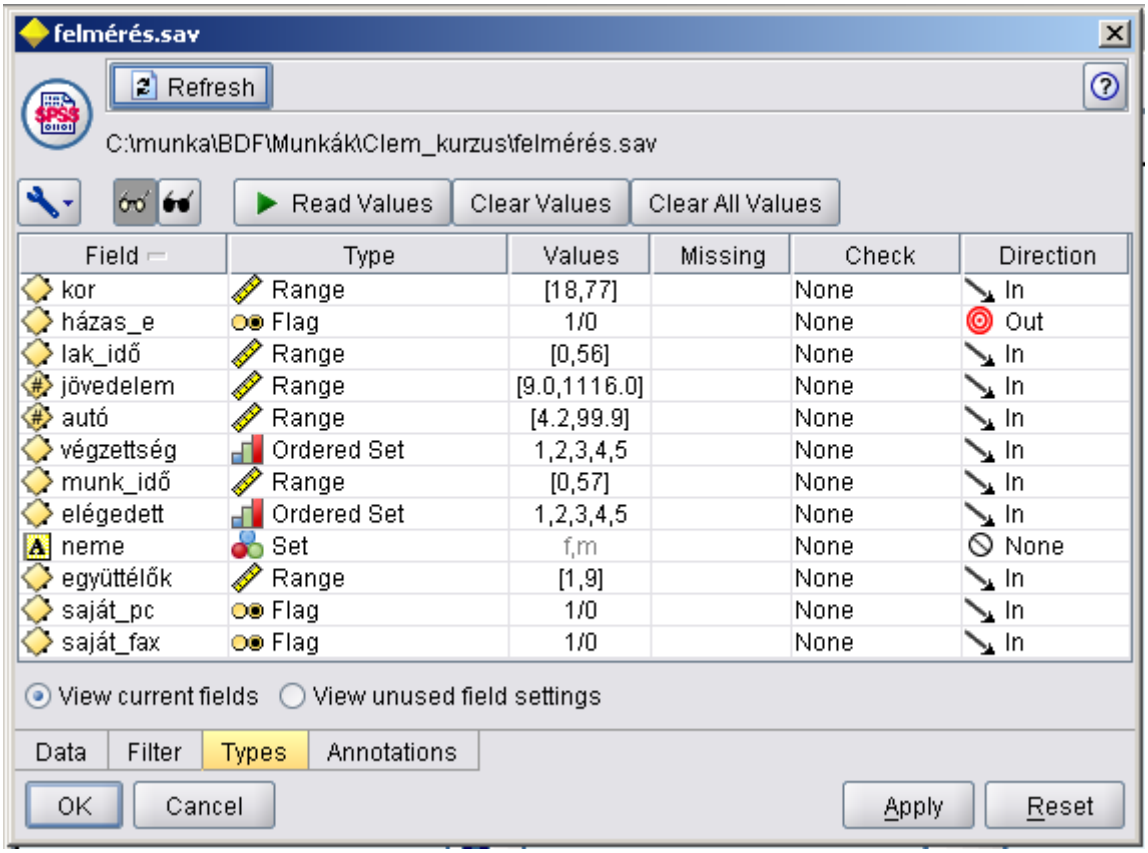
Arra szeretnénk választ kapni, hogy bizonyos változók adatai mennyire határozzák meg egy konkrét változó adatait (ehhez mindig egy konkrét célváltozóra van szükség). Mivel előfordulhat, hogy akár több száz változó szerepel egy adathalmazban ezért gyakran célszerű leszűkíteni, mely változók adatai jöhetnek szóba. A Feature selection node-nak az a célja, hogy az adott célváltozó szempontjából lényeges bemeneti változókat felderítse.

Legyen kitűzött cél, hogy a többi változók adataiból „jósoljuk" meg a *házas_e*, *jövedelem*, *elégedett*, *saját_pc* változók értékét. Természetesen mindegyik változóhoz külön vizsgálat szükséges.

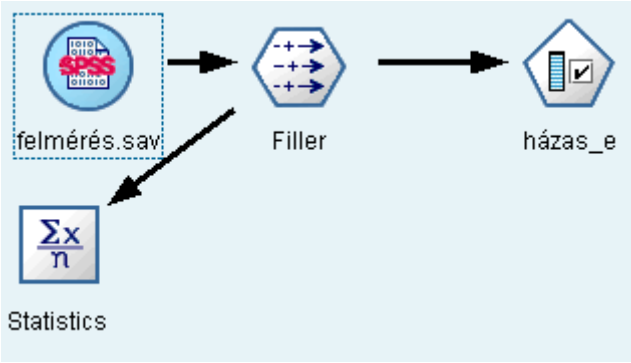
Kezdjük a házas_e változóval!

Adjuk meg a source node types fülén a *házas_e* változót kimenetnek, a többit bemenetnek!

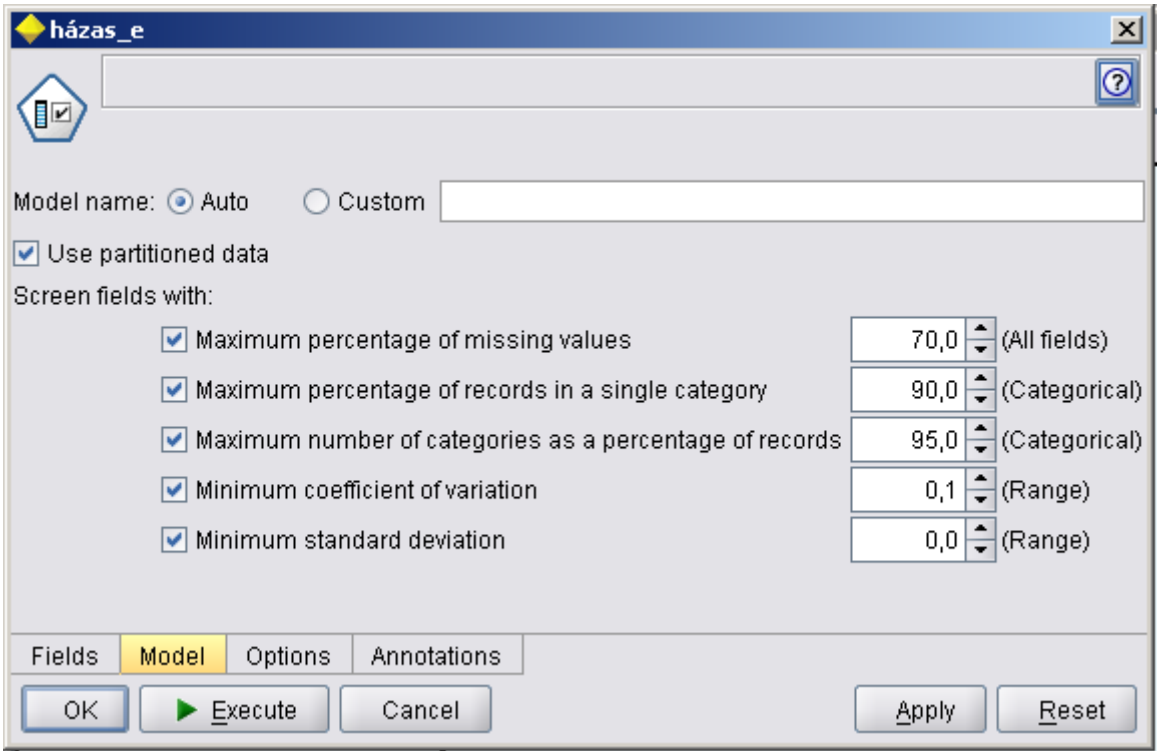
Kapcsoljuk ki teljesen a modellezésből a *neme* mezőt, ez szöveges adatot tartalmaz, a matematikai számítások nem tudják használni.



Fűzzünk egy Feature selection node-ot a Filler node után!



Nézzük meg a beillesztett node-on a Model fület!



MAXIMUM PERCENTAGE OF MISSING VALUES. Nem végez számításokat még azokra a változókra, amelyekben az itt megadott százalékérték feletti mennyiségű hiányzó adat van.

MAXIMUM PERCENTAGE OF RECORDS IN A SINGLE CATEGORY. Nem végez számításokat még azokra a diszkrét adatokat tartalmazó változókra, amelyekben az itt megadott százaléknál több azonos értékű adat van. (pl. a *„neme”* változó ilyen, ha szinte minden rekord nő.)

MAXIMUM NUMBER OF CATEGORIES AS A PERCENTAGE OF RECORDS. Nem végez számításokat még azokra a diszkrét adatokat tartalmazó változókra, amelyekben az itt megadott százaléknál több az eltérő adatok száma. (pl. ha autómárka nevű változóban ahány rekord, majdnem annyiféle autómárka van).

MINIMUM COEFFICIENT OF VARIATION. Nem végez számításokat még azokra a numerikus, Range típusú adatokat tartalmazó változókra, amelyekben az itt megadott érték alatt van a relatív szórás (szórás/átlag) értéke. (Kiszűrhetjük azokat a numerikus változókat, melyek nem lényegesen eltérő adatokat tartalmaznak.)

MINIMUM STANDARD DEVIATION. Nem végez számításokat még azokra a numerikus, Range típusú adatokat tartalmazó változókra, amelyekben az itt megadott értéknél kisebb a szórás. (Hasonló a célja, mint az előző beállításnak.)

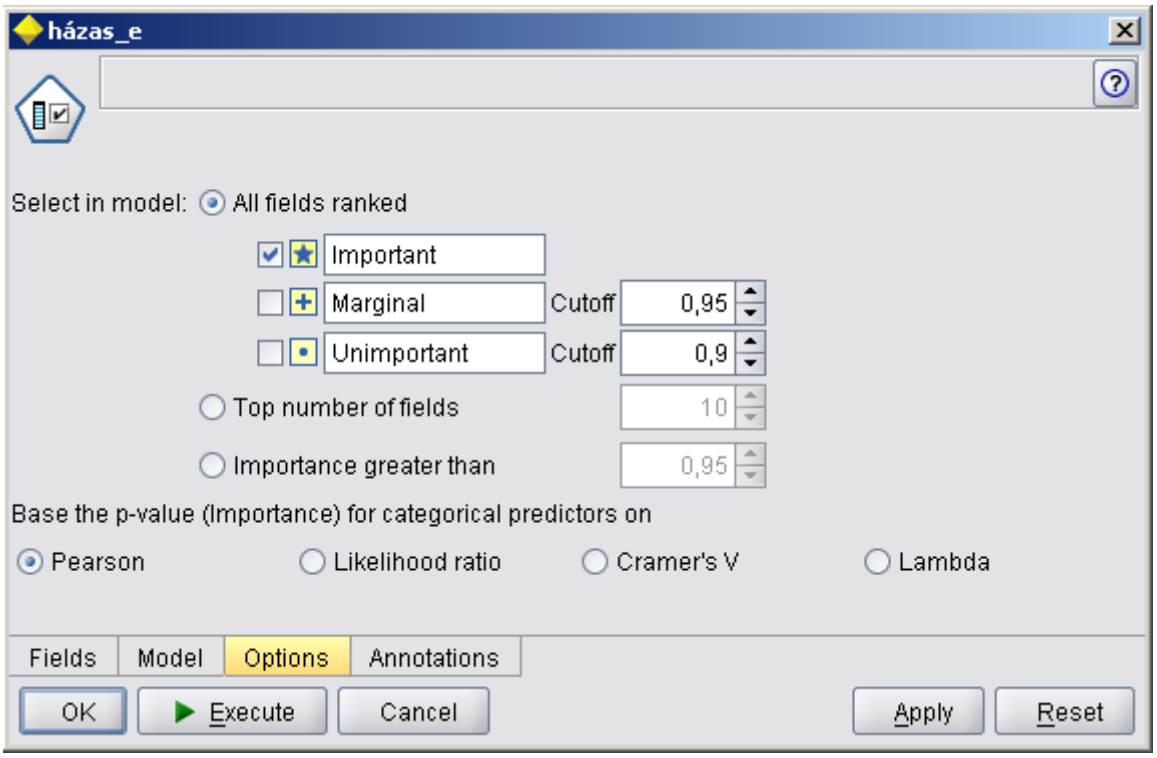
Vessünk egy pillantást az Options fülre!

Itt a bemenő változók fontosság szerinti rangsorával kapcsolatban tehetünk beállításokat.

A célváltozó szerinti „fontosságot” matematikai számítások adják, s meg lehet adni, hogy melyeket jelölje a gép **IMPORTANT** (fontos), **MARGINAL** (határeset) és **UNIMPORTANT**

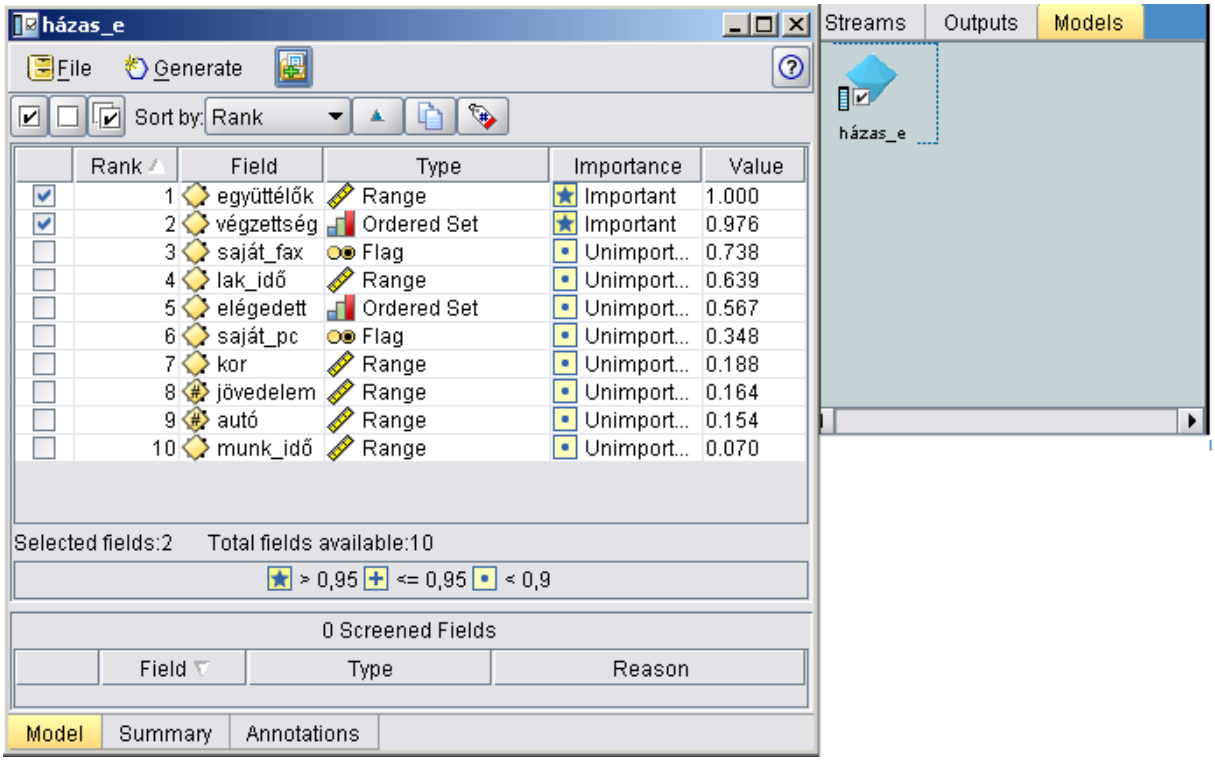
(nem fontos) címkével. (A fontosság százalékban kapott érték.) Beállítható a matematikai módszer is, de ez csak igen nagy adathalmaz esetén jelent némi eltérést.

*A „fontosság" csak lehetőséget jelent, pontosabban csak annyit, hogy a **nem fontosnak jelölt változók biztosan nem tartalmaznak információt**, „jóslatot" a célváltozóra vonatkozóan. Megeshet, hogy a fontosnak mutatott változók sem tartalmaznak elég információt a célváltozóhoz.*



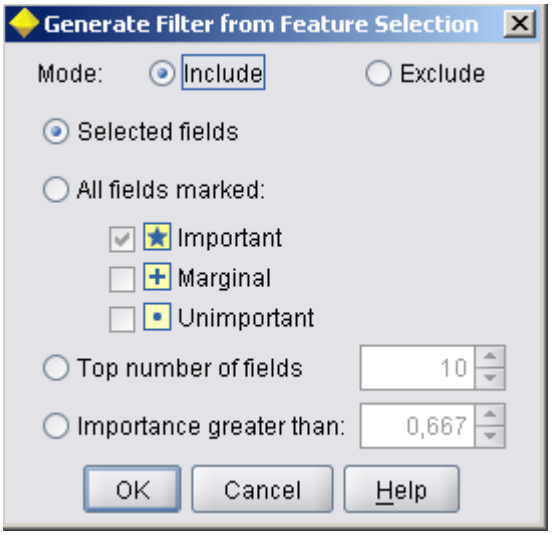
Futtassuk a Feature Selection-t!

A kapott kész modell a következőket mutatja:



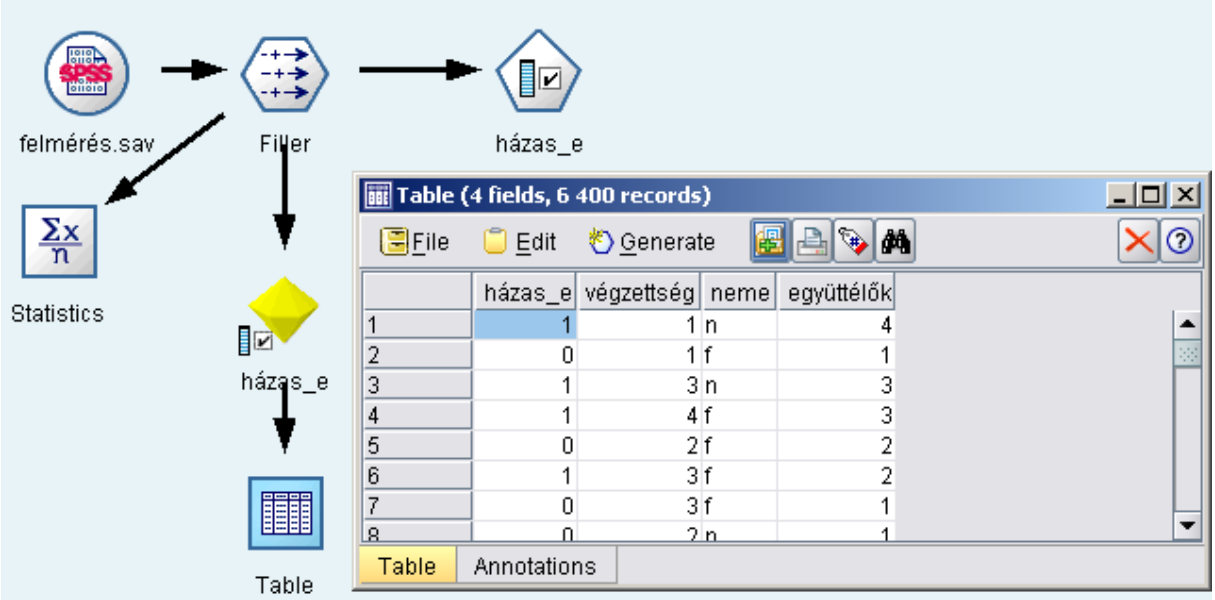
Látható, hogy két fontos változó van, a többi lényegtelen a *házas_e* célváltozó szempontjából. Ha a kész Feature selection modellt beszúrjuk, akkor filter node-ként fog viselkedni. Azokat a változókat fogja kiválasztani, melyek jelölőnégyzetében pipa van. Alapértelmezésben ezek a fontosként mutatott változók.

A párbeszédpanel tetején látható Generate gombra kattintva azt is megtehetjük, hogy Filter node-ot generálunk, mely csak a megjelölt változókat adja eredményül a célváltozóval együtt:



Illesszük be a Filler node után a kész modellt és egy táblát!

Futtassuk a táblát!

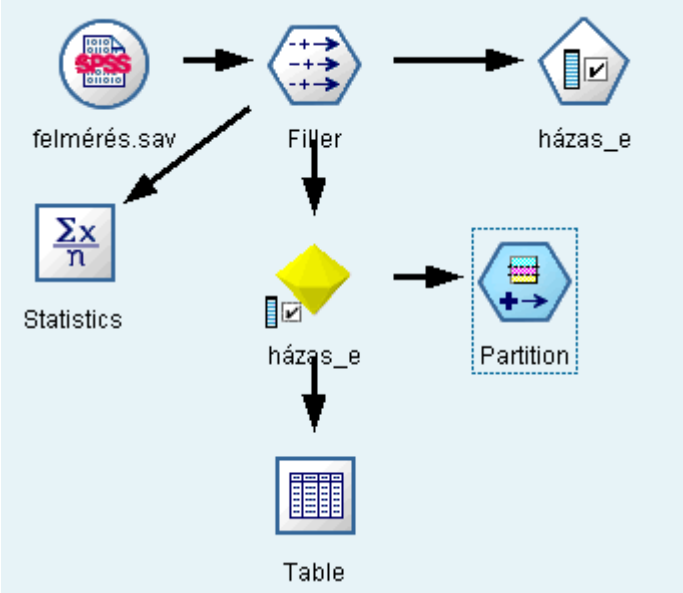


A fenti eredményt kapjuk, látható, hogy a nem fontos változók oszlopai eltűntek, a *neme* változó pedig megmaradt – ő nem vett részt a modellépítésben.

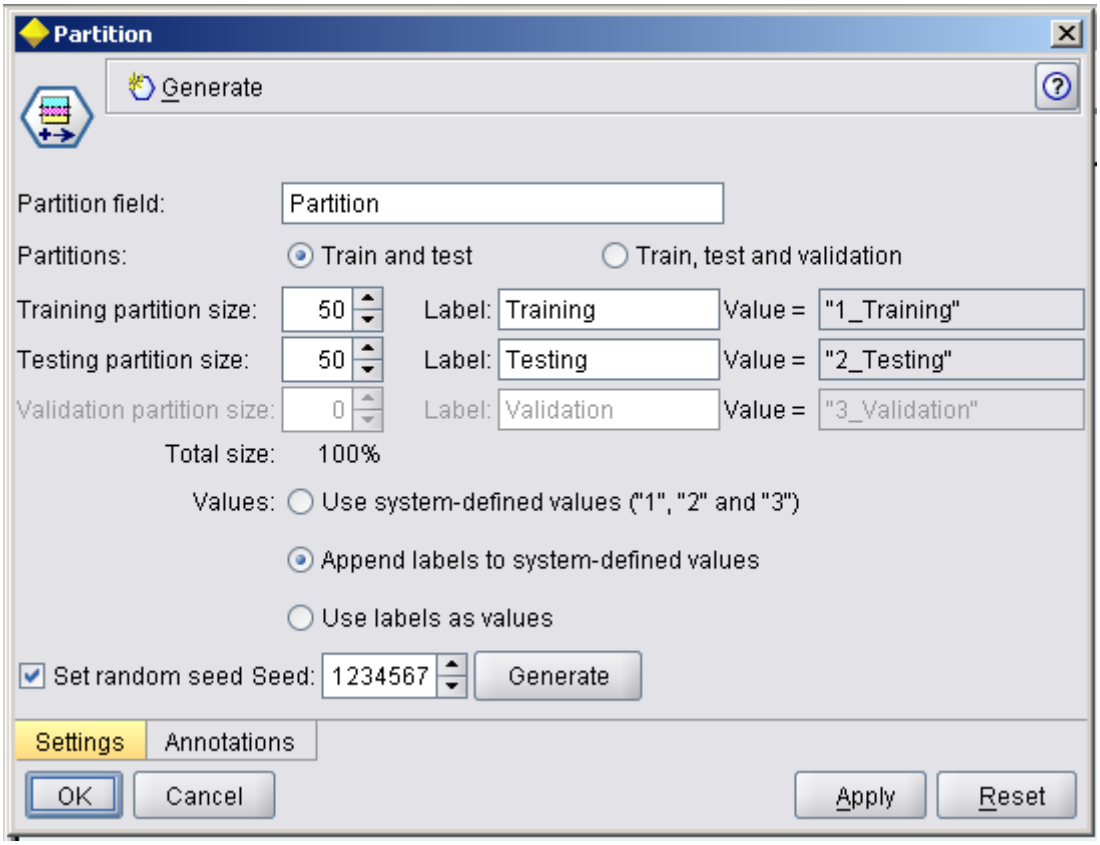
Tanuló algoritmusok előkészítése – adathalmazok kialakítása

A tanuló algoritmusok modellépítő node-okként vannak a Modeler-be építve. Mint fentebb kiderült, minden tanuló algoritmust három vagy négy fázisban, három vagy négy adathalmazon kell működtetni. A modellépítés, tesztelés vagy érvényesítés céljára egyszerűen megadhatók e különböző rekordhalmazok: ha kellően nagy az adathalmaz, akkor a **PARTITION NODE** segítségével képezhetjük a részeket. A **PARTITION NODE** egy Set típusú változót hoz létre, melynek értékei azt jelzik, hogy egy adott rekord a modellépítő, tesztelő vagy érvényesítő adatok közt szerepel-e. Az adathalmazokat véletlenszerűen generálja a node, így használható rendezett adathalmazokban és idősorokban is. A keletkező új Partition mező automatikusan partition modellbemenet lesz (a **TYPE NODE**-nál a **DIRECTION** rovatnál leírva szerepelt e beállítás). Akár mi magunk is megadhatunk olyan mezőt, mely a rekordhalmazokat jelöl ki, de ezt egy **TYPE NODE**-nál a **DIRECTION** rovatban be kell állítani, hogy a modellek az itt megadott rekordhalmazokat tekintsék modellépítő, tesztelő, érvényesítő adatoknak (a modell node-ok **USE PARTITIONED DATA** beállításánál).

Szúrjunk be egy Partition Node-ot a FIELD OPS palettáról a Feature selection készmodellt jelentő „gyémánt" node után!



Nézzük meg a node beállításait!



PARTITION FIELD. Megadható a keletkező új mező neve

PARTITIONS. Megadható, hogy két vagy három lépés számára különítsen el adathalmazokat.

TRAIN AND TEST. Modellépítés és tesztelés

TRAIN, TEST, AND VALIDATION. Modellépítés, tesztelés és érvényesítés

PARTITION SIZE. Minden rekordhalmaz mérete százalékban megadható. Ha ez összesen 100-nál nagyobb, piros hibaüzenettel jelzi a párbeszédpanel.

VALUES. Milyen konkrét értékek jelezzék a rekordhalmazokat:

USE SYSTEM-DEFINED VALUES ("1", "2" and "3"): 1, 2 és 3 numerikus tárolású adatok

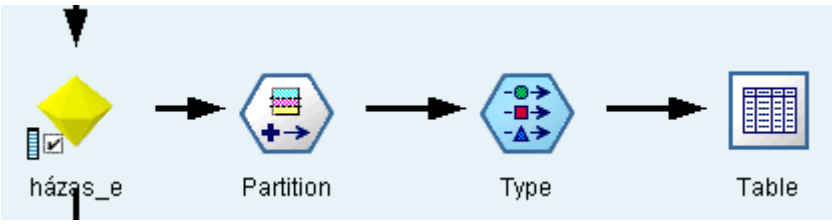
APPEND LABELS TO SYSTEM-DEFINED VALUES: 1_Training, 2_Testing, stb. értékek

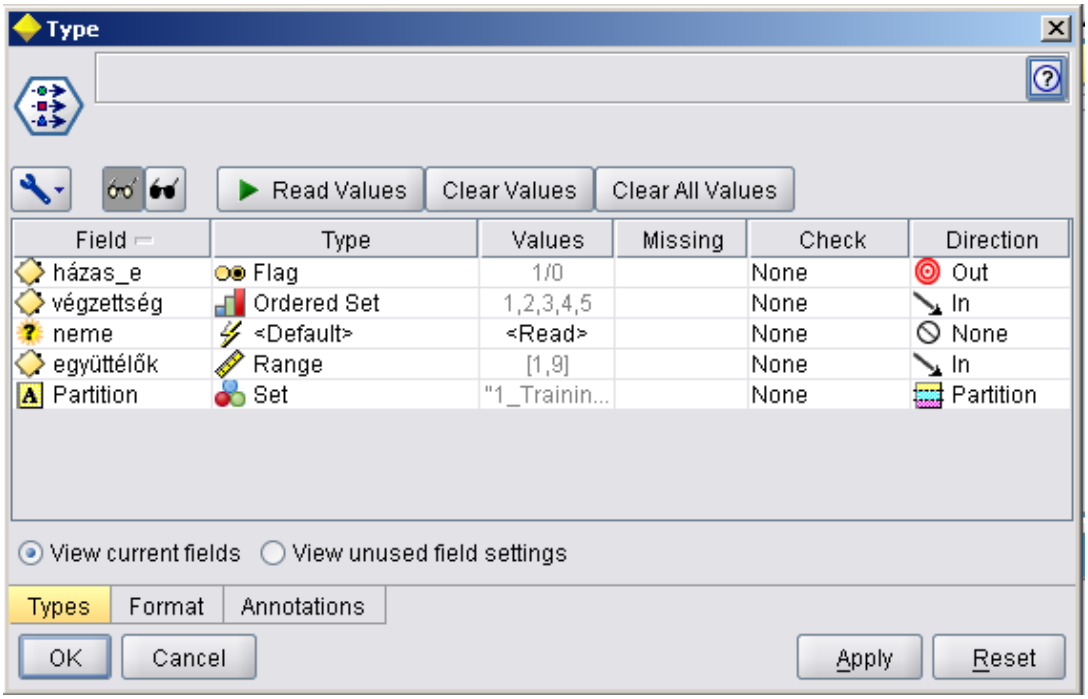
USE LABELS AS VALUES: Training, Testing, stb. értékek

SET RANDOM SEED: A Modeler véletlenszám-generátora számára megadott kezdeti érték. Különböző kezdeti értékek különböző véletlen rekordhalmazokat adnak meg.

Futtassuk a megadott beállításokkal a Partition node-ot!

Nézzük meg az eredményt egy Type és egy Table node-on!





A Type node **DIRECTION** rovata mutatja, hogy a modellépítés számára a *partition* mező nem bemenő vagy célváltozó, hanem adathalmazokat kijelölő mező. Ilyen mezőt a Partition node nélkül is létre lehetett volna hozni, de így sokkal egyszerűbb volt.

Table (5 fields, 6 400 records)

File Edit Generate

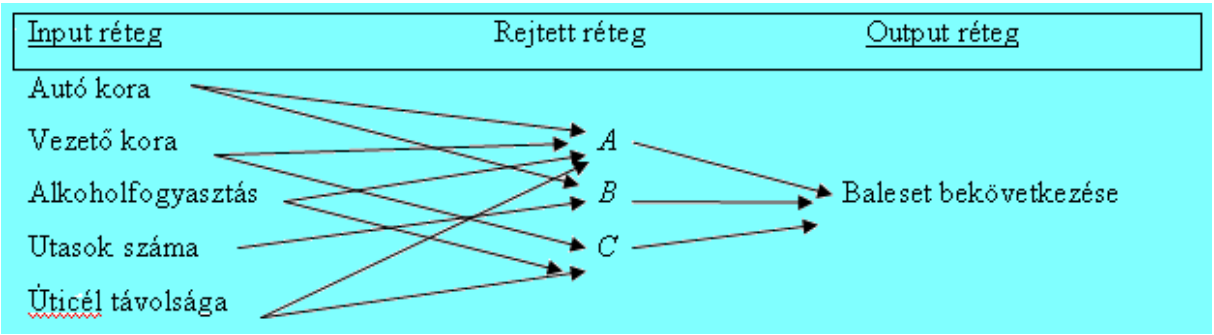
	házas_e	végzettség	neme	együttélők	Partition
1	1	1	n	4	1_Training
2	0	1	f	1	1_Training
3	1	3	n	3	2_Testing
4	1	4	f	3	2_Testing
5	0	2	f	2	1_Training
6	1	3	f	2	1_Training
7	0	3	f	1	2_Testing
8	0	2	n	1	1_Training
9	0	1	n	2	1_Training
10	1	3	f	6	1_Training
11	1	3	n	2	2_Testing

Table Annotations

Töröljük ki a most beillesztett type és table node-okat!

A Neural net modell használata

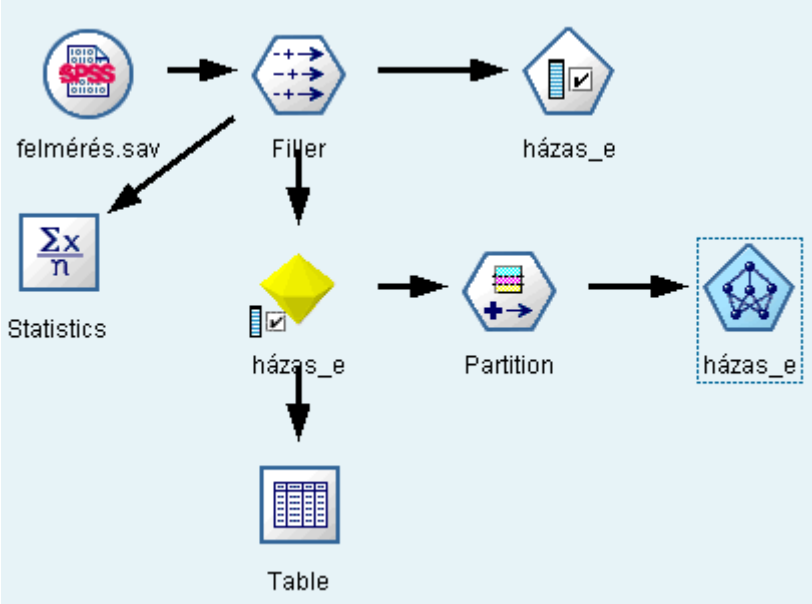
A neurális háló egy speciális gráffal - mint modellel - dolgozik. A bemenő változók adatai egy képzelt „rétegnek” a csomópontjai, ezek értékeinek súlyozásával kapott eredményeket kapják meg az úgynevezett rejtett „réteg” csomópontjai. Ezek egy adott küszöbérték elérésétől függően adatot adnak tovább a „célréteg” csomópontjainak, melyek nem mások, mint a célváltozók. A célváltozók is az előző rétegből kapott értékektől függően vesznek fel értékeket. A csomópontokat ezentúl **neuron**oknak nevezzük.



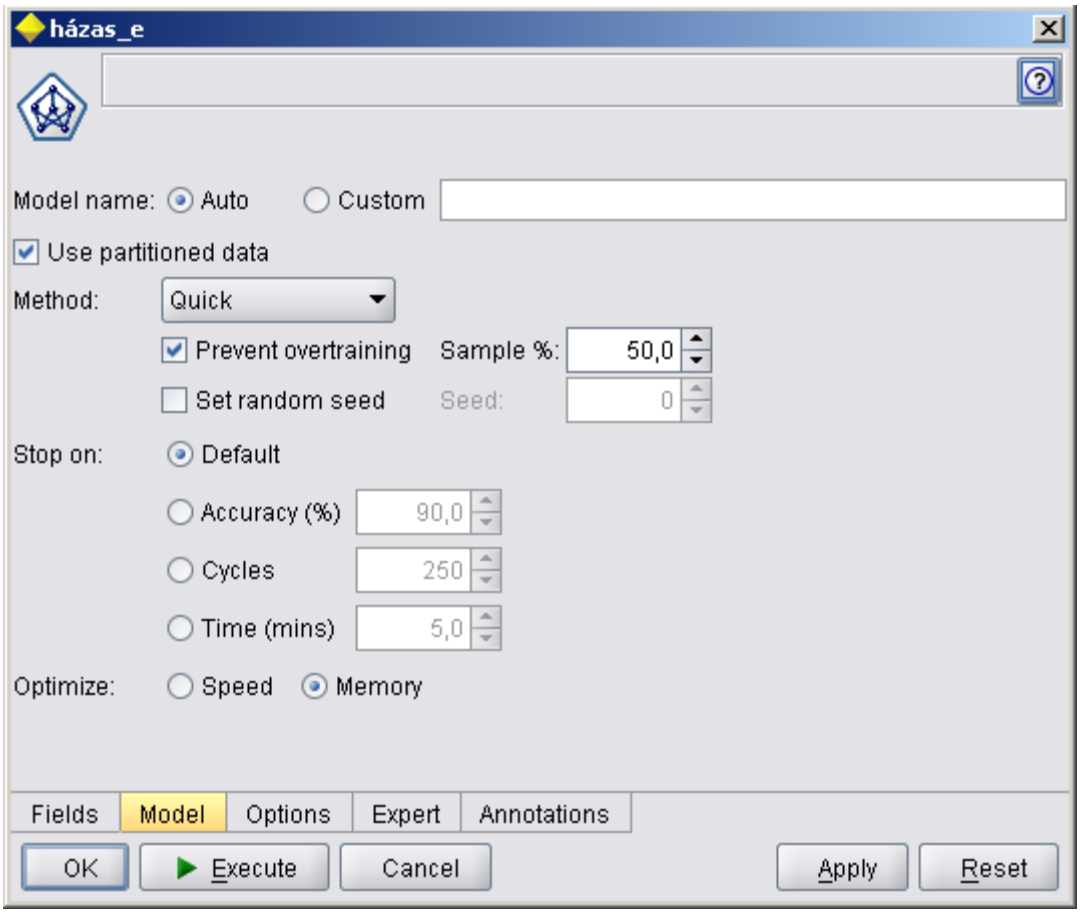
A példaként bemutatott hálón a nyilak helyén matematikai súlyozás áll, s a súlyozottan összeadott adatértékek, ha elérnek egy adott küszöbértéket, akkor vált át a következő réteg neuronja eggyel nagyobb értékre.

A *Neural net* típusú node neurális háló – tehát egy célváltozó meghatározására épít modellt „tanuló” algoritmussal. A súlyozást véletlenszerűen kezdi egy előzetes modellel, aztán próbál egyre jobb és jobb modellt alkotni. A tanulóalgoritmus a modellt folyamatosan változtatja előre megadott ideig vagy állapotig.

Illesszünk egy Neural net modellépítő node-ot a Partition node után!



Nézzük meg a node beállításait!



USE PARTITIONED DATA: A Partition Node-nál beállított Training set alapján épít modellt.

METHOD: A modellépítés hat matematikai módszere közül választhatunk.

PREVENT OVERTRAINING: A modellépítés során egy gyorsan felépített „előzetes modellt" finomít a Modeler. Itt az előzetes modell építéséhez szükséges rekordok százalékos arányát lehet megadni.

SET RANDOM SEED. Az előzetes modellépítéshez szükséges véletlenszám kezdőértékének megadása.

STOP ON: Milyen kritérium elérése állítsa le a modell finomítását, hogyan álljon le az öntanulás

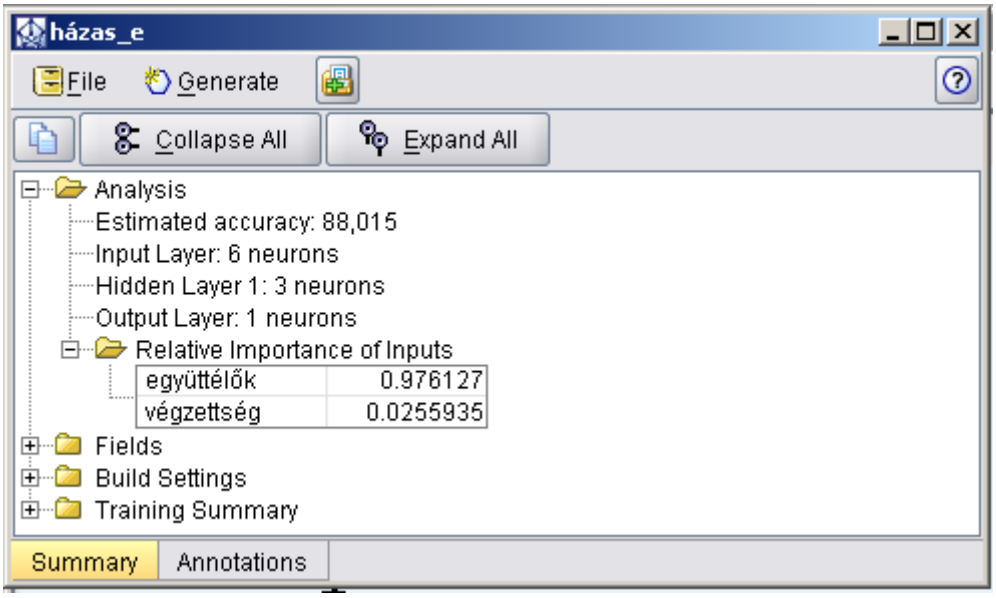
DEFAULT: Stream-ben beállított feltétel (itt 20 elemű halmazt jelent)

> **ACCURACY (%):** Akkor áll le a tanulás, ha a célváltozó modell alapján számított értéke a rekordok megadott %-ában a helyes értéket adja.
>
> **CYCLES:** Tanulási ciklusok száma.
>
> **TIME (MINS):** A tanulási idő percben megadott értéke. Ha ezt választjuk, akkor látványos futtatást láthatunk, modellépítés közben egy mozgó grafikonon mutatja, mi az eddigi legjobb neurális modell becslési százaléka (kék vonal) és emellett az éppen aktuális modell becslési aránya (piros vonal).

OPTIMIZE: Túl nagy adathalmazzal dolgozva hasznos, ha a tanulóalgoritmust idő szerint vagy memóriaigény szerint optimalizáljuk.

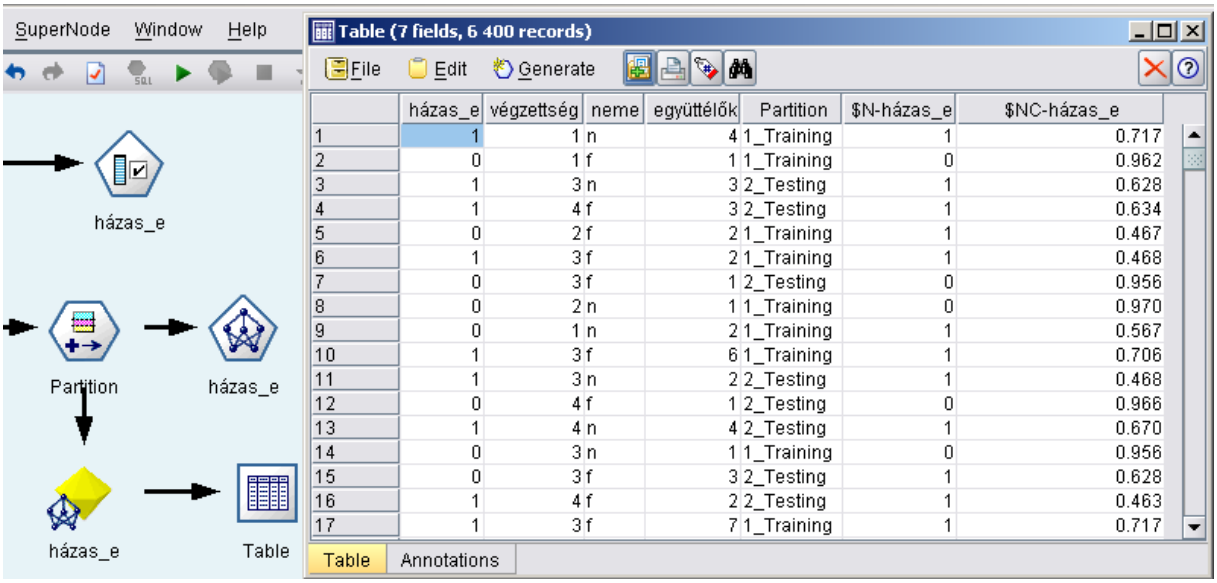
A tanulási idővel csínján kell bánni, mert ha túl hosszú ideig tanul a szoftver, akkor túltanulás léphet fel, azaz nem a mintában rejlő általános összefüggéseket, hanem magát a mintát tanulja be az algoritmus. Ilyen esetben a minta adatait alkalmazva nagy biztonsággal jó lesz a modell, más esetben azonban nem, mert a modell csak ennek a mintának a speciális sajátosságait tartalmazza.

Futtassuk a node-ot a felkínált beállításokkal, majd nézzük meg a kész modell tulajdonságait!



Látható, hogy a három rétegben levő neuronok számát és relatív fontosságukat mutatja a leírás. A modellépítő adathalmazon 88%-os biztonsággal jól „jósol" a modell, a modellben összesítve mindössze 2,5 %-nyi relatív jelentősége van a *végzettség* változónak, a háló súlyai elsősorban az *együttélők* változót veszik figyelembe.

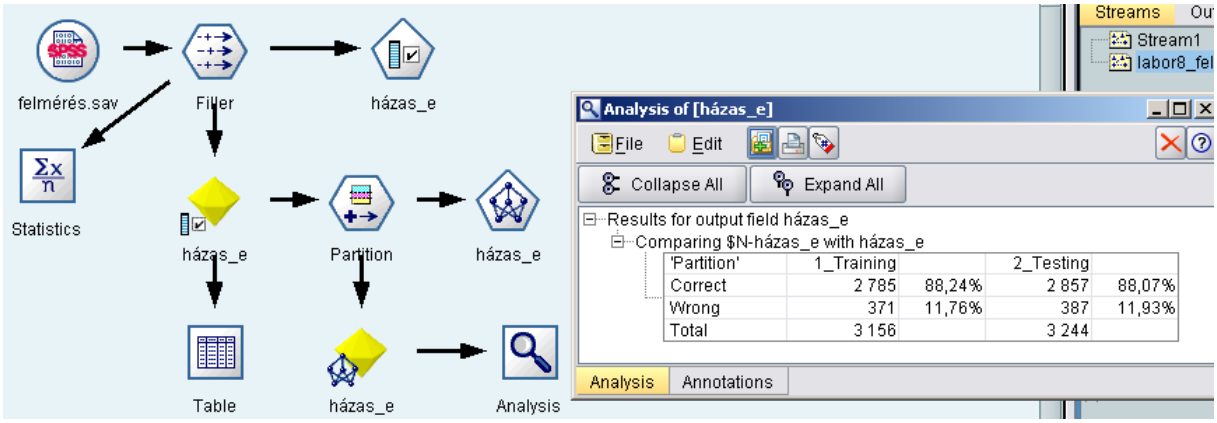
Szúrjuk be a kész modellt a Partition node után és nézzük meg az eredményt egy táblában!



	házas_e	végzettség	neme	együttélők	Partition	$N-házas_e	$NC-házas_e
1	1	1	n	4	1_Training	1	0.717
2	0	1	f	1	1_Training	0	0.962
3	1	3	n	3	2_Testing	1	0.628
4	1	4	f	3	2_Testing	1	0.634
5	0	2	f	2	1_Training	1	0.467
6	1	3	f	2	1_Training	1	0.468
7	0	3	f	1	2_Testing	0	0.956
8	0	2	n	1	1_Training	0	0.970
9	0	1	n	2	1_Training	1	0.567
10	1	3	f	6	1_Training	1	0.706
11	1	3	n	2	2_Testing	1	0.468
12	0	4	f	1	2_Testing	0	0.966
13	1	4	n	4	2_Testing	1	0.670
14	0	3	n	1	1_Training	0	0.956
15	0	3	f	3	2_Testing	1	0.628
16	1	4	f	2	2_Testing	1	0.463
17	1	3	f	7	1_Training	1	0.717

Két új mező keletkezett. A *$N-házas_e* mezőben a *házas_e* változó bemeneti változók alapján számított („jósolt”) értékét láthatjuk. A *$NC-házas_e* mező pedig a modellből számított pontos értéket mutatja (ennek kerekítése az előbbi).

Szúrjunk be a kész modell után a streambe a Table node helyett egy Analysis (output) node-ot és futtassuk le!

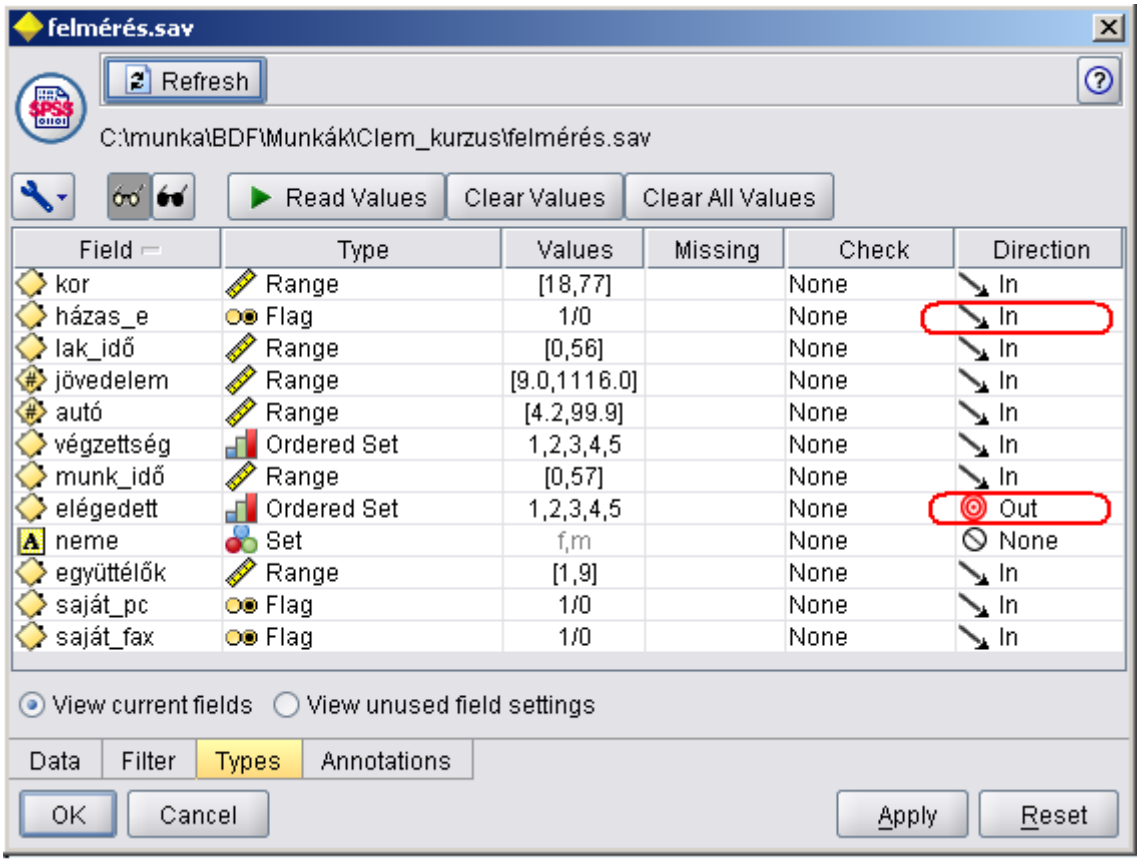


'Partition'	1_Training		2_Testing	
Correct	2 785	88,24%	2 857	88,07%
Wrong	371	11,76%	387	11,93%
Total	3 156		3 244	

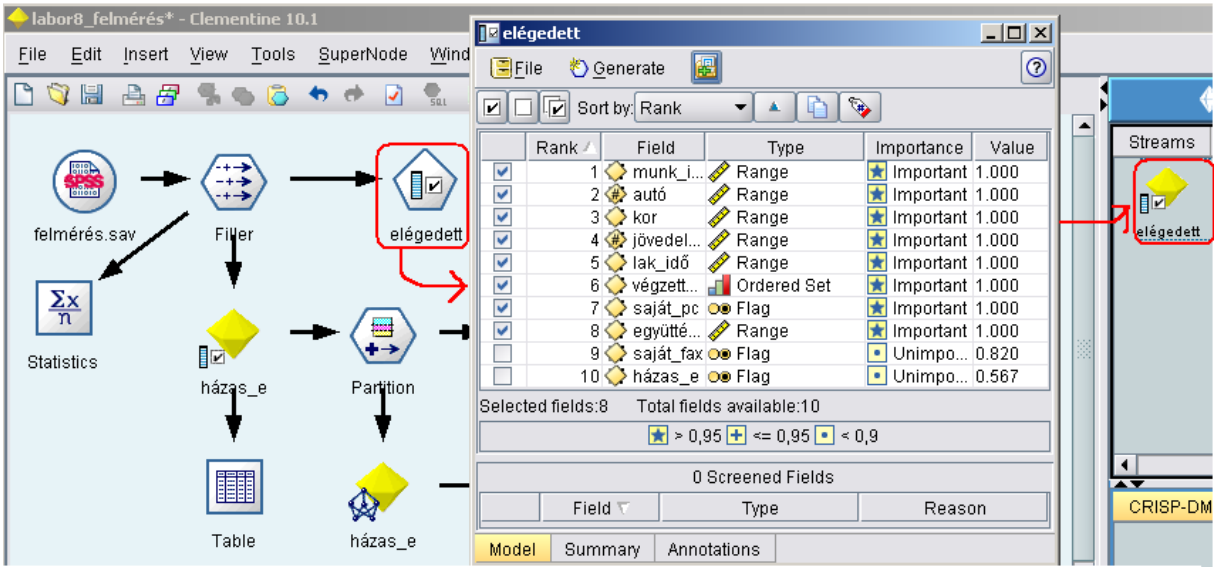
Az Analysis node a modellek hatékonyságát vizsgálja. A beállított két rekordhalmazra külön-külön mutatja, mekkora százalékban találja el a modell a tényleges célváltozó értékét, s milyen százalékban hibás a döntése. Ha a kész modell megfelelő, akkor kimenthetjük a gyorsmenü (jobbegér) megfelelő menüpontja segítségével.

Próbáljuk ugyanezt a modellépítést végigvinni a *házas_e* helyett az *elégedett* változóra nézve!

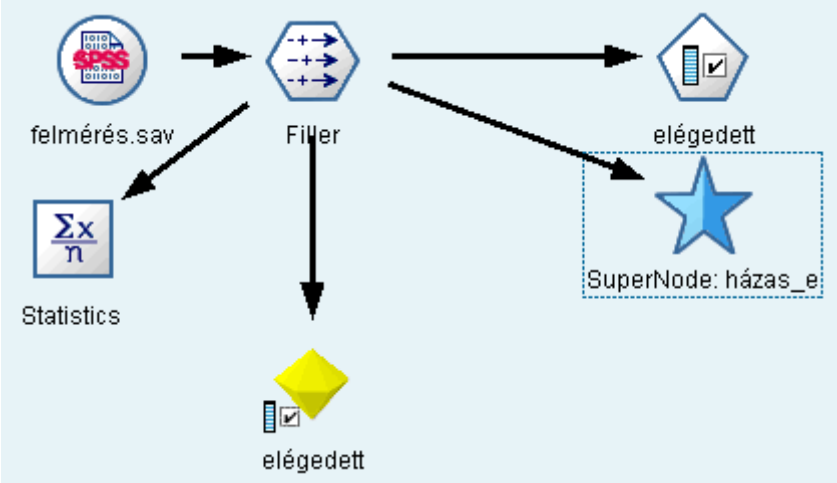
A szükséges lépések képekkel illusztrálva:



1. Source node Types fülén a célváltozó megváltoztatása

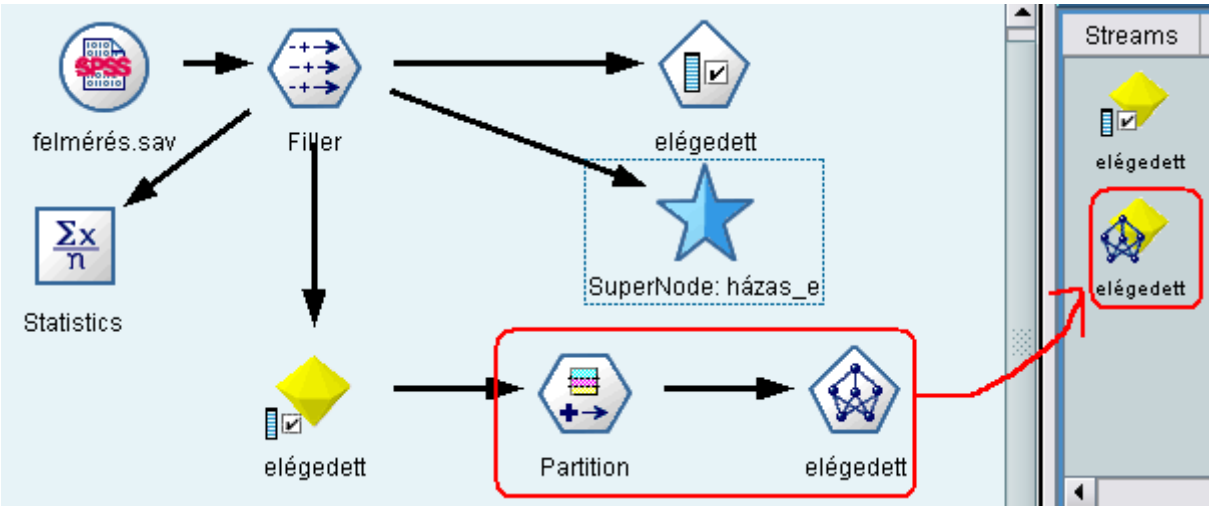


2. A Feature Node futtatása, kész modell tanulmányozása (8 fontos mező lesz)

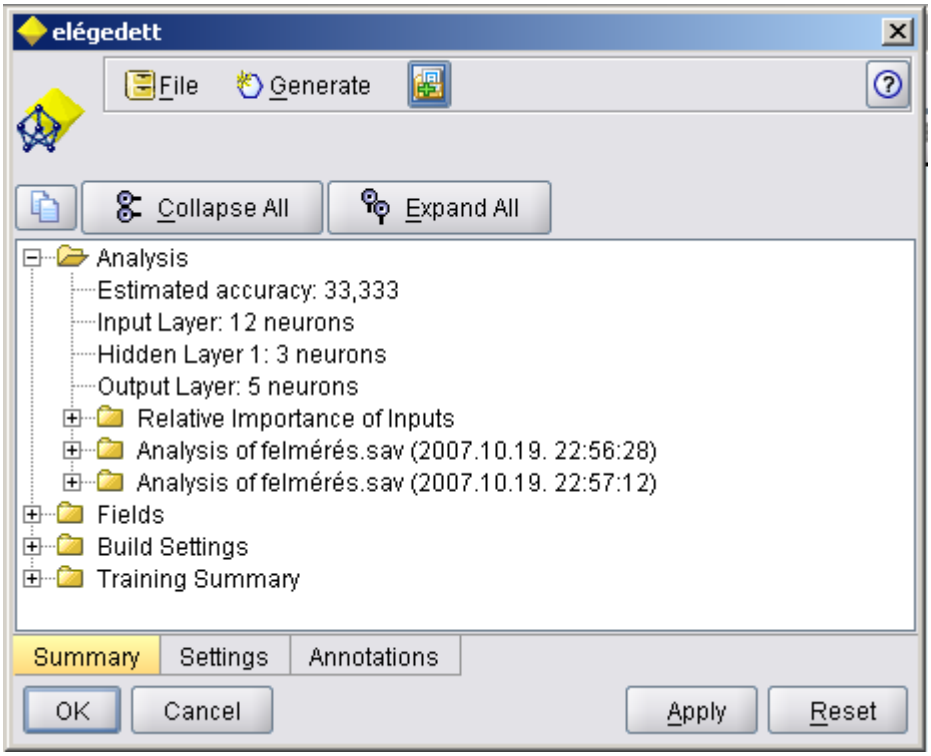


Helytakarékosságból egy SuperNode-ot készítettünk az iménti *házas_e* változóra vonatkozó vizsgálatainkból. Ennek neve: *SuperNode:házas_e* lett.

Ezután beszúrtuk a kész Feature selection modellt egy node-ként.

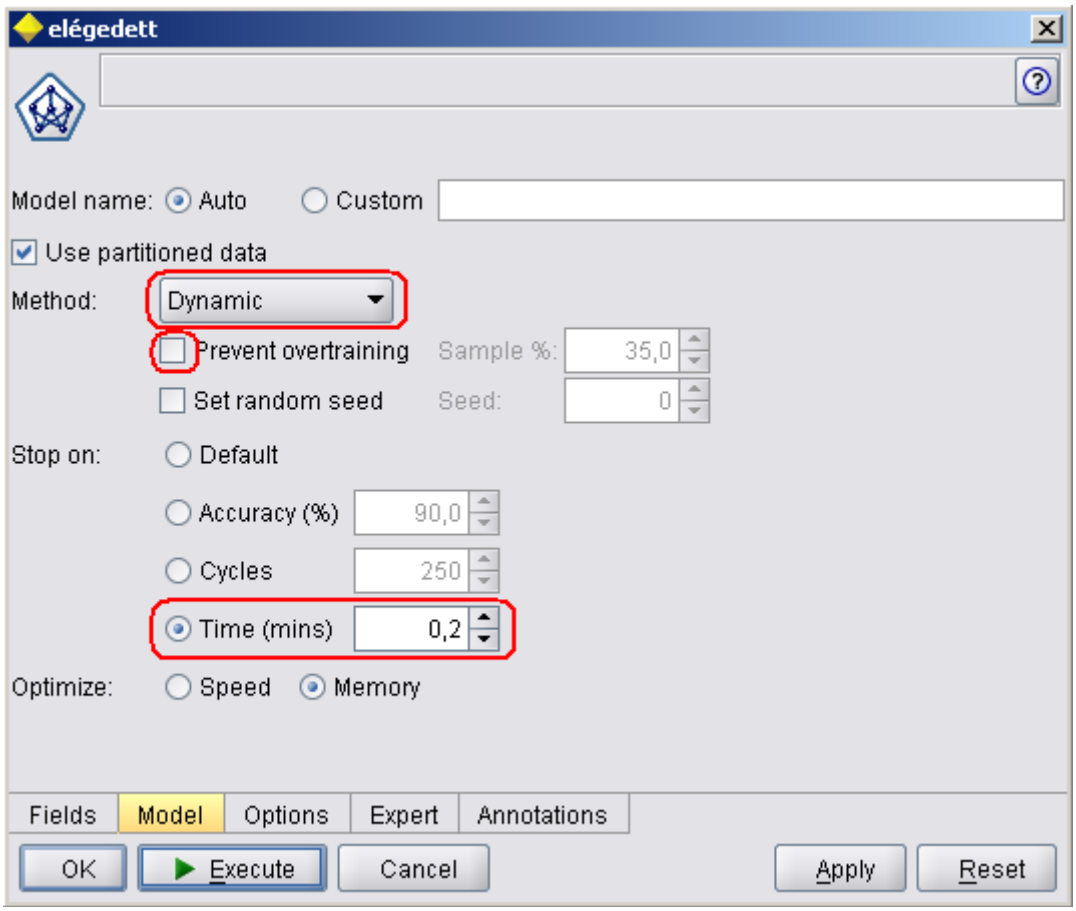


Két részre osztottuk a rekordhalmazt, mint az imént, majd beszúrtunk egy Neural net modellépítő node-ot a végére. Ezt futtatva a következő kész modellt kapjuk:

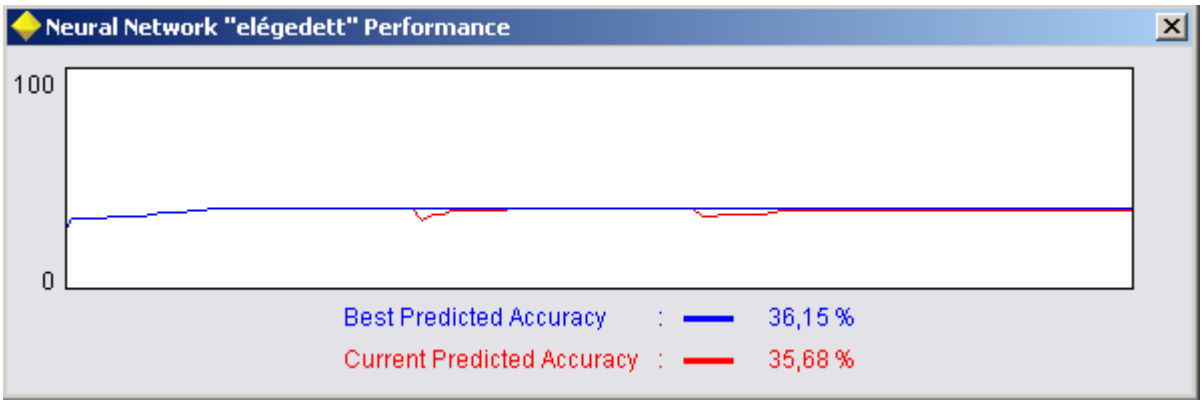


Igen kevés a kiírt 33%-os előrejelzés. Próbáljuk a módszert változtatni a modellépítő node beállításainál!

Állítsuk a módszert Dynamic-ra, kapcsoljuk ki az előzetes modellépítést és kössük a tanulás végét 0,2 percnyi idő elteltéhez!



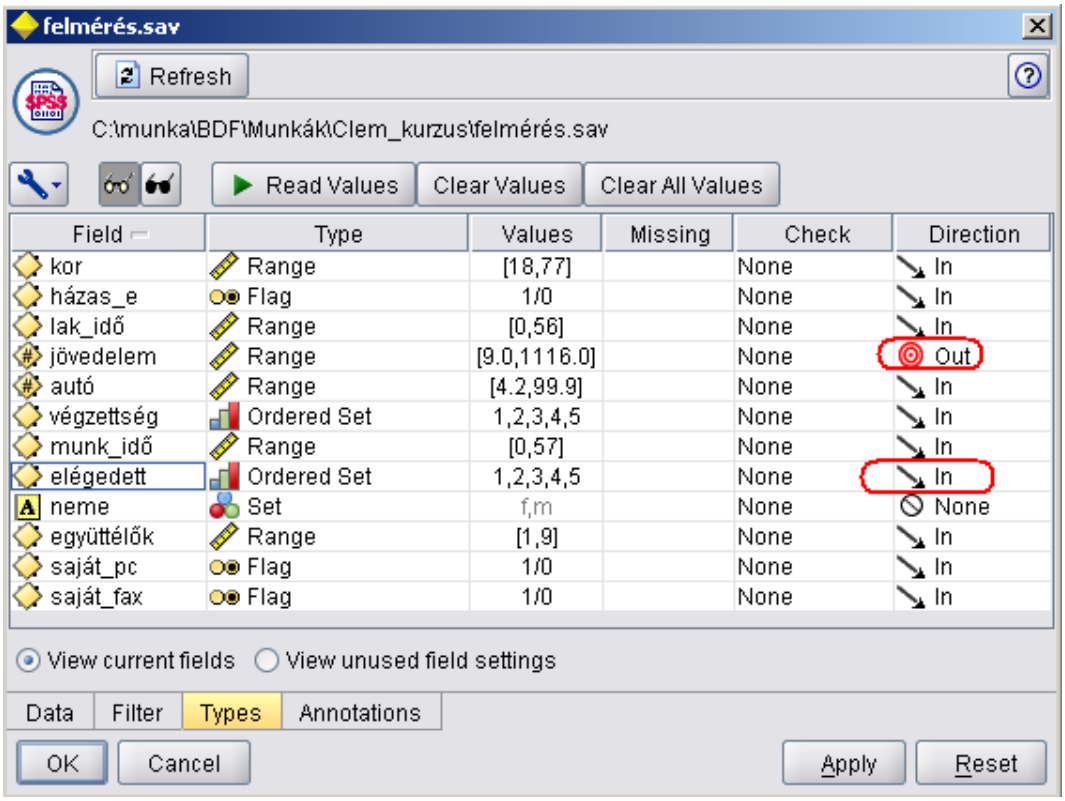
A modellépítést egy ablakban egy grafikon kíséri, mely a legjobb elért előrejelzési arányt és az éppen aktuális előrejelzési arányt mutatja az idő függvényében, azaz a modell folyamatos változtatása során.

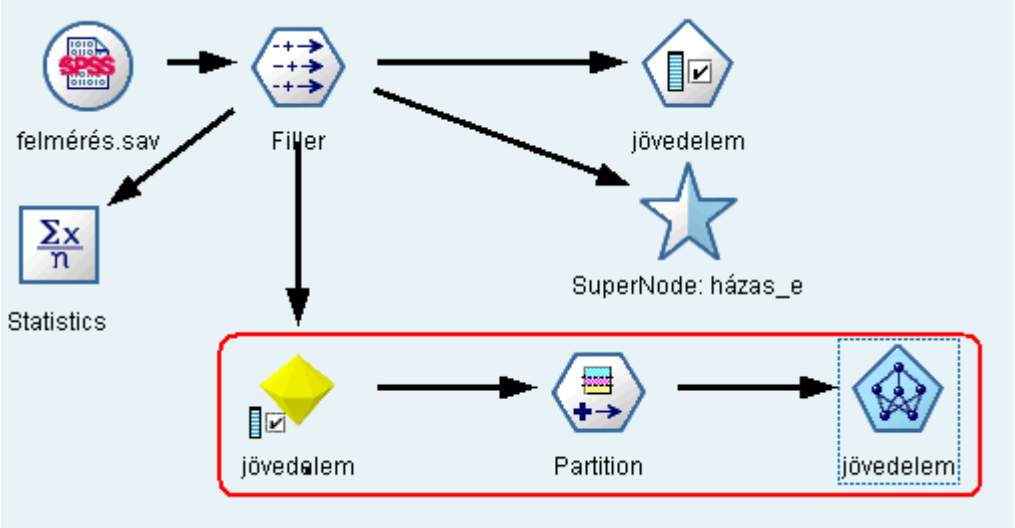


Több módosítás után úgy látszik, nem lehet jobb előrejelzési arányt elérni, a bemenő adatok nincsenek elég erős összefüggésben a célváltozóval.

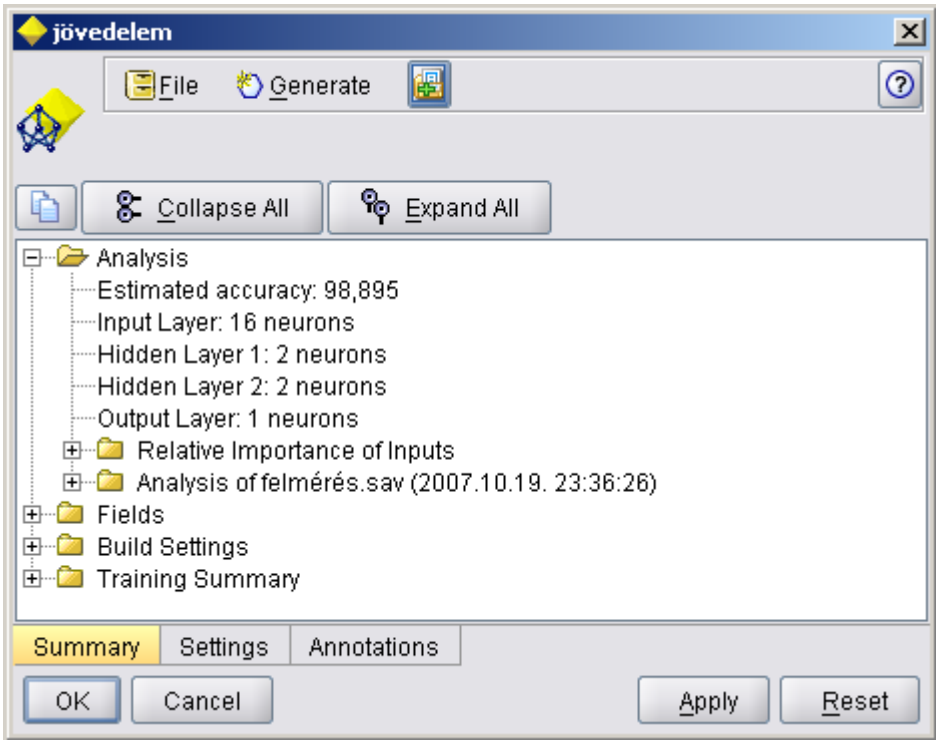
Próbáljuk ugyanezt a modellépítést és modell-ellenőrzést végigvinni a *jövedelem* változóra nézve!

Képekben:

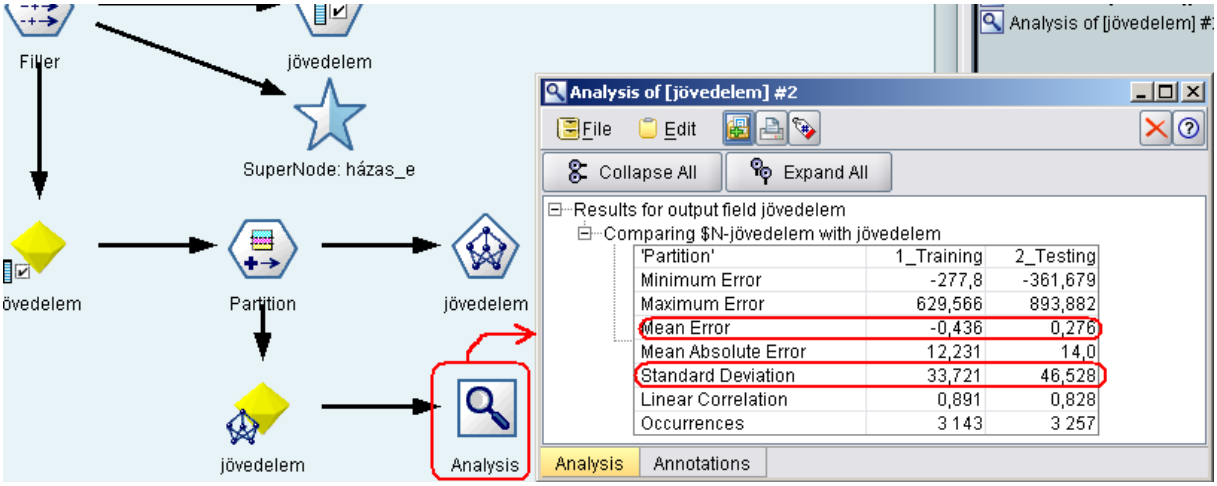




Az eredmény majdnem 99 %-os előrejelzés!



Készítsük el a modell analízisét is:



Mivel a célváltozó Range típusú, ezért nem lehet találati százalékot kimutatni, ehelyett az előrejelzés tévedésének átlagát és szórását érdemes figyelni a táblázatban. A 46-os szórásérték nem is olyan nagy, ha figyelembe vesszük, hogy az eredeti adatok 9-1116-ig terjedtek (ezer $-ban).

Építsünk neurális háló modellt a lak_idő és az autó célváltozókra!

Próbáljuk ki a neurális háló többi beállításait is! (a végeredményben nem számottevő a különbség)

Próbáljuk a Partition node-nál csökkenteni a Training set adathalmazt! (nő a másik adathalmaz hibája)

9. Döntési fa

E gyakorlat célja, hogy újabb modelleket ismerjünk meg. A már tanult neurális háló modell úgy keres összefüggéseket bemenő változók és a célváltozó között, hogy magát a modellt, a rejtett rétegben levő számításokat a felhasználó nem ismeri, csak az eredményt láthatja és használhatja.

A döntési fák ugyanilyen esetekben használhatók, de bepillantást adnak az összefüggések szerkezetébe.

Az adathalmaz

Egy amerikai biztosítási cég 5000 ügyféléről készült nyilvántartásból fogunk dolgozni. Az adatokat az *ügyfél_.sav* SPSS fájl tartalmazza.

Az egyik cél az lesz, hogy az ügyfelek sokféle adatának az ugyf_ev változóval való statisztikai kapcsolatát felderítsük. E változó azt jelzi, hogy egy adott személy hány éve ügyfele a cégnek.

Tanulmányozzuk az adathalmaz változóit az alábbi táblázat alapján!

Változónév	leírás
ugyf_ev	hány éve ügyfél (célváltozó)
ok	miért ügyfél (1-olcsó 2-kényelmes 3-szolgáltatások 4-egyéb 8-nincs válasz 9-nem tudja)
zona	földrajzi zóna
nem	neme (0:férfi, 1:nő)
kor	életkor
tanido	hány évet töltött tanulással
vegzettseg	végzettség kódja
fogl	foglalkozási kategória kódja
nyugdij	nyugdíjas (0-nem, 1-igen)
jovedelem	háztartás jövedelme (1000$)
ados_mertek	adósságának mértéke a jövedelem százalékában (x100)
hitelados	hitelkártya-adósság (1000$)
egyebados	egyéb adósság (1000$)
kesedelem	volt-e késedelmes banki adósságrendezése (0-nem 1-igen)
elegedett	munkájával mennyire elégedett (0-5)
hazas_e	házas-e (0-nem, 1-igen)
egyuttelok	háztartásában együtt élők száma
sajathaz	saját házában lakik-e (0-nem, 1-igen)
lakastipus	lakásának típusa (1-családi, 2-társas, 3-tömb, 4-mobil)
lakas_ev	hány éve lakik jelenlegi otthonában
autok	jelenlegi saját/bérelt autók száma
auto_sajat	saját autója van-e (-1-nincs, 0-bérli, 1-igen)
auto_kat	elsődleges autó értékkategória (-1:nincs, 0-3:árkat.)
kozl_eszk	munkába járás (1-saját 2-közös autó 3-tömegközl 4-nem gépjárművel 5-nem jár be)
bejar_ido	munkahelyre érés ideje (perc)
pol_konz	politikai nézetei 7-es skálán (1- nagyon liberális 7-nagyon konzervatív)
parttag	politikai párt tagja-e (0-nem 1-igen)
szavaz	szavazott a legutóbbi választáson
internet	Internet (0-nincs 1-modem 2-DSL 3-kábeles 4-egyéb)

tvnezes	múlt heti tévénézés (óra)
sajat_tv	van-e saját tv-je (0-nem, 1-igen)
sajat_CD	van-e saját CD-lejátszója (0-nem, 1-igen)
sajat_PC	van-e saját PC-je (0-nem, 1-igen)
ujsag	előfizet-e újságra (0-nem, 1-igen)
haziallat_db	háziállatai száma
macska_db	
kutya_db	
madar_db	
hullo_db	
kisallat_db	
hal_db	

Adatok beolvasása, előzetes adatvizsgálat

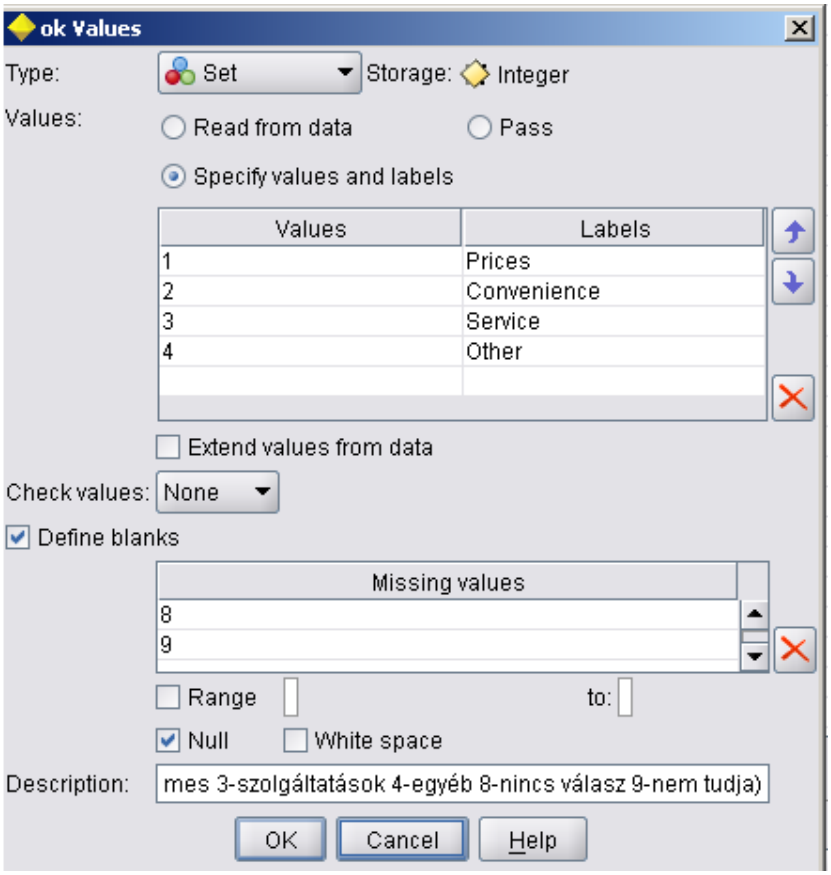
Hozzunk létre egy Source node-ot, mely az adatokat beolvassa!

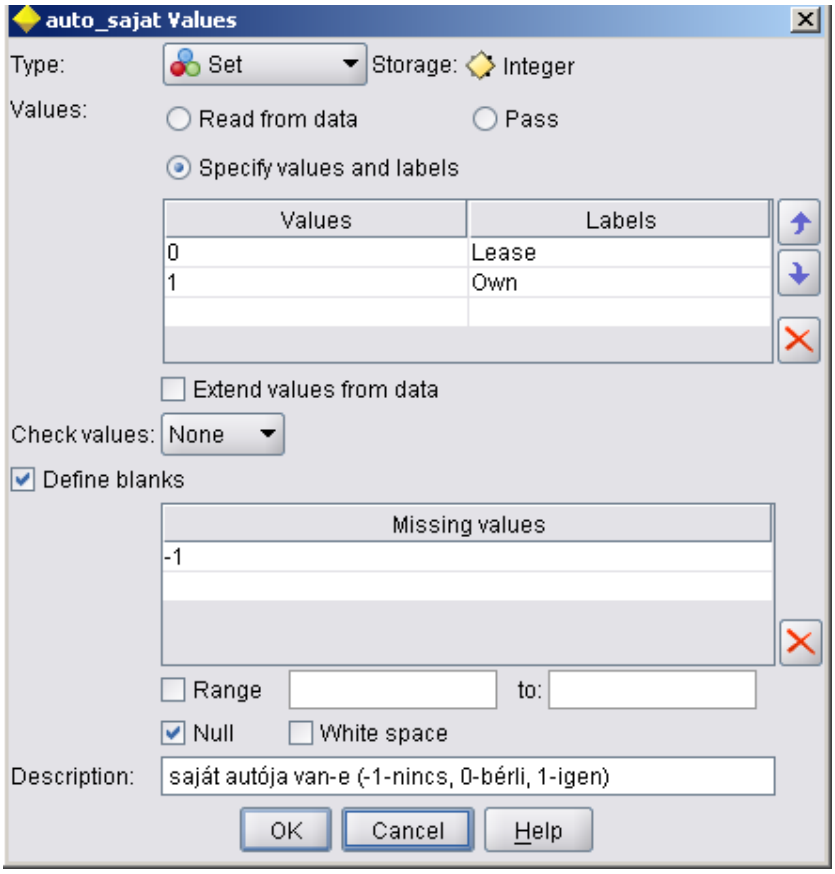
A célkitűzésnek megfelelően kell beállítani a változók kimenet-típusait (**DIRECTION**). Kimenetként kell tekinteni a modelleknek az *ugyf_ev* változót.

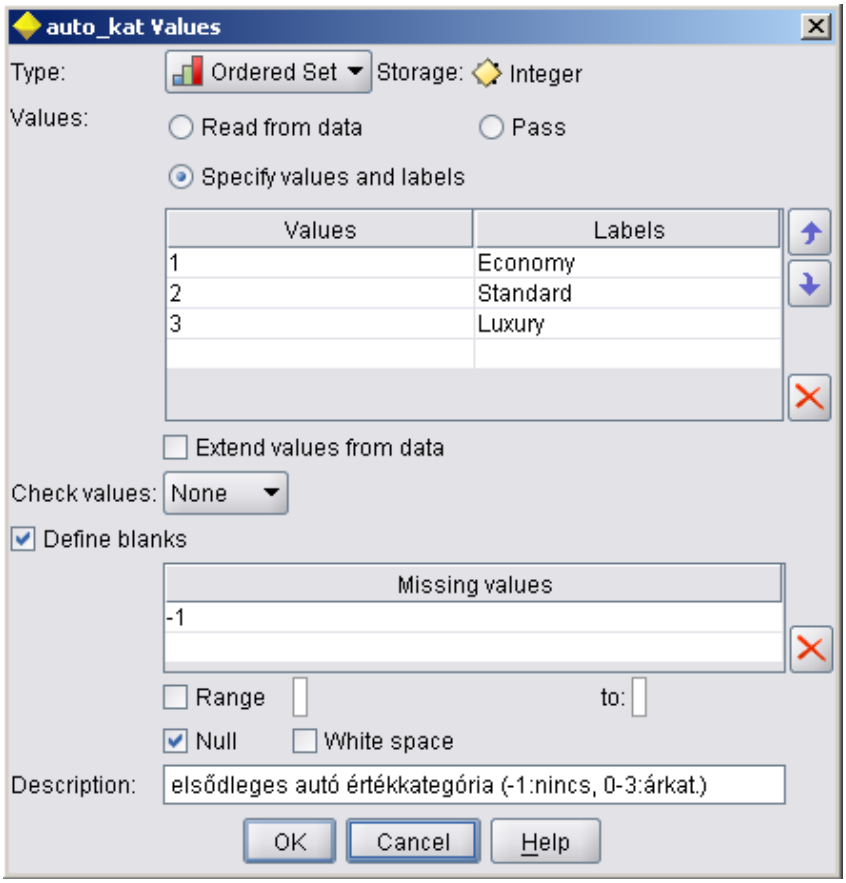
Az *ugyf_ev* változó legyen OUT**, a többi** IN **típusú!**

Figyeljük meg, hogy az SPSS fájlban már definiálva voltak az *ok*, *auto_sajat* és az *auto_kat* változók „blank" értékei.

Figyeljük meg e változók „blank" beállításait!


ok Values
Type:
Set
Storage:
Integer
Values:
Read from data
Pass
Specify values and labels
Values
Labels
1
Prices
2
Convenience
3
Service
4
Other
Extend values from data
Check values:
None
Define blanks
Missing values
8
9
Range
to:
Null
White space
Description:
mes 3-szolgáltatások 4-egyéb 8-nincs válasz 9-nem tudja)
OK
Cancel
Help

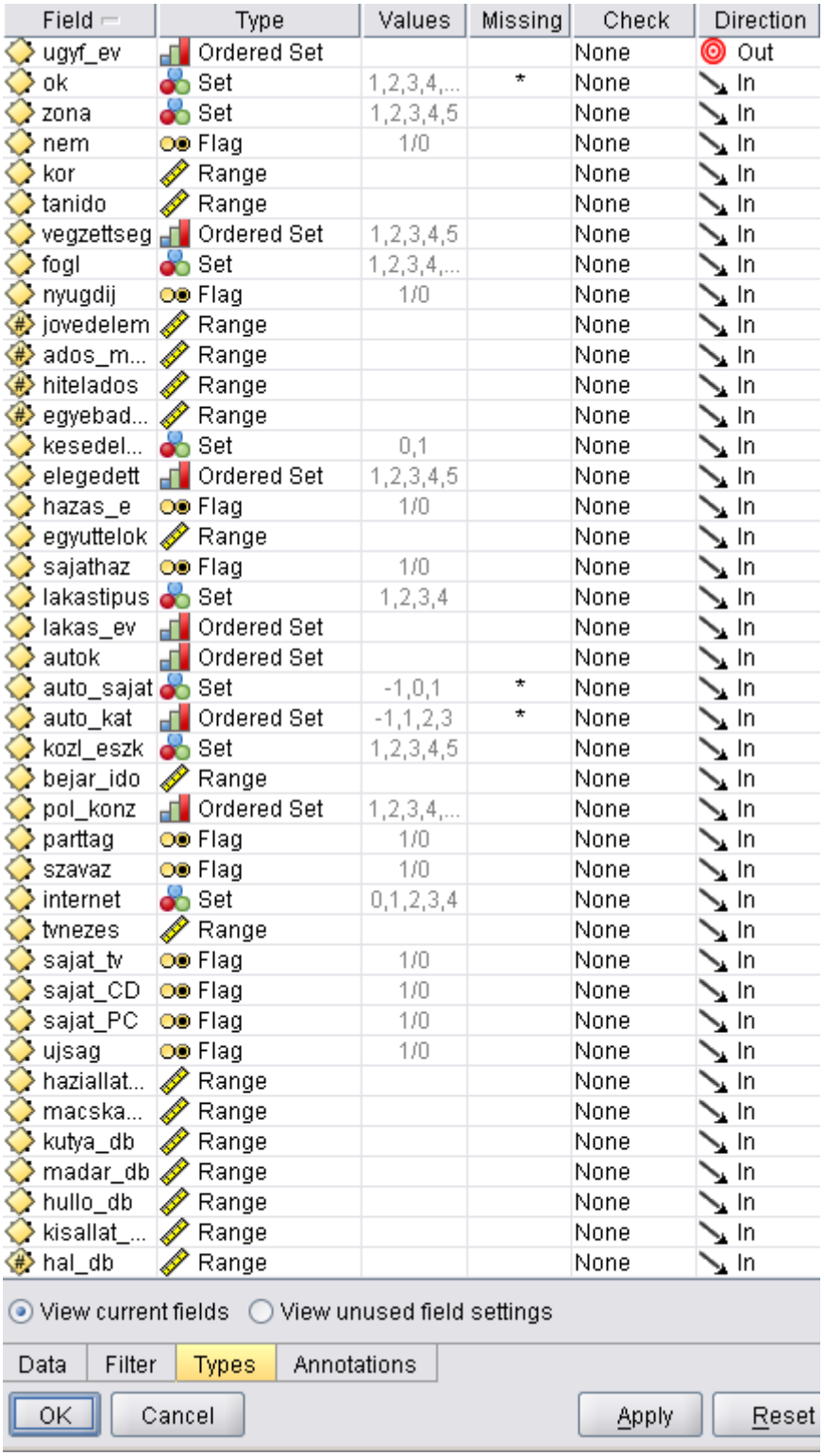

auto_sajat Values
Type:
Set
Storage:
Integer
Values:
Read from data
Pass
Specify values and labels
Values
Labels
0
Lease
1
Own
Extend values from data
Check values:
None
Define blanks
Missing values
-1
Range
to:
Null
White space
Description:
saját autója van-e (-1-nincs, 0-bérli, 1-igen)
OK
Cancel
Help



Az adattáblában a kétértékű Set típusú változókat Flag-ként kellene kezelni, ilyen pl. a *nyugdij* változó, melynek két értéke egy igaz-hamis adat, attól függően, hogy egy ügyfél nyugdíjas-e.

Állítsunk be minden kétértékű Set-nek beolvasott változót Flag típusúra! (*nem, nyugdij, hazas_e, kesedelem, sajathaz, parttag, szavaz, sajat_tv, sajat_PC, sajat_CD, ujsag*)

Az eddigi beállítások az alábbi ábrán követhetők:

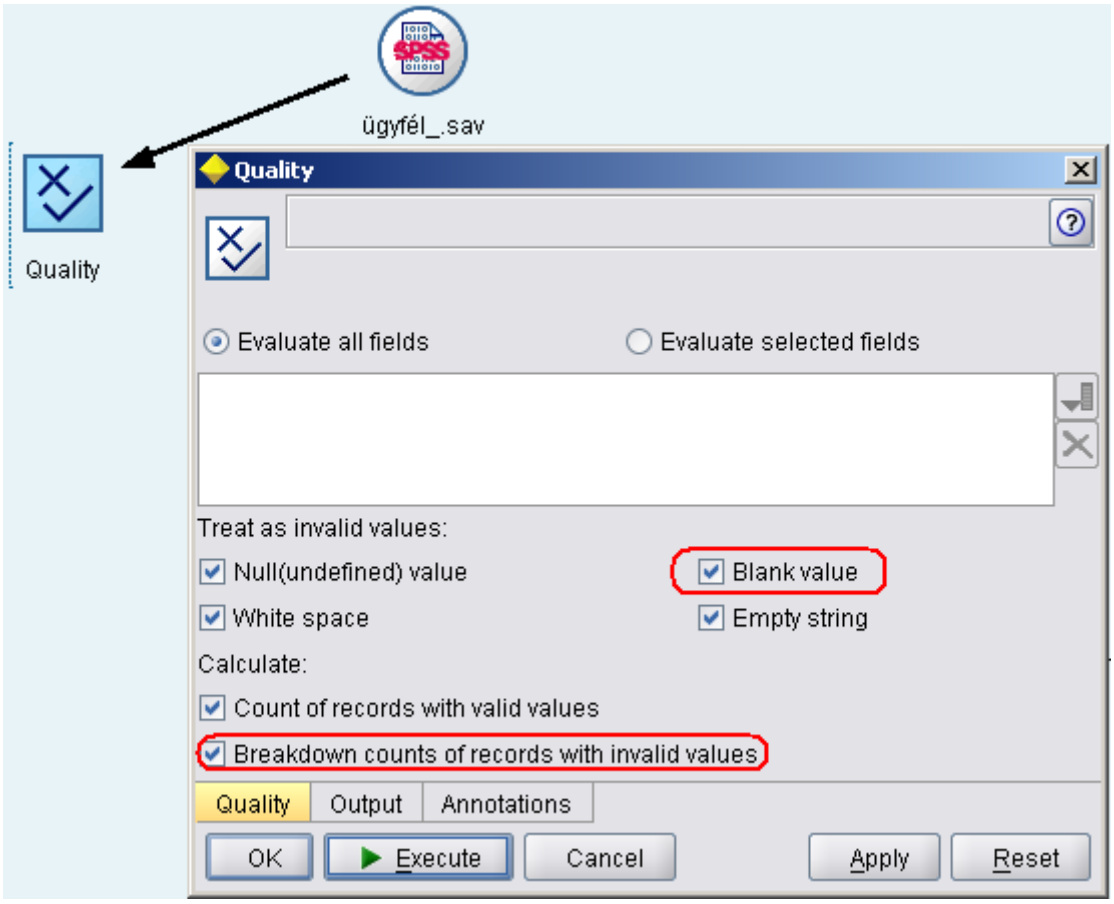


Field	Type	Values	Missing	Check	Direction
ugyf_ev	Ordered Set			None	Out
ok	Set	1,2,3,4,...	*	None	In
zona	Set	1,2,3,4,5		None	In
nem	Flag	1/0		None	In
kor	Range			None	In
tanido	Range			None	In
vegzettseg	Ordered Set	1,2,3,4,5		None	In
fogl	Set	1,2,3,4,...		None	In
nyugdij	Flag	1/0		None	In
jovedelem	Range			None	In
ados_m...	Range			None	In
hitelados	Range			None	In
egyebad...	Range			None	In
kesedel...	Set	0,1		None	In
elegedett	Ordered Set	1,2,3,4,5		None	In
hazas_e	Flag	1/0		None	In
egyuttelok	Range			None	In
sajathaz	Flag	1/0		None	In
lakastipus	Set	1,2,3,4		None	In
lakas_ev	Ordered Set			None	In
autok	Ordered Set			None	In
auto_sajat	Set	-1,0,1	*	None	In
auto_kat	Ordered Set	-1,1,2,3	*	None	In
kozl_eszk	Set	1,2,3,4,5		None	In
bejar_ido	Range			None	In
pol_konz	Ordered Set	1,2,3,4,...		None	In
parttag	Flag	1/0		None	In
szavaz	Flag	1/0		None	In
internet	Set	0,1,2,3,4		None	In
tvnezes	Range			None	In
sajat_tv	Flag	1/0		None	In
sajat_CD	Flag	1/0		None	In
sajat_PC	Flag	1/0		None	In
ujsag	Flag	1/0		None	In
haziallat...	Range			None	In
macska...	Range			None	In
kutya_db	Range			None	In
madar_db	Range			None	In
hullo_db	Range			None	In
kisallat_...	Range			None	In
hal_db	Range			None	In

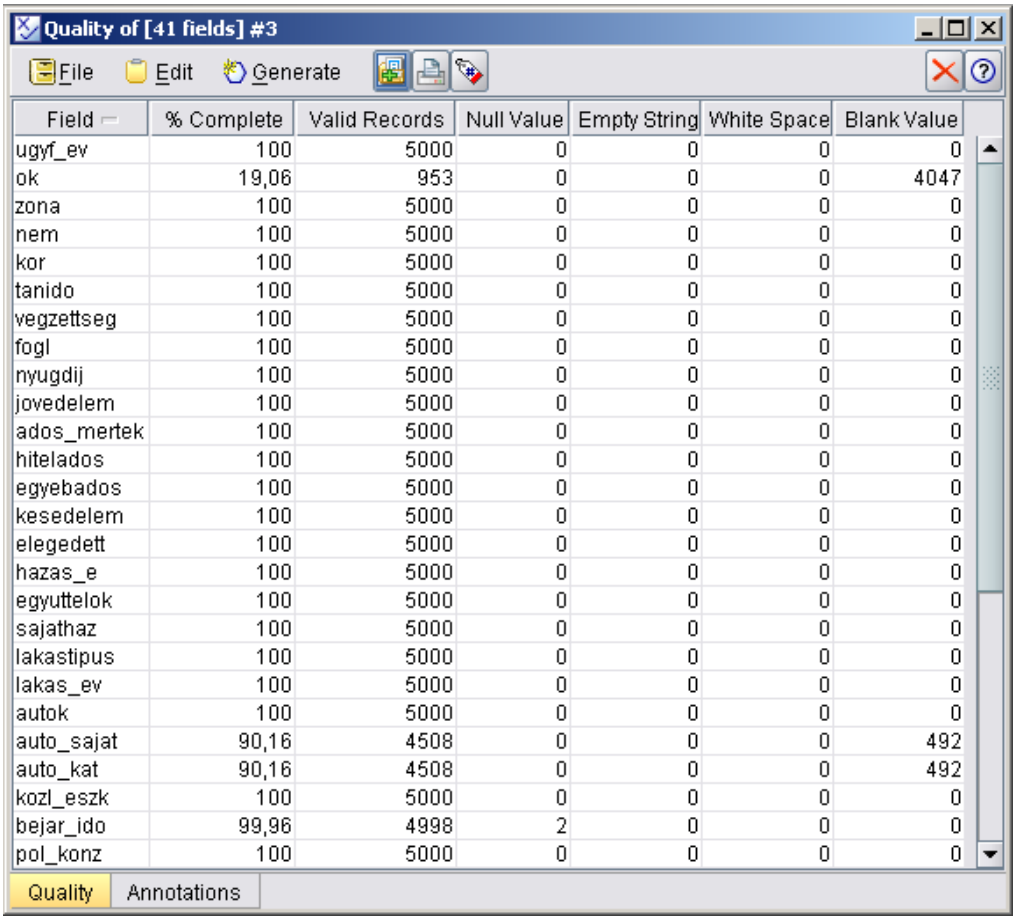
Végezzünk előzetes adatvizsgálatot!

Ehhez először egy Quality node-ot célszerű használni.

Adjunk a *Source node* után egy *Quality node*-ot, mely a „*Blank*” adatértékeket is kijelzi!



A pirossal jelzett beállításokat célszerű megtenni. Az eredmény:



Field	% Complete	Valid Records	Null Value	Empty String	White Space	Blank Value
ugyf_ev	100	5000	0	0	0	0
ok	19,06	953	0	0	0	4047
zona	100	5000	0	0	0	0
nem	100	5000	0	0	0	0
kor	100	5000	0	0	0	0
tanido	100	5000	0	0	0	0
vegzettseg	100	5000	0	0	0	0
fogl	100	5000	0	0	0	0
nyugdij	100	5000	0	0	0	0
jovedelem	100	5000	0	0	0	0
ados_mertek	100	5000	0	0	0	0
hitelados	100	5000	0	0	0	0
egyebados	100	5000	0	0	0	0
kesedelem	100	5000	0	0	0	0
elegedett	100	5000	0	0	0	0
hazas_e	100	5000	0	0	0	0
egyuttelok	100	5000	0	0	0	0
sajathaz	100	5000	0	0	0	0
lakastipus	100	5000	0	0	0	0
lakas_ev	100	5000	0	0	0	0
autok	100	5000	0	0	0	0
auto_sajat	90,16	4508	0	0	0	492
auto_kat	90,16	4508	0	0	0	492
kozl_eszk	100	5000	0	0	0	0
bejar_ido	99,96	4998	2	0	0	0
pol_konz	100	5000	0	0	0	0

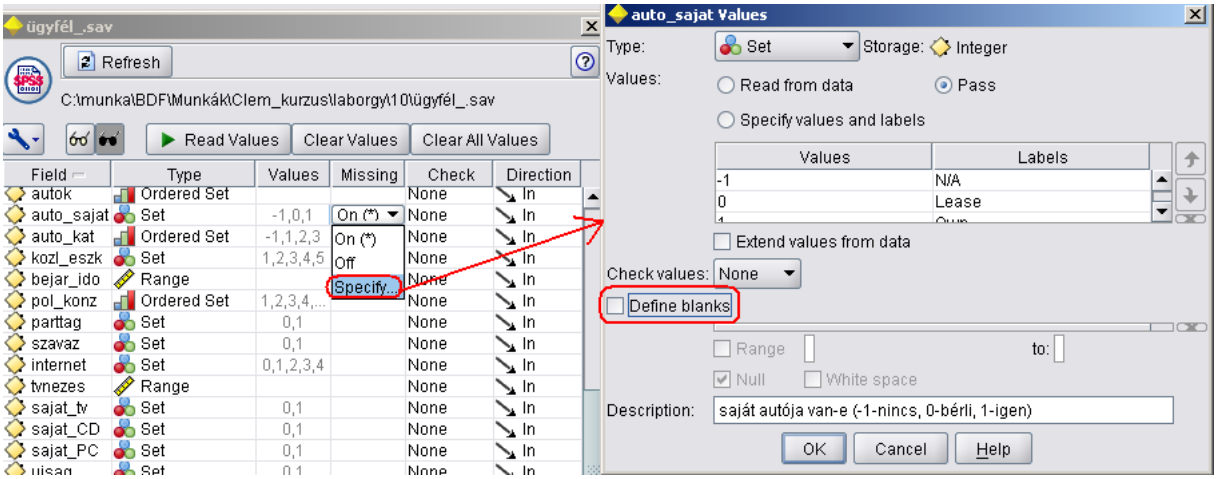
Négy mezővel van gond. A *bejar_ido* változó két kivétel-rekordja nem zavaró, az *auto_sajat* és az *auto_kat* változók 492 darab „*blank*”, azaz -1 értéket tartalmaznak, ez azt jelenti, hogy ennyi ügyfél nem rendelkezik autóval. Ez olyan jellegű információ, amit adatként lehetne tekinteni!

Az *ok* változó nagyobb gondot jelent. Az adatok 81 százalékában ez ténylegesen hasznavehetetlen információt szolgáltat, ezt a mezőt vegyük ki a további vizsgálatokból!

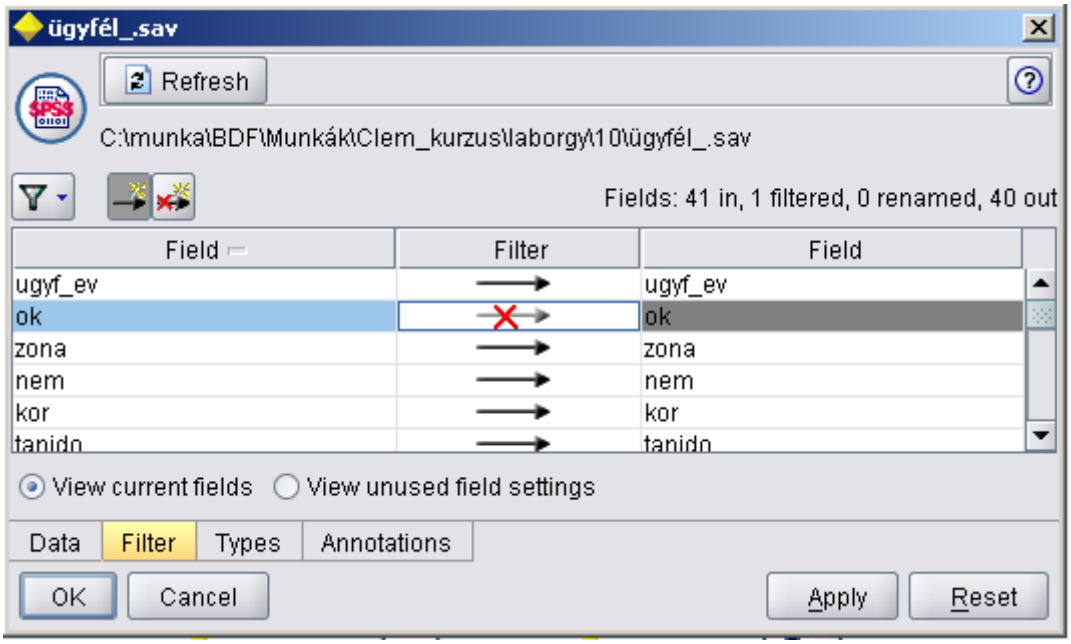
Állítsuk be a *Source node* TYPES fülén, hogy az *auto_sajat* és az *auto_kat* változó esetén a *-1* értékek érvényes adatként szerepeljenek!

Töröljük a *Source node* FILTER fülén az *ok* mezőt!

A *Source node* **TYPES** füléről indulva az *auto_sajat* és az *auto_kat* soroknál kell a pirossal jelölt helyen a **DEFINE BLANKS** jelölőnégyzetből a pipát kivenni:



A *Source node* **FILTER** fülén kell az ok változót törölni:

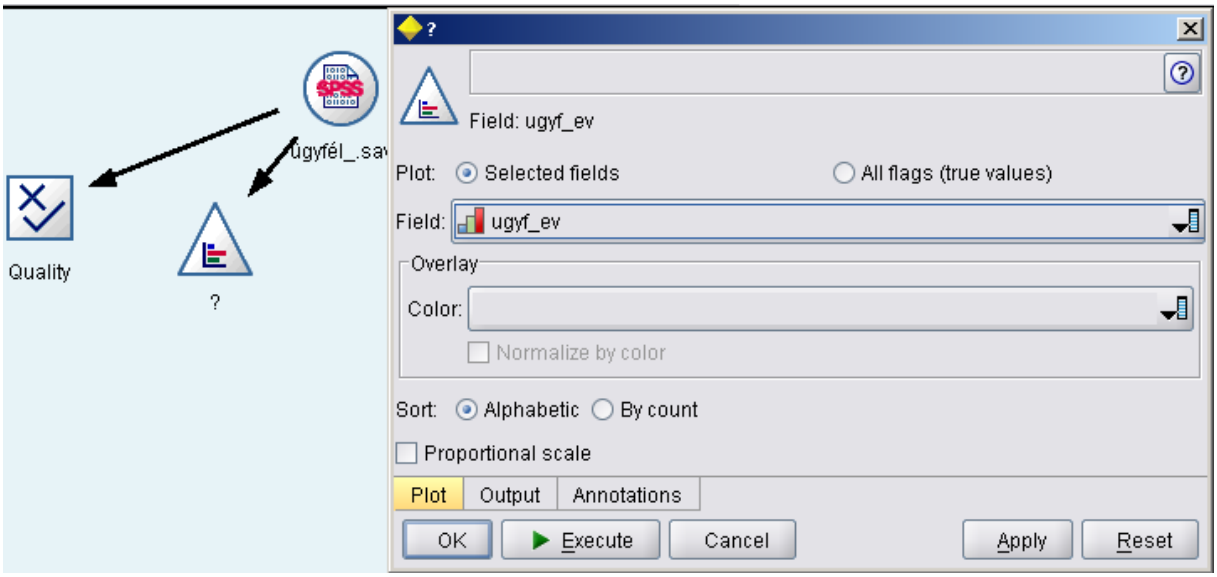


A célváltozó vizsgálata, átalakítása

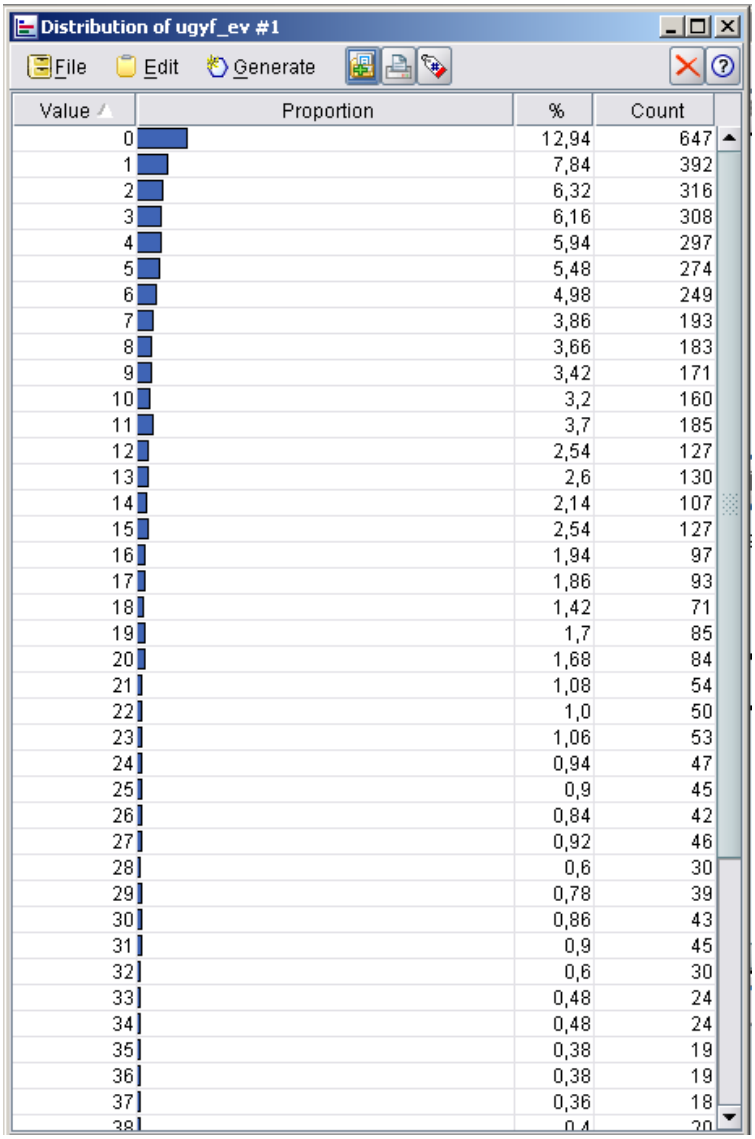
Célszerű megnézni az *ugyfel_ev* adatértékeinek megoszlását, nincs-e túl sokféle adatérték. (Ekkor a modellek nagy hibával dolgoznak, és nem lényeges különbözőségeket is lényegesnek tekintenek.)

Rajzoltassuk ki a célváltozó megoszlását!

Ezt pl. egy *Distribution* grafikonnal tehetjük meg.



Az eredmény:



Distribution of ugyf_ev #1

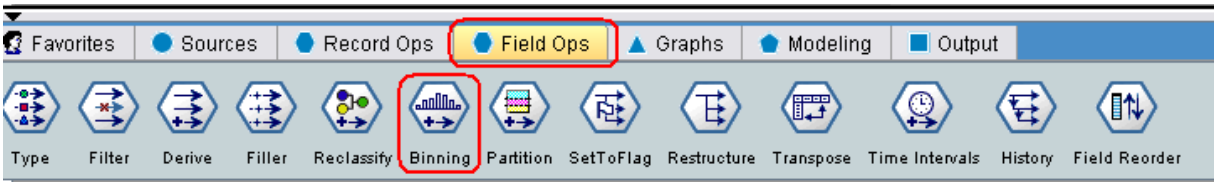
File Edit Generate

Value	Proportion	%	Count
0		12,94	647
1		7,84	392
2		6,32	316
3		6,16	308
4		5,94	297
5		5,48	274
6		4,98	249
7		3,86	193
8		3,66	183
9		3,42	171
10		3,2	160
11		3,7	185
12		2,54	127
13		2,6	130
14		2,14	107
15		2,54	127
16		1,94	97
17		1,86	93
18		1,42	71
19		1,7	85
20		1,68	84
21		1,08	54
22		1,0	50
23		1,06	53
24		0,94	47
25		0,9	45
26		0,84	42
27		0,92	46
28		0,6	30
29		0,78	39
30		0,86	43
31		0,9	45
32		0,6	30
33		0,48	24
34		0,48	24
35		0,38	19
36		0,38	19
37		0,36	18

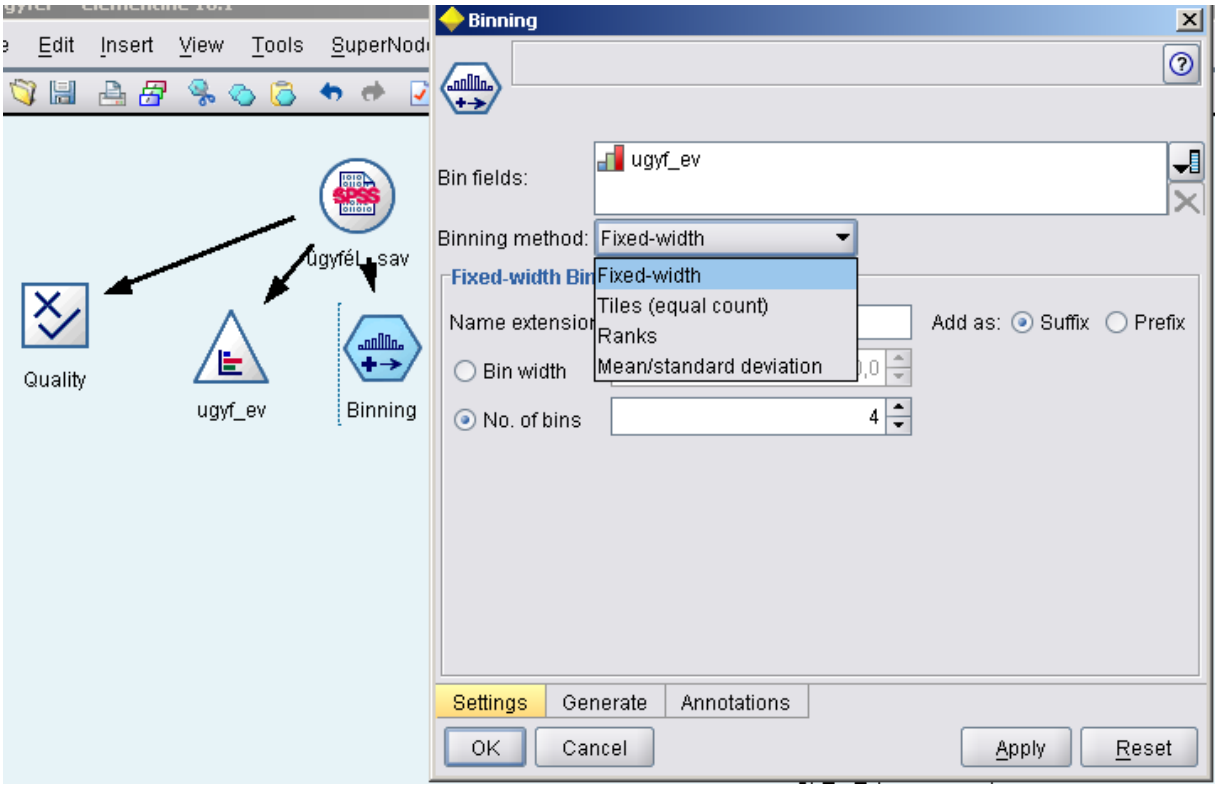
Az eredmény azt mutatja, hogy nem érdemes megkülönböztetni a célváltozó 0-52 éves értékeit, célszerű a változót egyszerűbbé tenni. A statisztikában gyakori probléma ez,

ennek kivédésére az adatokat osztályokba sorolják. A leggyakrabban használt osztályozási szempont az, hogy a szórás hányszorosának megfelelő távolságra helyezkedik el egy adat az adatok átlagától.

Készítsünk egy új mezőt, mely az *ugyf_ev* változó adatainak átlagtól való eltérése alapján sorolja az ügyfeleket legfeljebb öt csoportba!



Ezt egy **BINNING NODE** fogja elvégezni. Adjunk a stream-hez egy ilyen node-ot a *source node* után, s nézzük a beállításait:



A *Binning node*-okat jól használhatjuk nagy adattömeg egyszerűsítésére. Ezek mindig új mező(ke)t generálnak, melynek neve a meglévő mezőnév kiegészítésével keletkezik.

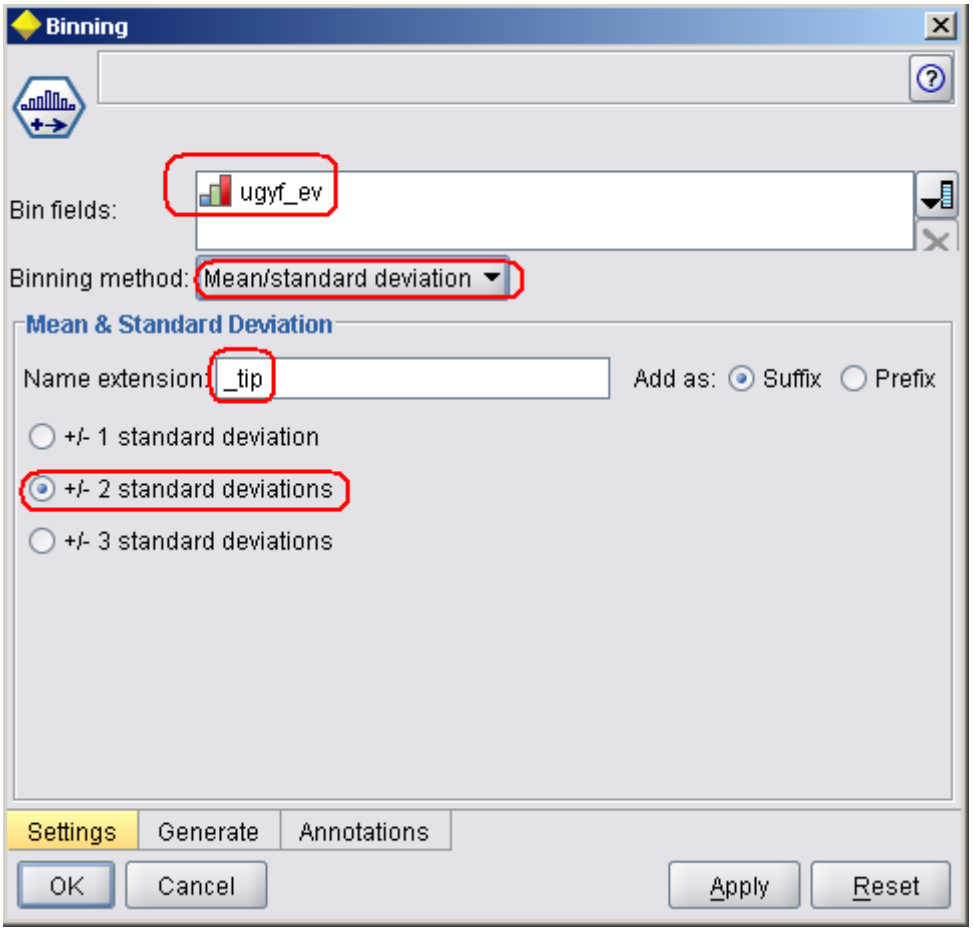
A **SETTINGS** fül **BINNING METHOD** listájából a sávokra osztás módszerét adhatjuk meg:

FIXED WIDTH: adatértékek szerinti azonos szélességű sávok képzése, **TILES:** azonos rekordszámú részekre bontás, **RANKS:** növekvő vagy csökkenő sorrendű sorszámozás, a **MEAN/STANDARD DEVIATION** lehetőség pedig aszerint képez sávokat, hogy az adatok átlagától pozitív vagy negatív irányban a szórás hányszorosának megfelelő távolságra van az adott adat. (Erre lesz szükségünk most.)

Az **ADD AS: SUFFIX** beállítás a név után, az **ADD AS PREFIX** a név elé teszi a megfelelő mezőjelölő szócskát. (A mi stream-ünknél ez a _tip kódszó lesz, mely utal az ügyfél típusára.)

Minden esetben generálhatunk a **GENERATE** fül segítségével egy új *Derive node*-ot is, mely számítások végzése nélkül, csak logikai feltételekkel készíti el a sávokat. Ez a lehetőség csak akkor használható, ha az adott *Binning node*-ot egyszer már futtattuk.

A következő beállításokat kell megtennünk:



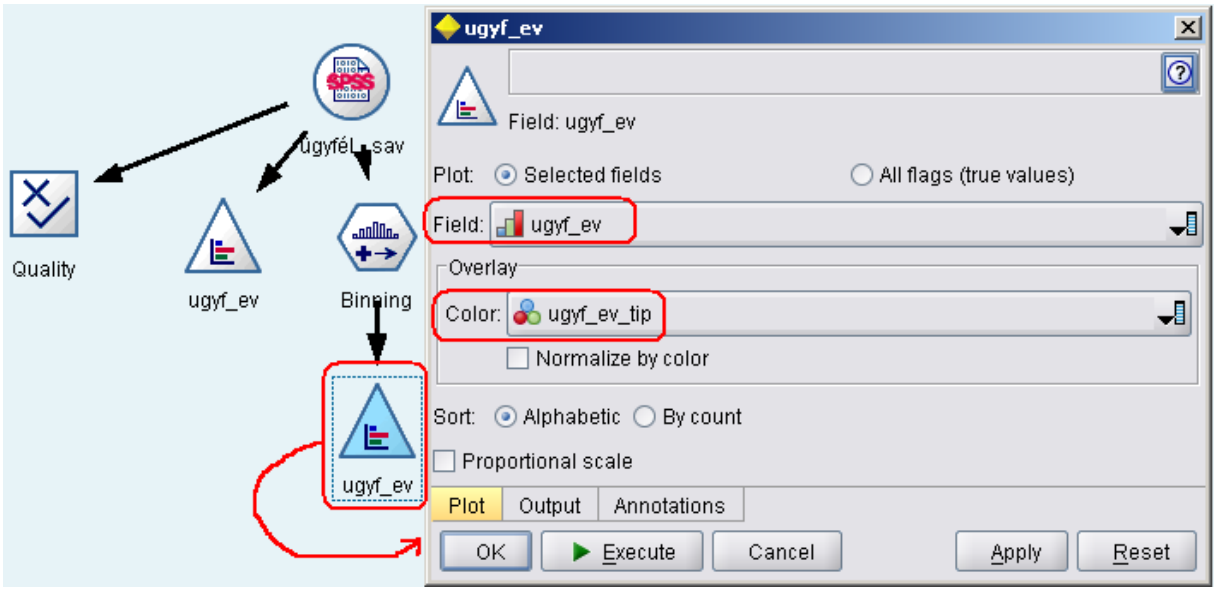
Itt a **+/- 2 STANDARD DEVIATIONS** beállítás azt jelenti, hogy maximálisan 5 csoportot képzünk a rekordokból. A számegyenesen szemlélteti ezt az alábbi ábra:

A generált egyszerűsített mező értékei

Legszemléletesebben ennek eredményét egy *Distribution* grafikonnal nézhetjük meg, úgy, hogy a keletkezett új *ugyfel_ev_tip* mező (a vizsgálat szempontjából már ez is változó!) szerint színezzük az imént már kirajzolt megoszlási grafikont.

Figyeljük meg az *ugyfel_ev_* és az *ugyfel_ev_tip* változók viszonyát egy színezett megoszlási grafikonon!

Az alábbi ábra alapján kell eljárni:



Az egyszerűsítés megfelelő, ezt mutatja a futtatás:

Value	Proportion	%	Count
0		12,94	647
1		7,84	392
2		6,32	316
3		6,16	308
4		5,94	297
5		5,48	274
6		4,98	249
7		3,86	193
8		3,66	183
9		3,42	171
10		3,2	160
11		3,7	185
12		2,54	127
13		2,6	130
14		2,14	107
15		2,54	127
16		1,94	97
17		1,86	93
18		1,42	71
19		1,7	85
20		1,68	84
21		1,08	54
22		1,0	50
23		1,06	53
24		0,94	47
25		0,9	45
26		0,84	42
27		0,92	46
28		0,6	30
29		0,78	39
30		0,86	43

ugyf_ev_tip

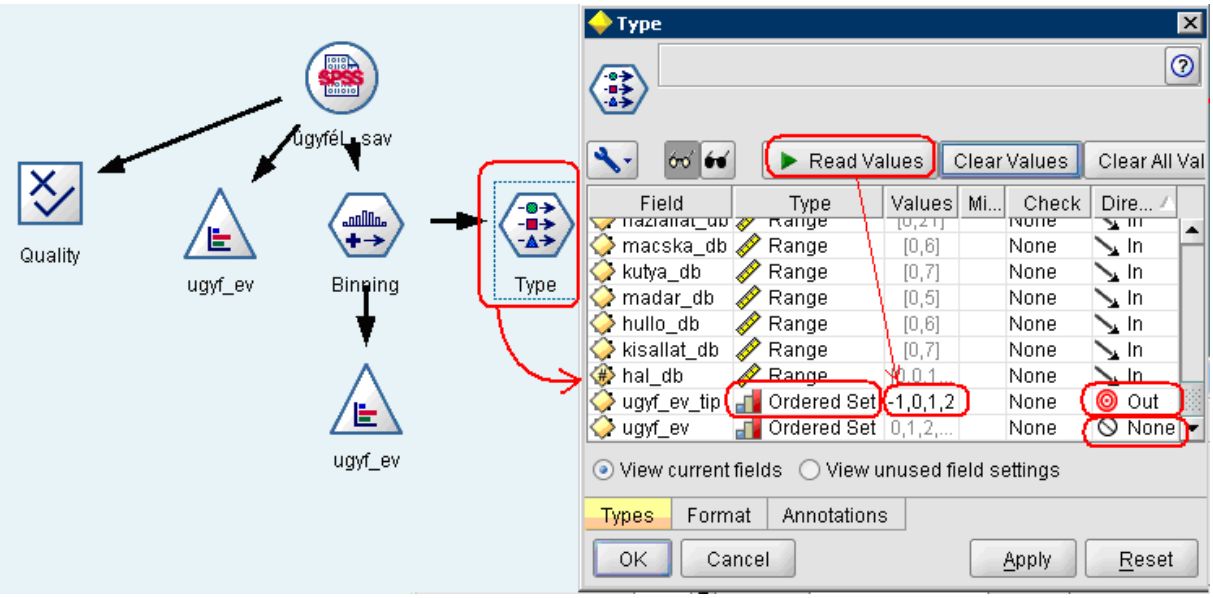
-1 0 1 2

Négy kategória keletkezett, az elsőben a friss ügyfelek, a másodikban a 20 évnél régebbi, a harmadikban a 20 és 30 év közötti, a negyedikben a 30 évnél régebbi ügyfelek vannak. A kategóriaértékek: -1, 0, 1 és 2. Látható, hogy -2 értékű adatok nem keletkeztek, ezt az *ugyfel_ev* változó erős antiszimmetriája indokolja.

Modellépítés előkészítése, lényeges változók

Egy *Type node* segítségével adjuk meg célváltozónak az eddigi *ugyf_ev* helyett az *ugyf_ev_tip* változót!

Az alábbi beállításokra van szükség:



Az *ugyf_ev_tip* változót **ORDERED SET** típusúvá kell tenni, a **VALUES** oszlopnál *<Read>*-re állítva be kell olvastatni az adatokat a **READ VALUES** gombbal, a **DIRECTION** oszlopban pedig *Out*-ra kell állítani. Az *ugyf_ev* változó nem vesz részt a modellépítésben, ezért itt *None*-ra van kapcsolva.

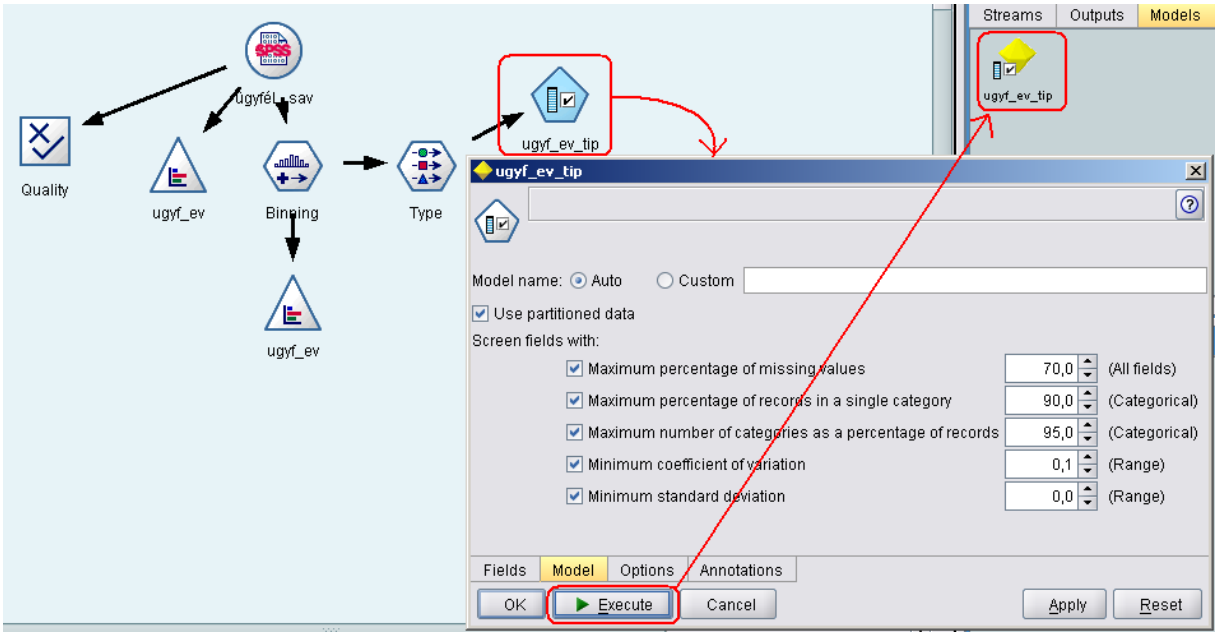
E két változó sora azért egymás alatt látszik az ábrán, mert a beállítások után a **DIRECTION** oszlop tartalma szerinti növekvő sorrend lett bekapcsolva.

Ezután a már tanult módon ki szeretnénk választani a célváltozó szempontjából szóba jöhető változókat.

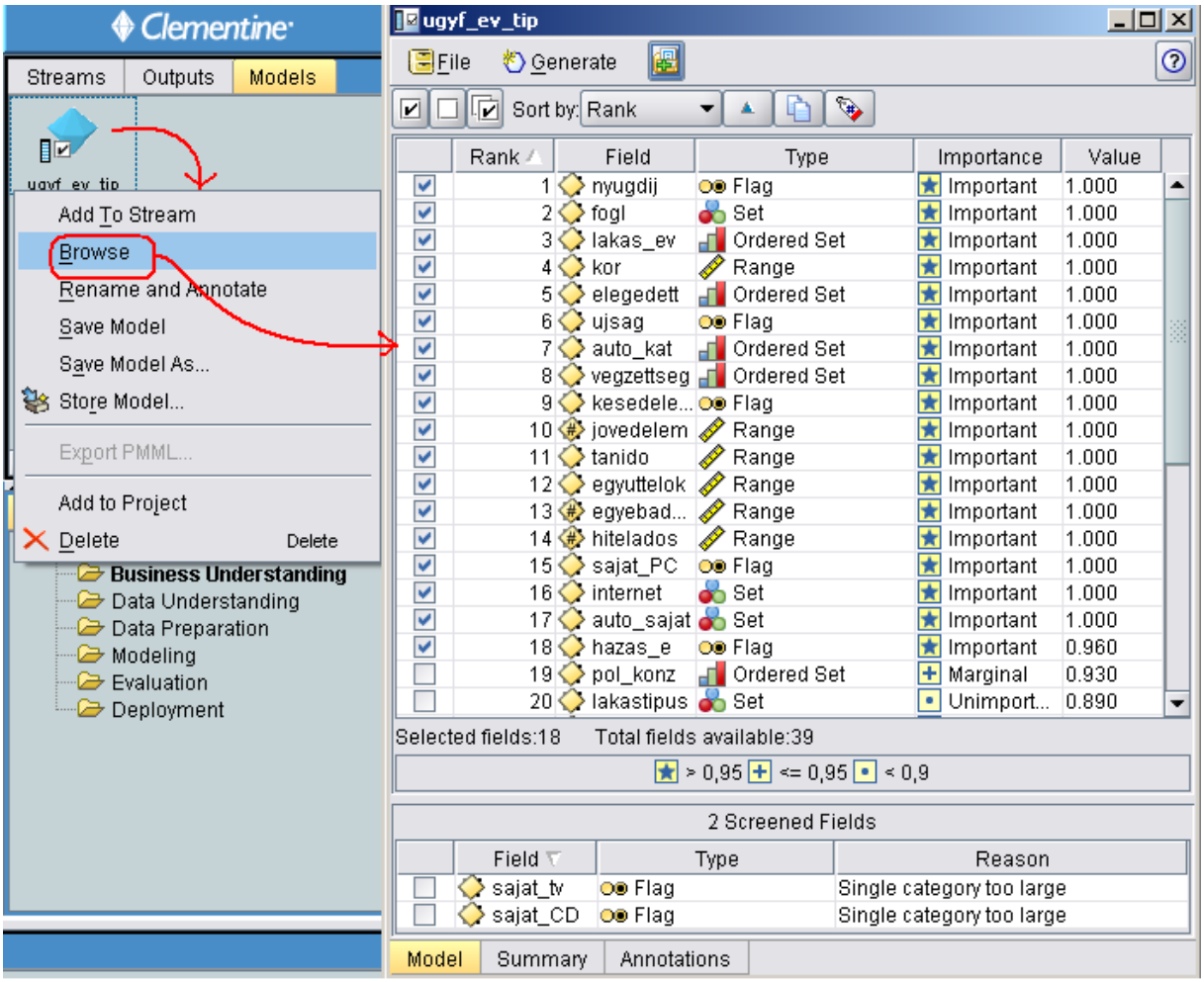
Egy *Feature Selection node* segítségével építsünk olyan modellt, mely megadja az *ugyf_ev_tip* szempontjából releváns változókat!

Egy *Feature Selection* modellépítő node-ot kell beilleszteni a *Type node* után. Ennek párbeszédpaneljén megtarthatjuk a gépi beállításokat, majd lefuttathatjuk. A jobb felső *Models* ablakban jelenik meg a kész modell.

Az alábbi ábra mutatja a node-ot, a beállításait, és a futtatás eredményét:



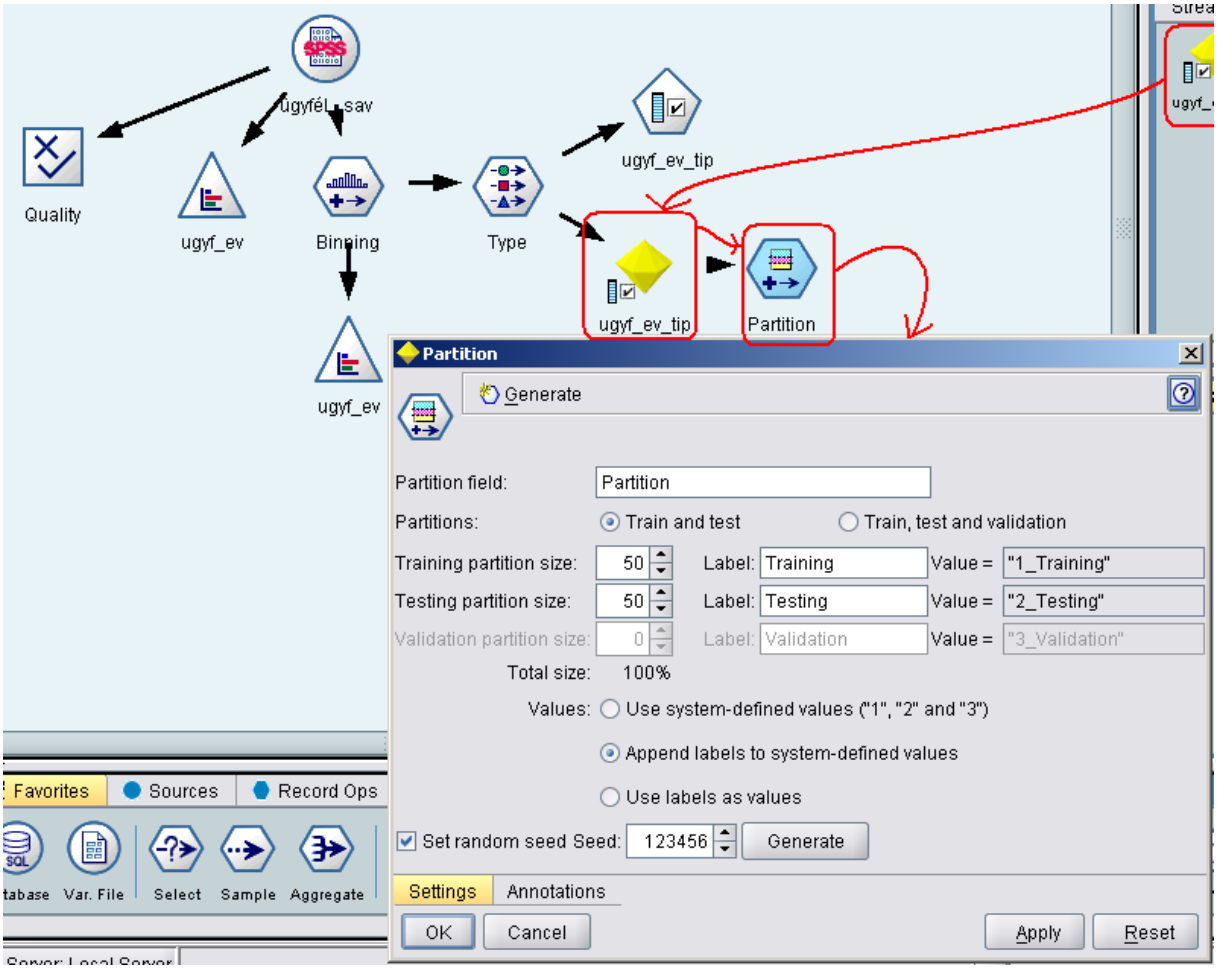
Figyeljük meg a kész modellben a fontosnak jelölt változókat!



Az ábra mutatja, hogy 18 darab (célváltozó szempontjából) fontos változó van az adathalmazban. Két változó eleve kiesik a vizsgálatból, mert a rekordok döntő részében ugyanazt az egyetlen értéket tartalmazzák.

Építsük be a kész modellt a *Type node* után, majd válasszuk szét a rekordok halmazát egy modellépítő és egy tesztelő részre!

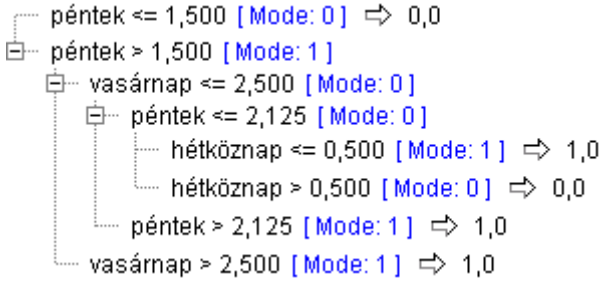
Az alábbi ábra mutatja a tennivalókat:



A két rekordhalmaz garantálja, hogy az 5000 ügyfél véletlenszerűen kiválasztott egyik fele alapján épüljön fel a döntési fa modell, a rekordhalmaz másik felén pedig teszteljük majd a kész modellt.

Döntési fa algoritmusok és kész modellek

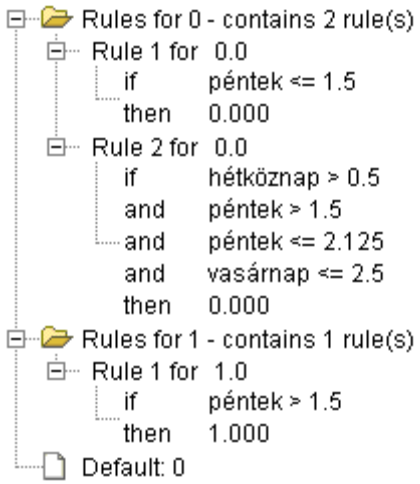
A döntési fa (decision tree) modellek döntések sorozatain keresztül adják meg egy-egy rekordban a bemeneti változók értékeiből a célváltozó számított („tippelt") értékét. A modellben szereplő döntések egyszerű feltételekre adott válaszok. A feltételek megkövetelhetnek *Range* típusú változó esetében intervallumhoz való tartozást, Set típusok esetén pedig egy bizonyos részhalmazhoz való tartozást:



E döntési fában például egy 0 vagy 1 értéket felvevő célváltozó számítási szabályai olvashatók.

Például ha a *péntek* változó 1,5-nél nagyobb értékű, a *vasárnap* változó 2,5-nél kisebb vagy egyenlő és a péntek változó 2,125-nél is nagyobb, akkor a célváltozó értéke 1. A [Mode] értékek azt jelzik, hogy ha egy adott ponton abbahagynánk a fa olvasását, akkor ott éppen milyen értéket kell adni a célváltozónak, hogy a legkisebbet tévedjük.

A döntési fa áttekinthető, ám ugyanazok a feltételek egyszerűbb, csoportosított formában is felírhatók:



Ezt szabályhalmaznak (ruleset) nevezzük. A szabályokat folyamatosan kiolvasva kell értelmezni (if=ha, and=és, then=akkor). Ha egy rekordra egyik szabály sem érvényes, akkor az utolsó sorban álló *Default* érték legyen a számított célváltozóban.

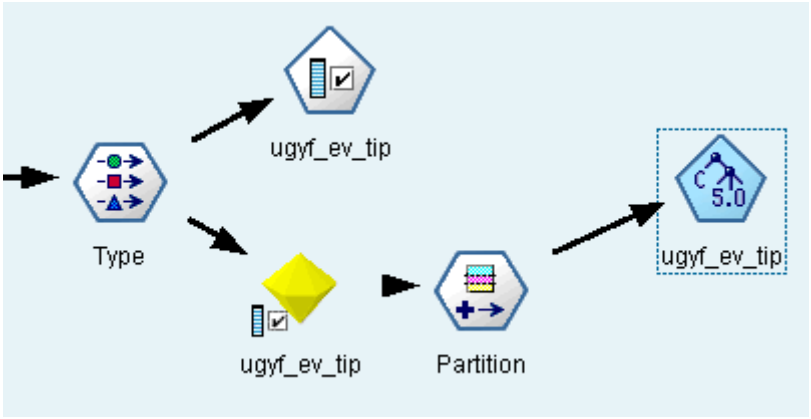
A döntési fa a bemeneti változók, a szabályhalmaz pedig a célváltozó szempontjából csoportosítva írja le ugyanazt a döntési struktúrát.

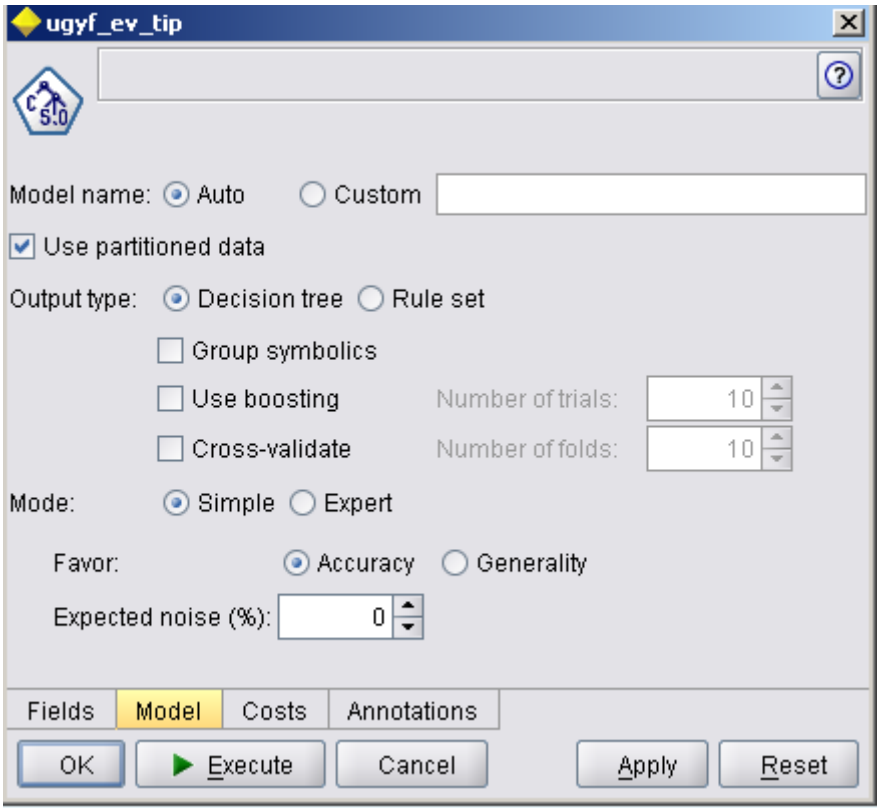
A faépítés építkező és lebontó lépések sorozata. Az első fázisban minden új ág építése a legnagyobb információnyereség statisztikai elve alapján halad tovább, a második fázisban, a fa tisztításánál túltanulásra utaló ágak lemetszése történik.

Ez utóbbira feltétlenül szükség van, hisz túltanítással elérhető az is, hogy az algoritmus akár 100%-os pontossággal adja meg a célváltozó értékeit a modellépítő rekordhalmazon.

C 5.0 döntési fa ***használata***

Illesszünk egy C5.0 modellépítő node-ot a Partition node után és figyeljük meg a node beállításait!





MODEL FÜL:

OUTPUT TYPE. Itt választani lehet, hogy a végeredményt döntési fa vagy szabályhalmaz alakban adja-e meg a node.

GROUP SYMBOLICS: Ha bejelöljük, akkor szimbolikus értékek hasonlósága alapján csoportosítva dolgozik. Ha például a HAJ mezőn belül a barna és a vörös értékek hasonló rekordokat adnak, akkor e két értéket együttesen vizsgálja. Ha nem jelöljük be, akkor minden hajszín egyenrangúként vesz részt a további faépítésben.

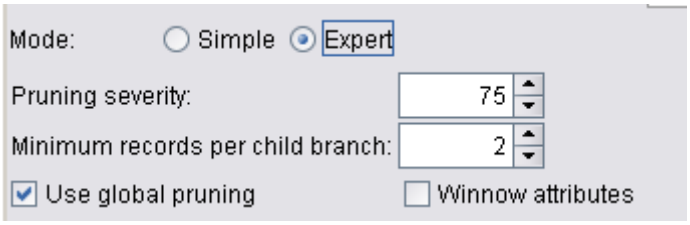
USE BOOSTING: A fa továbbfejlesztését végző algoritmus bekapcsolása. Az elsőként kapott döntési fát újravizsgálja, milyen rekordokban tér el a tényleges adatoktól, ezután olyan fa-részt generál, mely az eltéréseket korrigálja. Itt megadható az is, hányszori újravizsgálást kérünk. A fa bonyolultságát és a faépítés idejét egyaránt növeli.

CROSS-VALIDATE: Akkor célszerű használni, ha olyan kevés rekordunk van, hogy nem érdemes modellépítő és tesztelő rekordhalmazra felosztani. Részhalmazokat választ ki

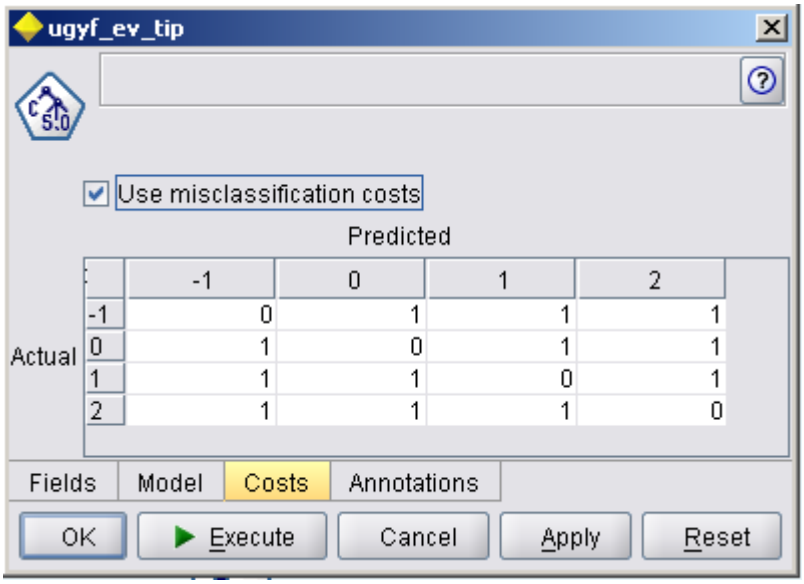
és azokon teszteli az újra és újra felépített fát, s olyan végeredményt ad, mely a többféle részhalmazon is hasonló eredményt ad.

MODE:

1. **SIMPLE** A gép beállításaival dolgozik a
   - **FAVOR:** A tanulás leállítását lehet megadni. Az **ACCURACY** beállításával az algoritmus sokáig tanul, a felépített fa az az egyedi adathalmaz egyedi tulajdonságait minél hűbben tartalmazza. Ha azonban tesztelő adatokkal folytatnánk a munkát, célszerűbb a túltanulást megelőzni a **GENERALITY** lehetőséggel.
   - **EXPECTED NOISE (%):** Zajos vagy hibás adatok elvárható aránya. (Ez is leállási feltétel.)



2. **EXPERT:** a felhasználó megadhat bizonyos belső paramétereket a modellépítő algoritmusnak
   - **PRUNING SEVERITY:** A lokális nyesésnél milyen egy faághoz tartozó minimális elemszám esetén generáljon új ágat. Az értéket növelve kisebb és tömörebb fát kapunk. Ha csökkentjük, pontosabb fát kapunk.
   - **MINIMUM RECORDS PER CHILD BRANCH:** A fa egy ágán levő minimális elemszám. E beállítással korlátozható a hasadások száma. Zajt tartalmazó adatok esetén célszerű ezt az értéket növelni. Alapértéke 2.
   - **USE GLOBAL PRUNING:** A faépítést két lépcsőben végeztethetjük e beállítással. Az első lépcsőben egy kevés ágú fát épít a gép, a levelein csoportosított adatokkal, majd a csoportokat helyi részfák építésével finomítja. Lassabb, de általában eredményesebb algoritmust ad. Alapértelmezésben ez a lehetőség be van kapcsolva.
   - **WINNOW ATTRIBUTES:** E beállítás esetén a faépítés előtt statisztikai vizsgálattal kiválasztja a fontos mezőket az algoritmus és a kevésbé fontosnak ítélt mezőket kihagyja a faépítésből. Akkor célszerű használni, ha sok mező van, vagy meg akarjuk előzni a sok mezőből származó túltanulást.



COSTS FÜL:

USE MISCLASSIFICATION COSTS: A faépítés során egy-egy új ág építése előtt összeadja a gép a tévedések értékét a szóbajöhető folytatásoknál. Ezután a folytatásról aszerint dönt, hogy mekkora tévedés-összeget eredményez az adott választás – mindig a legkisebb összeget fogja választani. A tévedéseket súlyozhatjuk az itt látható táblával, így számszerűen dominánsnak vagy jelentéktelennek adhatunk meg minden tévedéstípust.

Az **ACTUAL** értékek a célváltozó tényleges adatértékei, a Predicted értékek pedig a hozzájuk tartozó számított eredményeket jelentik. (Itt a -1, a 0, 1 és a 2 az *ugyf_ev_tip* változó lehetséges értékei.)

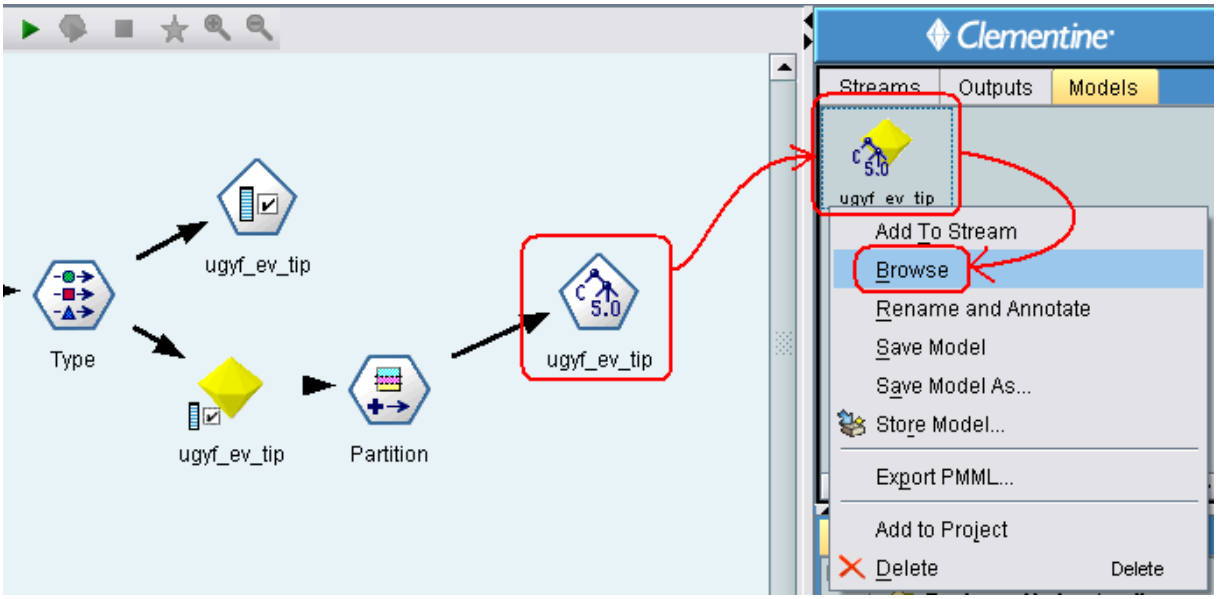
A tévedési súlyok táblázatából is következtethetünk a következő fontos követelményre:

A C 5.0 *algoritmus célváltozója csak diszkrét típus (Discrete, Ordered set, Set vagy Flag) lehet!*

Futtassuk a *C 5.0* modellépítő node-ot a gép által adott beállításokkal!

Tanulmányozzuk a kész modellt!

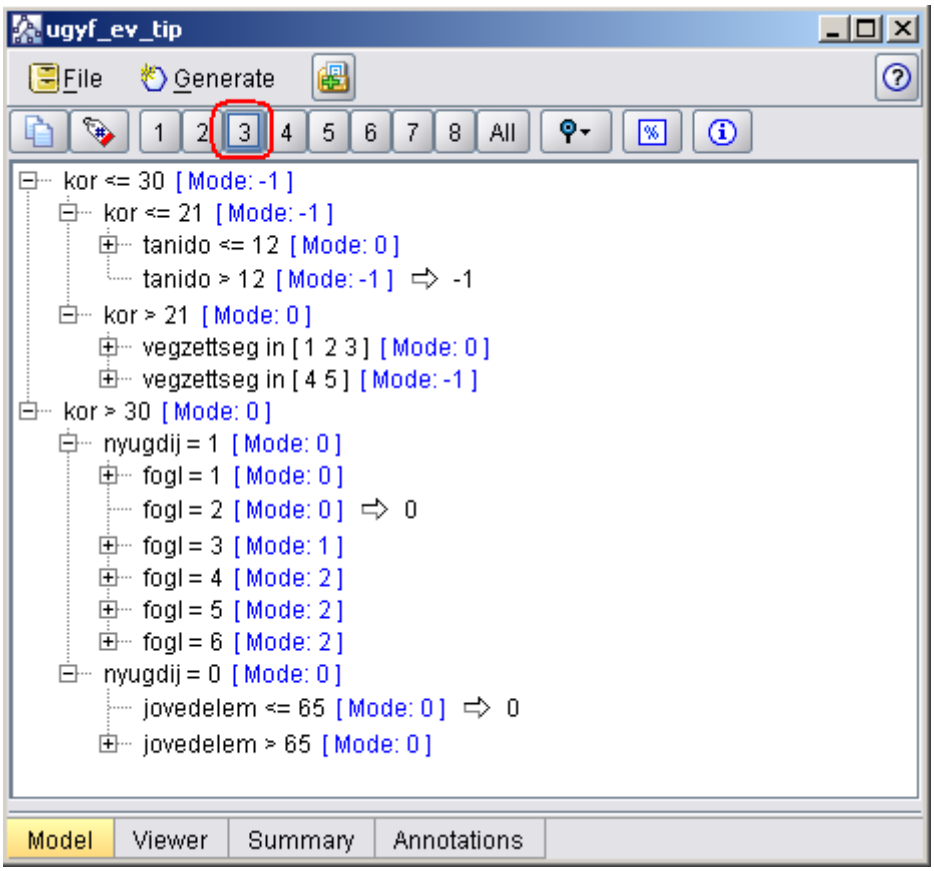
Ehhez a jobb felső *Models* ablakban megjelenő modell gyorsmenüjében a *Browse* menüpontot kell választanunk:



A kész modell szerkezetét fogjuk látni.

Bontsuk ki a kapott döntési fát 3 szint mélységig!

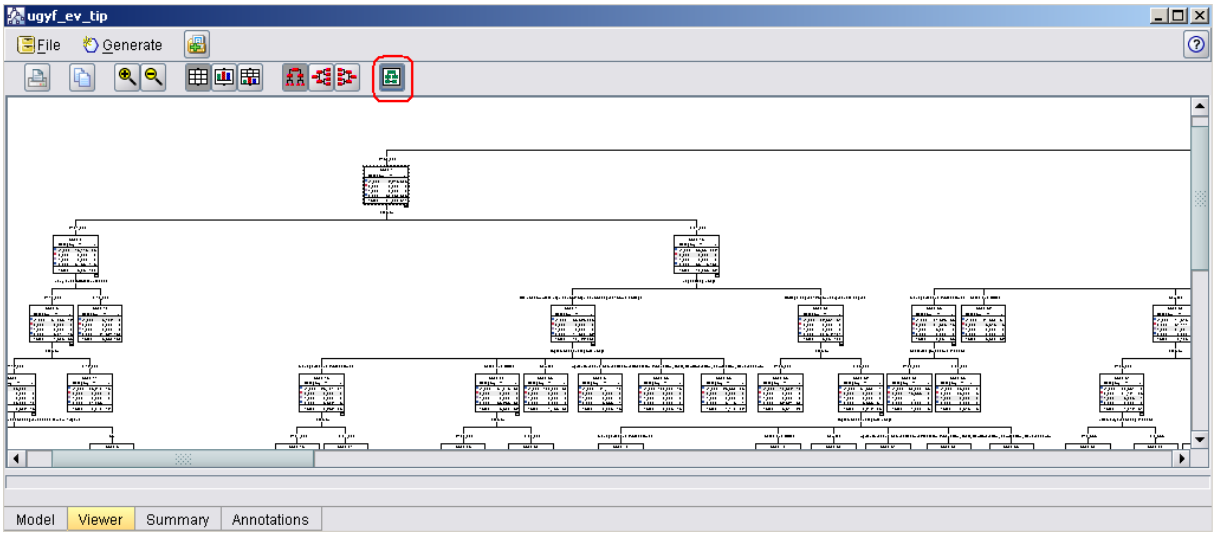
Ehhez az ábrán pirossal jelzett 3-as gombra kell kattintani:



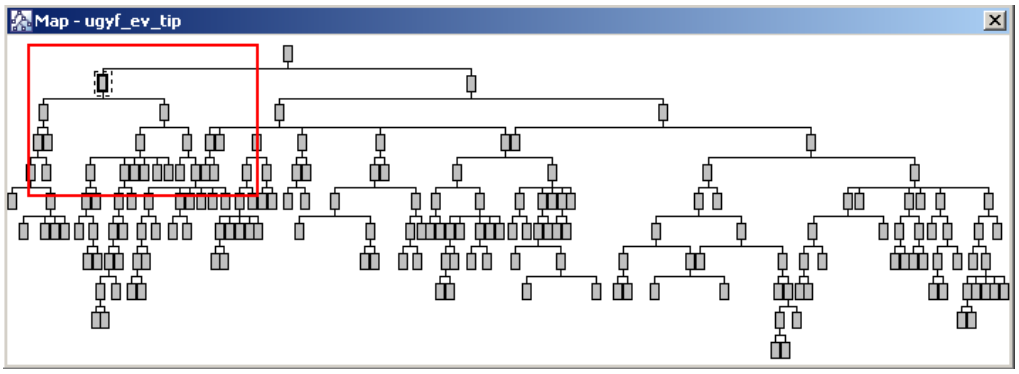
Látható, hogy abban, hogy milyen régi egy-egy ügyfél, a legfontosabb meghatározó a *kor* változó – ez logikus is. A 30 év felettiek esetében a nyugdíjas állapot a következő lényeges változó, a 30 év feletti nyugdíjasok esetében a foglalkozás lényeges, stb. Így szemléletesen végigkövethető a teljes döntési mechanizmus.

A párbeszédpanel tetején látható gombokkal azt lehet szabályozni, hogy hány döntési szintet mutasson az ablak, az **ALL** hatására minden szintet kibont.

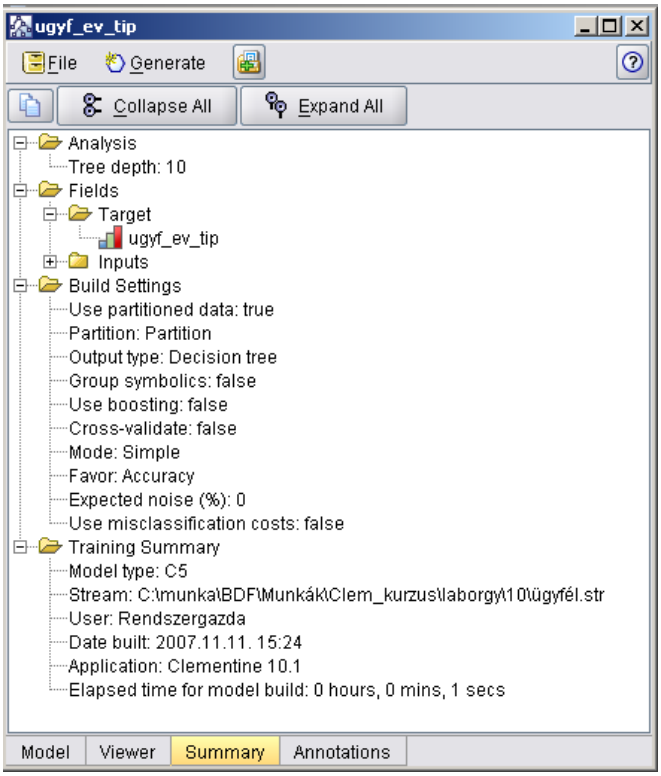
A **VIEWER** fülön a döntési fa gráfként látható. Kibontható, kicsinyíthető és nagyítható, elrendezést és megjelenítést lehet beállítani.



A fenti ábrán pirossal jelzett gomb egy térképet mutat a gráfról:



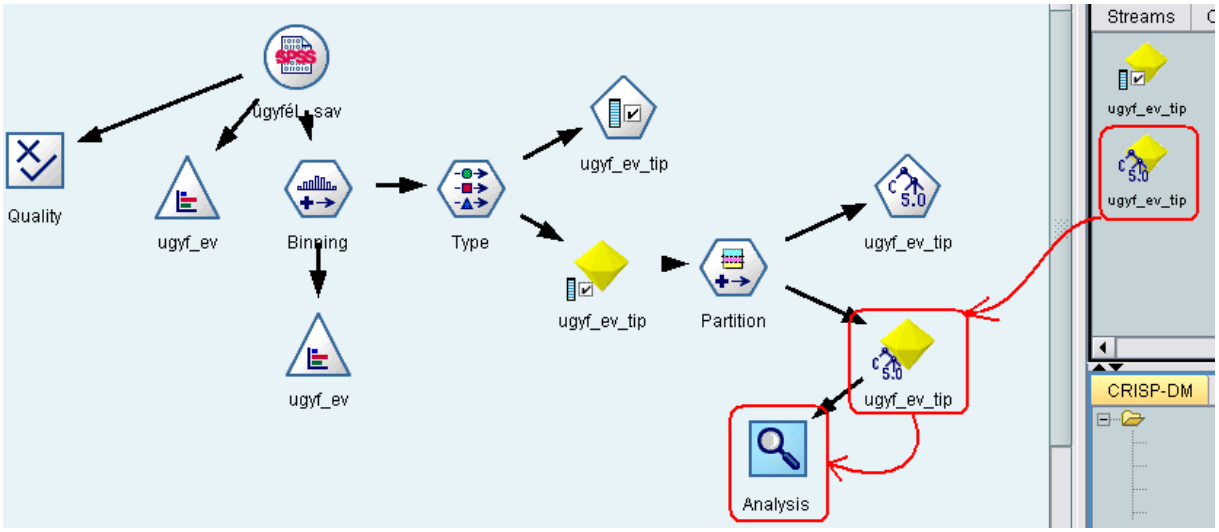
A **SUMMARY** fülön áttekintő összegzést kaphatunk a modellről. (**EXPAND ALL:** mindent mutat)



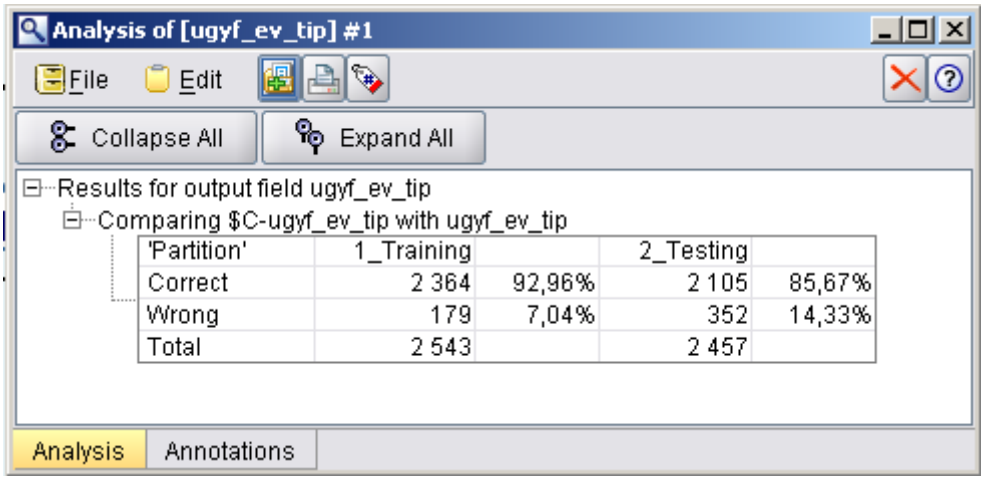
Itt többségben a modellépítő node beállításait látjuk, de szerepel pl. a *Tree depth*, azaz a fa mélysége (szintjei száma) is.

Illesszük be a kész modellt a *Partition node* után, majd elemeztessük a Modeler**-rel a kész modellt a két rekordhalmazon!**

Az ábra szerint kell eljárni:



Az Analysis node-ot már ismerjük, a célváltozót hasonlítja össze az azonos nevű számított változók értékeivel. Hagyjuk meg a gépi beállításait, majd futtassuk!



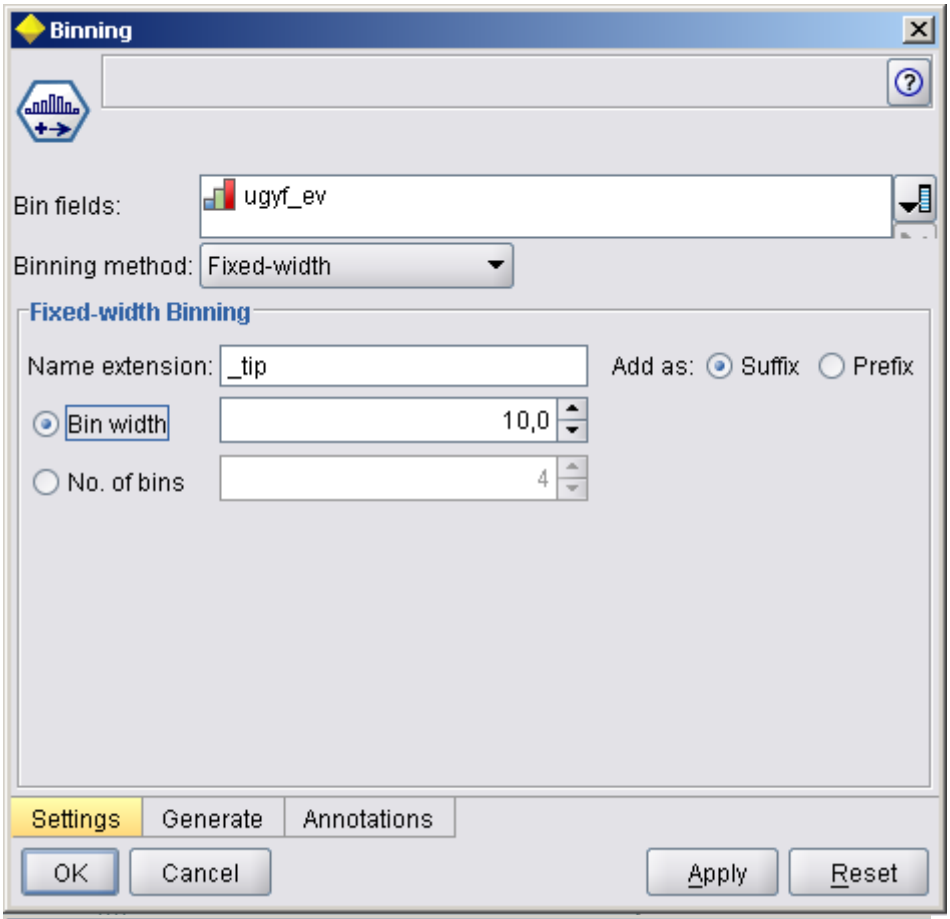
'Partition'	1_Training		2_Testing	
Correct	2 364	92,96%	2 105	85,67%
Wrong	179	7,04%	352	14,33%
Total	2 543		2 457	

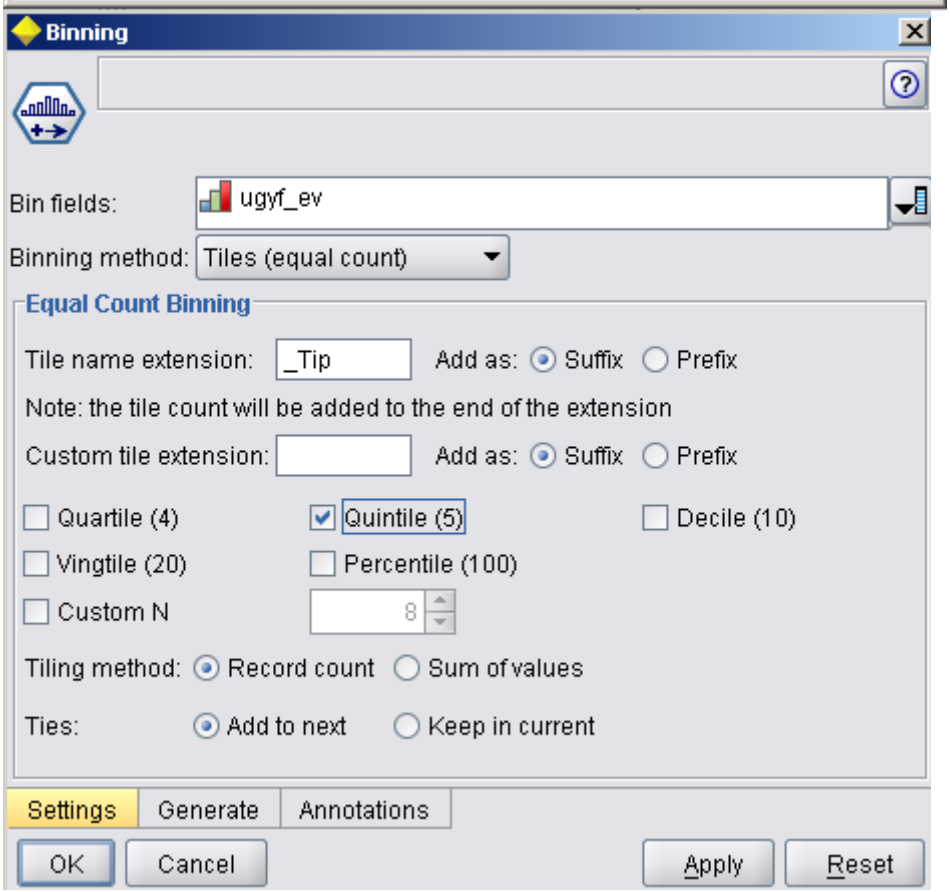
Látható, hogy a modellépítő adathalmazon 93%-ban helyesen számol a döntési fa, de a tesztelő adathalmazon csak 86%-ban. Ennek valószínű oka a túltanulás.

Gyakorló feladatok:

Készítsünk más típusú adategyszerűsítést a Binning node-dal! Először 10 szélességű sávokkal, majd 5 azonos elemszámú résszel dolgozva építsünk döntési fát! Nézzük meg az új modell „jóságának" elemzését is!

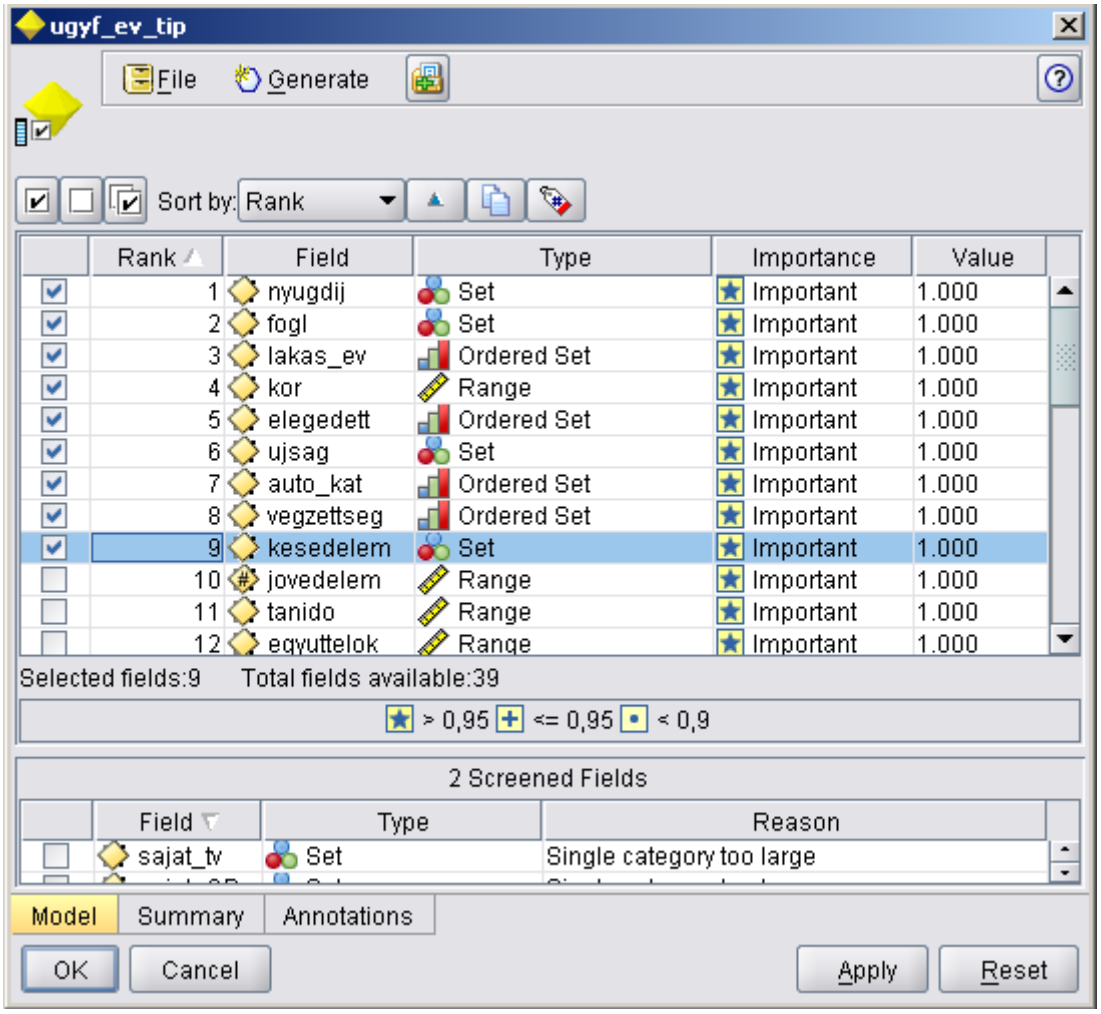
Segítség:


Binning
Bin fields:
ugyf_ev
Binning method:
Fixed-width
Fixed-width Binning
Name extension:
_tip
Add as:
Suffix
Prefix
Bin width
10,0
No. of bins
4
Settings
Generate
Annotations
OK
Cancel
Apply
Reset


Binning
Bin fields:
ugyf_ev
Binning method:
Tiles (equal count)
Equal Count Binning
Tile name extension:
_Tip
Add as:
Suffix
Prefix
Note: the tile count will be added to the end of the extension
Custom tile extension:
Add as:
Suffix
Prefix
Quartile (4)
Quintile (5)
Decile (10)
Vingtile (20)
Percentile (100)
Custom N
8
Tiling method:
Record count
Sum of values
Ties:
Add to next
Keep in current
Settings
Generate
Annotations
OK
Cancel
Apply
Reset

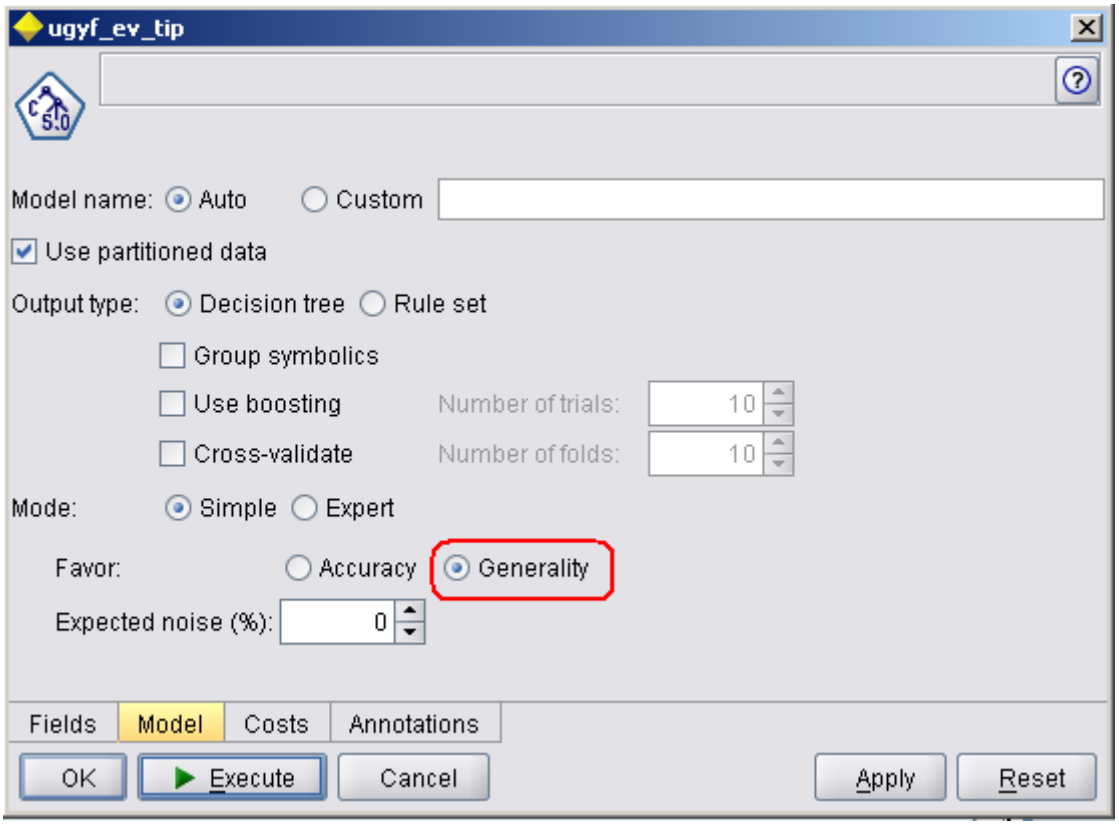
A Feature Selection node-nál csak a 10 legfontosabb változót hagyjuk meg, így építsünk döntési fát! Nézzük meg az új modell „jóságának" elemzését is!

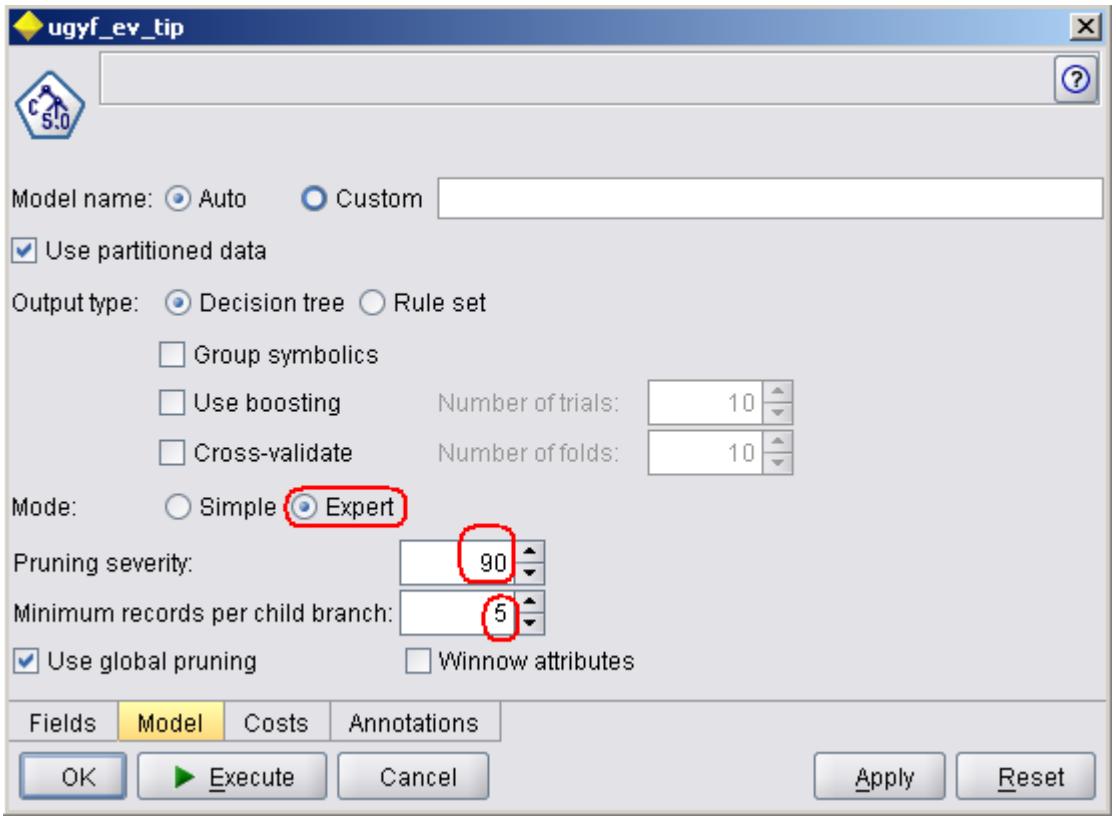
Segítség:



Csökkentsük a túltanulást a modellépítő node beállításai segítségével! Próbáljuk ki a SIMPLE módszer GENERALITY beállításával, majd az EXPERT módszer PRUNING SEVERITY=90 és MINIMUM RECORDS PER CHILD BRANCH=5 beállításával! Nézzük meg mindkét módszer által épített modell „jóságának" elemzését is!

Segítség:

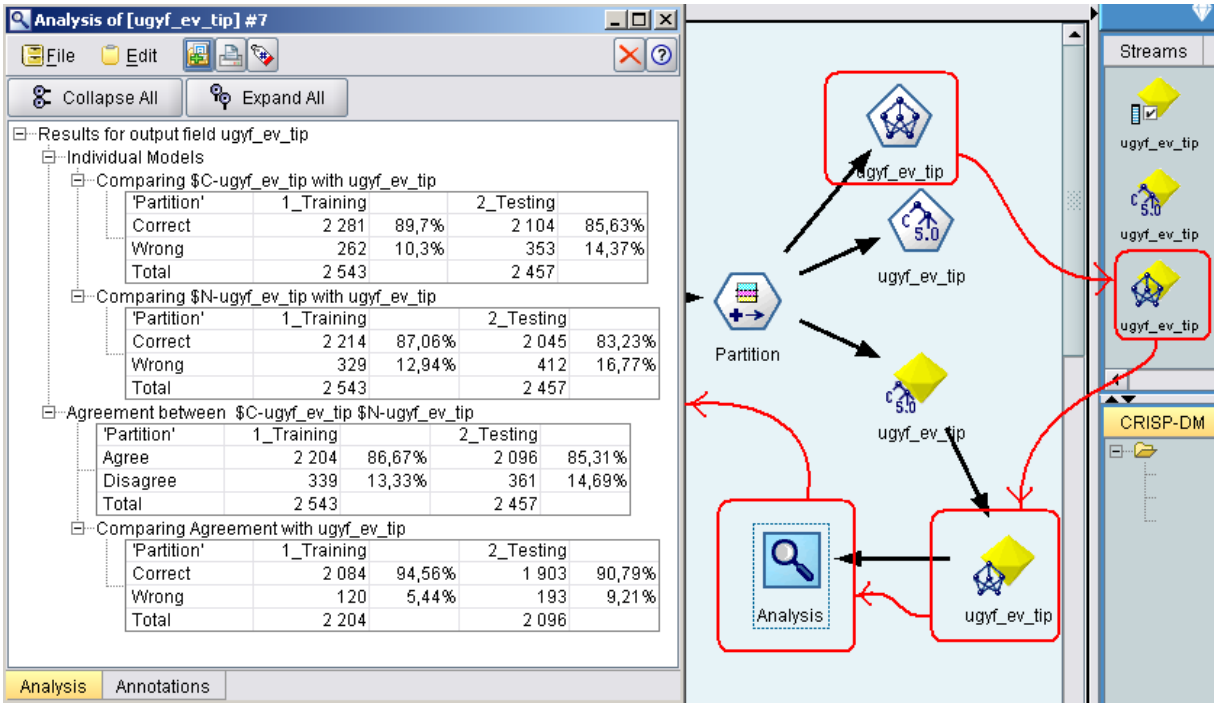




A túltanulást csökkenteni tudjuk mindkét változtatással, de nem tudunk eredményesebb modellt adni a tesztelő rekordhalmazra.

Az ebben a gyakorlatban megépített C5.0 modellt építsük meg neurális hálóként is és elemeztessük a kettőt együtt egy Analysis node-dal!

Segítség:



Az esetek többségében a neurális háló és a döntési fa ugyanannyira eredményes. Ez annak köszönhető, hogy a célváltozóra vonatkozó információt a többi adatokból mind a kettő módszer hasonló hatásfokkal képes „kibányászni".

Próbáljunk egy új streammel más célváltozóhoz modellt építeni, például a *pol_konz* **(politikai beállítódás) változóhoz!**

Megfigyelhető, hogy nem minden adathalmazból és nem minden célváltozóra lehet jó modellt építeni. Ennek oka, hogy bizonyos változók nem adnak elég információt más adatokra nézve. Az adatbányászat célja éppen az, hogy felkutassa és megadja a világban meglévő összefüggéseket és megmondja, mely adatok között nem érdemes összefüggést keresni.

10. Webes adatbányászat

E gyakorlat célja a webbányászat alapjainak megismerése. A Clementine-hoz kiegészítésként telepíthetők olyan szoftver-részek, melyek segítségével web-serverek látogatottságát, a látogatók viselkedését vizsgálhatjuk.

A web-server szoftverek különféle esemény-naplózási lehetőségeket kínálnak. A rendszergazdák a biztonság és a látogatottság figyelése céljából naplóztatják a web-oldalakon történő eseményeket, ezekből a naplófájlokból (**LOG FÁJLOKBÓL**) dolgozik a Clementine Web Mining.

A most következő két laborgyakorlat során a szombathelyi Berzsenyi Dániel Főiskola portáljának log fájljaiból fogunk dolgozni.

Minden adatbányászati munka első lépése az adathalmaz alapos ismerete. A webbányászati algoritmusok használata előtt tanulmányozni kell az elemezendő portál részeit.

Íme a portál kinézete (www.bdf.hu):

A portálhoz tartozó eseményként naplózva van minden kérés, amellyel egy user kommunikál a portállal - általában valamilyen fájlt kér le a web-serverről. (Legtöbbször egy linken való kattintás hatására történik ez.) A kéréseket a log fájl tárolja.

Íme egy log fájl első néhány bekezdése:

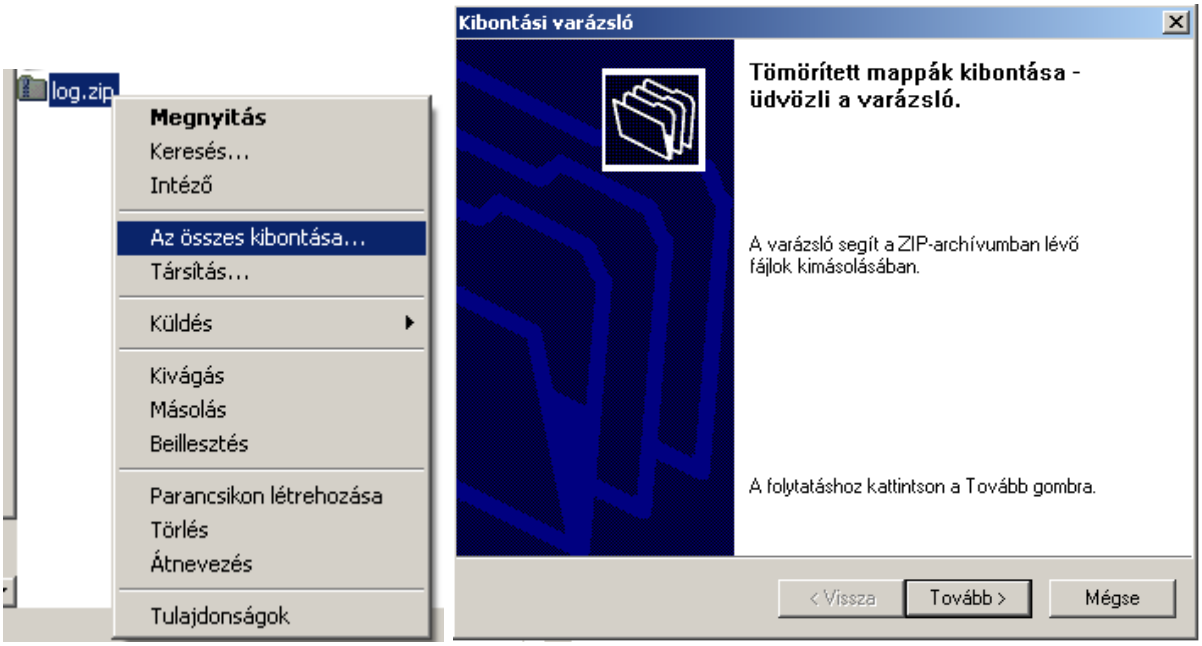
```
#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2006-08-28 00:05:39
#Fields: date time s-sitename s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status sc-substatus sc-win32-status
2006-08-28 00:05:39 W3SVC855251 193.224.74.10 GET /_vti_bin/owssvr.dll - 80 - 72.30.98.75 Mozilla/5.0+(compatible;+Yahoo!+Slurp;+http://help.yahoo.com/help/us/ysearch/slurp) 404 0 0
2006-08-28 00:05:39 W3SVC855251 193.224.74.10 GET /konyvtar/munkatars.htm - 80 - 72.30.252.107 Mozilla/5.0+(compatible;+Yahoo!+Slurp;+http://help.yahoo.com/help/us/ysearch/slurp) 304 0 0
2006-08-28 00:09:22 W3SVC855251 193.224.74.10 GET /_vti_bin/owssvr.dll - 80 - 195.70.35.179 KummHttp/1.1+(compatible;+KummClient;+Linux+rulez) 302 0 0
2006-08-28 00:09:22 W3SVC855251 193.224.74.10 GET /Default.aspx - 80 - 195.70.35.179 KummHttp/1.1+(compatible;+KummClient;+Linux+rulez) 200 0 64
2006-08-28 00:09:23 W3SVC855251 193.224.74.10 GET /_vti_bin/owssvr.dll - 80 - 64.4.8.130 msnbot/1.0+(+http://search.msn.com/msnbot.htm) 404 0 0
2006-08-28 00:09:25 W3SVC855251 193.224.74.10 GET /konyvtar/macslist.html - 80 - 64.4.8.130
```

Ezt kapja bemenetként a **WEB MINING (SOURCE) NODE.** (Az adatbányászat a Clementine-ban egyetlen Suorce node-nak, a Web mining node-nak használatát jelenti.) Az eredeti szoftver nincs felkészítve a web-bányászatra, de a Web Mining kiegészítő csomagot feltelepítve ezzel a node-dal bővül a Clementine **SOURCE NODE** palettája. A csomag tartalmaz még ezen kívül számos használható demo-streamet, melyek a webbányászat módszereit mutatják be, s kis változtatással segíthetnek bármely portállal kapcsolatos kérdések megválaszolásában.

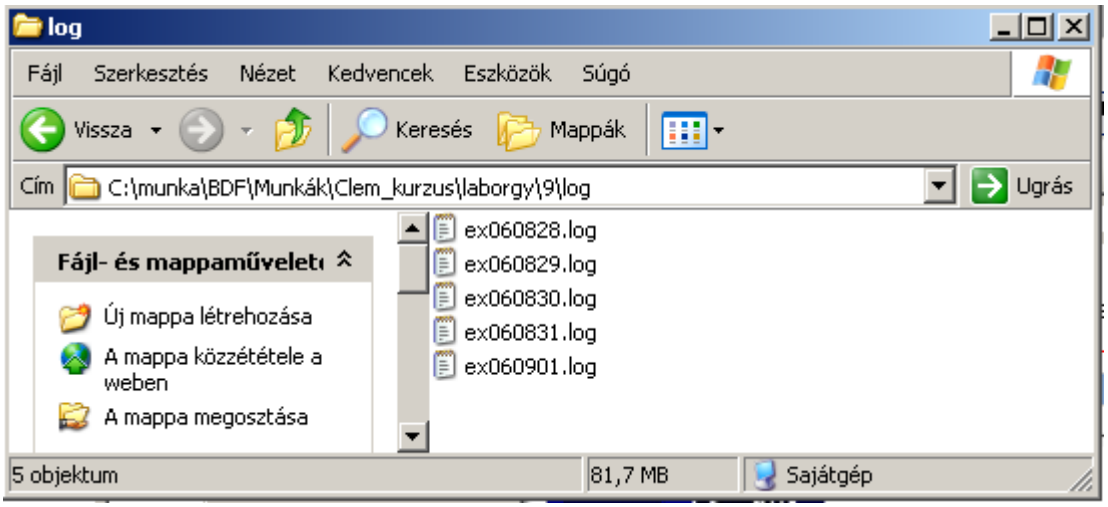
Log file-ok előkészítése

Bontsuk ki a laborgyakorlat fájljai között található tömörített log.zip fájlt, hogy a Web Mining node dolgozhasson belőle!

Ehhez a log.zip fájl jobbegér-kattintással elérhető gyorsmenüjéből *Az összes kibontása...* menüpontot kell választani. A megjelenő kibontási varázsló párbeszédpanelein csak Enter-eket kell ütni.



A végeredmény egy új *log* nevű mappa, mely öt nap eseményeit naplózó log fájlokat tartalmaz, 2006. augusztus 28-tól szeptember 1-ig:



A fájlok összmérete arra utal, hogy e szöveges állományok sok információt tartalmaznak. Ha Jegyzettömbbel nézzük meg tartalmukat, a megnyitás még 10 másodpercet is igénybe vehet, és mindegyik log fájlban százezres nagyságrendű lesz a sorok száma.

Eseménydefiníciók

Mielőtt összeállítanánk egy egyszerű „webbányász" streamet, elő kell készíteni a Web Mining node használatát. Meg kell mondani a web mining node-nak, hogy a naplófájlból milyen eseményeket szeretnénk figyelni, milyen sorokat (rekordokat) olvasson be a log fájlból.

A Web Mining node a log fájlból következtet, hogy mely kérések történtek ugyanabban az időben ugyanarról a gépről. Ennek alapján különféle felhasználókat (**USEREKET**) különböztet meg, illetve egy-egy user egy-egy alkalommal történő kéréseinek sorozatát látogatásnak (**VISITNEK**) tekinti.

A besorolások nem mindig tökéletesek. Például arról nincs információ, konkrétan ki ült a kérést küldő gép előtt, így egy gépről dolgozó több tényleges személyt is egy usernek tekinthet a Web Mining, továbbá ha az internetező felhasználó feláll a gépe mellől és visszatérve órák múlva kattint újra a portálon, akkor tevékenységét egy hosszú visitnek látjuk, holott erről valójában szó sincs.

A Web Mining node egy olyan adattáblát készít a log fájlokból, melynek a következő mezői vannak:

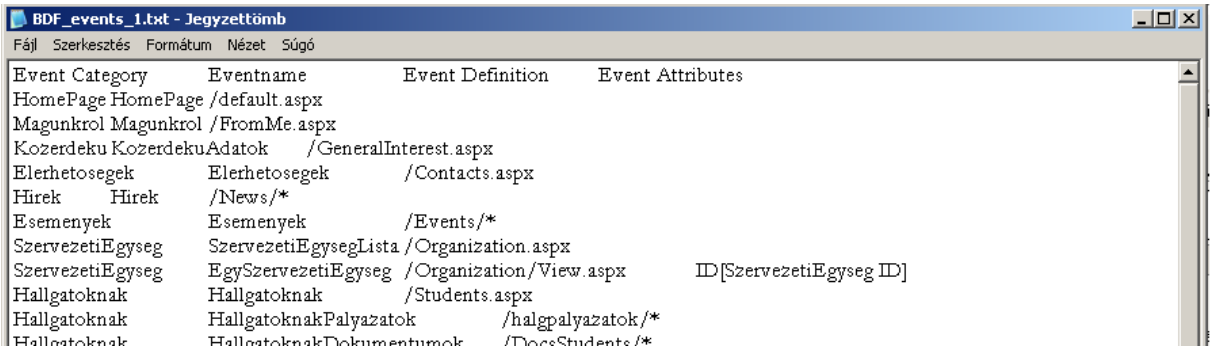
Mezőnév	*Jelentése*
Event ID	Az esemény egyedi sorszáma
Event Category	**Esemény kategóriája – szöveges értékét a felhasználó adja meg az eseménytáblázattal**
Event Name	**Esemény neve – szöveges értékét a felhasználó adja meg az eseménytáblázattal**
Resource	Az URL-nek az eseményre utaló töredéke
Event Timestamp	Esemény ideje
Visit ID	Az eseményt tartalmazó visit egyedi sorszáma

Mezőnév	*Jelentése*
Visit Start Timestamp	Az eseményt tartalmazó visit kezdetének időpontja
User ID	User egyedi sorszáma
User Type	User azonosításának módját jelző kódszám: 1 belső usernévvel azonosított, 2 cookie-val (sütivel) azonosított, 3 hostnévvel azonosított
Authorized User Name	User belső azonosítója, ha van
User Cookie	User-süti értéke, ha van
Hostname	User hostneve
Attribute ID	Ha van eseménnyel kapcsolatos attribútum, annak egyedi azonosítója
Attribute Name	Ha van eseménnyel kapcsolatos attribútum, annak neve
Attribute Value	Ha van eseménnyel kapcsolatos attribútum, annak értéke

A kiemelt két mező tartalmát a felhasználó adhatja meg egy úgynevezett eseménytáblázattal. Az eseménytáblázat szabályozza azt is, hogy az adattábla milyen rekordokat tartalmazzon: azok a sorok kerülnek be a log file-ból a Clementine adattáblába, amelyeknek e két mezője értéket kap.

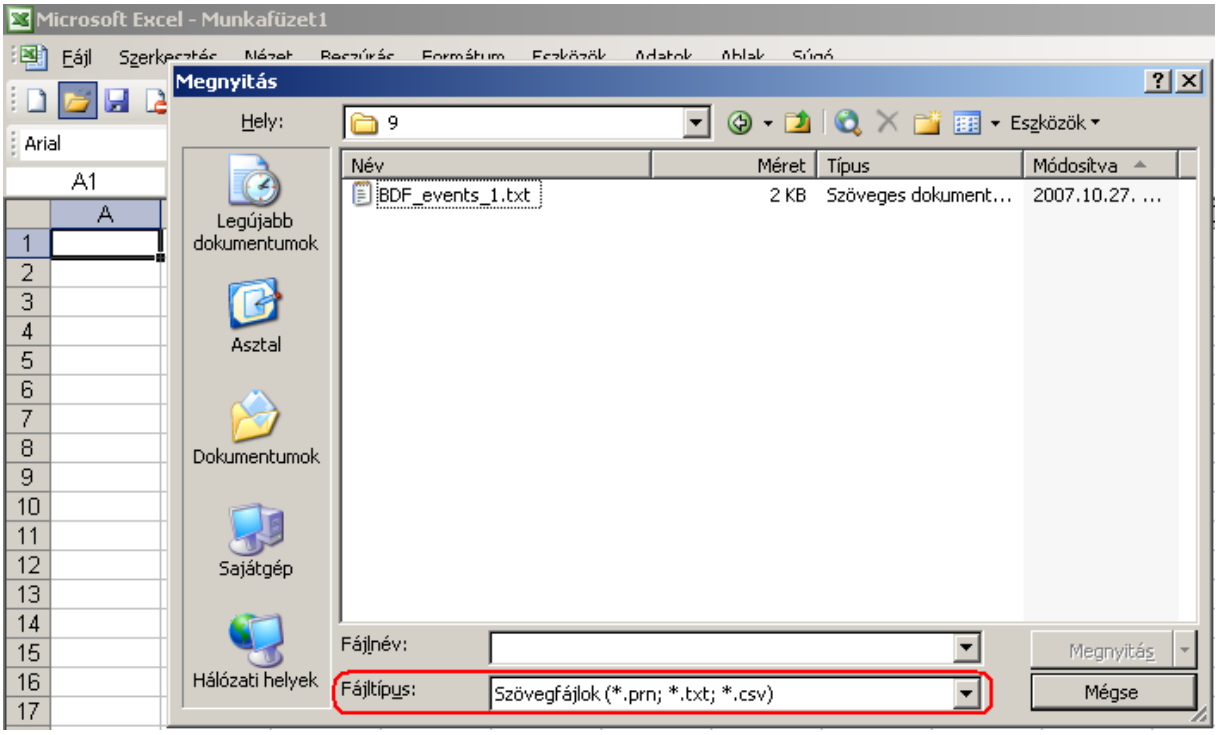
E gyakorlat segédletei között szerepel már egy ilyen eseménytábla, ez a *BDF_events_1.txt* fájl.

Ha (*Jegyzettömb*bel) megnyitjuk, ezt látjuk:

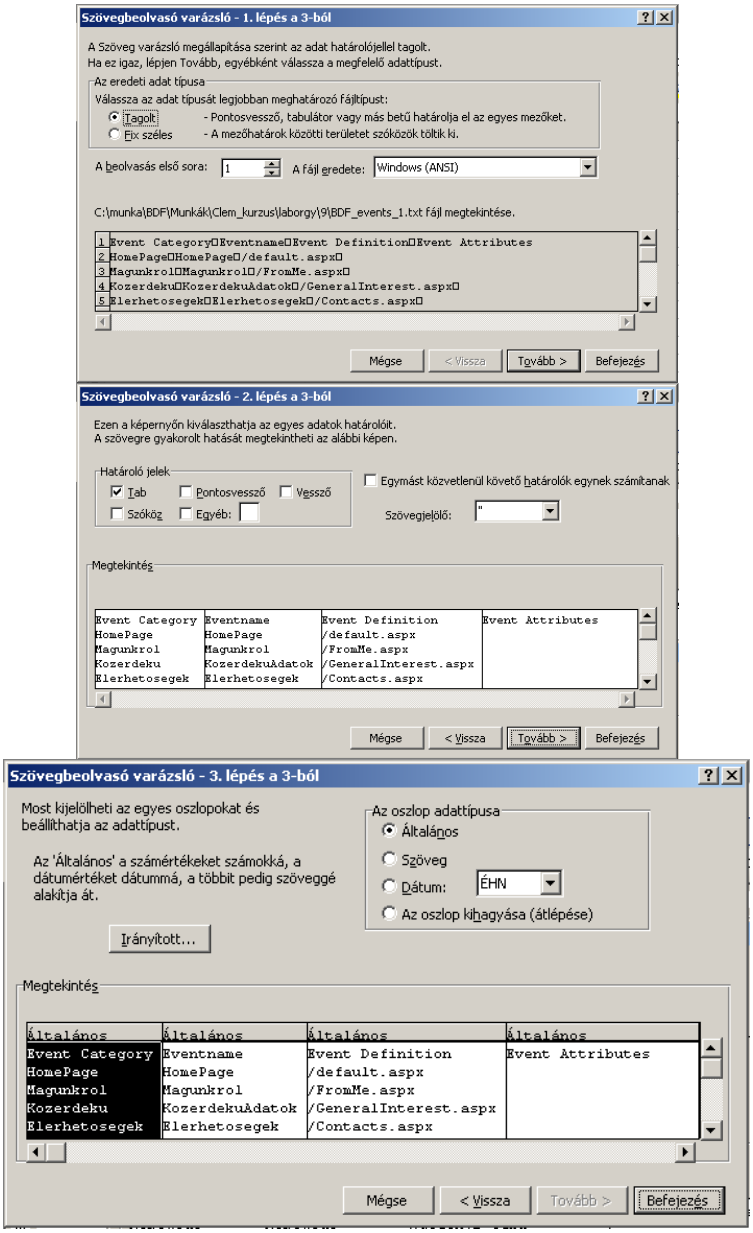


Mivel ez tabulátorokkal tagolt táblázat, ezért nem jól látszanak így az oszlopok.

Nyissuk meg a fájlt az Excel segítségével!



Ne felejtsük a Megnyitás párbeszédpanelen a fájltípust szövegfájlokra állítani! Megjelenik a szövegbeolvasó varázsló, üssünk Entert mindhárom lépésnél!



Most már áttekinthetően jelenik meg az eseménytáblázatunk, szélesítsük az oszlopokat, hogy minden adatot lássunk:

	A	B	C	D
1	Event Category	Eventname	Event Definition	Event Attributes
2	HomePage	HomePage	/default.aspx	
3	Magunkrol	Magunkrol	/FromMe.aspx	
4	Kozerdeku	KozerdekuAdatok	/GeneralInterest.aspx	
5	Elerhetosegek	Elerhetosegek	/Contacts.aspx	
6	Hirek	Hirek	/News/*	
7	Esemenyek	Esemenyek	/Events/*	
8	SzervezetiEgyseg	SzervezetiEgysegLista	/Organization.aspx	
9	SzervezetiEgyseg	EgySzervezetiEgyseg	/Organization/View.aspx	ID[SzervezetiEgyseg ID]
10	Hallgatoknak	Hallgatoknak	/Students.aspx	
11	Hallgatoknak	HallgatoknakPalyazatok	/halgpalyazatok/*	
12	Hallgatoknak	HallgatoknakDokumentumok	/DocsStudents/*	
13	Neptun	Neptun	/Neptun/*	
14	Karrier	Karrier	/szki/ki/*	
15	HOK	HOK	/HOK/*	
16	SzKI	SzKI	/szki/*	

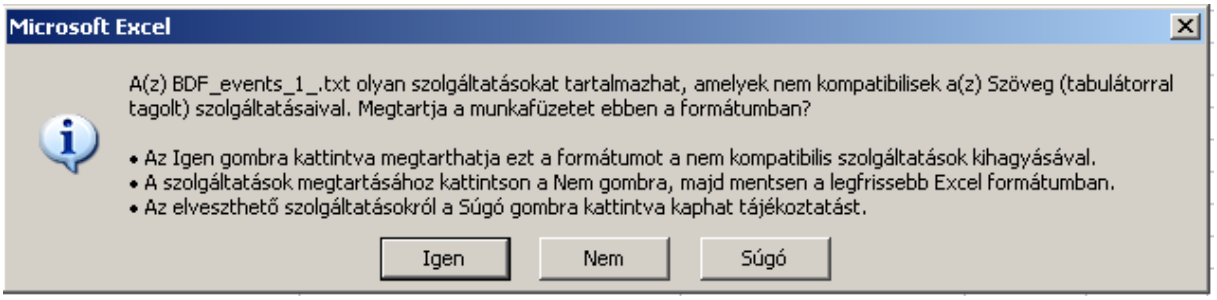
Az első sor megnevezéseket tartalmaz. Az *Event Category* és *Eventname* alatti cellákban tetszőleges elnevezéseket adhatunk meg – a log fájlból beolvasott sorokat (kéréseket) fogjuk így nevezni a Clementine által generált adattáblában. A harmadik és a negyedik oszlopba az kerül, hogy milyen URL-rész és milyen attribútum együttes előfordulása esetén kap az adott kérés valamely eseménykategóriát és eseménynevet a generált adattáblában. A log fájl minden egyes sorát a Clementine az eseménytáblázat soraival egymás után hasonlítja össze. Ha a log fájl vizsgált sorában álló kérés *Event Definition* és *Event Attributes* értéke megegyezik az eseménytáblázat egyik sorában megadott értékkel, akkor azt a logfájl-beli sort hozzáveszi a generált adattáblához, és e kérésre vonatkozóan nem nézi tovább az eseménytáblázatot (pedig elképzelhető, hogy még több feltételnek is megfelel), hanem továbblép a log fájl következő sorára (kérésére). Ha a log fájl valamely sora nem felel meg az eseménytáblázatban szereplő egyetlen *Event Definition + Event Attributes* együttes feltételnek sem, akkor a generált adattáblában ez a kérés nem fog rekordként szerepelni.

Egészítsük ki az eseménytáblázatot egy olyan sorral, mely a főiskolai Alkalmazott Informatika és Információmenedzsment Tanszék látogatói kéréseihez rendel eseményt! A tanszék honlapjának fájljaira mutató URL-rész: */ttk/mszi/informatika*, **a hozzárendelt rekordok eseménykategóriája legyen Tanszek, az esemény neve Itanszek!**

A most megismert logika alapján nem tehetjük az eseménydefiníció sorát sem a /ttk, sem a /ttk/mszi elé, hiszen akkor a tanszéki oldalakra mutató kérésekig nem fog eljutni a kiértékelés, csak TTK vagy MSZI nevű eseményként fogja beolvasni a Clementine. Ezért 35. sorként szúrjuk be az Excel vagy a Jegyzettömb listájába. A megadott URL-rész mögé írjunk egy /* -ot, ami az eseménytáblázatban azt jelenti, hogy az e szövegtöredéknél hosszabb URL-részletű kéréseket is ezen eseményhez rendelt rekordok közt szerepeltetjük.

31	Sulinet Tanfolyam	Sulinet Tanfolyam	/tnf/*
32	RegEleaAkad	RegEleaAkad	/ela/*
33	ITOK	ITOK	/itok/*
34	NemzetkoziKapcsolatok	NemzetkoziKapcsolatok	/nemzetkozikapcsolatok/*
35	Tanszek	Itanszek	/ttk/mszi/informatika/*
36	Intezet	MSZI	/ttk/mszi/*
37	Kar	BTK	/btk/*
38	Kar	TMK	/tmk/*
39	Kar	TTK	/ttk/*
40	Egyeb	Egyeb	/*
41			

Ha az Excelben végezzük a sor beszúrását, akkor a következő figyelmeztetésre (nem Excel formátumú mentés) még *Igen*-nel kell felelni, majd kilépni az Excelből.



Ha a Jegyzettömbben szúrtuk be az új esemény sorát az eseménytáblázatba, akkor arra kell figyelni, hogy az egy sorban álló elemeket tabulátorjelekkel kell elválasztani.

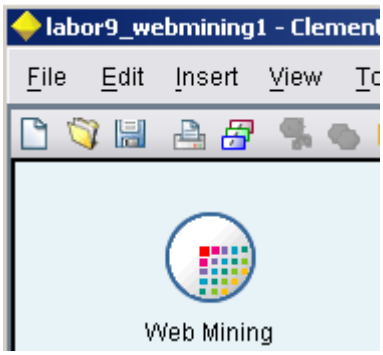
Az eseménytáblázatnak mindig tabulátorjelekkel elválasztott szöveges fájlnak kell lennie! Kiterjesztése elvileg bármi lehet, de javasolt a .txt kiterjesztéssel utalni arra, hogy egyszerű szöveget tartalmaz.

Figyeljük meg, hogy az Egyéb esemény révén minden portálhoz tartozó kérést beolvassunk!

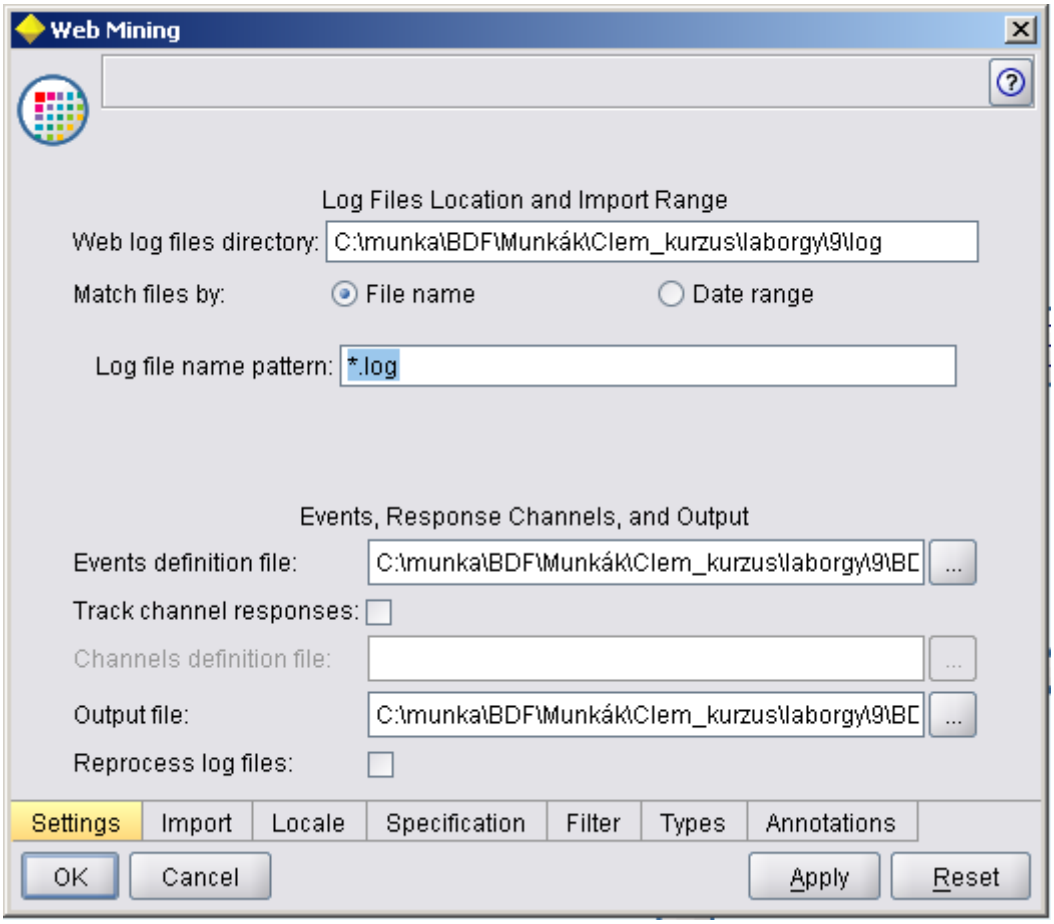
A Web Mining node használata

Egy üres streambe szúrjunk be egy Web Mining node-ot!

Mentsük a streamet *labor9_webmining.str* **néven!**



Állítsuk be a node tulajdonságait a Settings fülön az alábbiaknak megfelelően!



WEB LOG FILES DIRECTORY: Itt adhatjuk meg a feldolgozni kívánt log fájlok mappáját. Ha az eddigieknek megfelelően jártunk el, akkor ennek az elérési útnak a vége: *\9\log* , az eleje pedig annak a mappának az elérési útja, ahonnan a *9* nevű mappa nyílik. Sajnos itt nincs tallózási lehetőség, tehát be kell billentyűzni a *log* mappa elérési útját.

MATCH FILES BY: Itt azt állíthatjuk be, hogy fájlnév vagy dátum szerint olvasson be kéréseket a *log* fájlokból. Mivel példánkban a megadott öt nap log fájljai vannak csak meg, ezért célszerűbb a **FILE NAME** lehetőséget bekapcsolni.

LOG FILE NAME PATTERN: Itt egy fáljlmaszk adható meg, hogy milyen nevű és kiterjesztésű log fájlokat olvasson be a node. Esetünkben a **.log* maszk utal arra, hogy bármilyen nevű, *.log* kiterjesztésű fájlokat szeretnénk beolvasni a megadott mappából.

EVENTS DEFINITION FILE: Itt kell megadni az eseménytáblát, tallózással megkereshetjük vagy bebillentyűzhetjük a már tárgyalt *BDF_events_1.txt* elérési útját.

OUTPUT FILE: Itt egy olyan output (kimeneti) fájl elérési útja adható meg, amelyet minden beolvasás után újra létrehoz a Clementine. E fájl a node kimeneteként keletkező adattáblát tartalmazza. Eredetileg a párbeszédpanel a *$WEBMINING_ROOT/output/webminingoutput.txt* nevet kínálja fel, állítsuk át ezt a *...\9\BDF_events_out.txt* elérési útra, hogy közvetlenül megnézhessük a node kimenetét és szükség esetén felhasználhassuk. (A ... három pont annak a mappának az elérési útját jelzi, ahonnan a *9* nevű mappa nyílik.)

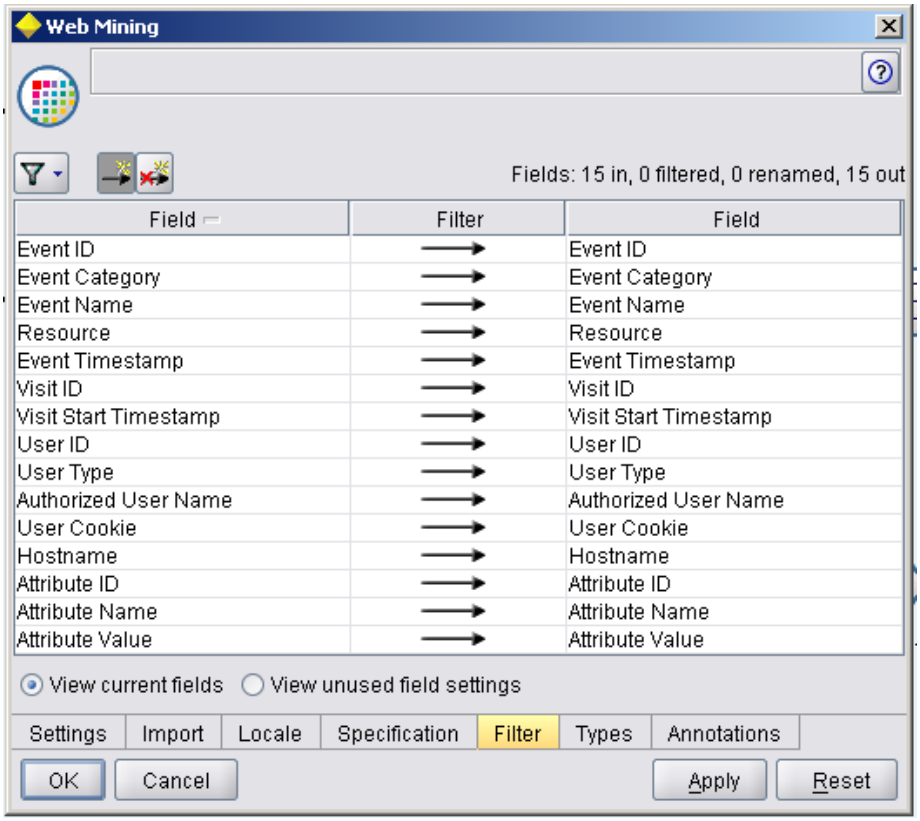
REPROCESS LOG FILES: Itt azt állíthatjuk be, hogy a node változatlan újrafuttatása esetén újra feldolgozza-e a log fájlok adatait vagy az egyszer már létrehozott output fájlt beolvasva képezzen adattáblát a node.

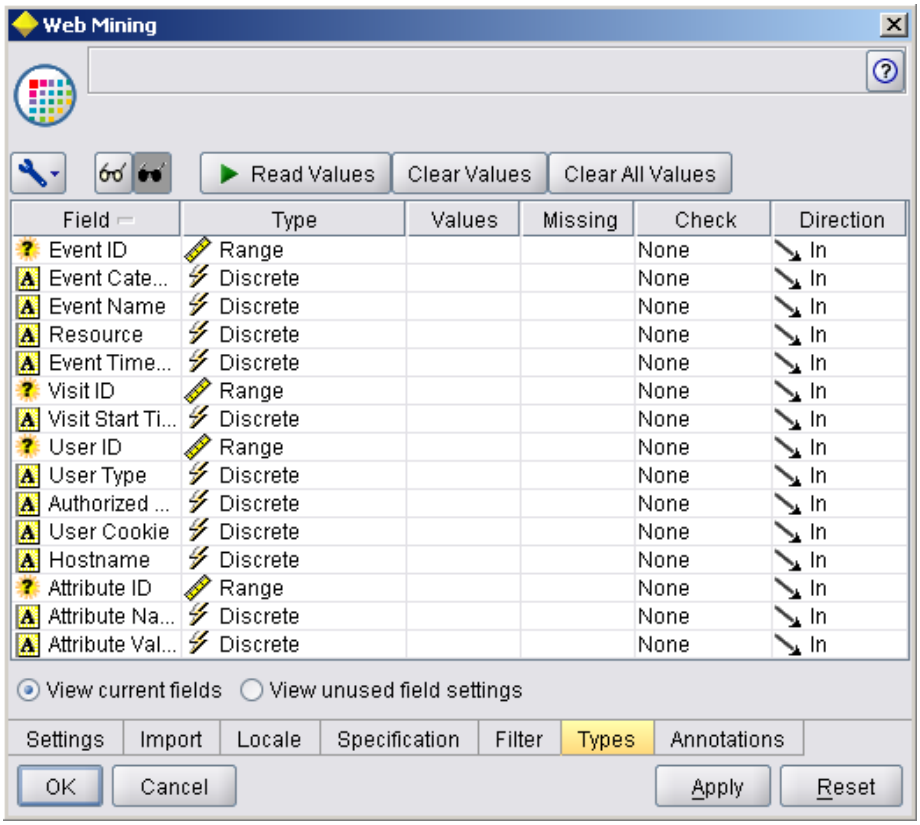
Ha egy adatbányász stream-et építünk, kiegészítünk, tesztelünk, vagy valamilyen okból többször futtatunk, célszerű a ***REPROCESS LOG FILES*** *jelölőnégyzetből a pipát kivenni, mert a log fájlok beolvasása aránylag hosszú ideig tarthat (hosszabb időszak, vagy nagy forgalmú portál vizsgálata esetén akár órákig is). Az output fájl beolvasása ugyanazt az eredményt szolgáltatja aránylag rövid idő alatt.*

A **CHANNELS** beállítások és a beolvasást irányító **IMPORT, LOCALE, SPECIFICATION** fülek megismerése túlmutat e kurzus keretein.

Vegyük ki a **TRACK CHANNEL RESPONSES** jelölőnégyzetből a pipát, mert nem adunk meg több adatot.

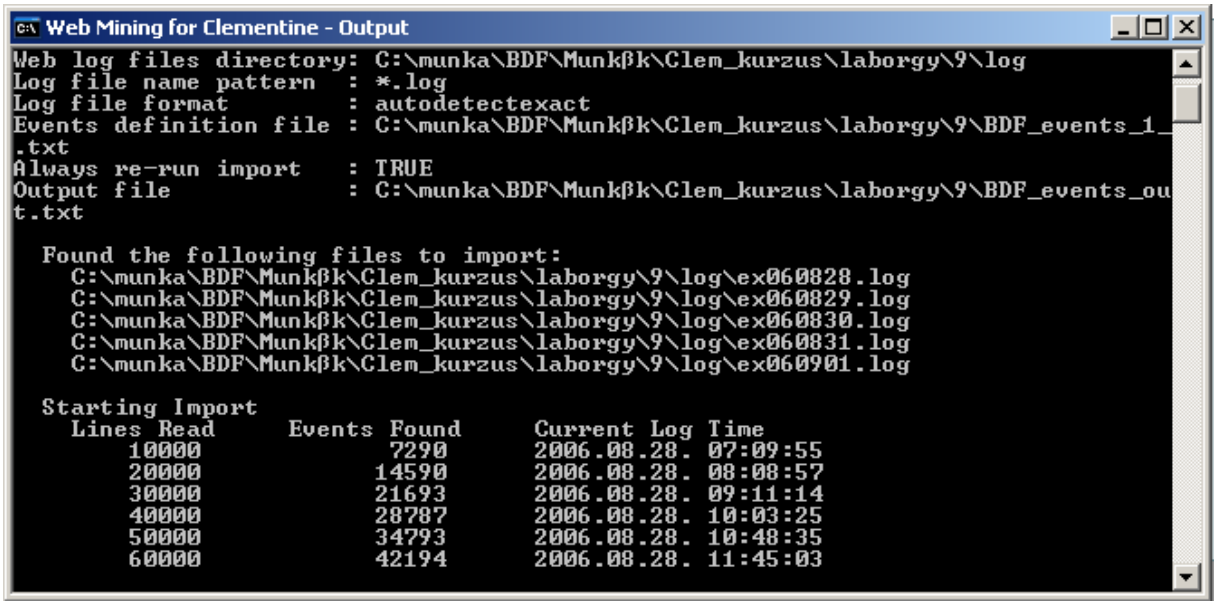
Figyeljük meg a Filter és a Types fülek tartalmát!





Azokat a mezőneveket olvashatjuk itt, amelyeket az Eseménydefiníciók fejezetben ismertettünk. Látható, hogy az ID mezők tárolási típusa ismeretlen, a többi mező tárolási típusa szöveges – holott e mezők közt két Timestamp (dátumot és időpontot együtt tartalmazó „időpecsét") is szerepel – ezeket át kell majd alakítani, hogy használhassuk.

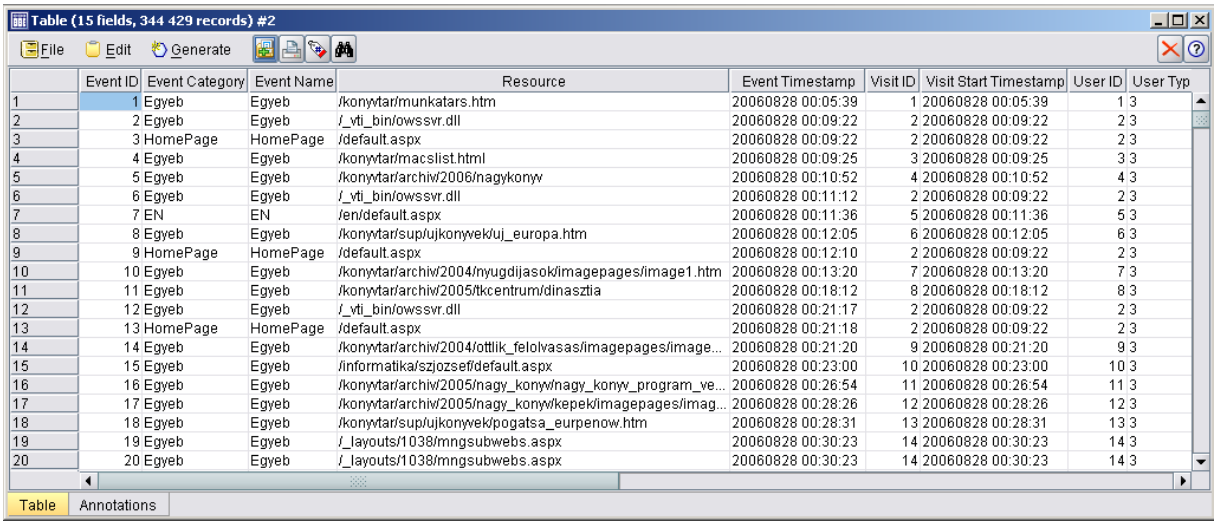
Szúrjunk be egy táblát a Web mining node után és futtassuk!



A futás kezdetén egy karakteres ablakban tájékoztat a gép a log fájlok beolvasásának állapotáról.

Az első futtatás teljes ideje a táblázat megjelenítéséig kb. egy perc. Mivel kikapcsoltuk a Reprocess log files kapcsolót, ezért a második futtatásnál már a kész *BDF_events_out.txt* -ből dolgozik a node, így ezek után már rövidebb lesz a futási idő (kb. 10 másodperc).

Futtassuk még egyszer a táblát, s figyeljük meg, hogy másodszor mennyivel rövidebb ideig dolgozik a gép!

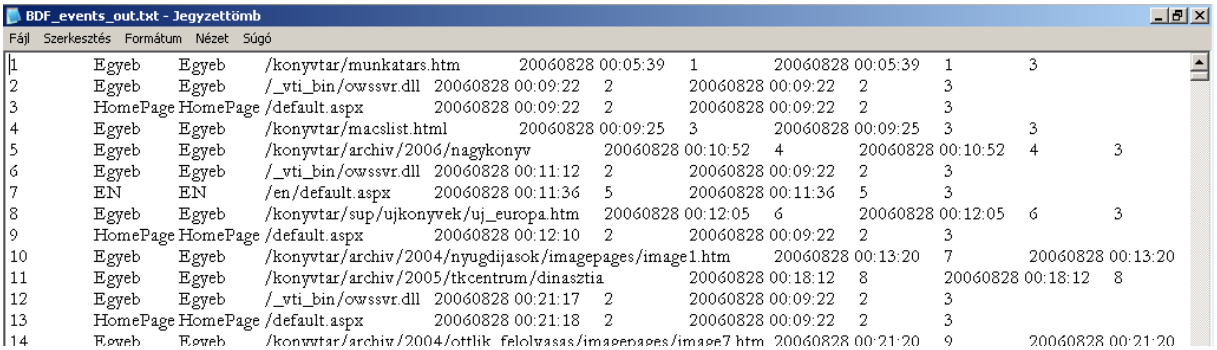


	Event ID	Event Category	Event Name	Resource	Event Timestamp	Visit ID	Visit Start Timestamp	User ID	User Typ
1	1	Egyeb	Egyeb	/konyvtar/munkatars.htm	20060828 00:05:39	1	20060828 00:05:39	1	3
2	2	Egyeb	Egyeb	/_vti_bin/owssvr.dll	20060828 00:09:22	2	20060828 00:09:22	2	3
3	3	HomePage	HomePage	/default.aspx	20060828 00:09:22	2	20060828 00:09:22	2	3
4	4	Egyeb	Egyeb	/konyvtar/macslist.html	20060828 00:09:25	3	20060828 00:09:25	3	3
5	5	Egyeb	Egyeb	/konyvtar/archiv/2006/nagykonyv	20060828 00:10:52	4	20060828 00:10:52	4	3
6	6	Egyeb	Egyeb	/_vti_bin/owssvr.dll	20060828 00:11:12	2	20060828 00:09:22	2	3
7	7	EN	EN	/en/default.aspx	20060828 00:11:36	5	20060828 00:11:36	5	3
8	8	Egyeb	Egyeb	/konyvtar/sup/ujkonyvek/uj_europa.htm	20060828 00:12:05	6	20060828 00:12:05	6	3
9	9	HomePage	HomePage	/default.aspx	20060828 00:12:10	2	20060828 00:09:22	2	3
10	10	Egyeb	Egyeb	/konyvtar/archiv/2004/nyugdijasok/imagepages/image1.htm	20060828 00:13:20	7	20060828 00:13:20	7	3
11	11	Egyeb	Egyeb	/konyvtar/archiv/2005/tkcentrum/dinasztia	20060828 00:18:12	8	20060828 00:18:12	8	3
12	12	Egyeb	Egyeb	/_vti_bin/owssvr.dll	20060828 00:21:17	2	20060828 00:09:22	2	3
13	13	HomePage	HomePage	/default.aspx	20060828 00:21:18	2	20060828 00:09:22	2	3
14	14	Egyeb	Egyeb	/konyvtar/archiv/2004/ottlik_felolvasas/imagepages/image...	20060828 00:21:20	9	20060828 00:21:20	9	3
15	15	Egyeb	Egyeb	/informatika/szjozsef/default.aspx	20060828 00:23:00	10	20060828 00:23:00	10	3
16	16	Egyeb	Egyeb	/konyvtar/archiv/2005/nagy_konyv/nagy_konyv_program_ve...	20060828 00:26:54	11	20060828 00:26:54	11	3
17	17	Egyeb	Egyeb	/konyvtar/archiv/2005/nagy_konyv/kepek/imagepages/imag...	20060828 00:28:26	12	20060828 00:28:26	12	3
18	18	Egyeb	Egyeb	/konyvtar/sup/ujkonyvek/pogatsa_eurpenow.htm	20060828 00:28:31	13	20060828 00:28:31	13	3
19	19	Egyeb	Egyeb	/_layouts/1038/mngsubwebs.aspx	20060828 00:30:23	14	20060828 00:30:23	14	3
20	20	Egyeb	Egyeb	/_layouts/1038/mngsubwebs.aspx	20060828 00:30:23	14	20060828 00:30:23	14	3

A végeredményül kapott táblában több mint háromszázezer rekord van, mindegyik egy-egy olyan felhasználói kérés, amit az eseménytáblázatban definiáltunk. Ezentúl ezeket a rekordokat többnyire *esemény*nek fogjuk nevezni – durván azt lehetne mondani, ezek egy-egy portálrészre irányuló kattintásokat jeleznek.

Nézzük meg, mit tartalmaz a BDF_events_out.txt fájl!

Látható, hogy ugyanezt a Web mining node futtatásával kapott táblát tartalmazza mezőnevek nélkül, tabulátorokkal tagolt szövegfájlként.




```
1   Egyeb     Egyeb     /konyvtar/munkatars.htm                    20060828 00:05:39  1  20060828 00:05:39  1  3
2   Egyeb     Egyeb     /_vti_bin/owssvr.dll   20060828 00:09:22   2  20060828 00:09:22  2  3
3   HomePage  HomePage  /default.aspx          20060828 00:09:22   2  20060828 00:09:22  2  3
4   Egyeb     Egyeb     /konyvtar/macslist.html    20060828 00:09:25   3  20060828 00:09:25  3  3
5   Egyeb     Egyeb     /konyvtar/archiv/2006/nagykonyv            20060828 00:10:52  4  20060828 00:10:52  4  3
6   Egyeb     Egyeb     /_vti_bin/owssvr.dll   20060828 00:11:12   2  20060828 00:09:22  2  3
7   EN        EN        /en/default.aspx       20060828 00:11:36   5  20060828 00:11:36  5  3
8   Egyeb     Egyeb     /konyvtar/sup/ujkonyvek/uj_europa.htm      20060828 00:12:05  6  20060828 00:12:05  6  3
9   HomePage  HomePage  /default.aspx          20060828 00:12:10   2  20060828 00:09:22  2  3
10  Egyeb     Egyeb     /konyvtar/archiv/2004/nyugdijasok/imagepages/image1.htm    20060828 00:13:20  7  20060828 00:13:20
11  Egyeb     Egyeb     /konyvtar/archiv/2005/tkcentrum/dinasztia   20060828 00:18:12  8  20060828 00:18:12  8
12  Egyeb     Egyeb     /_vti_bin/owssvr.dll   20060828 00:21:17   2  20060828 00:09:22  2  3
13  HomePage  HomePage  /default.aspx          20060828 00:21:18   2  20060828 00:09:22  2  3
14  Egyeb     Egyeb     /konyvtar/archiv/2004/ottlik_felolvasas/imagepages/image7.htm  20060828 00:21:20  9  20060828 00:21:20
```

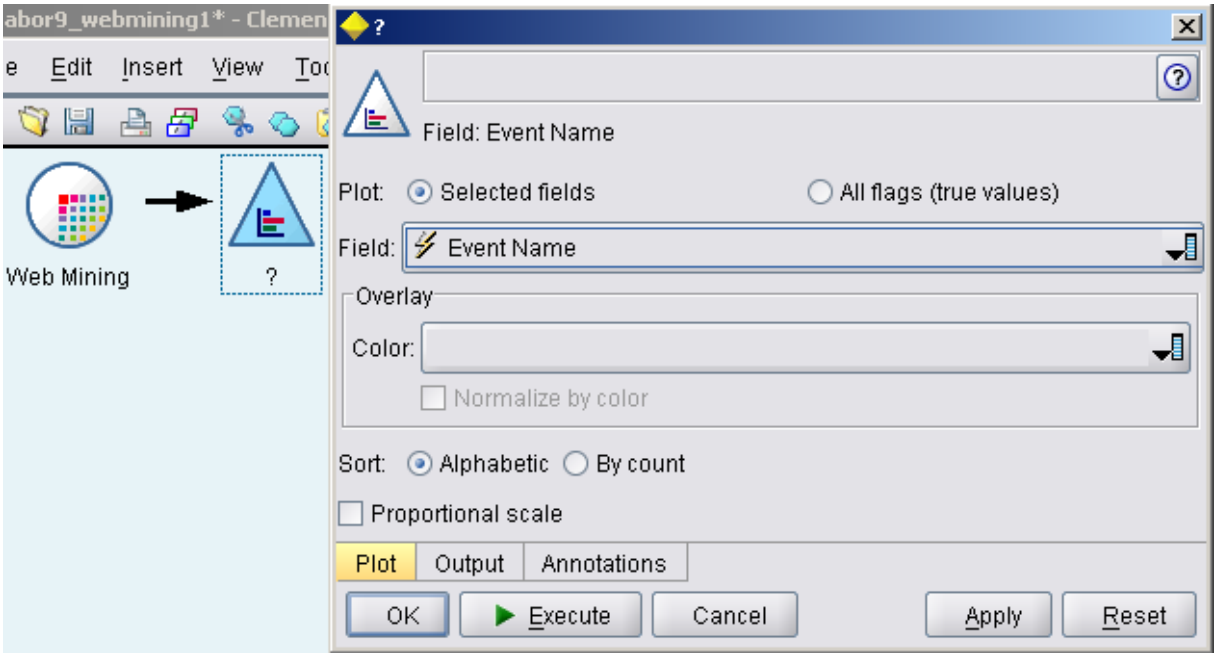
Event statisztikák

A megfigyelt események gyakoriságát, az egyes honlaprészeken történt kérések – leegyszerűsítve: az oda irányuló kattintások – számát, megoszlását szeretnénk megfigyelni. Külön össze szeretnénk hasonlítani az egyes karokhoz kapcsolódó ilyen események számát.

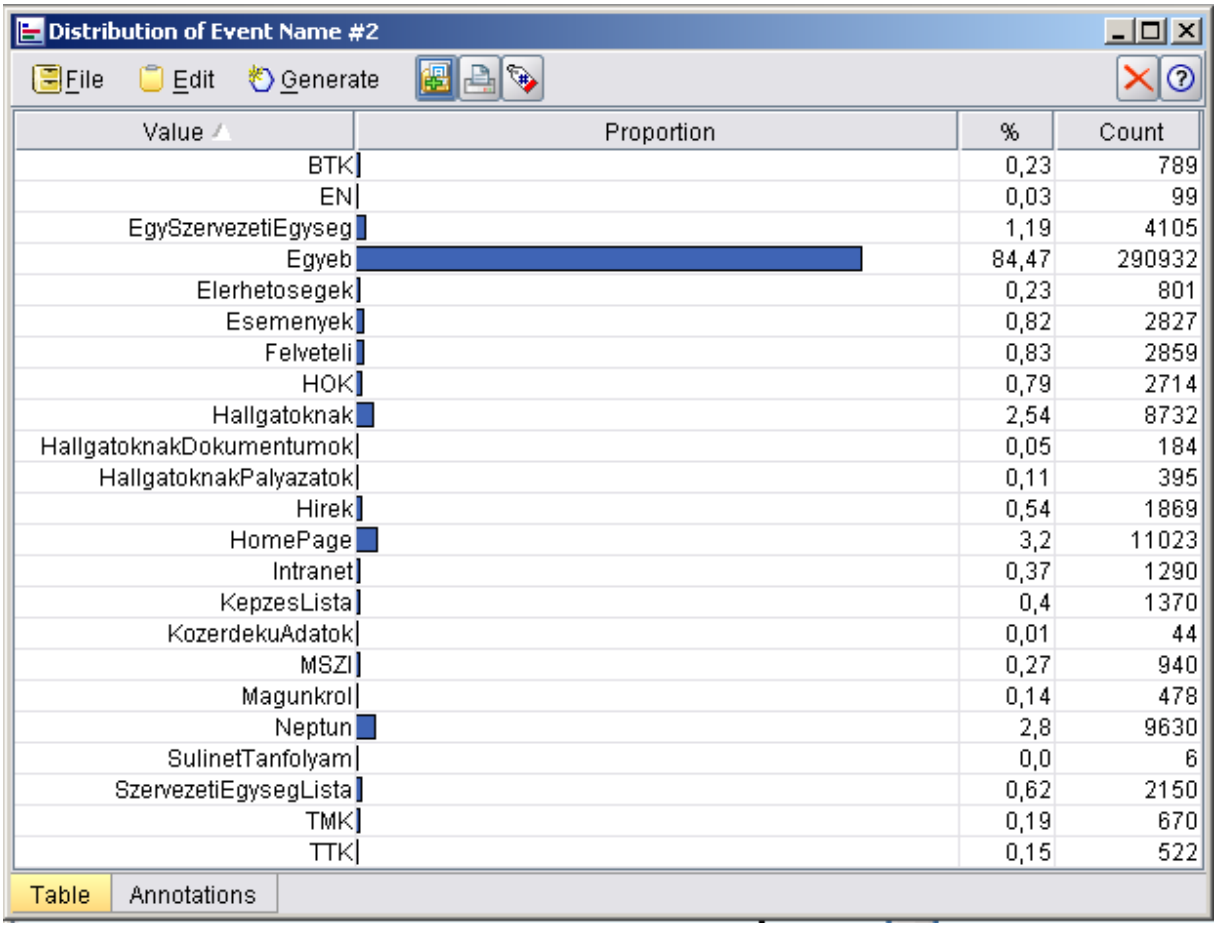
Először egy átfogó eseménystatisztikát készítünk minden megfigyelt eseményről.

Rajzoltassuk ki a Clementine-nal a definiált események eloszlását!

Célszerű az imént betett táblát kitörölni az eddigi streamből és helyette egy *Distribution (graph) node*-ot betenni:



Ha beállítjuk a grafikon paneljén az *Event Name* mezőt és futtatjuk a grafikont, ezt kapjuk:

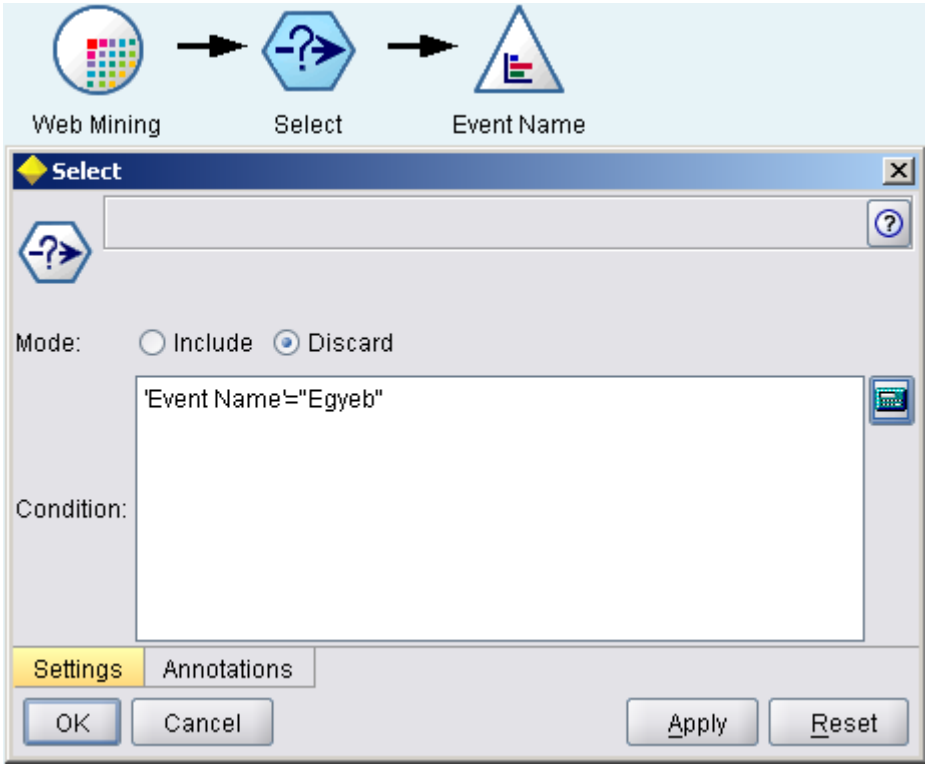


Value	Proportion	%	Count
BTK		0,23	789
EN		0,03	99
EgySzervezetiEgyseg		1,19	4105
Egyeb		84,47	290932
Elerhetosegek		0,23	801
Esemenyek		0,82	2827
Felveteli		0,83	2859
HOK		0,79	2714
Hallgatoknak		2,54	8732
HallgatoknakDokumentumok		0,05	184
HallgatoknakPalyazatok		0,11	395
Hirek		0,54	1869
HomePage		3,2	11023
Intranet		0,37	1290
KepzesLista		0,4	1370
KozerdekuAdatok		0,01	44
MSZI		0,27	940
Magunkrol		0,14	478
Neptun		2,8	9630
SulinetTanfolyam		0,0	6
SzervezetiEgysegLista		0,62	2150
TMK		0,19	670
TTK		0,15	522

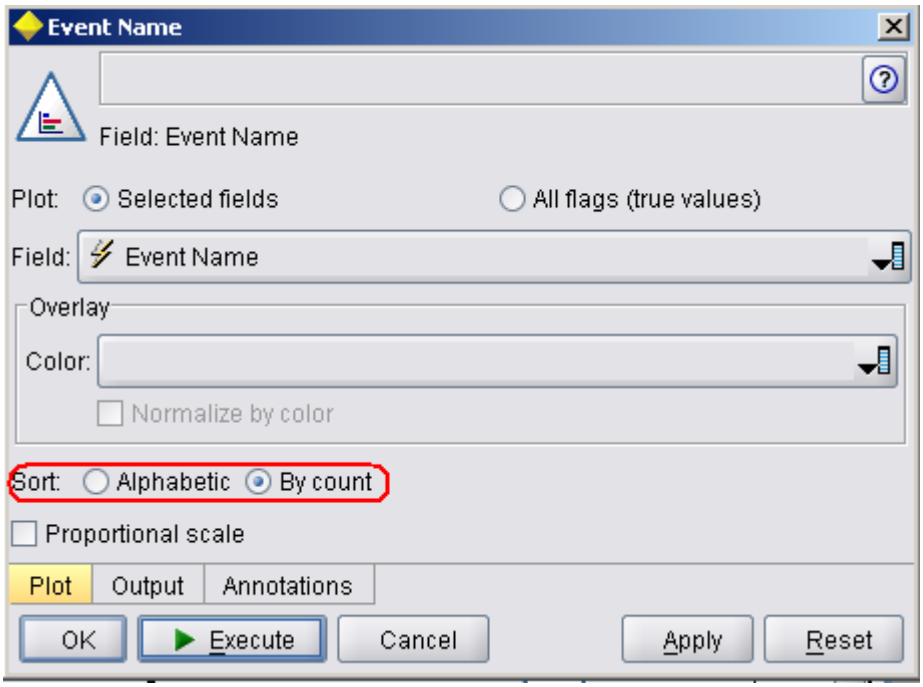
Megfigyelhetjük, hogy az „egyéb” események nagy száma miatt nem jól kivehető a többi események aránya.

Készítsük el ugyanezt a megoszlási grafikont az „Egyéb” események kihagyásával! Állítsunk be az események száma szerinti csökkenő sorrendet is!

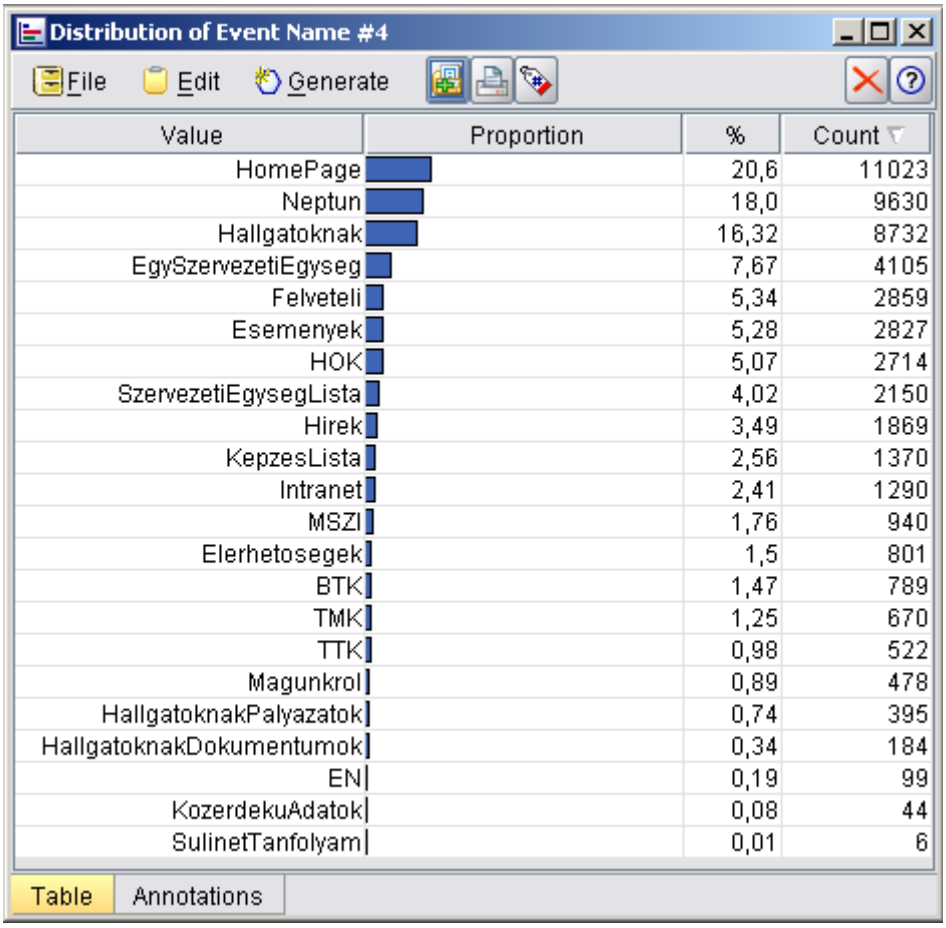
Az első feladatrészhez a Select node fog segíteni, melyet a Web Mining node és a Distribution Node közé kell beilleszteni és az alábbi beállításokat kell rajta tenni:



A második feladatrész megvalósításához a Distribution Node-on kell bekapcsolni a **SORT – BY COUNT** beállítást:



A futtatás után pedig ezt látjuk:



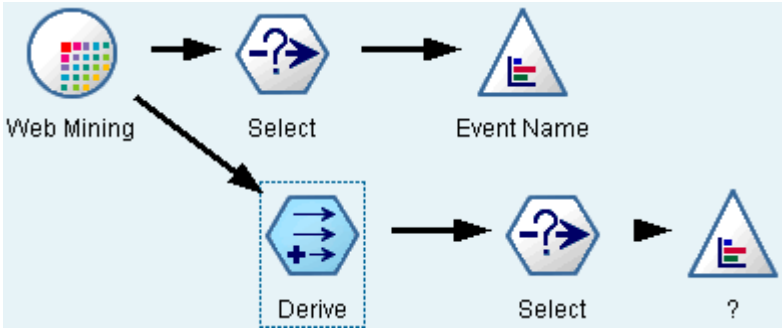
Value	Proportion	%	Count
HomePage		20,6	11023
Neptun		18,0	9630
Hallgatoknak		16,32	8732
EgySzervezetiEgyseg		7,67	4105
Felveteli		5,34	2859
Esemenyek		5,28	2827
HOK		5,07	2714
SzervezetiEgysegLista		4,02	2150
Hirek		3,49	1869
KepzesLista		2,56	1370
Intranet		2,41	1290
MSZI		1,76	940
Elerhetosegek		1,5	801
BTK		1,47	789
TMK		1,25	670
TTK		0,98	522
Magunkrol		0,89	478
HallgatoknakPalyazatok		0,74	395
HallgatoknakDokumentumok		0,34	184
EN		0,19	99
KozerdekuAdatok		0,08	44
SulinetTanfolyam		0,01	6

Úgy tűnik, hogy a fejezet elején felvetett másik problémára is megoldást kaptunk, mert a BTK, TMK és TTK karok sorrendje leolvasható. Ez azonban nem jó! Hiszen ha az eseménydefiníciót megnézzük, akkor látható, hogy az *MSZI* és az *Itanszek* esemény is a */ttk* URL-részhez tartozik, tehát ezeket is hozzá kell venni a TTK eseményeihez. A log file minden sorát csak egyszer sorolja be eseményként a Web Mining node, ezért pl. az MSZI eseményeit nem tekinti automatikusan TTK eseménynek is. Ezt korrigálni kell.

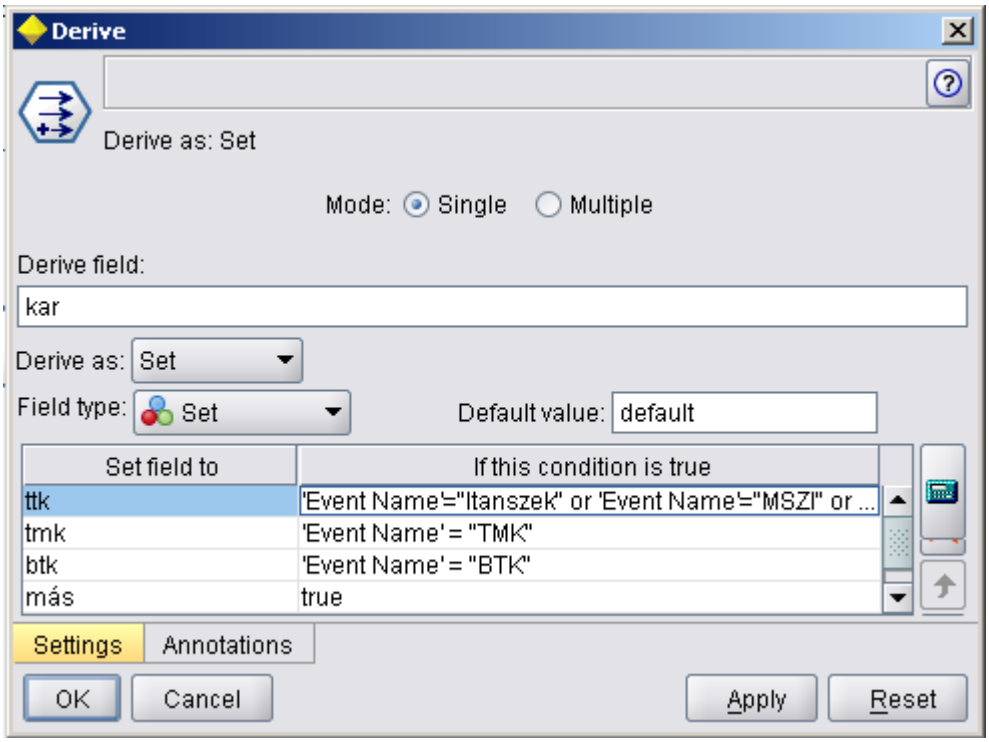
E célból egy új mezőt kell a táblához adni, melyben az egyes karok portáljain belüli eseményeket megjelöljük.

Adjunk egy *Derive node*-dal egy *kar* mezőt az adattáblához, melyben a „*ttk*”, „*btk*”, „*mszi*”, „*más*” értékek vannak attól függően, hogy a három kari portál közül melyikhez (vagy egyikhez sem) tartozik az esemény!

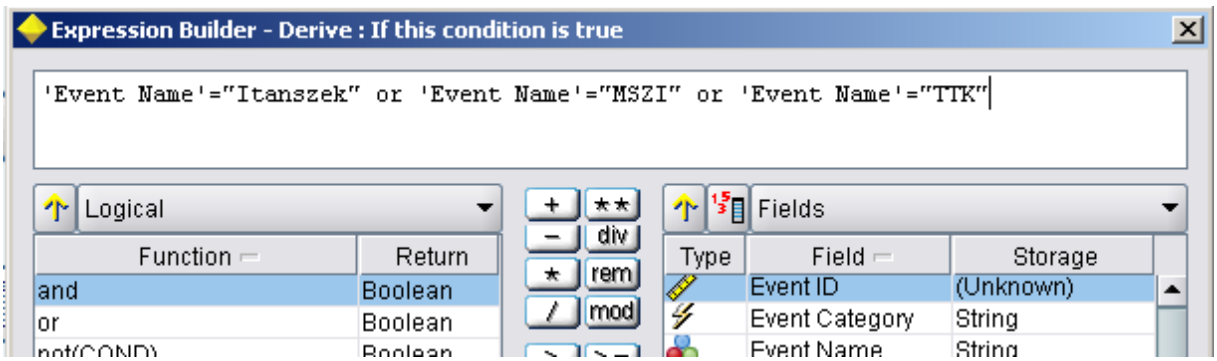
Közvetlenül a Source node után célszerű egy Derive, egy Select és egy Distribution node-ot beilleszteni.



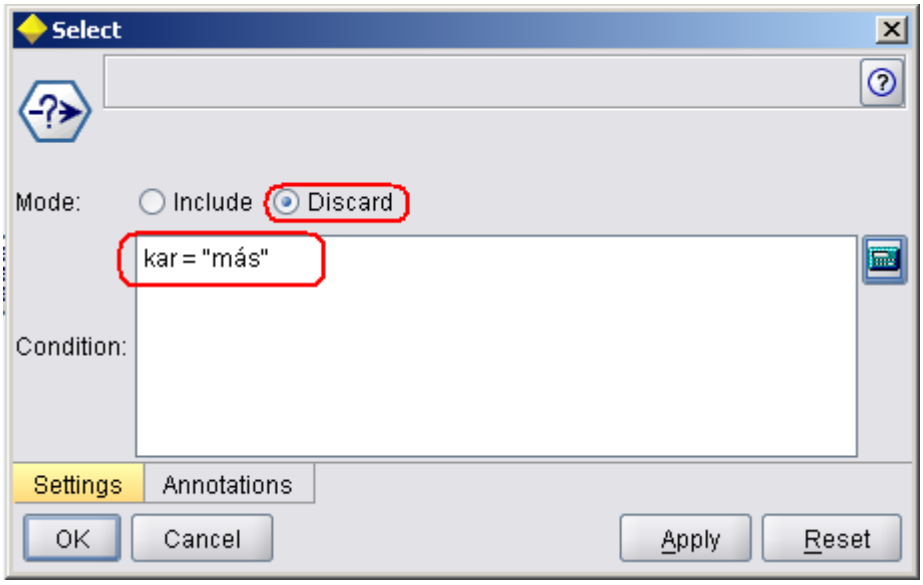
A *Derive node*-nál az új mező neve legyen *kar*, **SET** legyen a típusa és a *„btk”*, a *„tmk”* és a *„más”* adatérték feltételét egyszerűen az alábbiak szerint adjuk meg:



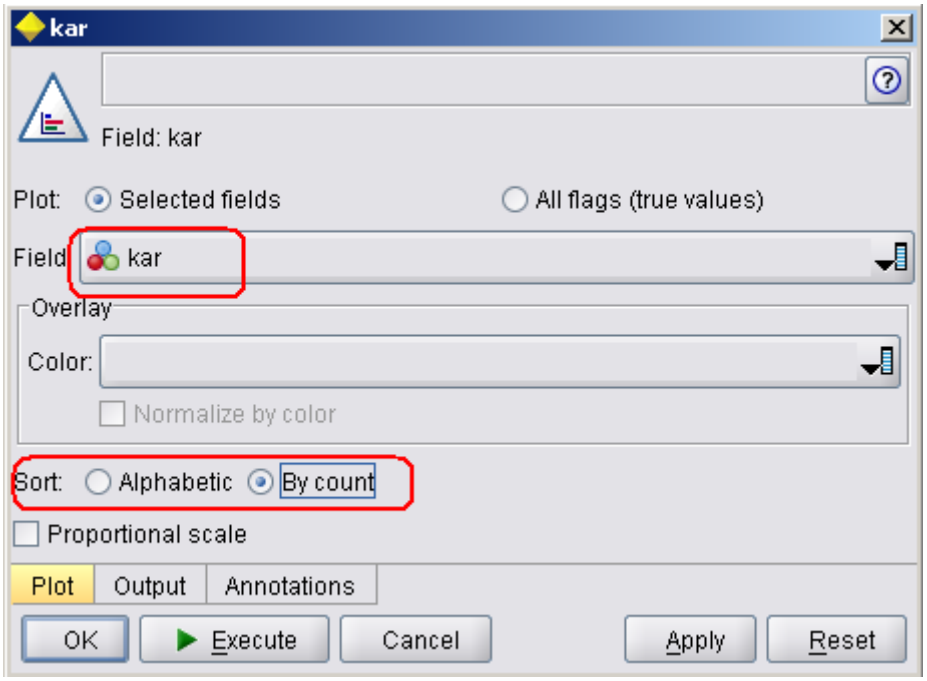
A *„ttk”* adatérték feltételének beviteléhez hívjuk be a kifejezésszerkesztőt és a következő kifejezést vigyük be:



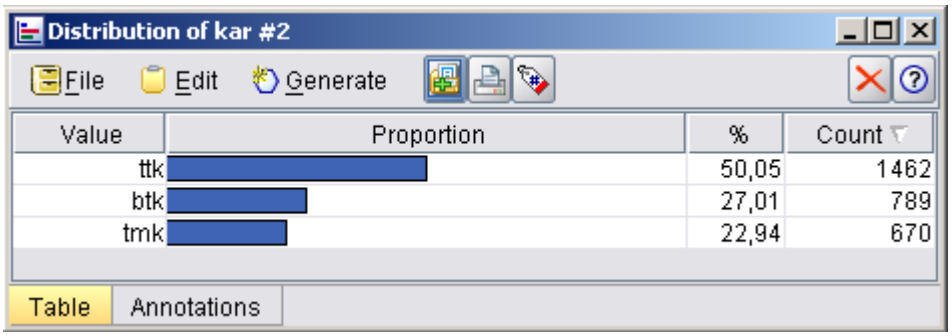
A Select node beállításai:



Végül a Distribution Node beállításai:



A futtatás eredménye:



Visit statisztikák

A visitek leegyszerűsítve azt a kattintássorozatot (egymás utáni kéréseket) jelentik, ami egyetlen user folyamatos tevékenységét jelzi a portálon való „tartózkodása” alatt. A *Visit ID* mező alapján lehet azonosítani az egyes visiteket. Ez sokféle vizsgálatra ad lehetőséget: meg lehet állapítani a visitek kezdetét jelentő eseményt, a végüket jelentő eseményt, az időtartamukat, a bennük történt események számát, azt, hogy hányféle és milyen nevű események történtek bennük, stb.

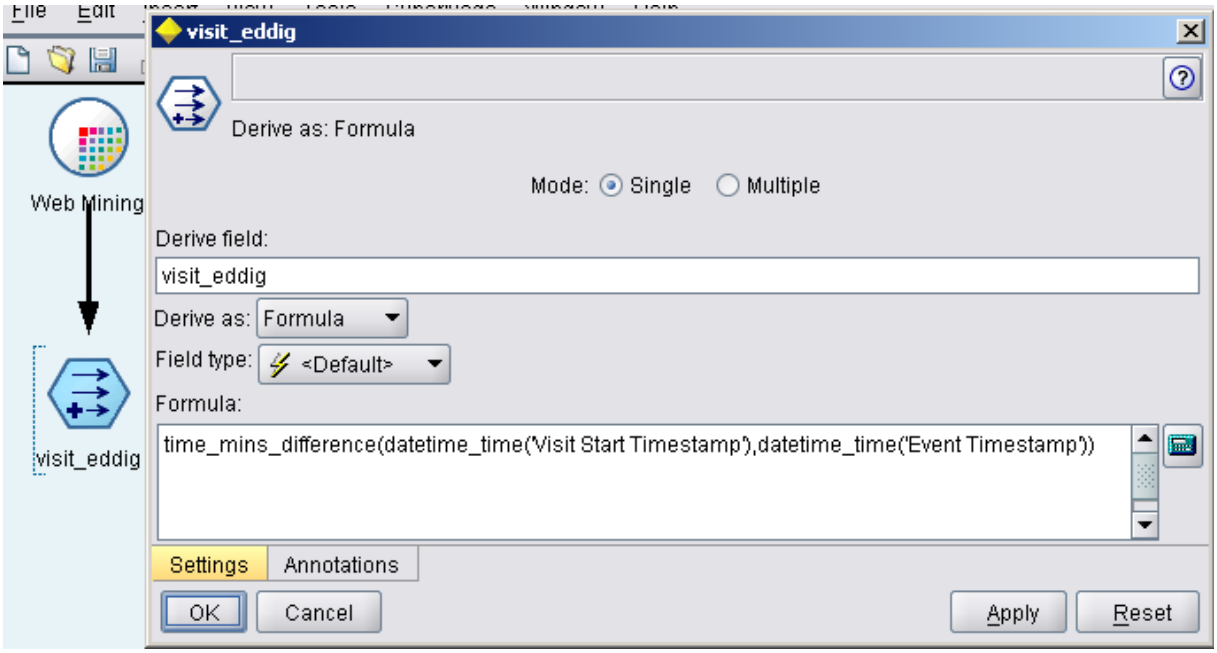
E fejezetben célként tűzzük ki, hogy az öt nap alatti visitek számát, átlagos időtartamát, és a bennük történt kattintások átlagos számát megállapítsuk.

Alakítsuk át az adattáblát úgy, hogy minden visithez egy rekord tartozzon és benne a visit azonosítója, percben mért időtartama, továbbá a hozzá tartozó események darabszáma jelenjen meg!

A feladatot három node-dal oldhatjuk meg.

Ehhez először egy olyan mezőt kell létrehozni, melyben az eseményekhez a visit kezdetétől addig eltelt idő van tárolva. Két „időpecsét” adott minden eseményhez, az „ő visitjének” a kezdete és a saját időpontja. Ezek különbségét kellene egy új mezőben képezni.

Szúrjunk be a *Web Mining node* után egy *Derive node*-ot és állítsuk be az alábbiak szerint:



Az új mező tartalmánál a **DATETIME_TIME()** és a **TIME_MINS_DIFFERENCE()** függvényeket használjuk. Az első a Timestamp formátumból csak az órát, percet és másodpercet tartja meg, a második pedig két időpont közti különbséget adja meg percben.

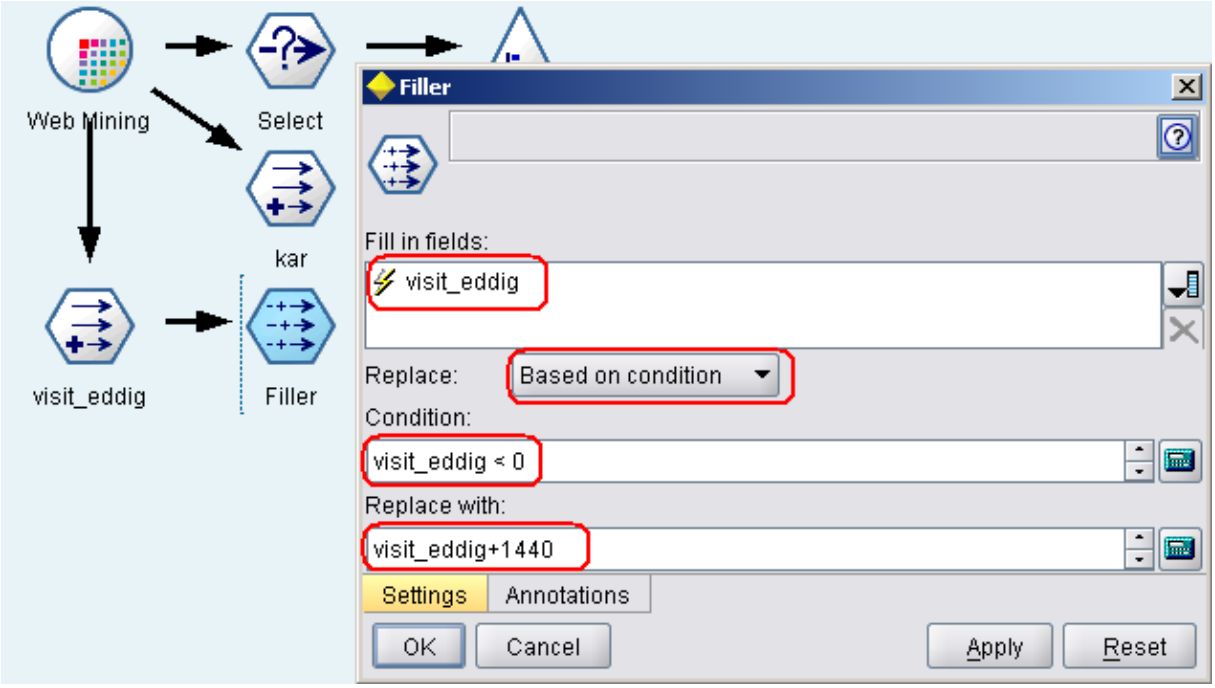
Tehát a beírandó kifejezés:

time_mins_difference(datetime_time('Visit Start Timestamp'),datetime_time('Event Timestamp'))

Sajnos keletkezhetnek negatív értékek is, ha éjfél előtt kezdődik egy user visitje és éjfél után fejeződik be.

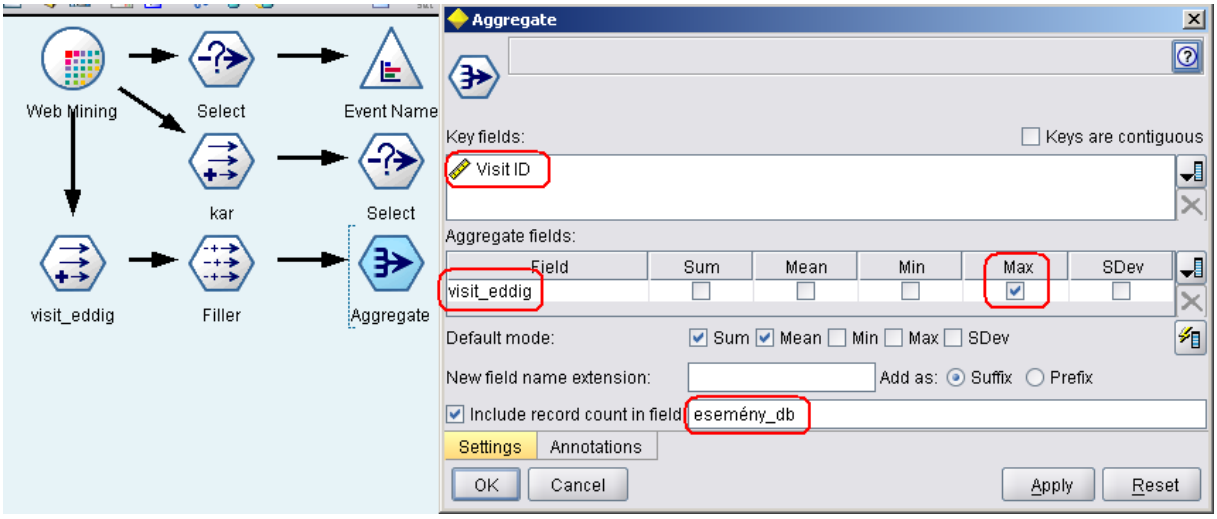
A kapott negatív értékekhez adjuk hozzá az egy napban levő percek számát, hogy a helyes időeltérést adják eredményül!

Egy napban 24*60=1440 perc van. Egy *Filler node*-dal a *<0* feltételhez kötve kell 1440-nel növelni a *visit_eddig* mező tartalmát:



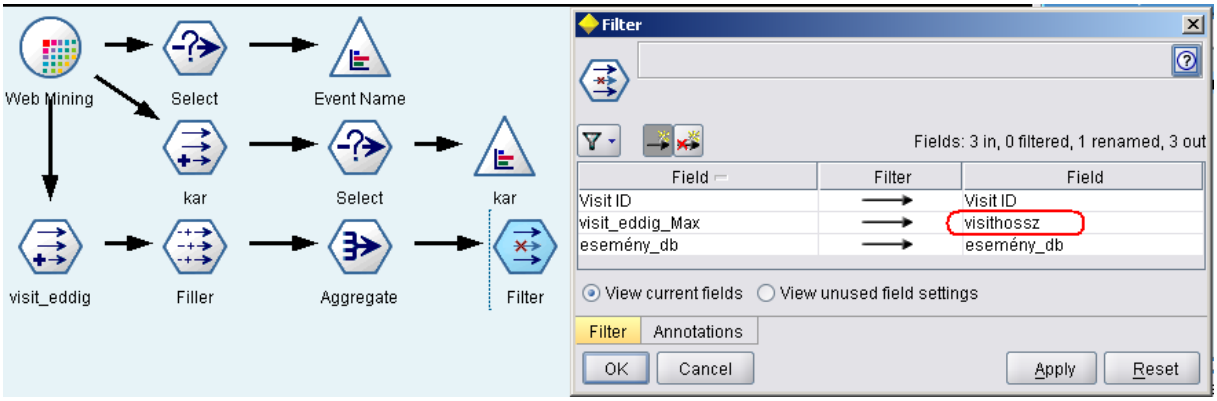
Aggregáljuk (összesítsük) az adattáblánkat úgy, hogy minden rekord egy visitet tartalmazzon, ebben pedig jelenjen meg a hozzá tartozó legnagyobb *visit_eddig* érték (ez a visit hosszát fogja jelenteni) és a visithez tartozó események darabszáma!

Egy *Aggregate* node segítségével tehetjük ezt meg. Az összesítési kulcs a *Visit ID* kell legyen, így minden visitazonosítóhoz egy rekord tartozik majd, a hozzá tartozó legnagyobb *visit_eddig* értéket a **MAX** beállítással, a hozzá tartozó események (eddigi rekordok) számát pedig az **INCLUDE RECORD COUNT IN FIELD** beállítással kaphatjuk meg. Az esemény-darabszámot jelző mezőnek az *esemény_db* nevet célszerű adni:



Már csak az hiányzik, hogy a gép által elnevezett új *visit_eddig_Max* nevű mezőt átnevezzük egy jellemzőbb névre, hiszen ez a visitek tényleges hosszát tartalmazza.

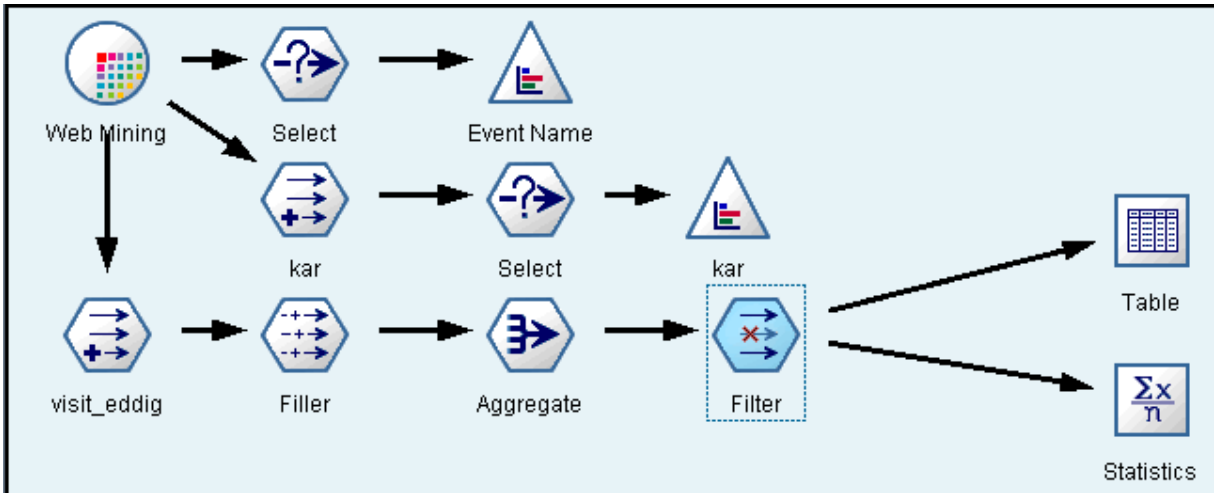
Nevezzük át a mezőt egy *Filter node*-dal!



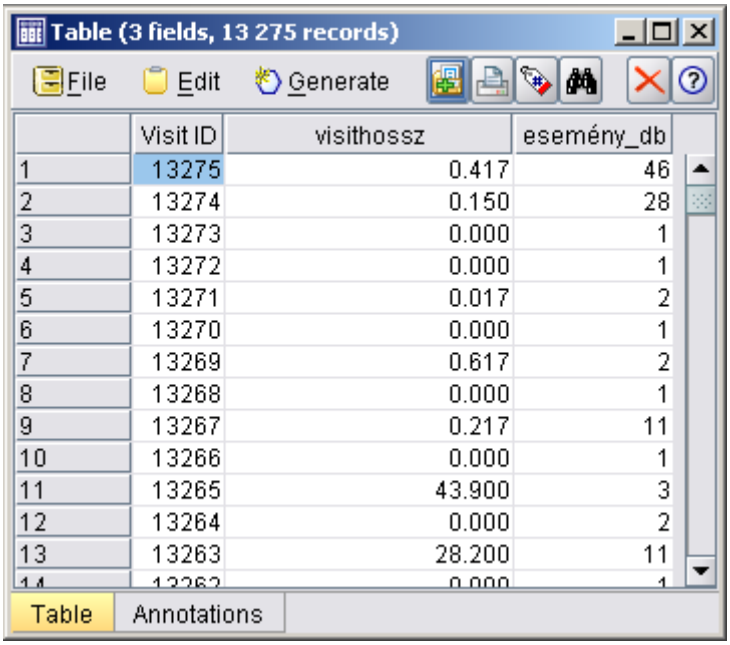
Az ábrán látható az eddigi stream és pirossal megjelölve az átnevezés megadása.

Nézzük meg egy táblában az eredményt és jelenítsük meg az átlagos visithosszat és az események (kattintások) átlagos visiten belüli darabszámát!

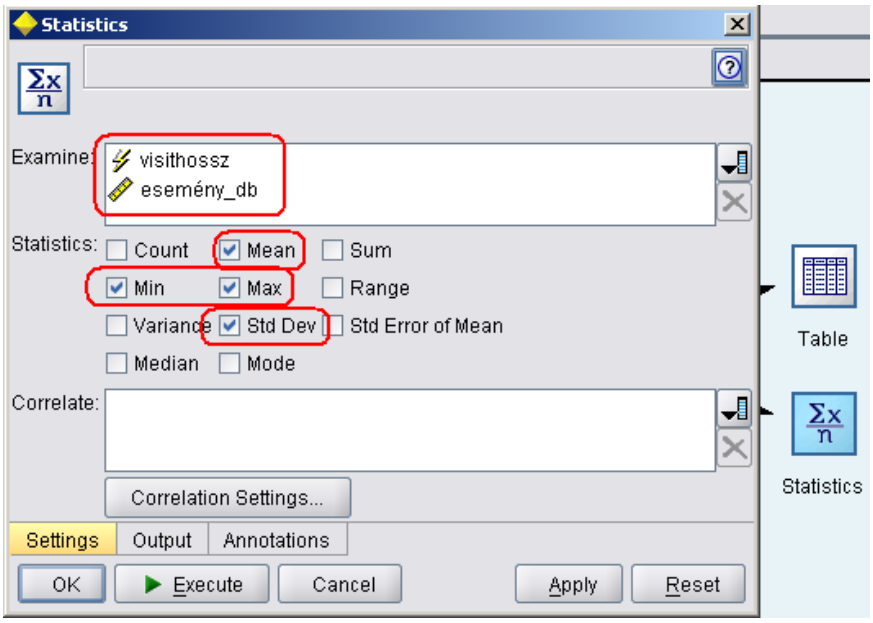
Két új node-ot kell tehát a stream-hez adni:



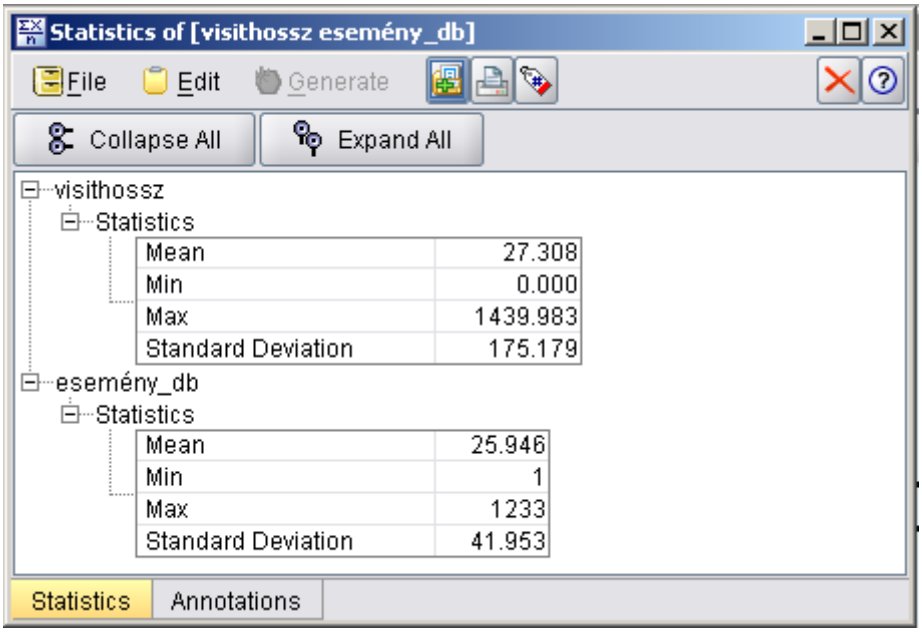
A *Table node* használatát nem kell részleteznünk, csak a futtatási eredményt mutatjuk:



A *Statistics node* beállításai például a következők lehetnek:



A futtatás eredménye pedig ez:



Sajnos, a visithossz *mező nem pontosan adja a portálon való böngészés időtartamát, hiszen vannak egyetlen eseményt tartalmazó visitek is, melyekhez értelemszerűen 0 percnyi idő tartozik. Ennél pontosabb kimutatást nem lehet készíteni, mert a log fájlban nincs információ, hogy valójában mennyi időt töltött a felhasználó egyetlen lekért lap nézegetésével. (Az is lehet, hogy a user felállt a gépe mellől és csak hosszú idő múlva tért vissza, stb.) Ezt az eredmények kiértékelésénél figyelembe kell venni, a túl hosszú visiteket érdemes ilyen szemmel vizsgálni.*

User statisztikák

A portál látogatói közül vannak usernévvel azonosítható bejelentkezők és vannak olyanok, akik a főiskola valamelyik nyilvános gépéről érik el a portált. Hasznos lehet a portál elemzése szempontjából, hogy milyen arányban vannak a látogatók közt a *bejelentkezett* userek, a *belső* gépeken dolgozó, be nem jelentkezett látogatók és azok, akik egy *külső* gépről interneteznek. Az első csoportot az egyszerűség kedvéért bejelentetteknek, a második csoportot belsőknek, a harmadikat külsőknek fogjuk elnevezni, célunk, hogy egy mezőt hozzunk létre, mely e három típust különbözteti meg minden kérés esetén. E gyakorlat során célunk az lesz, hogy e három usertípus megoszlását vizsgáljuk.

Vizsgálhatnánk még a különféle userek visitjeinek számát, a hozzájuk tartozó események számát, az egyes userek leggyakoribb kéréseit, a userekhez tartozó leggyakoribb eseményeket, stb. de e kurzus keretei közt nem fér el az összes vizsgálat.

Először leolvassuk a userek IP címeit. Ezt a *Hostname* mező tartalmazza:

Hostname	Attribute ID
Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp):72.30.252.107	0
KummHttp/1.1 (compatible; KummClient; Linux rulez):195.70.35.179	0
KummHttp/1.1 (compatible; KummClient; Linux rulez):195.70.35.179	0
msnbot/1.0 (http://search.msn.com/msnbot.htm):64.4.8.130	0
msnbot/1.0 (http://search.msn.com/msnbot.htm):64.4.8.112	0
KummHttp/1.1 (compatible; KummClient; Linux rulez):195.70.35.179	0
msnbot/1.0 (http://search.msn.com/msnbot.htm):65.54.188.56	0
Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp):72.30.133.139	0
KummHttp/1.1 (compatible; KummClient; Linux rulez):195.70.35.179	0

A táblázatrészből látható, hogy minden IP-cím a *Hostname* mezőben levő string végén szerepel és **„:”** karakter előzi meg. Az IP címet tehát string-függvényekkel „leválaszthatjuk” a stringről. Három függvényt célszerű használni.

- A **LENGTH**(*string-adat*) függvény a string-adat hosszát adja numerikus értékként.
- A **LOCCAR_BACK**(*karakter, string-adat*) a megadott karakternek a string-adatban való utolsó előfordulási helyét adja vissza numerikus értékként.

Figyelem! Egyetlen karaktert nem stringként, idézőjelekkel határolva, hanem külön karakterre utaló formátumban, balra dőlő aposztrófok között kell megadni a Clem nyelvben, pl. **`a` .** *(A határolójelek írása AltGr+7 billentyűkombinációval történhet.)*

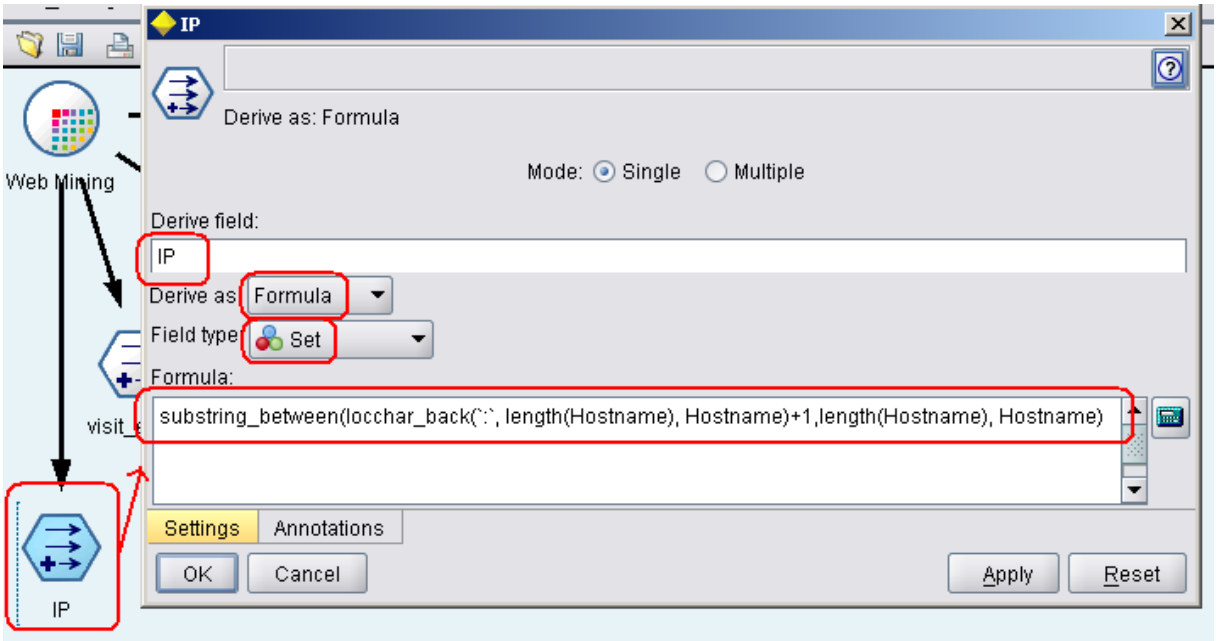
- A **SUBSTRING_BETWEEN**(*numerikus adat , numerikus adat , string-adat*) függvény az első numerikus értéktől a másodikig terjedő részét adja a benne szereplő string-adatnak.

Mindezeket figyelembe véve a következő képlet szolgáltatja egy-egy user IP-címét:

substring_between(locchar_back(`:`, length(Hostname), Hostname)+1,length(Hostname), Hostname)

Készítsünk olyan új *IP* nevű mezőt, mely minden eseménynél az eseményt indító user IP címét tartalmazza! (Szöveges tárolású adatok legyenek.)

Ehhez egy *Derive node* szükséges. Szúrjuk be a *Web Mining node* után közvetlenül és a megadott képletet írjuk be a beállításaihoz a következő módon:



Most a táblához egy „bejelentett”, „belső” és „külső” usereket jelző mezőt kellene hozzávenni. A bejelentettek rekordjaiban az *Authorized User Name* mező nem üres. A következő feltétellel lehet például ezeket kiválasztani:

length('Authorized User Name')>0

A belső userek IP címe vagy 10-essel kezdődik, vagy 193.224.74-gyel vagy 193.225.199-cel. Ezek kiválasztásához szükség lesz a **STARTSTRING**(*numerikus adat, string-adat*)

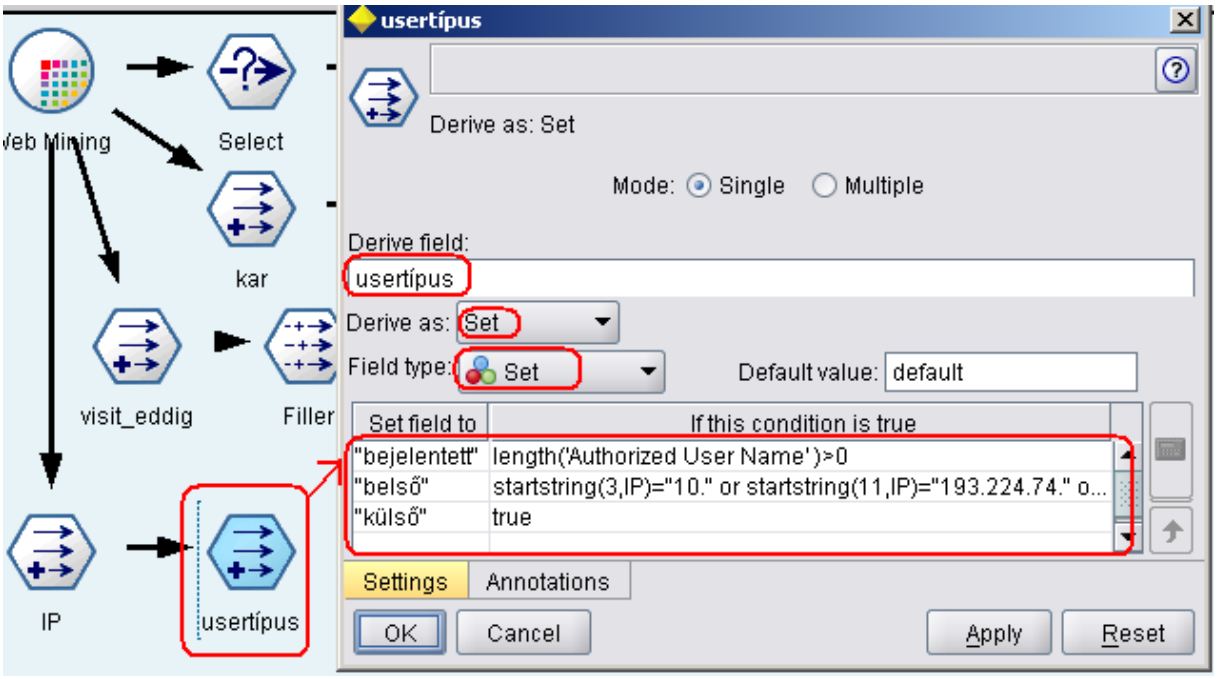
függvényre, mely a numerikus értéknek megfelelő betűszámú részét adja a string-adatnak az elejétől kezdve.

A következő feltétel tehát éppen a belső userek rekordjaira lesz igaz:

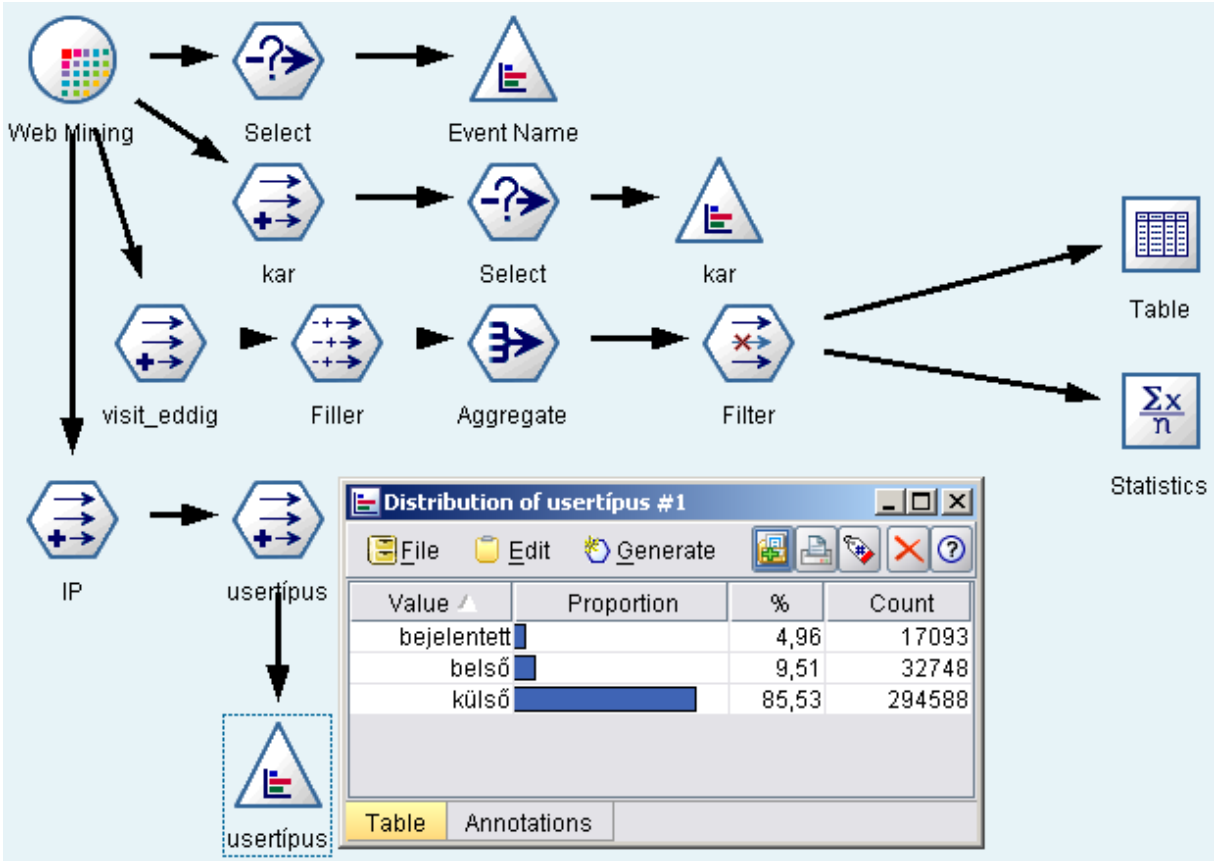
startstring(3,IP)="10." or startstring(11,IP)="193.224.74." or startstring(12,IP)="193.225.199."

Készítsünk olyan új *usertípus* nevű mezőt, mely minden eseménynél az eseményt indító user típusát tartalmazza! (Lehetséges értékei: "*bejelentett*", "*belső*", "*külső*" – szöveges tárolású adatok.)

Ehhez egy újabb *Derive node* szükséges. Mivel háromféle szöveges értéket kell tartalmazzon, ezért Set típusú mezőt kell létrehoznunk. Szúrjuk be az előző node után és állítsuk be a beállításait a következő módon:



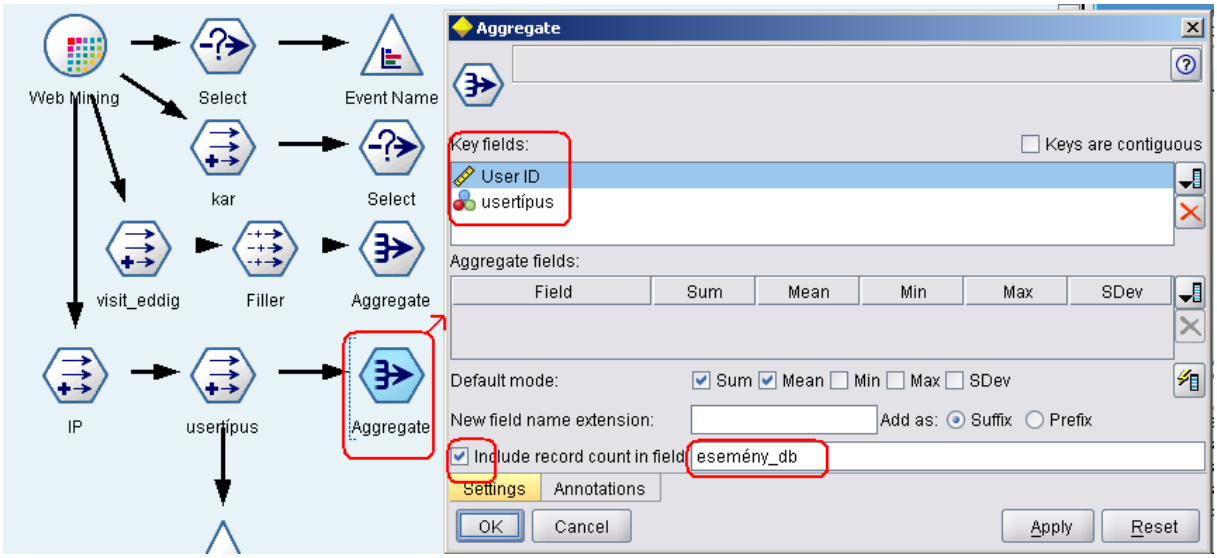
Az események usertípus szerinti megoszlását már könnyen kiirathatjuk egy *Plot node*-dal. Tegyük meg!



Lehet, hogy egy userhez több, a másikhoz kevesebb esemény tartozik.

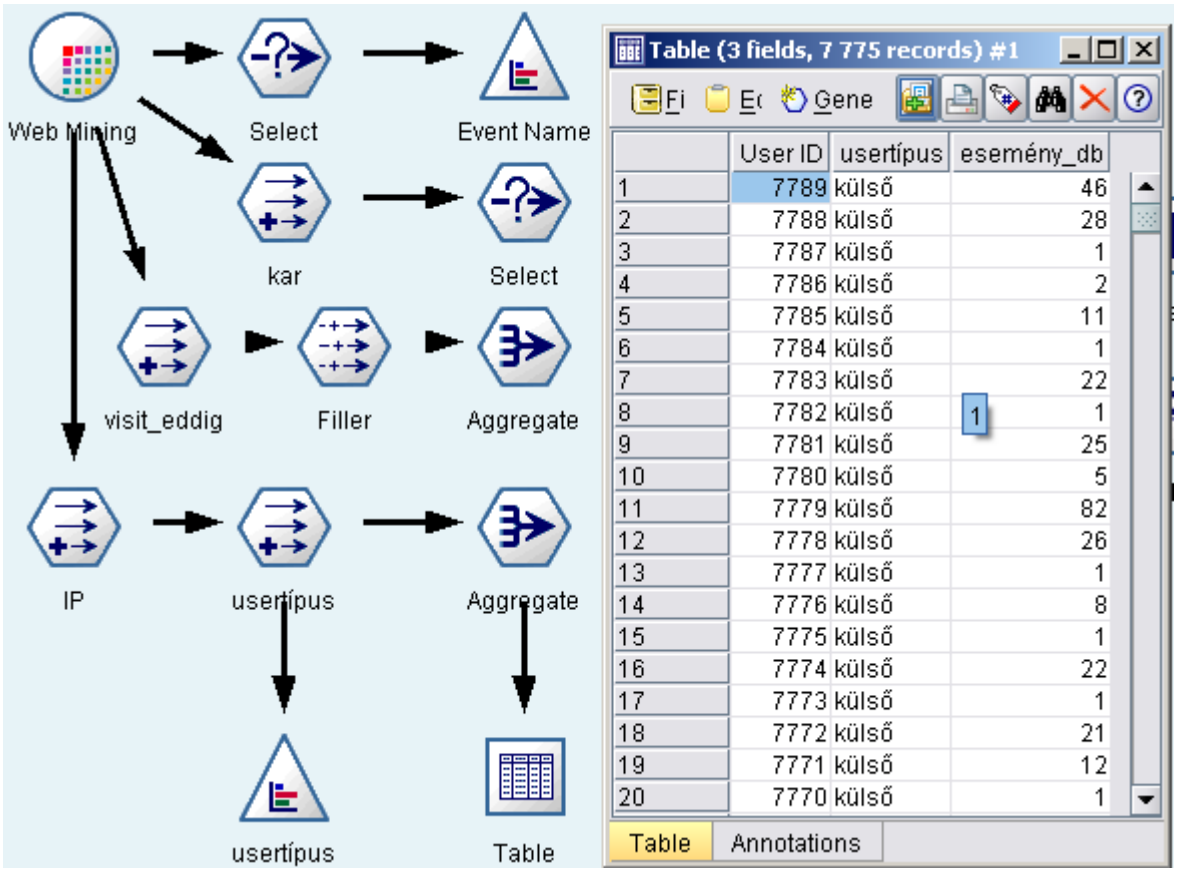
Tároljuk egy mezőben, hogy melyik userekhez összesen mennyi esemény tartozik!

Egy *Aggregate node* megteszi ezt:



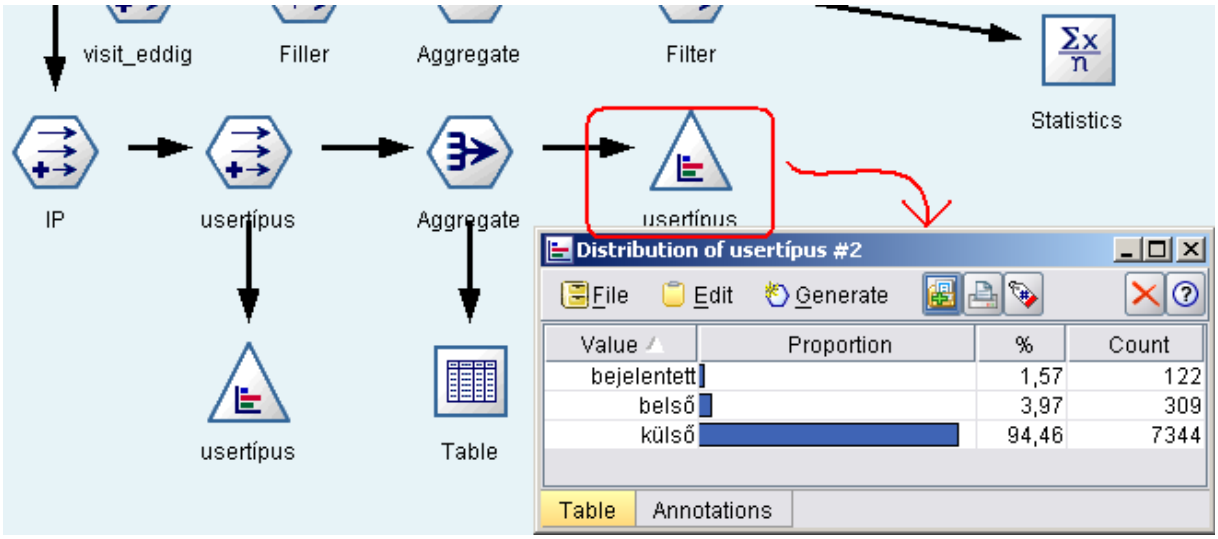
A megadott beállításokkal minden rekord egy usert fog jelenteni, mellette a usertípus és a hozzá tartozó rekordok (események) száma fog tartozni.

Egy táblában ezt meg is tekinthetjük, 7775 darab usert kaptunk:



Jelenítsük meg az userek usertipus szerinti megoszlását!

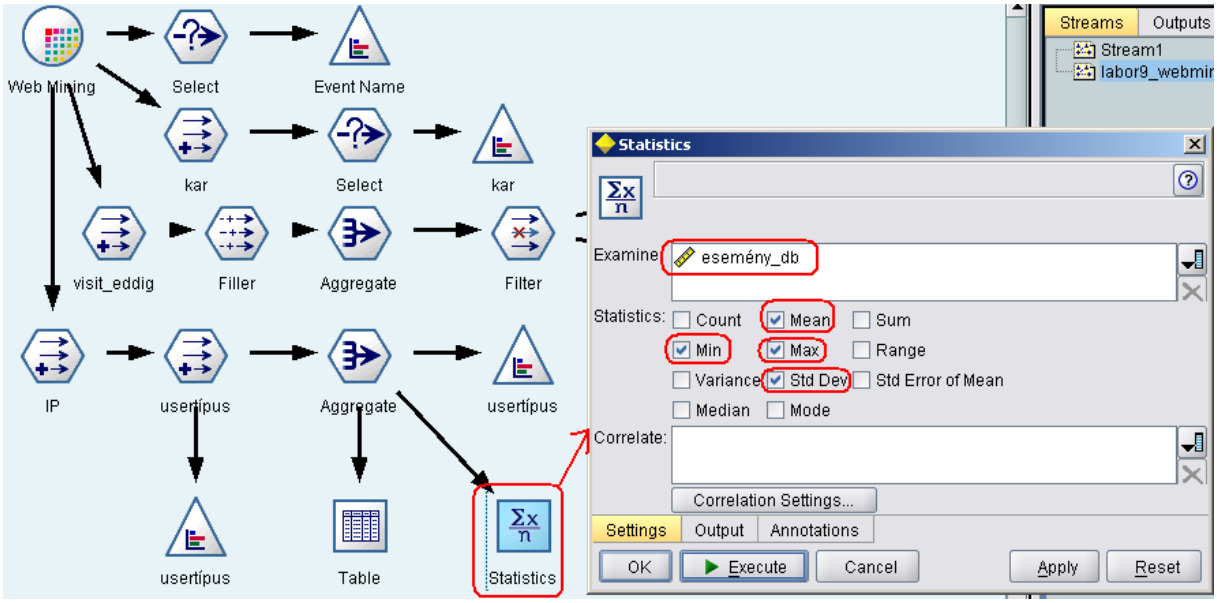
Ehhez egy *Distribution node*-ra lesz szükség. A futtatás eredménye:



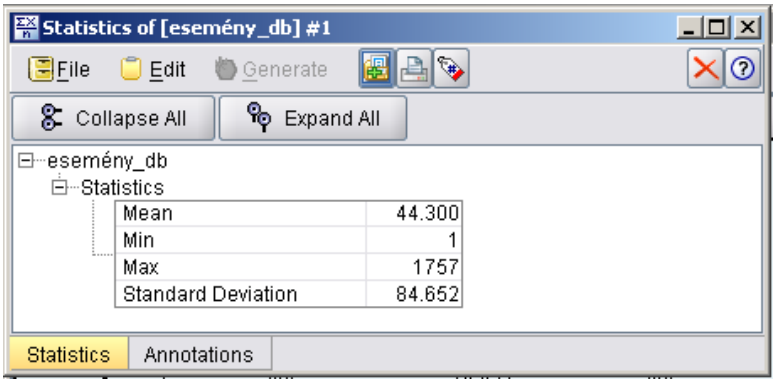
Látható, hogy az előző megoszlási grafikonhoz képest az arányok változtak. Ha belegondolunk, ez arra utal, hogy a belső és a bejelentett userek többet dolgoztak (több eseményt generáltak) a portálon, mint a külső látogatók.

Számítsuk ki, egy userhez átlagosan hány esemény tartozik!

Ezt egy *Statistics node* számítja ki nekünk, a következő beállításokkal:



A futtatás eredménye adja az egy userre eső események átlagos számát, legkisebb és legnagyobb értékét, illetve szórását:



Mivel az egy visitre eső események átlagos száma 25, ebből megbecsülhető, hogy egy userre átlagosan másfél visit esik – ezt természetesen a Clementine-nal pontosan is meg lehet határozni.

Gyakorló feladatok:

Egészítsük ki az eseménytáblát úgy, hogy a */konyvtar/* **URL-töredéknek megfelelő eseményeket is figyelhessük! (Ezek a főiskolai könyvtár web-oldalaira irányuló kérések.)**

Nézzük meg a *könyvtár* **eseményeit az esemény-statisztikákban!**

Készítsünk olyan új mezőt, melynek segítségével kimutathatjuk, milyen százalékos arányban fordulnak elő 0 hosszúságú visitek!

Készítsünk olyan új mezőt, melynek segítségével kimutathatjuk, milyen százalékos arányban fordulnak elő két óránál hosszabb visitek!

Készítsük el a különböző userek visitre vonatkozó statisztikáit! A visiteket aggregáló node-nál állítsuk be a KEY FIELDS beállításnál az User ID-t is, és ezután aggregáljunk User ID szerint! Számíttassuk ki ezután az egy userre eső visitek számát!

Számítsuk ki, hány esemény tartozik átlagosan egy belső, egy külső és egy bennfentes userhez!