Dataconda Tutorial*

Michele Samorani
University of Alberta School of Business

More tutorials are available on Youtube

What is Dataconda?

- Software program to generate a mining table from a relational database
- State-of-the-art attribute generation
- Full version is free for research and teaching purposes

1. Install *Dataconda* (www.dataconda.net)
2. (optional but recommended) Install *R* (http://cran.r-project.org/bin/windows/base/)
3. (optional but recommended) Install *Weka* (http://www.cs.waikato.ac.nz/ml/weka/downloading.html)

Outline

1. Motivation
2. Attribute generation algorithm
3. How to use Dataconda
    1. Load data
    2. Generate Attributes
    3. Interpret the Output
    4. Extend Dataconda
4. Some Experiments

Motivation

Idealized vs Real view of classification

Real

Idealized

Flat Mining table

Classifier

New knowledge

Relational Database

This is the actual critical step
(and the main task performed by Dataconda)

A lot of research focuses on this step

Table 2.1: The *Purchases* table

PurchaseID	Date	Online	ClientID	ProductID	Return
Pur1	Oct 10	1	Cli1	Pro1	1
Pur2	Oct 11	0	Cli2	Pro2	0
Pur3	Oct 14	0	Cli1	Pro2	0
Pur4	Oct 31	0	Cli3	Pro3	1

Class

Table 2.2: The *Clients* table

ClientID	Gender	Age
Cli1	M	33
Cli2	F	45
Cli3	M	28

Table 2.3: The *Products* table

ProductID	Price
Pro1	$200
Pro2	$100
Pro3	$160

- Problem: classify purchases by Return
- First step: build a flat mining table:

Purchase ID							**Return (0/1)**
Pur1							1
Pur2							0
Pur3							0
Pur4							1

Building the mining table

- **MANUALLY**
  - The analyst formulates hypotheses
  - Computes only the attributes that she suspects will confirm or reject the hypothesis
  - Cons:
    - Time consuming
    - Limited knowledge discovery

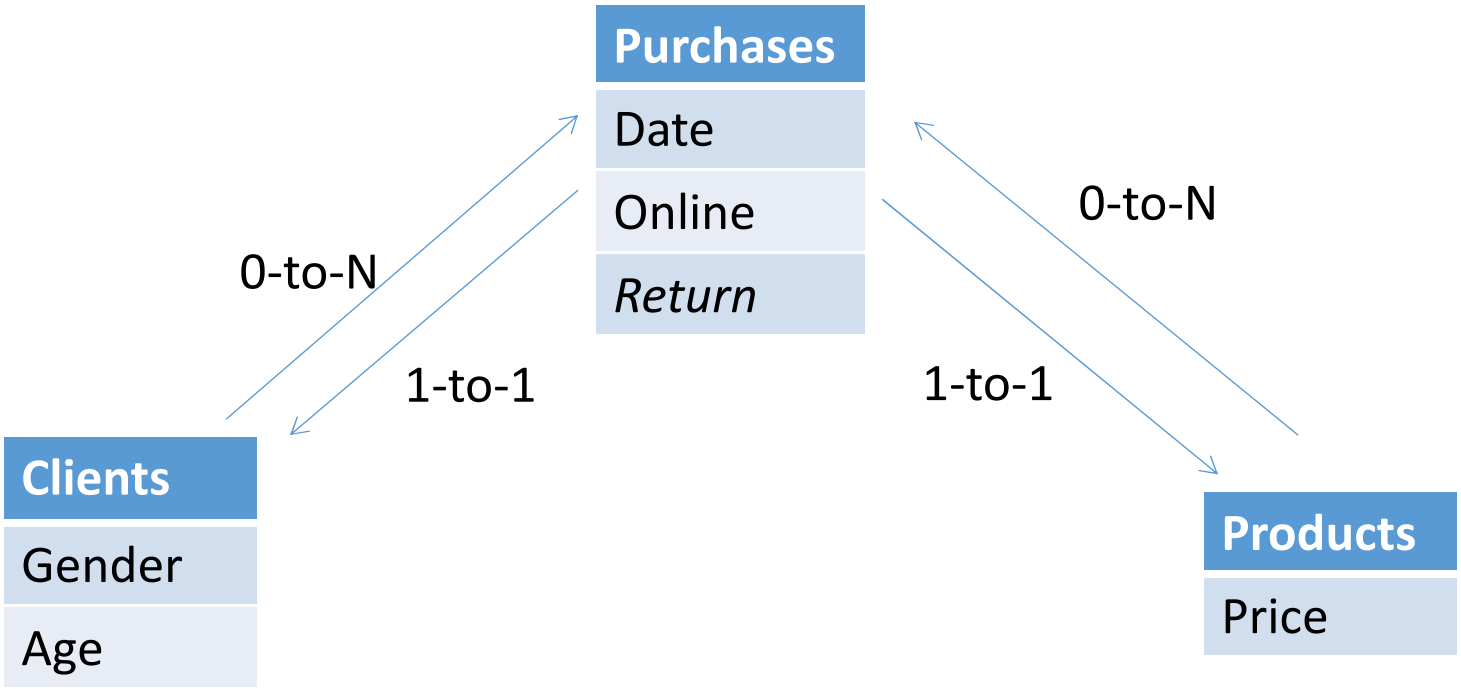
- **AUTOMATICALLY**
  - A software generates the hypotheses (attributes)
  - Pros:
    - Fast
    - Enhanced knowledge discovery

QUICK DEMO…

Attribute Generation Algorithm

Building attributes automatically

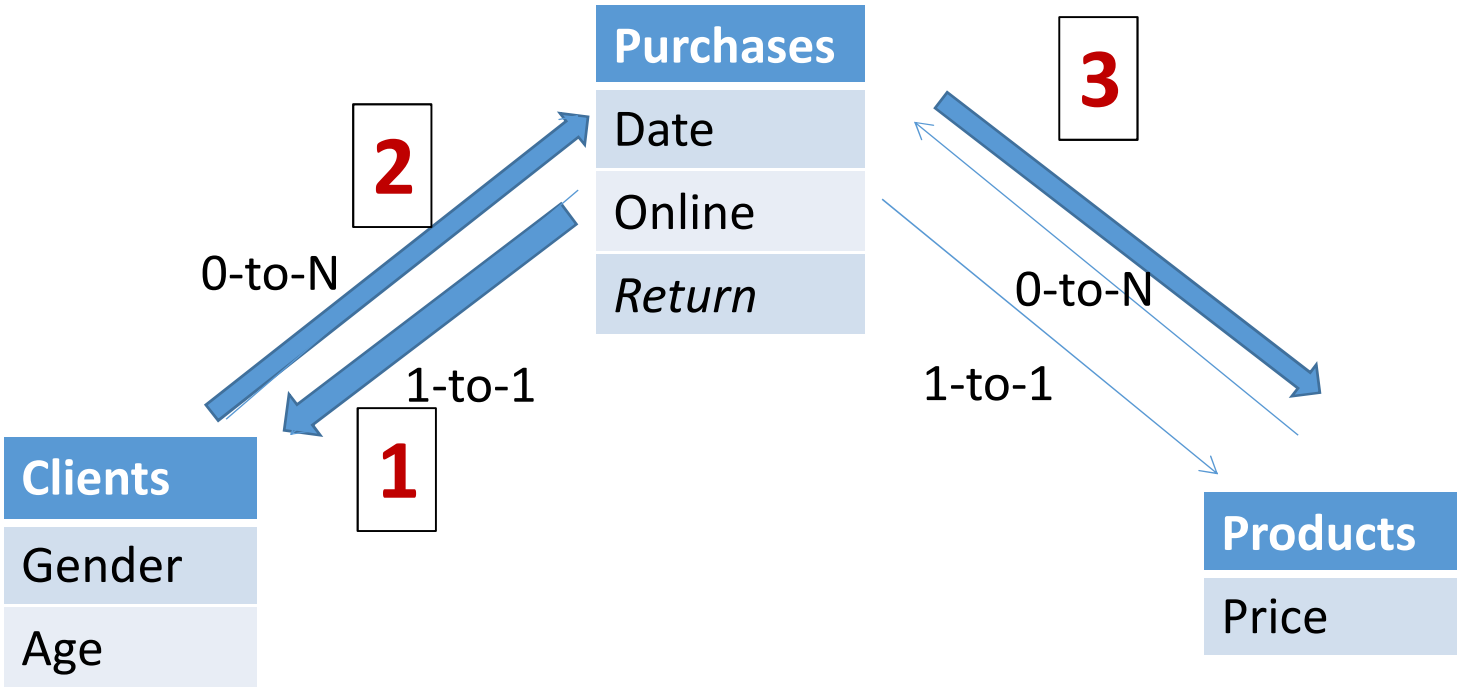
- Consider the *Entity-Relationship* diagram of the database



- The idea is to add attributes to the *target table* (Purchases)

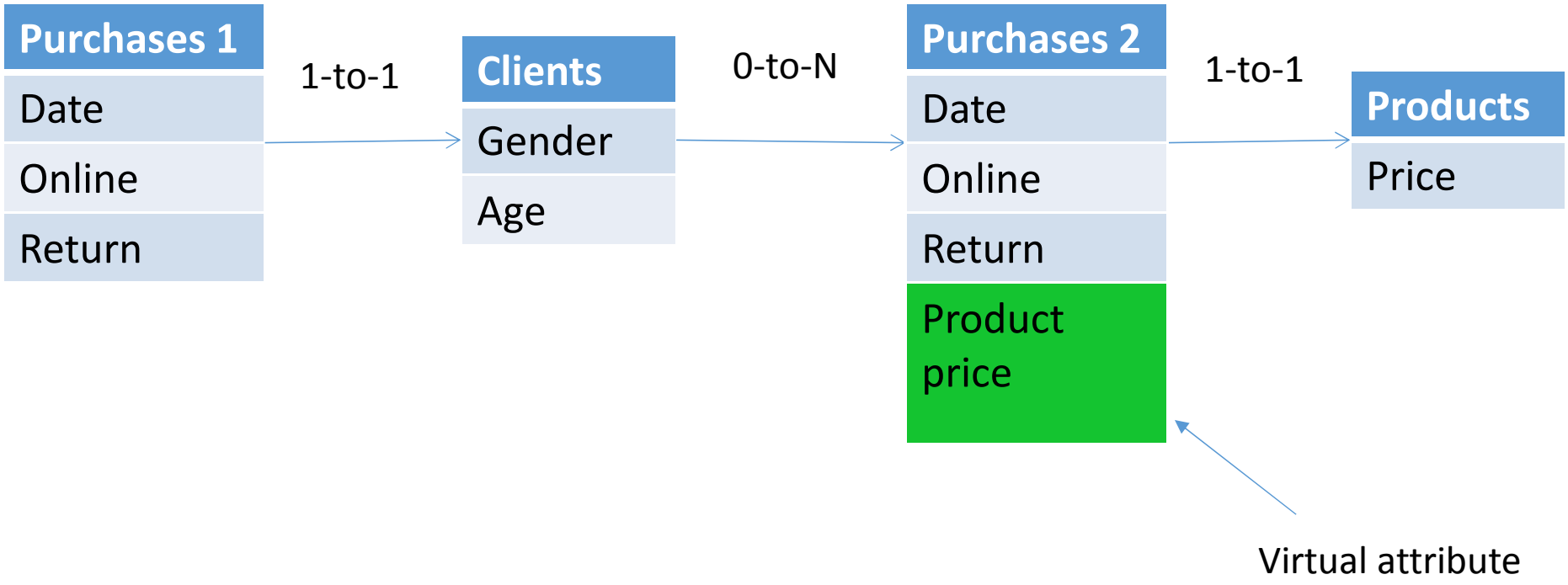
Building attributes automatically

- Step 1: choose a path $t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_n$ from the target table

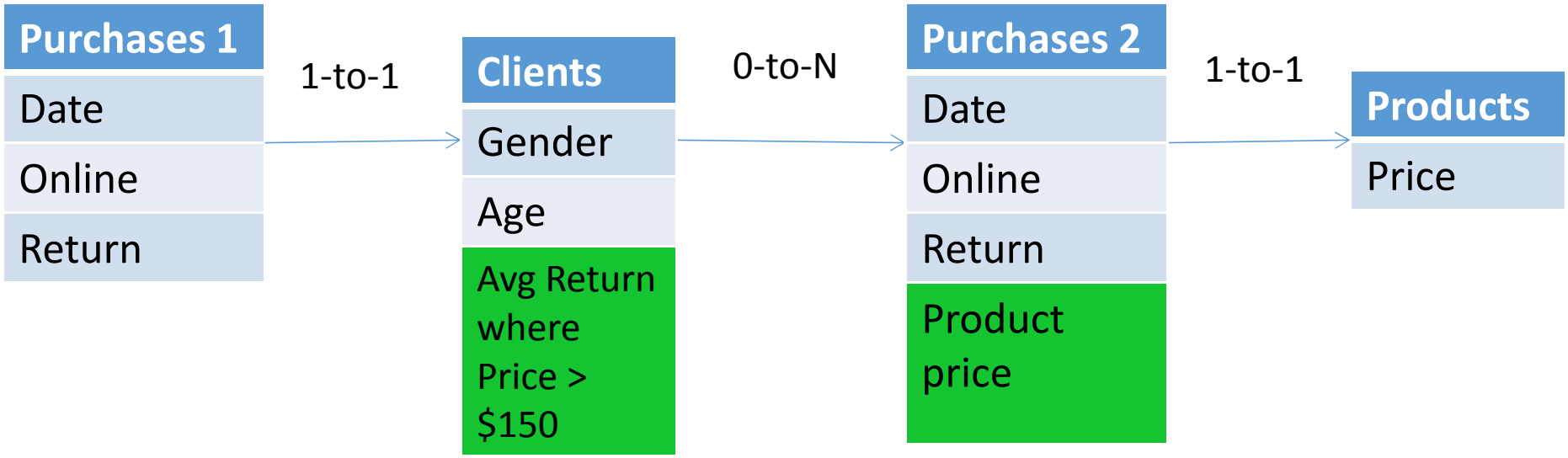


4 tables → Attributes at depth 4

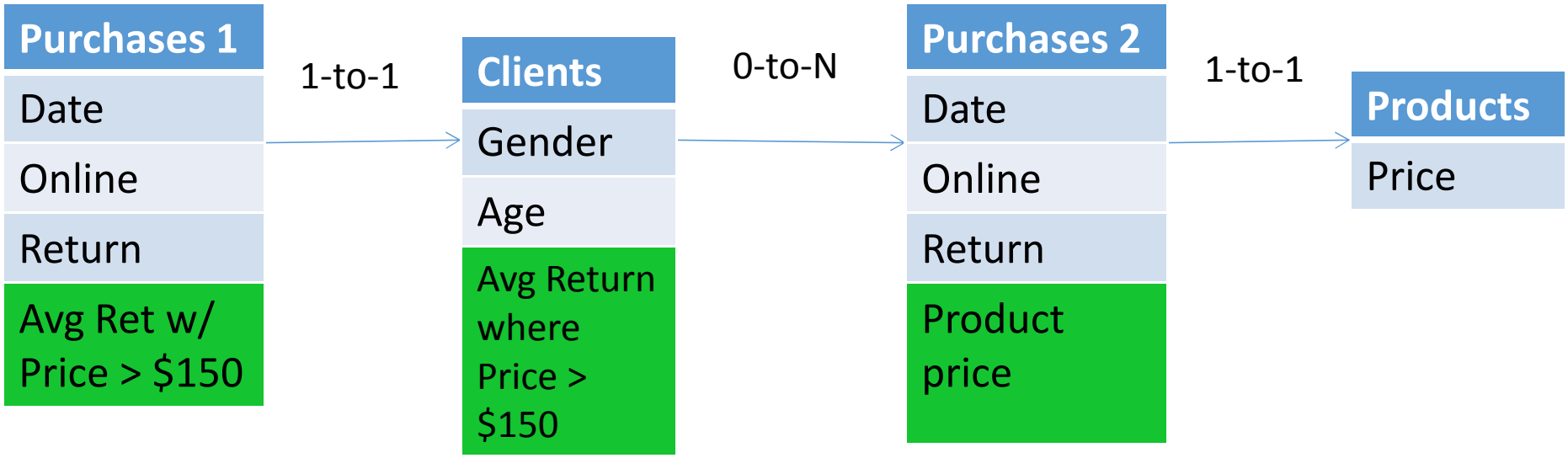
Step 2: “roll up” from the end of the path



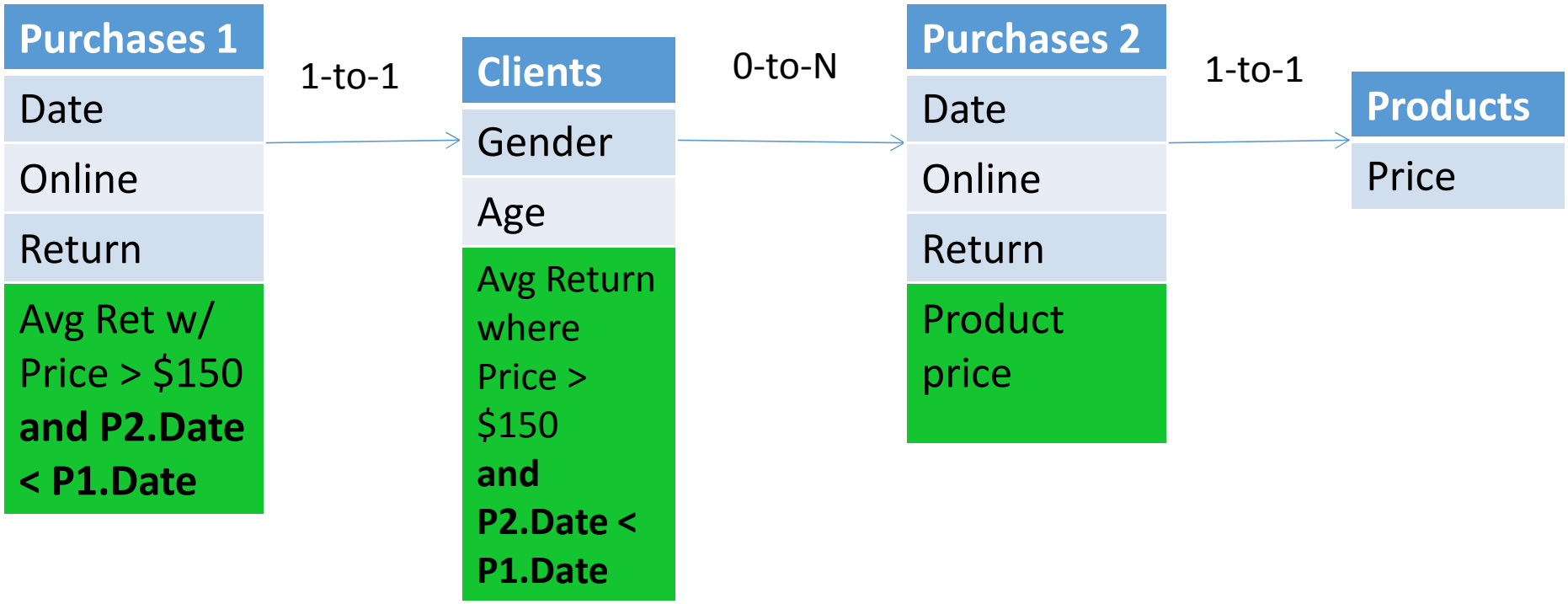
Add a column to *Purchase 2*, which brings in information from *Products*



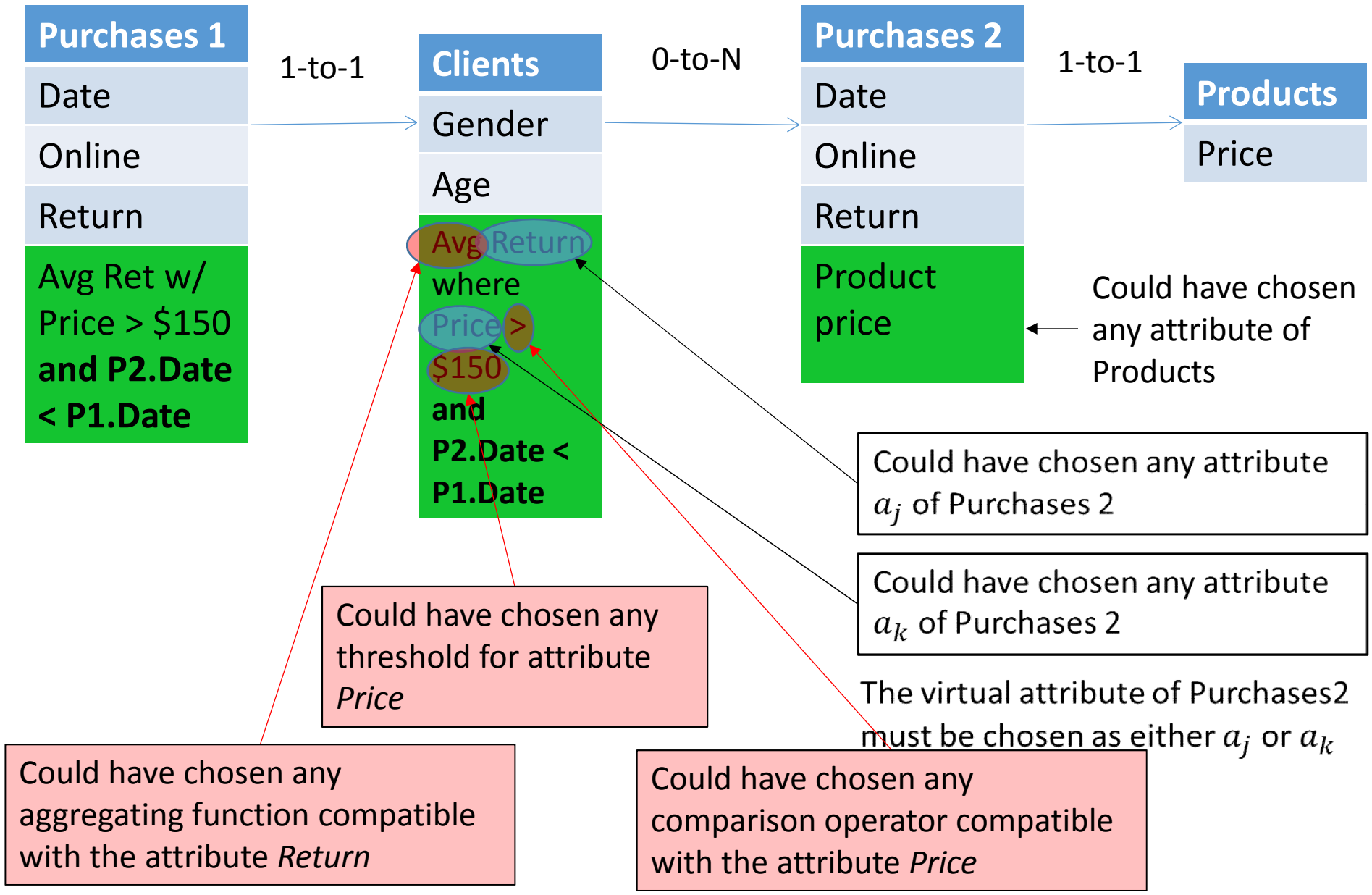
Add a column to *Clients*, which brings in information from *Purchases 2*



Add a column to *Purchases 1*, which brings in information from *Clients*



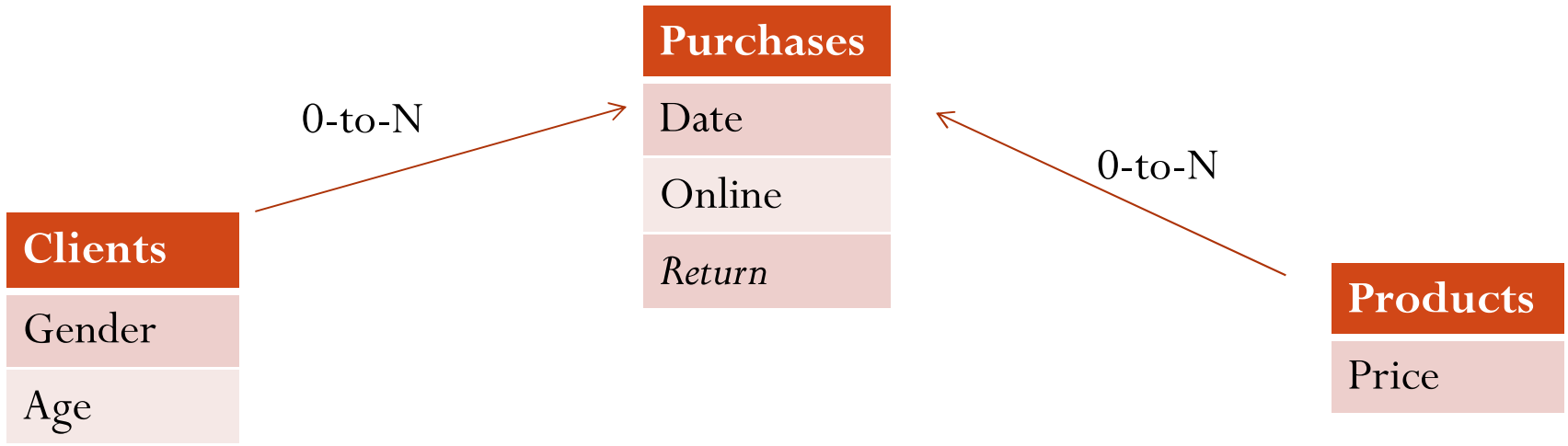
Add the where condition on the date



The red boxes above represent user settings. Let's see how to configure them…

How to Use Dataconda

Simulated Data

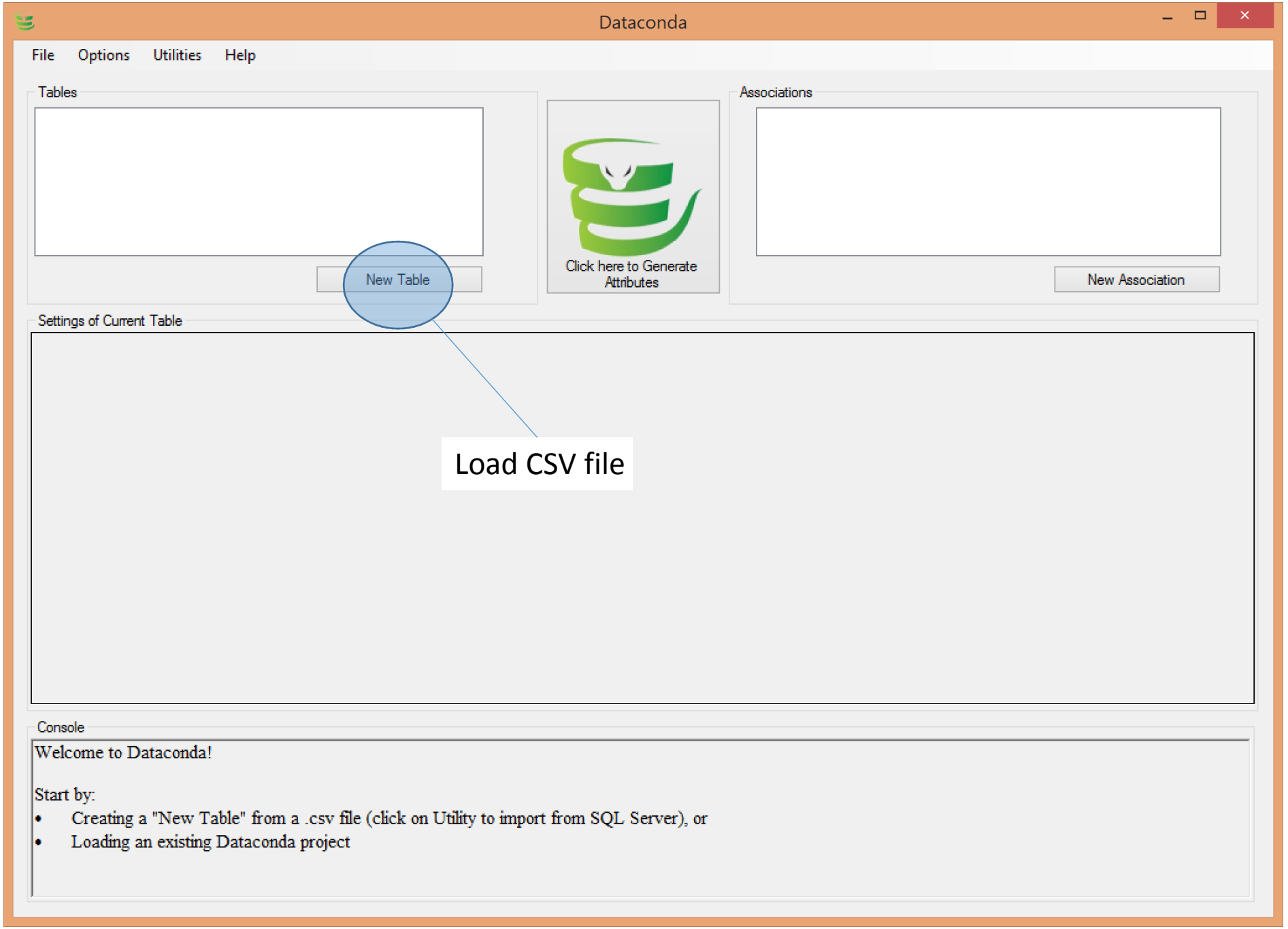


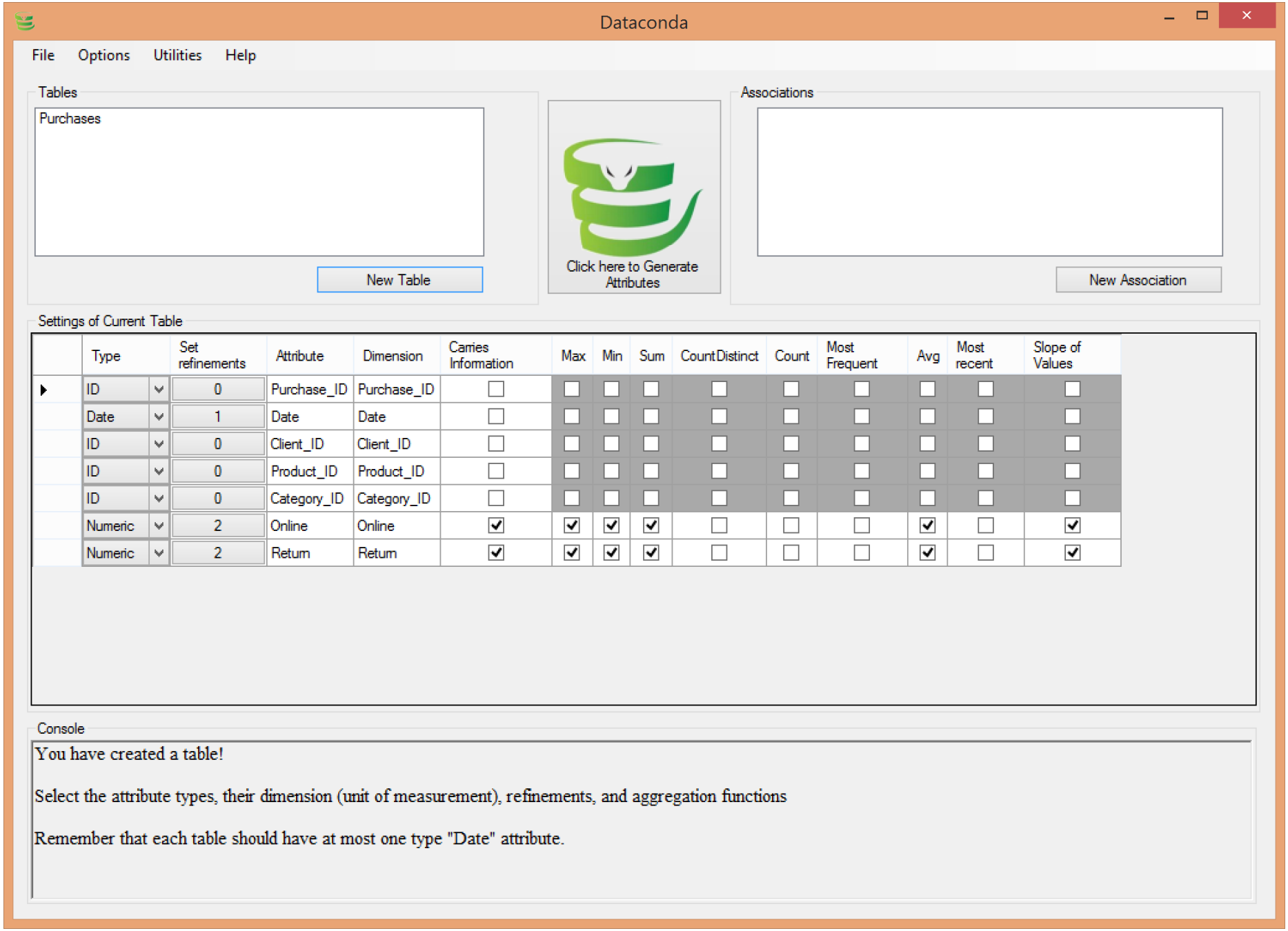
Build all the tables randomly,
except for the target attribute **Return**:

P(Return) ~ logit^{-1}(-3

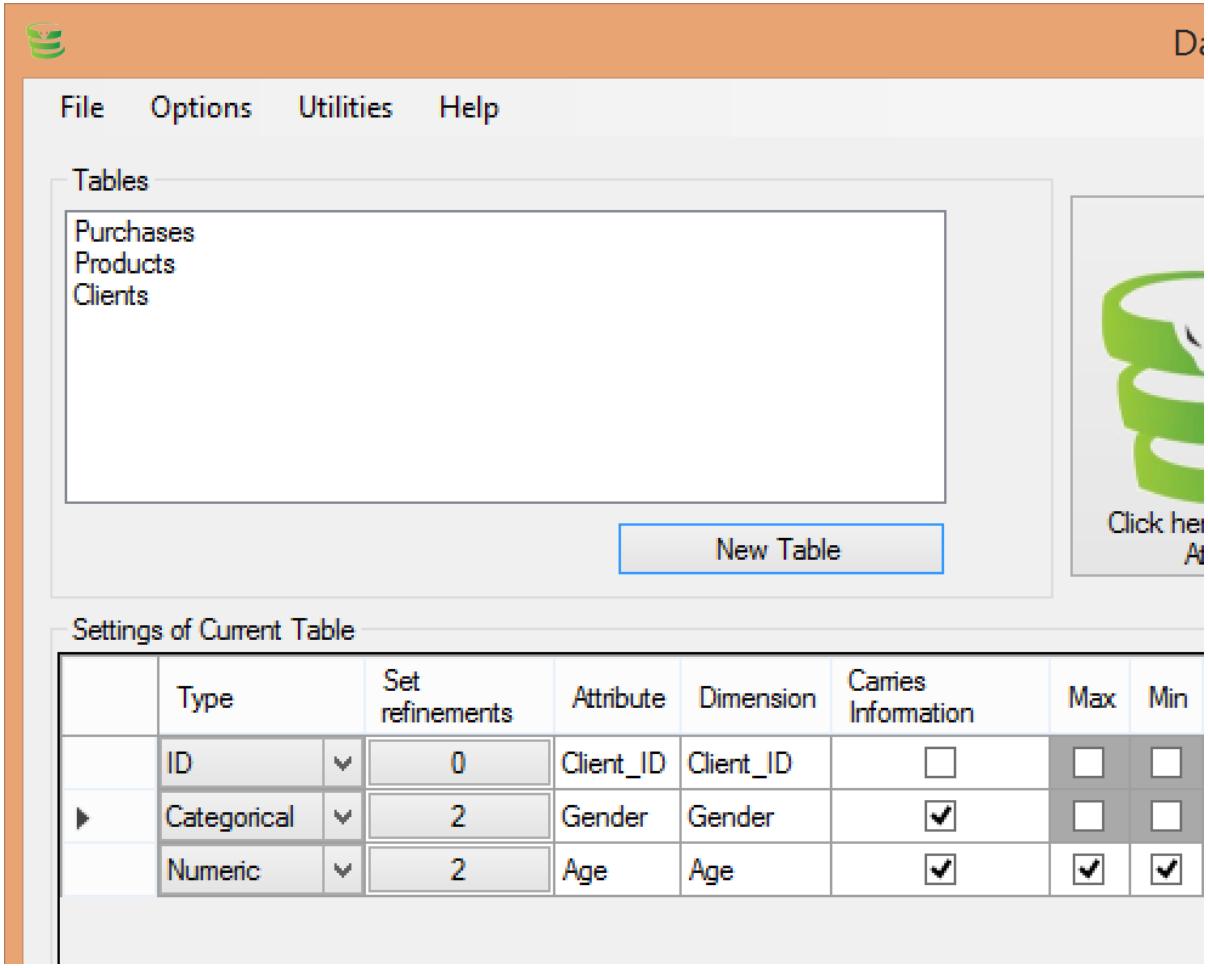
Depth 2 → + 0.02 * price
Depth 3 → + 2 * customer's past return rate
Depth 4 → - 0.1 * age of the client who most recently purchased the current product online)

TRUE VARIABLES

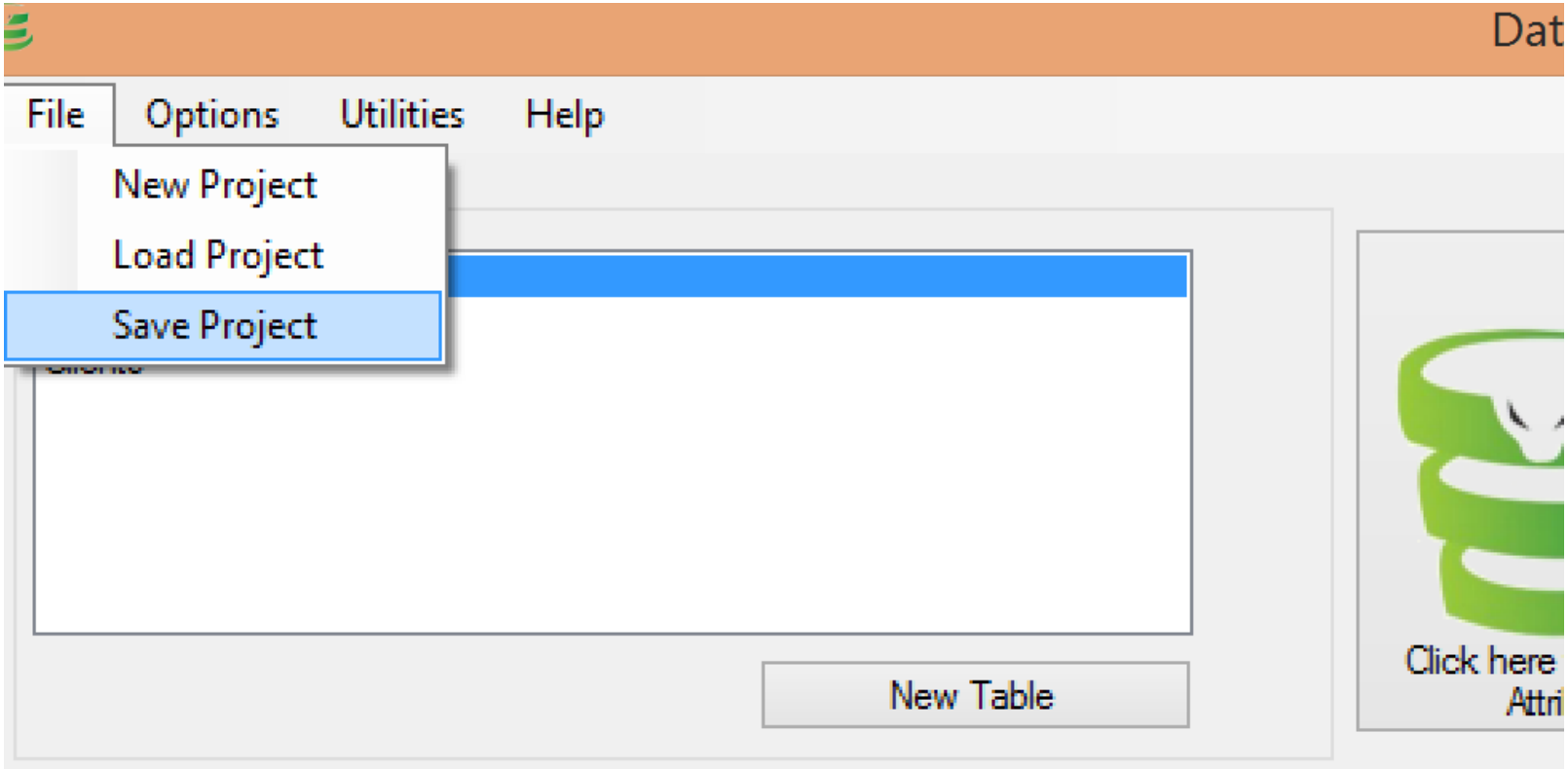

Dataconda
File
Options
Utilities
Help
Tables
New Table
Click here to Generate Attributes
Associations
New Association
Settings of Current Table
Load CSV file
Console
Welcome to Dataconda!
Start by:
• Creating a "New Table" from a .csv file (click on Utility to import from SQL Server), or
• Loading an existing Dataconda project


Dataconda
File
Options
Utilities
Help
Tables
Purchases
New Table
Click here to Generate Attributes
Associations
New Association
Settings of Current Table
Type
Set refinements
Attribute
Dimension
Carries Information
Max
Min
Sum
Count Distinct
Count
Most Frequent
Avg
Most recent
Slope of Values
ID 0 Purchase_ID Purchase_ID
Date 1 Date Date
ID 0 Client_ID Client_ID
ID 0 Product_ID Product_ID
ID 0 Category_ID Category_ID
Numeric 2 Online Online
Numeric 2 Return Return
Console
You have created a table!
Select the attribute types, their dimension (unit of measurement), refinements, and aggregation functions
Remember that each table should have at most one type "Date" attribute.

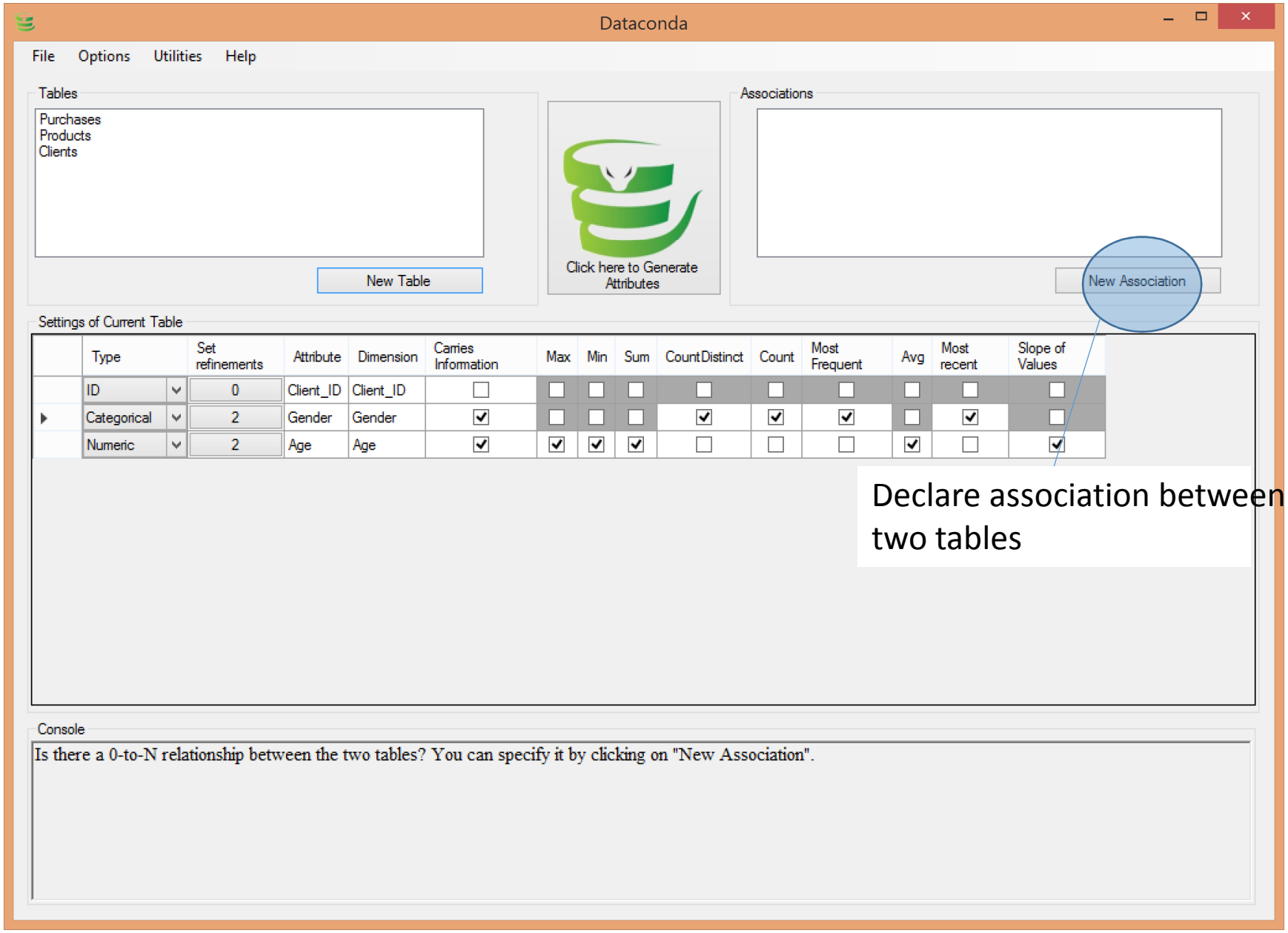
Load all three tables

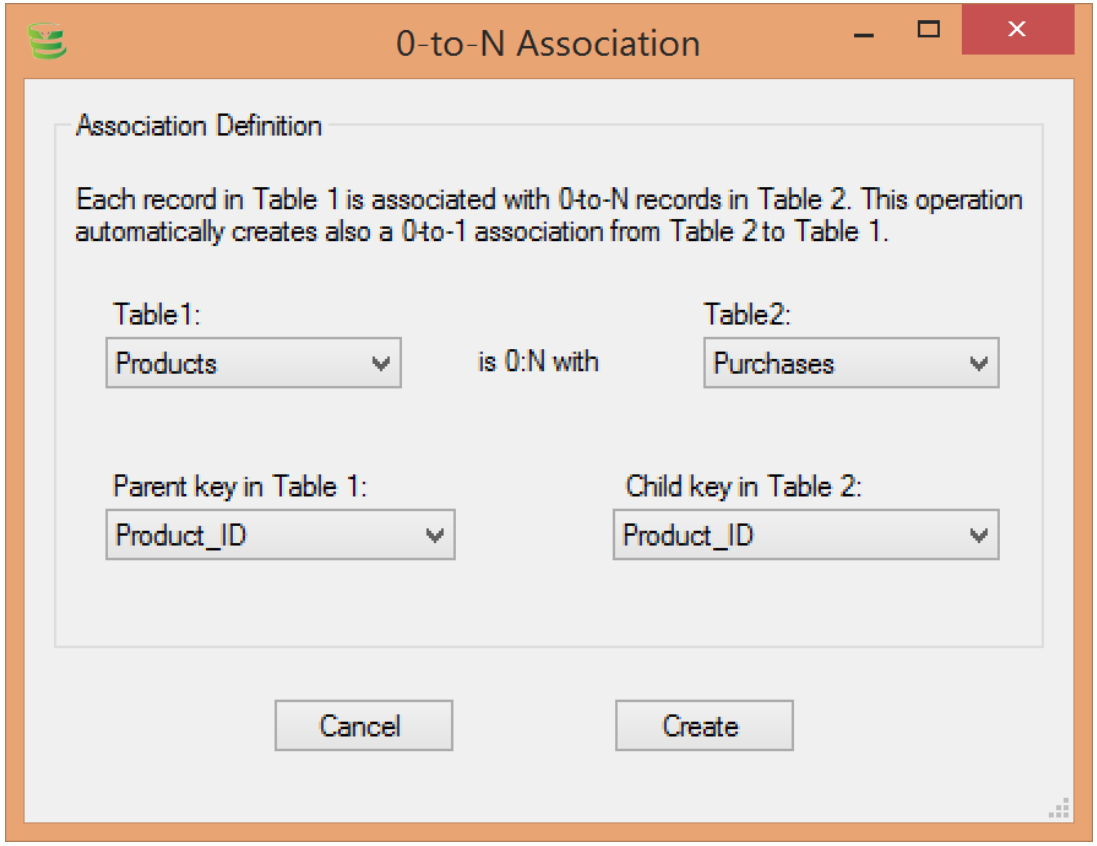


Save/Load project



You can save the project at any moment.
Note that the project file does NOT contain the data. So, it needs to be saved in the same folder as the csv files

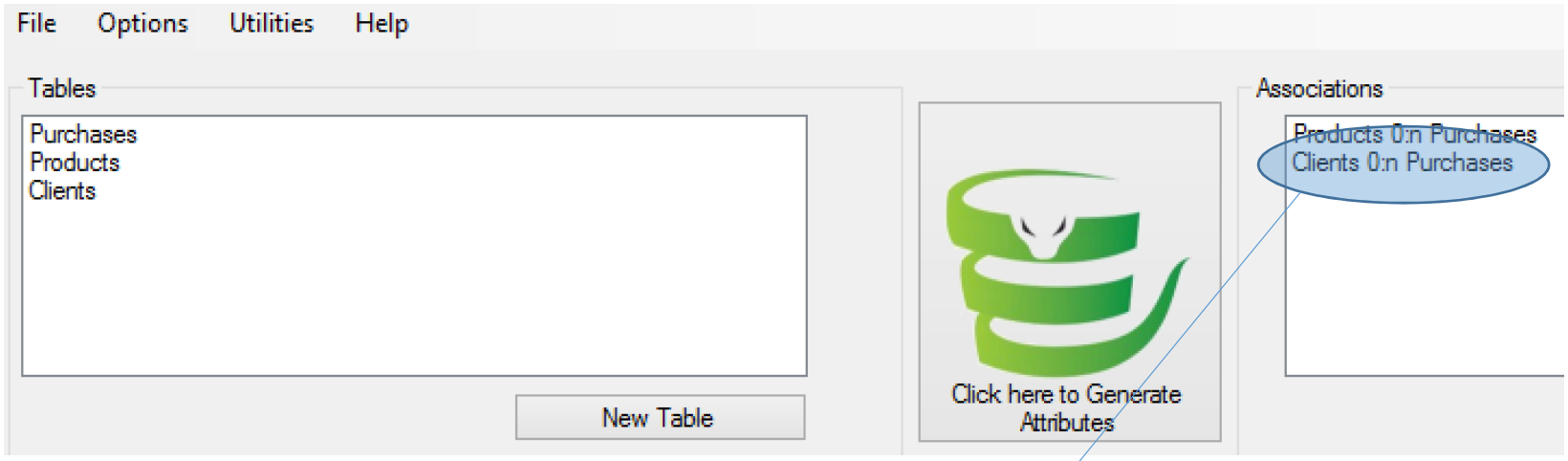

Dataconda
File
Options
Utilities
Help
Tables
Purchases
Products
Clients
New Table
Click here to Generate Attributes
Associations
New Association
Settings of Current Table
Type
Set refinements
Attribute
Dimension
Carries Information
Max
Min
Sum
CountDistinct
Count
Most Frequent
Avg
Most recent
Slope of Values
ID
0
Client_ID
Client_ID
Categorical
2
Gender
Gender
Numeric
2
Age
Age
Declare association between two tables
Console
Is there a 0-to-N relationship between the two tables? You can specify it by clicking on "New Association".



Dataconda only supports *0:n* and *0:1* associations

In the example above, we declare in one step:

- A 0:n association from Products to Purchases
- A 0:1 association from Purchases to Products



Declare the Clients → Purchases association

Settings of Table Purchases

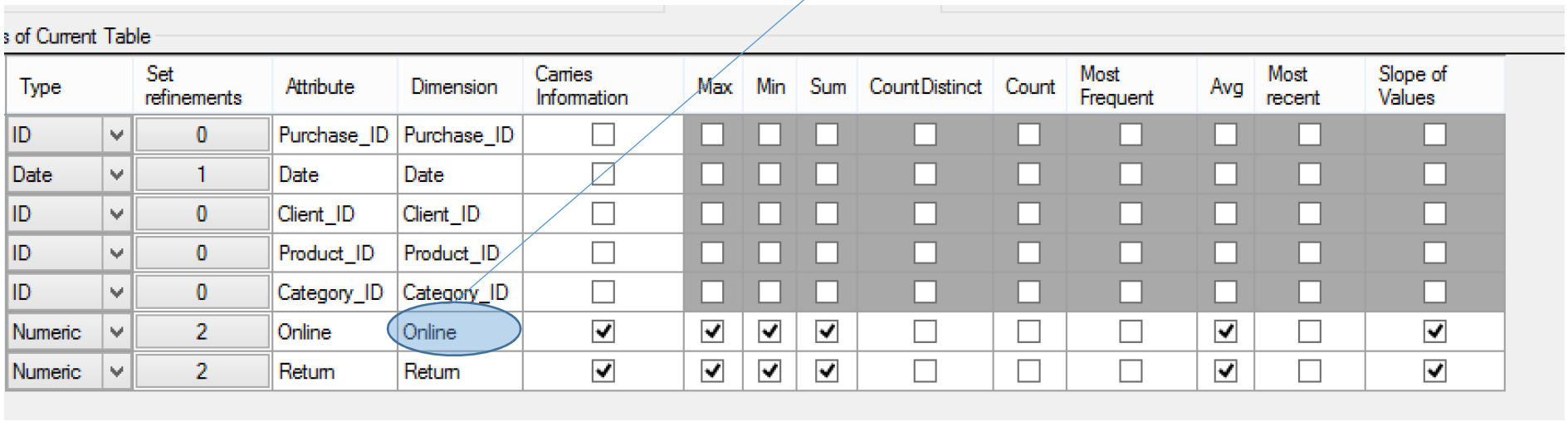
Type of attribute ∈ {Categorical, Numeric, ID, Date}
There can be only one Date attribute per table,
and the records MUST be sorted by date ascending

s of Current Table

Type	Set refinements	Attribute	Dimension	Carries Information	Max	Min	Sum	Count Distinct	Count	Most Frequent	Avg	Most recent	Slope of Values
ID	0	Purchase_ID	Purchase_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Date	1	Date	Date	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
ID	0	Client_ID	Client_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
ID	0	Product_ID	Product_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
ID	0	Category_ID	Category_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Numeric	2	Online	Online	☑	☑	☑	☑	☐	☐	☐	☑	☐	☑
Numeric	2	Return	Return	☑	☑	☑	☑	☐	☐	☐	☑	☐	☑

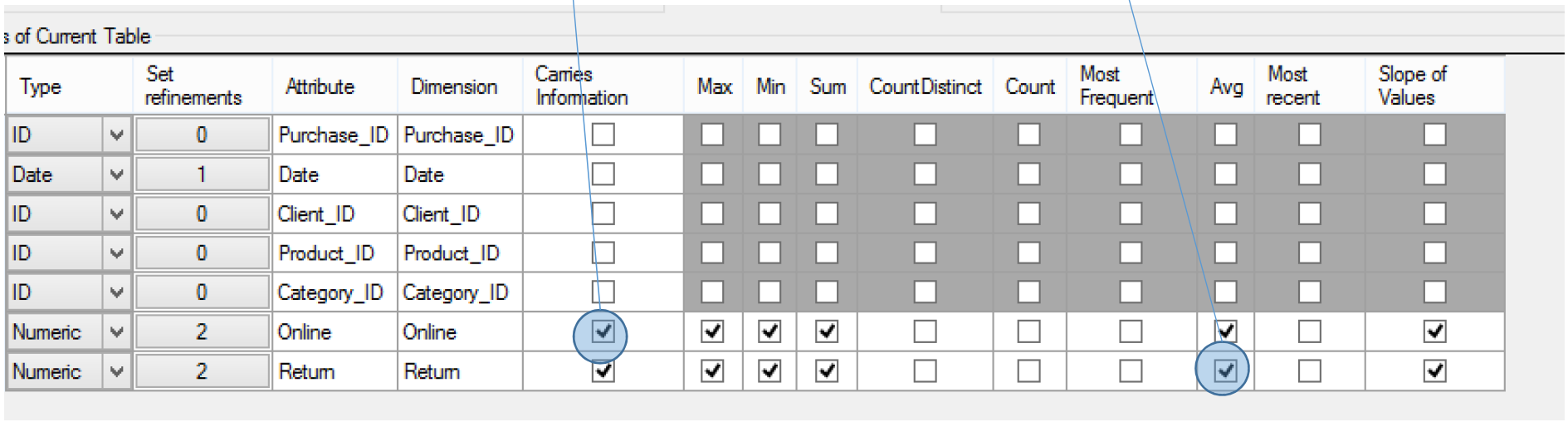
Column name in the csv file

***Dimension:* the unit of measurement. It is used to generate *where* conditions.** For example, if two tables t_1 and t_2 have *price* attributes expressed in dollars, you should set the dimension of both attributes to "dollars". That way, Dataconda will generate conditions like *where* $t_1.price > t_2.price$



Generates attributes and refinements based on this attribute

The aggregator AVG will be used on the attribute Return



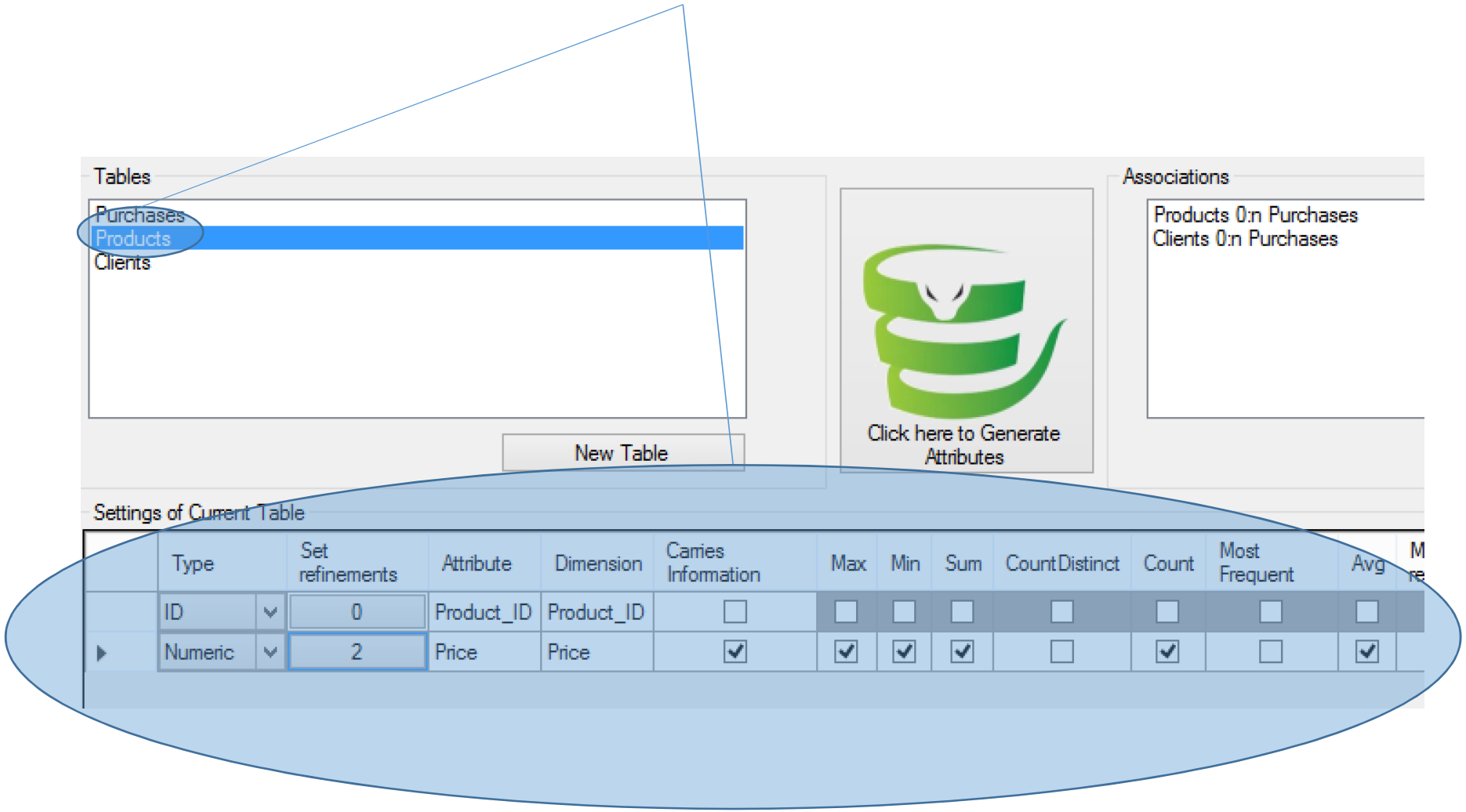
s of Current Table

Type	Set refinements	Attribute	Dimension	Carries Information	Max	Min	Sum	Count Distinct	Count	Most Frequent	Avg	Most recent	Slope of Values
ID	0	Purchase_ID	Purchase_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Date	1	Date	Date	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
ID	0	Client_ID	Client_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
ID	0	Product_ID	Product_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
ID	0	Category_ID	Category_ID	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Numeric	2	Online	Online	☑	☑	☑	☑	☐	☐	☐	☑	☐	☑
Numeric	2	Return	Return	☑	☑	☑	☑	☐	☐	☐	☑	☐	☑

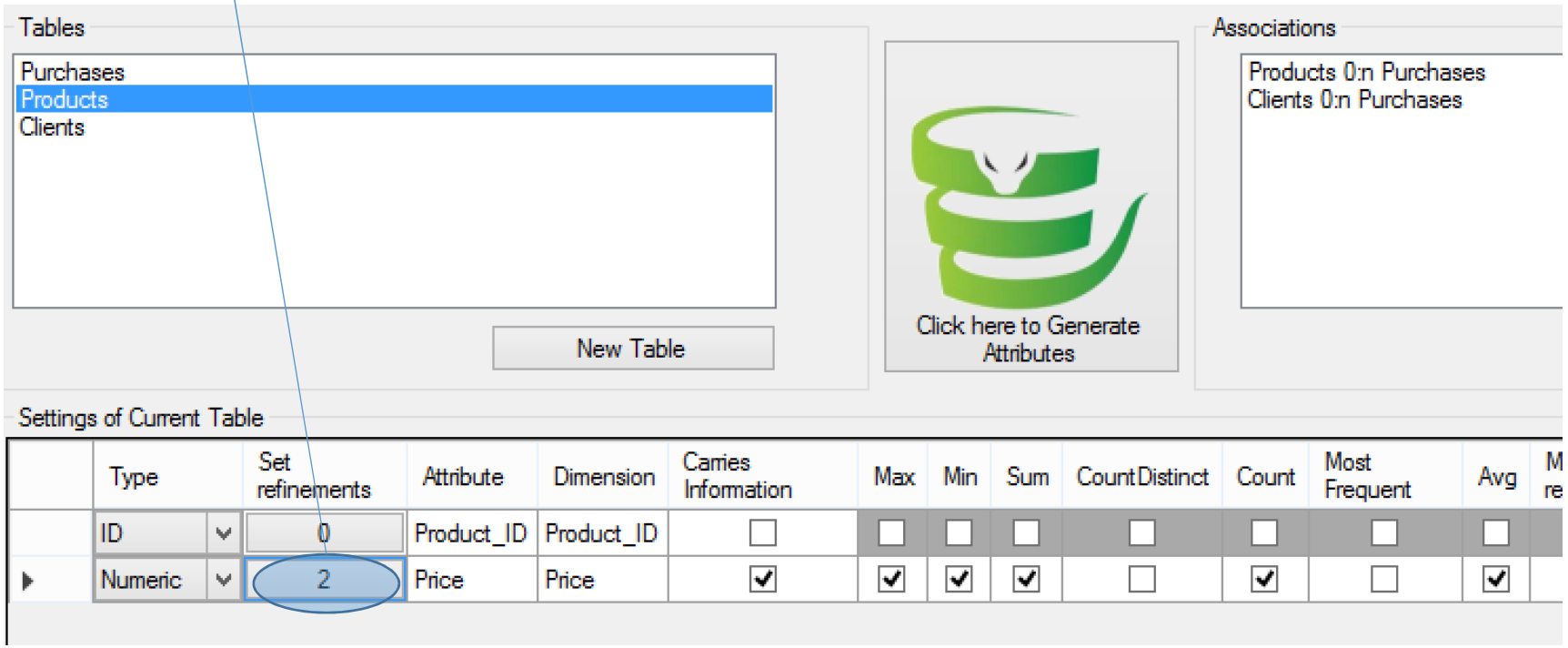
Since Online and Return are 0-1 attributes, it could make sense to define them as categorical or numeric.

Here, we define them as numeric so that the mathematical aggregators (MAX, MIN, AVG, etc) are enabled.

Selecting a different table will show the options for the attributes of that table



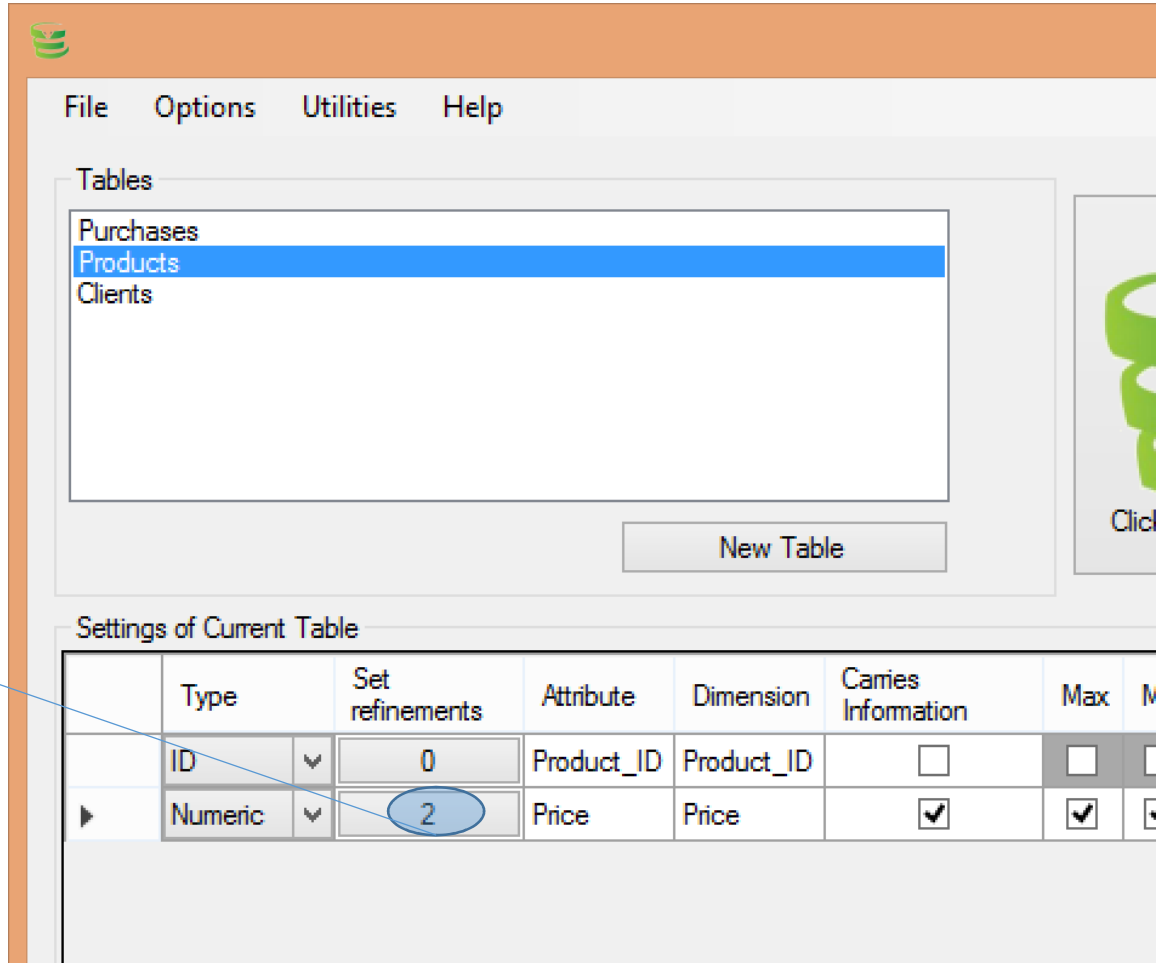
Click to set the refinements for this attribute



Refinements

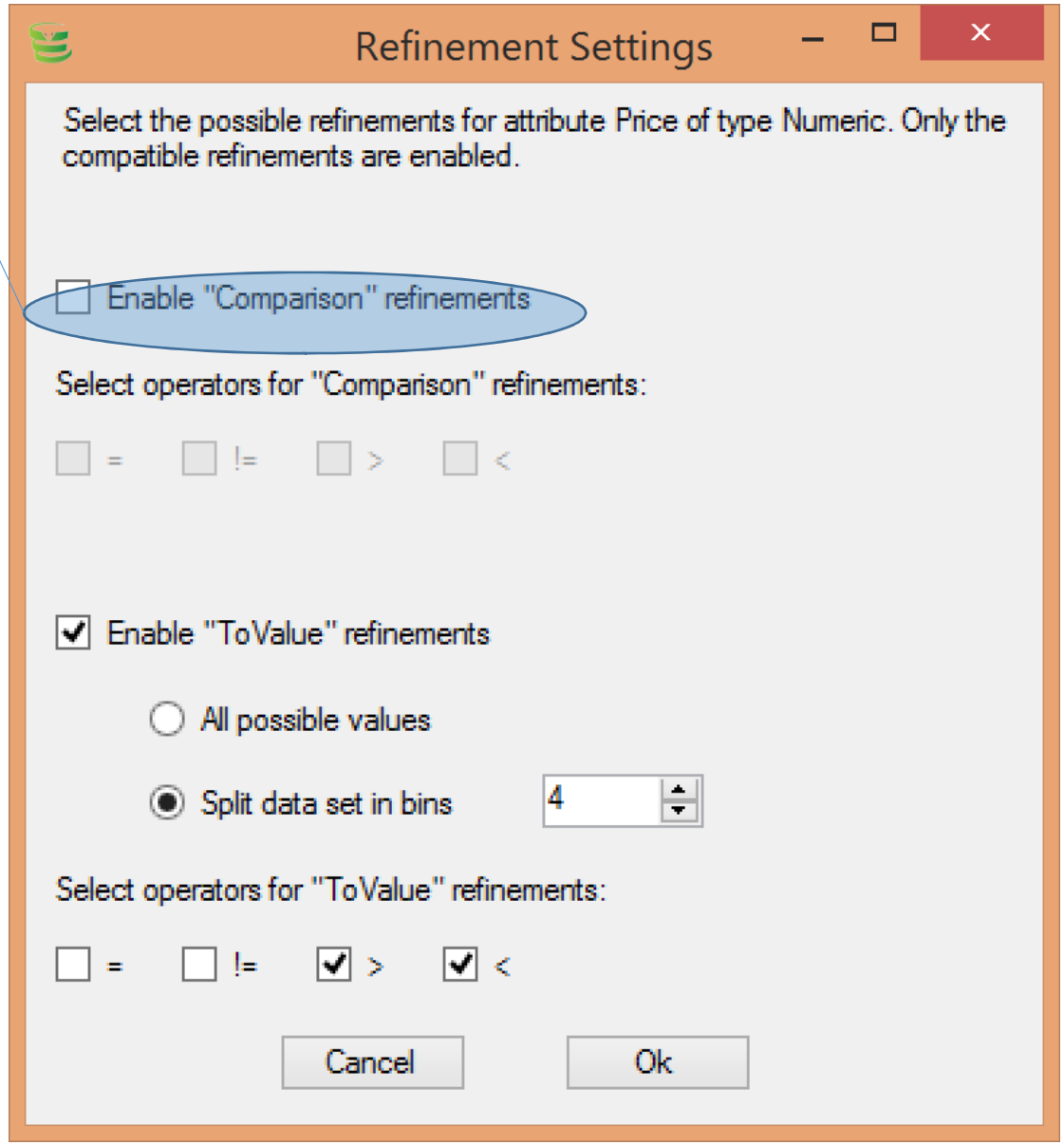
Refinements are the SQL “where” conditions

Click to set refinements



Refinements

Comparison refinements:
If the path considered passes two times through the table products, you might want a refinement like: *“where Products1.price > Products2.price”*

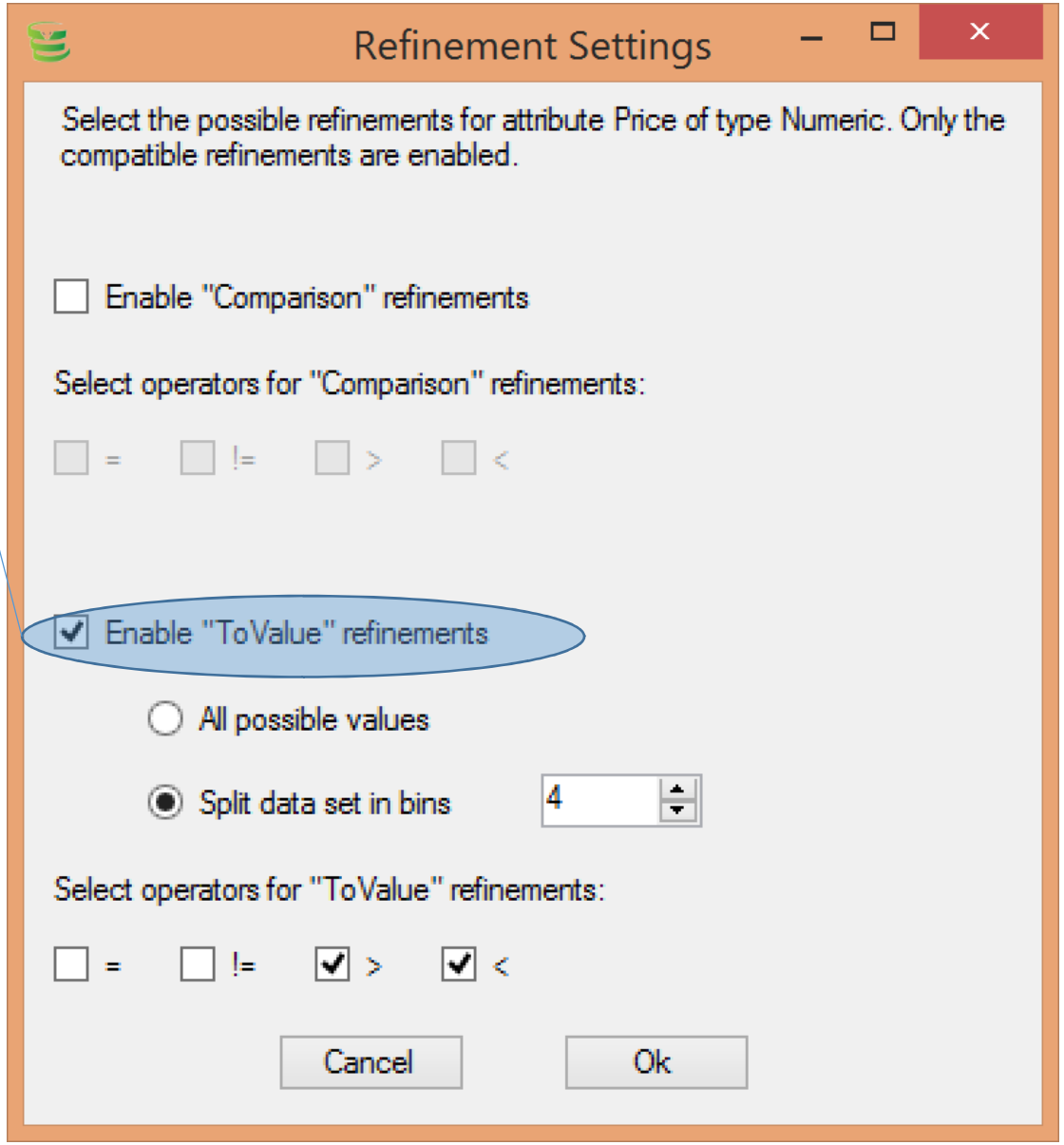


Refinements

ToValue refinements:
You might want a refinement like:
"where Products.price > 200"

By selecting *"All possible values"*, we enable the generation of the refinements:
"where Products.price > val"
(for each distinct value *val* of *Price*)

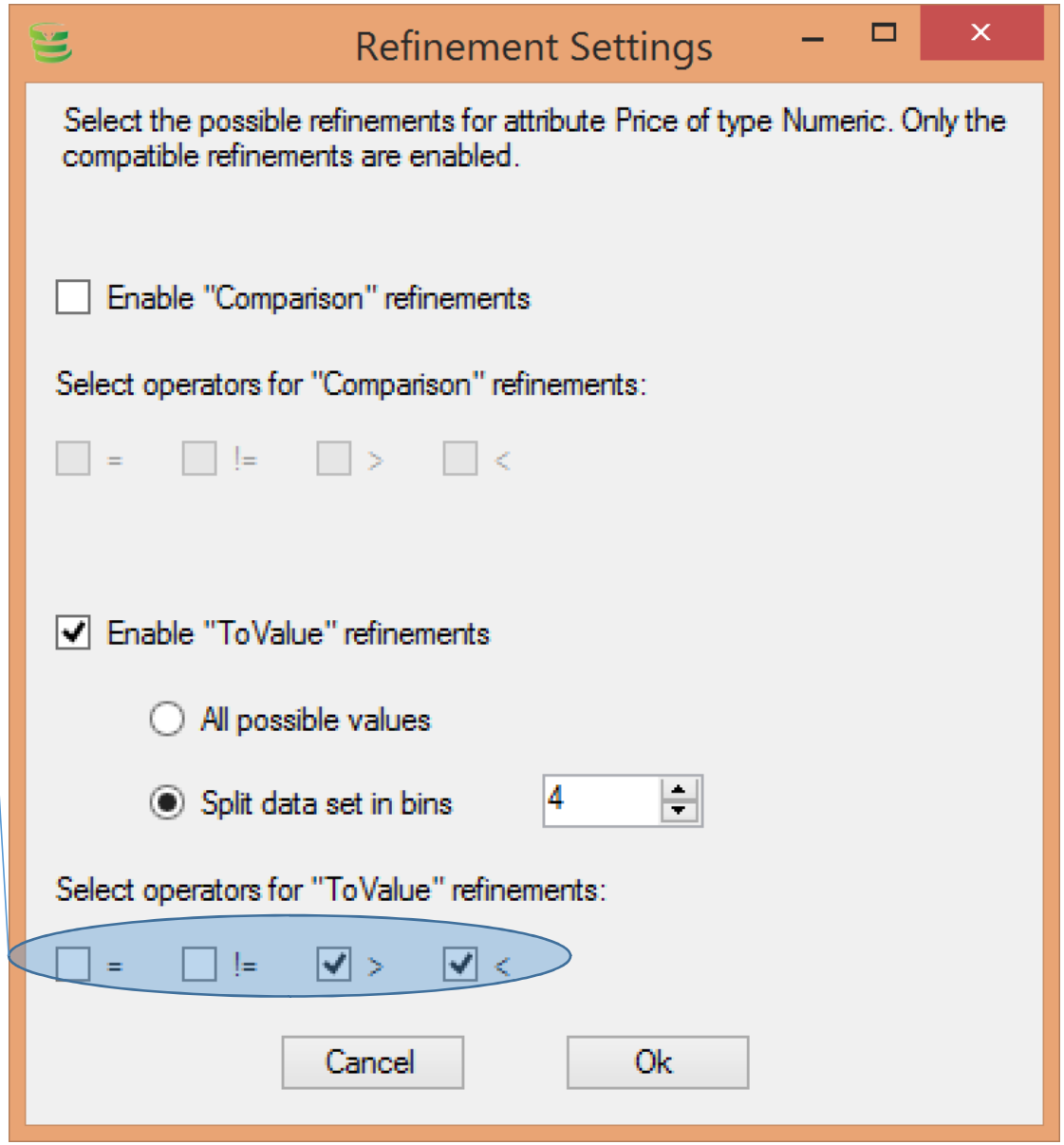
Alternatively, we can split the values of *Price* in bins. The binning performed is by "equal width"



Refinements

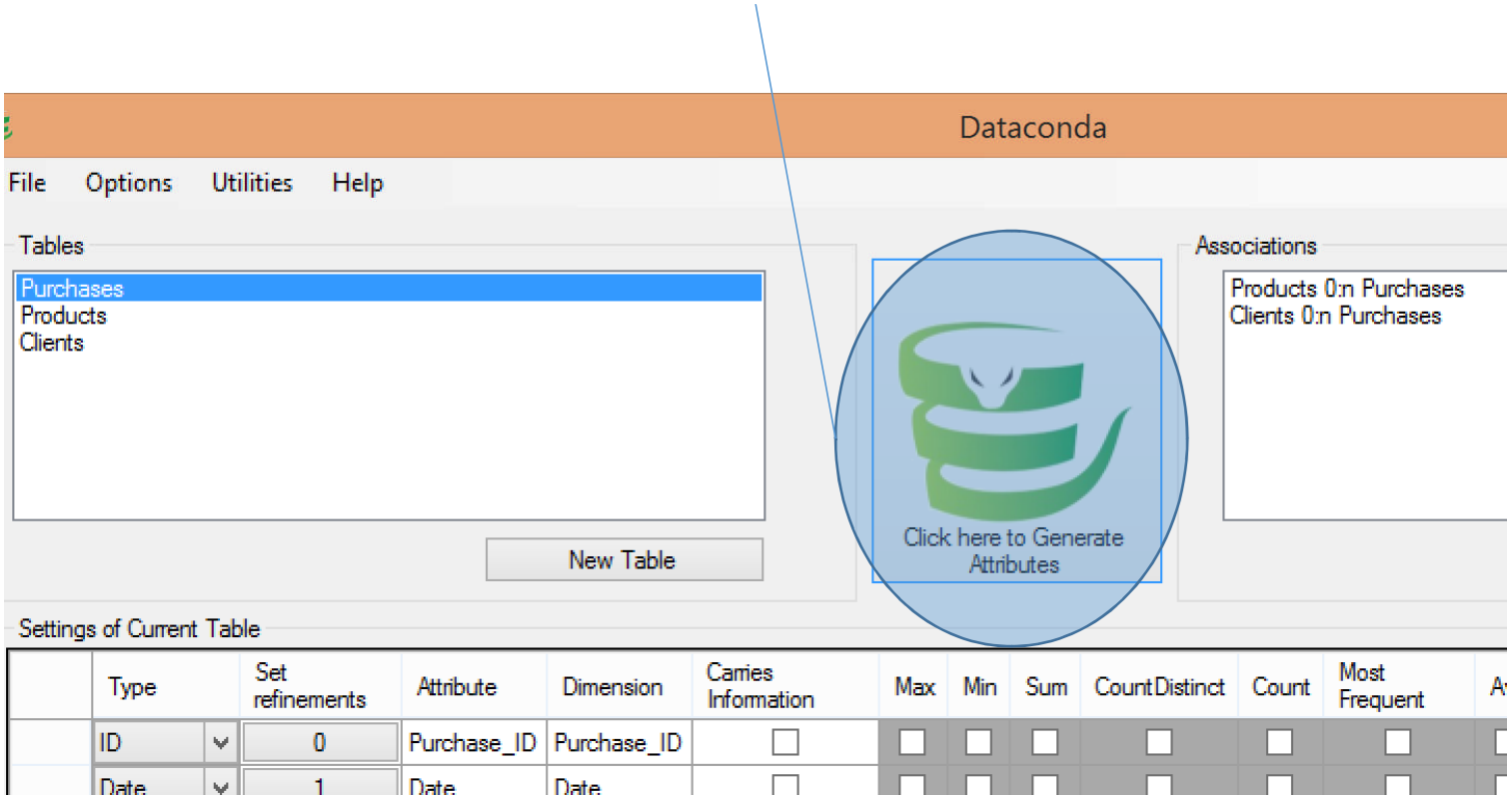
Operators

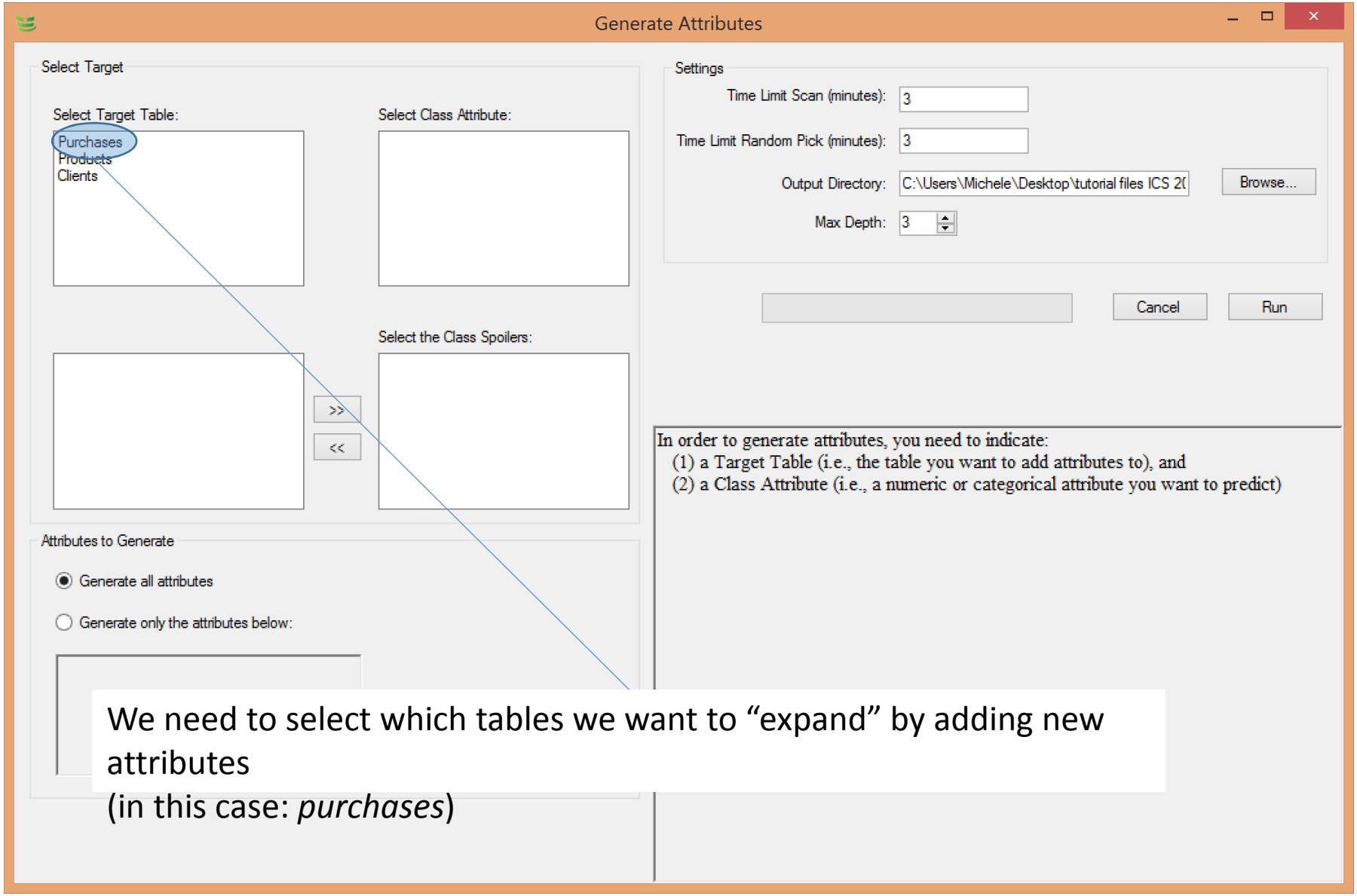
We can select which operators to use in the where condition



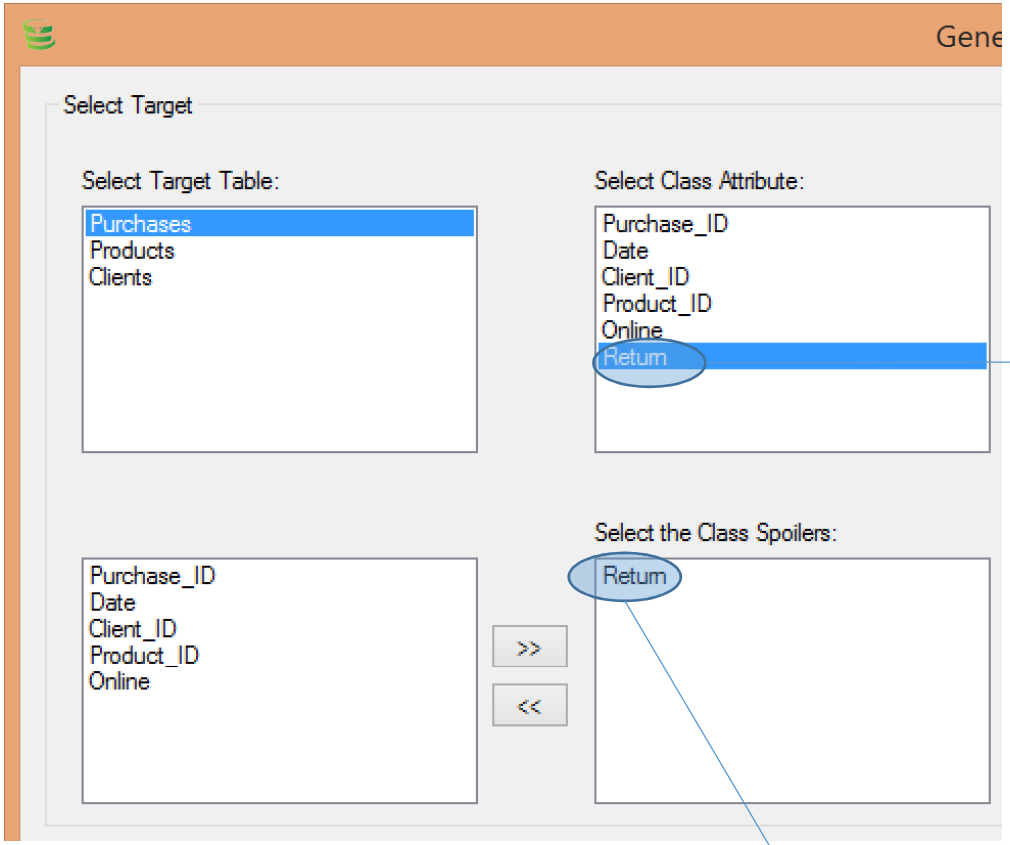
After declaring attribute types, associations, enabled aggregations and refinements,

It’s time to generate the flat table





We need to select which tables we want to "expand" by adding new attributes
(in this case: *purchases*)



We need to select which attribute is the target attribute
(in this case: *return*)

Dataconda places the target attribute among the *class spoilers*

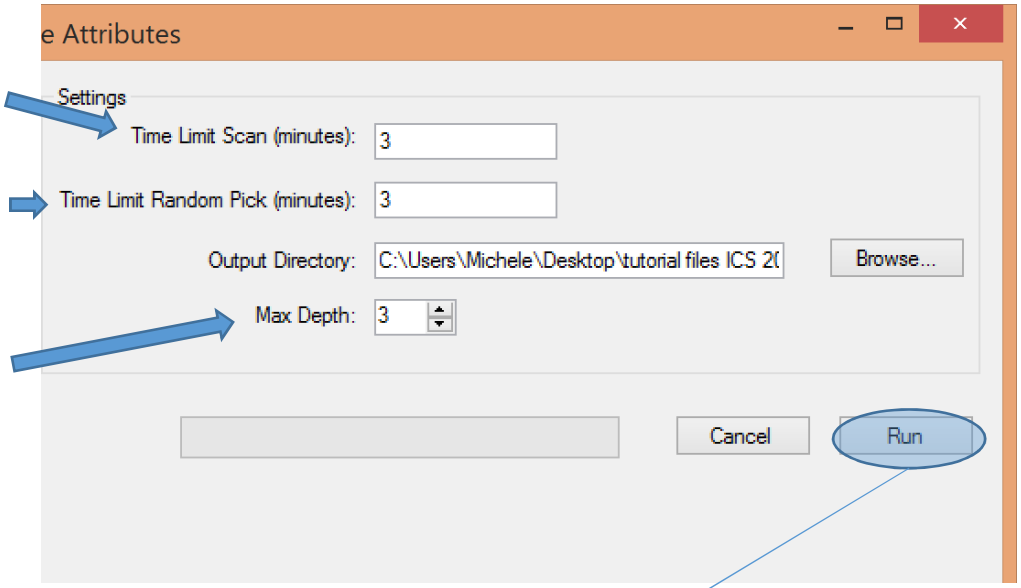
Class spoilers are those attributes that should not be used to predict the target attribute, because the target attribute is functionally dependent on them. If we used them, the classification rule would be trivial.

Clearly, we cannot use the target attribute to predict itself. Otherwise, the classification rule would be "If return = 1, then predict 1". In this case, *return* is the only class spoiler. If the *Purchases* table contained an attribute "ReasonForReturn" (defective product, client doesn't like it, not returned, …), then ReasonForReturn would also be a class spoiler

Time spent generating attributes “in order” (scan time)

When the Scan Time is up, Dataconda starts generating more complex attributes in random order

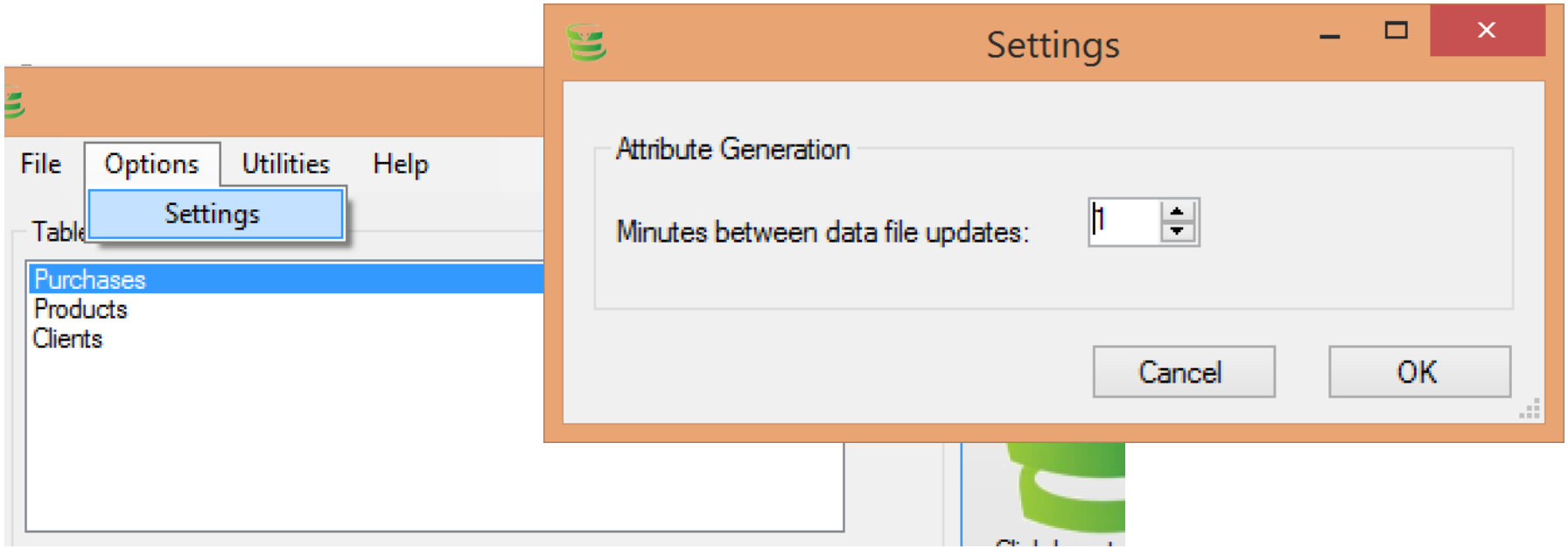
Maximum depth of the paths used to generate attributes (see slide 6)



Time to press *Run*!

Note:
Increasing the max depth will result in many more attributes (good). However, they may also be very complex and hard to interpret

An important option



- If the attribute generation procedure is executed for a long time, you might want an intermediate output once in a while
- Note that too frequent updates slow down the procedure

Data.csv and *Data.arff*

The target attribute is still there

- These files contain the flat table

A	B	C	
A128785228003063964_2_2	Am7334981810602272740_2_2	A2688933330912742441_3_3	A328771(
1	'10'	454	'M'
0	'4'	125	'M'
0	'6'	103	'M'
0	'7'	432	'M'
0	'1'	490	'M'
0	'4'	125	'M'
0	'5'	261	'M'
0	'4'	125	'M'
1	'1'	490	'M'

KS	KT
Am7720857665522209076_4p2_4	'Return'
	1
	1
	1
	1
	1
	1
	1
	1
	1

- Since here the target table is *Purchases*, the flat table will have the same number of rows as *Purchases*…
- …but a lot more attributes! These attributes are new attributes for *Purchases*
- The *arff* file can be opened directly in *Weka*
- The meaning of the generated attributes is reported in *attributes.txt*

Attributes.txt

For example, this attribute is the maximum value of the attribute *Online* among all past purchases relative to the current product.

Practically, its value is 1 if the product was purchased online at least once prior to the current purchase

The description reports the path on which this attribute was built – in this case: purchases → products → purchases

```
Am7318961810941188030_4p1_4:
Numeric,Online
DESCRIPTION: Max(Online) among past Purchases of Products
0:Target->Max(Online)
1:Purchases->Max(Online)
2:Products=>Max(Online) where Date LessThan 1:Date
3:Purchases.Online
```

```
Am8341016844213767212_4p1_4:
Numeric,Online
DESCRIPTION: Min(Online) among past Purchases of Products
0:Target->Min(Online)
1:Purchases->Min(Online)
2:Products=>Min(Online) where Date LessThan 1:Date
3:Purchases.Online
```

```
A7468550537736559003_4p1_4:
Numeric,Online
```

Analysis output.txt

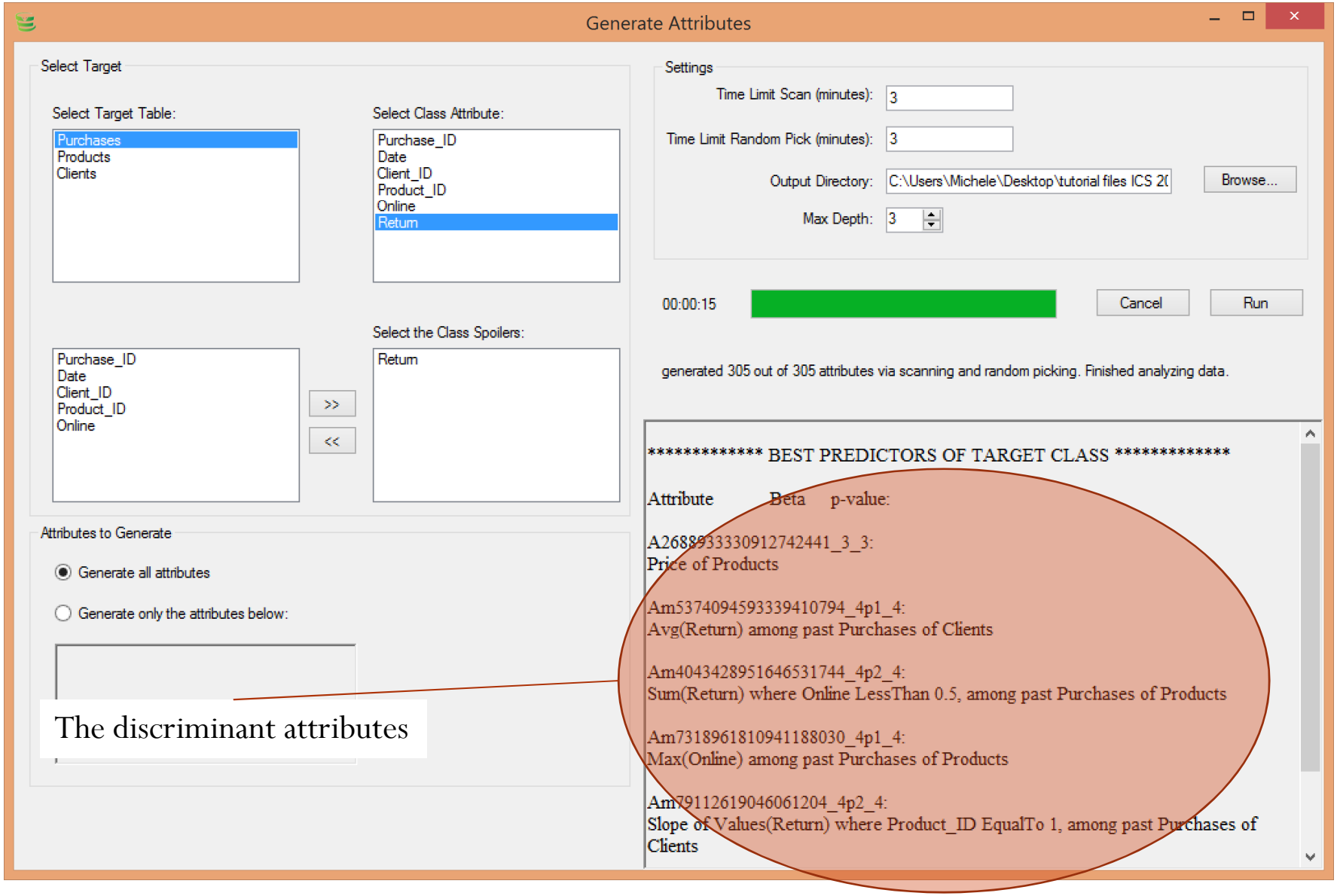
- After generating the attributes, if R is installed and if the output folder contains a file *Rtemplate.R*, *Dataconda* executes an attribute selection procedure in order to find the best predictors

price →
Client's return rate →

```
Coefficients: (4 not defined because of singularities)
                                 Estimate Std. Error z value Pr(>|z|)
(Intercept)                     1.605e+02  7.620e+01   2.106   0.0352 *
A2688933330912742441_3_3        1.580e-02  2.437e-03   6.482 9.06e-11 ***
Am5814594725006700583_4p2_4    -1.608e+00  1.149e+00  -1.400   0.1615
Am5374094593339410794_4p1_4     3.349e+00  8.027e-01   4.172 3.01e-05 ***
```

- The *price* has a significant (***) impact on *return*. The correlation is positive (because its coefficient 1.580e-02 is greater than 0)
- The *client's return rate* has also a significant (***) impact on *return*. The correlation is positive (because its coefficient 3.349 is greater than 0)

The results are also reported in the window



Important

- When preparing the csv file, make sure that the rows are "*order by date asc*"
- When saving/loading the project, remember to place it in the same folder as the csv files
- Do not keep the data.csv file open in Excel when executing the attribute generation procedure. Otherwise, Dataconda will give an error.

Two ways to extend Dataconda

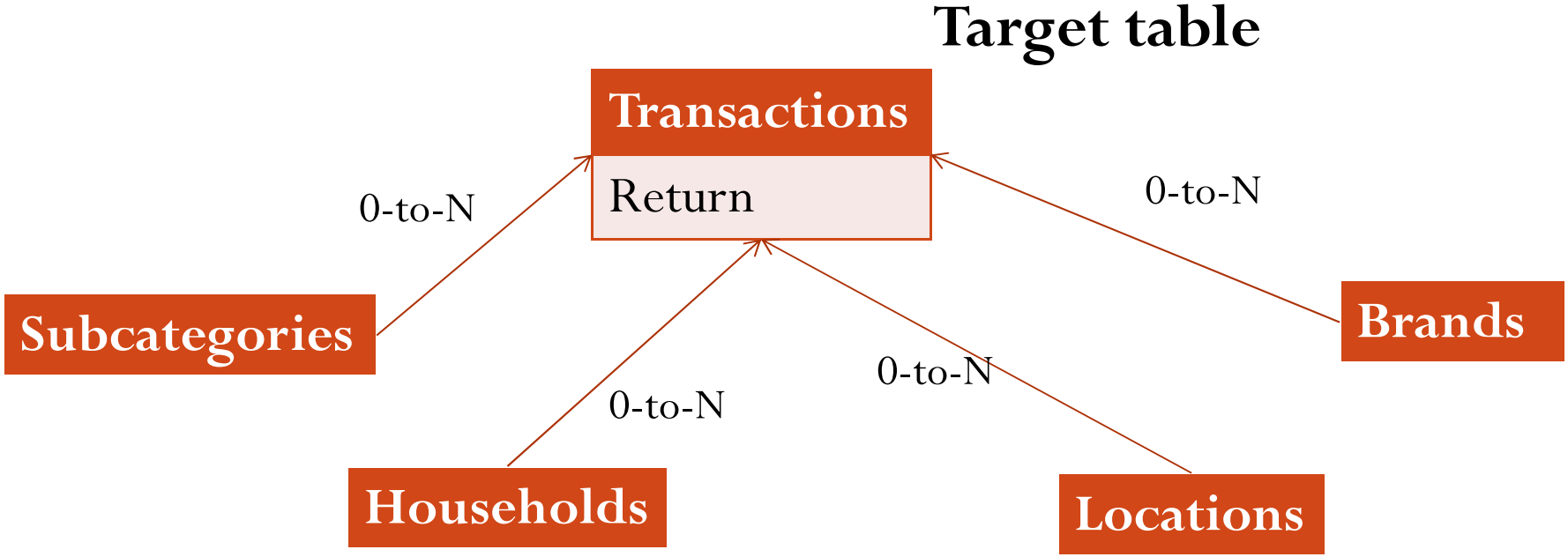
1. **Write attribute selection procedure (in *R*)**
   - After generating attributes, Dataconda executes the file Rscript.R in the folder
   - The default attribute selection procedure is based on Lasso

2. **Write aggregating functions (in *.NET*)**
   - Extend the interface *dataconda.core.IAggregatingFunction*
   - Place the dll in the program folder of Dataconda
   - The new function will appear in the GUI

Some Experiments

Real Data (ISMS Dataset)

- ISMS Dataset: all transactions of 1% of the registered customers of Circuit City. In total: 173,262 transactions

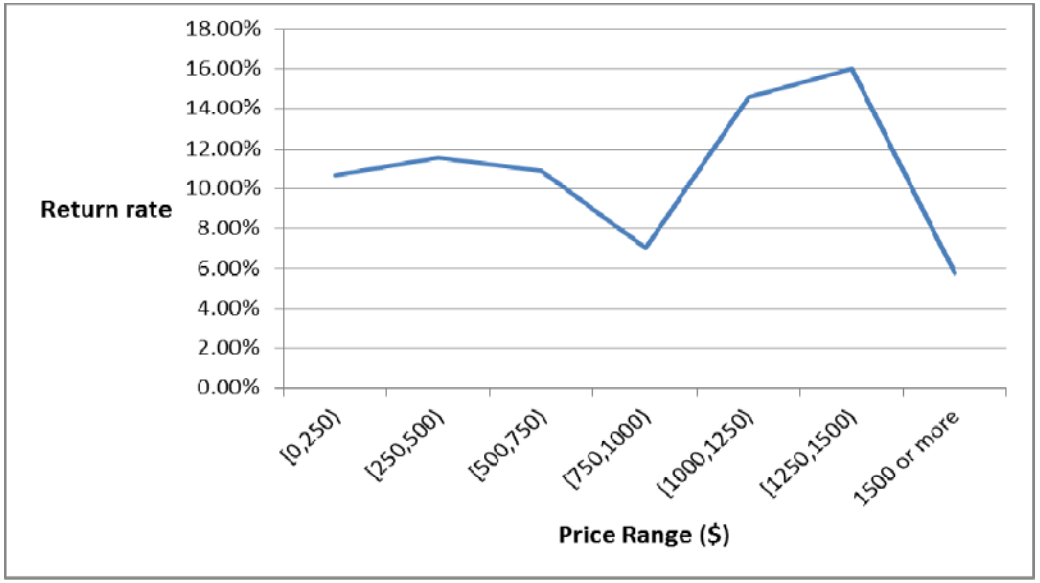


- Too large!
- Sample 1,000 customers and retain all their transactions
- End up with a target table with ~6,000 rows
- Generated 2,496 attributes up to depth 4 in 1h 21

Attribute 1 (β = .53, pval < 0.01)

- Max(RETURN_BINARY) where Price $\leq$ \$1,500, among past TRANSACTIONS of HOUSEHOLDS

Att value	Prob return	Count
0	11.5%	39713
1	34.0%	45030
NULL	11.8%	37789



Attribute 2 (β = -.45, pval < 0.01)

- Max(INCOME of HOUSEHOLDS) where INCOME ≥ 7.4, among past TRANSACTIONS of BRANDS

Attr Value	Prob return	Count
8	24.2%	1135
9	19.8%	118664

Conclusion

- Relational attribute generation:
  - Underdeveloped field with high potential
    - Find new knowledge
    - Increase classification accuracy
- Dataconda 
  - www.dataconda.net
  - Full version is free for research and teaching purposes
  - Can be extended:
    - Add new aggregating functions
    - Change the attribute selection procedure
  - Can be improved:
    - Integrate it with a DBMS
- Thank you very much!

Contact: samorani@ualberta.ca