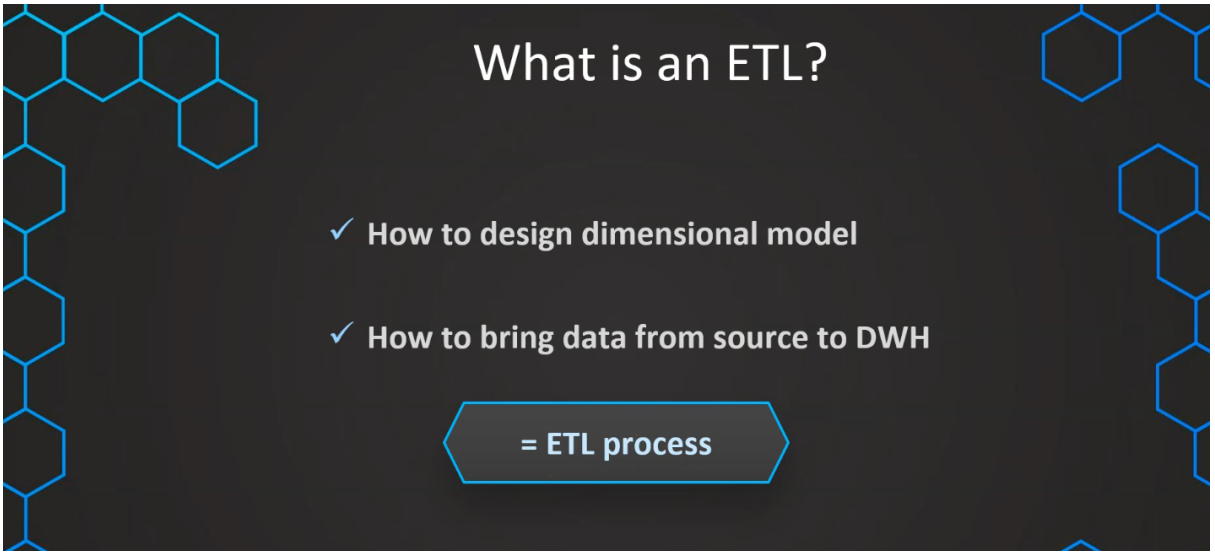
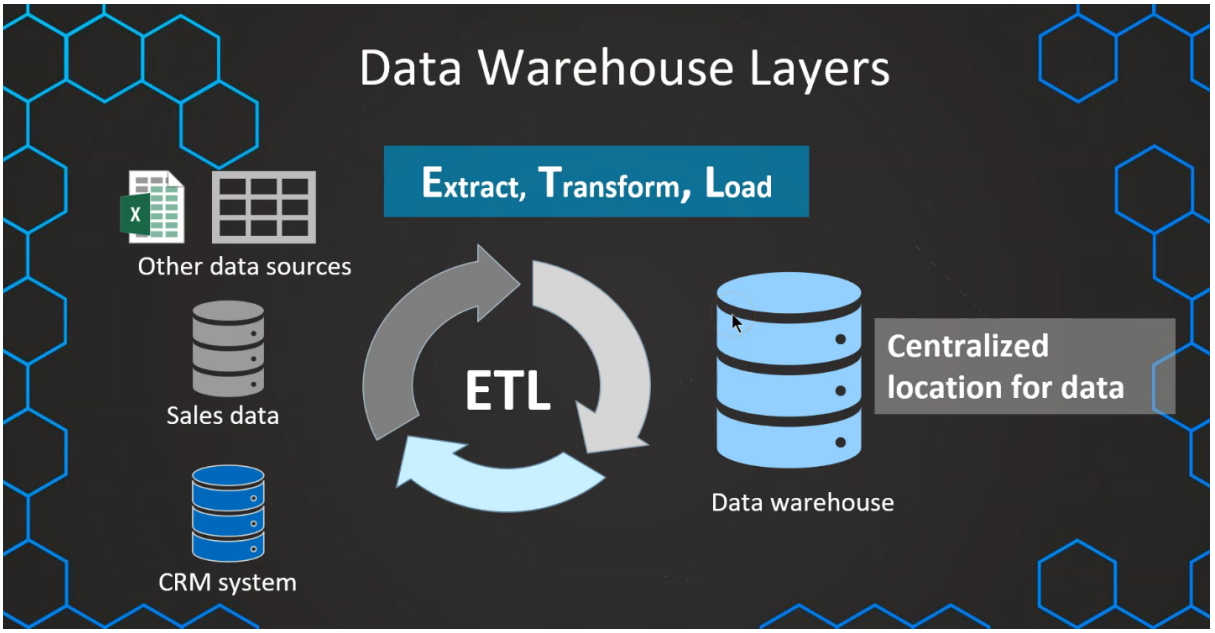
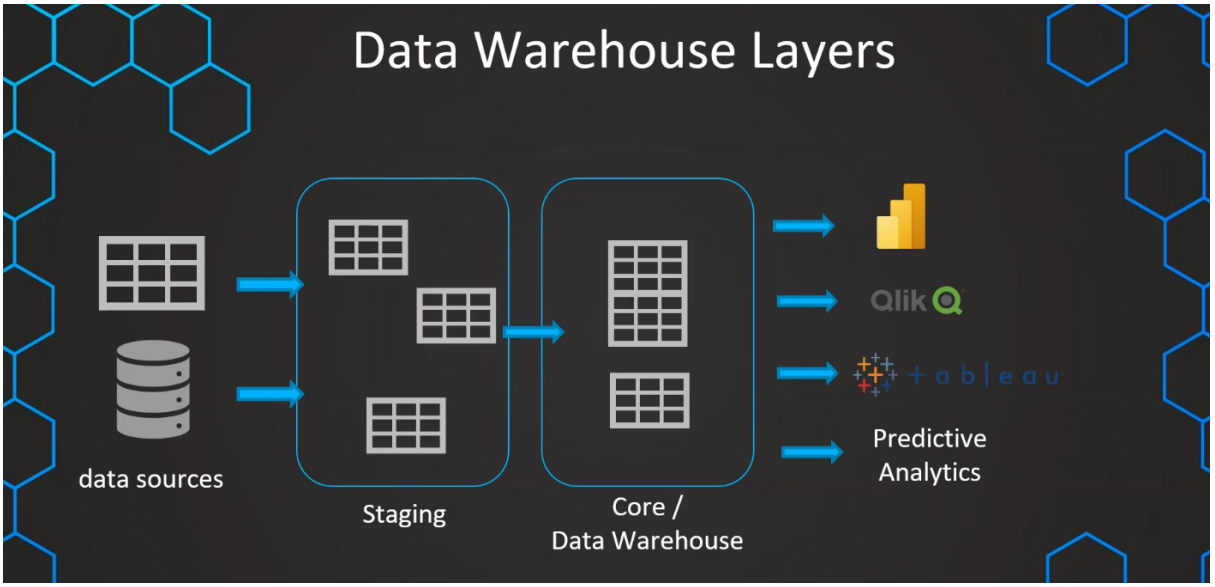
9a. ETL Process



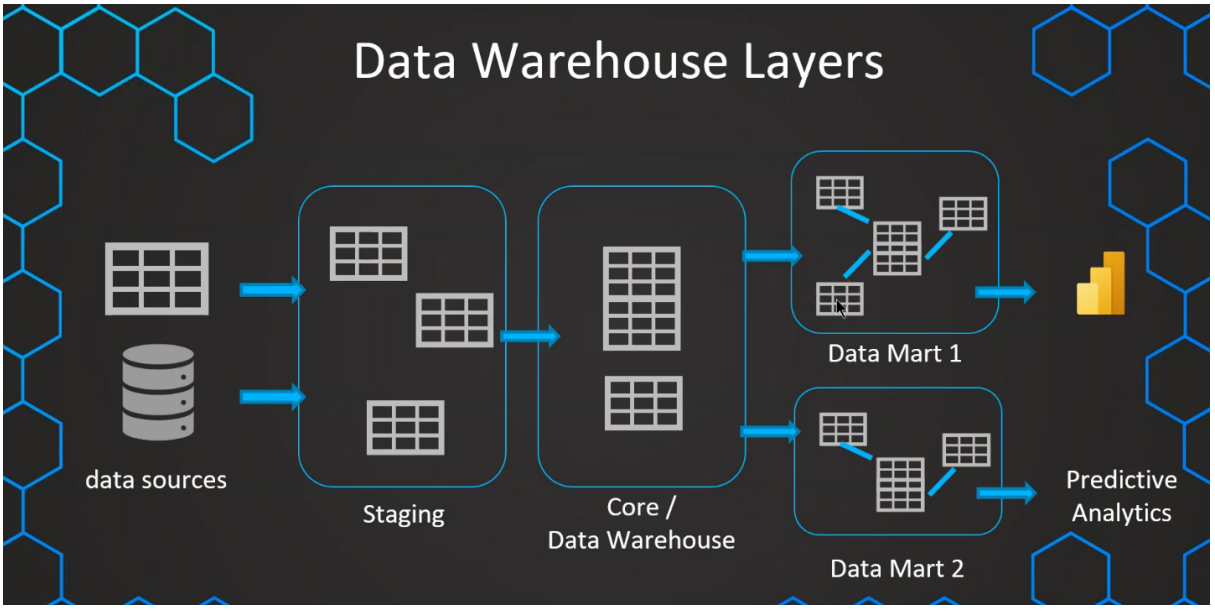
- we are ready to bring in the data from our sources and model that with our ETL and bring it into our data warehouse. And that is basically what our ETL process is doing.
- And now let's quickly revisit also what we have already learned about the ETL process.
- So again, we have learned that there are data sources, and, of course, our data for our data warehouse can come from all of these different sources. And they need to be integrated. They need to be cleaned. They need to be transformed and integrated in our data warehouse. And all of that we do with this ETL tool.
- And then, of course, this ETL tool is using, or basically, this is structured in extracting the data first from the data sources. Then we need to transform, that's what we've said. And then in the end, of course, load it into our centralized location, so our final data warehouse.



- And we have seen also that there are different layers in our data warehouse. So we have learned that from the data sources, we first extract the data into our staging layer. And this is now the place where we have all of the data in tables. Maybe we have done some minor cleaning. We have extracted all of the relevant information.
- And then during the load from staging to the core we are transforming the data. So this is basically where now our dimensional modeling is taking place.
- So we use the staging data, transform it, and this transformed data will be loaded into the core. And then from there we can use the data with all of the different applications.

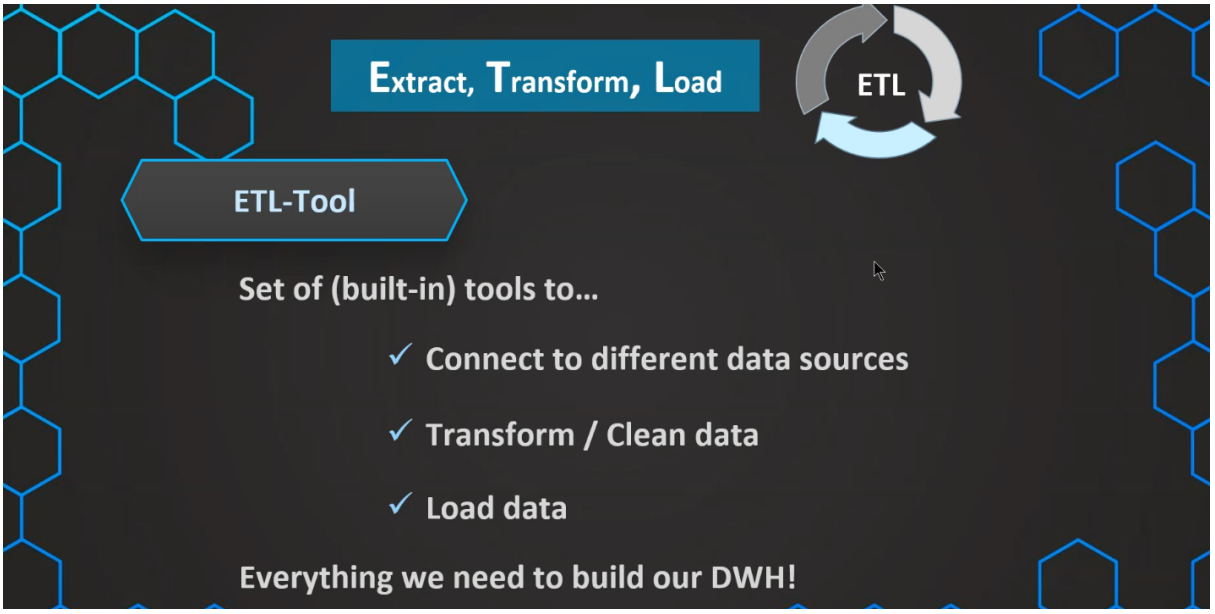


- And of course, we've also learned that in most cases it makes sense to have one data mart for one specific use case. And like that, we can tailor the data mart more specifically to a given use case, which, again, improves the value of the data that we have in our data warehouse.

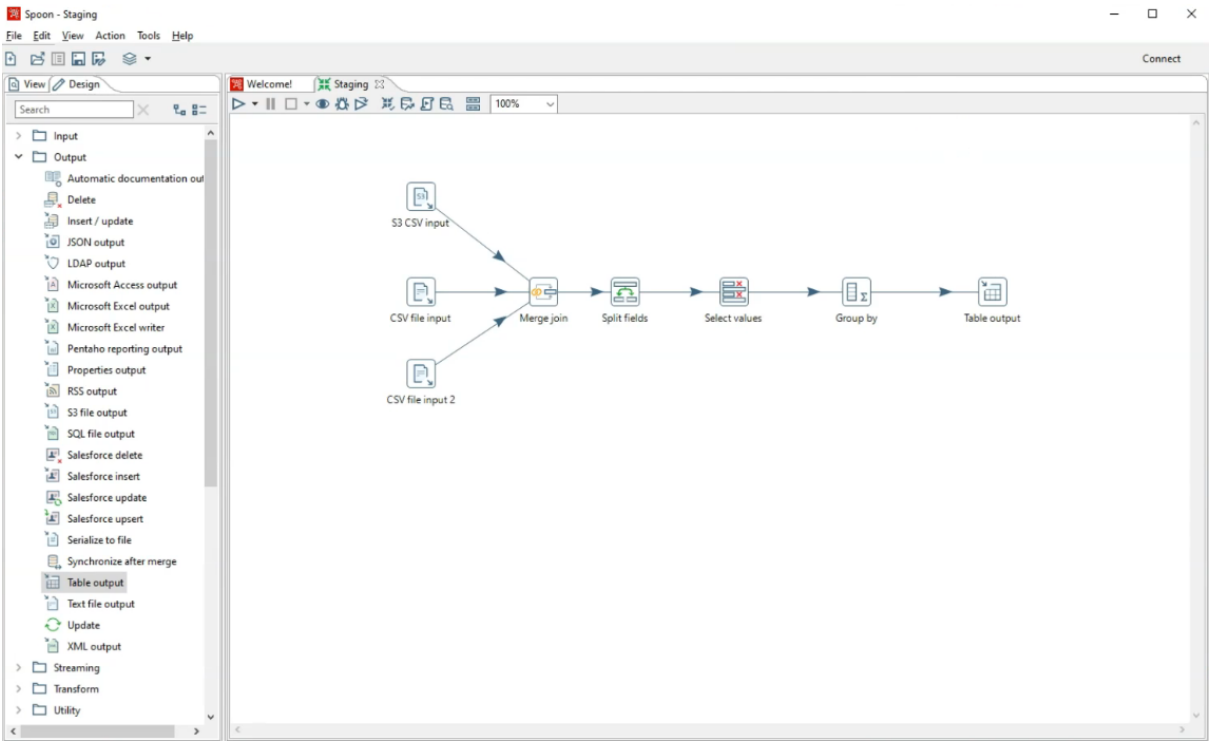


- Because now it's, again, easier to use. The performance is, again, even better.
- But now how is this implemented in practice? Our ETL is, of course, done using so-called ETL tools.
- And an ETL tool, there are many out there. And also later on we'll discuss some of these important tools. And they are, in the end, just a set of built-in tools that we can use to
- First, of course, connect to different data sources. This is where the extract part is important. So we need to extract the data from different data sources.
- Then, of course, we have many built-in tools to transform our data, to change data types, to add additional columns, to clean our data, to model our data, so to restructure it.
- And this is basically the main part in our ETL tools and in our ETL process. And, of course, then we have, again, a set of tools to write the data back in different formats.

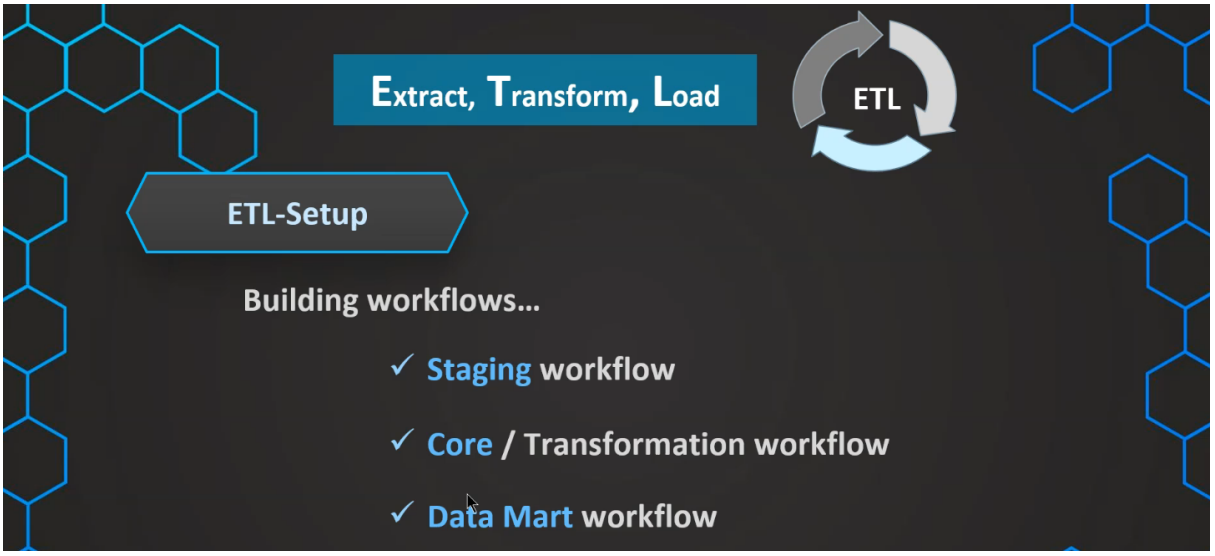
- So, of course, we are interested in writing the data back into databases, so our data warehouse.



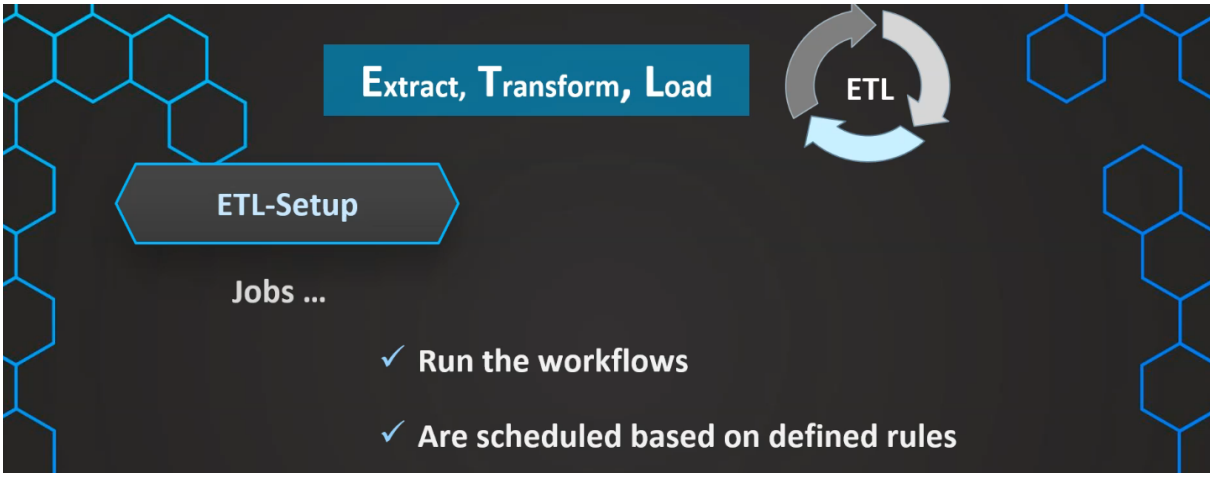
- But theoretically, of course, there are many, many more options. And these ETL tools usually have thousands of different tools and possibilities where we can do things. And we are usually only using just a very small subset of that.
- But in our case, we have said that we are interested in loading the data in a data warehouse.
- So basically this is all that we need to build our data warehouse. And now with this ETL tool, we are usually setting this up in a way that we have different workflows.
- So you understand that we have these different phases, these different layers, and accordingly, we build separate workflows.
- So we have one workflow, this is for the staging. And if you have a look into the ETL tools, this is basically how a workflow can look like.



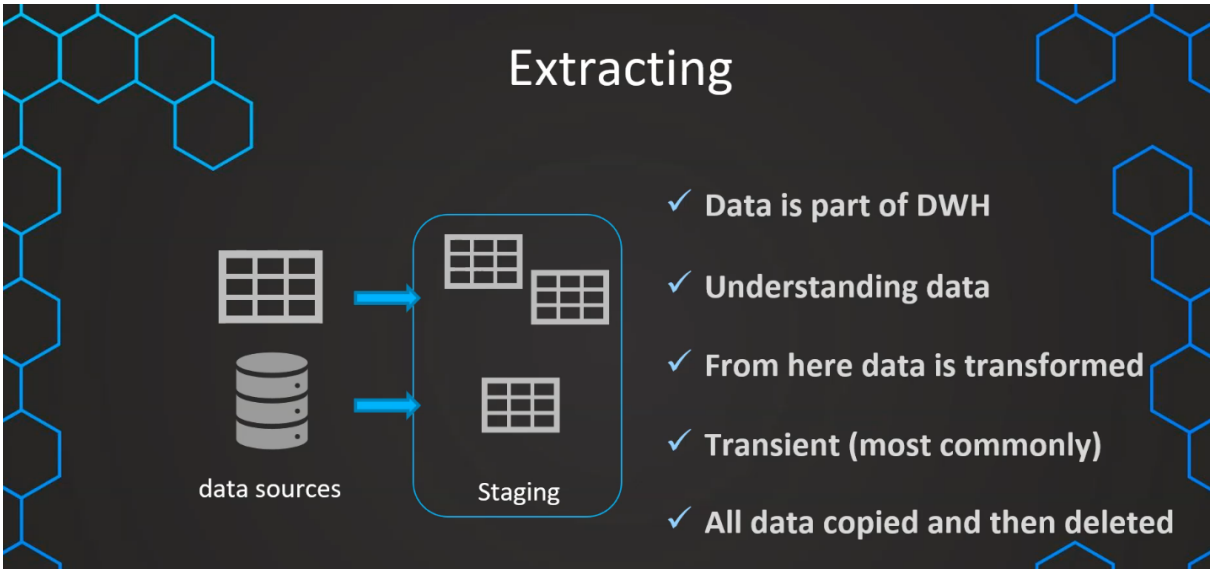
- So we just have these drag and drop built-in tools that we can drag.
- So we can enter some credentials, we can enter some connection details, and then just some specific data will be extracted.
- We can then do all of our transformations. So this is a workflow that can be set up.
- And then in the end, of course, we can write that again into our tables.
- So this could be one workflow example for a staging layer. And also then we have, usually, accordingly, a staging schema in our database.
- It could be also possible that we have for the staging, for each of the different layers, one separate database.
- But usually this is handled via different schemas.
- So, of course, we have the staging workflow, then we have for the core, and also usually for the data mart, a workflow. Again, this is not set in stone, but this is just a common default strategy that is used.



- And then once we have set up these workflows, of course, things are also scheduled using jobs.
- So again, once we have created and set up all of these workflows, they are used now to do the data cleaning, the data extraction, and all of the processes. And those jobs are now used to run these workflows based on defined rules, so on specific times in specific frequencies.
- So this is the broad overview of how our ETL is set up. And now that we have understood that, we, of course, want to dive deeper into the different steps, the different processes that we are doing in our ETL process.

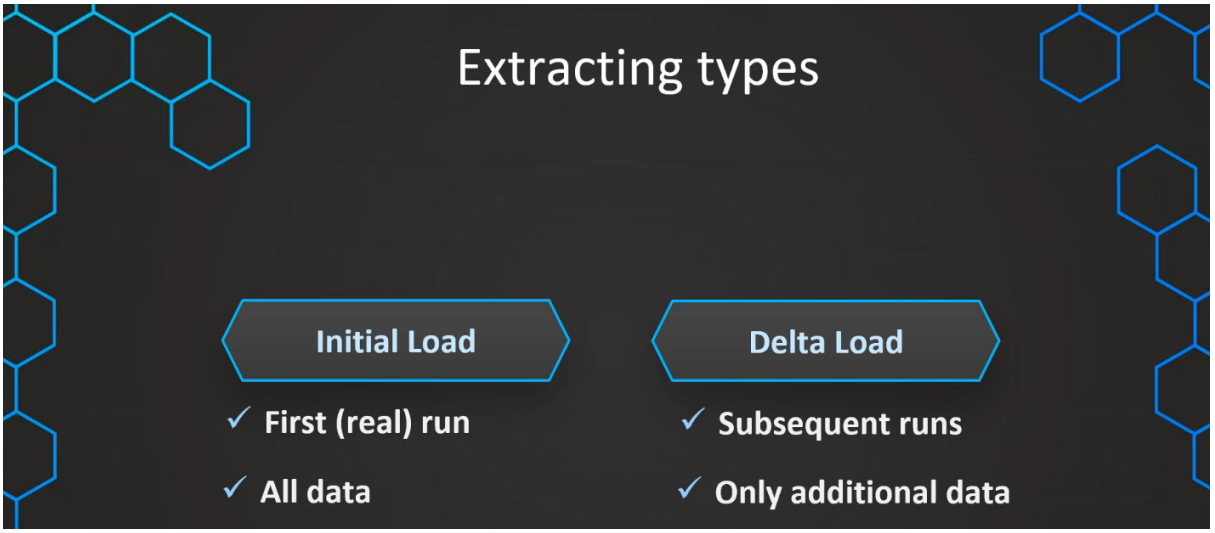


Extract



- We have already seen that we need to extract the data from the data sources into our staging layer.
- And once we have done that, the data is now already part of our data warehouse in the staging layer.
- And of course, we do that to not put any unnecessary load on this source systems. So of course, they are production systems and we cannot risk to slow them down and we cannot use that to just work on the data, understand the data, and plan our transformational steps.
- Therefore, we need to have the data in our staging environment.
- And of course, now this is also the place where all of the data is finally available in SQL tables. So that's why we need to bring it into this staging layer.
- And from here we can now do our transformations and we can plan them out and load that with these transformations into the next layer.
- And also remember that the typical type of staging layer is of the **transient** type. This is the most common type and that means that all of the data once it has been copied from the staging layer with the transformations will be deleted or truncated from the staging layer.
- And then until the next run will be empty till new data is been loaded into the staging layer. And then only this new data will again be then copied into the next layer, which is the core layer. So this is the common type, the transient type.

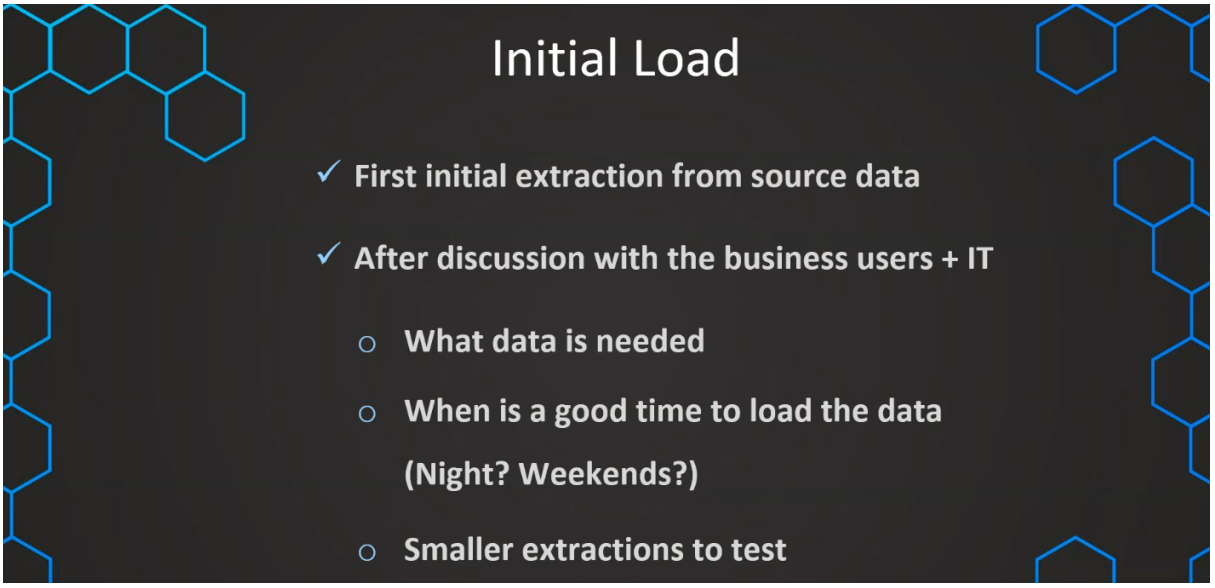
- Of course, we can also in some cases have the permanent type, but we are focusing on the transient type because this is the common type.
- so these are the fundamentals about the extracting of the data.



- We have now two different types. So first, we have the initial load and then also we have the so-called delta load.
- So of course the initial load, as the name suggests is the first real run of our ETL.
- So of course we can do some testings before, some small extractions.
- But the first time when we want to load, all of the relevant data, this is called the initial load. And then of course we also have the delta load which is the subsequent load.
- And now we don't load all of the data anymore, but only the additional data that has been occurred in the source system. So only the new data.

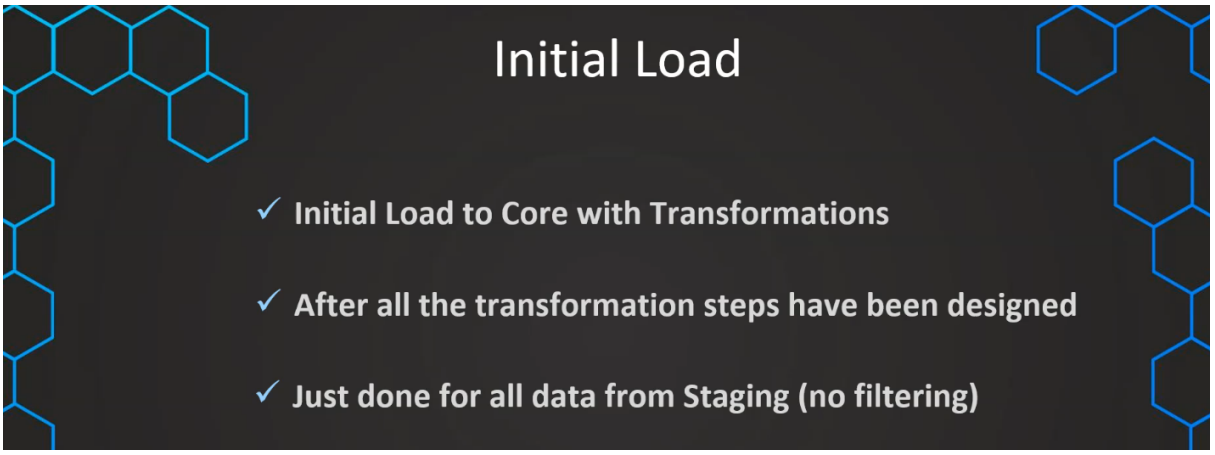
Initial Load

- now we want to start with the initial load of our data. So how does this work?



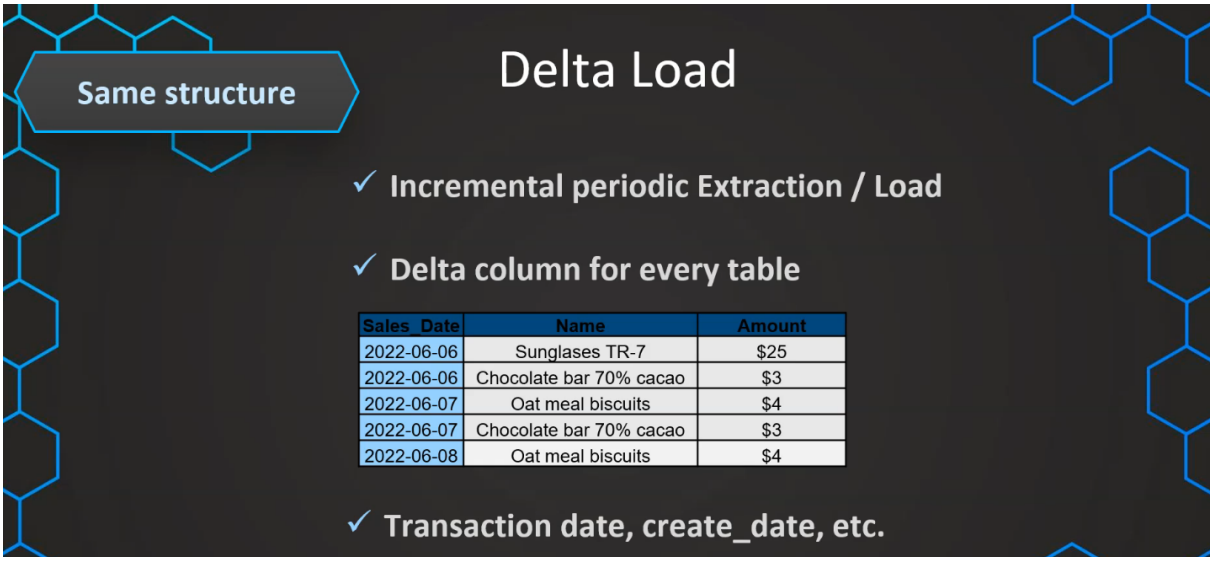
- Of course, first, we need to extract in the first step all of the initial data from the source system.
- So this is what we call Initial Load. And we have to keep in mind that this is usually taking place after some discussions with both the business users and the IT responsibles.
- So usually there are some business users that are using the reports and doing the reporting, and then there are also some core IT responsible people that are administrating the source systems or the databases, so they are more of the technical responsibles.
- And we usually have a discussion with both of them what data we can extract, how the data is structured, and what data is needed by the business users.
- And once we have decided what data we need, we also of course need to discuss what is now a good time for that initial load.
- So of course, that initial load, since we load all of the data all at once, takes a lot of time and puts the highest burden or the highest load on the source systems.
- Therefore, we also need to discuss, since these are productive systems, what should be a good time. So usually, sometime when the company is not operating, so maybe during the night, or if it's taking more time, it can be also during the weekend, during Sundays.

- And this, of course, needs to be discussed with those responsibles so that we are sure that we are not slowing these productive systems unnecessarily down, and therefore, we need to discuss it with them,
- and of course, we can also make some smaller extractions just to test how much time is needed for these initial loads, so that we have some estimate and we can tell them it probably takes three hours, when is a good period during the week when we can load that data.
- So this is then the most crucial moment for this extraction of the data from the source systems.



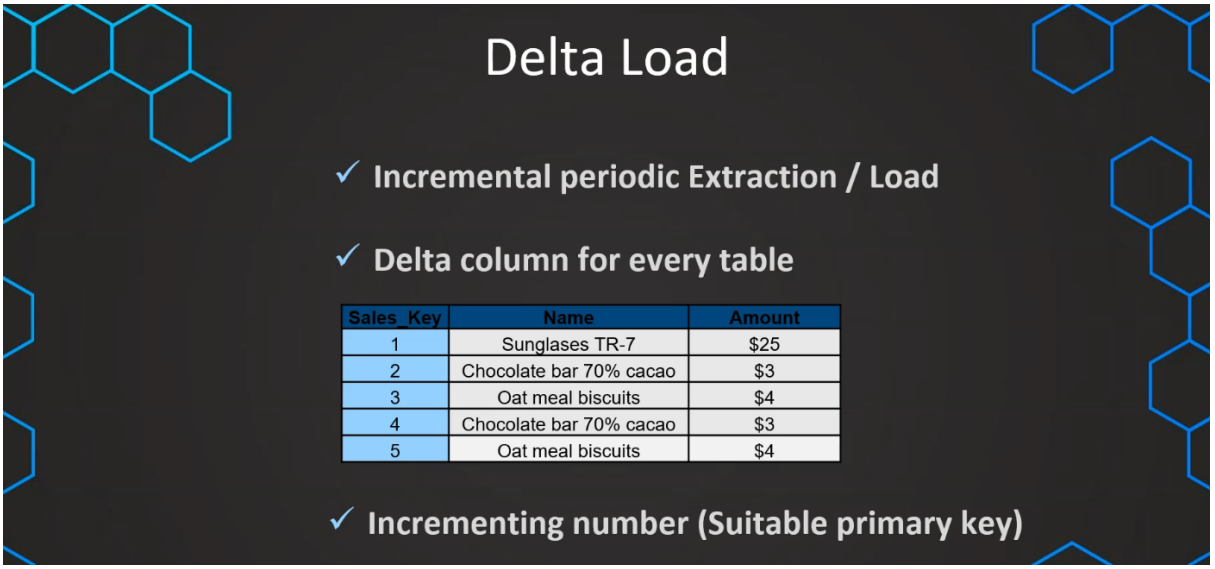
- But of course, then once we have the data in the staging layer, we also have some initial load to the core layer where all of the transformations will be applied.
- And this of course, happens then after we have designed and planned and tested all of the transformation steps we have designed in our ETL tool.
- So this is basically then the second step of the initial load, and in this case, we just again, copy all of the data from staging into the core layer.
- So this is the initial load, and then there is also the second type, which is the Delta Load. And this delta load is now for the next round. So we need to make sure that now we only load the new data

Delta load

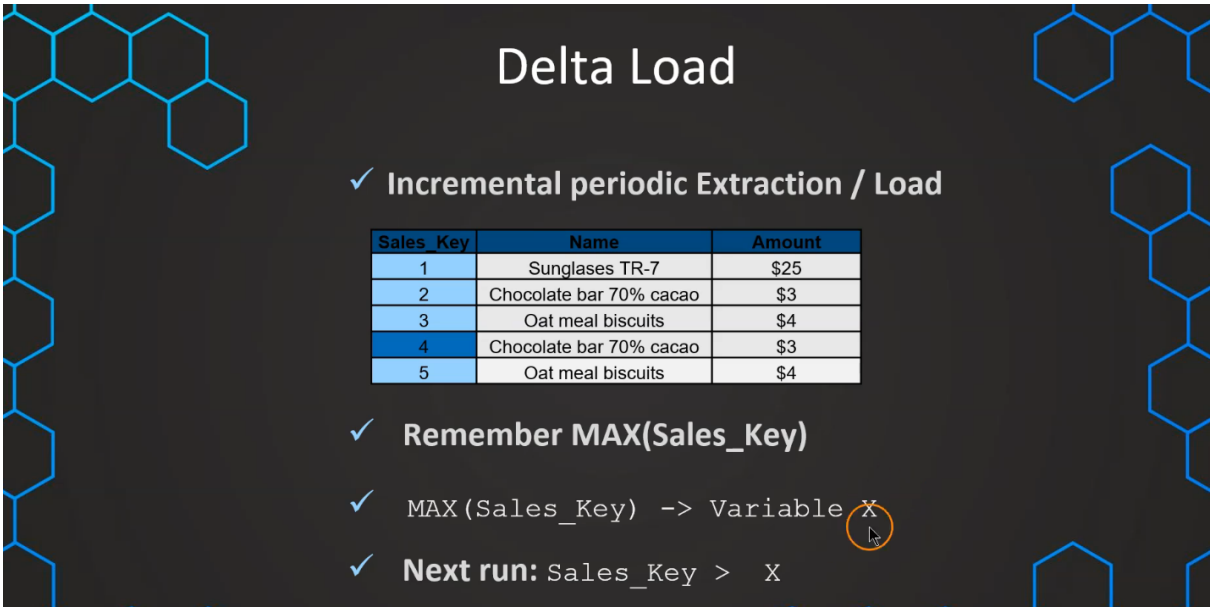


- Now that we have understood the initial load, we want to step things up and understand how now, once we have loaded all of that data in the initial load, we can now also implement the so-called Delta load.
- So again, the Delta load is this process where we are now incrementally loading the data, and we do this now on a regular basis.
- So we set a frequency maybe once per day, maybe every night. We load now the new data from the source system that we have not loaded before, and we of course, bring it first into the staging layer, and then we bring it also into the core layer.
- So this is why we need to always have a Delta column for this process. And this needs to be available for every table that we are using in our data warehouse.
- And typically, this is a timestamp of a transaction or some type of create date. So even though you can see that in this case, it's a date, ideally, it is really a timestamp.
- Usually, this is available in the source systems, and with that, we can identify the new data that has not been loaded yet in our ETL process.
- So this is what we always should have available to be able to set up a Delta load in our ETL process.
- And note also that we are not changing anything in the workflow, so everything just remains the same. We have the same transformations, so our ETL process basically has exactly the same structure, and it is, in fact,

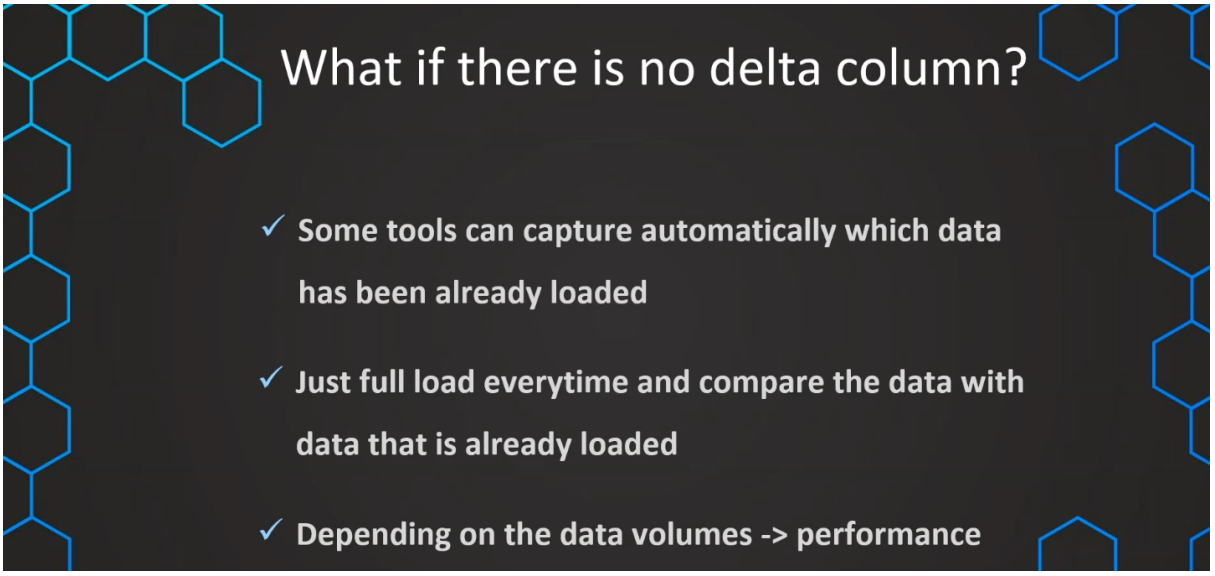
the same ETL process, except that now, we run this periodically, and we also have some filter in place on these Delta columns.



- So now not all of the data is always loaded, but only the data that has not been loaded before. And also, sometimes if we don't have some timestamp available, we can have a look if there are some other possibilities,
- so some other columns that we can use to identify new data. So this could be alternatively, also be something like a primary key, but of course, we have sometimes difficulties if it's just a natural key that is just maybe a random set of numbers.
- So if we want to use a primary key, we need to make sure that it's a incremental number.
- So something that we can really use to identify what is a new row that has not been loaded before. And how can this now be implemented in practice?



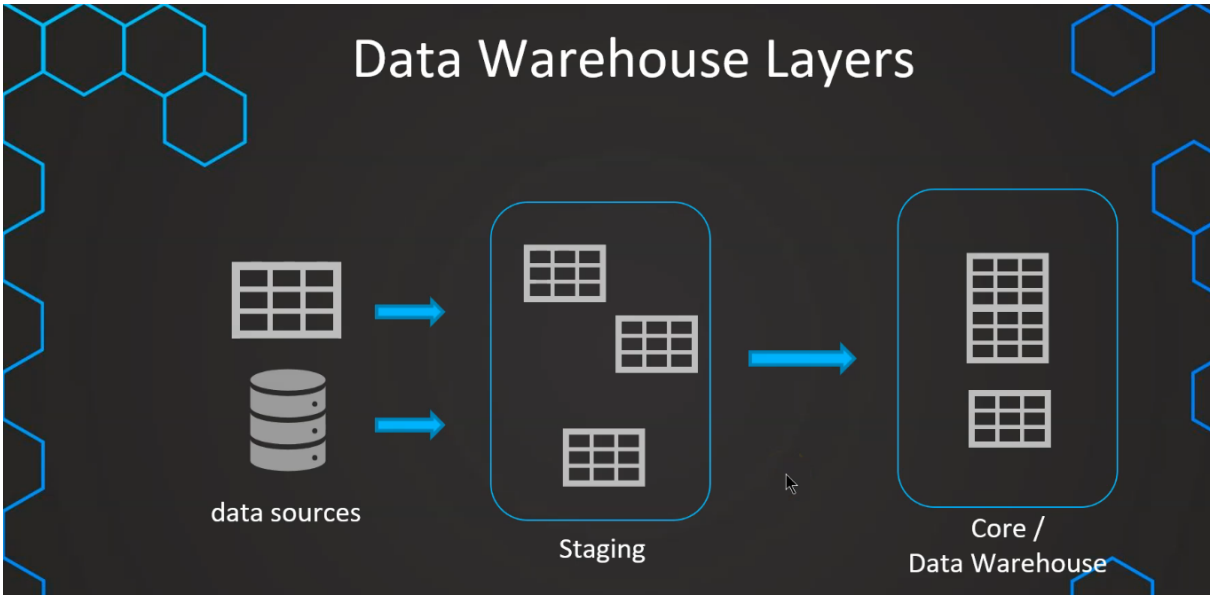
- Let's assume in our last ETL run, the row with the highest value on the sales key, or also possibly on our transaction date, on our timestamp, is the value, for example, in our case, four.
- So we basically remember the maximum value of our timestamp or our Delta column. So in our case, we just take the maximum of the sales key, which is four, and then in the next run, when we read the data from the source systems, put this filter on this sales key.
- So we just store the result usually in a variable. So for example, it's a variable called X, and this has now the value four, and then in the next run, we just load that data where the sales key is larger than four.
- So in this case, larger than four is only five, so in the next run, we would only load this row number five.



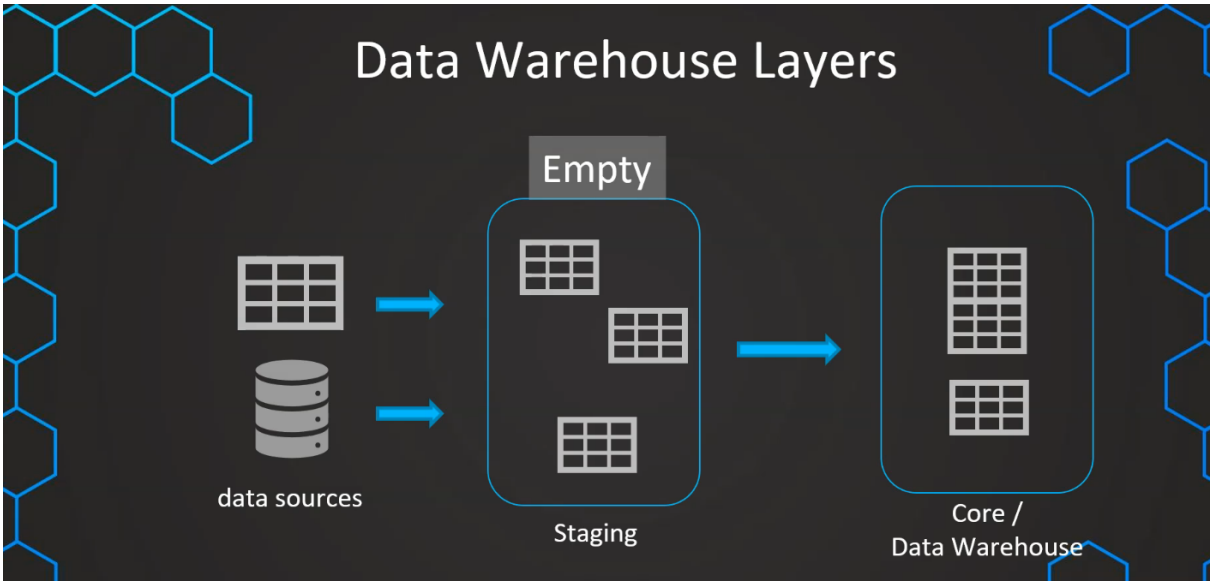
- But now the question might be, what is, if there is no Delta column, and first of all, usually, there is a Delta column, usually, there is a timestamp that we can use, but sometimes, if the data is really not in good shape and we don't have a Delta column available, we have a few alternatives.
- There are even some tools that can capture automatically what data has been loaded already by the metadata. So in that case, if we have such a tool and this is available to us, we don't need to worry, because then this is done automatically, just by the metadata, what data has been changed, what data is new, and this is then extracted automatically.
- Usually, this is not the case, though. In such situations, this can be the case, usually for dimension tables, where we don't have a time available.
- And in that case, we can do also a full load every time our ETL process is running.
- And then we just compare this new data with the already loaded data, if there are any changes, if there are any additional columns.
- Of course, typically, this is the case, not for our fact tables, because there, we usually do have a timestamp, but for our dimension tables.
- And then again, here, we have to be a little bit careful, how much data is that? So depending on the volume, if this is a huge dimension table, we have to be a little bit careful with the performance. So how much time do we spend on the source systems? What are the times where we can do this load?

- But since this is usually dimension tables, and they are typically not super large, we should be okay, and we usually then don't slow the source systems down too much, or we can also just decide to run this during the night, for example, if there are some hours where we can put a little bit of load on the source systems.
- And also, of course, what we have to keep in mind that the more data we load, the longer also our ETL process can take.
- So this is then also a little bit critical, if, for example, we want to have a very high frequency on our ETL.
- So for example, the business users request that this ETL process needs to update the data, let's say every 30 minutes, and then our ETL load could take maybe 40 minutes.
- So this is because maybe we need to do some full loads, and of course, they take more time, but this is then, of course, something that we have to live with.
- So this is something that we have to keep in mind, and therefore, if we are thinking about doing a full load, we just should keep that in mind and also decide if the table is not too large to make a full load, so this is something that we should do.

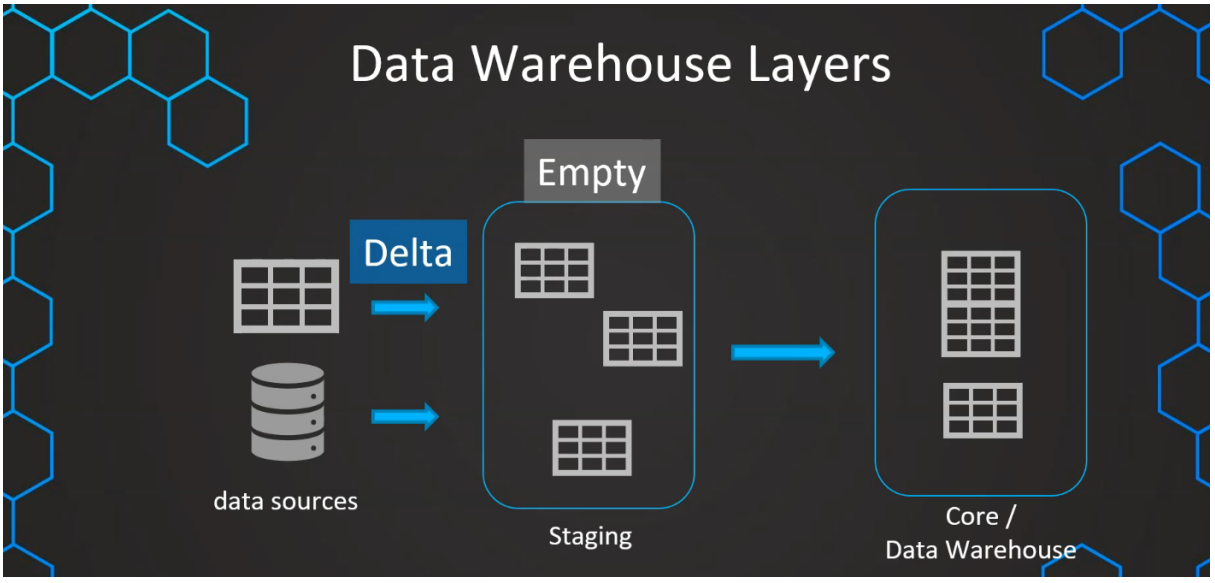
Load workflow



- Now that we've worked through the extract process, we want to see how the data can be loaded, and we want to look at the whole workflow to better understand how the load process then works.
- So we have seen that once we have run our ETL and then we want to run it again, the staging layer is empty, because we always, after each run empty the staging layer.



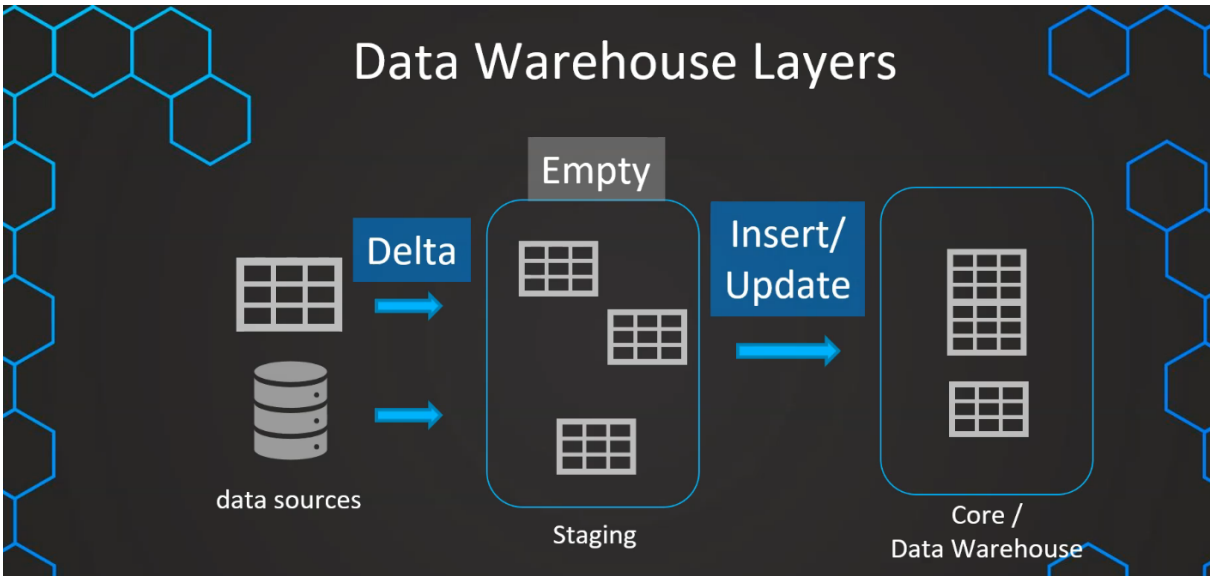
- So now we assume that this staging layer is empty, and now we of course run a Delta.



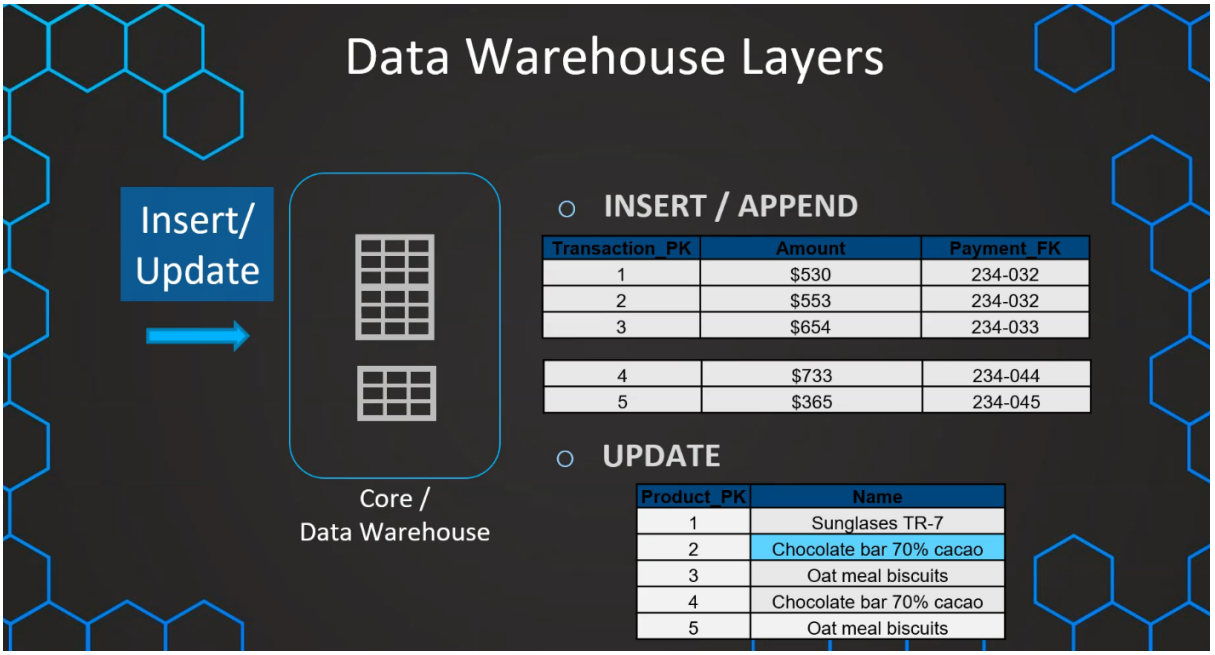
- So we have stored the value of the maximum value of the Delta column and now we know what is new data in our source systems and then this data is

just filtered and extracted into our staging environment.

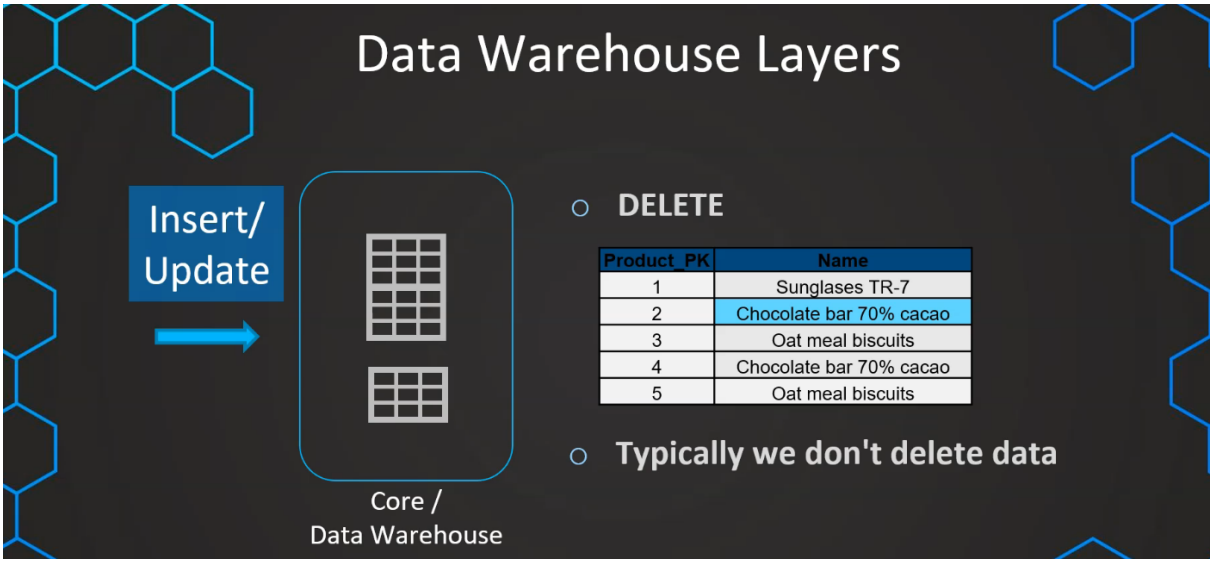
- And now from this staging environment all of the data will be pushed with the transformations into our core layer, and we use usually insert update for that.



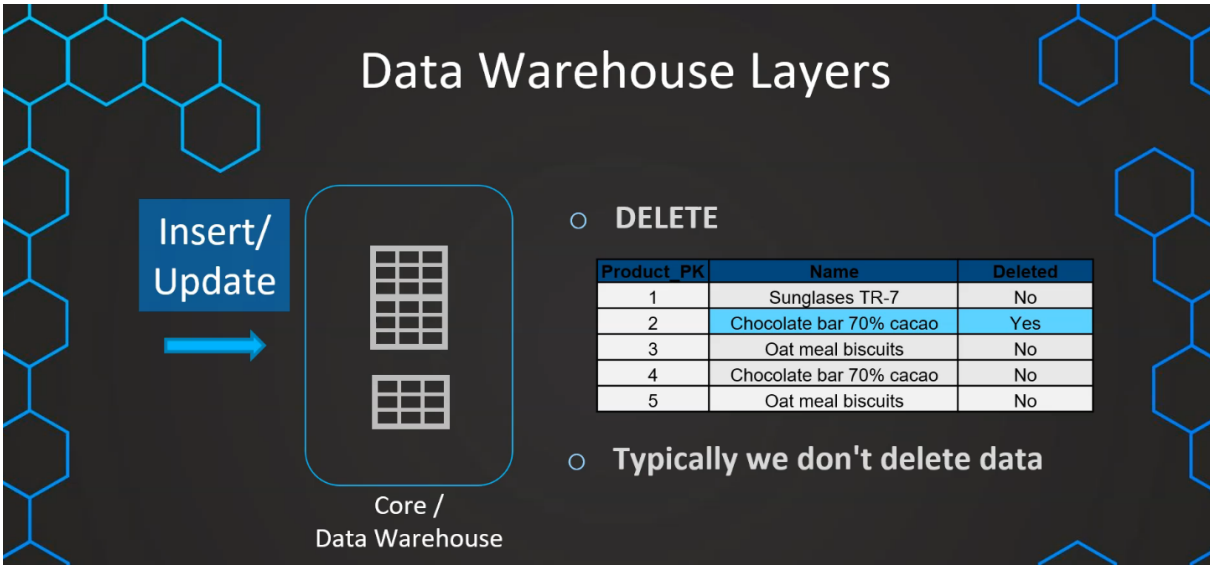
- That means we have different types of how the data is written into this core layer, and of course, this is depending on whether the data might already exist, so there might be some updates necessary, or if the data can just be appended.
- So let's have a look at the several options that we have available in here.
- So the common type is just a very simple **insert/update**. That means we get additional data that is not present yet that has never been loaded into our core layer and then this data will just be appended in our table.



- So now after we have seen these rows have existed before, we append those two additional rows. So this is a very simple case.
- Sometimes this is a more rare case, we also have some updates.
- So this can be now also done automatically via, for example insert update tools that once a value already exists. So for example, we have the primary key number two, and this value already exists and this value has changed then this value can also be updated.
- And again there are usually tools that handle that automatically. So they recognize usually based on the primary key if that data already exists and then if that data needs to be updated, and if it does not exist the entire row can just be appended to our table.
- These are the main operations insert and update that we are doing during our load process.
- And you might also note that I'm not mentioning delete, because delete in fact is something that is usually not occurring.

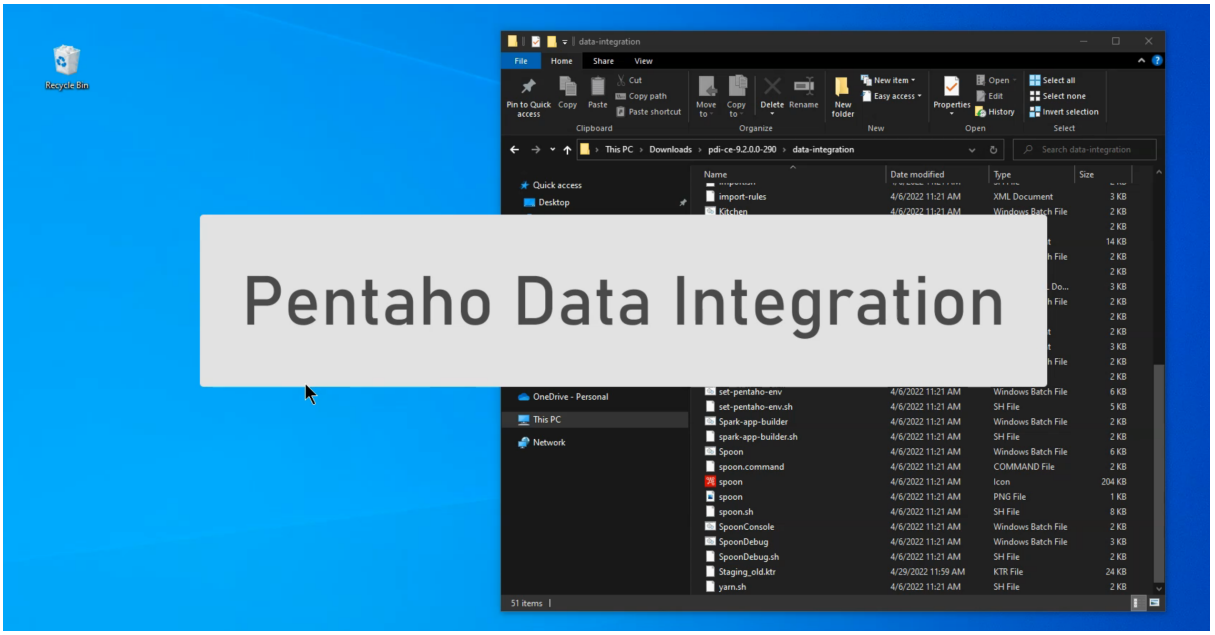


- We usually don't want to delete data from our data warehouse, because we want to maintain the history.
- Of course, in some cases it might be necessary, or we might want to do that, but usually in the common type, this is not necessary.
- So what is then you might be asking happening. or what might we want to do if a row in the source system gets deleted, for example a dimensional value gets removed from our source system.
- So a product is not there anymore, and it needs to be deleted. In that case still, we usually should not delete that row, but we can add an additional column that is just helping with the administration, and this is then basically a flag, whether this is maybe a current product or not, whether this is something that has been deleted from the source system or not.

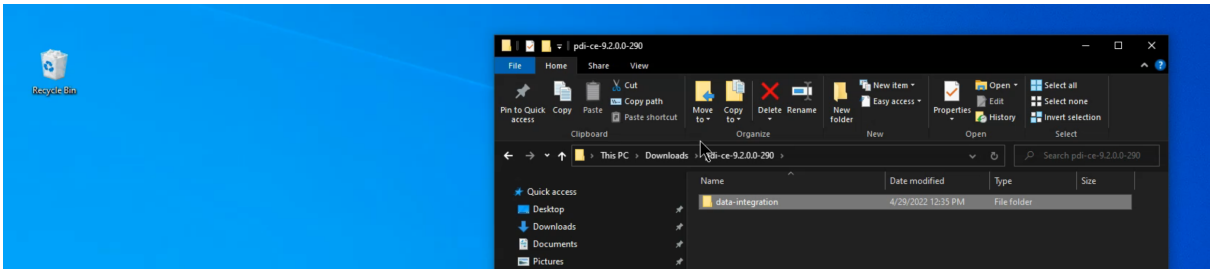


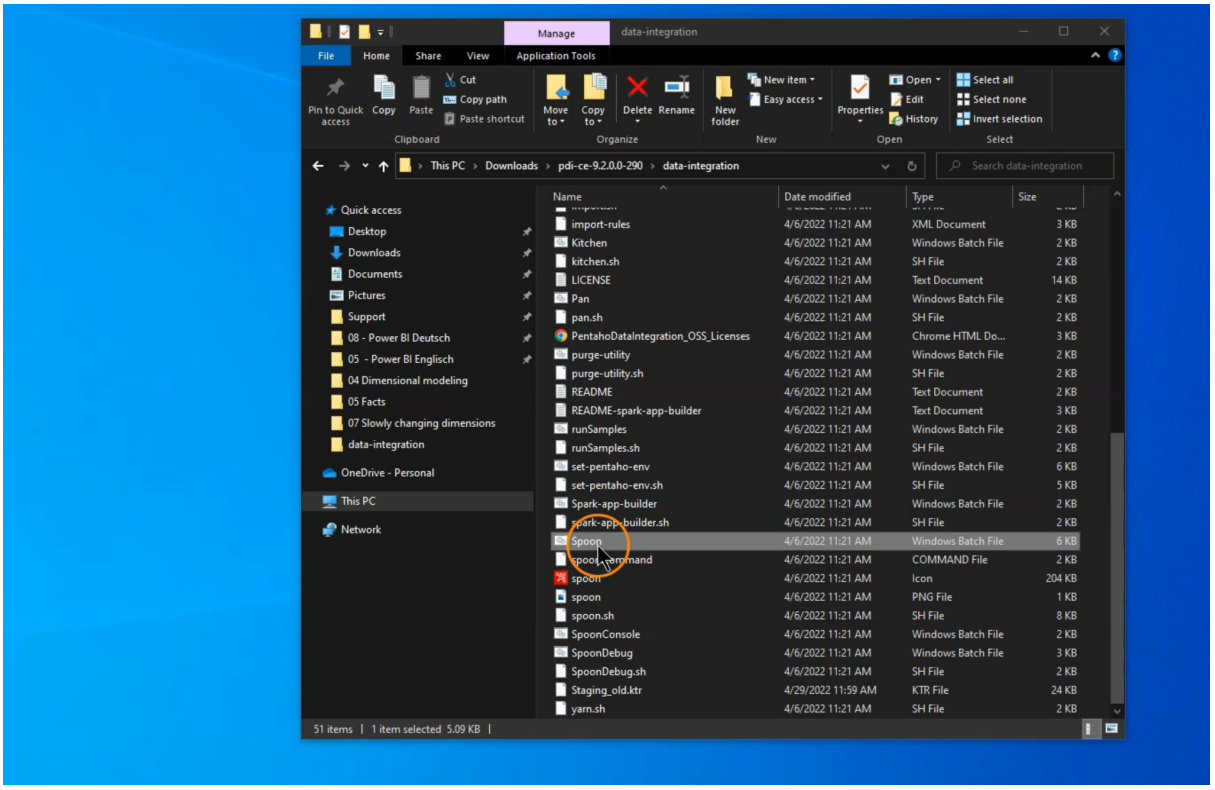
- So if there should be data deleted on our source systems, this is how we should usually handle that by just adding an additional column.
- So this is now how we load that data using insert update in our data warehouse, but now we also want to understand what are the different steps in between for our transformation process.

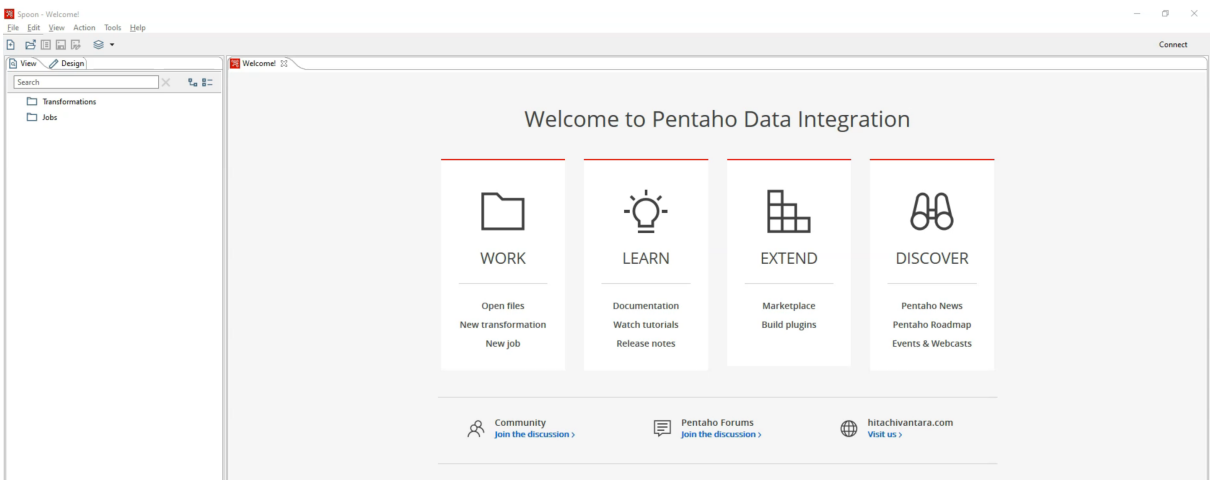
Demo: Quick intro to Pentaho



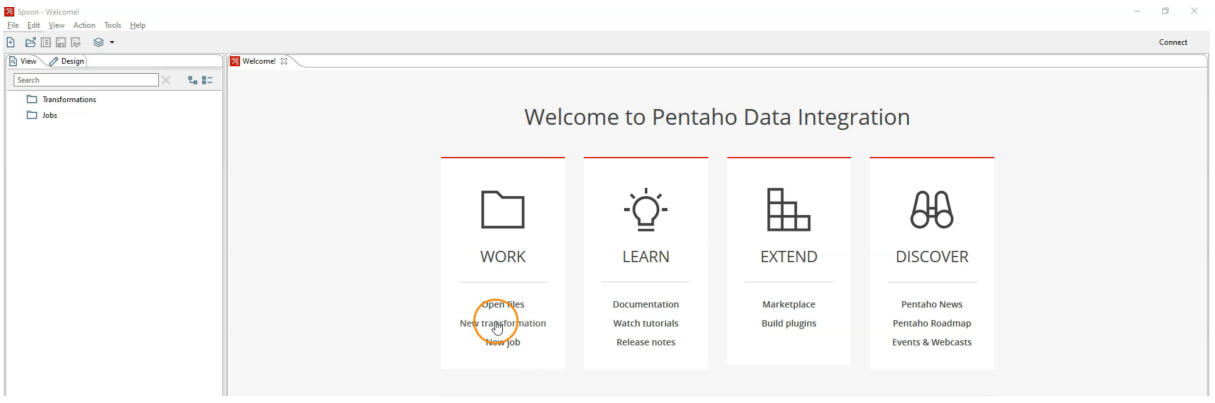
- We have extracted the zip folder for PDI (data-integeration) , go into that folder and open spoon.bat in Windows or spoon.sh in Mac



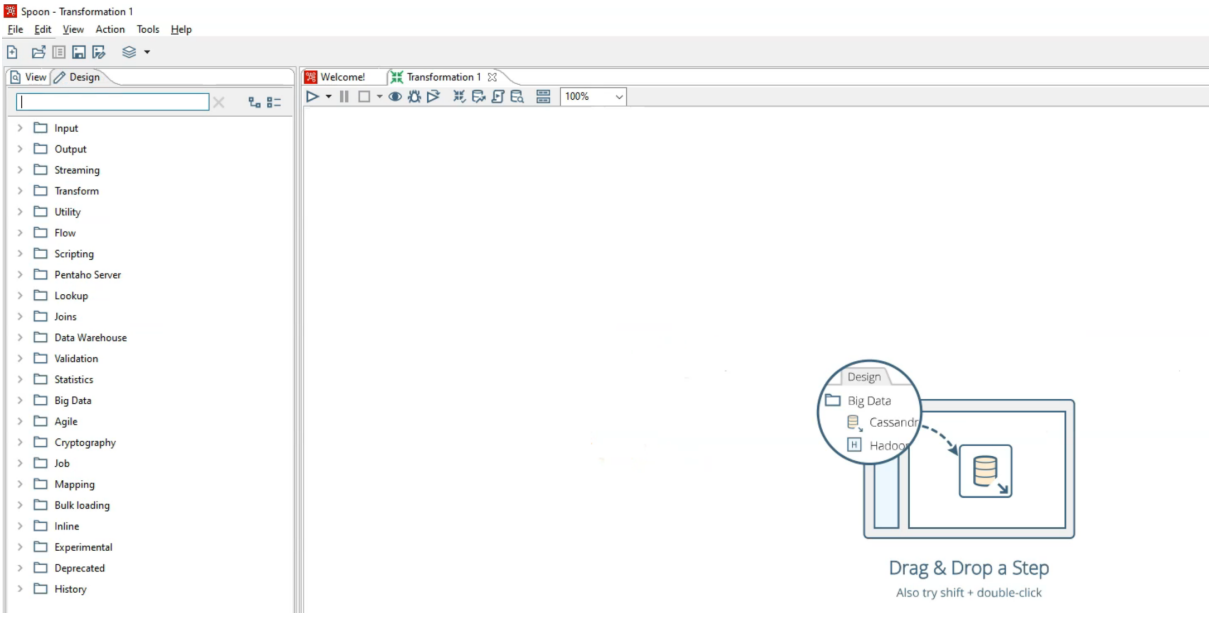




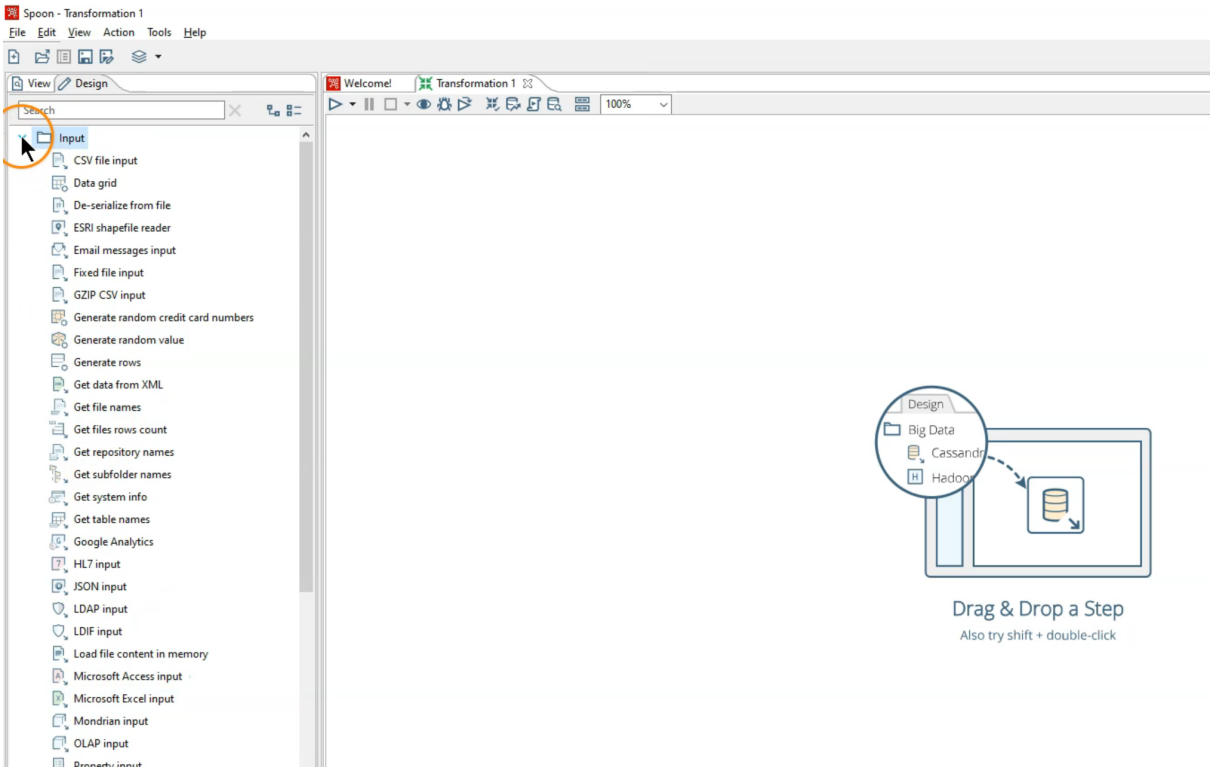
- Go to New transformation



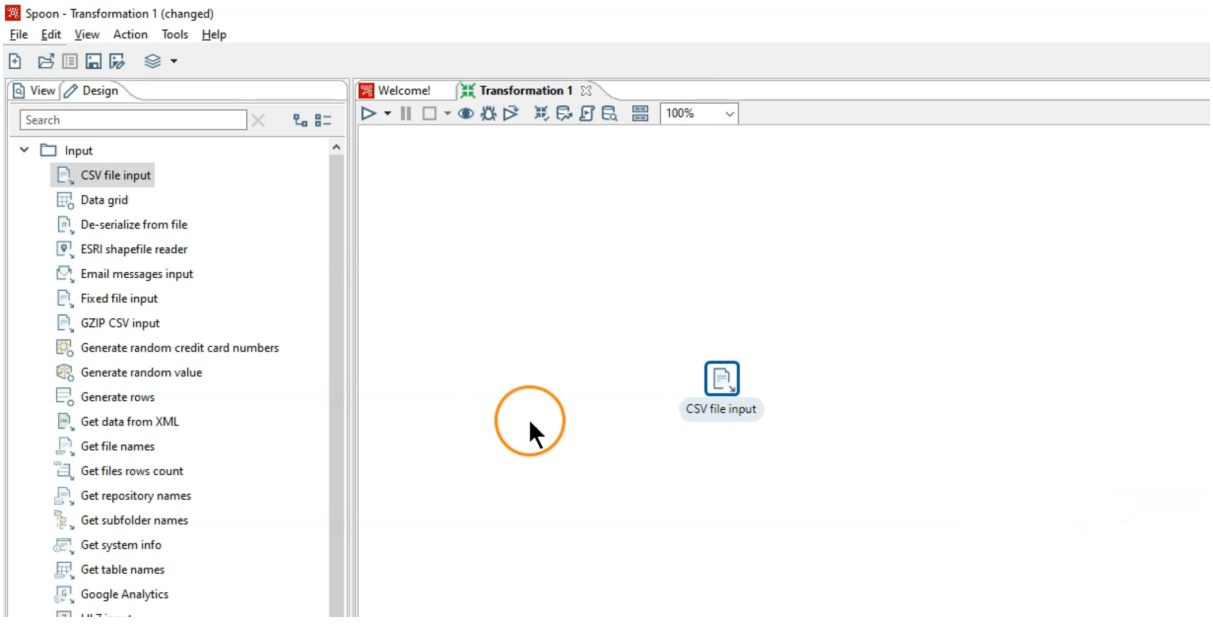
- For now already we can set up transformations. And in these transformations, we build a very short sequence of some transformational steps and then we can save the sequence into a transformation.
- And then all of our transformations, we can now place into a job. And again, in this job, we then have a sequence of transformations.
- And this job then can be scheduled. So basically this is then our ETL process.



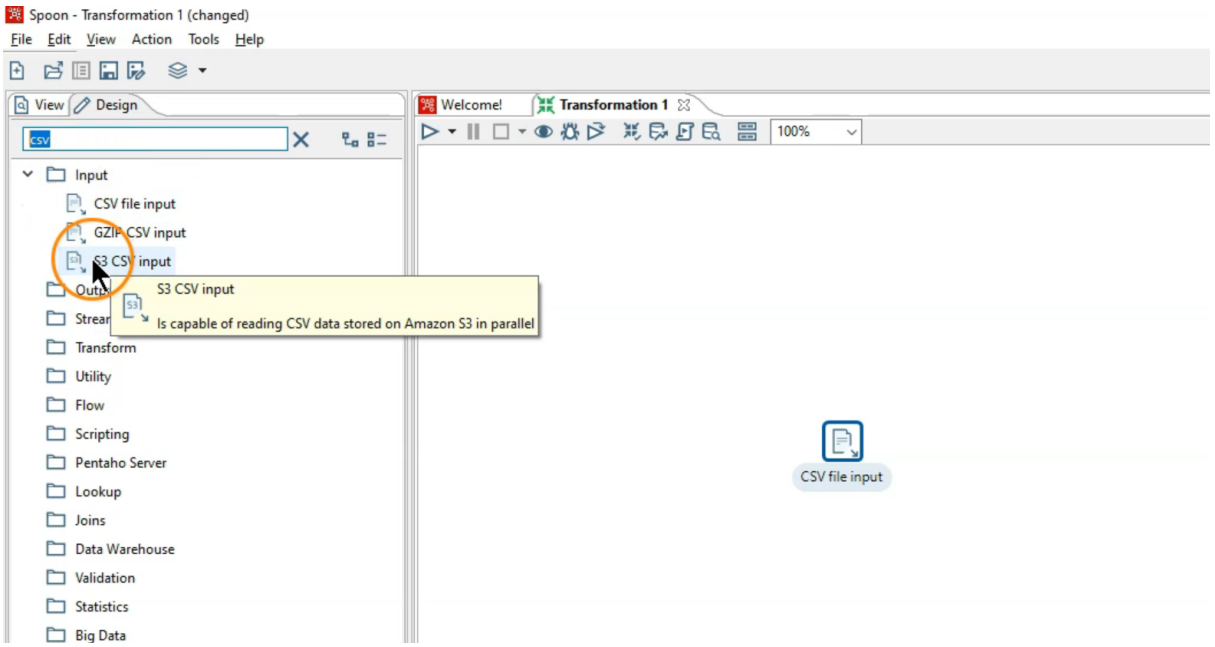
- It is named as “Transformation 1” and we have a big canvas to work with
- Here we can drag and drop different steps to setup our workflow
- We can see we have different kinds of transformations we can do, for example for input we have variety of options to read the data from



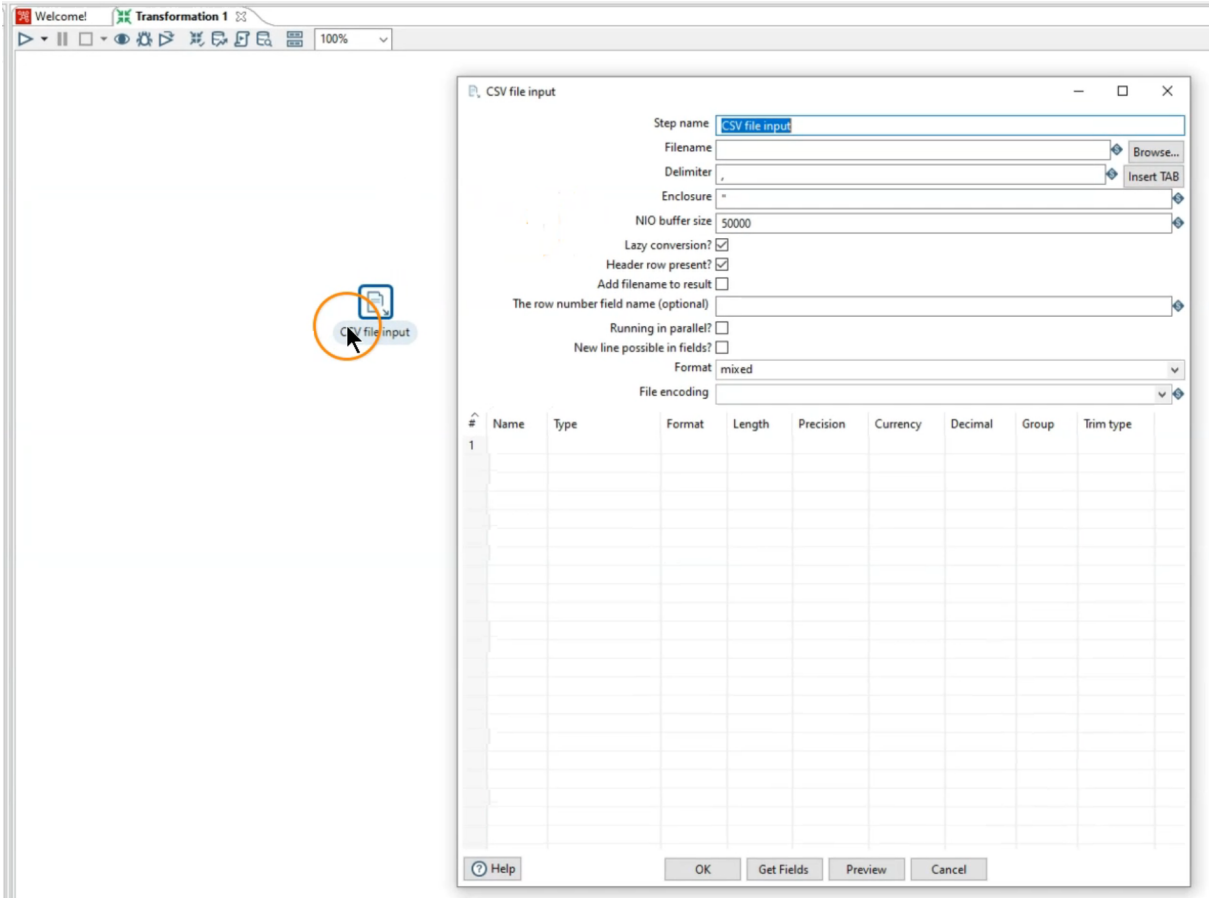
- To use any step/option we can double click on them or drag and drop them to the canvas



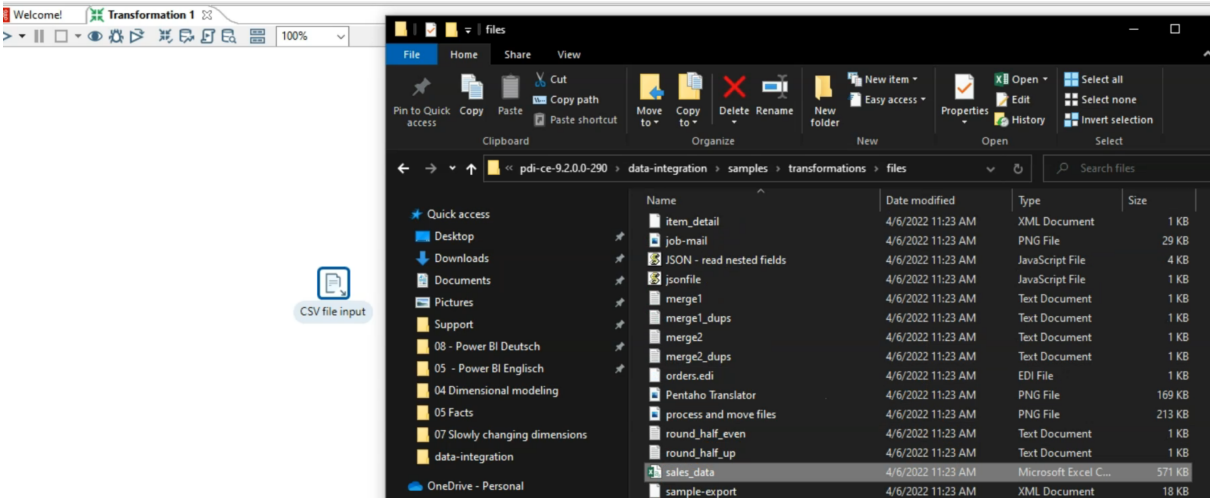
- We can also search in the searchbar



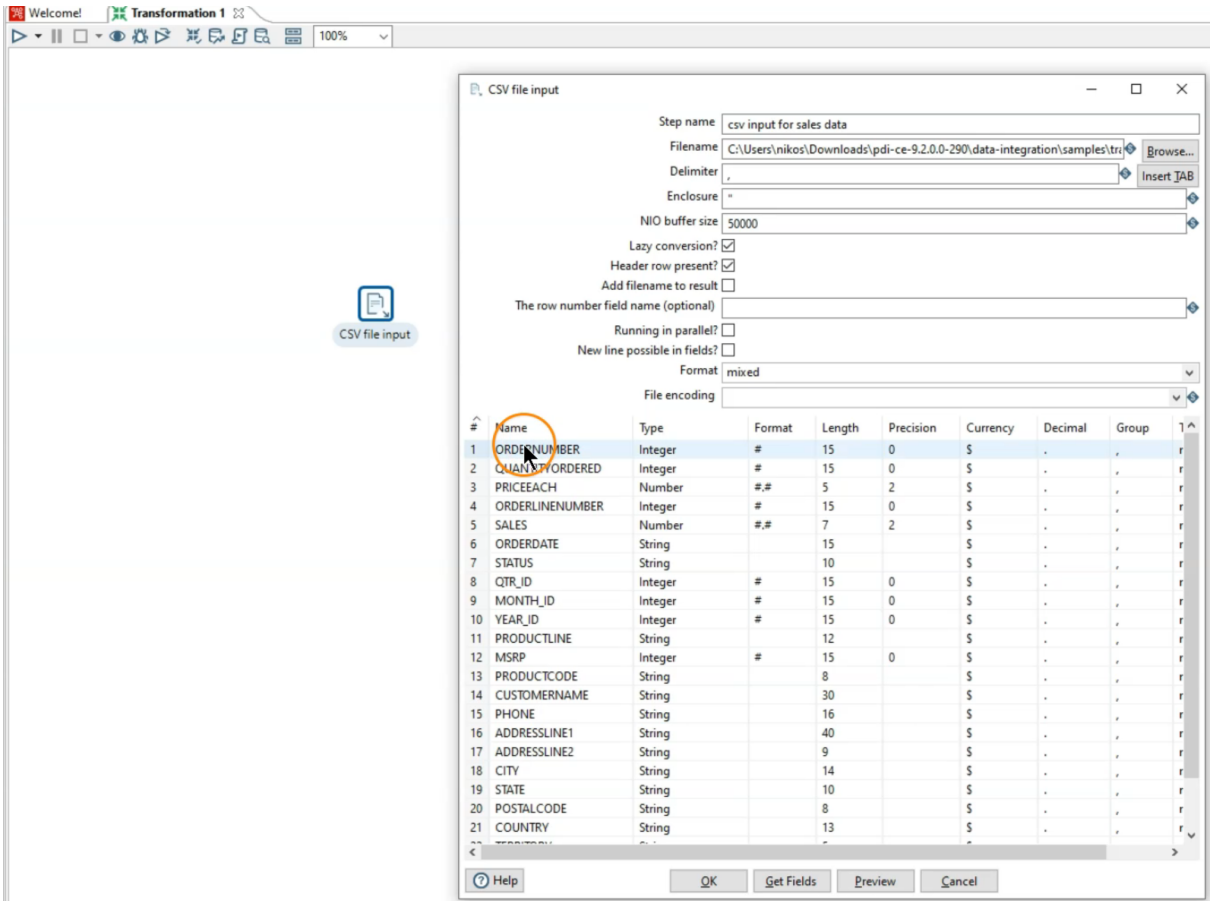
- To configue the step , we can double click on them and configure



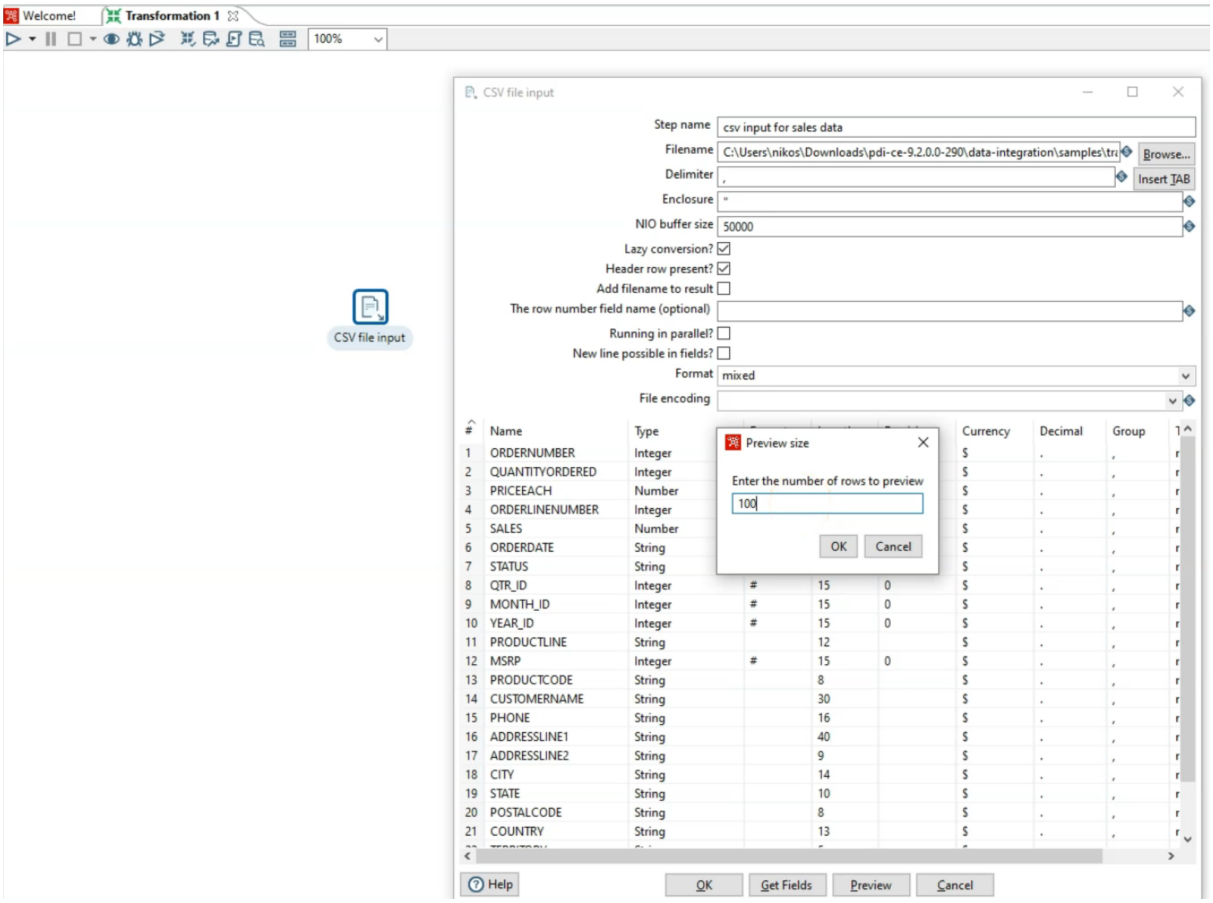
- Go to the path to find sample data
- data-integeration/samples/transformations/files

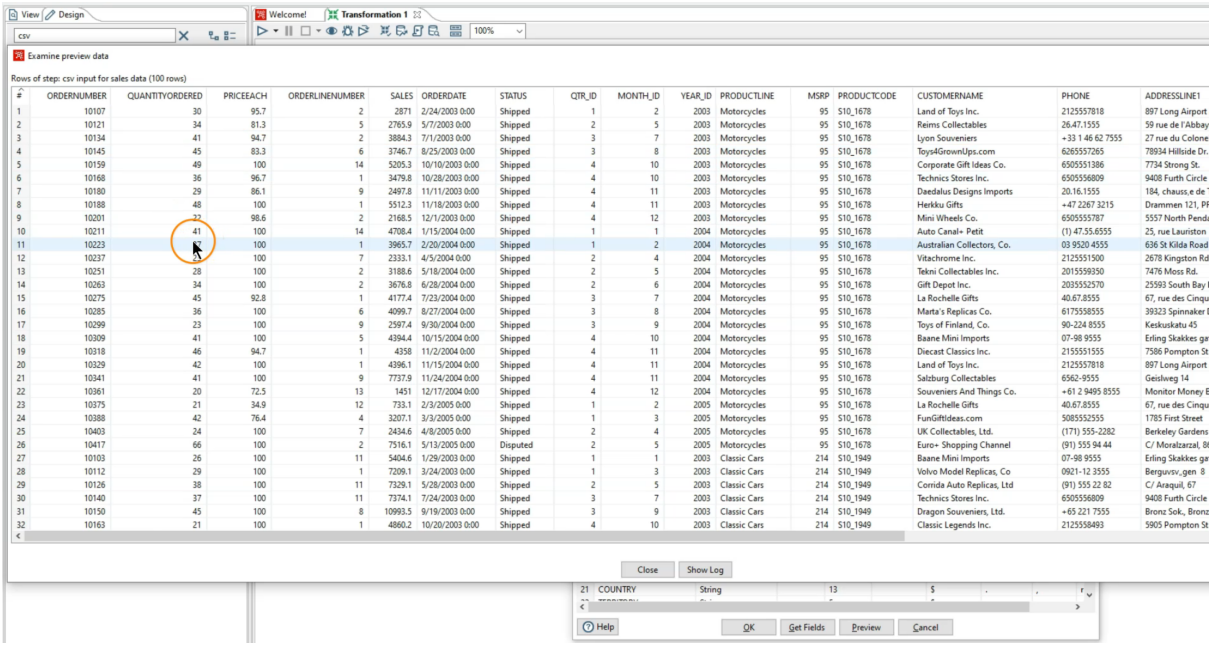


- We can also use the Browse option or we can copy paste the path
- Click on get fields to see list of all columns from the selected files

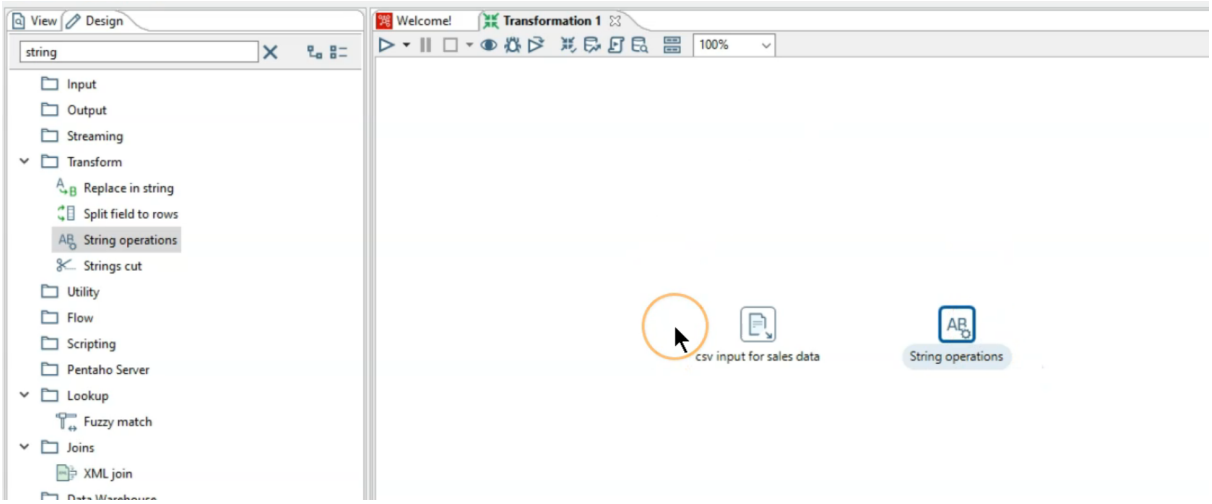


- We can click Preview and give number of rows to view the data

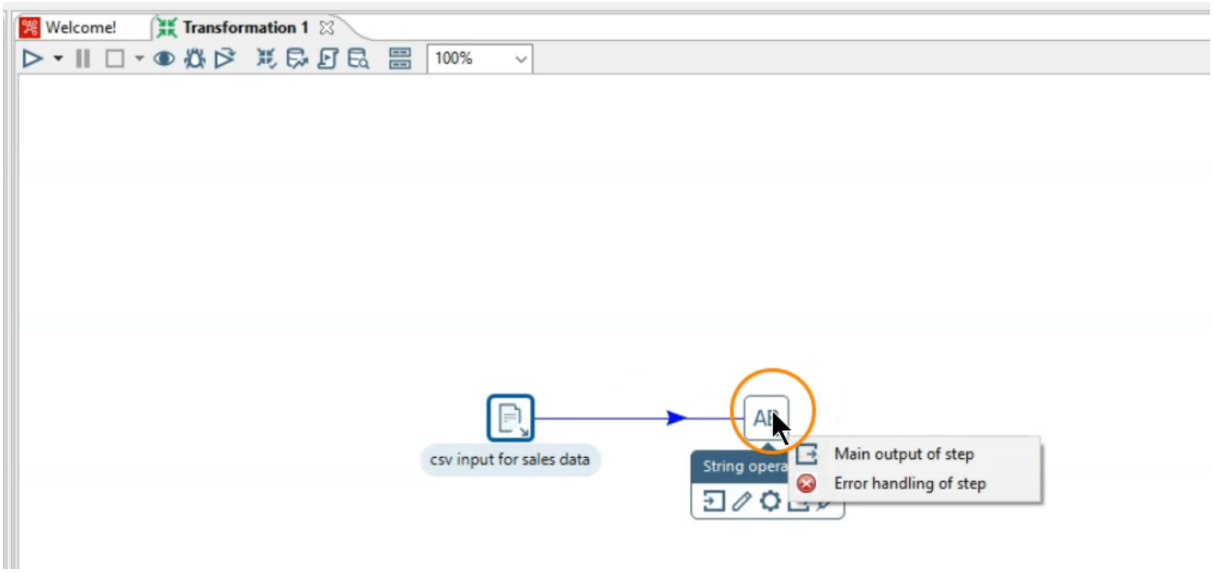




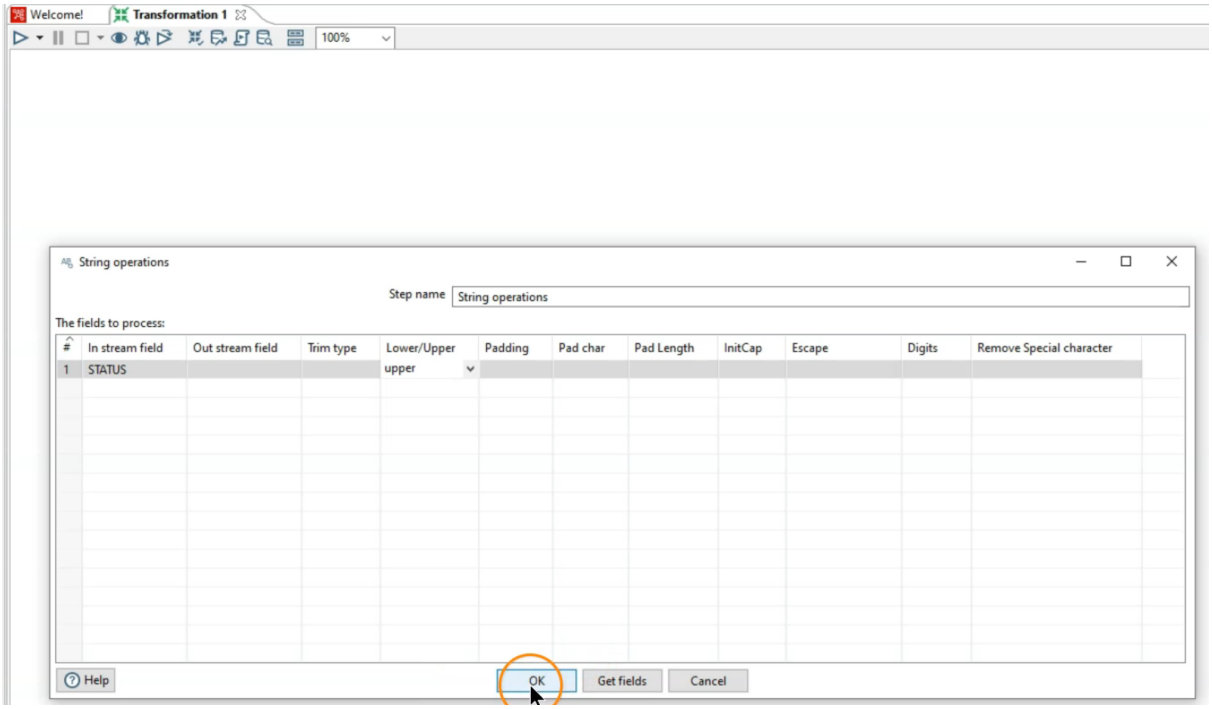
- Click OK to finalise the step
- We can do other operations like String operations



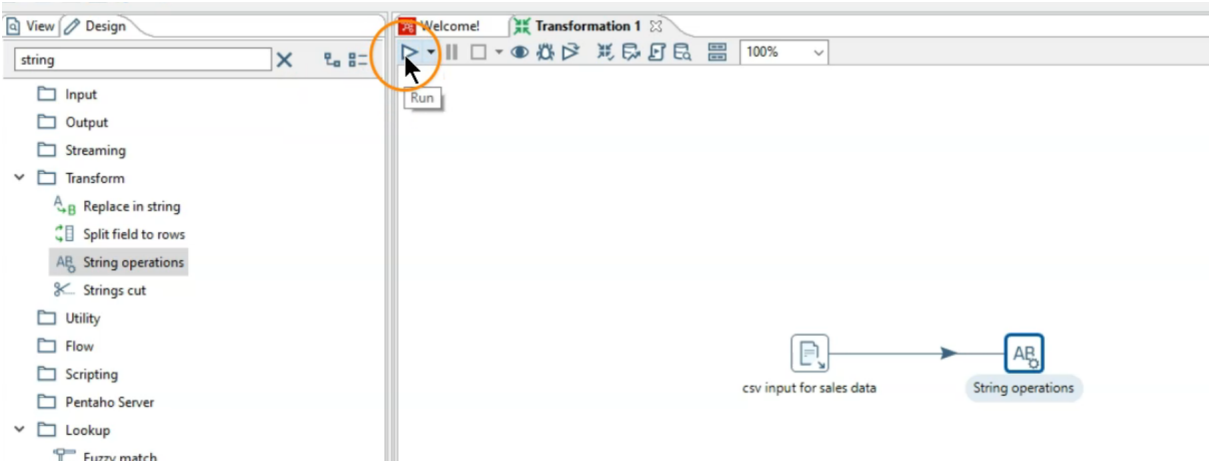
- We need to connect the steps to have them executed in sequence
- Hold Shift Key and drag and drop the arrow key to make the connection and chose Main output of Step

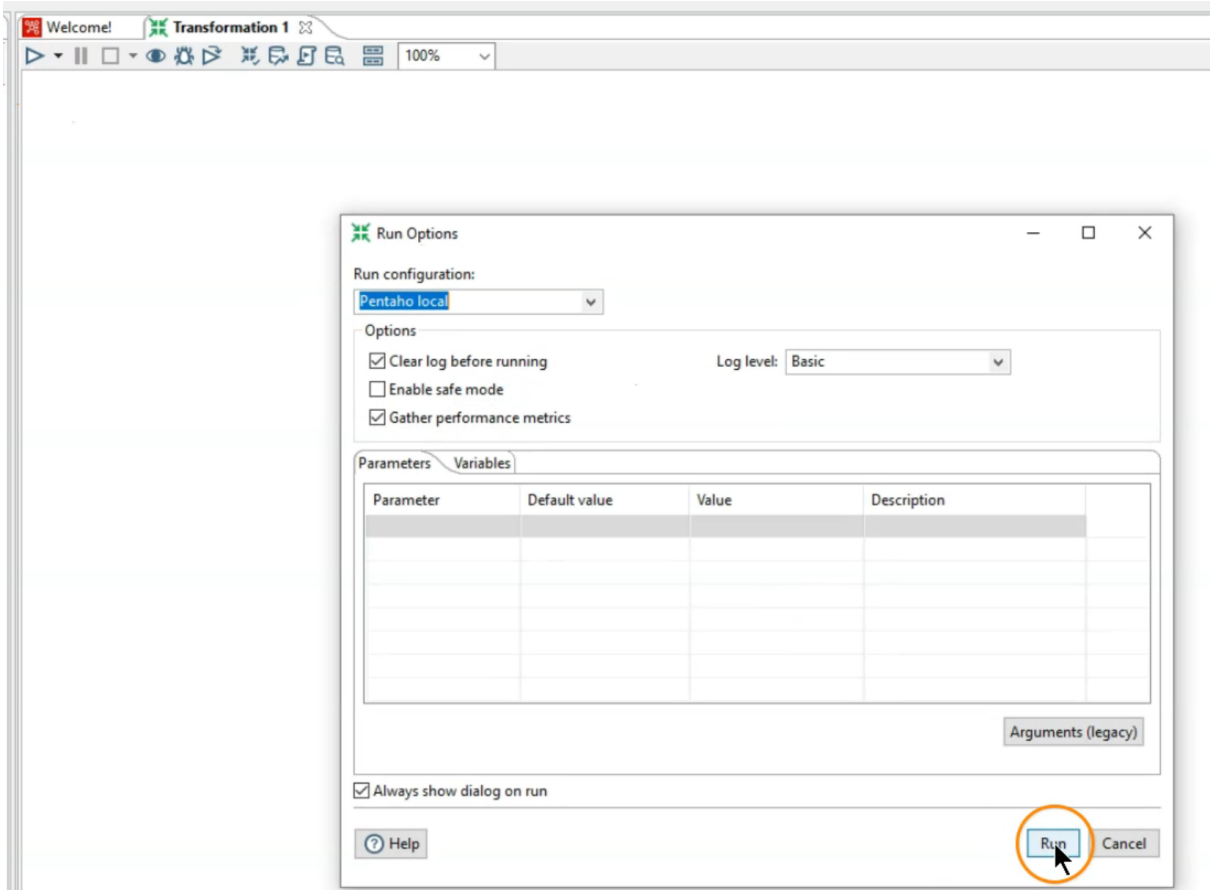


- Double click on the String operation and configure
- We want to chose the column we want to make the string operation and write what operation we need to make

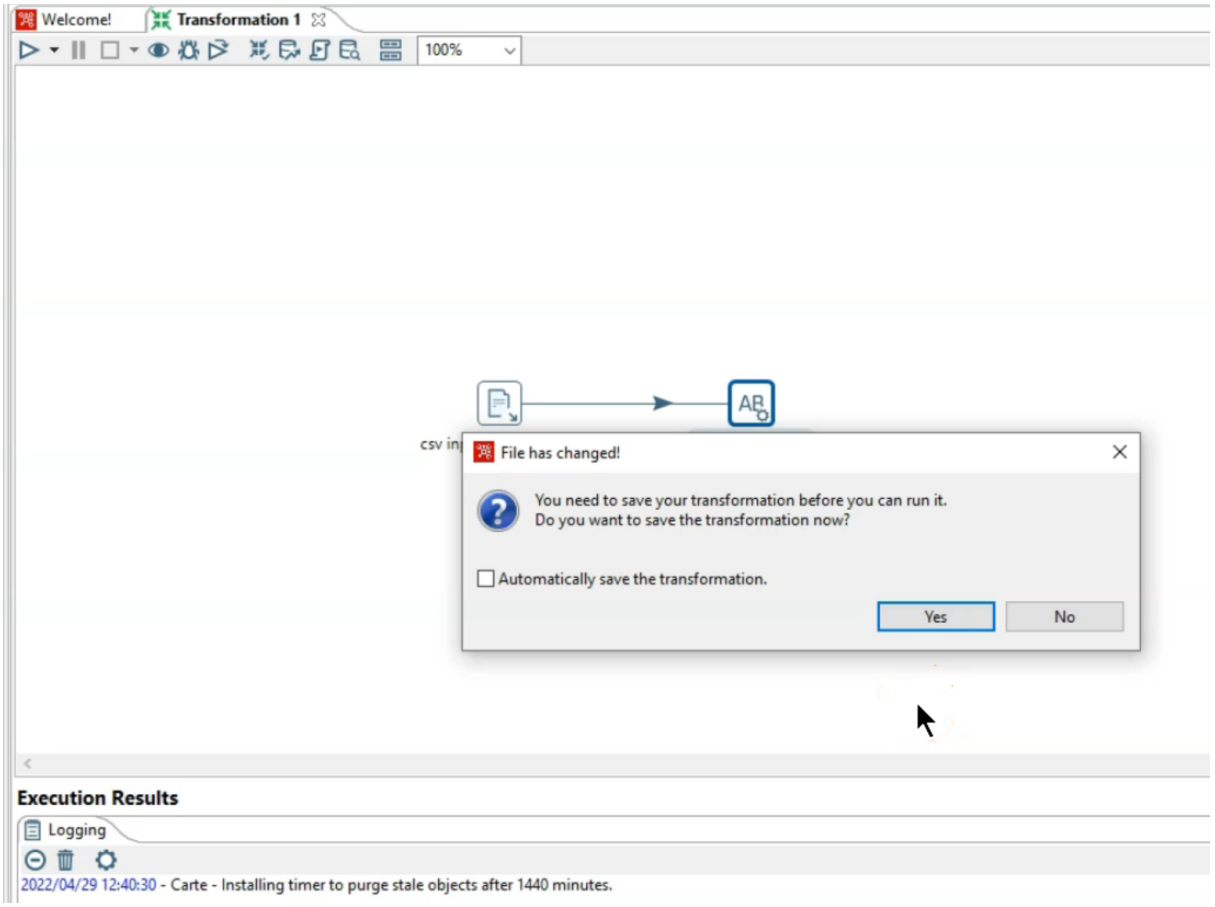


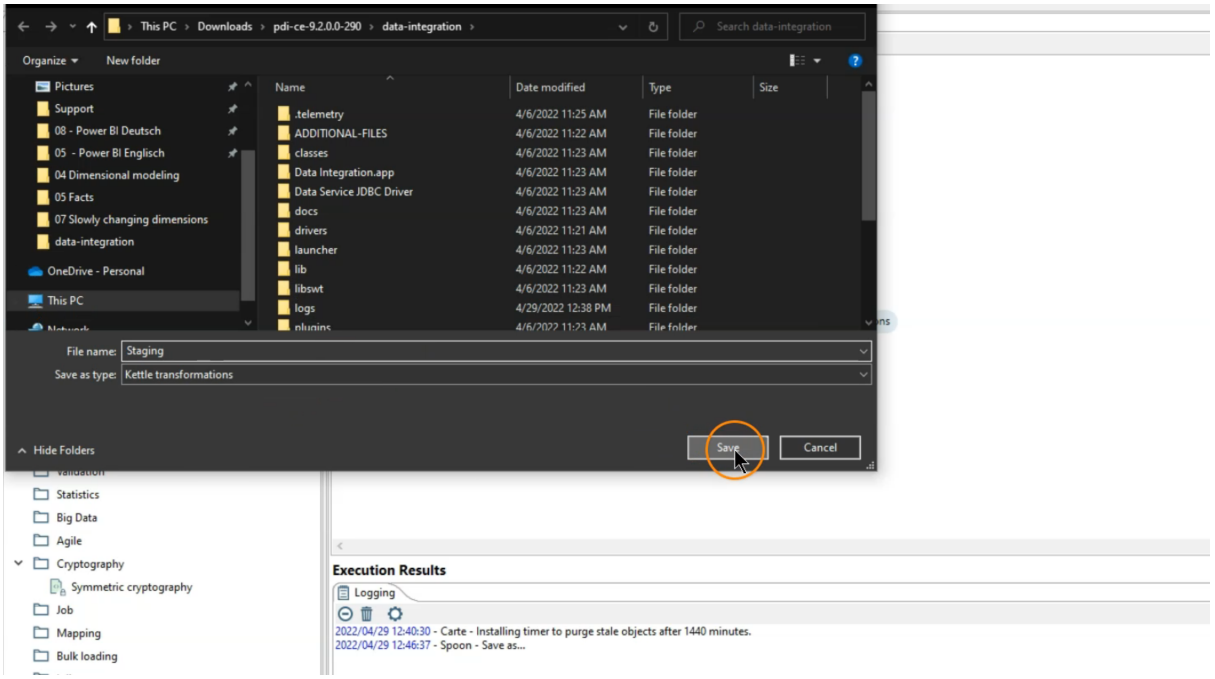
- Click on Play button to run the workflow

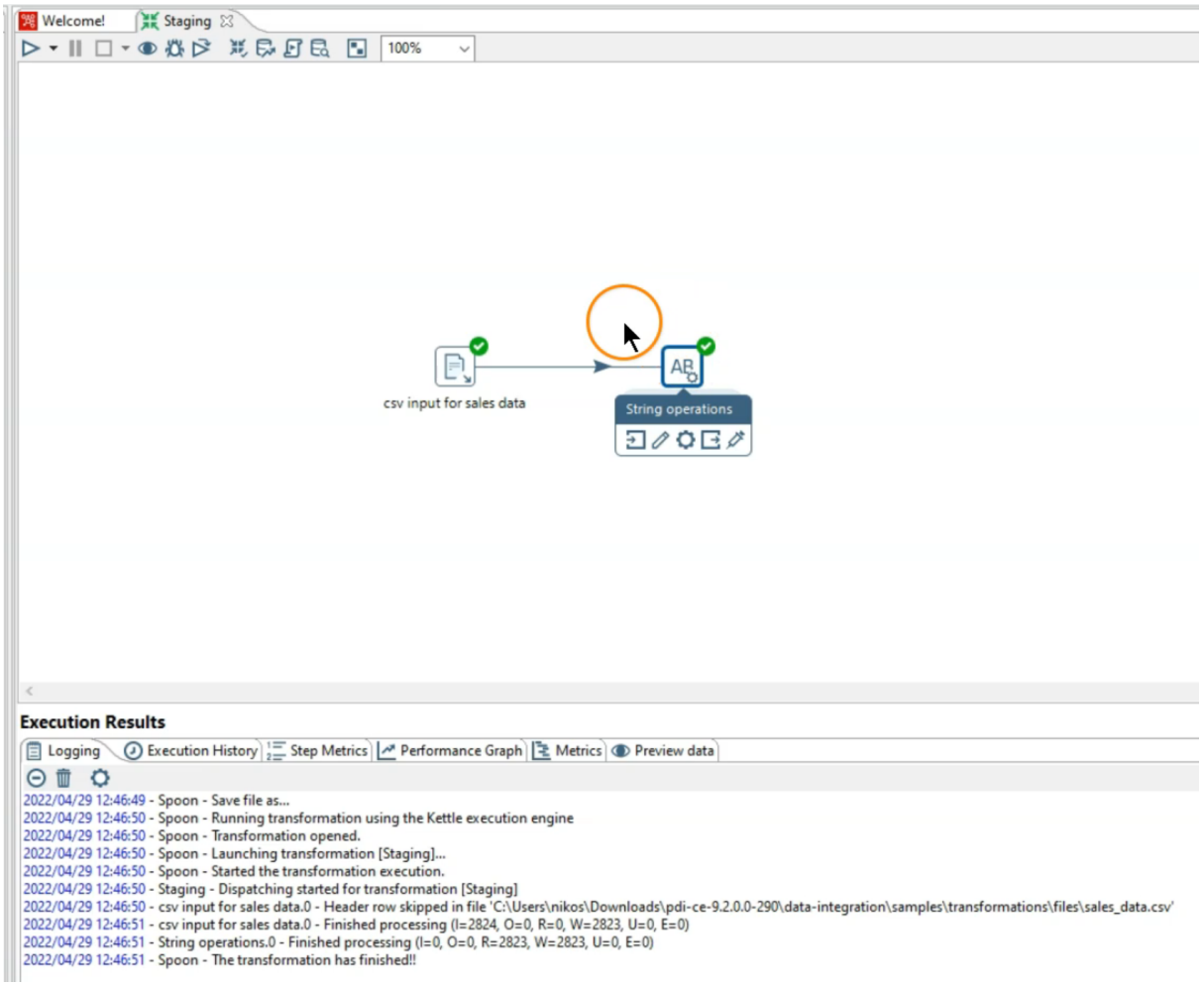




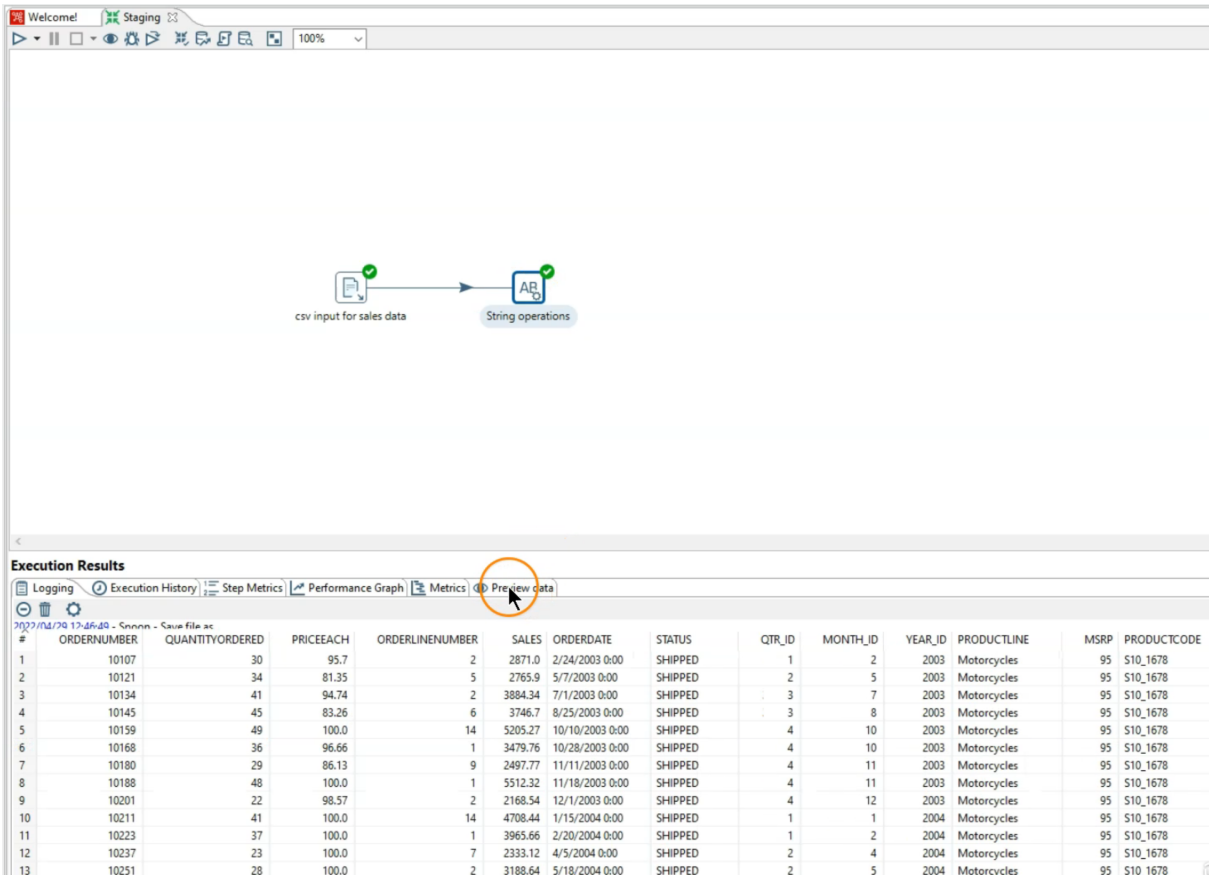
- Before we run our workflow we have to save it


Welcome!
Transformation 1
100%
csv in
File has changed!
You need to save your transformation before you can run it.
Do you want to save the transformation now?
Automatically save the transformation.
Yes
No
Execution Results
Logging
2022/04/29 12:40:30 - Carte - Installing timer to purge stale objects after 1440 minutes.


This PC
Downloads
pdi-ce-9.2.0.0-290
data-integration
Search data-integration
Organize
New folder
Pictures
Support
08 - Power BI Deutsch
05 - Power BI Englisch
04 Dimensional modeling
05 Facts
07 Slowly changing dimensions
data-integration
OneDrive - Personal
This PC
Name
Date modified
Type
Size
.telemetry 4/6/2022 11:25 AM File folder
ADDITIONAL-FILES 4/6/2022 11:22 AM File folder
classes 4/6/2022 11:23 AM File folder
Data Integration.app 4/6/2022 11:23 AM File folder
Data Service JDBC Driver 4/6/2022 11:23 AM File folder
docs 4/6/2022 11:23 AM File folder
drivers 4/6/2022 11:21 AM File folder
launcher 4/6/2022 11:23 AM File folder
lib 4/6/2022 11:22 AM File folder
libswt 4/6/2022 11:23 AM File folder
logs 4/29/2022 12:38 PM File folder
File name: Staging
Save as type: Kettle transformations
Hide Folders
Save
Cancel
Statistics
Big Data
Agile
Cryptography
Symmetric cryptography
Job
Mapping
Bulk loading
Execution Results
Logging
2022/04/29 12:40:30 - Carte - Installing timer to purge stale objects after 1440 minutes.
2022/04/29 12:46:37 - Spoon - Save as...

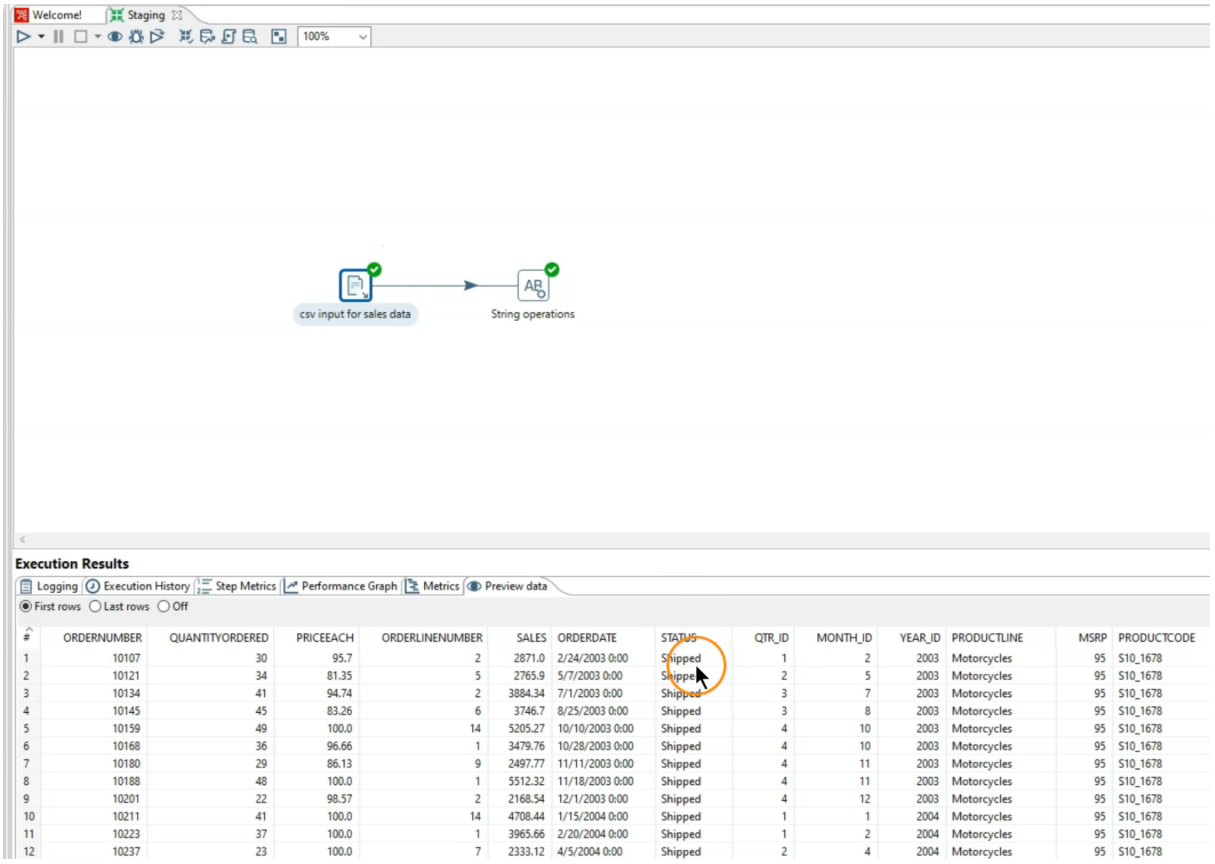


- Once the Job is finished, click on a particular step String operation step for example and then click on Preview data tab on the bottom, we can see the transformed data



#	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE
1	10107	30	95.7	2	2871.0	2/24/2003 0:00	SHIPPED	1	2	2003	Motorcycles	95	S10_1678
2	10121	34	81.35	5	2765.9	5/7/2003 0:00	SHIPPED	2	5	2003	Motorcycles	95	S10_1678
3	10134	41	94.74	2	3884.34	7/1/2003 0:00	SHIPPED	3	7	2003	Motorcycles	95	S10_1678
4	10145	45	83.26	6	3746.7	8/25/2003 0:00	SHIPPED	3	8	2003	Motorcycles	95	S10_1678
5	10159	49	100.0	14	5205.27	10/10/2003 0:00	SHIPPED	4	10	2003	Motorcycles	95	S10_1678
6	10168	36	96.66	1	3479.76	10/28/2003 0:00	SHIPPED	4	10	2003	Motorcycles	95	S10_1678
7	10180	29	86.13	9	2497.77	11/11/2003 0:00	SHIPPED	4	11	2003	Motorcycles	95	S10_1678
8	10188	48	100.0	1	5512.32	11/18/2003 0:00	SHIPPED	4	11	2003	Motorcycles	95	S10_1678
9	10201	22	98.57	2	2168.54	12/1/2003 0:00	SHIPPED	4	12	2003	Motorcycles	95	S10_1678
10	10211	41	100.0	14	4708.44	1/15/2004 0:00	SHIPPED	1	1	2004	Motorcycles	95	S10_1678
11	10223	37	100.0	1	3965.66	2/20/2004 0:00	SHIPPED	1	2	2004	Motorcycles	95	S10_1678
12	10237	23	100.0	7	2333.12	4/5/2004 0:00	SHIPPED	2	4	2004	Motorcycles	95	S10_1678
13	10251	28	100.0	2	3188.64	5/18/2004 0:00	SHIPPED	2	5	2004	Motorcycles	95	S10_1678

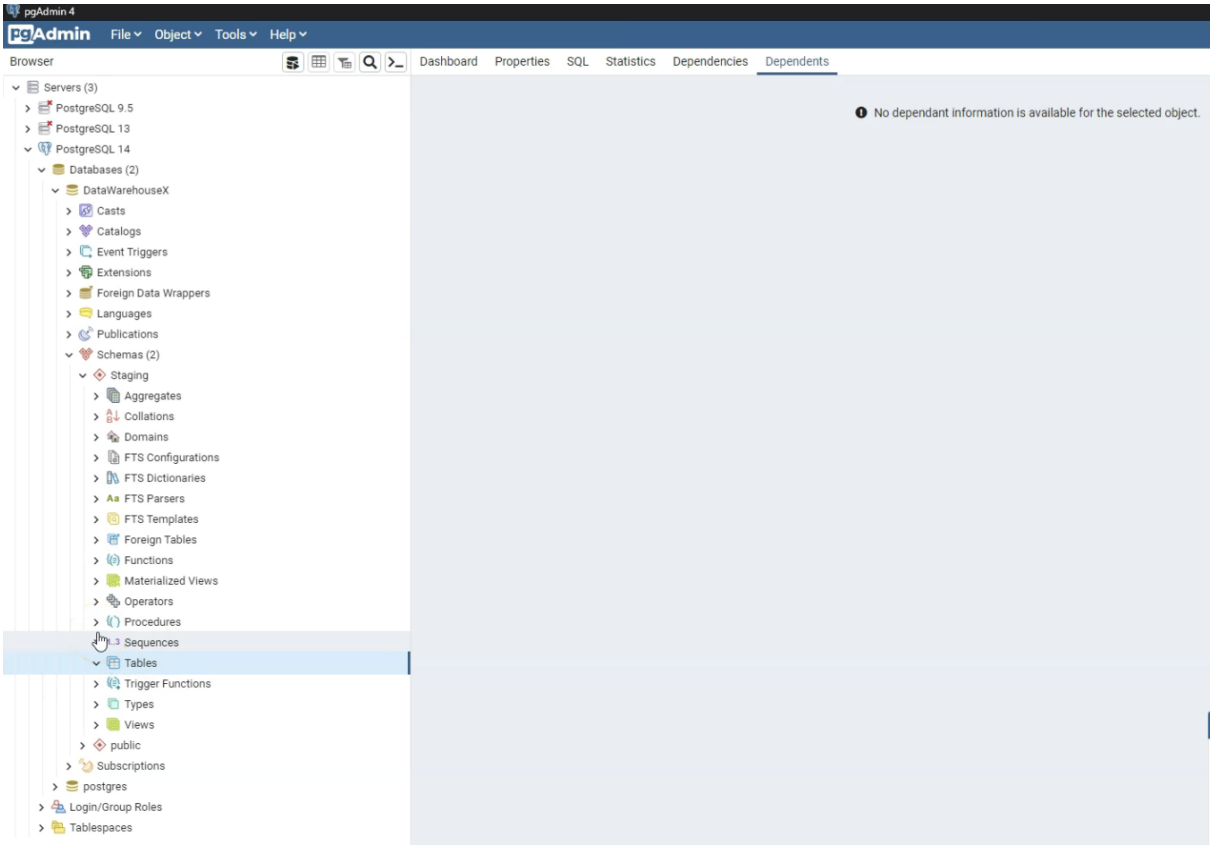
- Select previous step to see the data at that step i.e. csv input step before the string upper case transformation



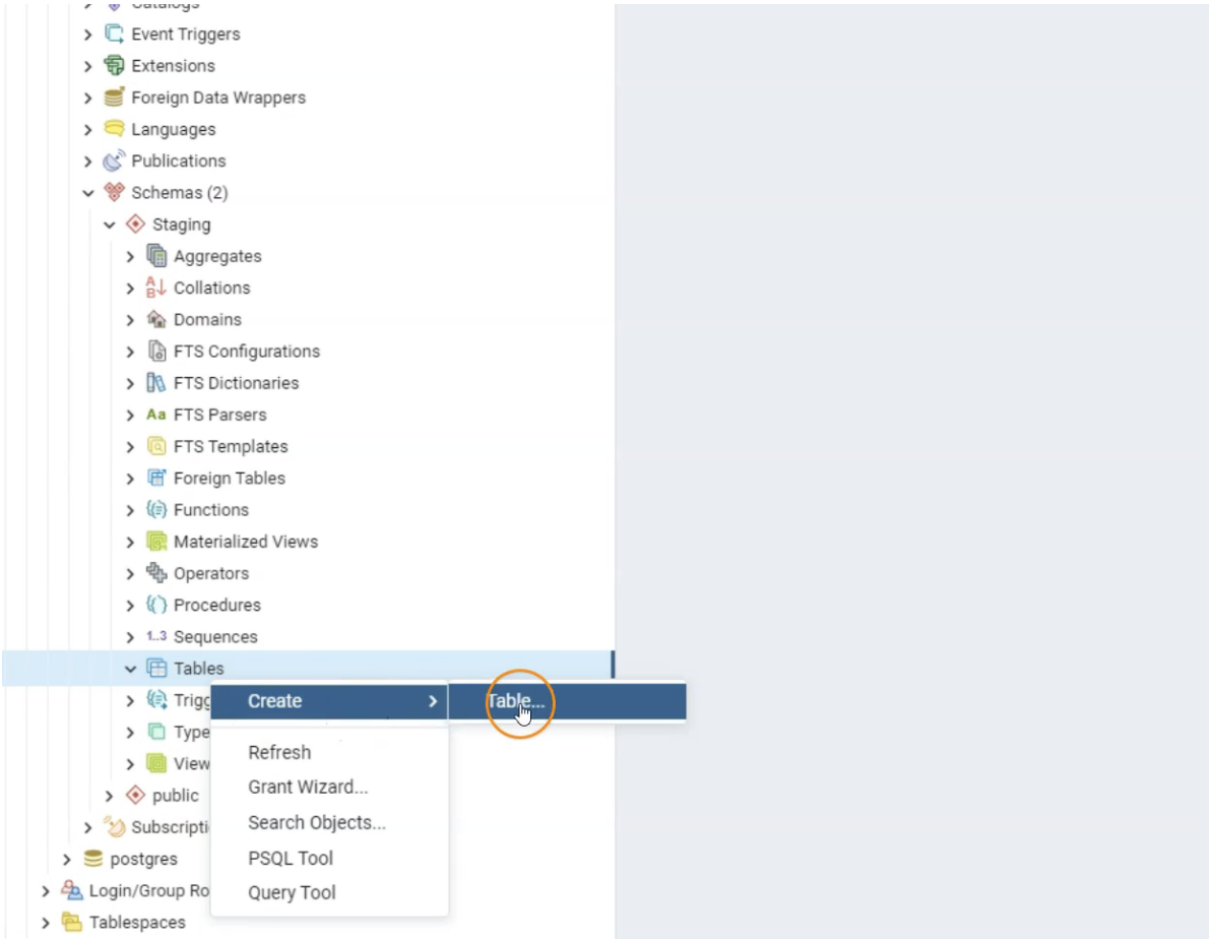
#	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE
1	10107	30	95.7	2	2871.0	2/24/2003 0:00	Shipped	1	2	2003	Motorcycles	95	S10_1678
2	10121	34	81.35	5	2765.9	5/7/2003 0:00	Shipped	2	5	2003	Motorcycles	95	S10_1678
3	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	7	2003	Motorcycles	95	S10_1678
4	10145	45	83.26	6	3746.7	8/25/2003 0:00	Shipped	3	8	2003	Motorcycles	95	S10_1678
5	10159	49	100.0	14	5205.27	10/10/2003 0:00	Shipped	4	10	2003	Motorcycles	95	S10_1678
6	10168	36	96.66	1	3479.76	10/28/2003 0:00	Shipped	4	10	2003	Motorcycles	95	S10_1678
7	10180	29	86.13	9	2497.77	11/11/2003 0:00	Shipped	4	11	2003	Motorcycles	95	S10_1678
8	10188	48	100.0	1	5512.32	11/18/2003 0:00	Shipped	4	11	2003	Motorcycles	95	S10_1678
9	10201	22	98.57	2	2168.54	12/1/2003 0:00	Shipped	4	12	2003	Motorcycles	95	S10_1678
10	10211	41	100.0	14	4708.44	1/15/2004 0:00	Shipped	1	1	2004	Motorcycles	95	S10_1678
11	10223	37	100.0	1	3965.66	2/20/2004 0:00	Shipped	1	2	2004	Motorcycles	95	S10_1678
12	10237	23	100.0	7	2333.12	4/5/2004 0:00	Shipped	2	4	2004	Motorcycles	95	S10_1678

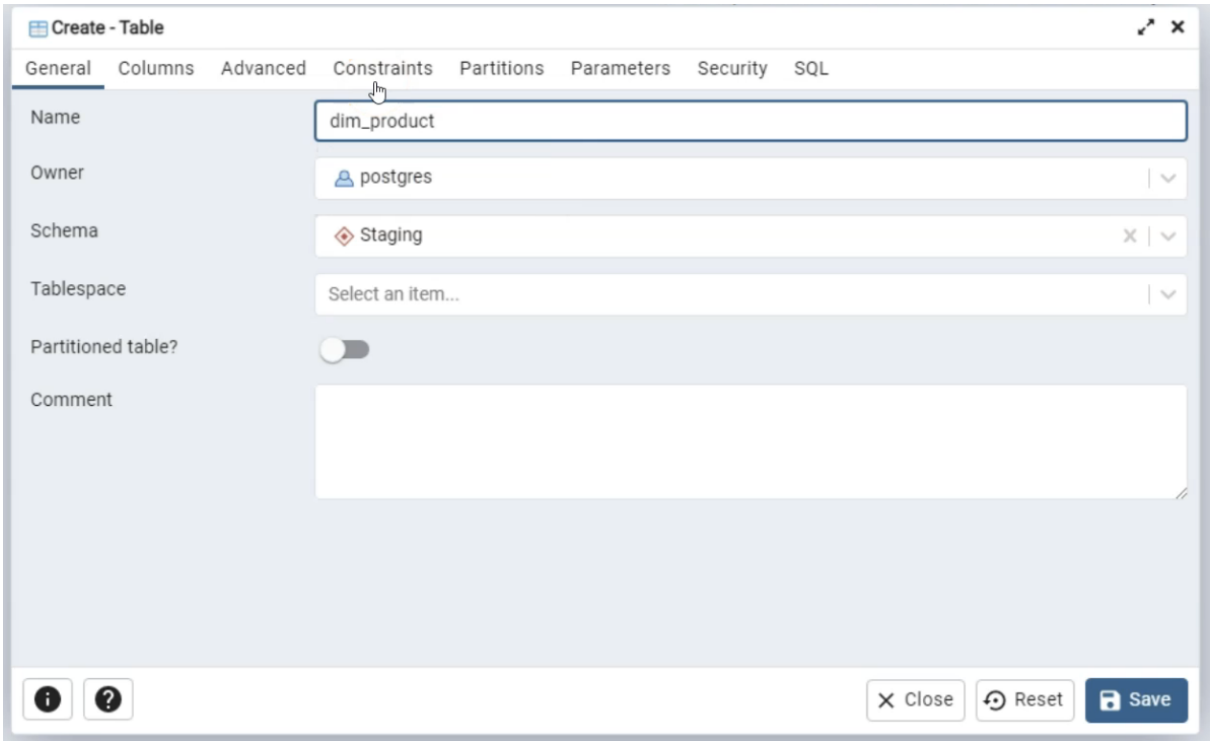
Demo: Setting up tables in SQL

- We want to load the data from a source and place it in the staging layer
- Go to postgresql, here we do not have any tables yet

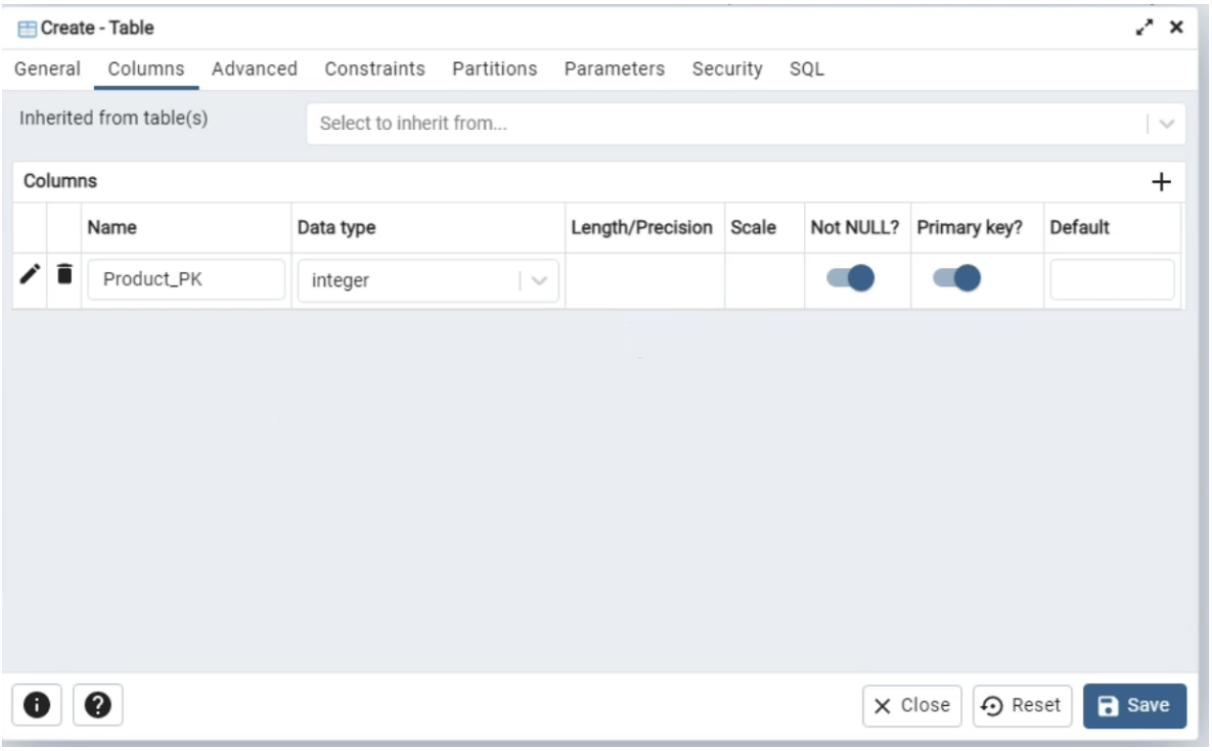


- Right click on Table and create tables

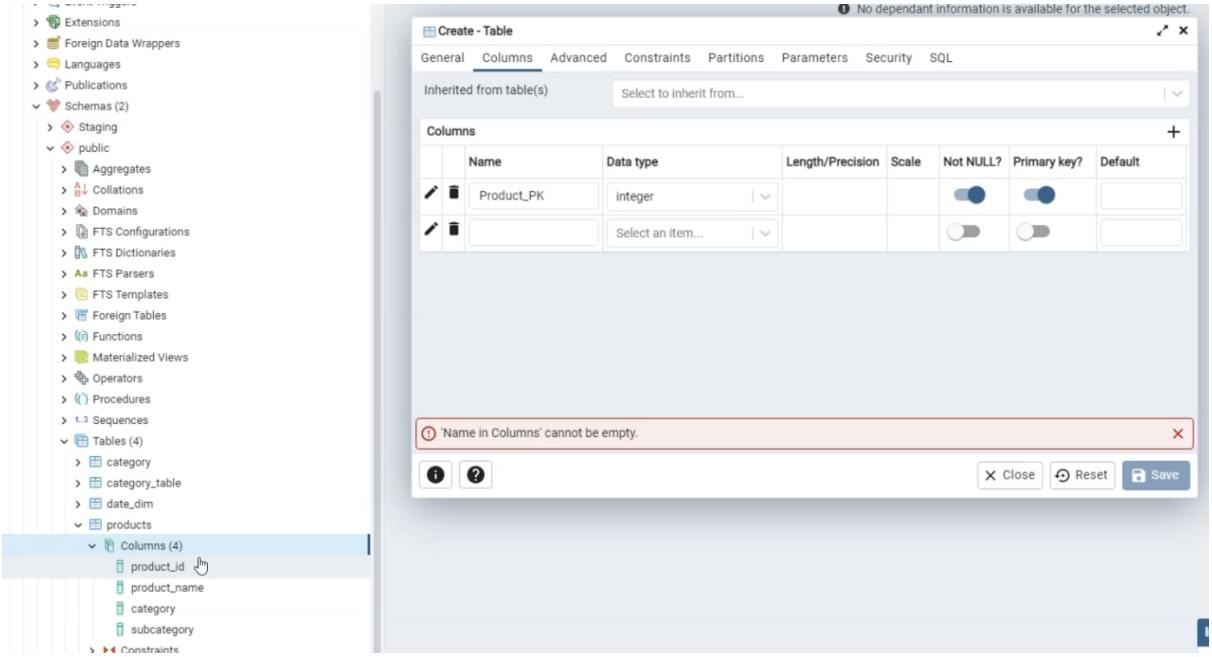


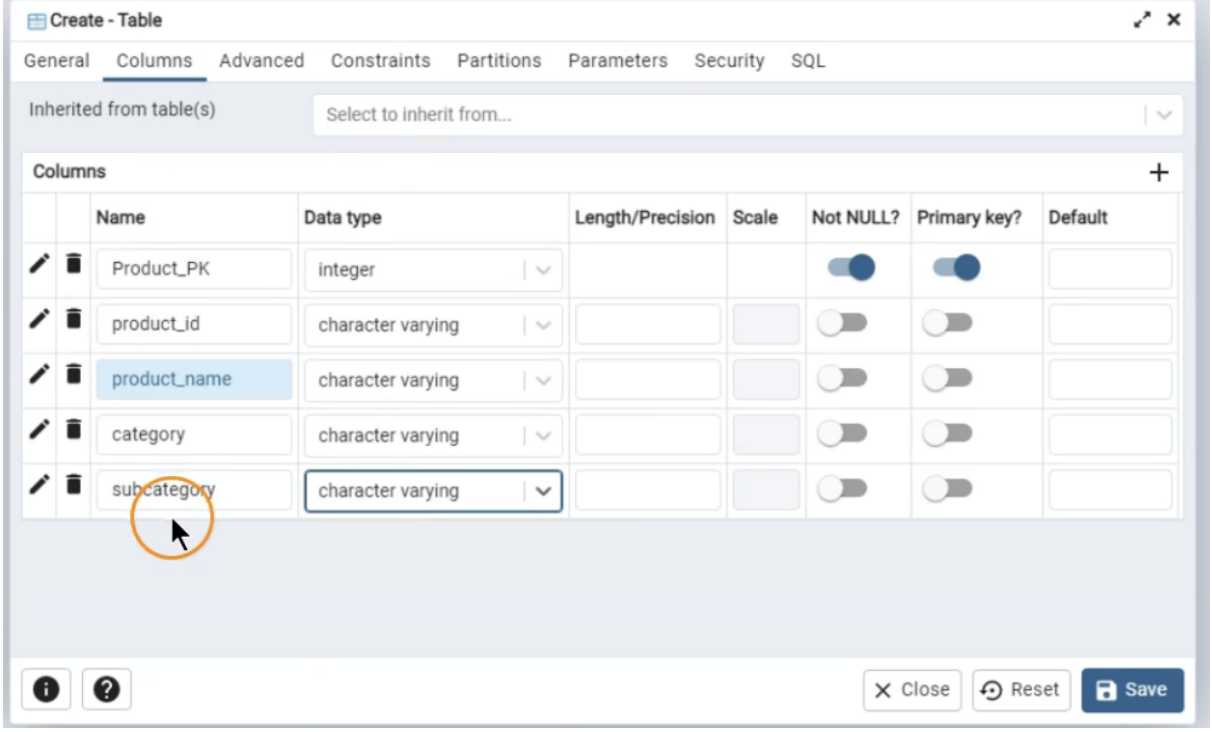


- Click on the + icon and add the columns for the dimenion table

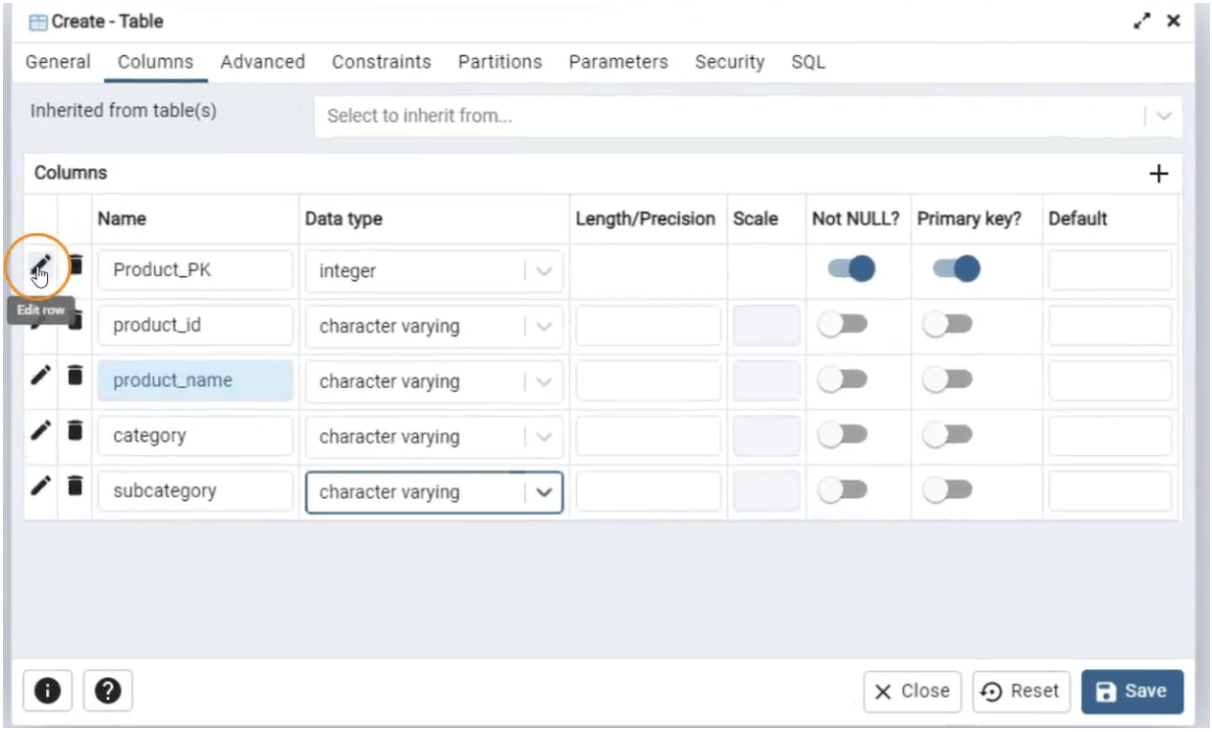


- We can refer the source data and add the columns to our dimenion table

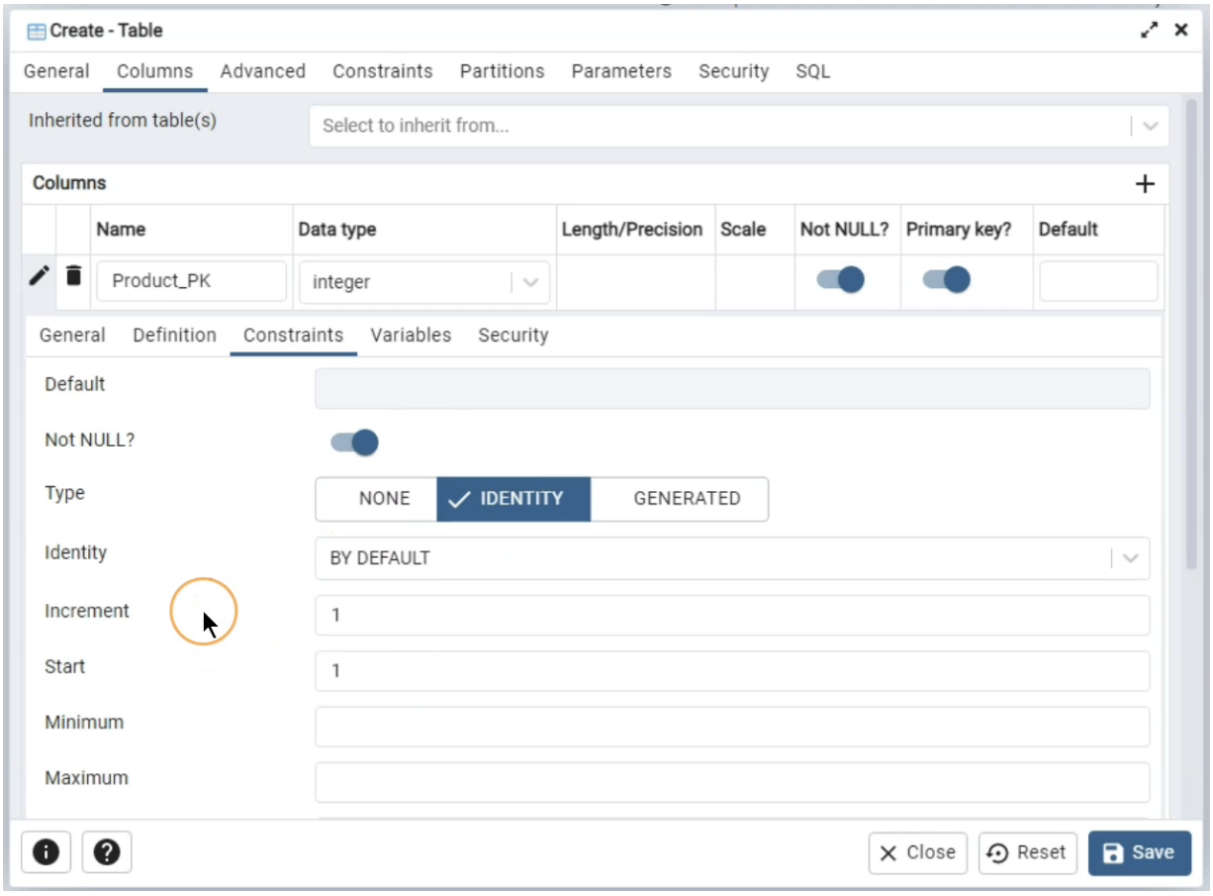


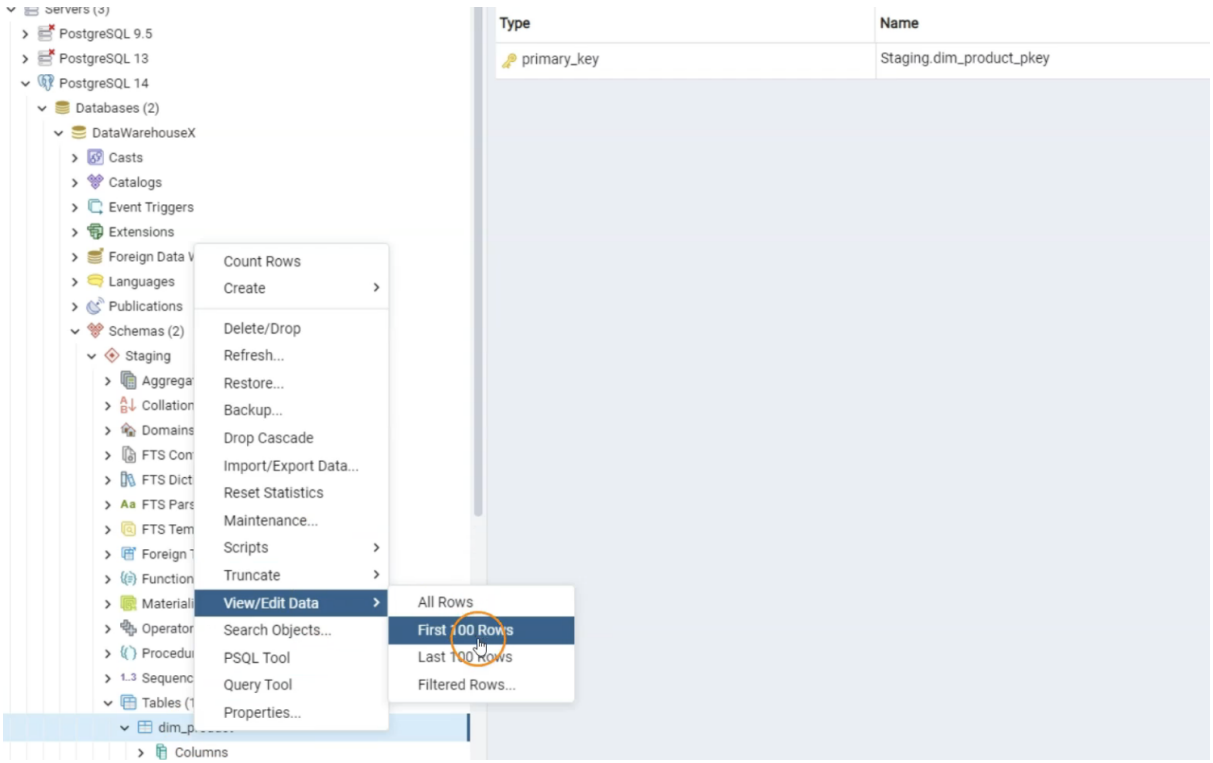


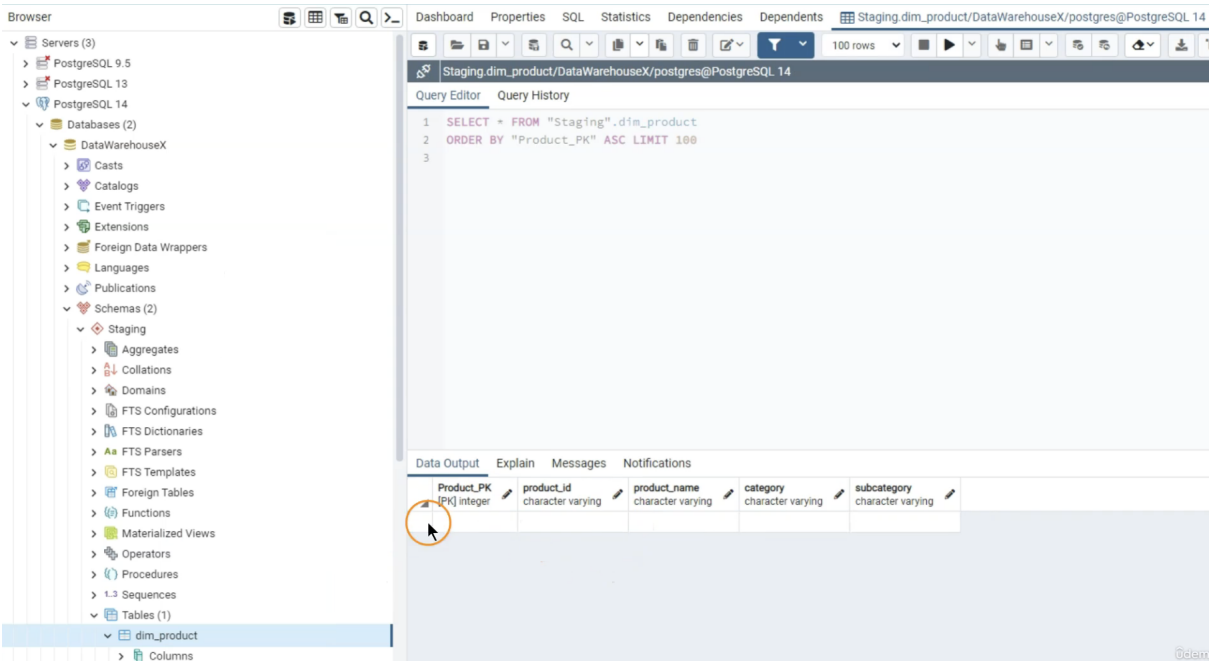
- We can click on the Edit icon to give default values



- For PK we can go to Constraints and select IDENTITY. This will automatically add the incremented PK value to the rows

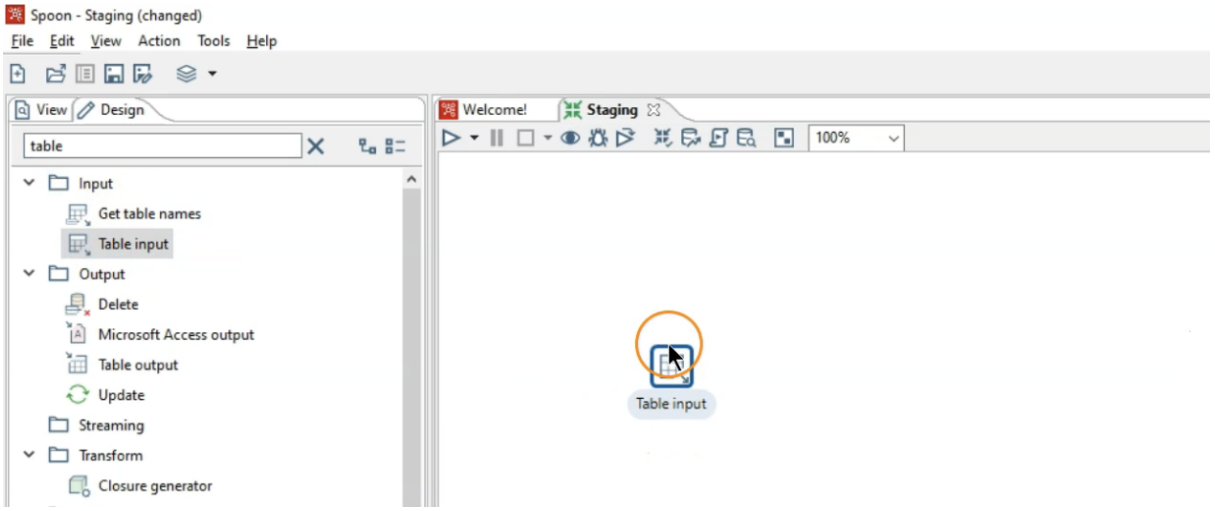

Create - Table
General
Columns
Advanced
Constraints
Partitions
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary key?
Default
Product_PK
integer
General
Definition
Constraints
Variables
Security
Default
Not NULL?
Type
NONE
IDENTITY
GENERATED
Identity
BY DEFAULT
Increment
1
Start
1
Minimum
Maximum
Close
Reset
Save


PostgreSQL 9.5
PostgreSQL 13
PostgreSQL 14
Databases (2)
DataWarehouseX
Casts
Catalogs
Event Triggers
Extensions
Languages
Publications
Schemas (2)
Staging
Count Rows
Create
Delete/Drop
Refresh...
Restore...
Backup...
Drop Cascade
Import/Export Data...
Reset Statistics
Maintenance...
Scripts
Truncate
View/Edit Data
Search Objects...
PSQL Tool
Query Tool
Properties...
All Rows
First 100 Rows
Last 100 Rows
Filtered Rows...
Columns
Type
Name
primary_key
Staging.dim_product_pkey

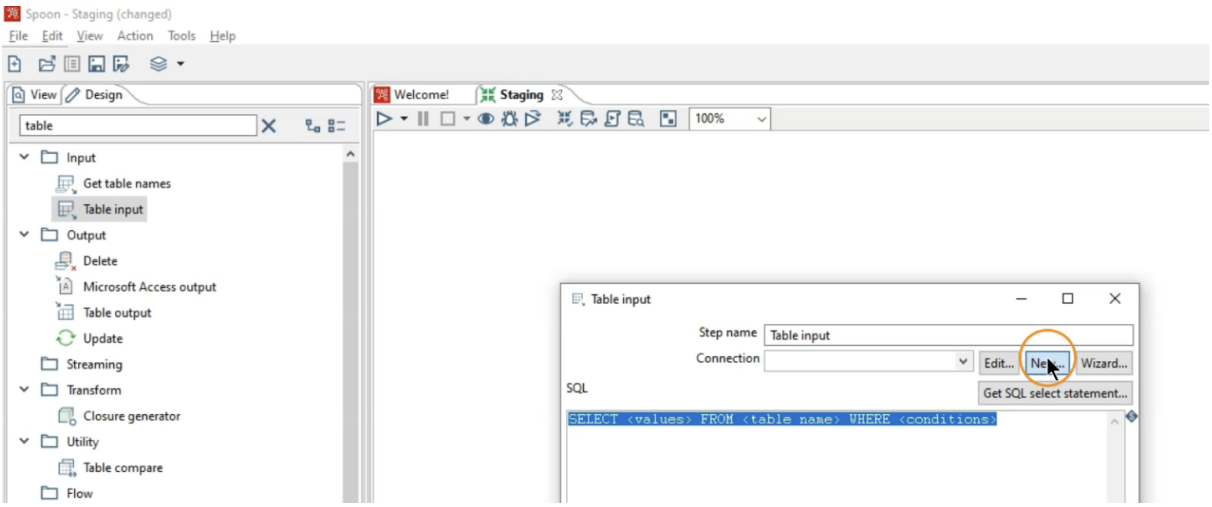


Demo: Initial Load example

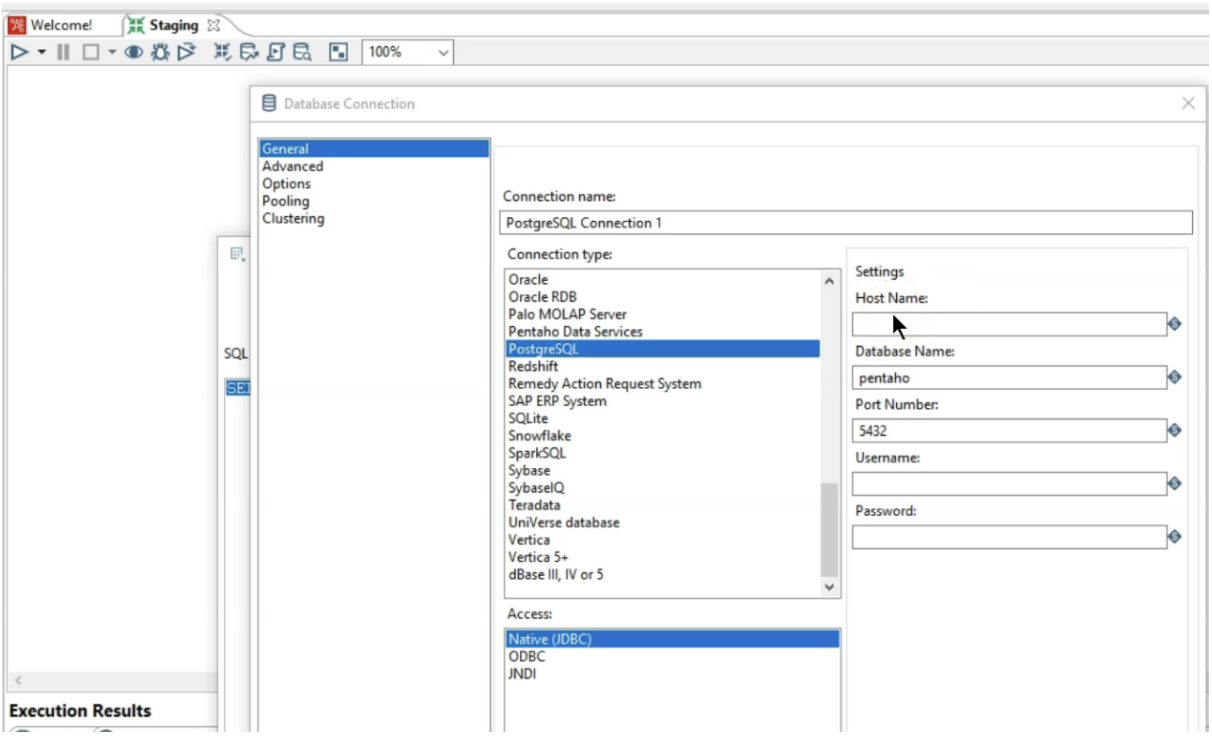
- Now we want to read the data from source, make little to no transformation and load the data into staging layer
- Now let's set up PDI to do this task
- Go to PDI and search for Table in search bar and drag it to the area



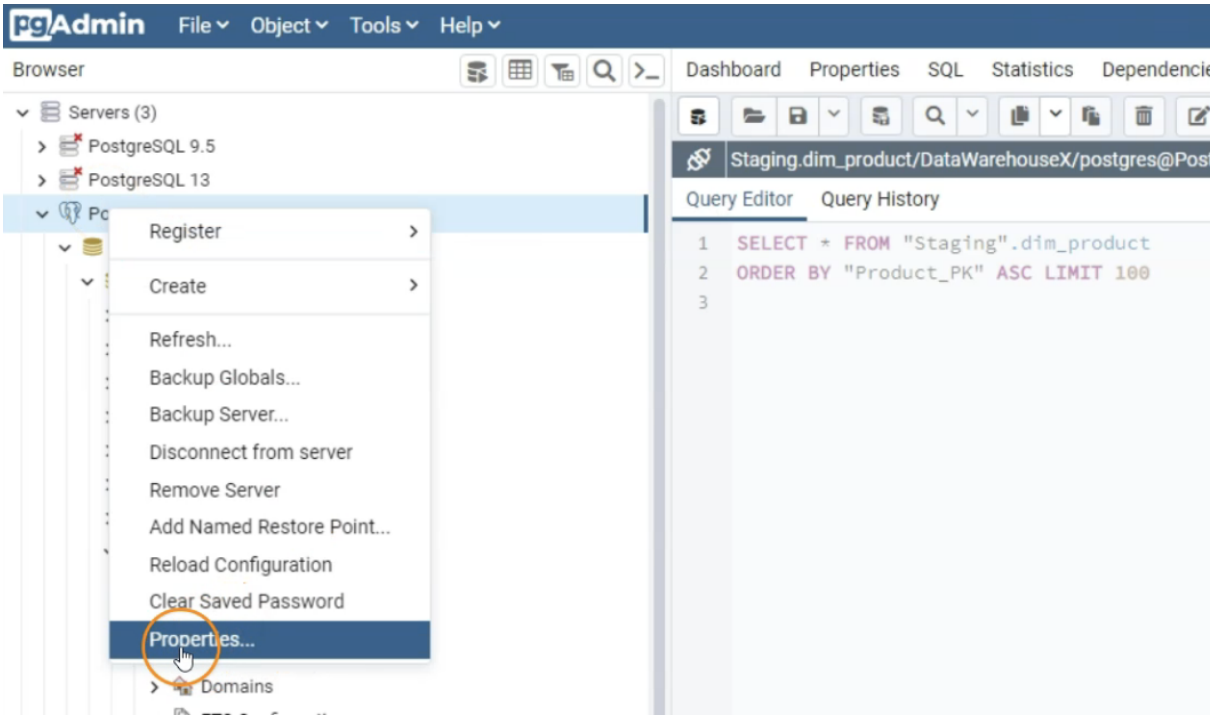
- Double click the Table Input icon to set it up, we can see the SQL query to read the data



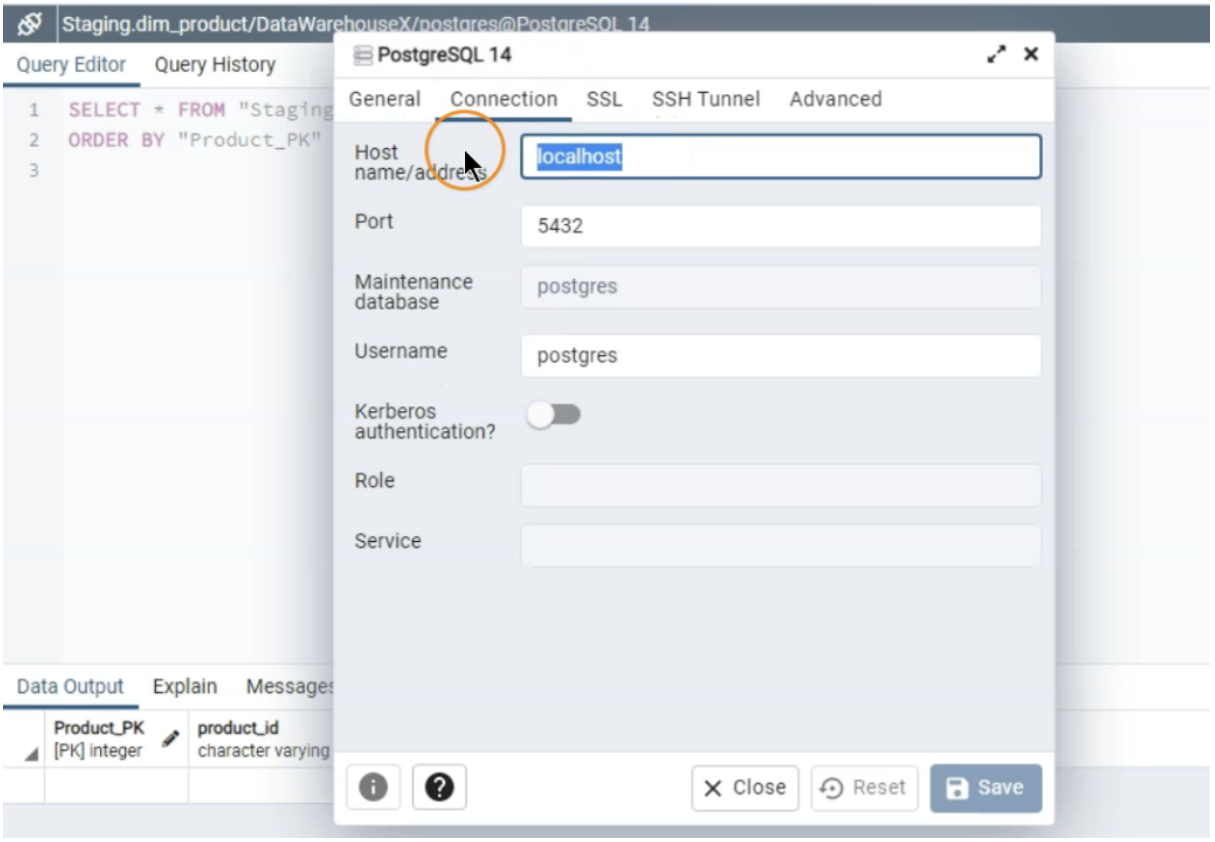
- To do any operation, we first have to set up the connection, click on New

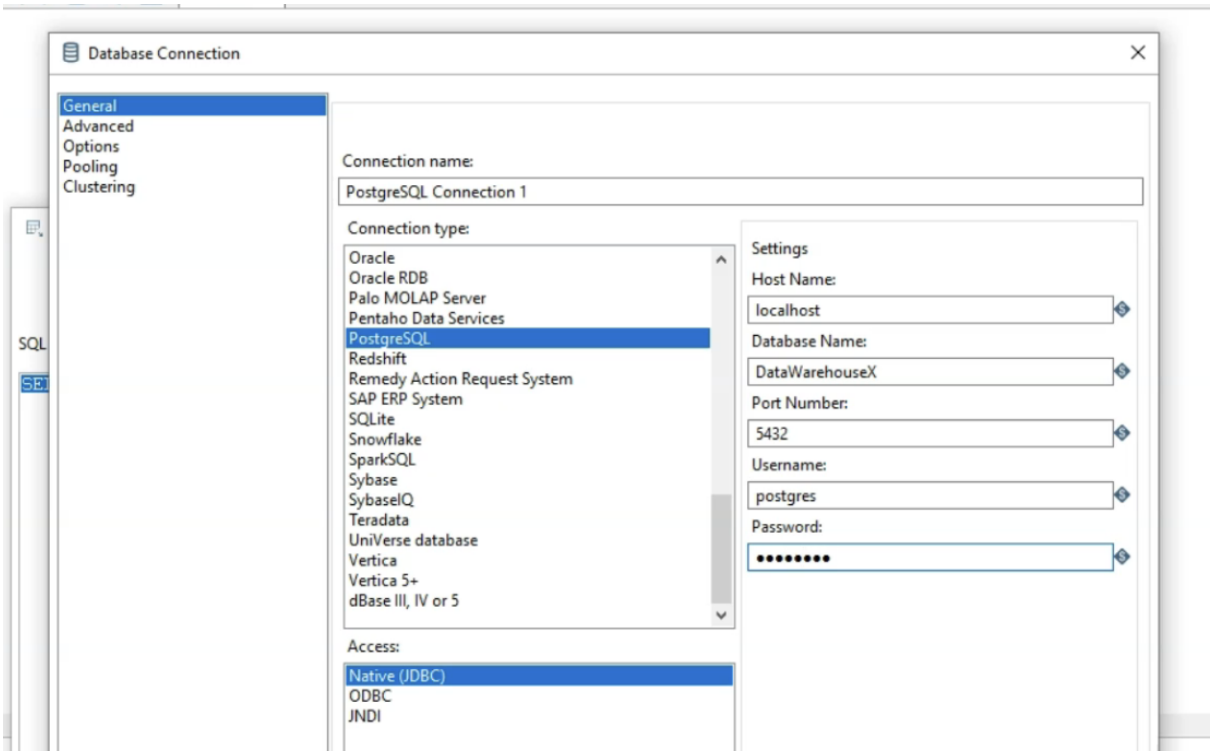


- Get the Database server details in pgAdmin
- Right click on the Postgresql> Properties

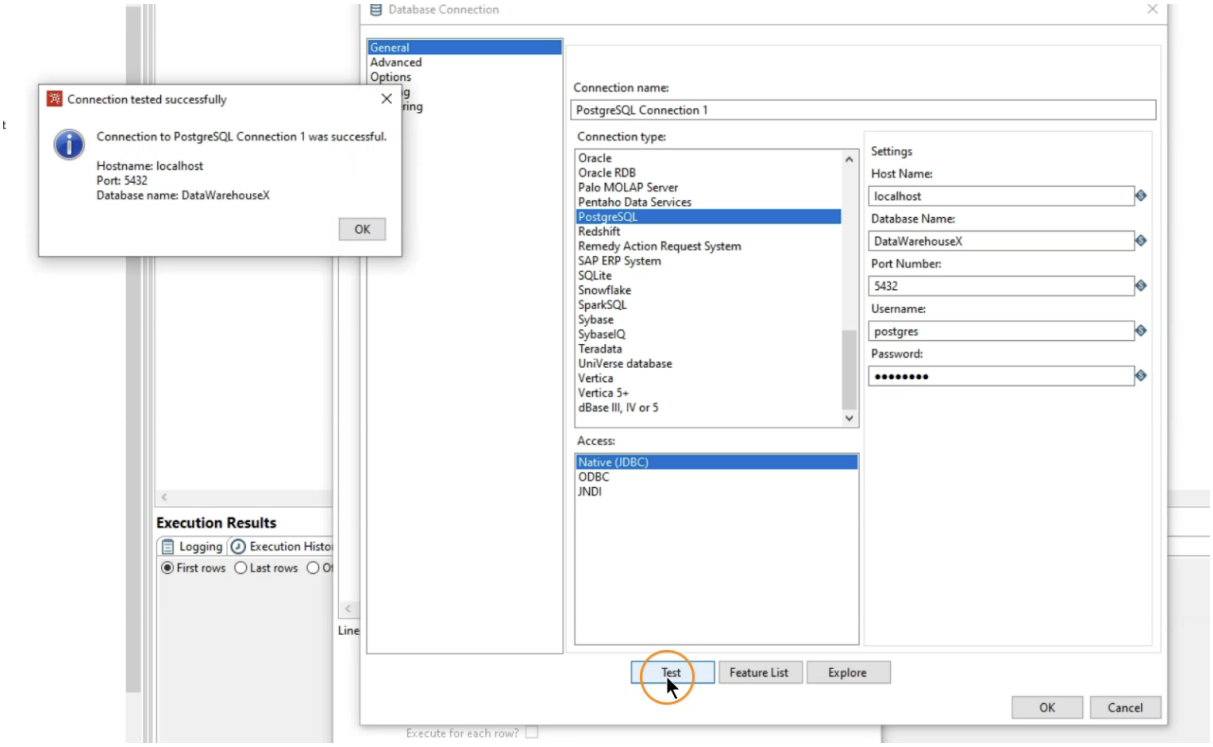


- Under Connection tab we find the sever connections

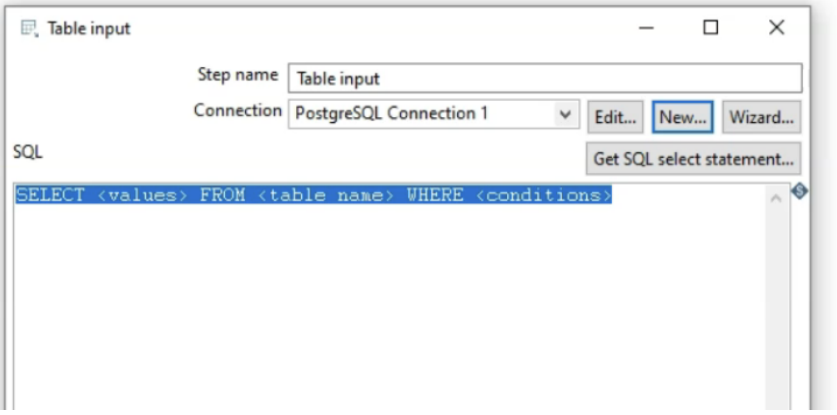




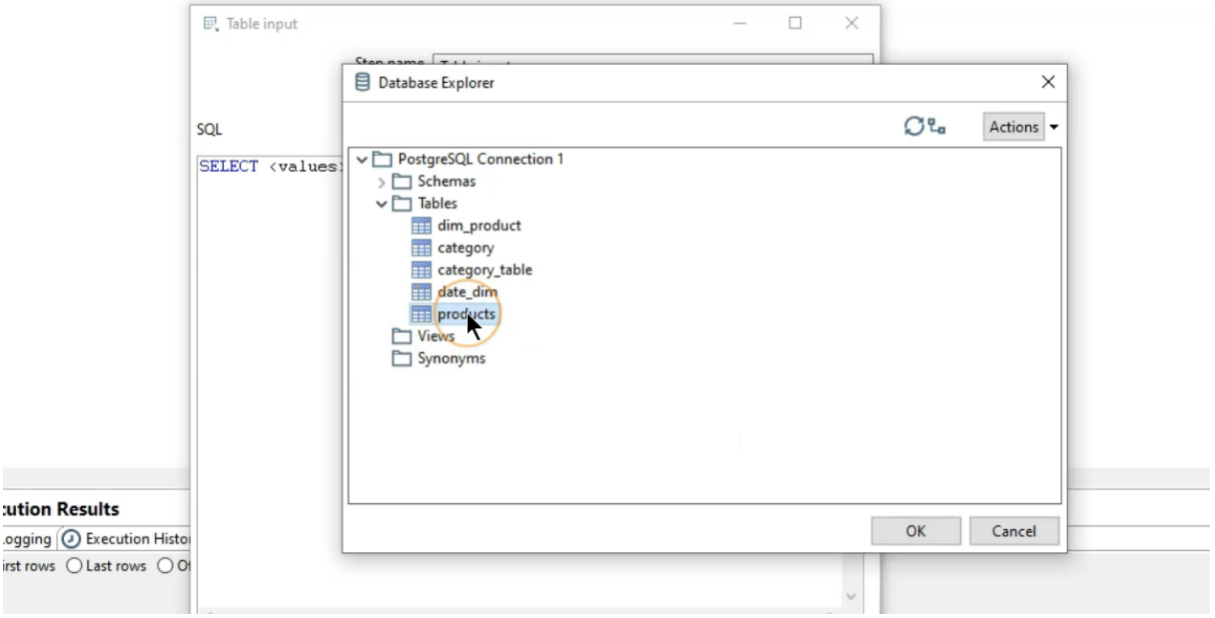
- Use the Test button to test the connection

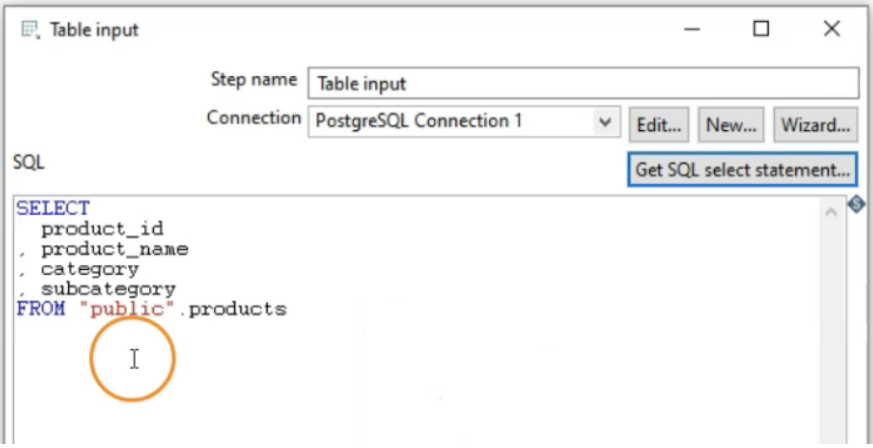


- Now we can see there is connection that is selected which we just created

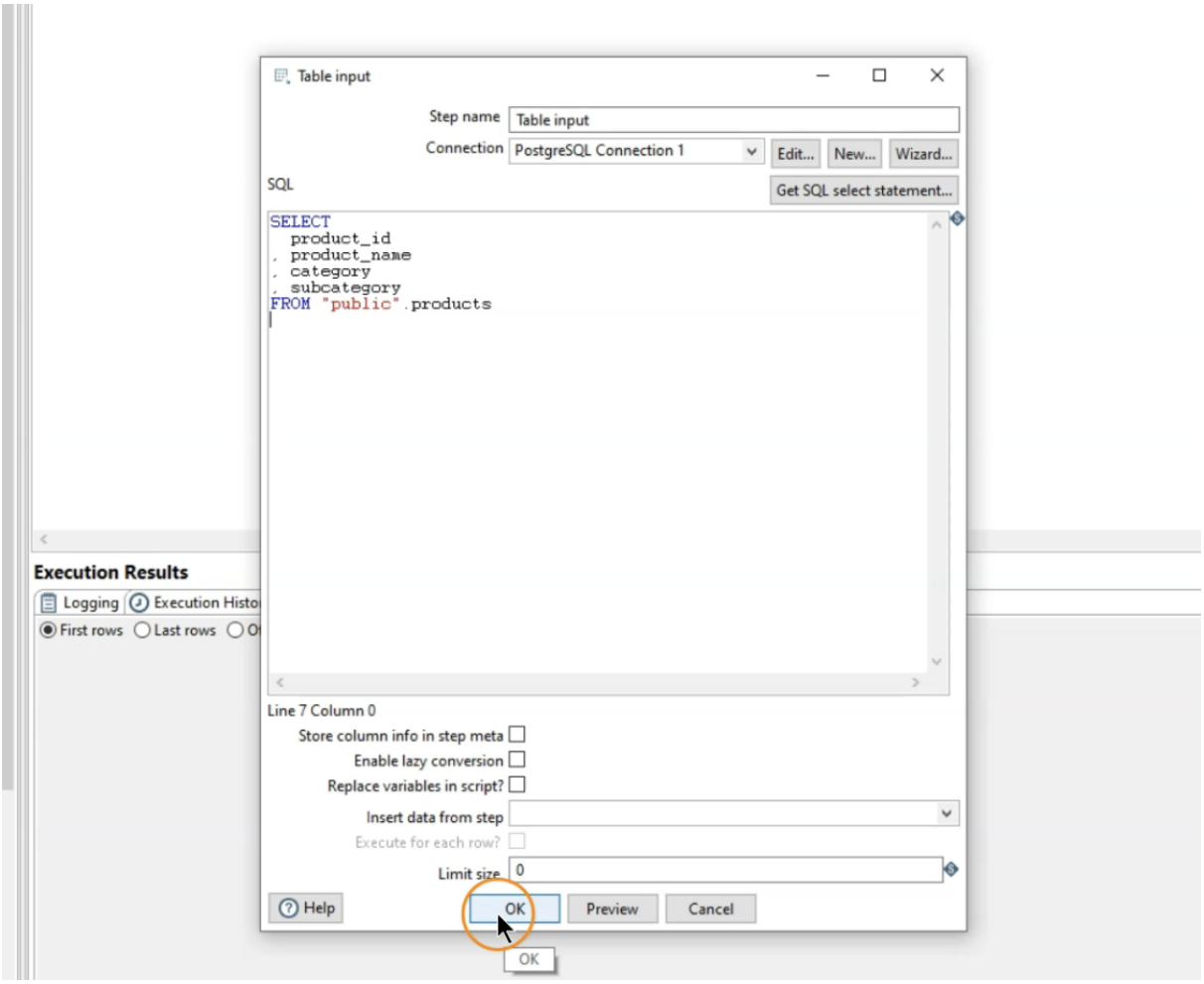


- We can write the query ourselves or click on Get SQL select statement button

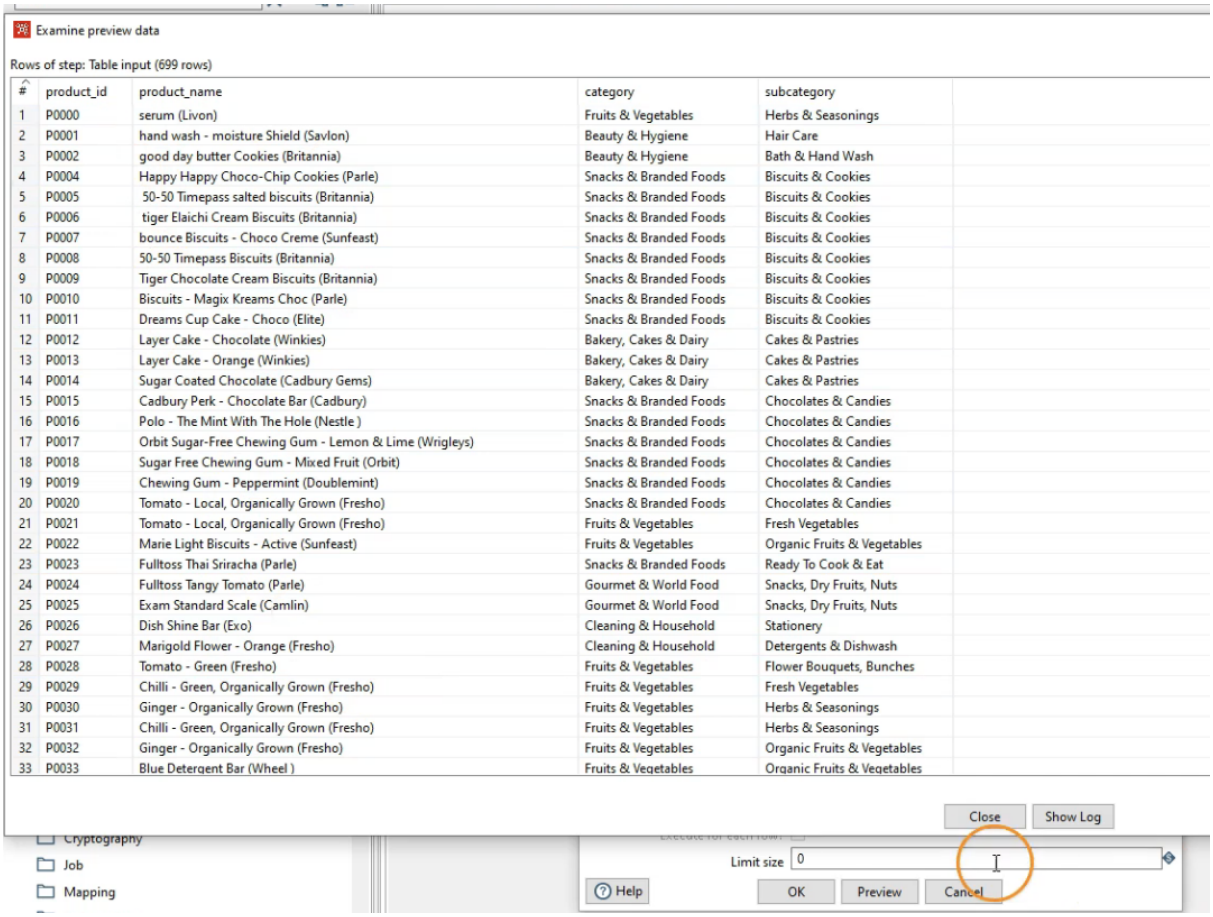




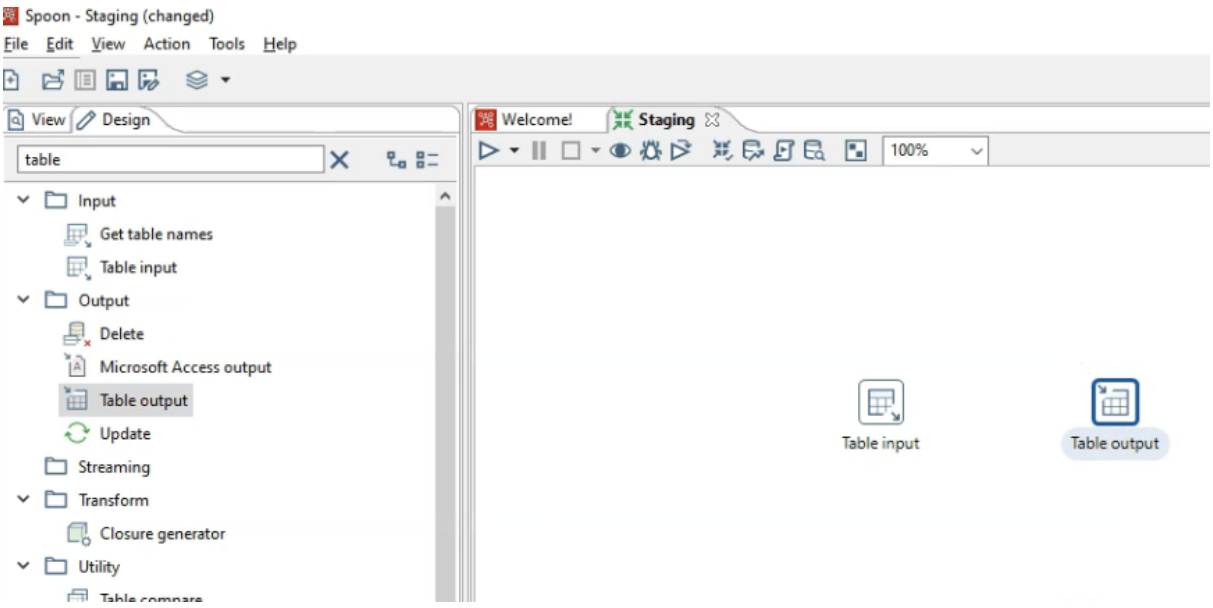
- We can also just use * i.e. SELECT * FROM “public”.products, but it is best practice to mention the column names



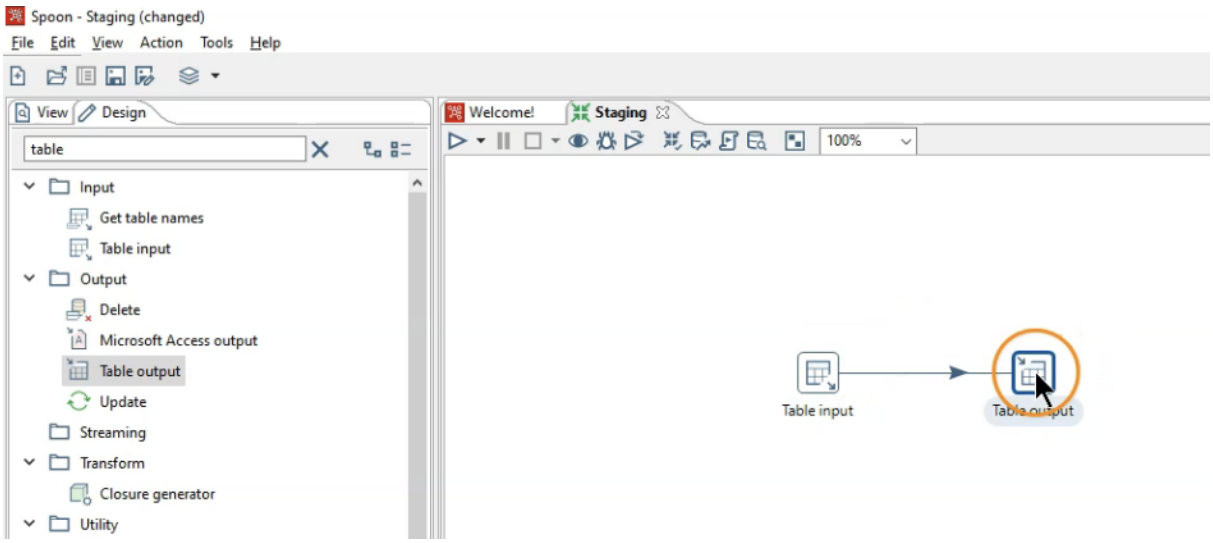
- We can also use the preview button and view the first few rows



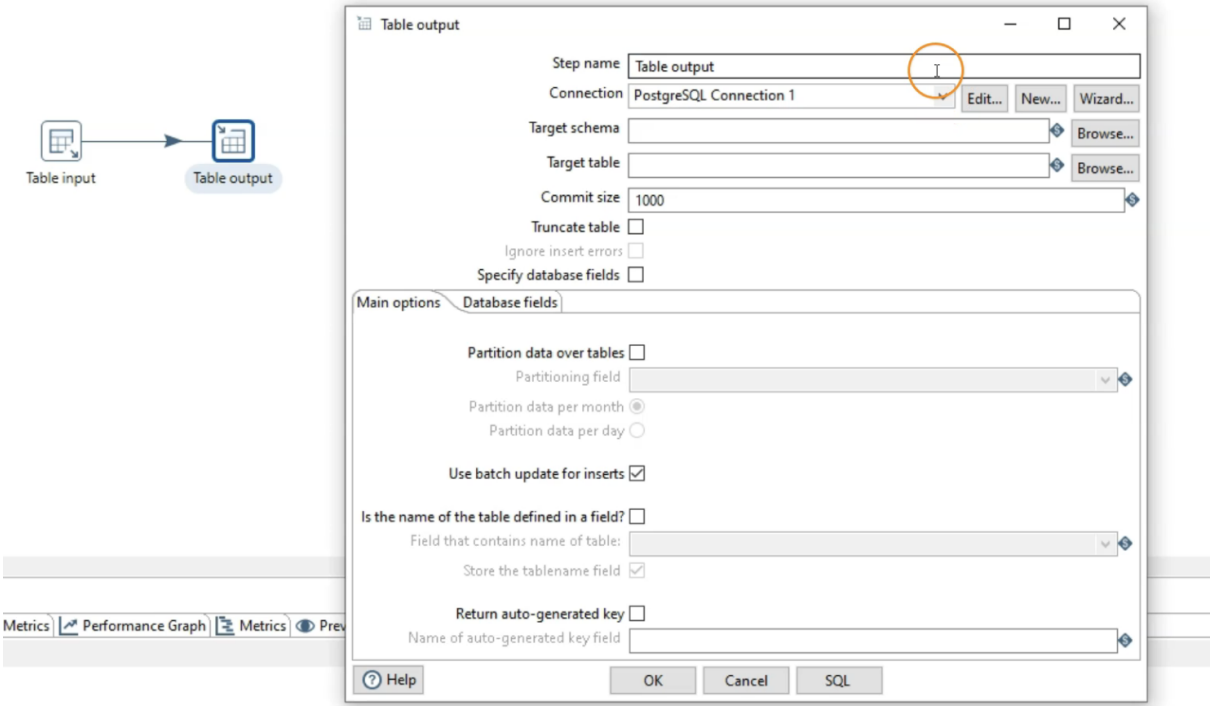
- This is just setting the read from source, now we have to write to the staging area table using Table Output in PDI



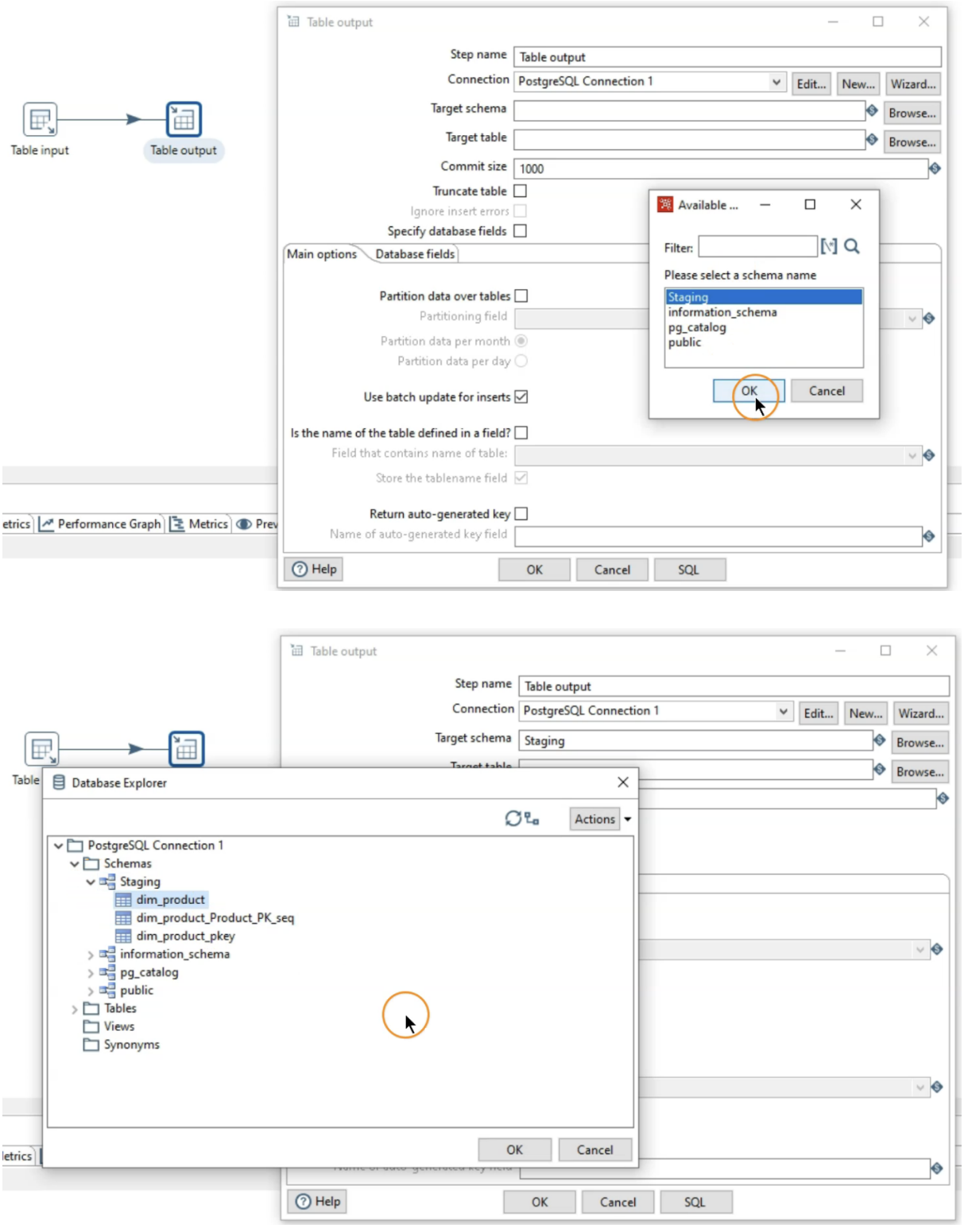
- Use the Shift Key and select Table input and drag it to Table Output to make the connection



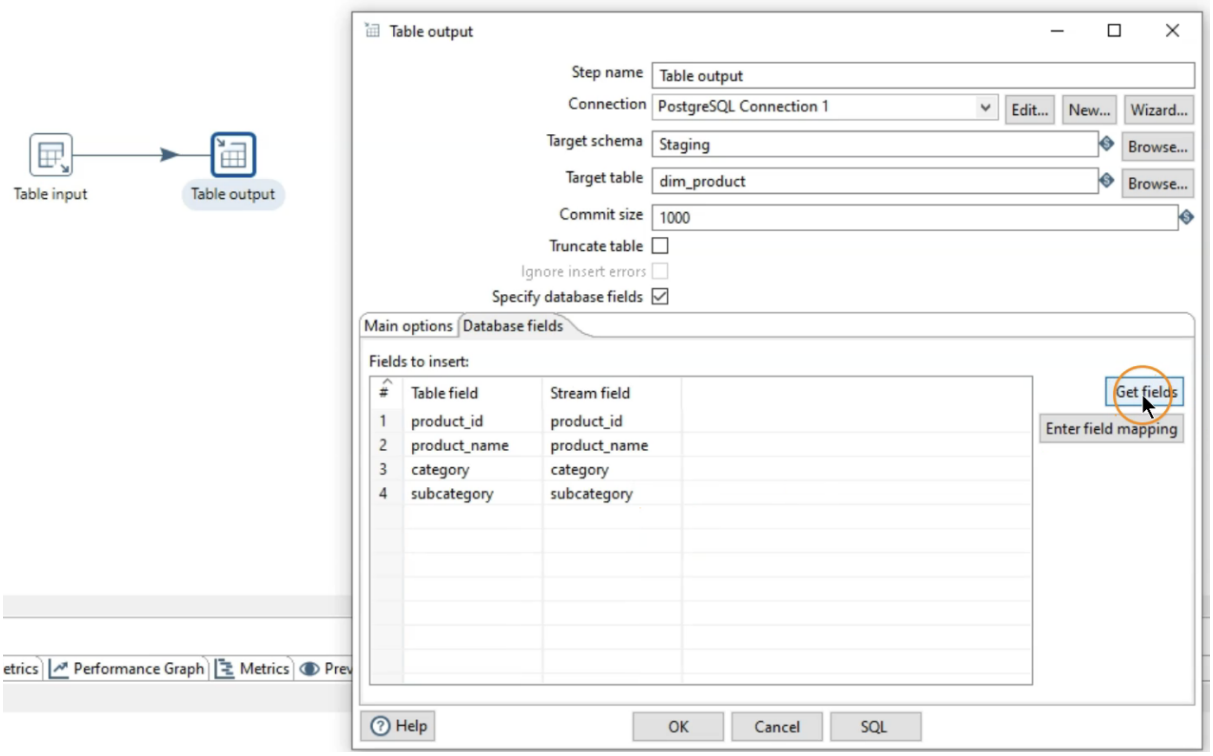
- Now double click the Table Output and set up the target table



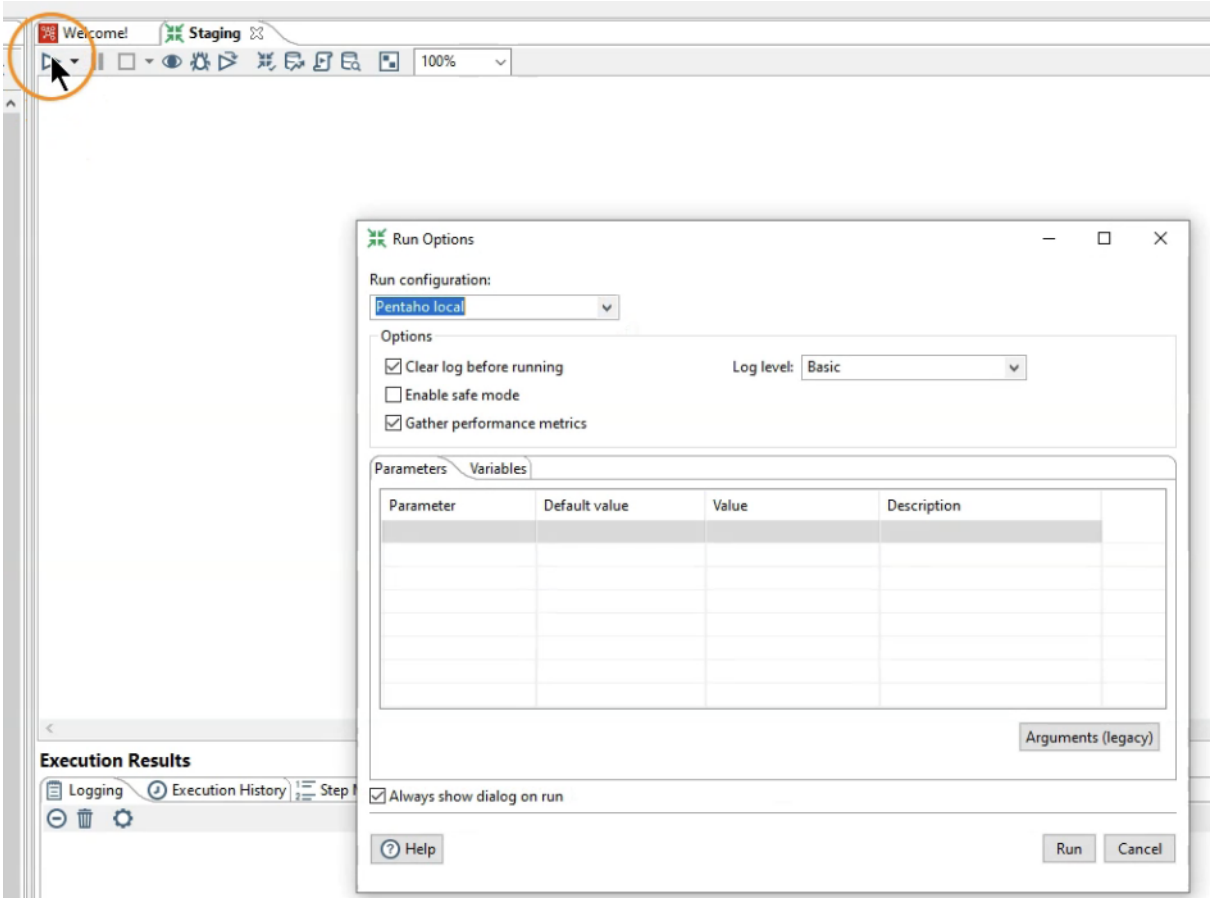
- Select the Target Schema and Target table



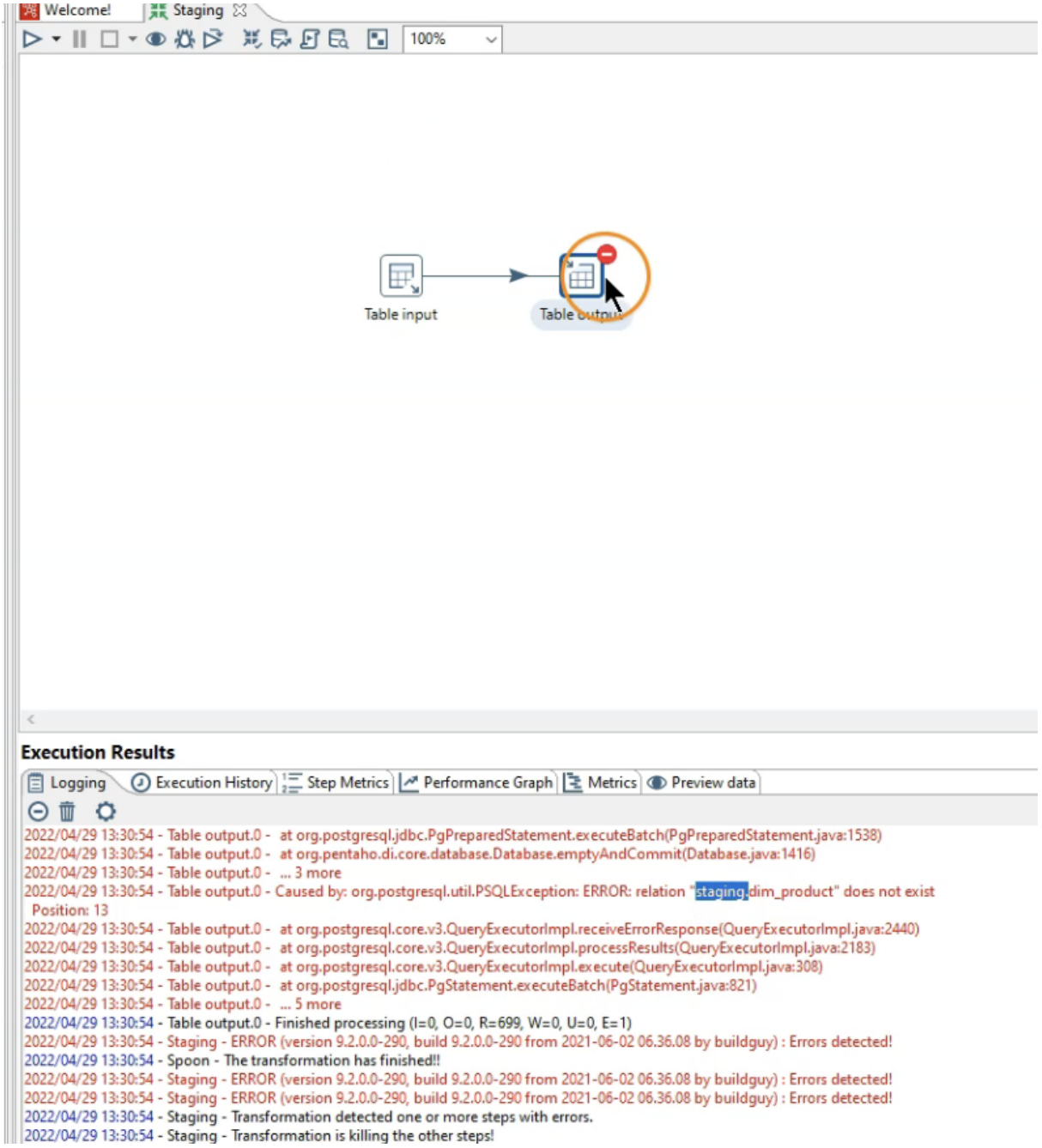
- We can also check Database fields and Click on Get database fields, to view the fields



- We can click on the field names and it gives a drop down, if the order is different and you have to map it to different columns between source and target table
- Click on Run button to run the process



- We have error because staging dim_product does not exist



- We have to use double quotes to make it case sensitive, else by default it is not case sensitive

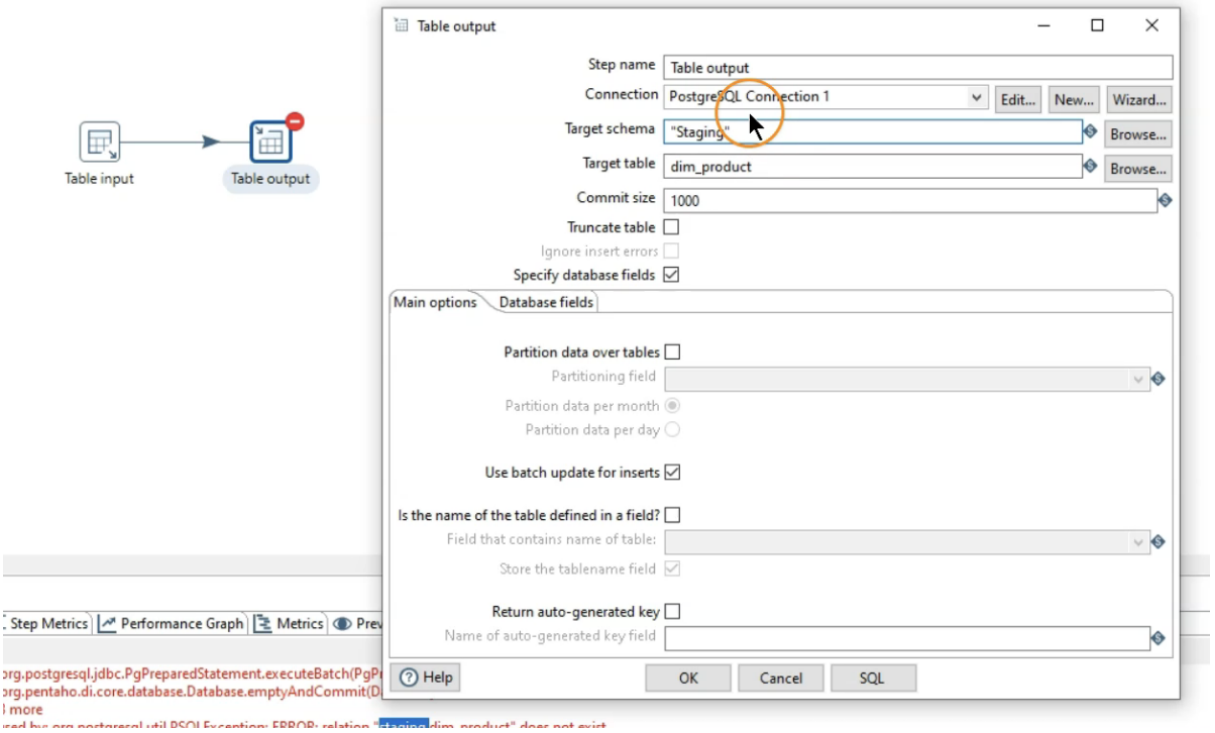
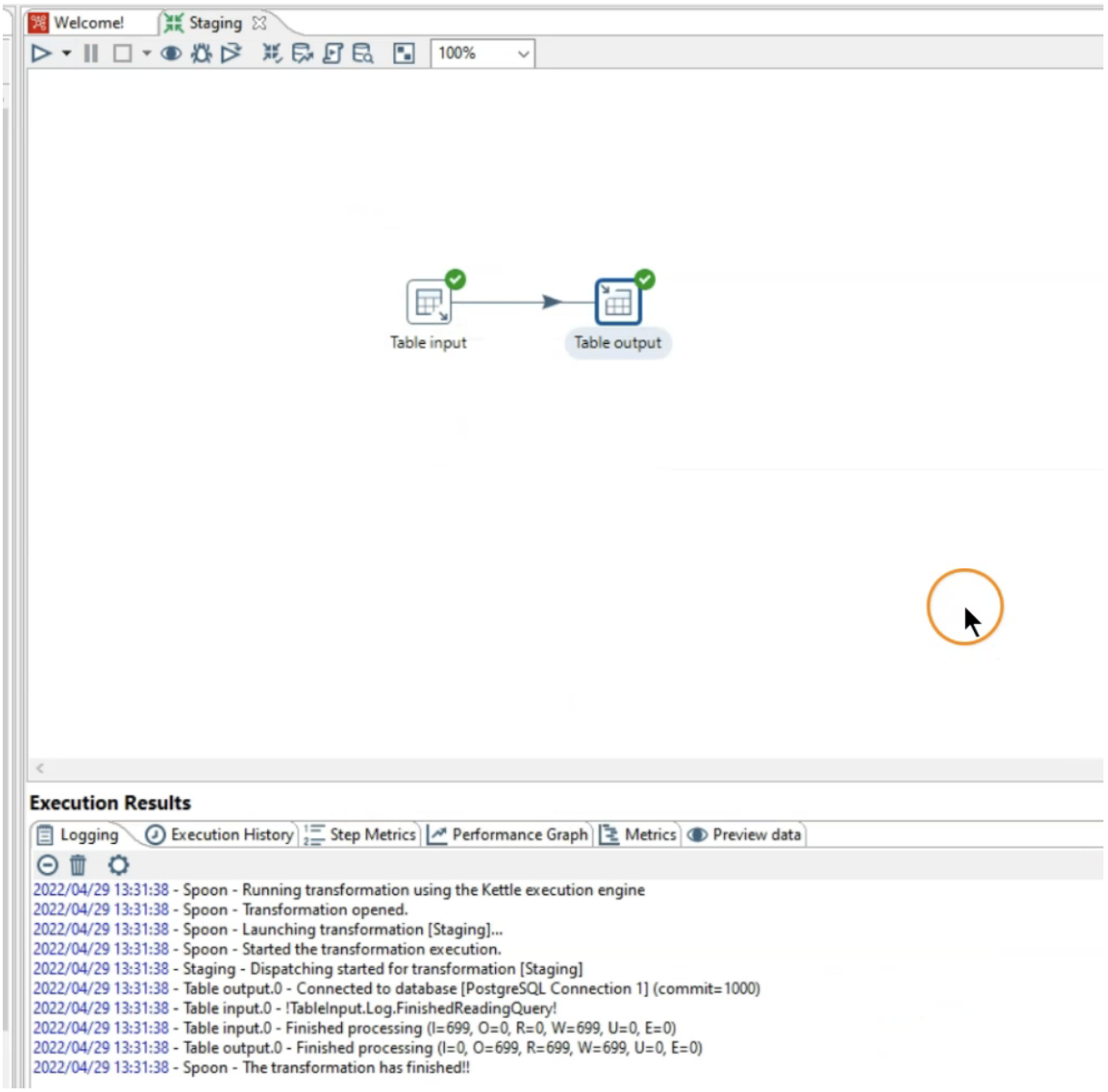
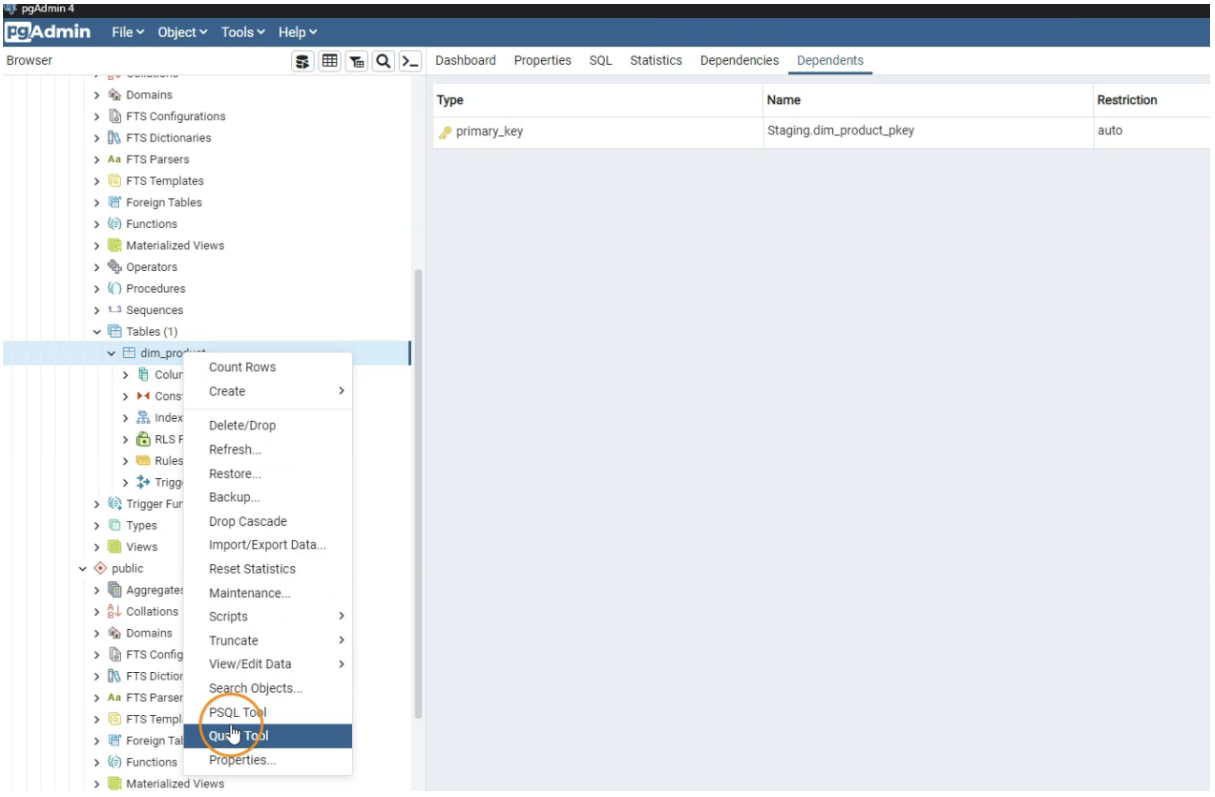
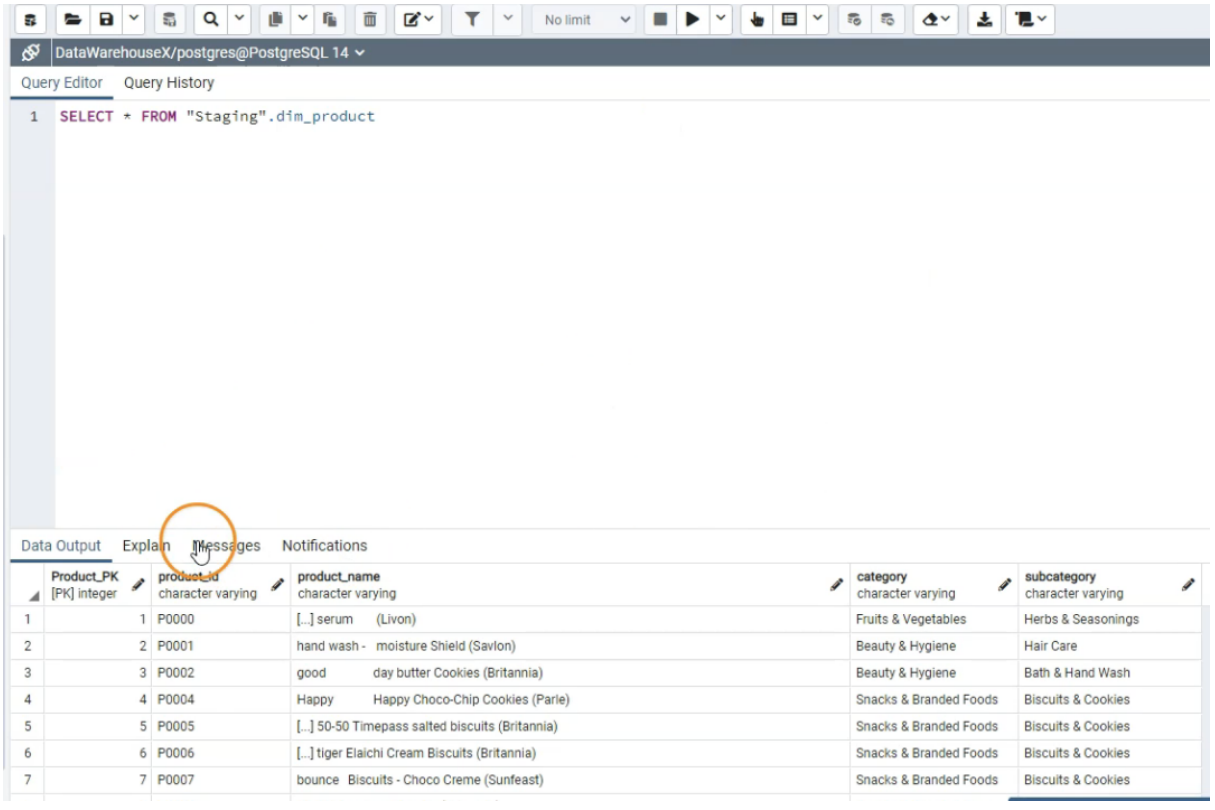


Table input
Table output
Table output
Step name
Table output
Connection
PostgreSQL Connection 1
Edit...
New...
Wizard...
Target schema
"Staging"
Browse...
Target table
dim_product
Browse...
Commit size
1000
Truncate table
Ignore insert errors
Specify database fields
Main options
Database fields
Partition data over tables
Partitioning field
Partition data per month
Partition data per day
Use batch update for inserts
Is the name of the table defined in a field?
Field that contains name of table:
Store the tablename field
Return auto-generated key
Name of auto-generated key field
Help
OK
Cancel
SQL
Step Metrics
Performance Graph
Metrics
org.postgresql.jdbc.PgPreparedStatement.executeBatch(PgP
org.pentaho.di.core.database.Database.emptyAndCommit(D
more

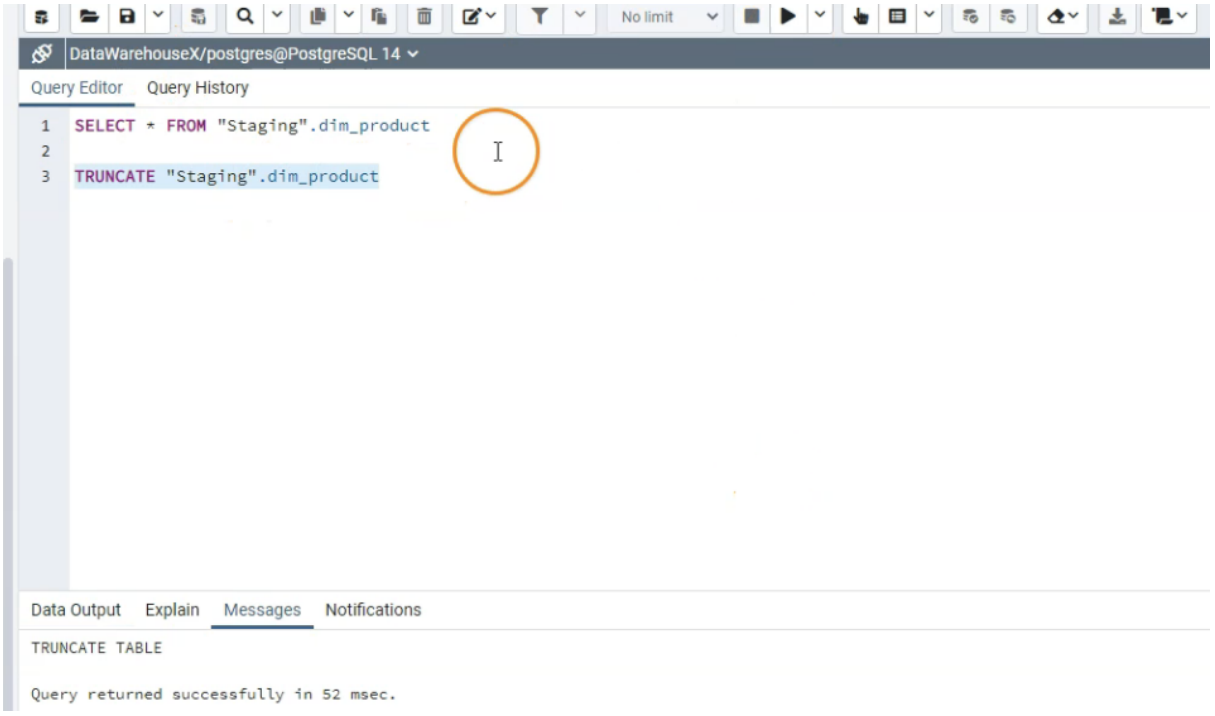


- Open query tool to view the data in the target table

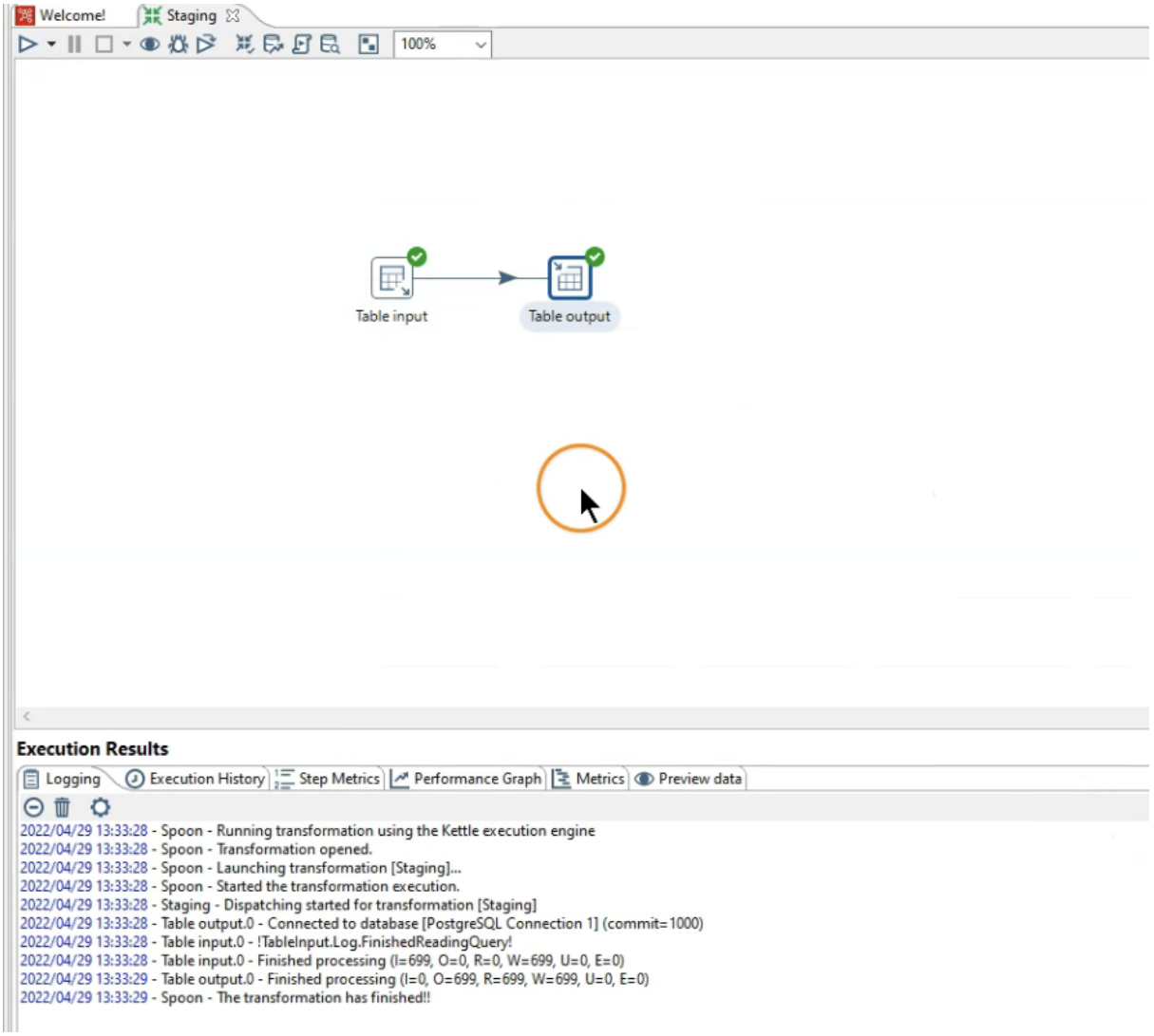


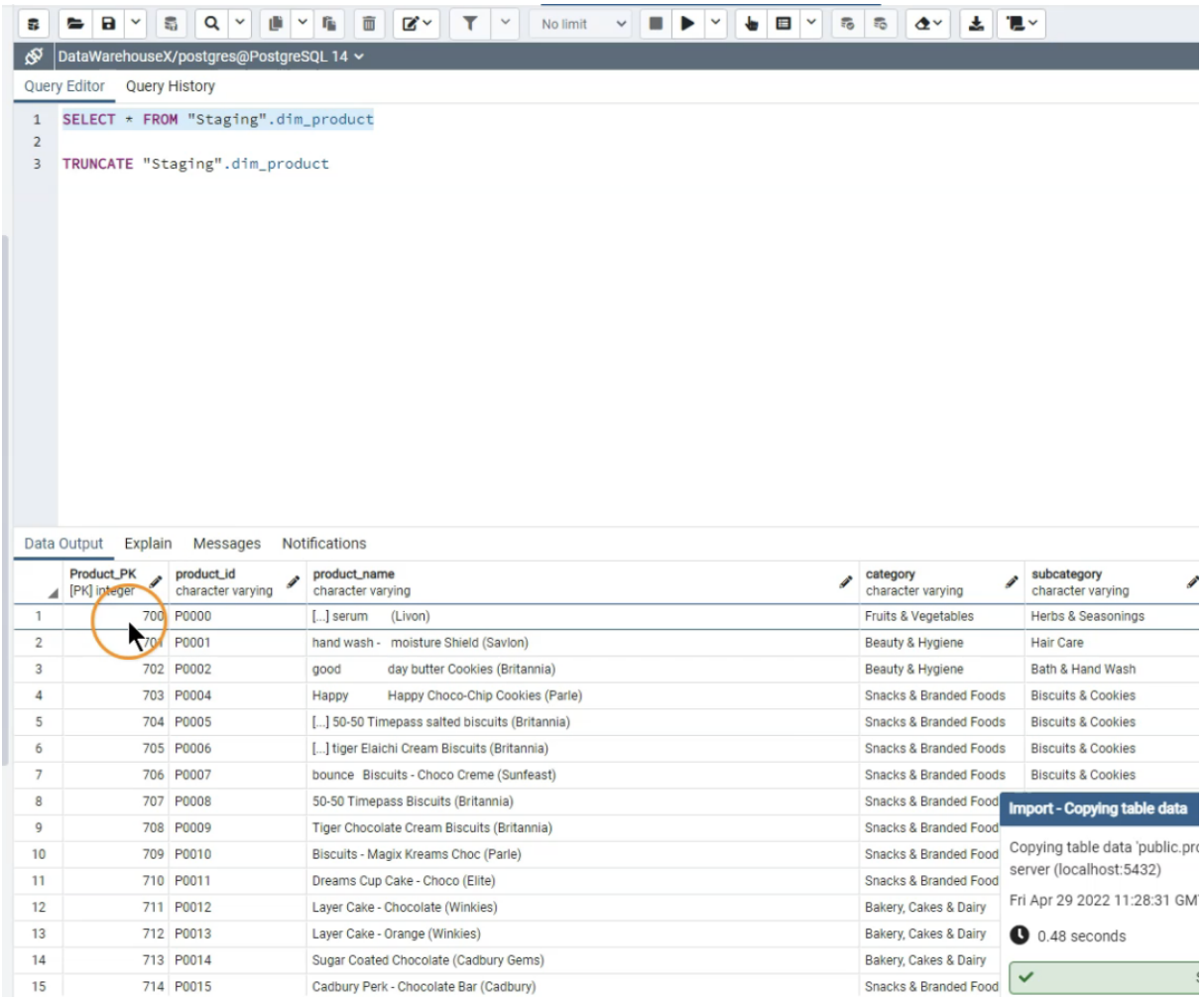


- Now we also have a primary key and even after we truncate the data which we do after each ETL process and the PK will start from where it left off and not from 1



- Run the transformation again


Welcome!
Staging
100%
Table input
Table output
Execution Results
Logging
Execution History
Step Metrics
Performance Graph
Metrics
Preview data
2022/04/29 13:33:28 - Spoon - Running transformation using the Kettle execution engine
2022/04/29 13:33:28 - Spoon - Transformation opened.
2022/04/29 13:33:28 - Spoon - Launching transformation [Staging]...
2022/04/29 13:33:28 - Spoon - Started the transformation execution.
2022/04/29 13:33:28 - Staging - Dispatching started for transformation [Staging]
2022/04/29 13:33:28 - Table output.0 - Connected to database [PostgreSQL Connection 1] (commit=1000)
2022/04/29 13:33:28 - Table input.0 - !TableInput.Log.FinishedReadingQuery!
2022/04/29 13:33:28 - Table input.0 - Finished processing (I=699, O=0, R=0, W=699, U=0, E=0)
2022/04/29 13:33:29 - Table output.0 - Finished processing (I=0, O=699, R=699, W=699, U=0, E=0)
2022/04/29 13:33:29 - Spoon - The transformation has finished!!



- The PK is not starting from 1

Demo: Delta load example

- This is usually done for Fact tables where we also have a delta column which is usually date time column using which we can identify what is the new row in the source data which we have not loaded into our datawarehouse
- Usually the delta load is done for Fact table but here for simplicity let us see the demo for the dimension table
- Let us take the product table, there can be new products that are added in the source data and we want to load only the new data that is not already loaded before
- Will be using Product_ID as our delta column

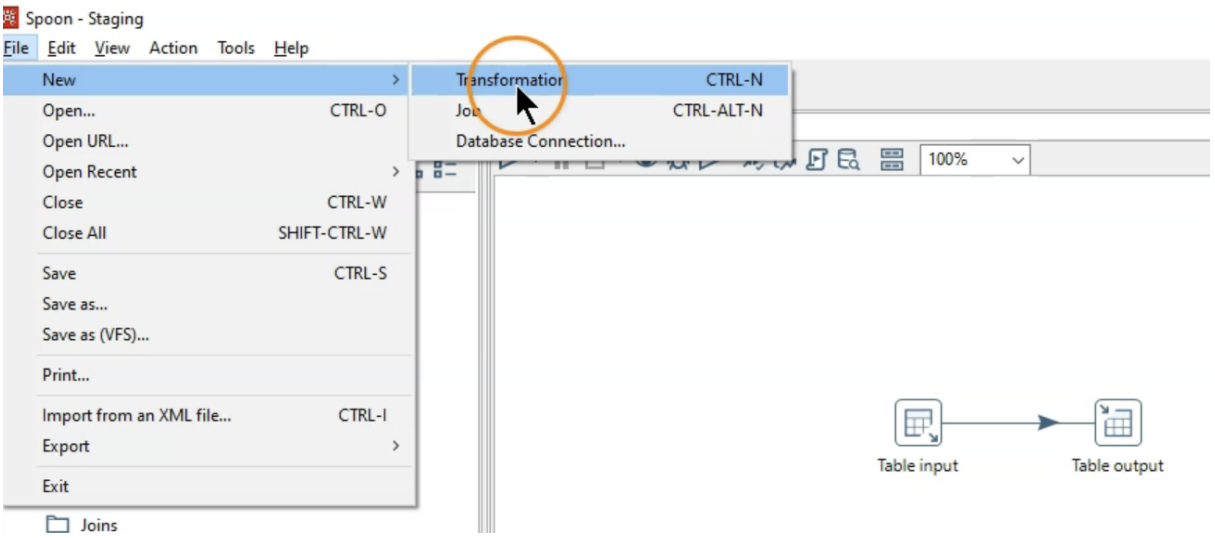
Query Editor | Query History

```
1  SELECT * FROM "Staging".dim_product
2  where product_id>''
3
4  TRUNCATE "Staging".dim_product
```

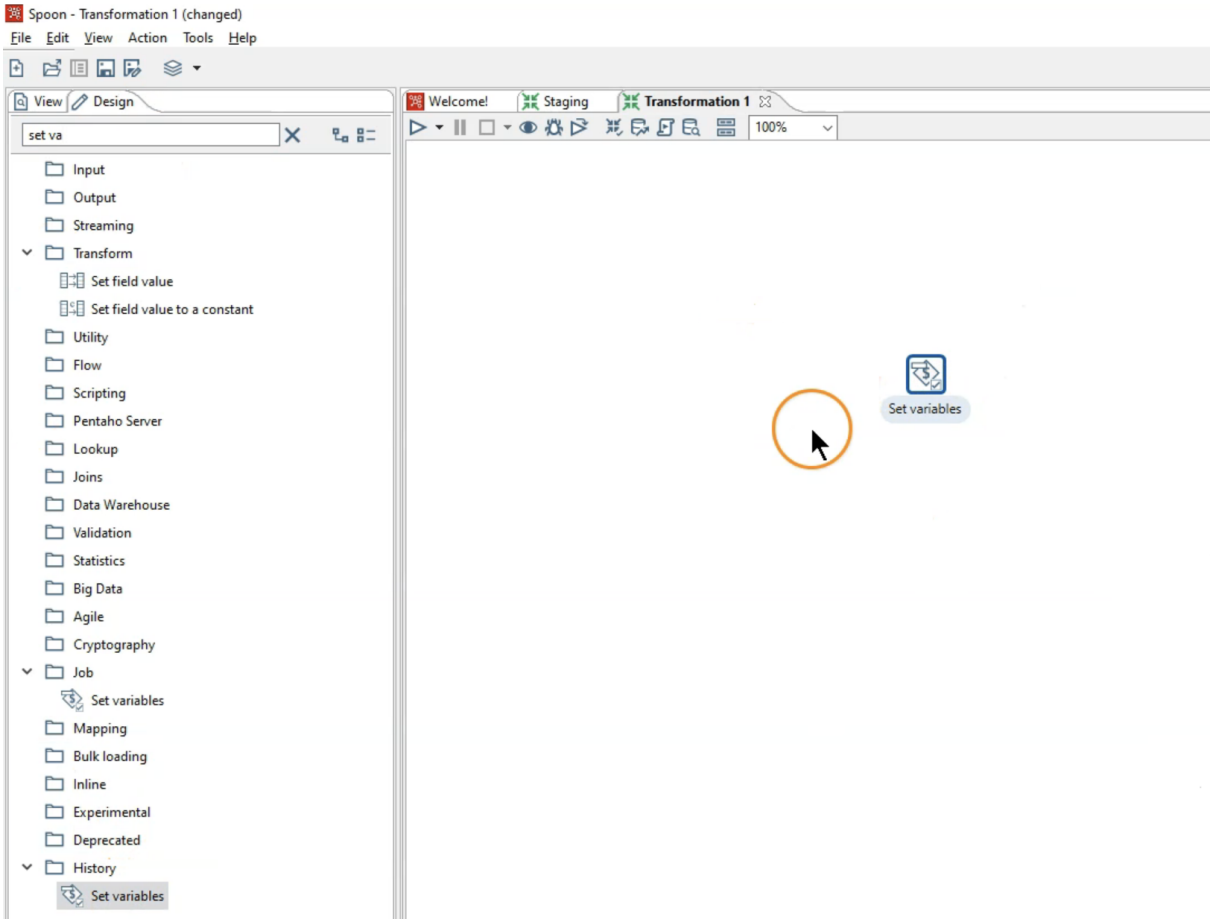
Data Output | Explain | Messages | Notifications

	Product_PK [PK] integer	product_id character varying	product_name character varying	category character varying	subcategory character varying
686	1385	P0733	Chocolate Wafer Bar - 4 Fingers Pack (Nestle)	Bakery, Cakes & Dairy	Dairy
687	1386	P0735	KitKat Chocolate Wafer Bar - 4 Fingers Pack (Nestle)	Snacks & Branded Foods	Biscuits & Cookies
688	1387	P0736	Organic Seeds - Black Mustard/Sasive (Organic Tattva)	Snacks & Branded Foods	Chocolates & Candies
689	1388	P0737	Organic Seeds - Brown Mustard/Sasive (Organic Tattva)	Foodgrains, Oil & Masala	Organic Staples
690	1389	P0738	Fevicol MR Angular Tip (Pidilite)	Foodgrains, Oil & Masala	Organic Staples
691	1390	P0739	Haldi Milk (mother dairy)	Cleaning & Household	Stationery
692	1391	P0740	Sterilised Double Toned Flavored Milk - Pista (THIRUMALA)	Bakery, Cakes & Dairy	Dairy
693	1392	P0741	Dark Fantasy - Choco Fills Biscuits - Cookies (Sunfeast)	Bakery, Cakes & Dairy	Dairy
694	1393	P0742	Coke Zero Soft Drink - No Sugar (Coca-Cola)	Snacks & Branded Foods	Biscuits & Cookies
695	1394	P0744	Bleaching Powder - Strong (Zermisol)	Beverages	Energy & Soft Drinks
696	1395	P0745	Cornflour (Weikfield)	Cleaning & Household	All Purpose Cleaners
697	1396	P0746	Haldi Chandan Soap (Jiva Ayurveda)	Foodgrains, Oil & Masala	Atta, Flours & Sooji
698	1397	P0747	Pure Magic Deuce Vanilla Biscuits (Britannia)	Beauty & Hygiene	Bath & Hand Wash
699	1398	P0748	[...] ()	Snacks & Branded Foods	Biscuits & Cookies

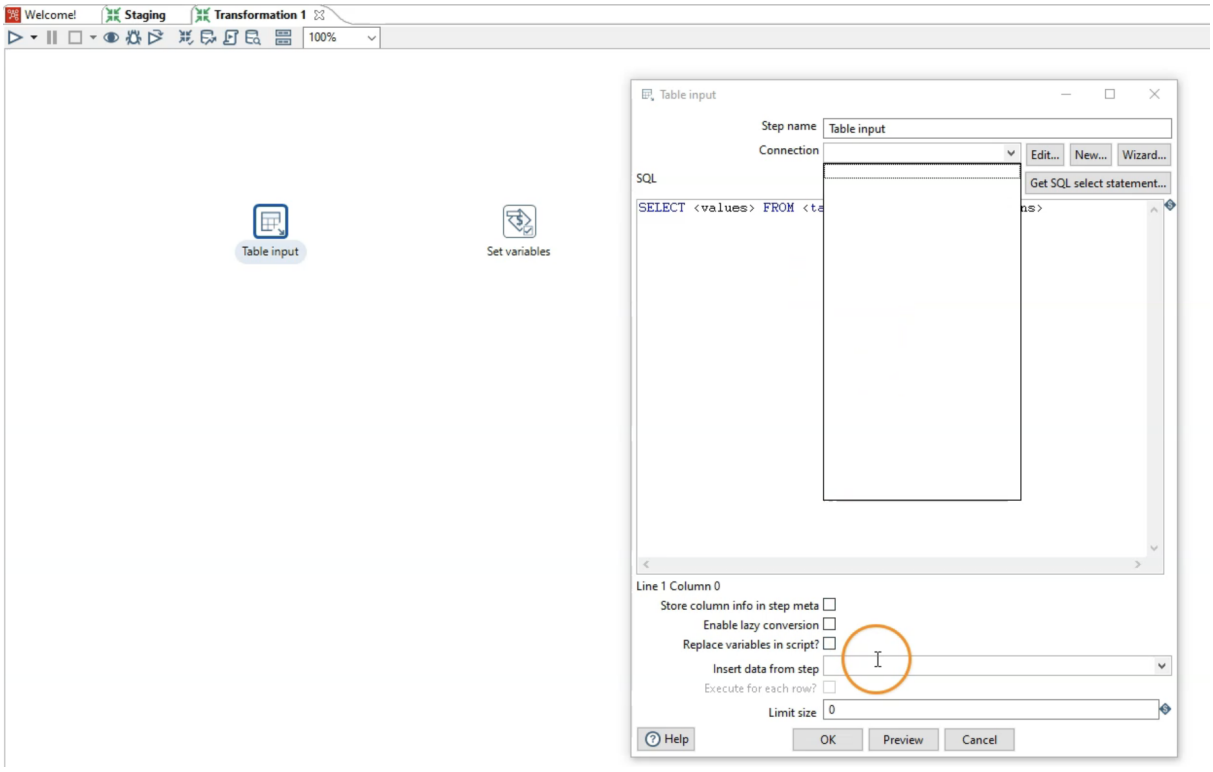
- That means we will have to store the maximum product ID in the product table from the data that we have already loaded
- So we will have to store the P0748 which is the last product_id value, so we can use it to load the new products that are added after this
- To save that maximum value of product_id we will need an additional transformation step



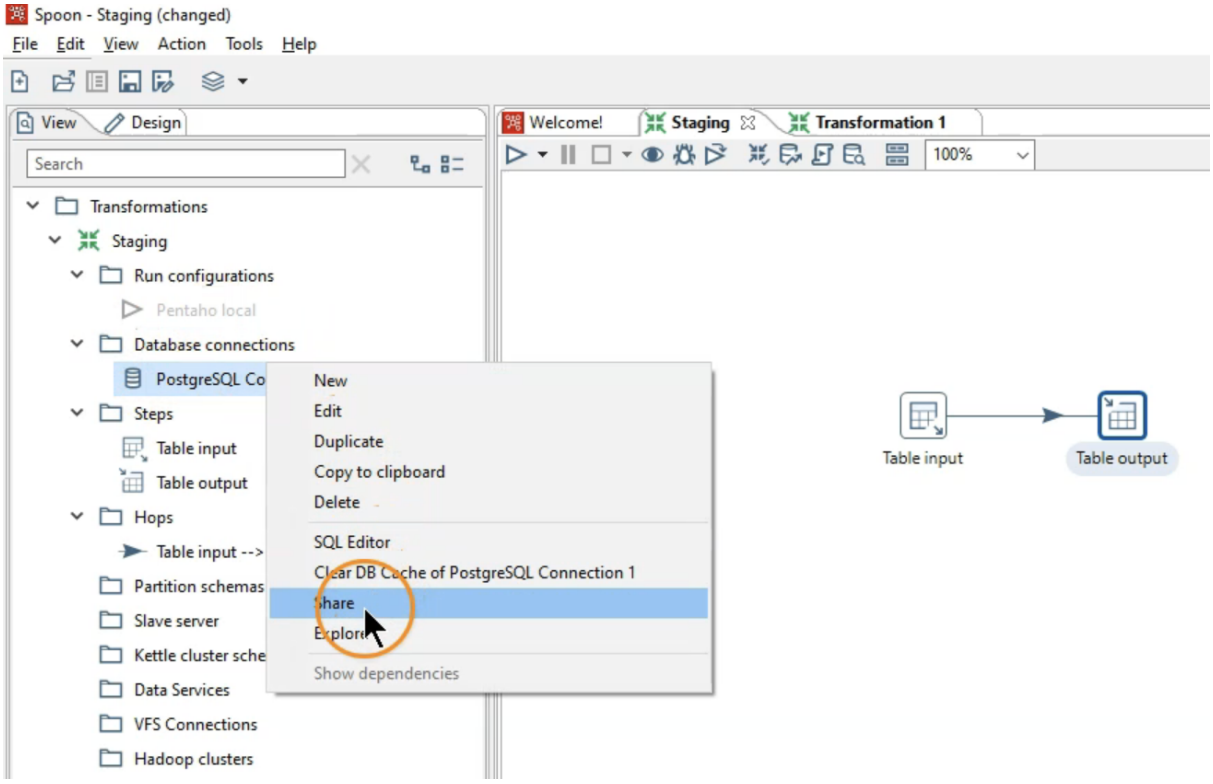
- For this we have to use set variable

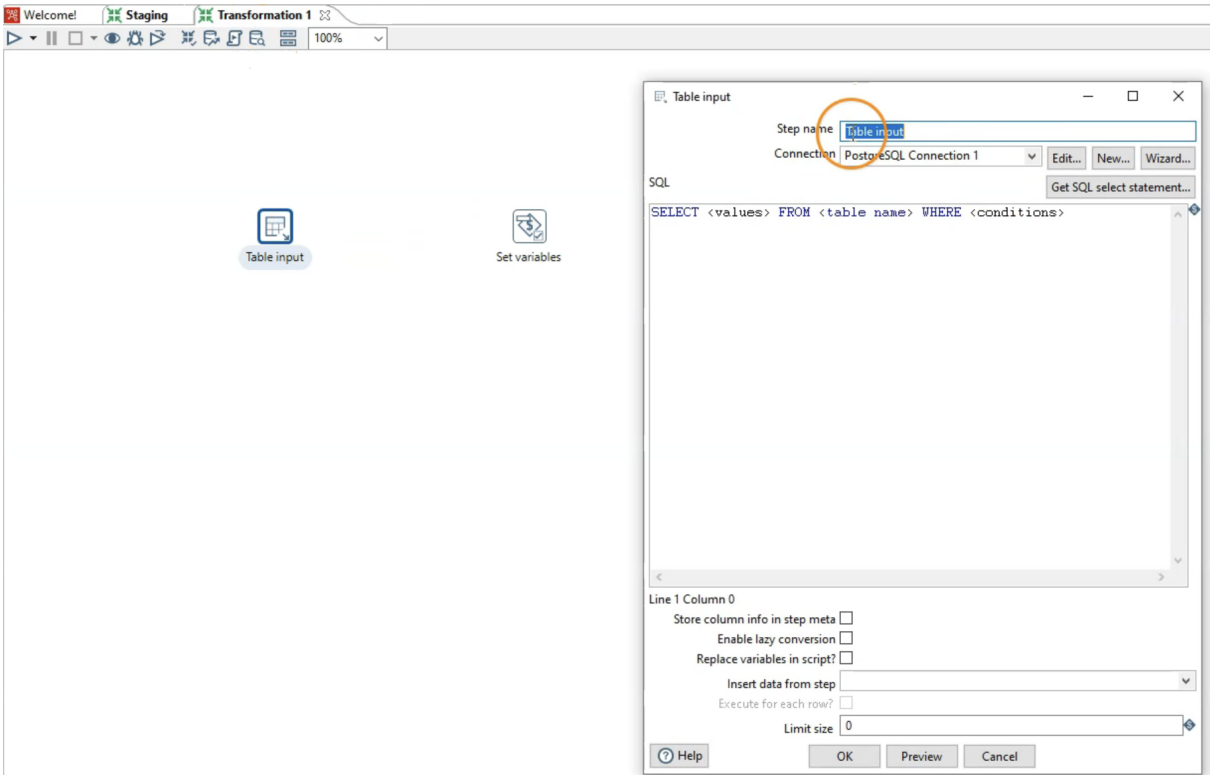


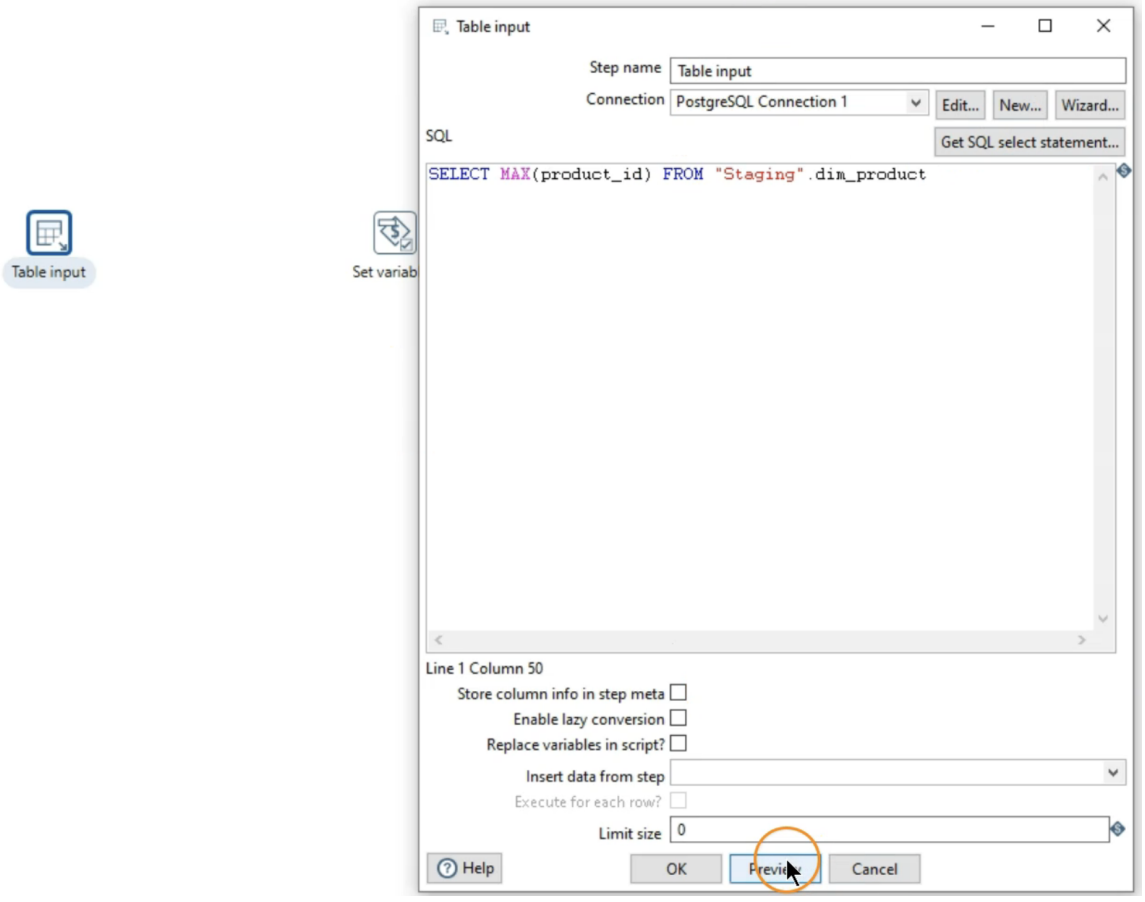
- We also need the Table input from where we will be using the maximum product_id



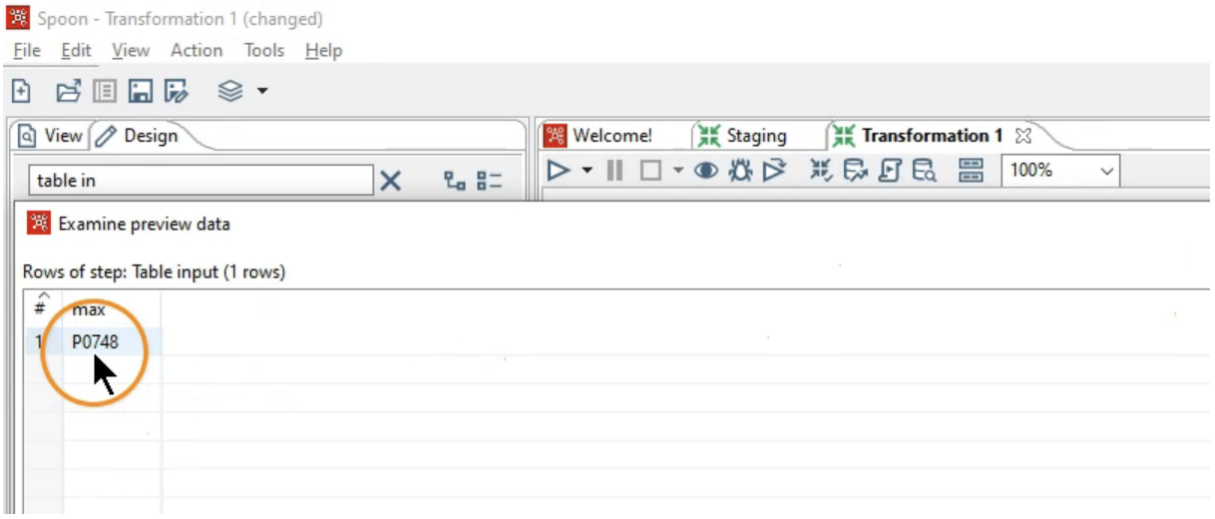
- But since we have created a new transformation Postgresql connection is no more available, for it to be available from previous transformation, we will have to share it



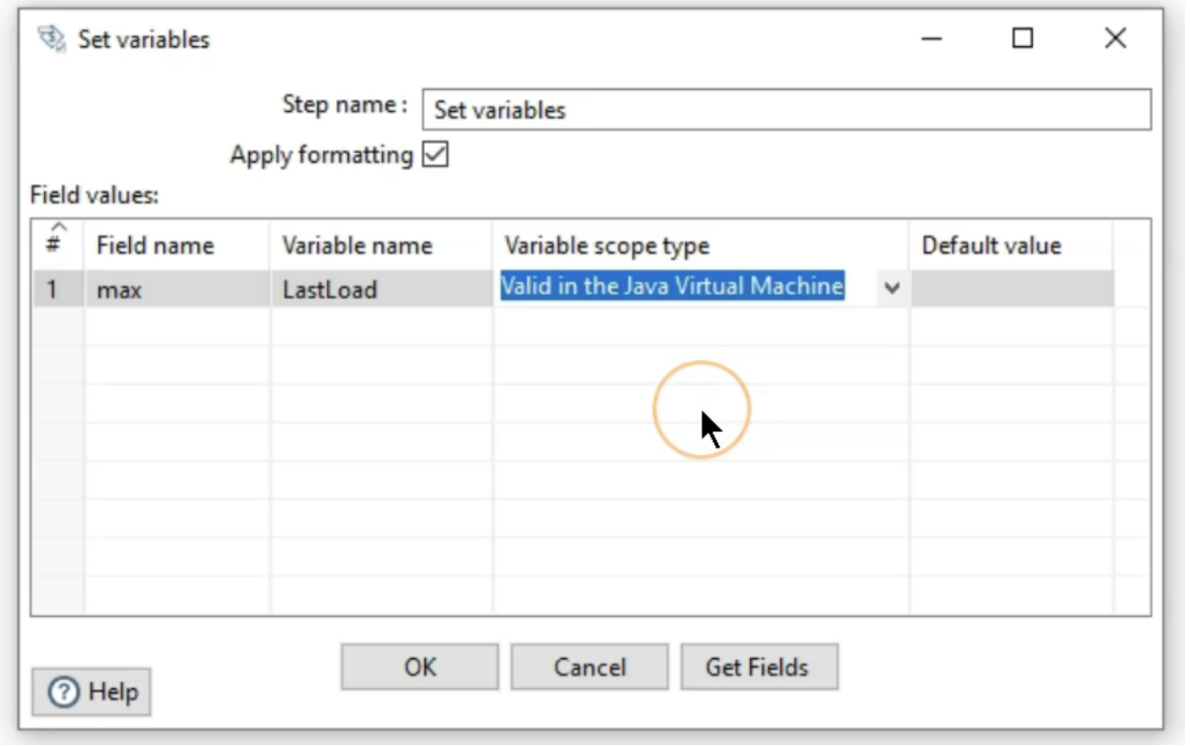




- We can preview if we are getting right value

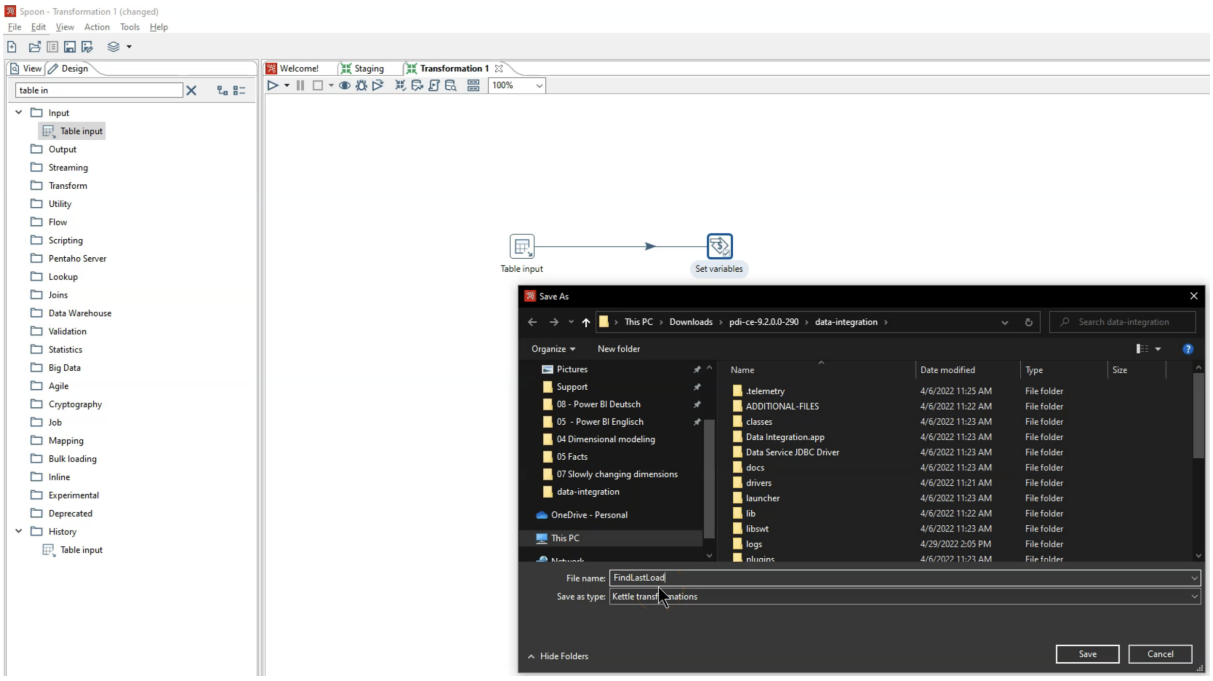


- Use Shift key and connect the two
- Double click the set variable and set the configurations

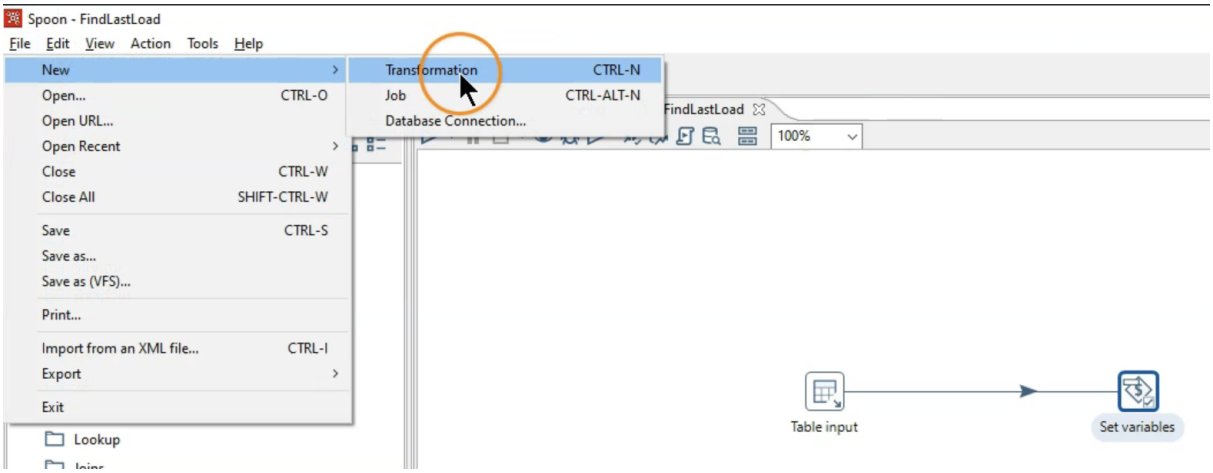


- Chose the Valid in the Java Virtual Machine so it will be available in other transformations

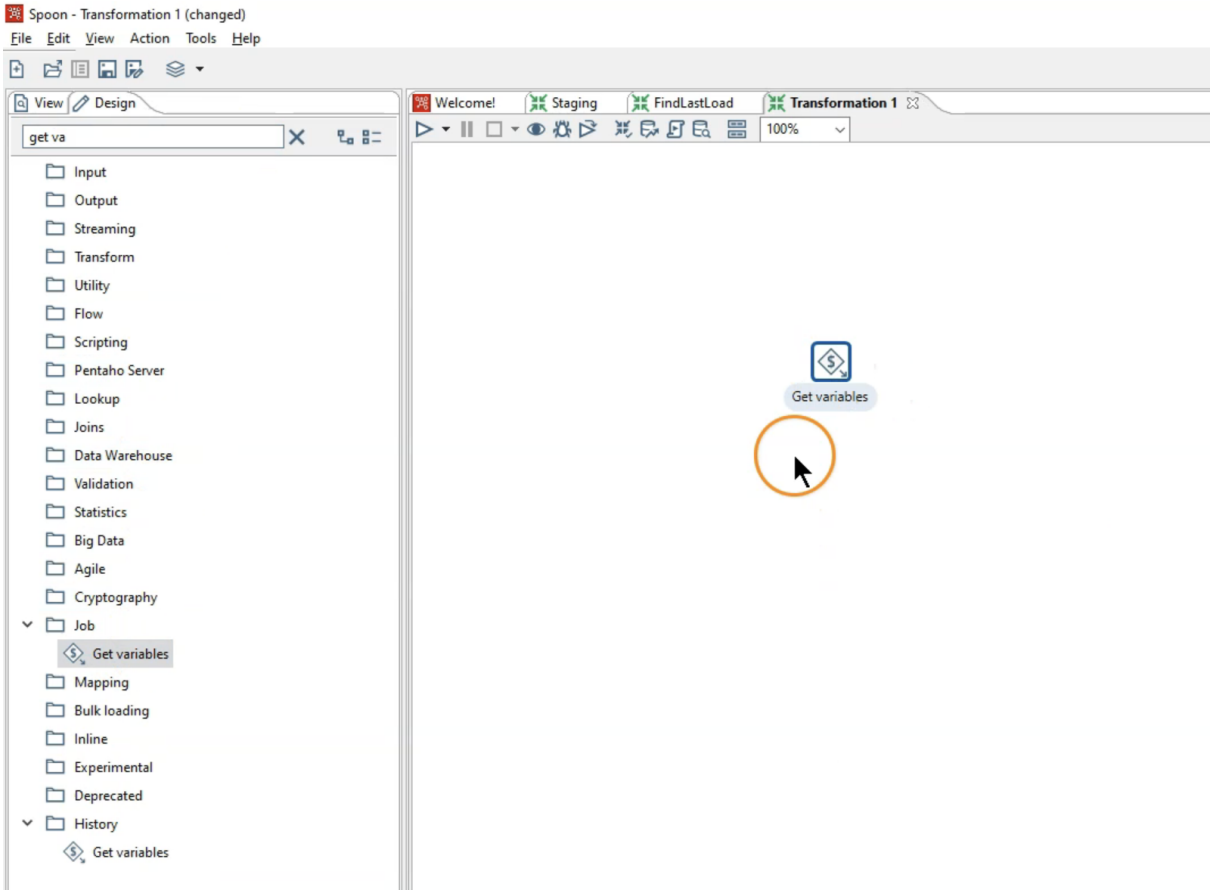
- Save this new transformation that is created. Go to File> Save As



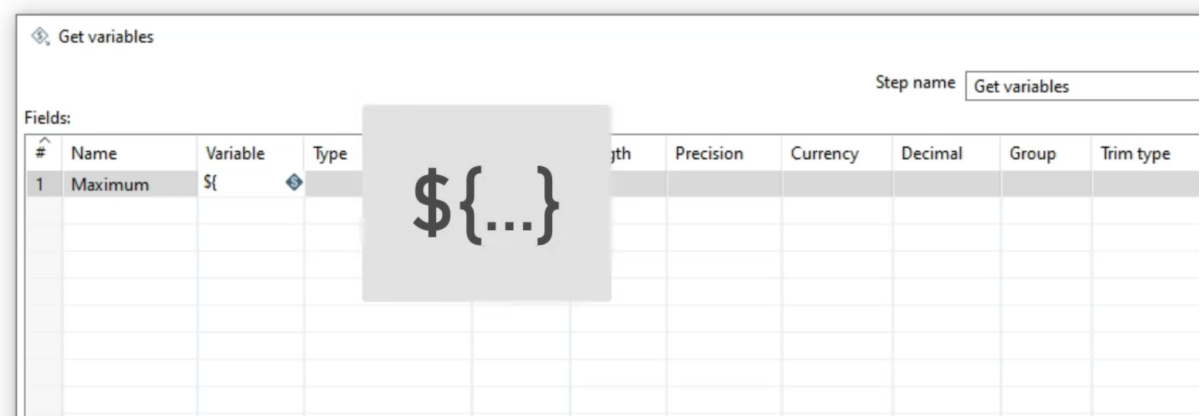
- Again create a new transformation to get that maximum value which we extracted in previous transformation

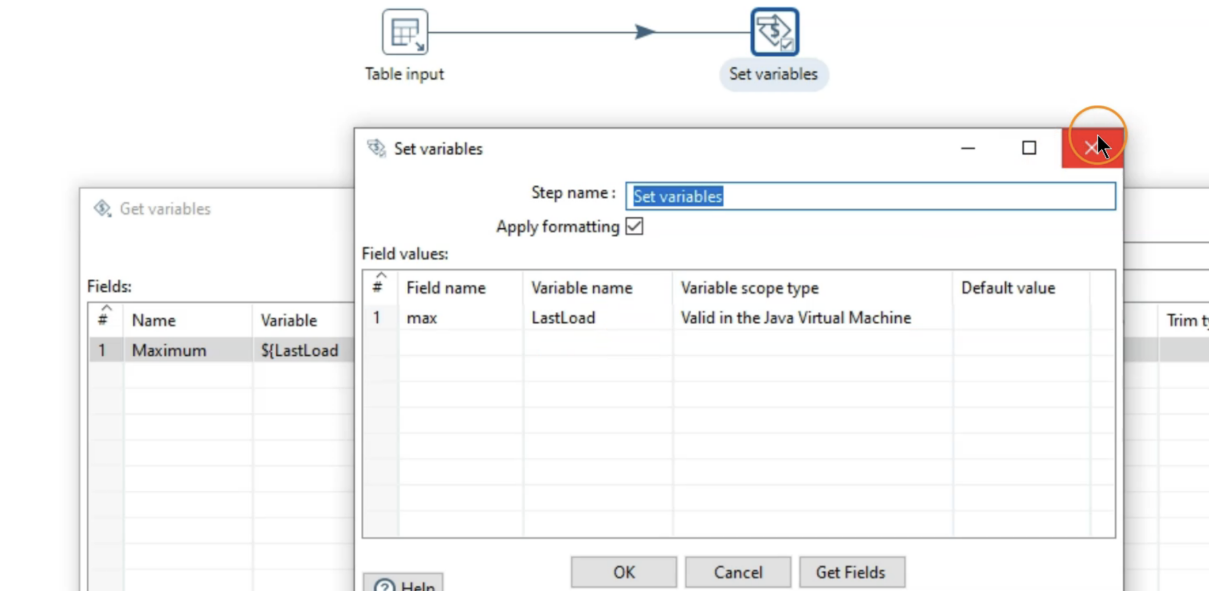


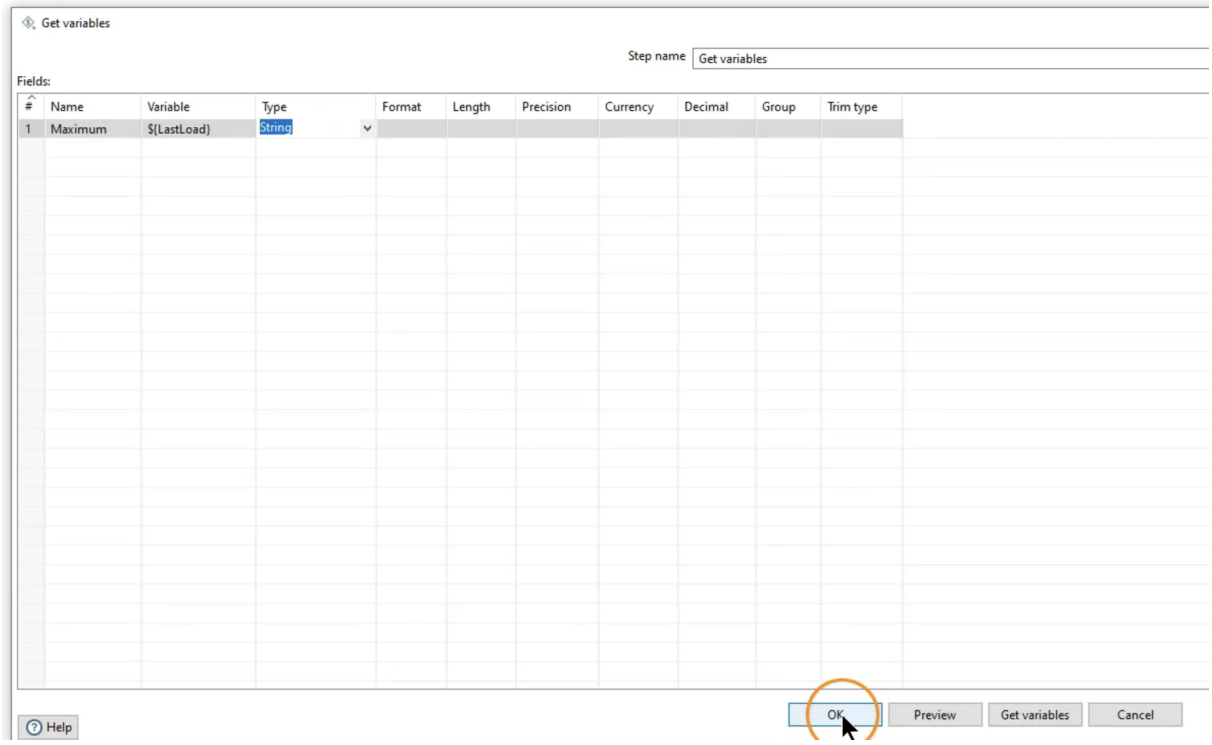
- Use Get variable



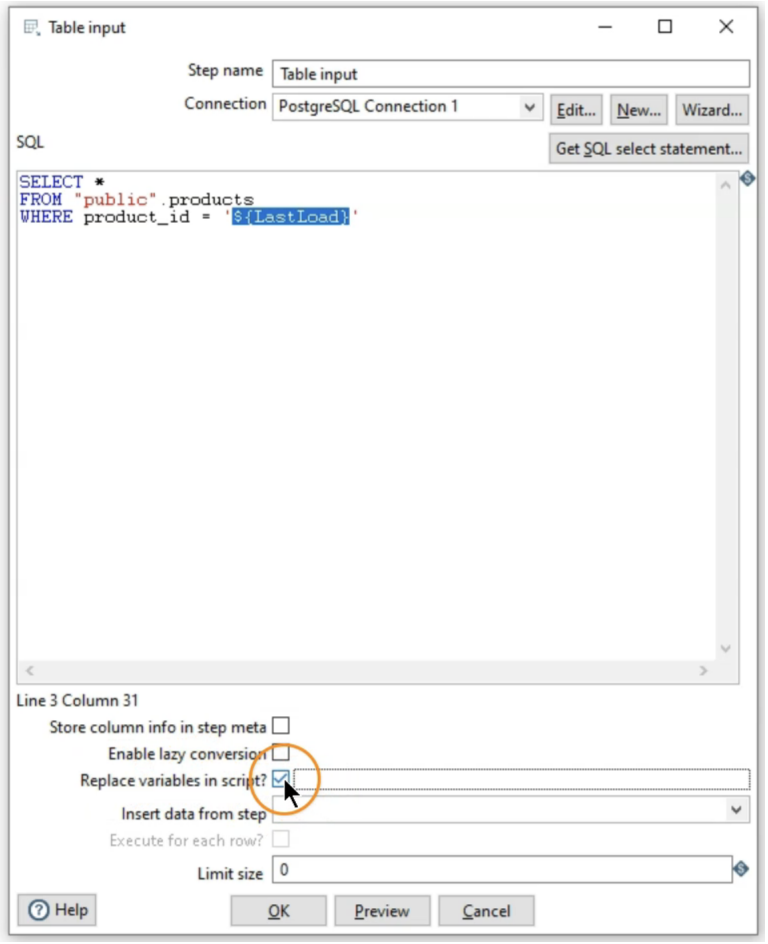
- To get the variable value, we need to use the ${...} format to get the value under Variable column



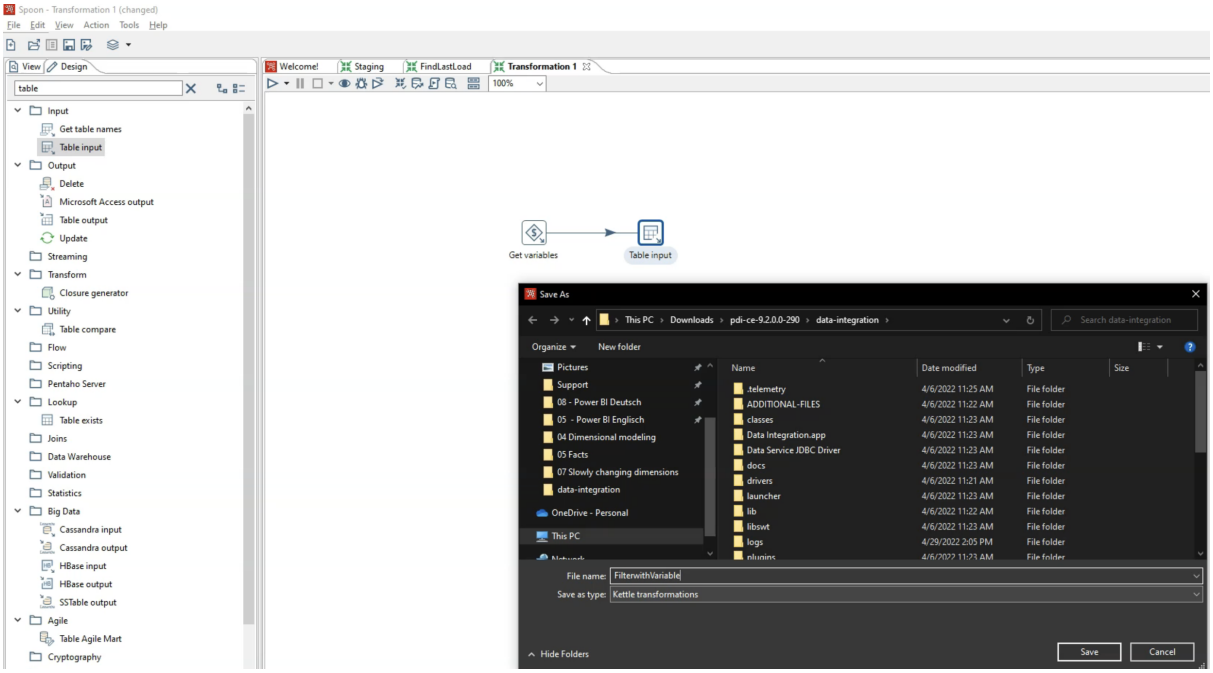




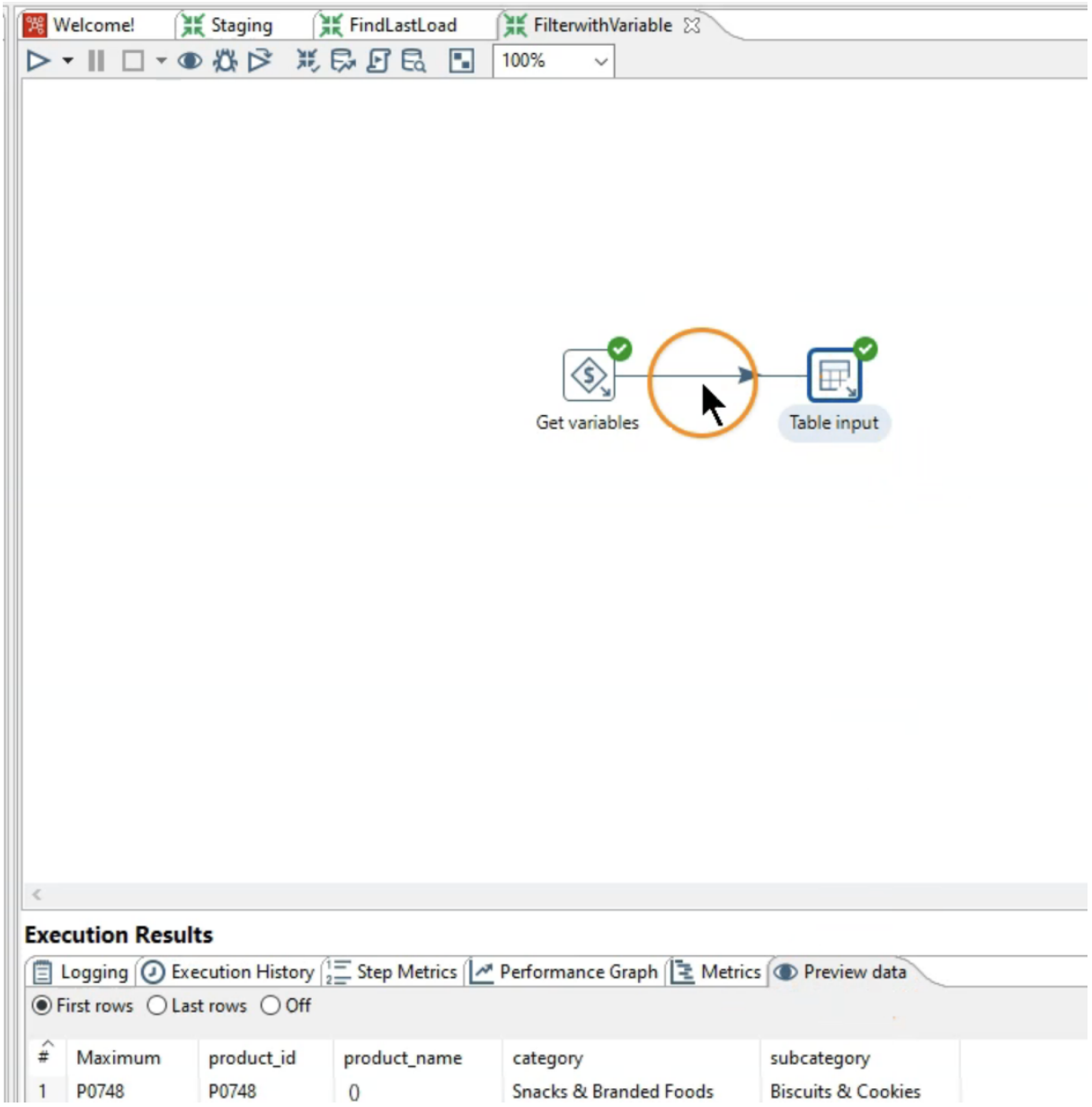
- We need to use the query and the variable where it is retrived and also have to check the Replace variable in script option

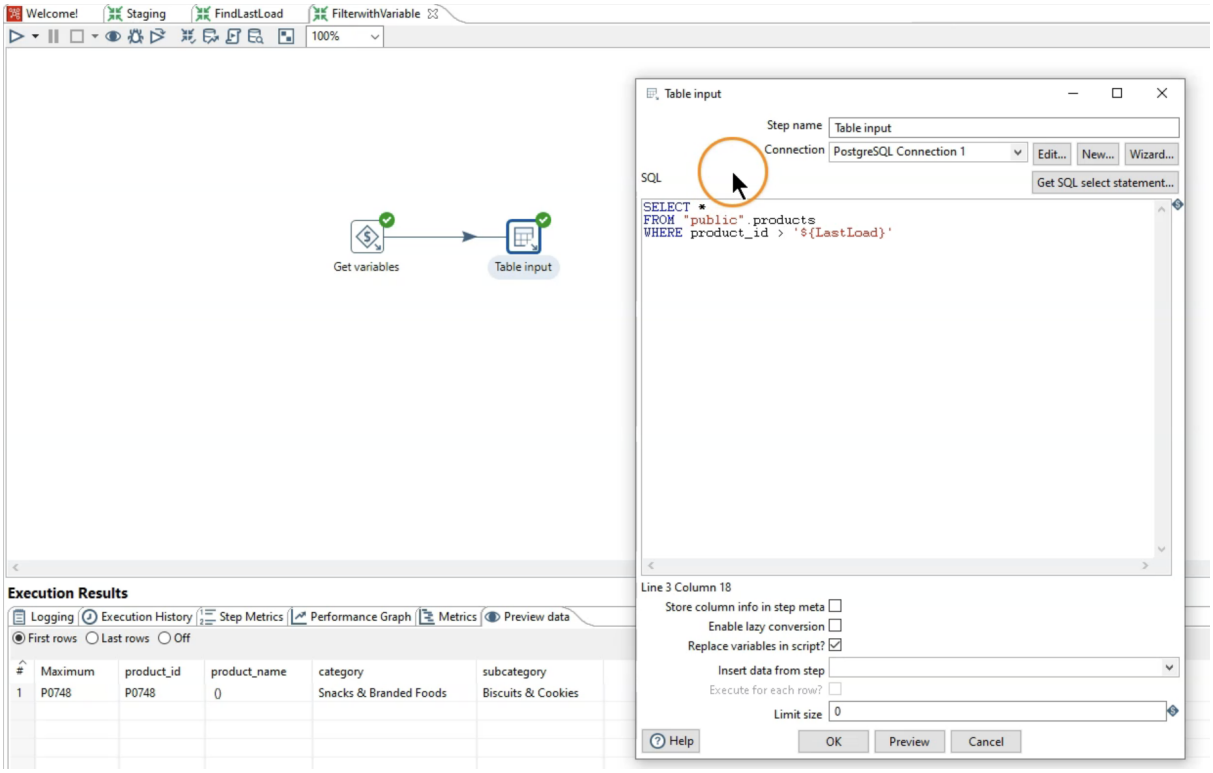


- Save this new transformation

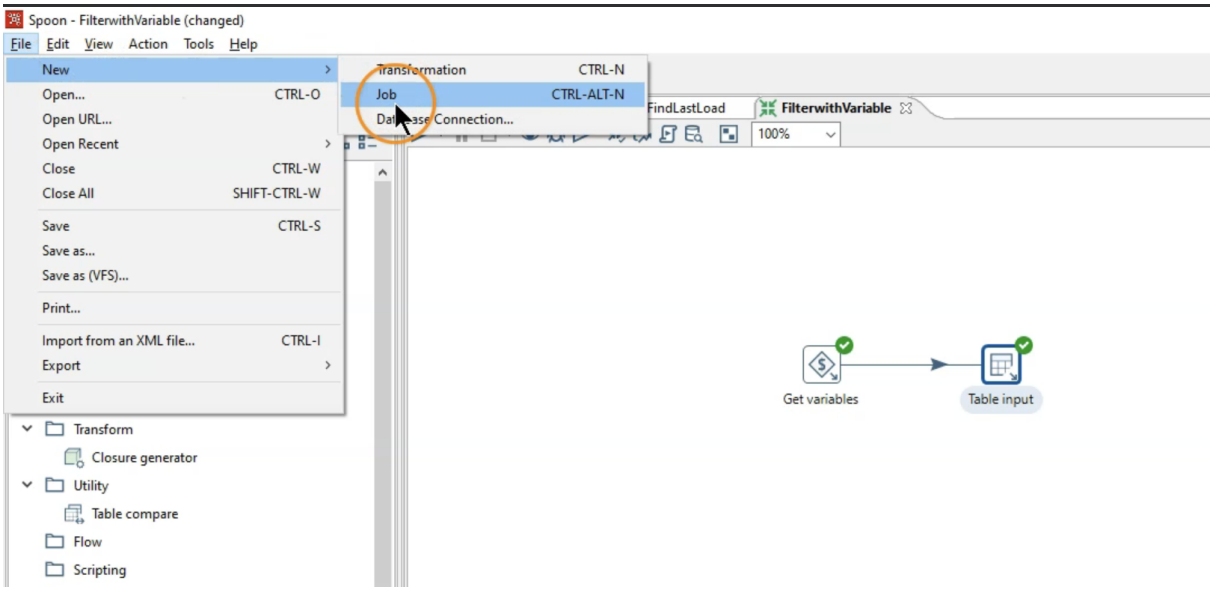


- If we run the FindLastLoad transformation and then run the FilterwithVariable we can see the data which we had set that is to fetch one value, now we can use this to fetch all records greater than the variable

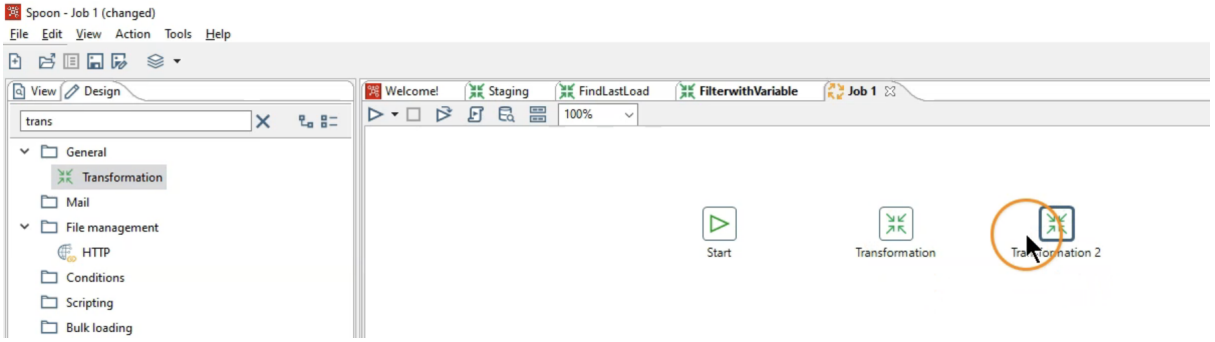




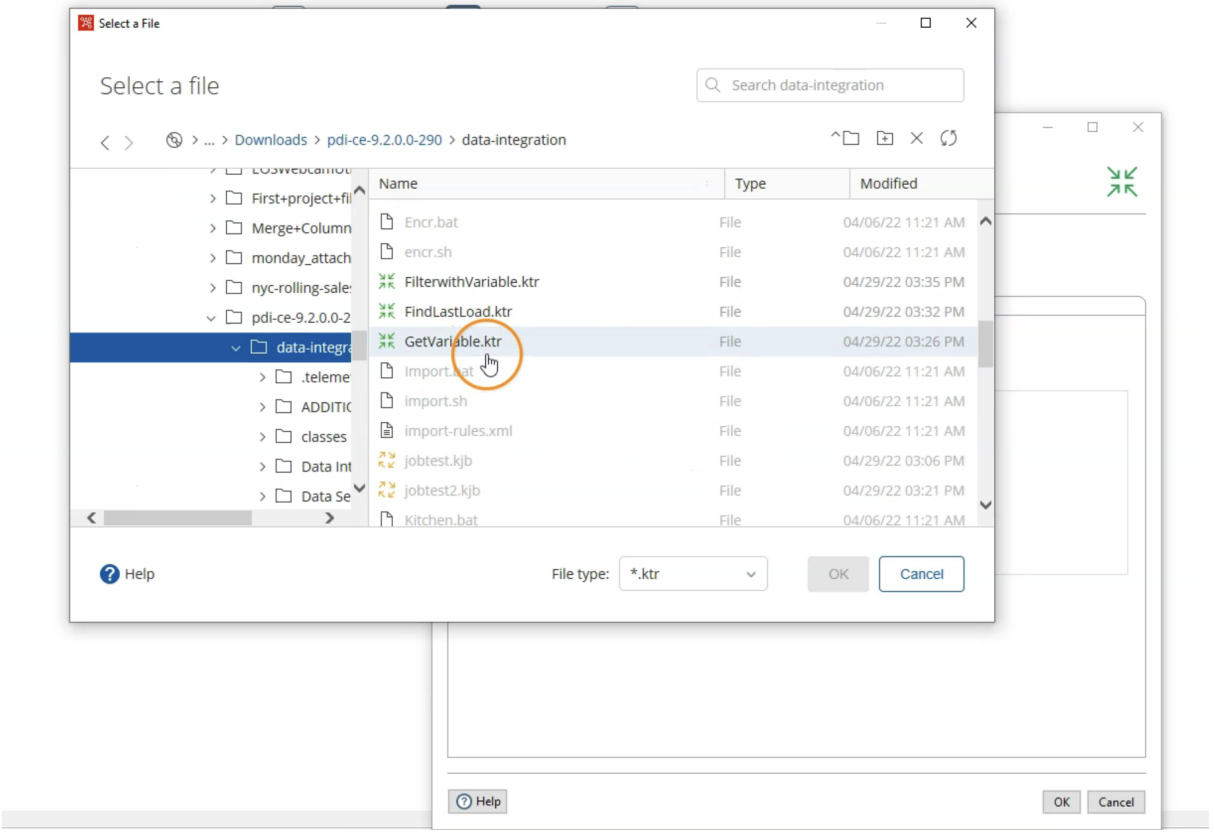
- Now we will set this all transformation in a Job

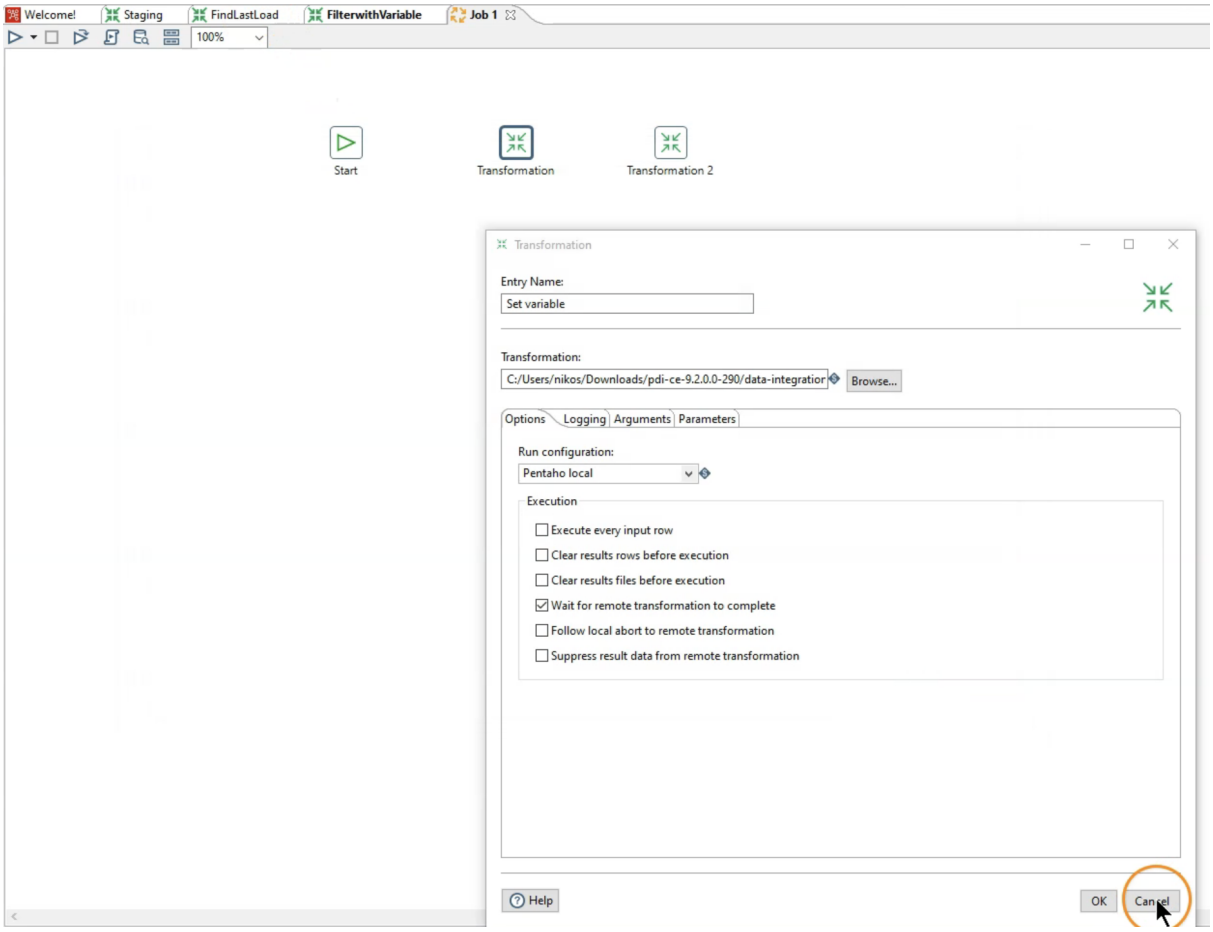


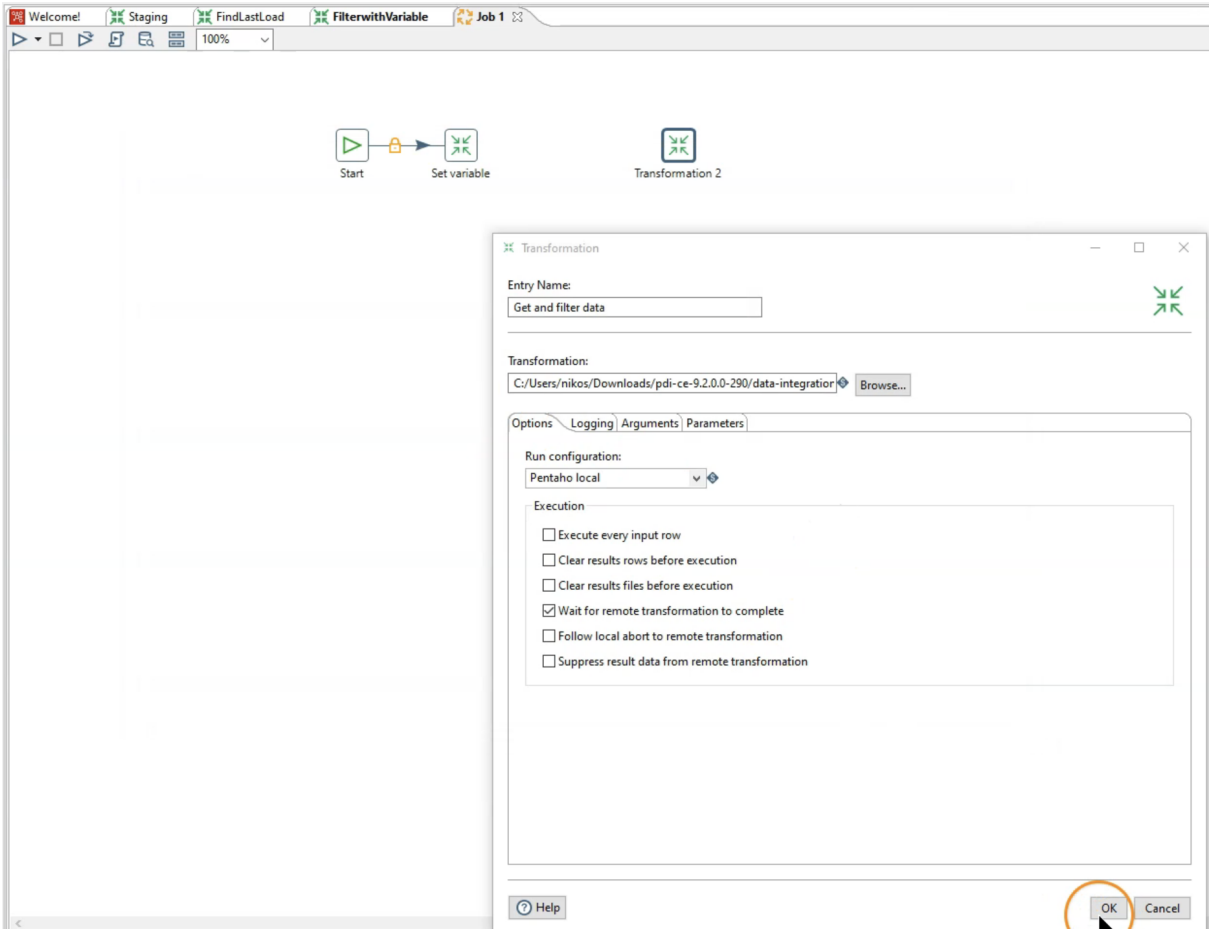
- First we always need a Start, then we can add the transformation



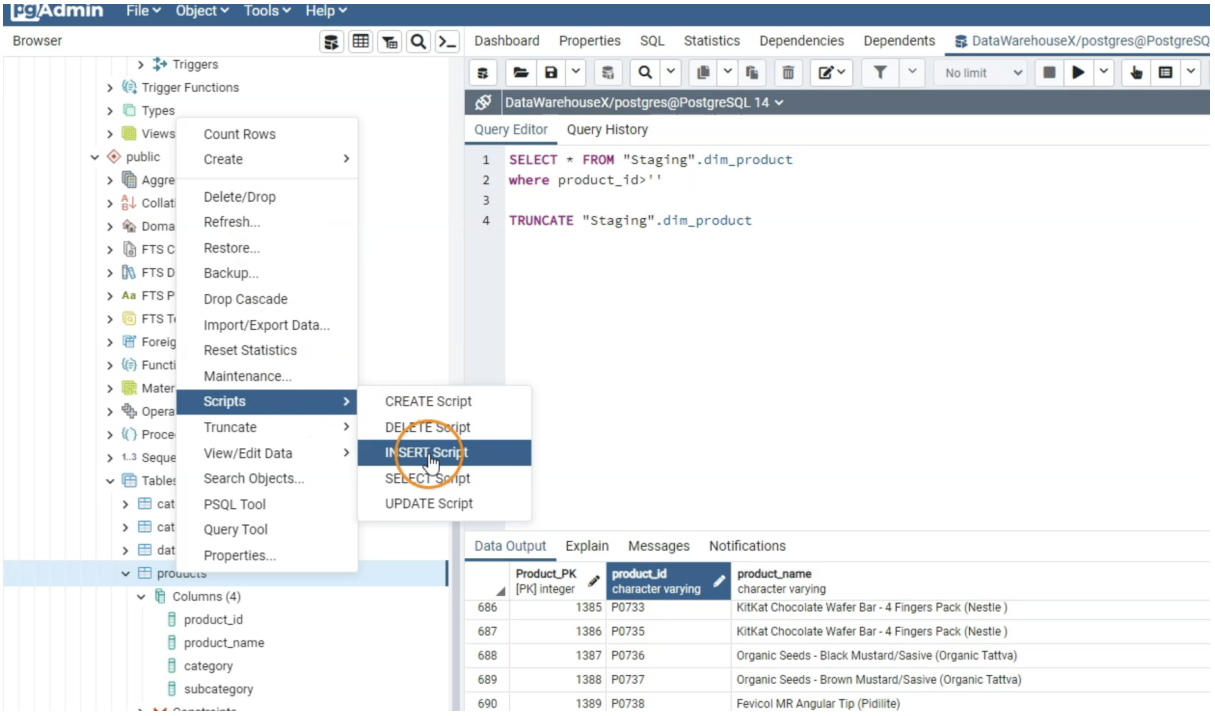
- Click on the Transformation to select the transformation we have saved previously

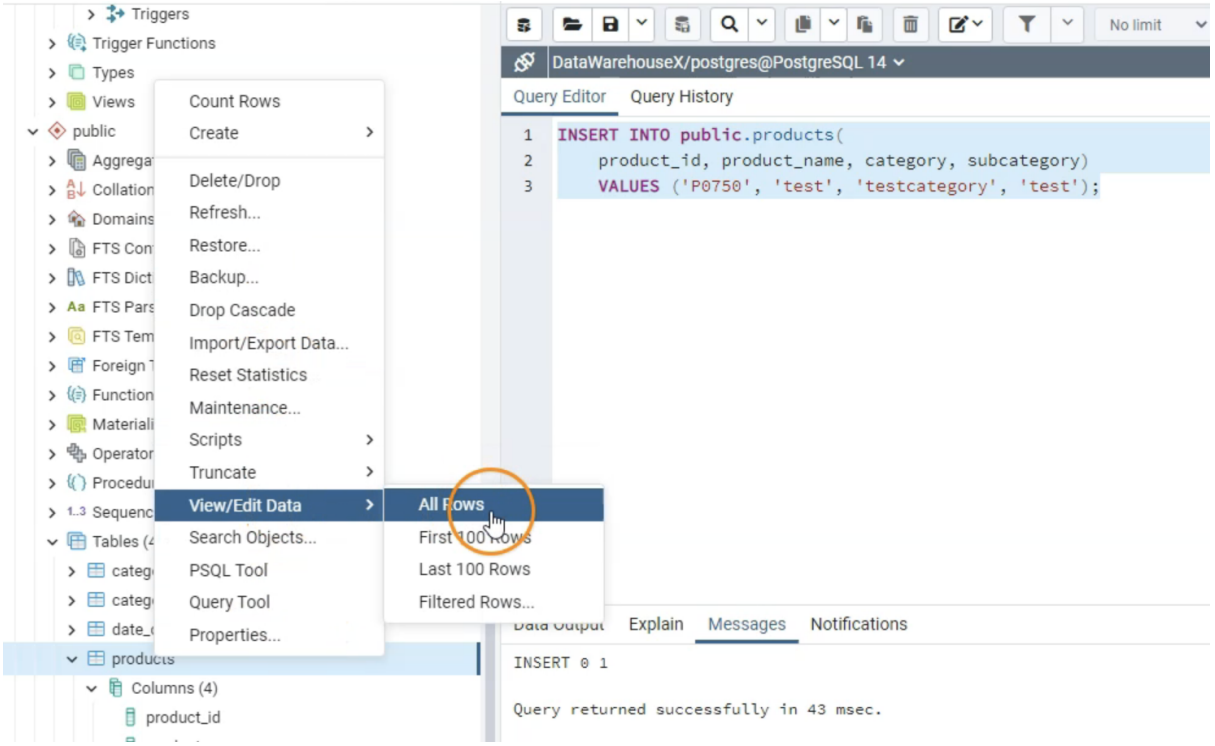


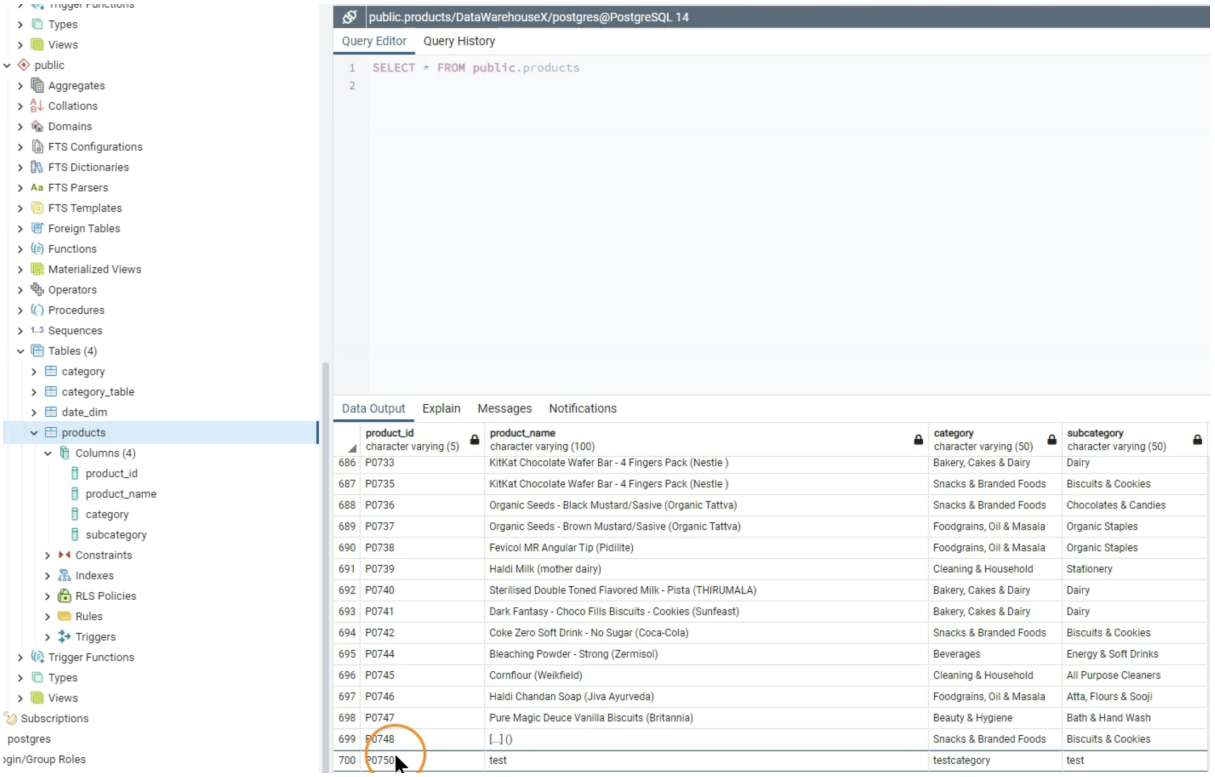

Welcome!
Staging
FindLastLoad
FilterwithVariable
Job 1
100%
Start
Transformation
Transformation 2
Transformation
Entry Name:
Set variable
Transformation:
C:/Users/nikos/Downloads/pdi-ce-9.2.0.0-290/data-integratior
Browse...
Options
Logging
Arguments
Parameters
Run configuration:
Pentaho local
Execution
Execute every input row
Clear results rows before execution
Clear results files before execution
Wait for remote transformation to complete
Follow local abort to remote transformation
Suppress result data from remote transformation
Help
OK
Cancel



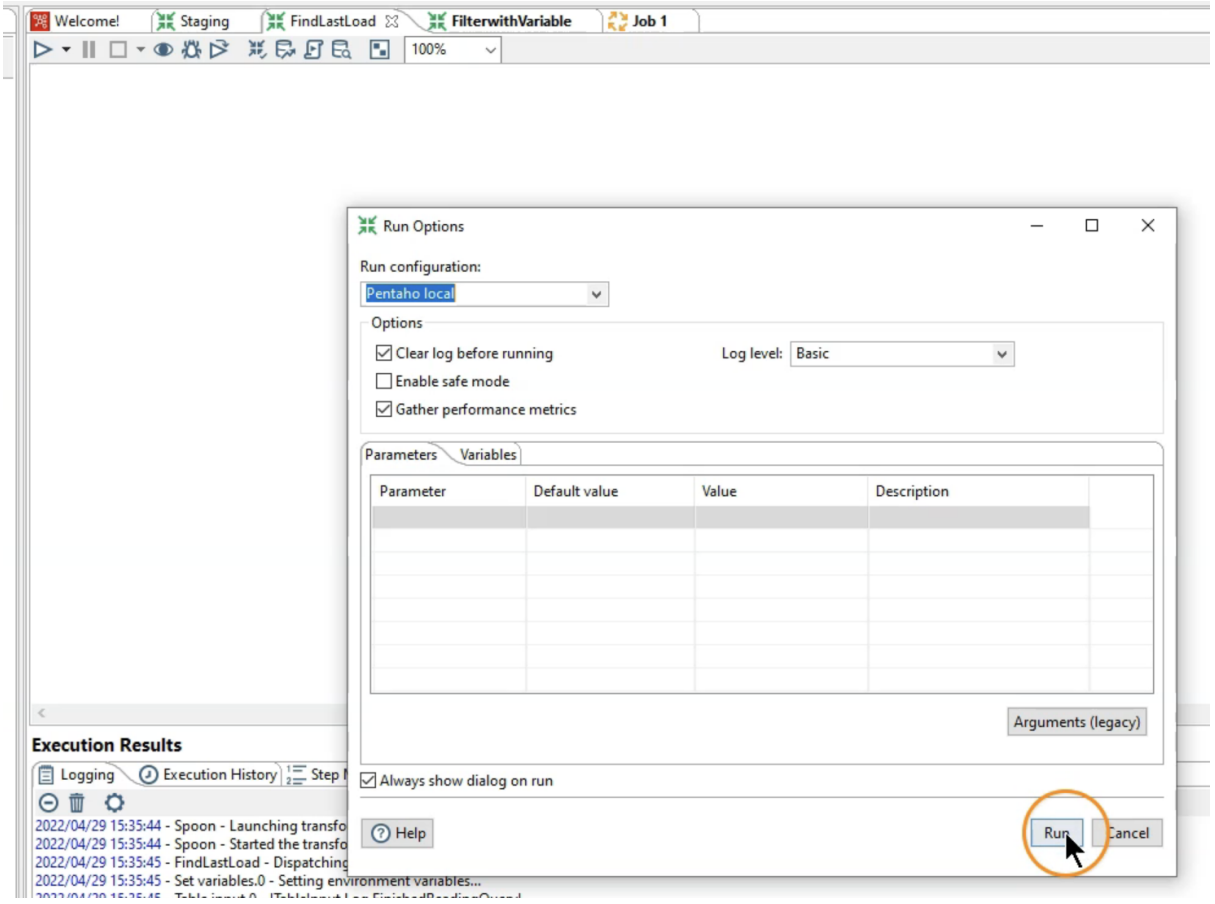
- Let us insert a new record to test in the source table

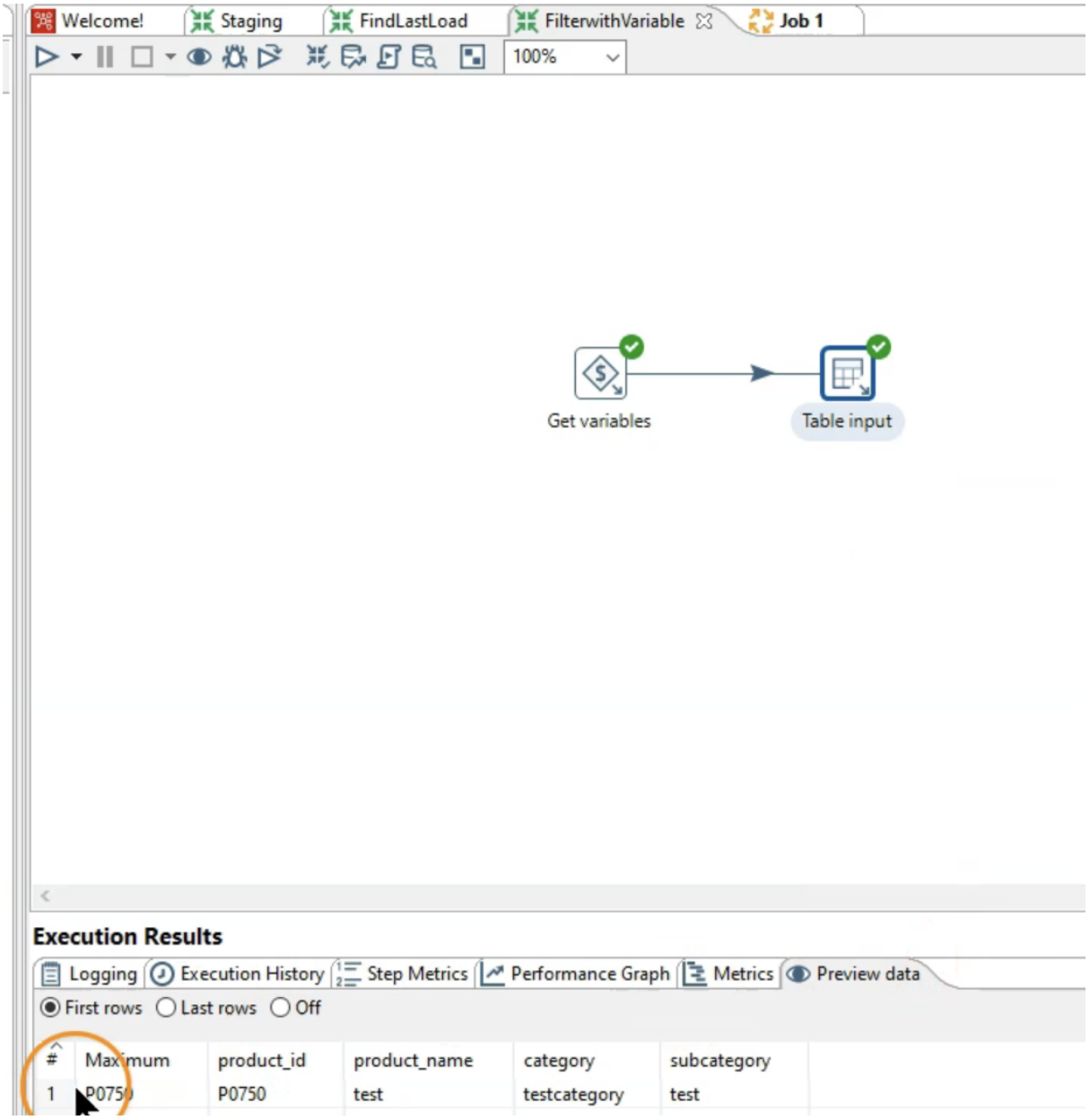






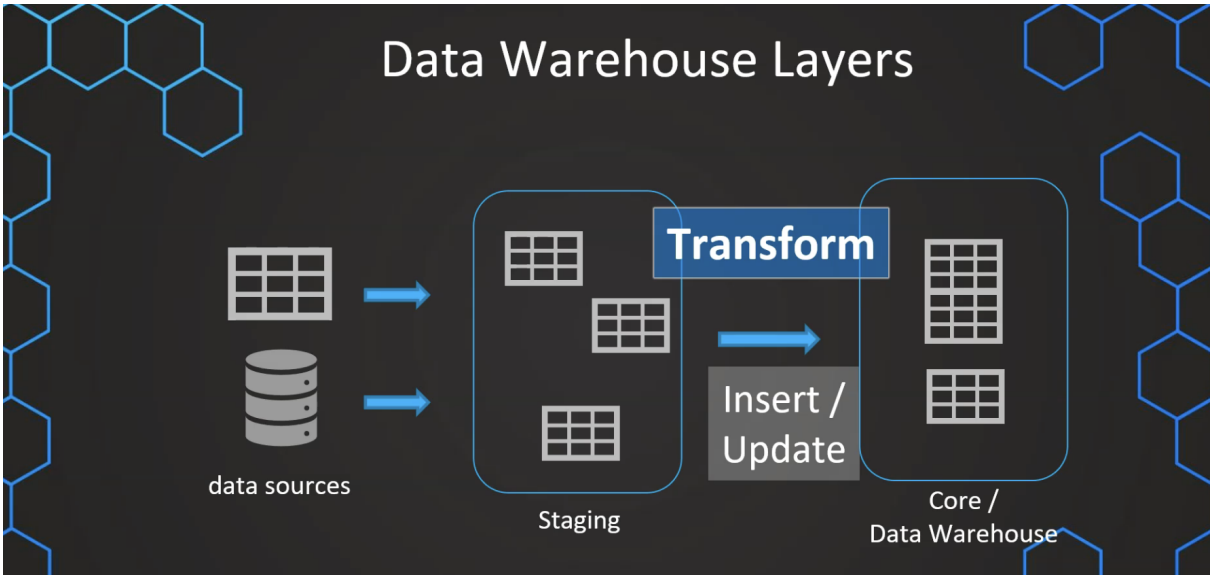
- Now let us run the transformations an preview the data

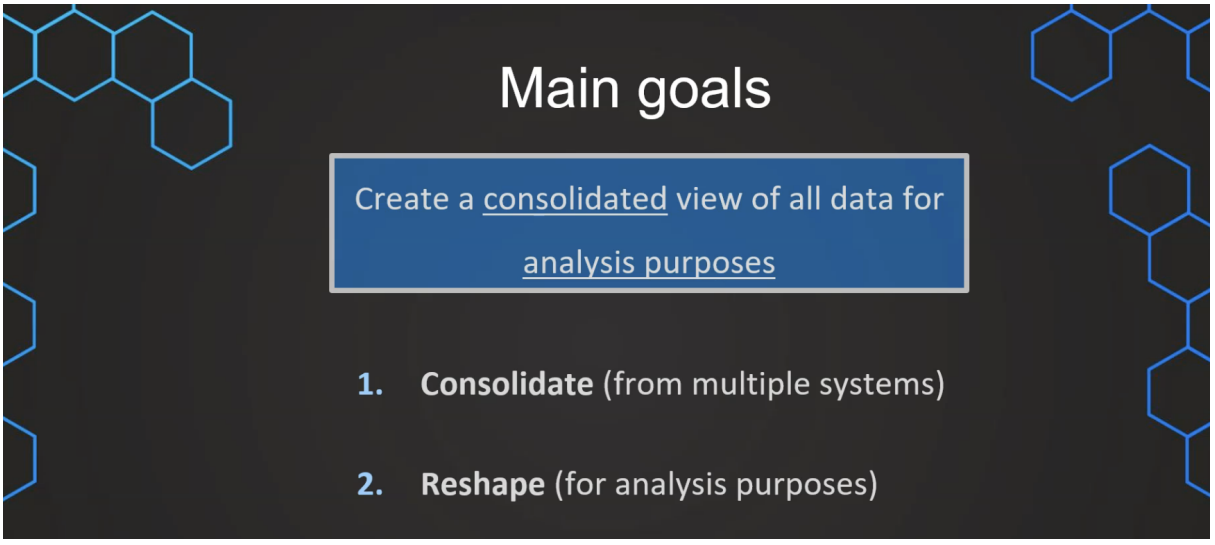

Welcome!
Staging
FindLastLoad
FilterwithVariable
Job 1
100%
Run Options
Run configuration:
Pentaho local
Options
Clear log before running
Log level:
Basic
Enable safe mode
Gather performance metrics
Parameters
Variables
Parameter
Default value
Value
Description
Arguments (legacy)
Always show dialog on run
Help
Run
Cancel
Execution Results
Logging
Execution History
2022/04/29 15:35:44 - Spoon - Launching transfo
2022/04/29 15:35:44 - Spoon - Started the transfo
2022/04/29 15:35:45 - FindLastLoad - Dispatching
2022/04/29 15:35:45 - Set variables.0 - Setting environment variables...



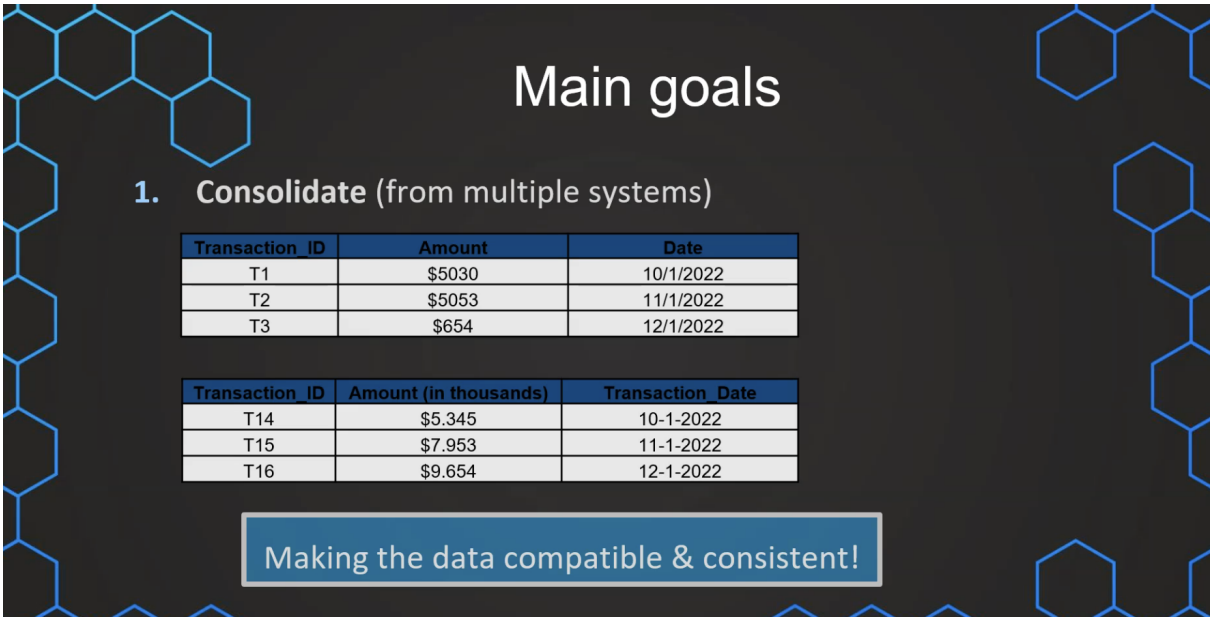
Transform

- We have to transform our data to get more out of our data and make our data more consistent

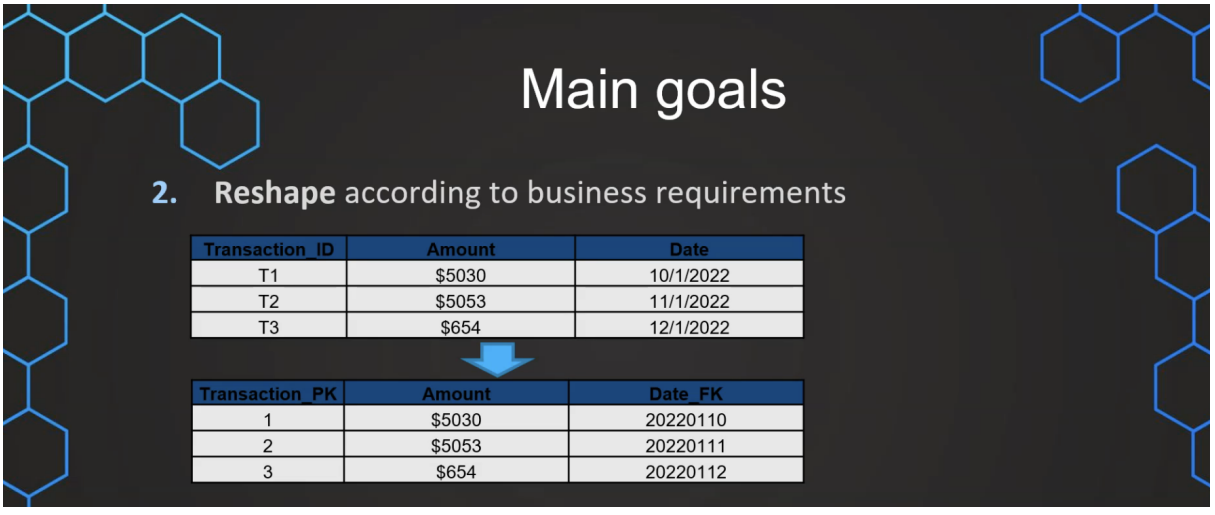




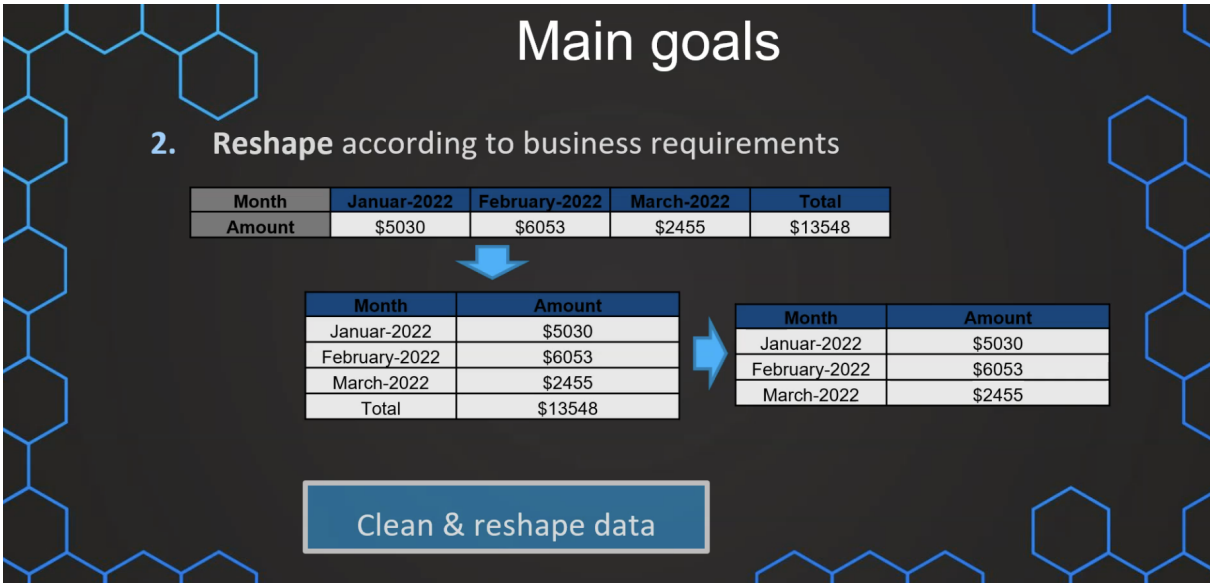
- Consolidated view is collecting data from multiple sources and this could include different things like data type conversions and column names, and different types of standards that we want to set to create a consolidated view.
- But then also we want to do some additional reshaping that is just there to fit the data to our analytical or reporting needs. So, we might want to add some additional information or just reshape the data so that it can be analyzed.



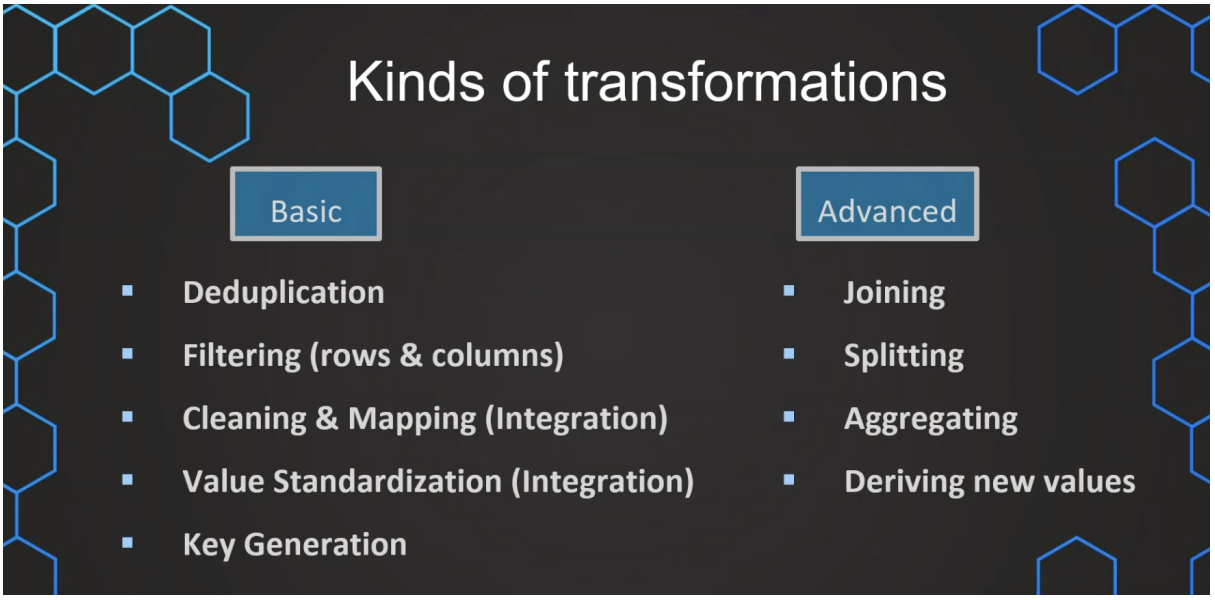
- we have different systems, and we can see that sometimes in those different systems where we are pulling data from, we can have different types of formats, different types of data types, column names, and all of that, we want to bring into one consolidated and standardized view.
- So for example, we see that in this column, we have the amount in thousands and also the data type then obviously it's also different, while we have a decimal number in the lower table, in the upper table, we have an integer number.
- And the same goes for different date formats. And this is all something that we want to consolidate to make the data compatible with each other so that we can load it into one single table. That's very important, of course, in our data warehouse.



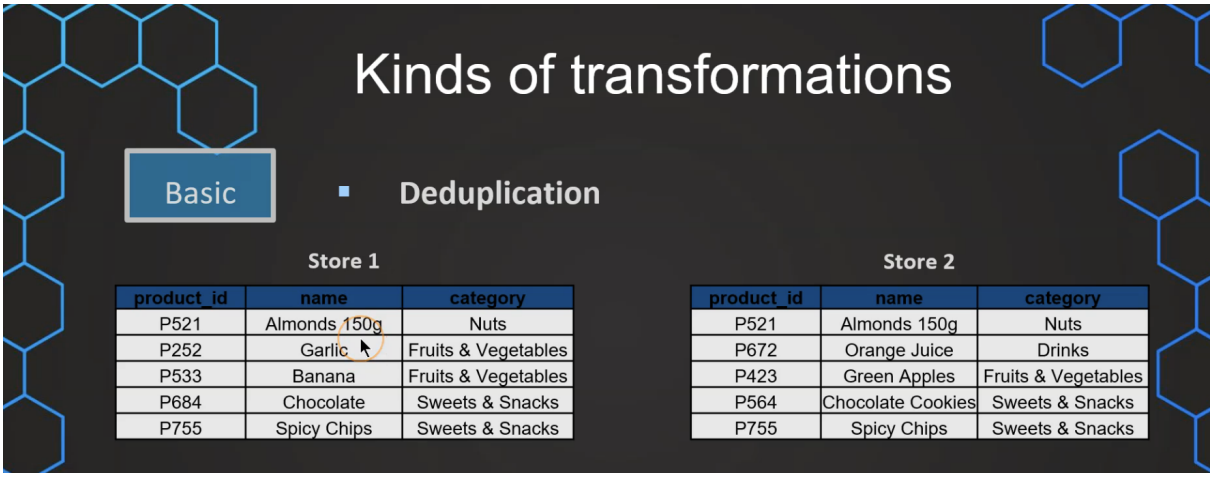
- So for example, if we have our table like this initially, of course we want to reshape the data so that we can use it in a dimensional way. This is what we want to do in our data warehouse.
- And this can include that we, for example, want to include foreign keys. So these are simple reshapings, restructurings of our data.

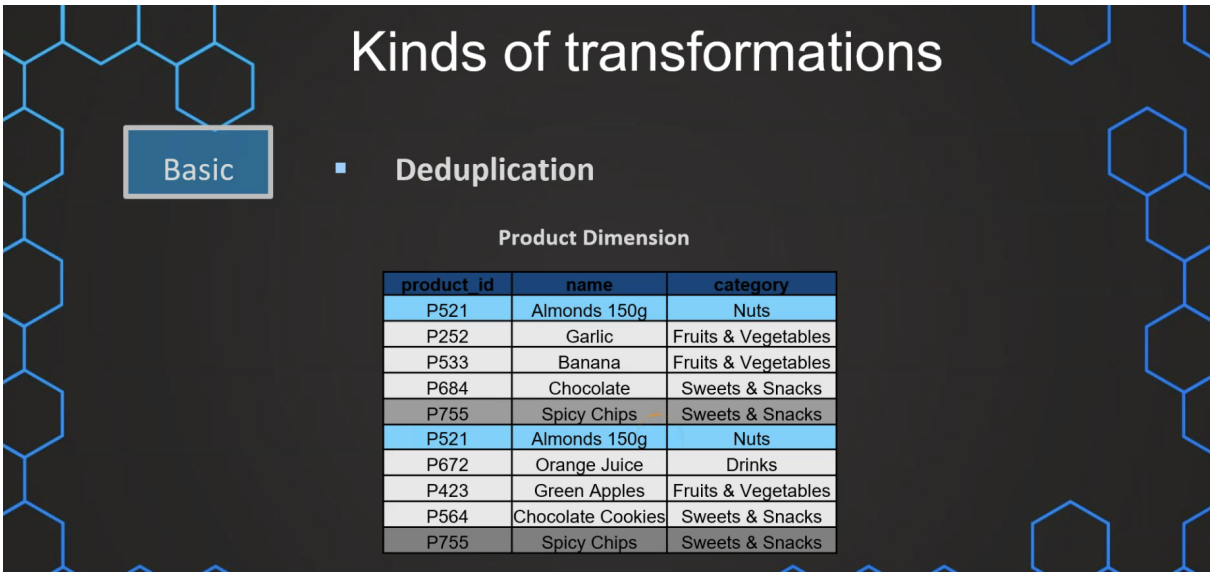


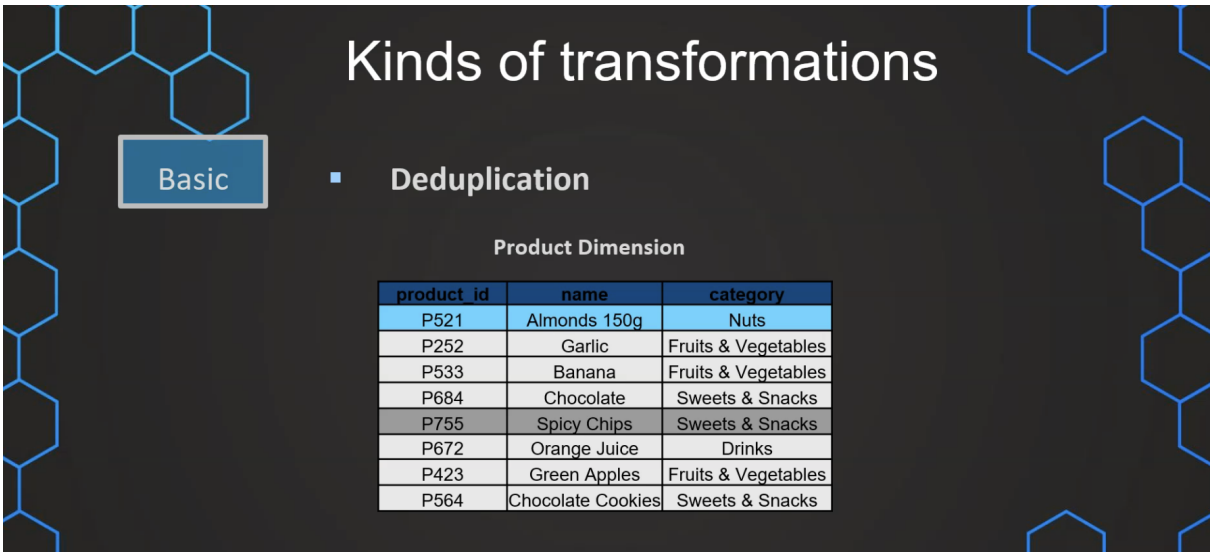
- Another example could be that we want to really change the shape of our data in a more drastic way like pivoting the data and removing the Total row



Basic transformations

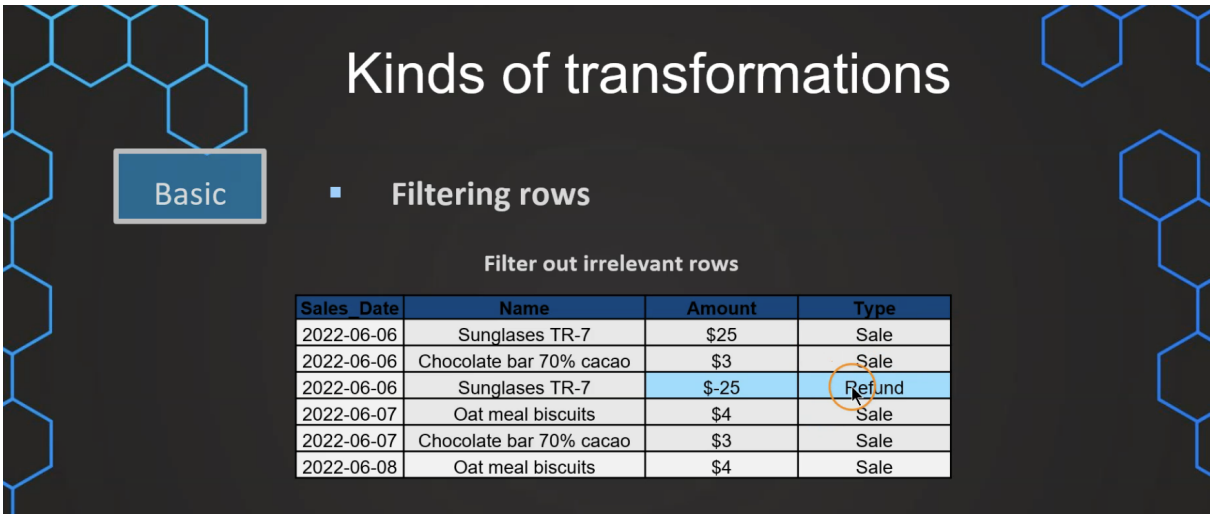




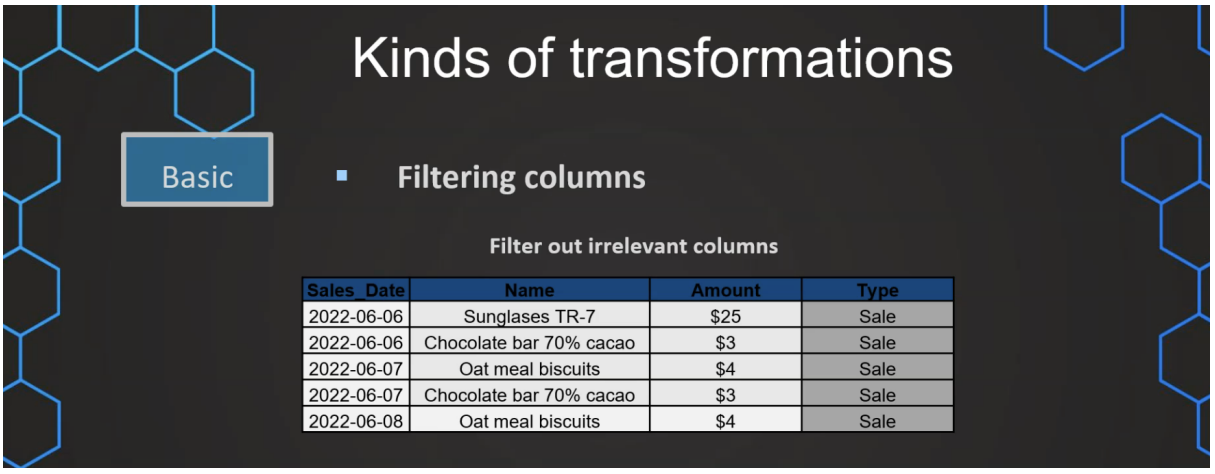


- In deduplication, after combining data from multiple sources and standardizing them there might be duplicate rows which we want to remove

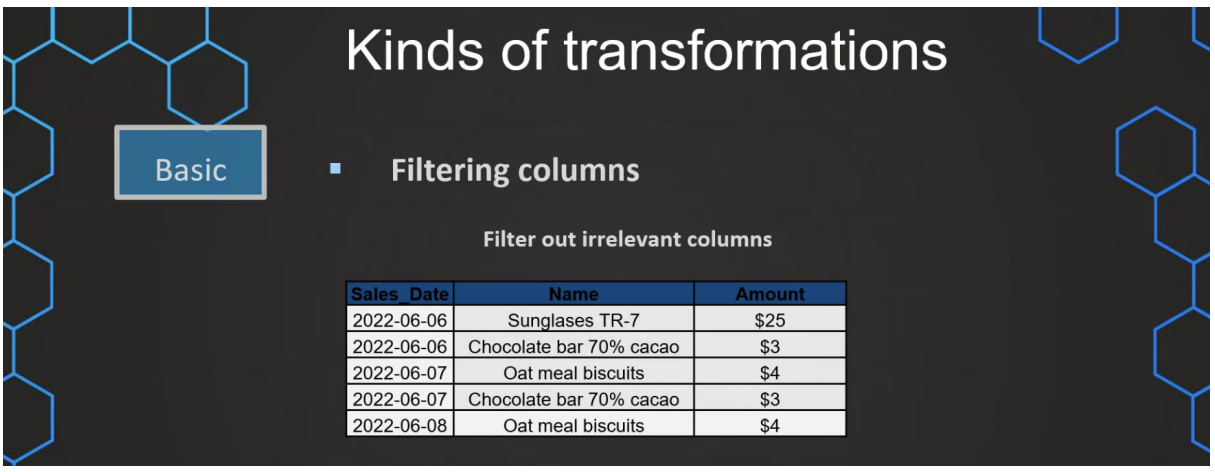
and keep only one using DISTINCT

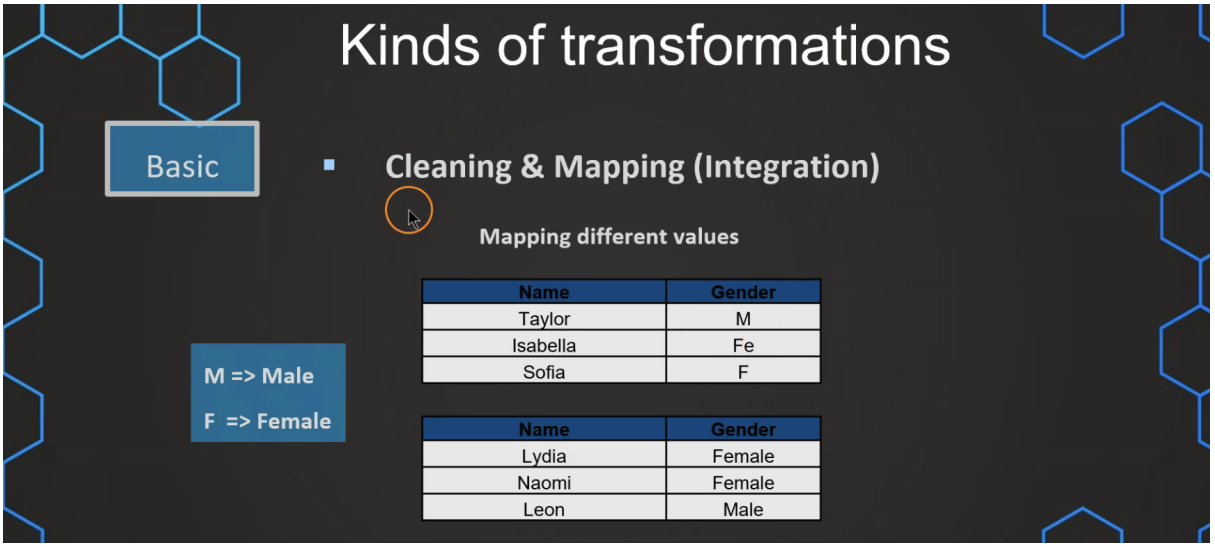


- Remove the values which are not relevant to us like Refund row

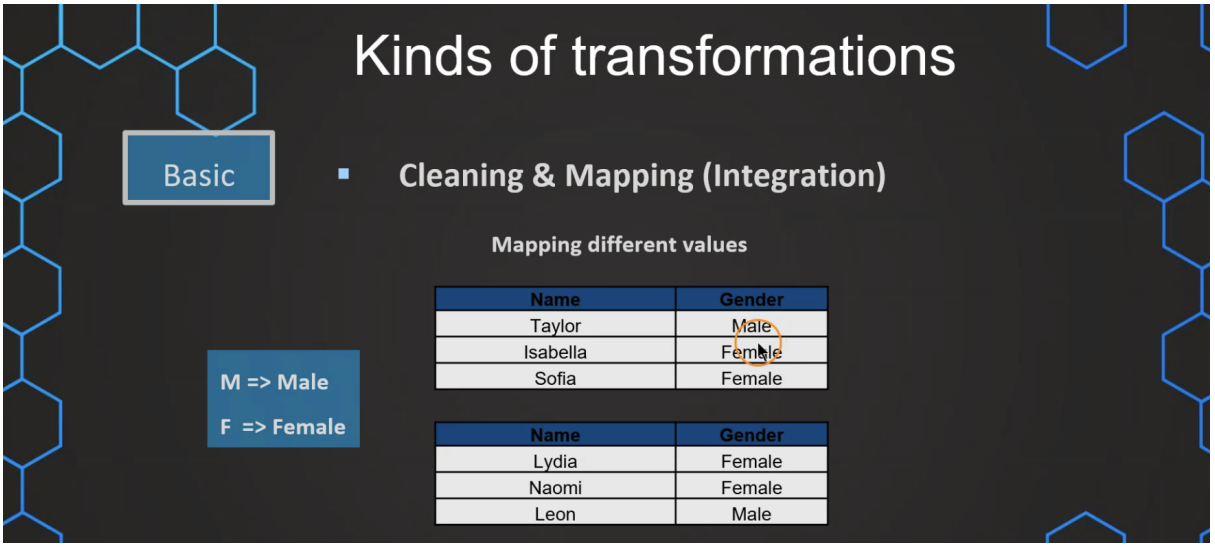


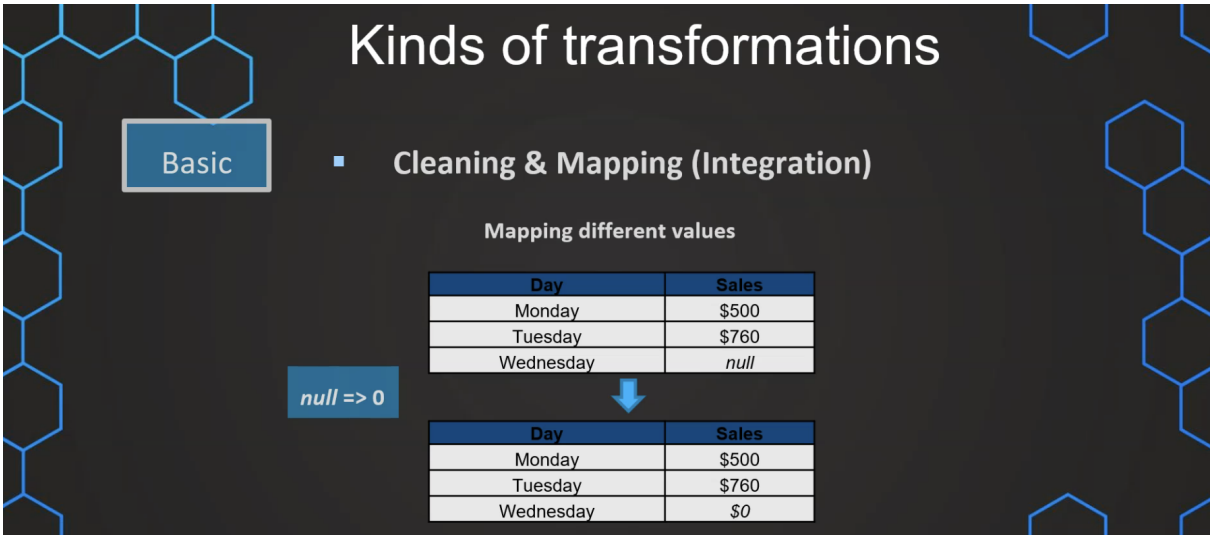
- We can also filter the columns which we don’t need, like the Type column as we have Same value for all rows



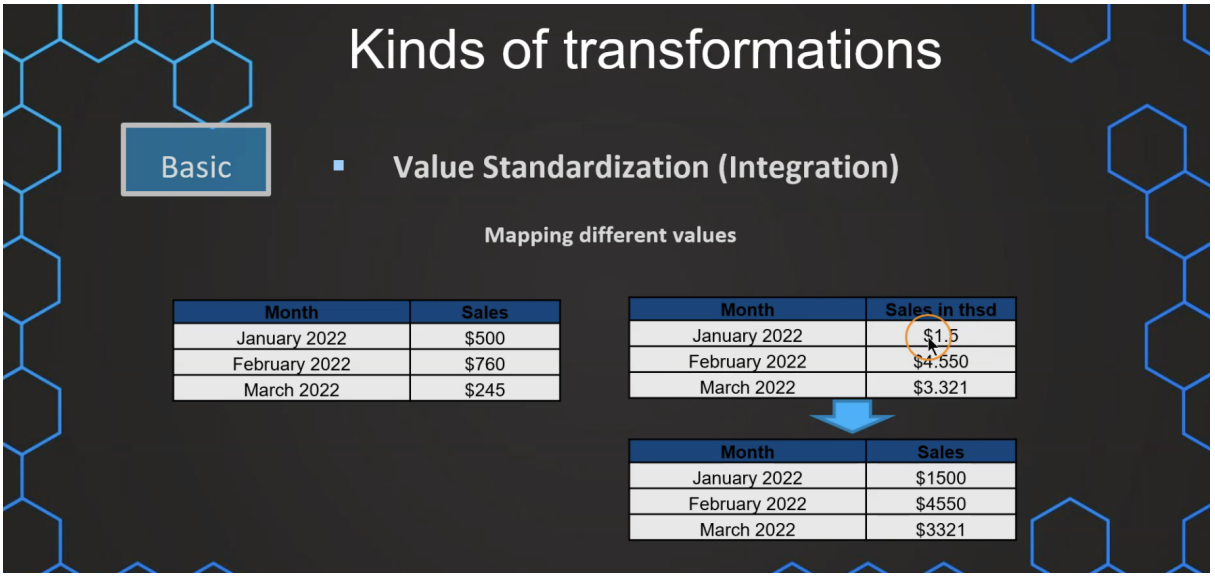


- Since we will combine the data from multiple system, the data might not be exactly same or we would like to standardize the values we are getting like the Male and Female values
- We also like to clean the data if the data is not good like Fe should be made F in 1st table and later we can map the value to Female

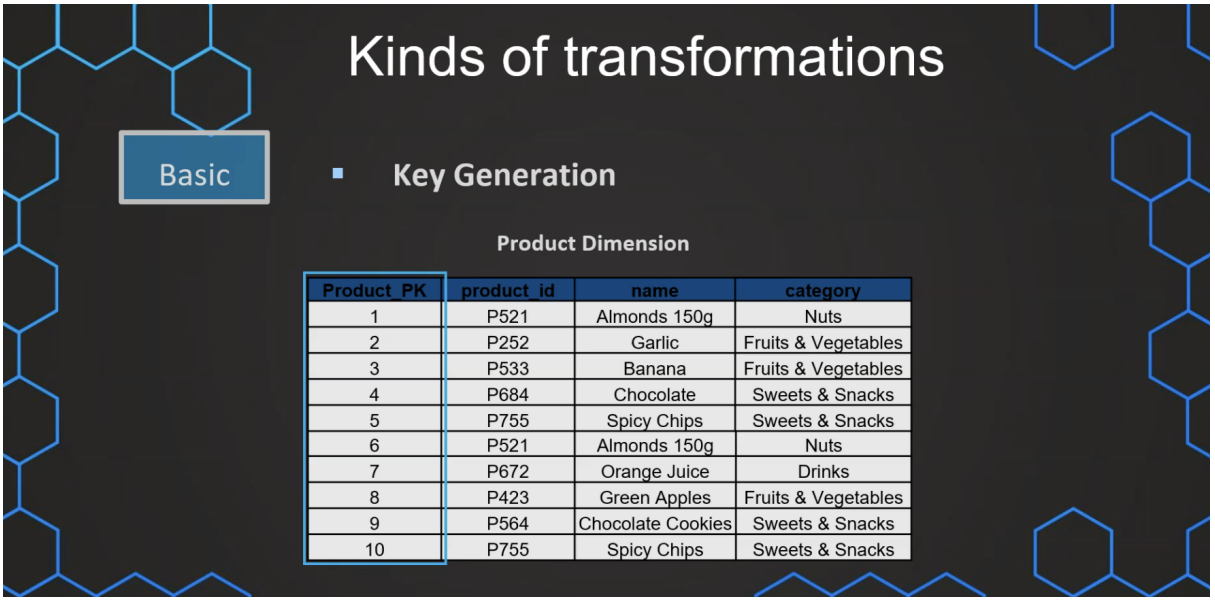




- We want to replace/handle null values



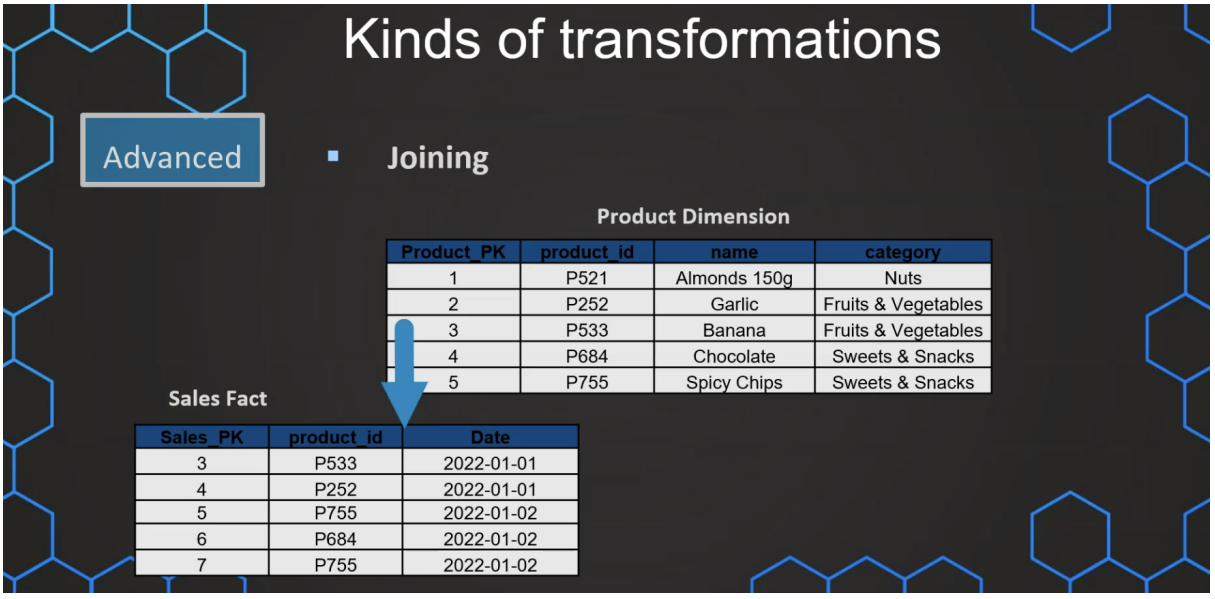
- We want to standardize the data i.e. in one table we might have the data in full amount, in another we might have it in thousands with decimal, we have to convert the data type and values to make it consistent across the data sources



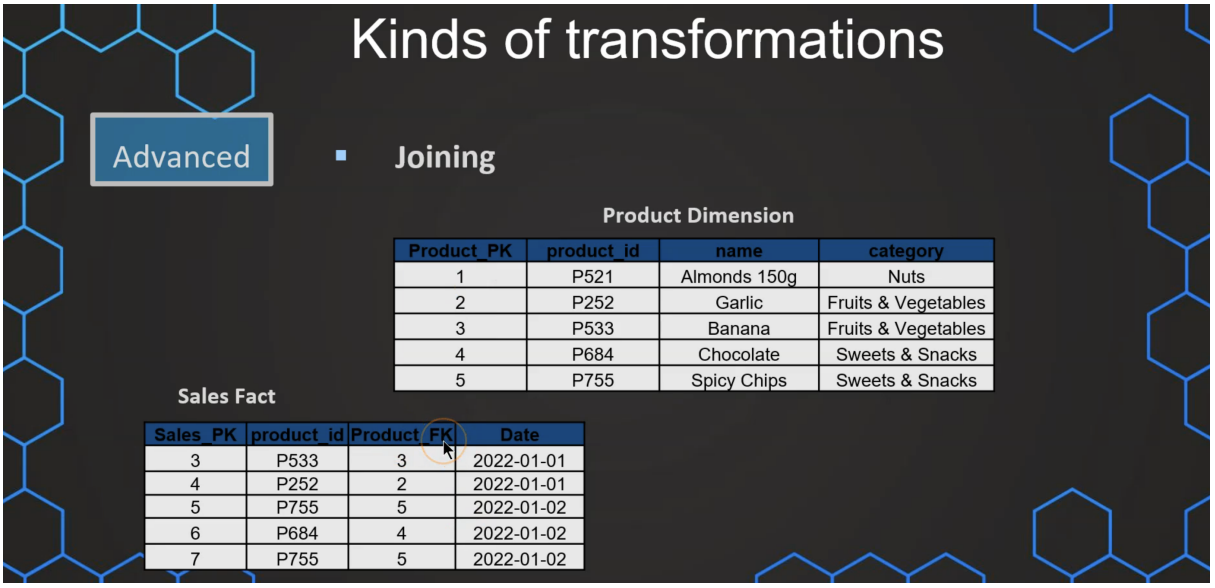
- we usually want to also add a key. And this can be auto-generated either in the database management system or in the ETL tool.
- And this is then our surrogate key, that is the replacement usually of our natural key. So we've learned that this is also something that we want to do and that is advised.

Advanced data transformations

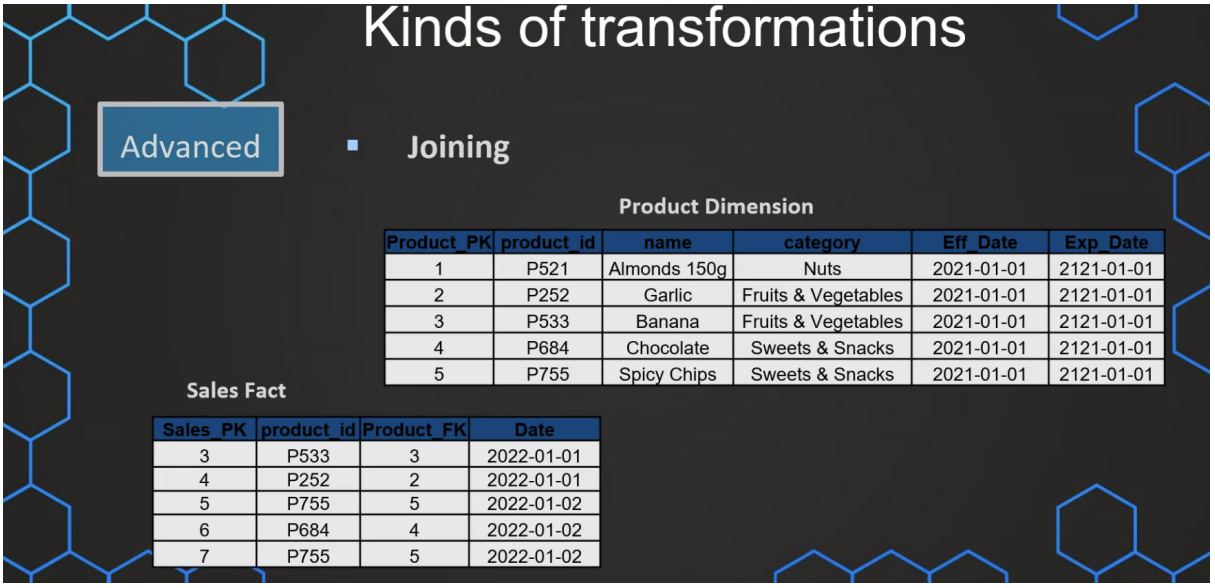
- Joining multiple tables together is something that is oftentimes necessary to get the foreign key into our fact table because sometimes we have only a natural key in our fact table, and that's why we add a surrogate key into our dimension table.



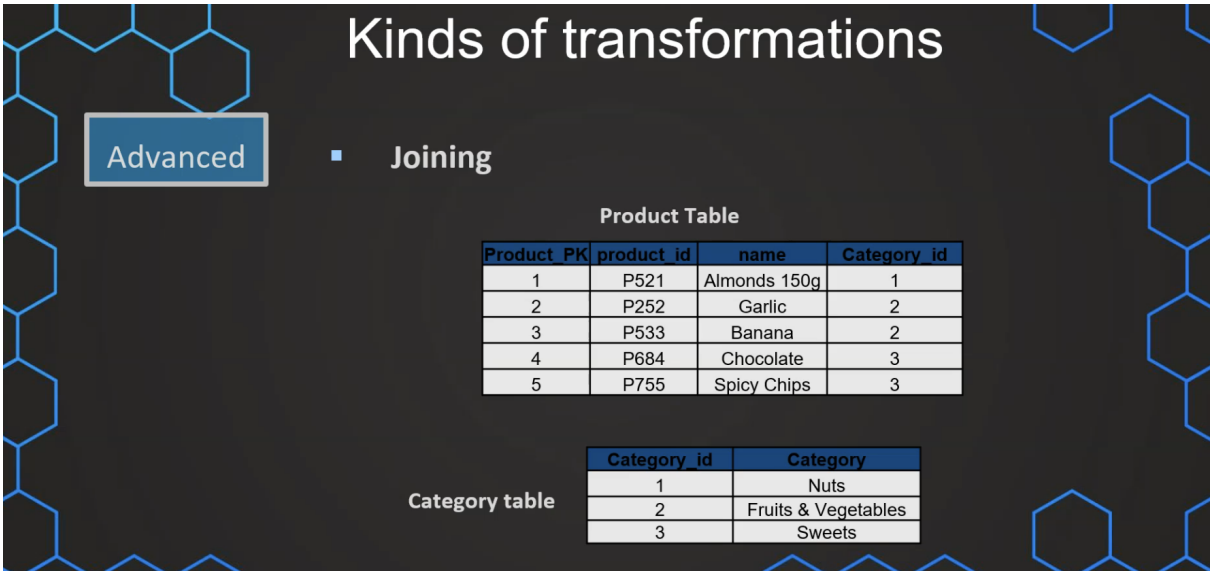
- But now we still need to reference this key as a foreign key in our fact table. So we have only this natural key and now we also want to bring in this surrogate key for the product dimension. And this is something that we do usually by joining the data and we use.
- In our case in this example want to use the product ID as the common column to make a join into our fact table. And with that, we get this additional product foreign key which is now just referencing this primary key.



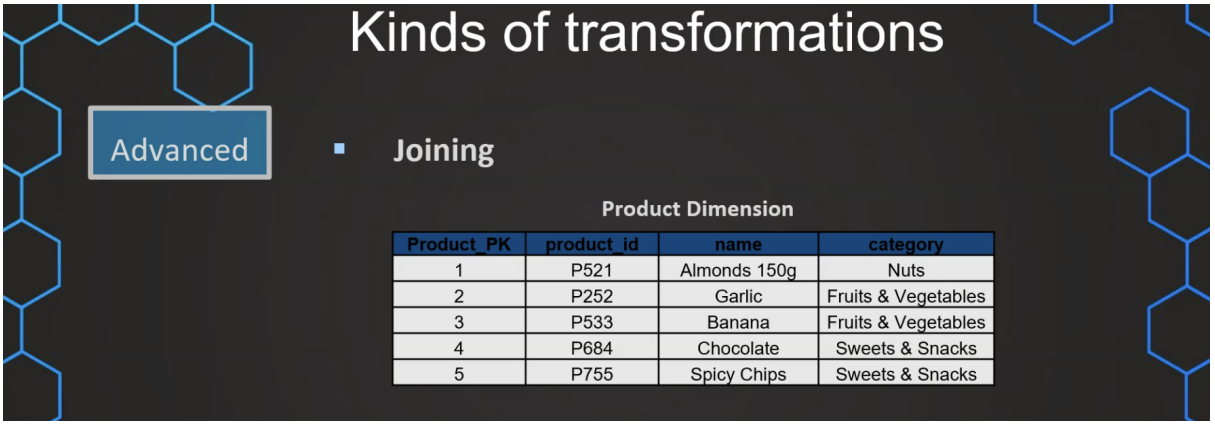
- So this basically works like a lookup where we now get also the foreign key into our facts.



- So this is something that we can do but sometimes if we talk about slowly changing dimensions we can also have some effective and expiry date.
- And this is then also in conjunction with a filtering or a second condition that the date in our effect is in between the effective and expiry date. Because otherwise we could get multiple values of the same natural key, but we end up with only one value if we filter the data to only the values where the current date or this transaction date column is in between the effective and expiry date.
- But this is also more of a specific use case if we are really working with slowly changing dimensions. Otherwise, a simple join will do the job.



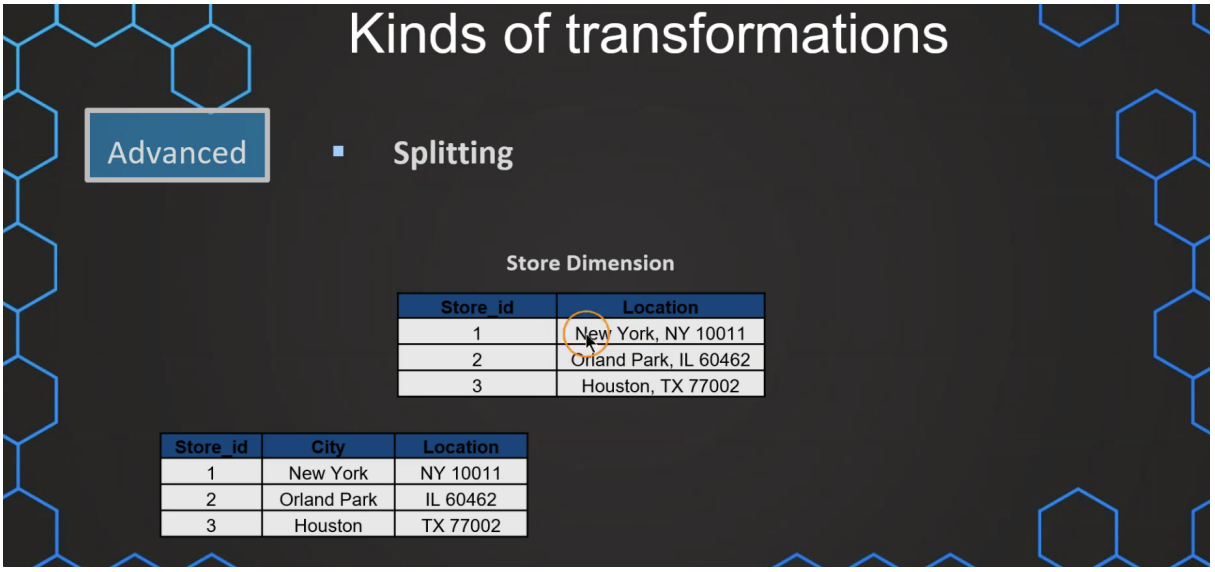
- And of course, also in some cases, we need to just get additional columns into our table. So in this case we have the product table and the category table in separate tables, but now we want to merge them together and we want to bring it into one compact product table.
- This is what we want to do to make it more user-friendly. And in this case, it's also better for the performance so that we don't need to do all of the joins always manually, which of course take also compute resources, and that's why we want to make a join using this category column.



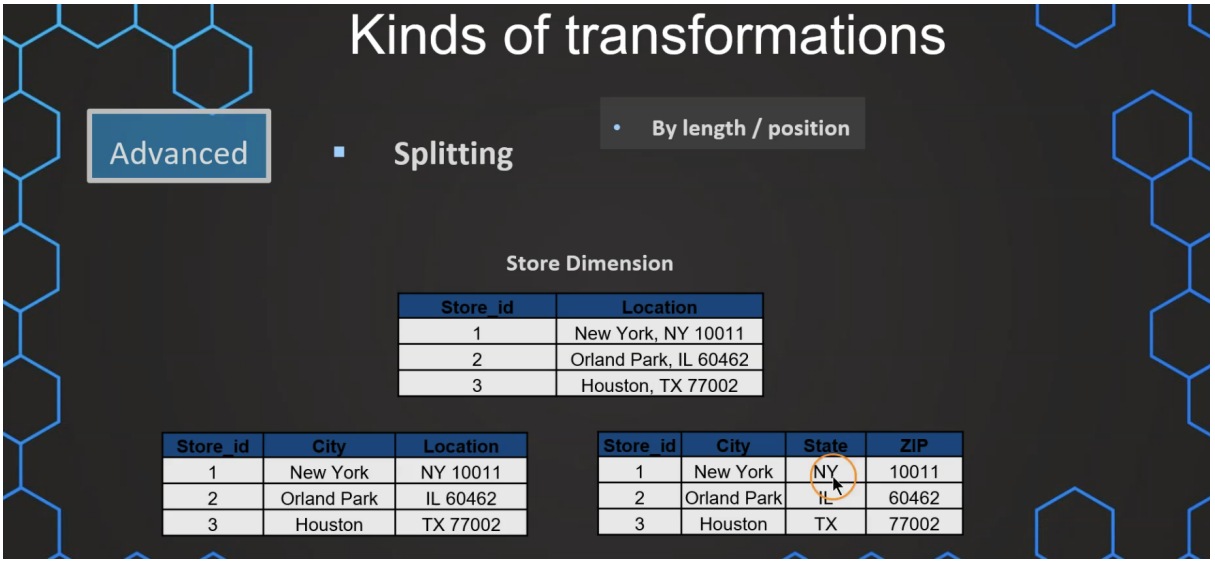
- And then we have one product dimension table that is now a lot better for the users. Another example of a transformation is splitting the data.

- So let's imagine we have this store dimension and in this location we basically have three types of information. We have the city, then we have the state and the zip code and now we want to split the column so that we get this information also in separate columns so that we can use them independently.

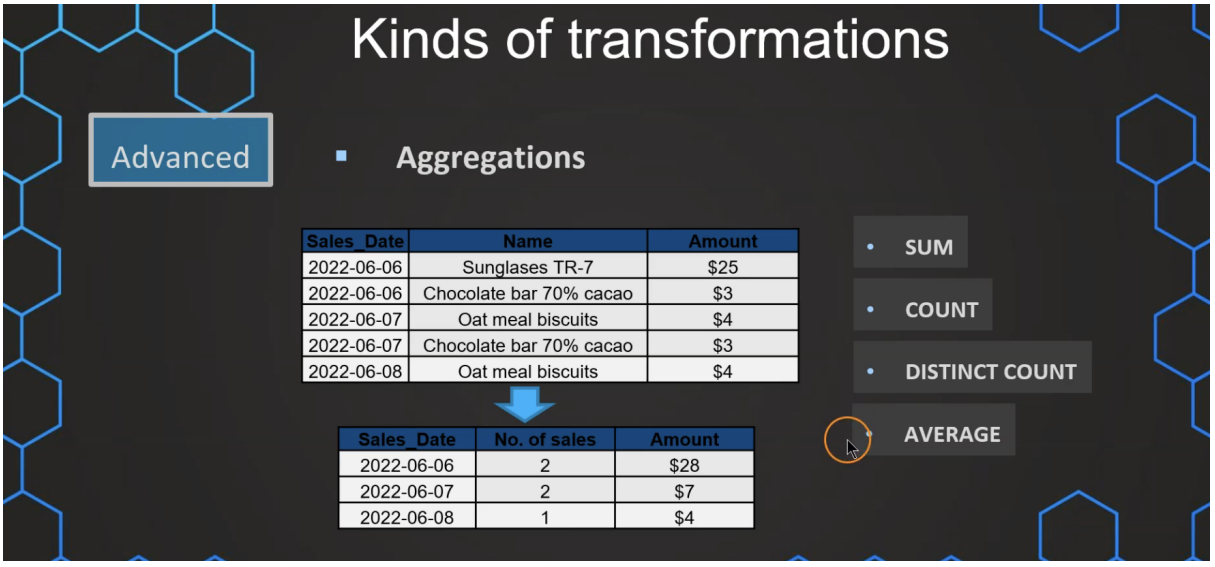
- For example, we can split first by this delimiter of a comma. So always after the comma we want to extract everything. So in this case, New York. Then again, in the second we have all of the other cities. And of course after that we can again split by a delimiter.



- So this could be a white space but also another type would be by length or by position. So that means always at the third position, make a split and put the first characters into the first column and the rest of these characters into another column.
- So for example in here we could say split after two characters. So then the first two letters would land in the first column, and then the rest would land in the other column, the zip code.
- So these are things that we can do either splitting by length or position or by a specific delimiter.

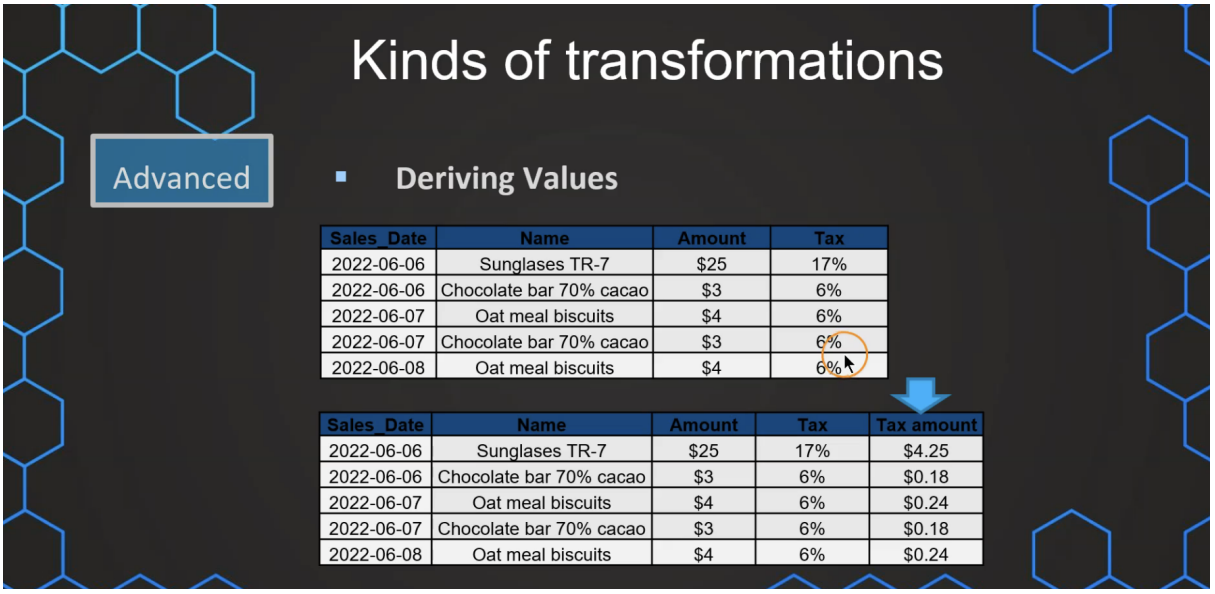


- Another thing that we can do, especially if we want to change the granularity, we can oftentimes aggregate the data and different ways of aggregating the data could be taking the sum.



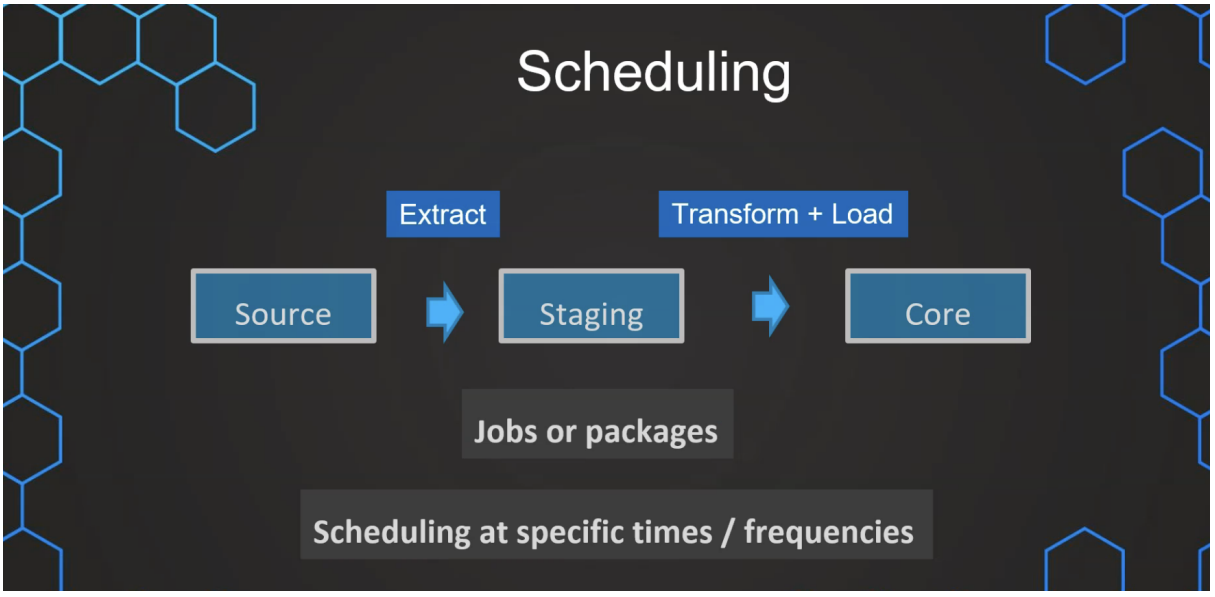
- So for example, in this case, we have the granularity of a single day, and that's why we want to sum all of the amounts of our sales for every single day.
- But also we want to count the number of sales. So we want to count basically the number of rows because one row is just the same as one sale one transaction. But for example, if we also want to understand how many different products we have sold, we can also use things like a distinct count.

- And then we just get the distinct number of values but also we can use other aggregations such as for example, an average.

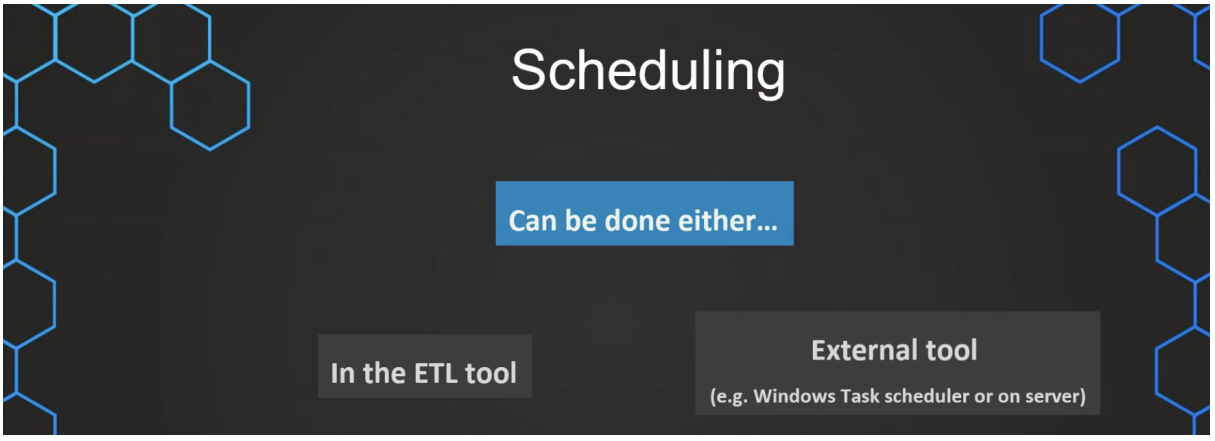


- And last but not least we can also calculate additional values. So we can subtract things or we can multiply things or we can also create groups.
- And in our case, we just want to calculate the tax amount in an absolute value. So we have, for example, the percentage value which is not so nice to be aggregated because we cannot sum these values up.
- We can see it's non additive, but we can calculate the absolute amount because we have also the sales amount. And with this tax percentage value we can create this additional column, tax amount which is basically the product of those two columns.
- And like this, we can get additional information and additional insight by deriving specific values.

Scheduling jobs

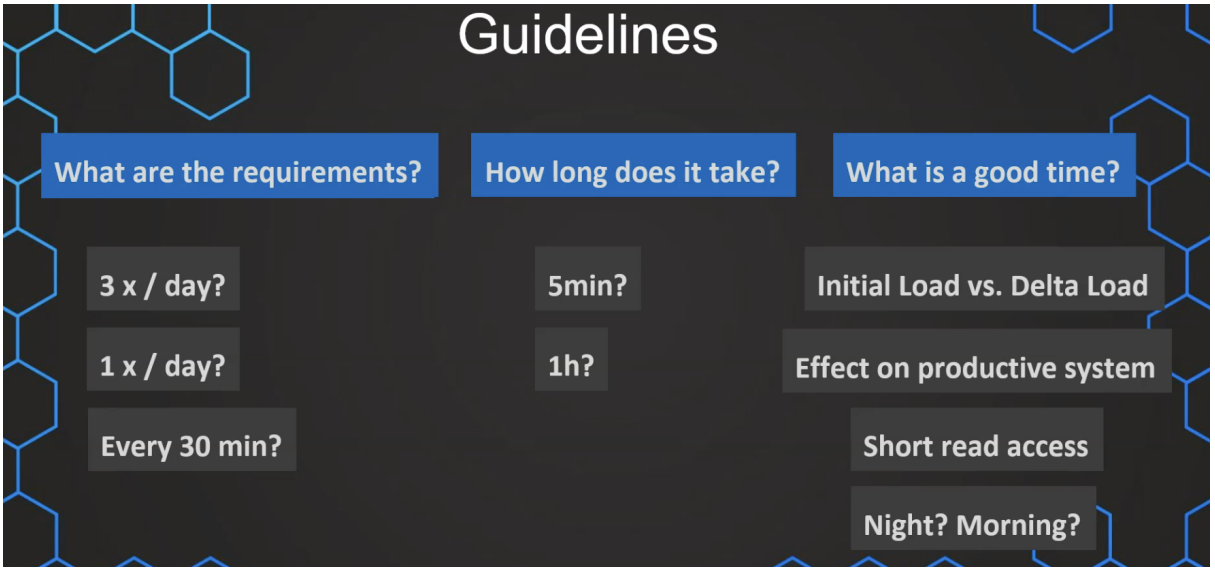


- Once we have developed and implemented our ETL process, we of course also want to schedule this job so that it's continuously extracting, and transforming, and loading the data so that it's always as up to date as we need to have it in our data warehouse.
- So usually, we can package our workflows, our transformations into jobs or packages. The terminology here is just depending on which type or which ETL tool we are using exactly, but in general, those jobs or packages can then be also scheduled at specific times or specific frequencies.



- Scheduling can be either done in the ETL tool itself. This usually requires an enterprise version, so for example, in Pentaho, it's in the Community version.
- In the ETL tool itself, not possible to schedule those jobs, but we, of course, have other possibilities as well.

- So if this is not possible in the ETL tool, because maybe we are using the free version, then this can be also just scheduled using an external tool, and this external tool just needs to execute basically this drop, and this can be done with Windows Scheduler or other things.
- And oftentimes, also, we want to deploy those packages or drops onto a server that is there specifically to execute and perform these ETLs, so this is something that is usually then also done.
- We need to start with the question, what are actually the requirements of the business users?
- So this requires us to talk to them and ask them how frequently do they need to have the data updated? And this is then, of course, our starting point, and of course, we are trying to fulfill those requirements as good as it's technically possible.

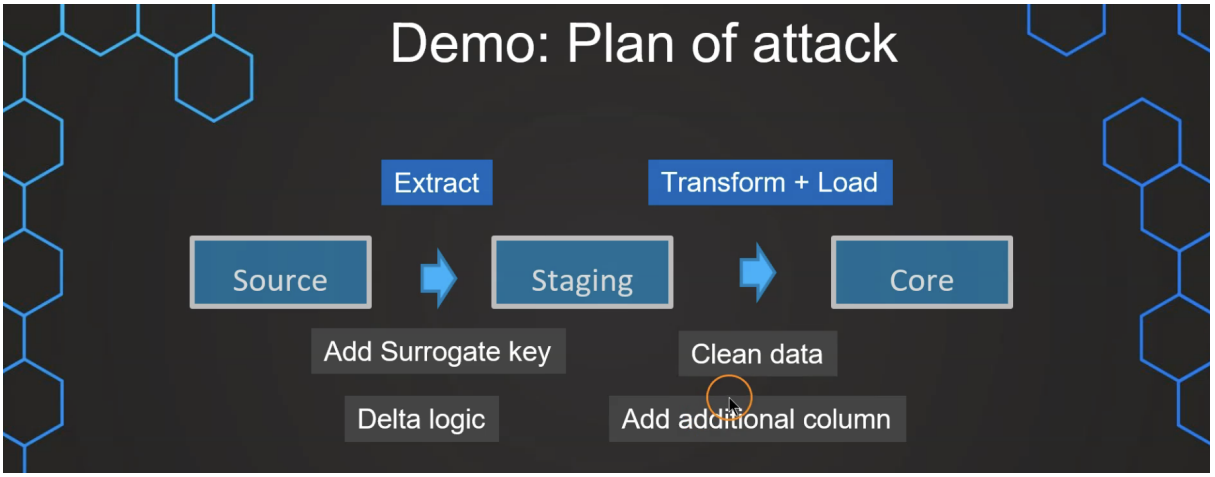


- And then if we have those requirements, of course, this is the wish of the customer or the business users, we need to face reality, and have a look at how long does our ETL actually take.
- So some ETLs can perform all of the operations in just one minute, in five minutes, or 10 minutes, or sometimes if we have larger amounts of data, and everything is a little bit more complex, it can also take up more time, and if the users require 30 minutes updates, but the ETL takes one hour, it's very easy to understand that there is some conflict here.
- So that's why if we have those two things, so how long does the ETL take? And how frequently does the customer or the business users want to have

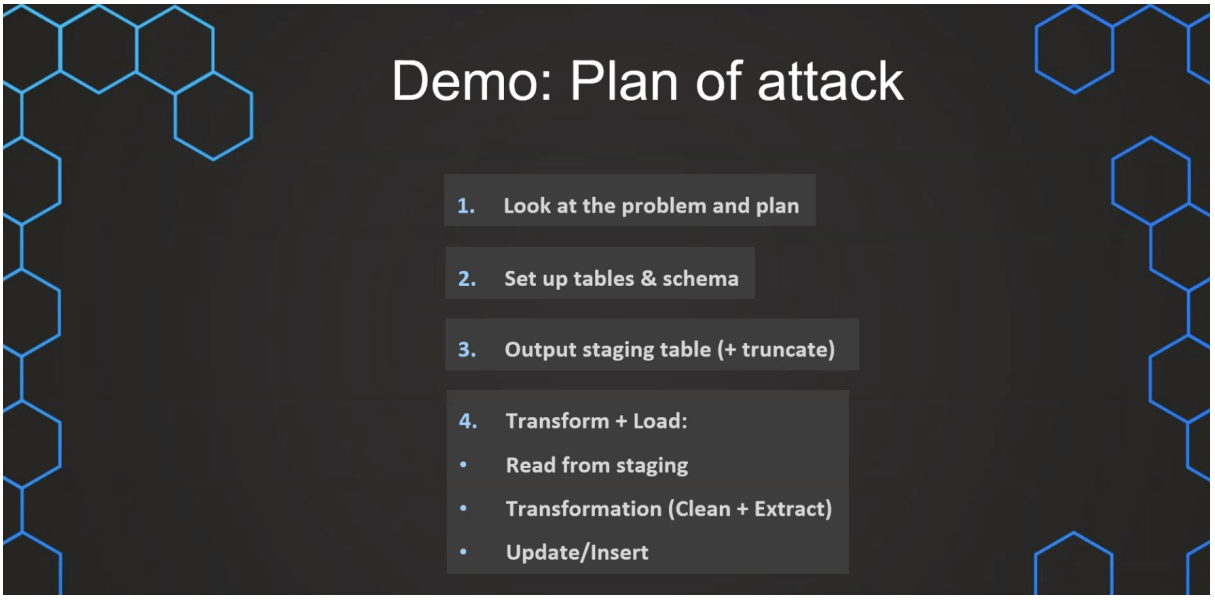
the data updated?

- We can find maybe a compromise, or maybe also completely fulfill the requirements. And then once we have figured out that frequency, we also need to talk about what is a good time for our ETL.
- So when should this be executed? Because we have to keep in mind that we are accessing productive source data, so oftentimes, these are productive systems, and the time that we spend with our ETL on these productive source systems for reading the data, this can have an effect on those productive systems.
- So we might slow them down, or even block them entirely if we are using up the resources, and maybe blocking them from doing their own operations.
- So that's why we need to consider this effect on the productive systems, and we can figure out the effect by, for example, just testing some quick data extractions, and then determining together with the relevant people that are responsible for those source systems how strong the effect is, and this is something we have to just discuss with the relevant people.
- And of course, there's a big difference between the initial load, because this takes usually a lot more time, and is a lot harder in the load, and then of course, also later on, it will be much easier with the much smaller delta load
- this is why we are doing the ETL in general and building our data warehouse, that we are just very shortly reading the data, so this is usually not a very heavy load, and if it's a delta load, this is usually an effect that is not so bad on these productive systems, so they can usually handle that
- But still, we should figure out a good time, and sometimes, if we especially talk about the initial load in the night, maybe in the weekends, or maybe before 6:00 AM in the morning, these can be all good times, but in the end, of course, as mentioned, we should talk to the responsible people, and then we can also just make some testings and discuss it together with them for what would be a good time to execute our ETL

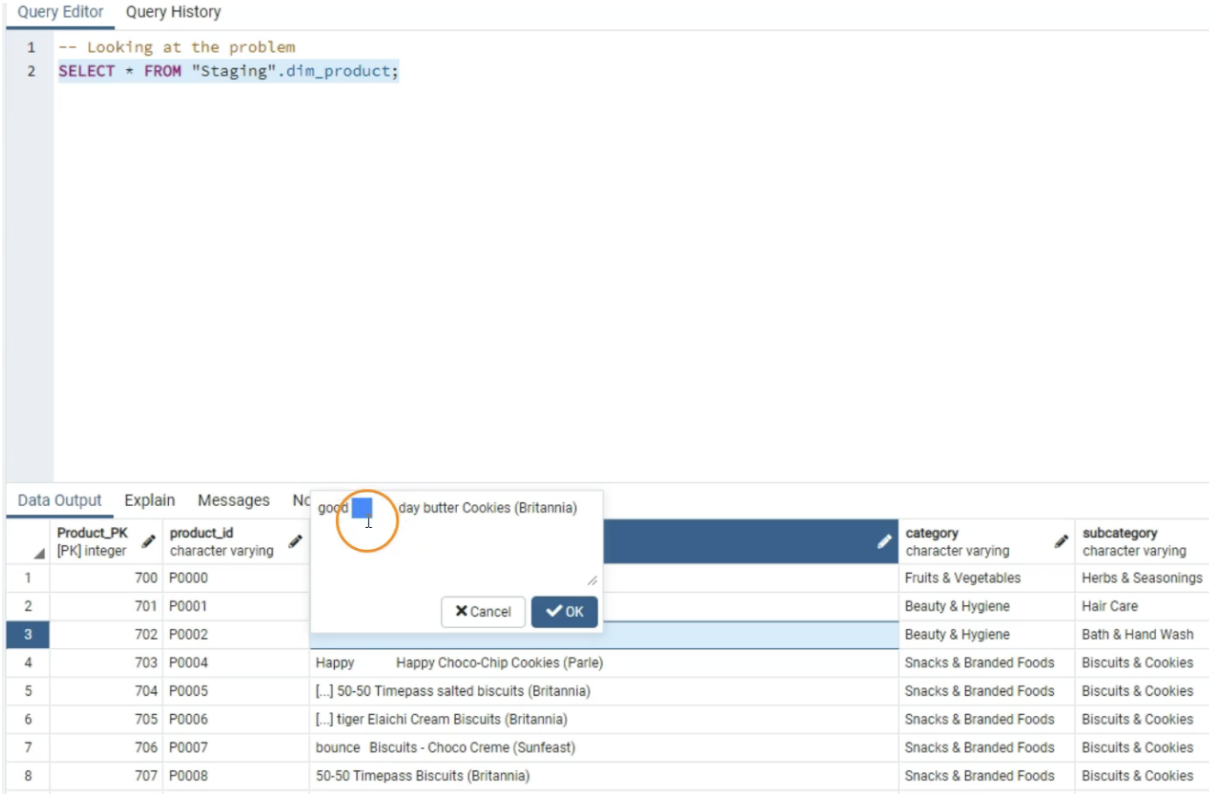
Demo: Planning next steps



- So we have seen that we have a data source, our product table, and in a way, we've already at least partly set up our staging. So we have set up the extract. So we also have added a surrogate key in the database management system. And with our variables, we have also defined the delta logic.
- We still need to output the data that we have gathered in the ETL and write that into our staging table. This we still have to set up. And then after we have set up this extract part, we want to now also load the data with some transformations into the core.
- And the transformations will be some cleaning and we want to add an additional column and we quickly also see what kind of column we want to also set up.



- Now we want to have a look at the single steps that we now need to implement
- First, we want to have a look at the data and what the problem is with the data, what parts we still need to clean and what kind of additional column we want to set up. So this is what we now want to start with and that's why we want to jump into pgAdmin. And in here we just want to first open up the query tool.

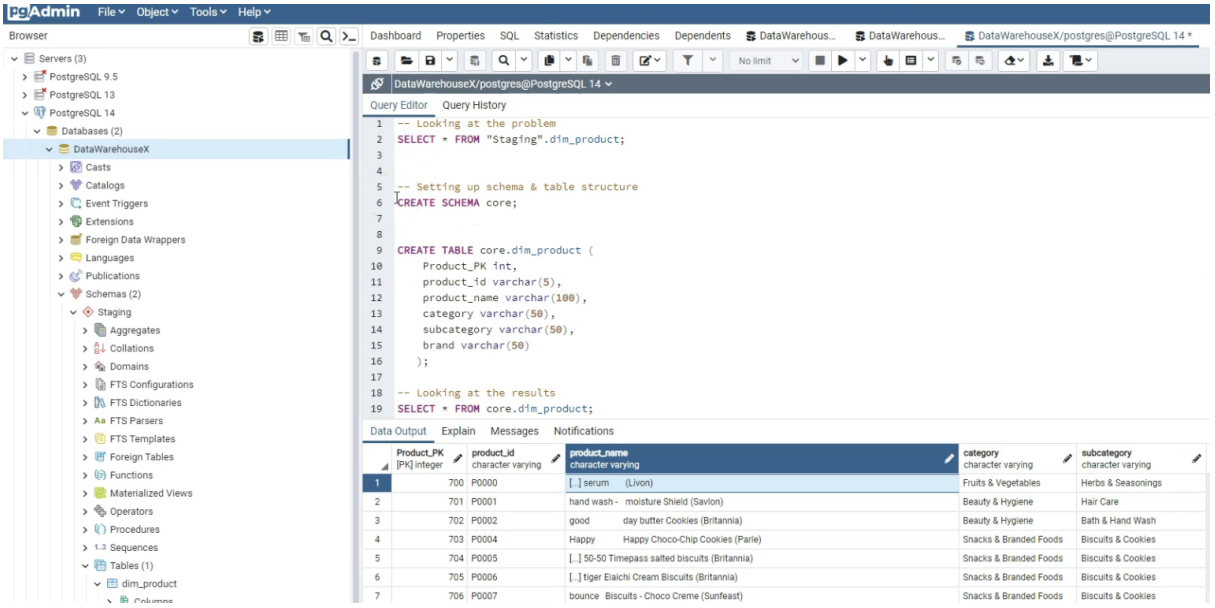


- we can see that the column product_name is really not very clean. There are some special characters. We can double click and see that there are some tabs included.
- So this is not white space, but really some characters that we don't want to have. In this case, this is tabs. And also sometimes as we can see in here, in the beginning even, we have also some characters and we want to get rid of that. So this is the cleaning part.
- And afterwards, we also notice that in parenthesis, we have also, in this case, the brand name. And we would like to extract everything that is here in parenthesis because this is better in a separate column because it's a separate information of our brand.

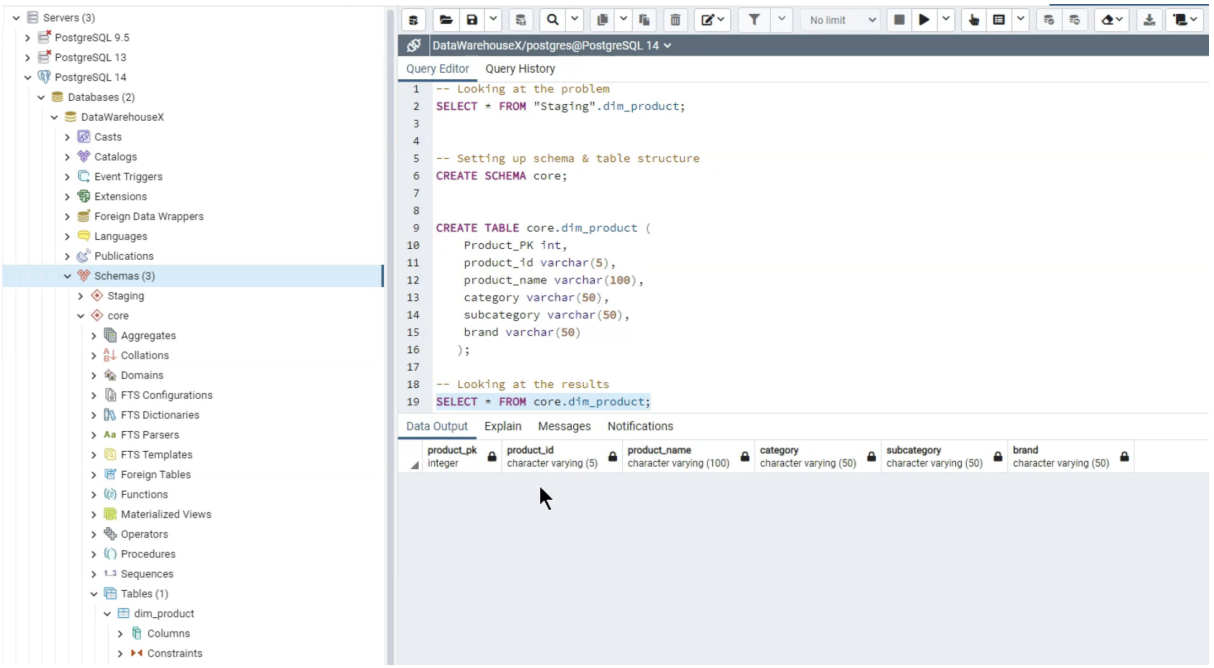
- And if we have that in a separate column, we can group the data also by the brand. And like this, it will be much nicer and we have more analytical value in that. So this is the problem.
- And now what are the steps we are going to do to solve this problem and continue with the creation of our ETL? That's why let's jump back into our plan of attack.
- So the next thing that we now need to do is we want to set up the tables and the schema.
- This is what we define in pgAdmin. So we want to set up the core layer and this will be just a schema. So a schema that is called core. It could be also a separate database for our core layer but we want to just go with a separate schema.
- In that, we can then set up our table, which will be our product dimension. And then afterwards, we need to still complete the staging layer in the ETL tool. That means that we want to also output the data that we have gathered from the delta.
- And also note that before every run, we always prior to that need to also truncate the table. That means we want to remove all data from the table.
- And then once we have finished this extract part, we are now ready for transforming the data and then loading it into the core layer.
- And this will be done by just reading the data first from the staging. Afterwards, we can then set up our transformations.
- We have learned we need to clean it and then we want to extract the data from these parentheses to set up a separate column.
- And then, of course, last but not least, we want to load that data also into the final table in our core schema. So this is our plan of attack.

Demo: Table setup and complete staging

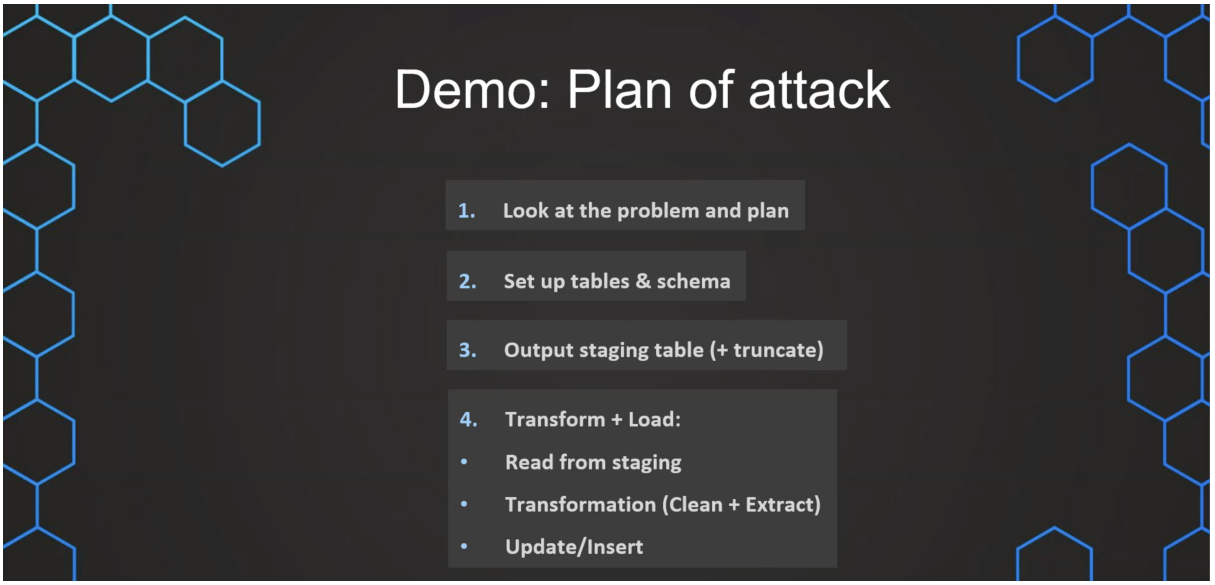
- Now we want to start with first setting up the tables and the schema.
- So we want to go therefore to PG admin. And in here we now want to set up the schema. The first step is to create this core schema. This is how we want to organize our data warehouse



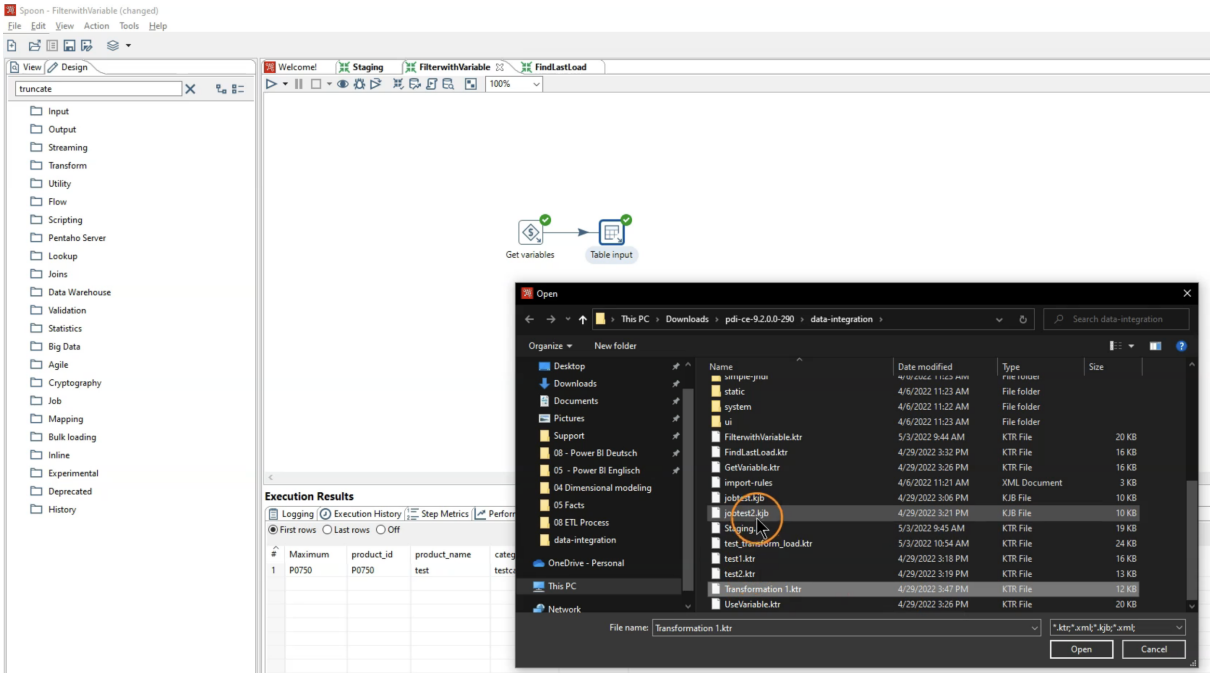
- So now if we refresh the schemas, we can right click and then select refresh. We have now this core. But of course, we don't have any tables included yet. And this is also what we would like to set up.
- And as we want to also add this additional column brand, we have that included as well. So that's why we want to set up this table structure. We highlight everything of create statement and then with F5, we can execute that.
- And now again, if we refresh, we will see that now there is also a table included, so this should be done if we refresh the schemas, we can now see that table.



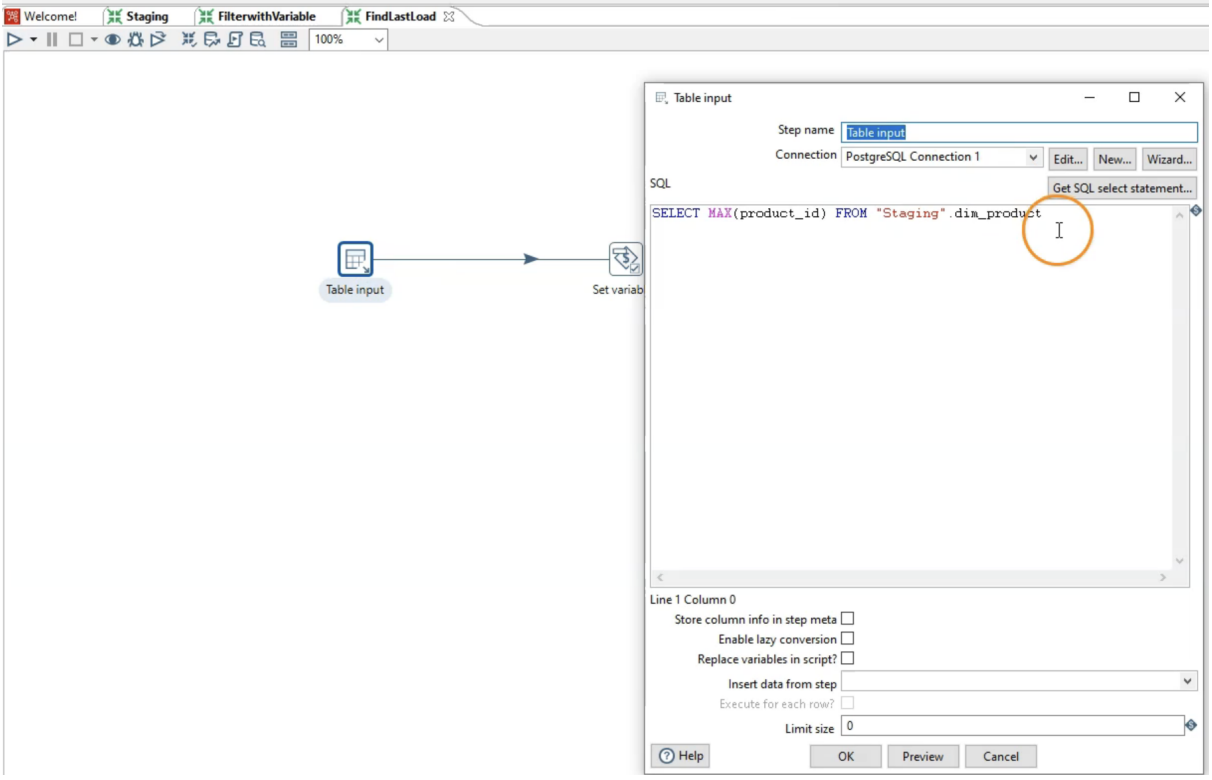
- But of course if we want to query from that table, we can see that currently this is empty. That's why we need to set up things in the ETL tool.
- But now as we have done that, we want to move on to the next step which is to set up our output for our staging table. So this is something that we still need to do and that's why we want to jump into Spoon.



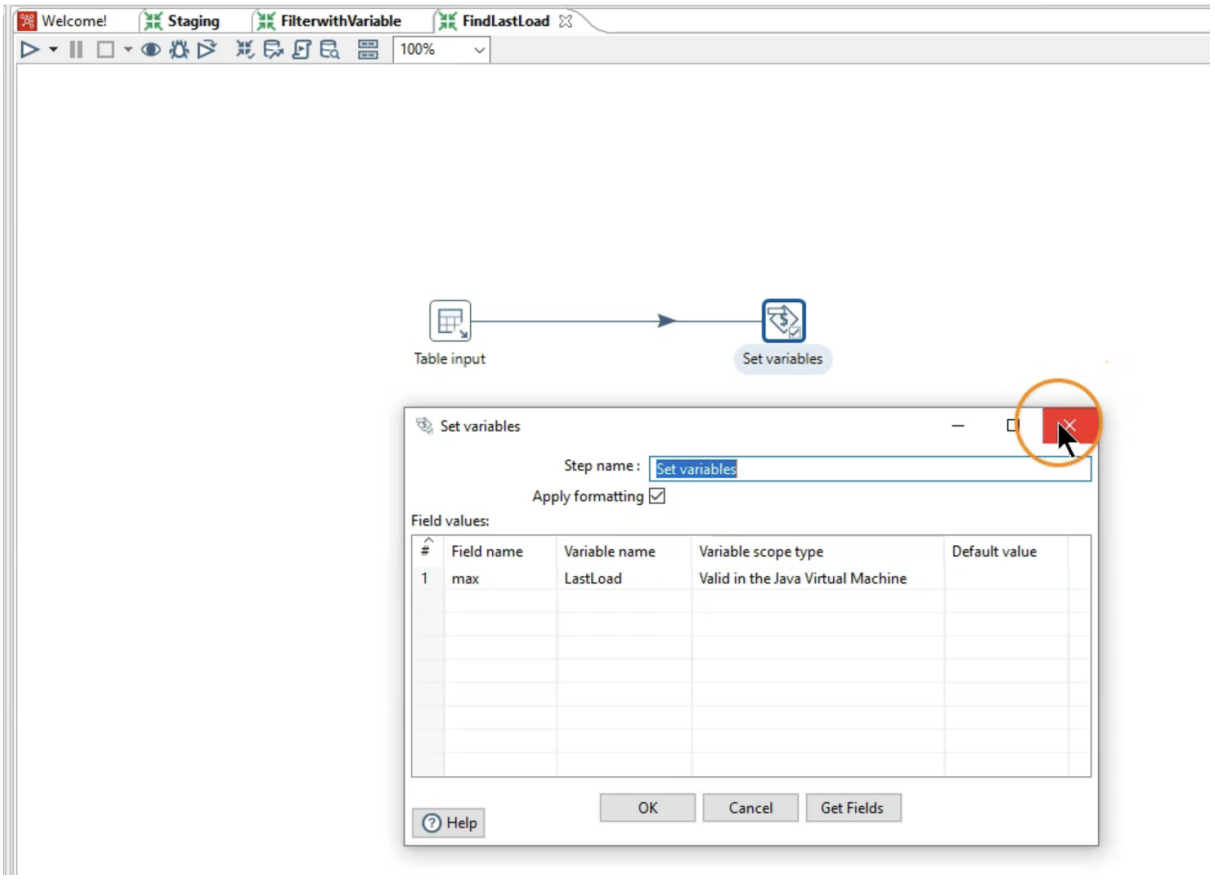
- Go to File>Open and open up those KTR files which we have previously saved.



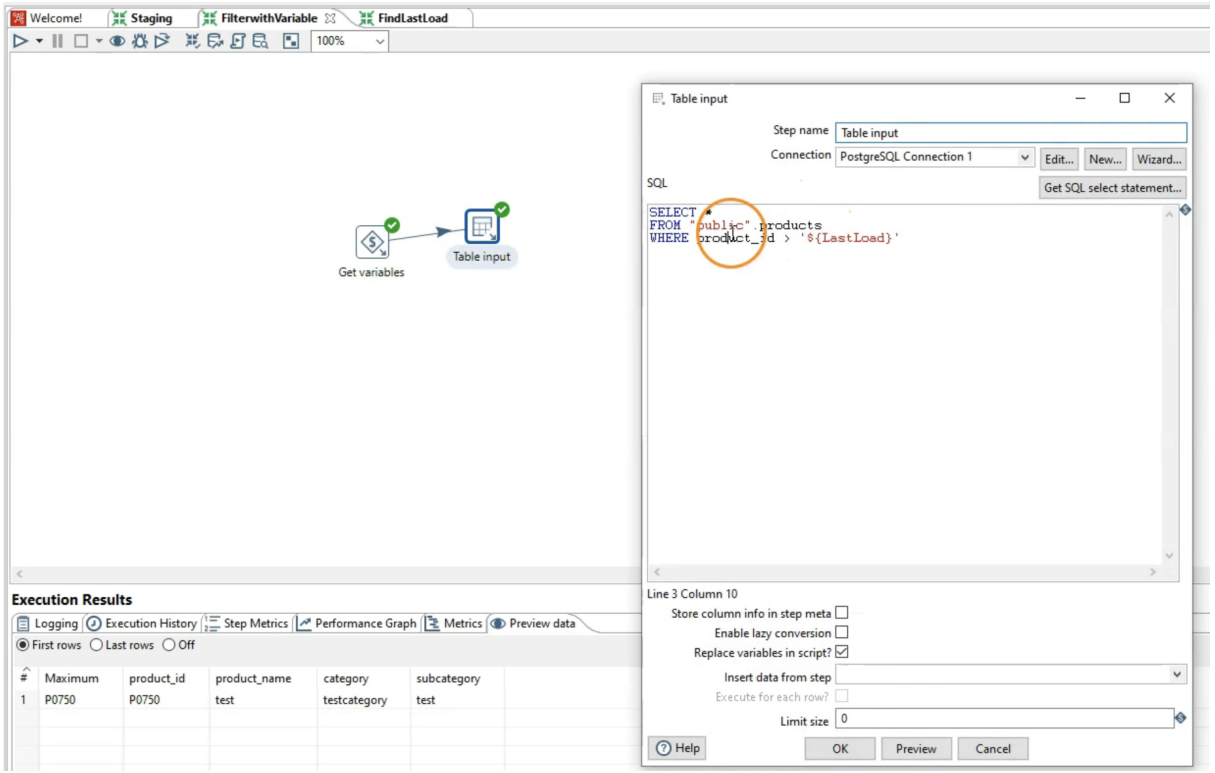
- So we have first find the last load where we have read from the table. So this was just getting the last value that we have loaded previously.



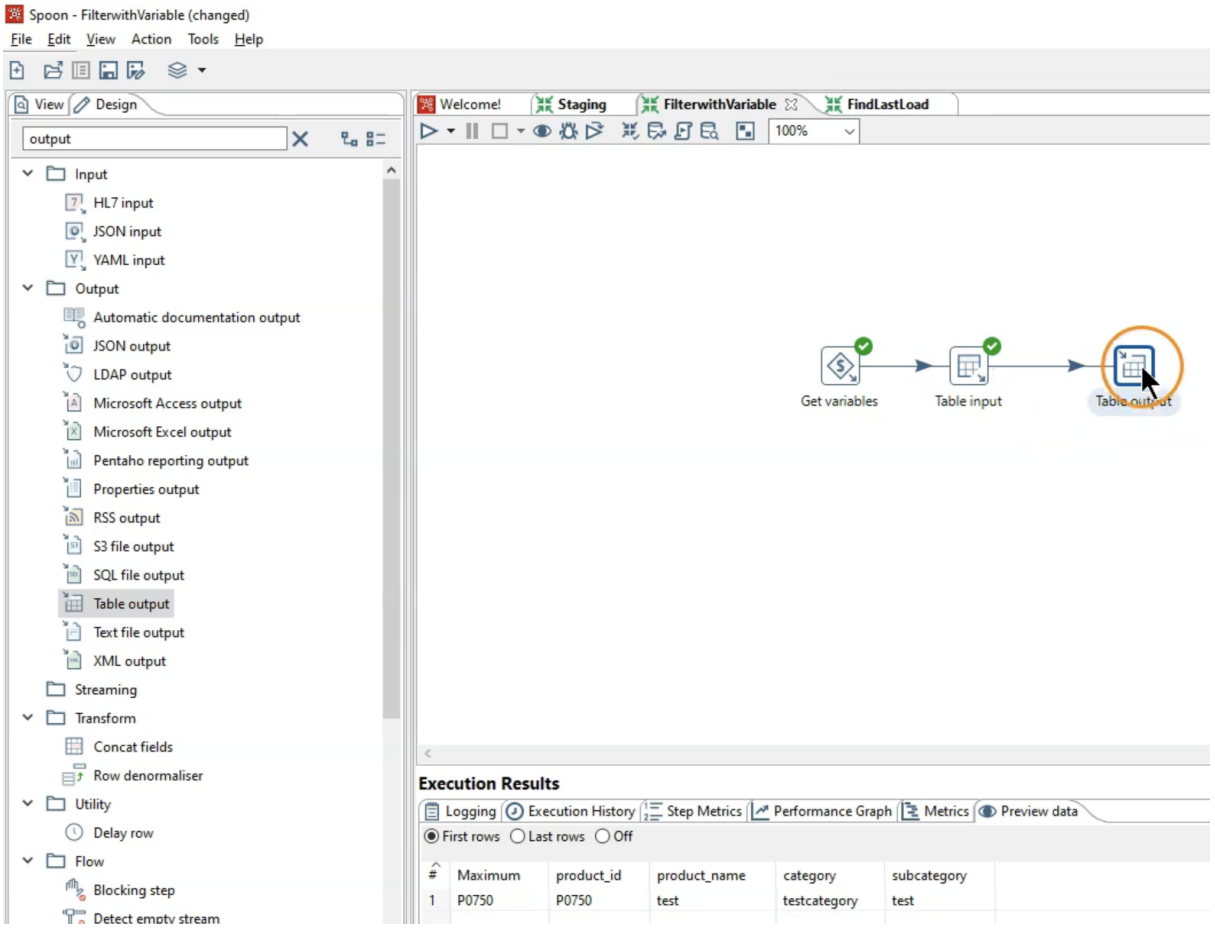
- And then we have stored that in this variable called last load.



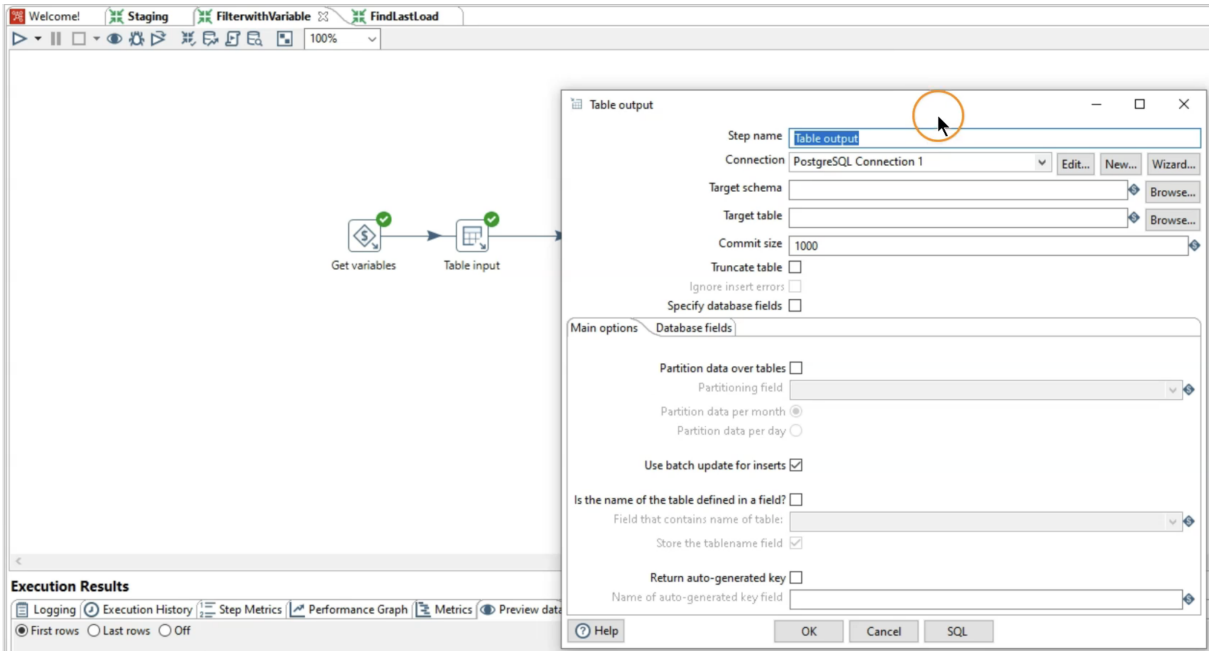
- And then afterwards we have gotten this variable and then read from the source table with this variable included.



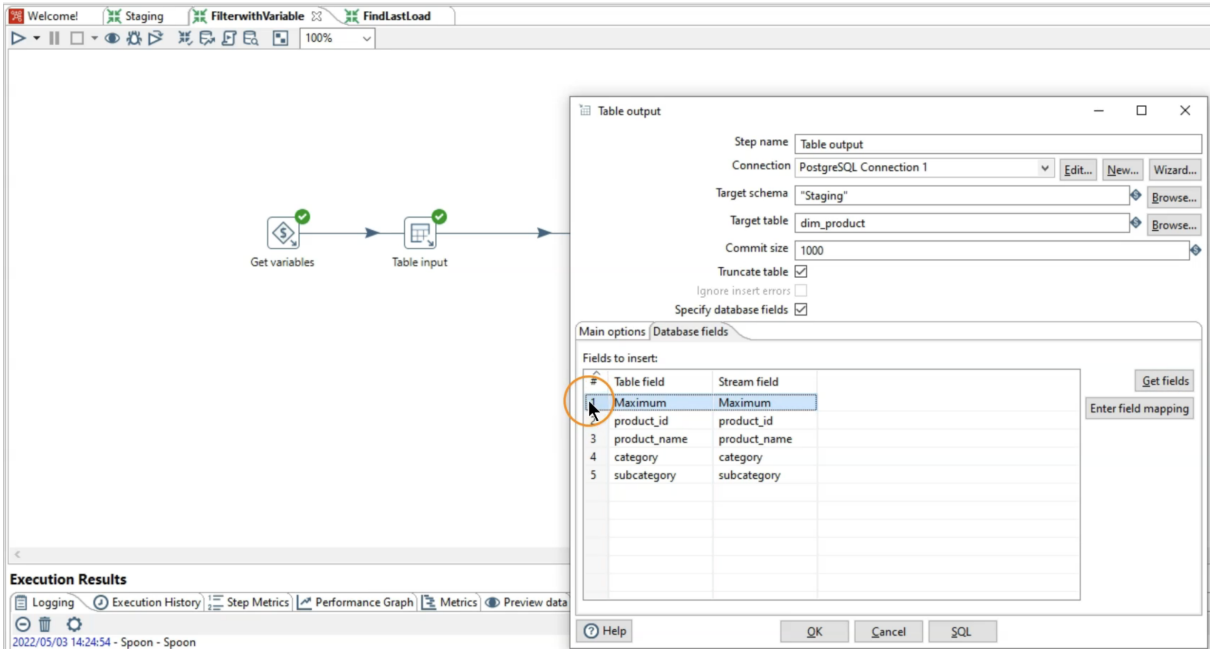
- And now of course this output that we have gotten in here, we want to write now back into our staging table.
- So this is what we still have to do and that's why we want to search for ***output in the left side bar***. And then we can find output table and we can drop that in here, connect it again with holding the Shift key and then we can simply double click.



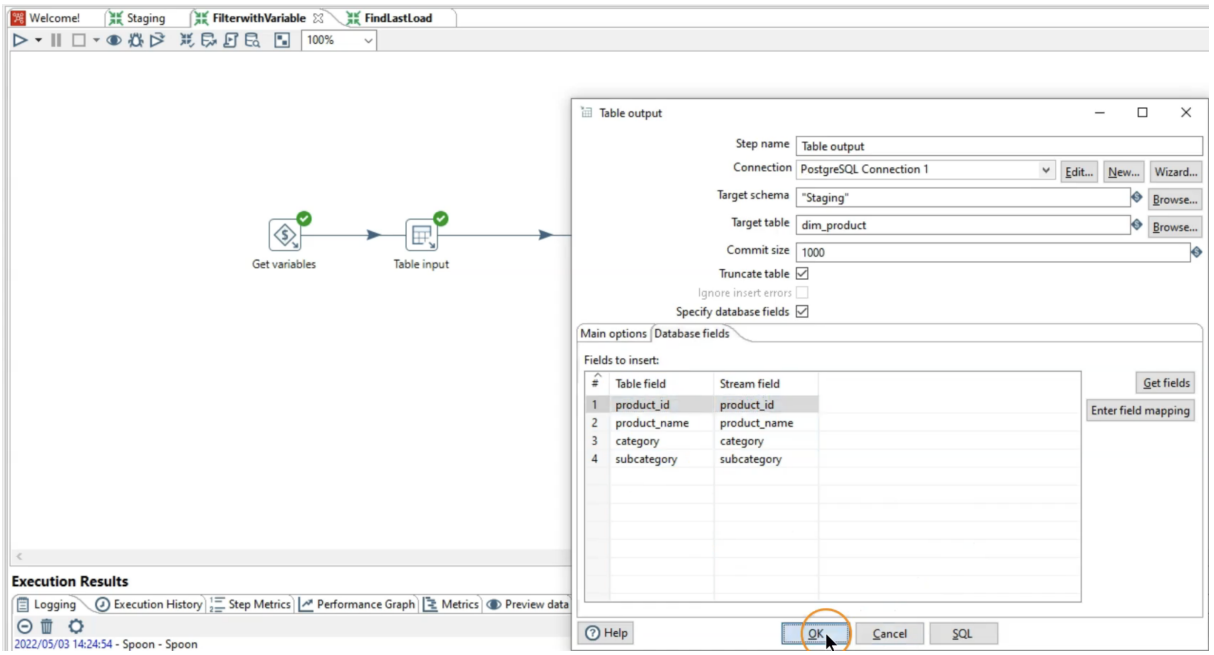
- The connection should still be included since we have previously shared this connection. So it's available in all of our transformations.

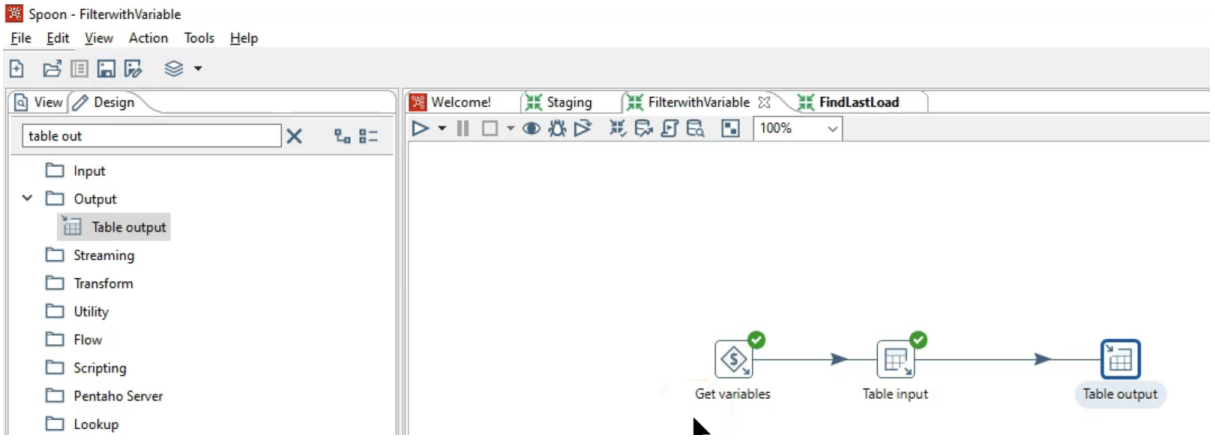


- Select the target schema with quotes to make it case sensitive while searching in postgresql
- Select target table and dim_product and make sure to check the truncate table so it is always truncate and load for staging layer
- Go to Database Fields and click on Get Fields

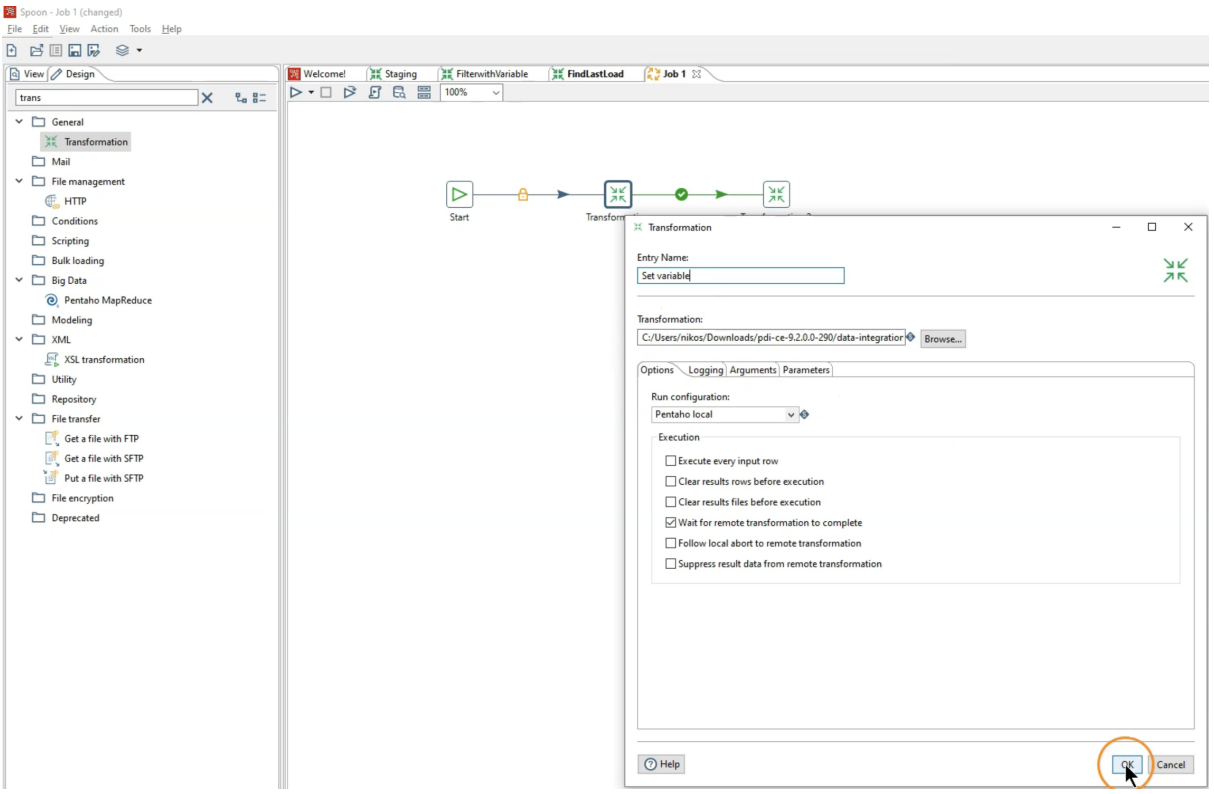


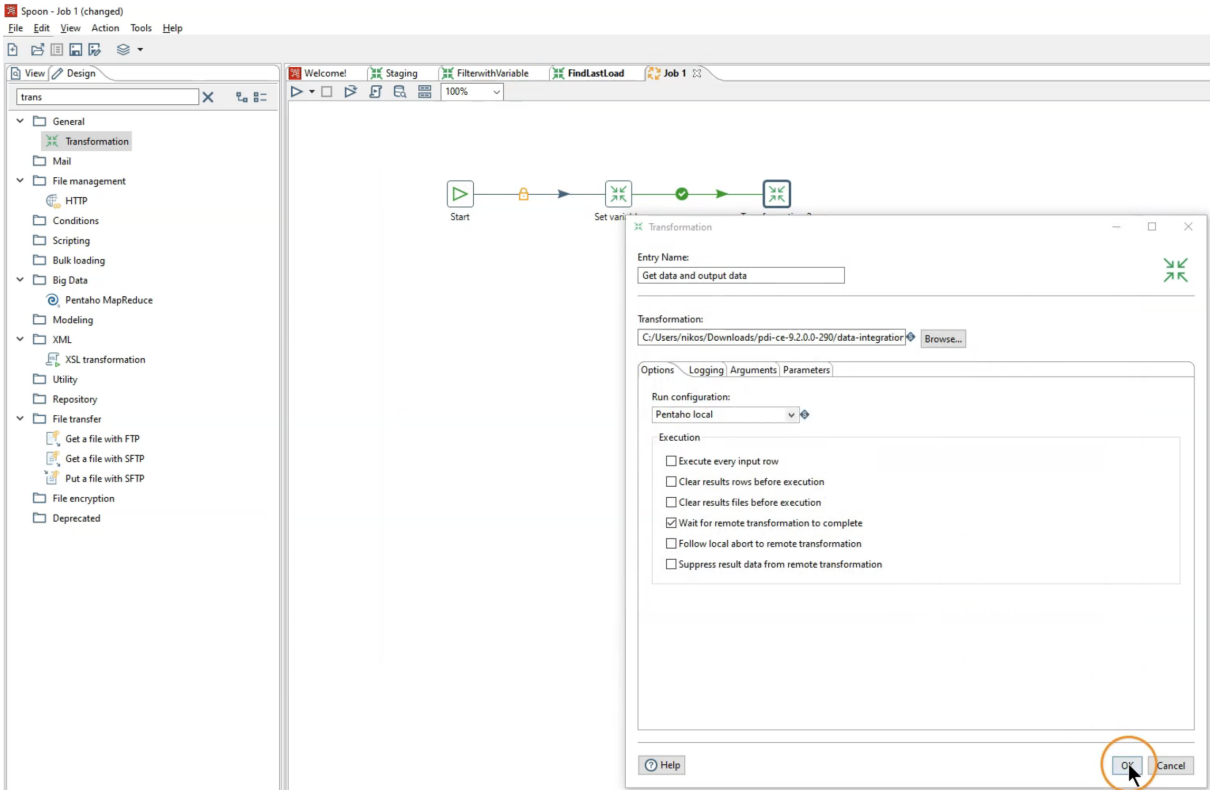
- But we don’t need the maximum column , so we can select and remove it





- Now create a new job. Go to File>Create>Job
- Of course we need to have start in the beginning and two transformations, one to get the loast load id variable and then use it to filter and load the data to our staging layer
- Go to each transformation , double click on it and browse our transformation which we have saved and then provide an entry name

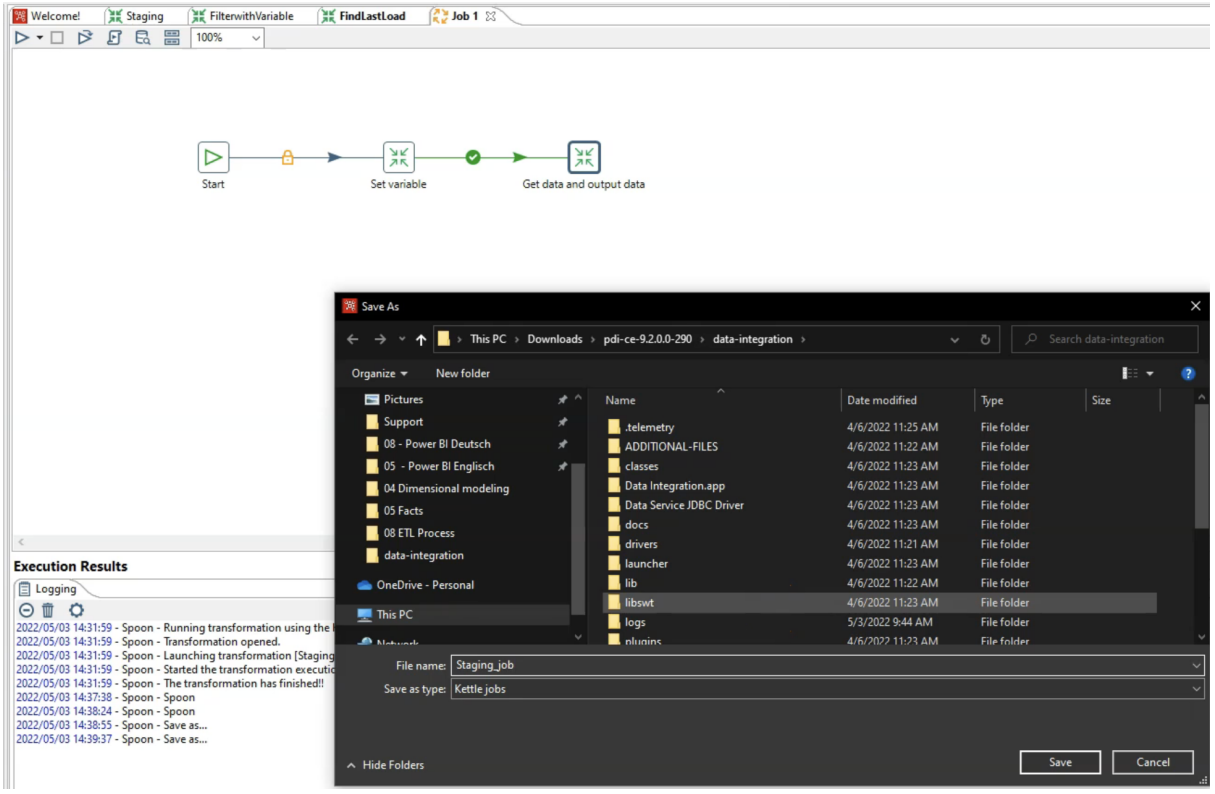


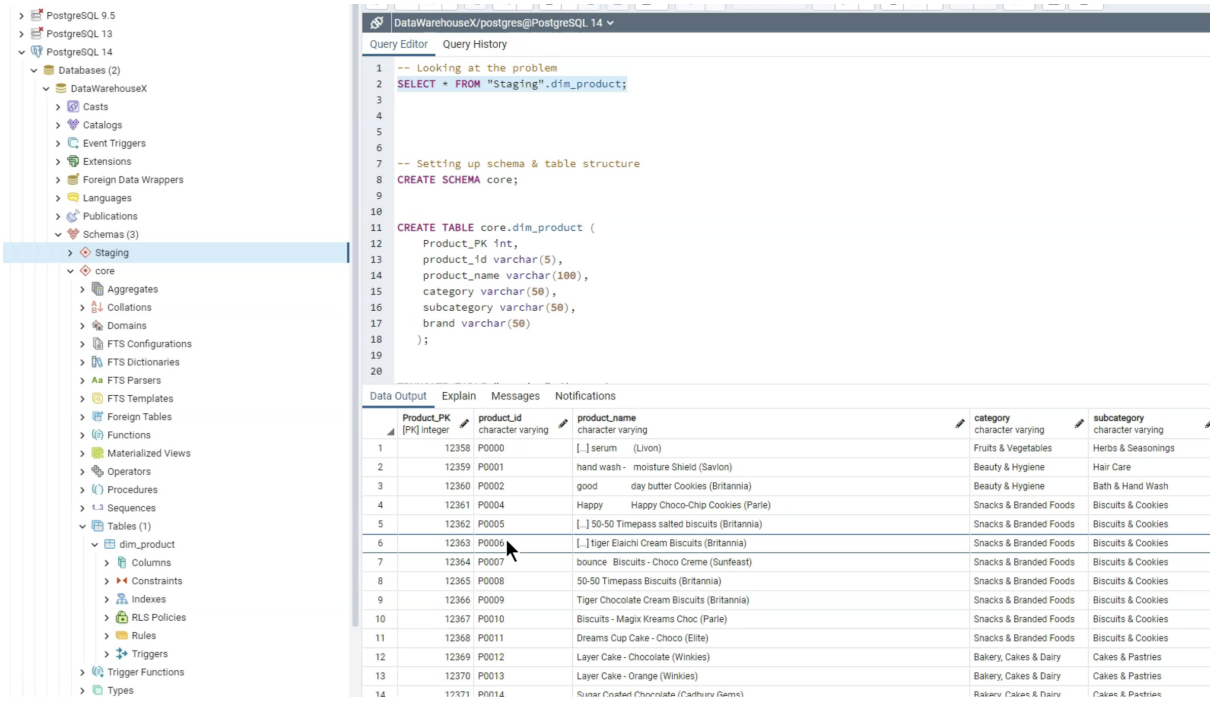


- Before running the Job, truncate the staging table using below command

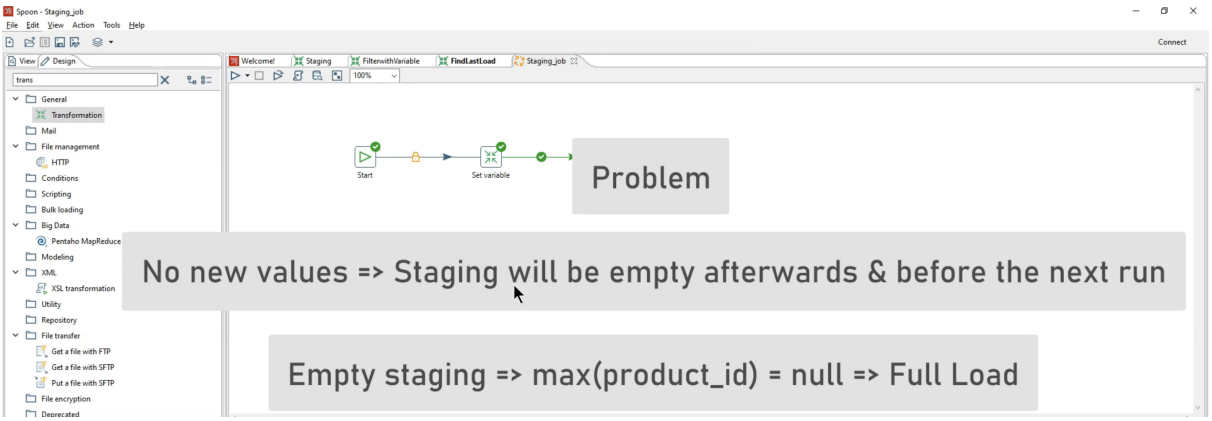
```
truncate table "Staging".dim_product;
```

- Now run the job, and make sure to save the Job before running and click on Play button

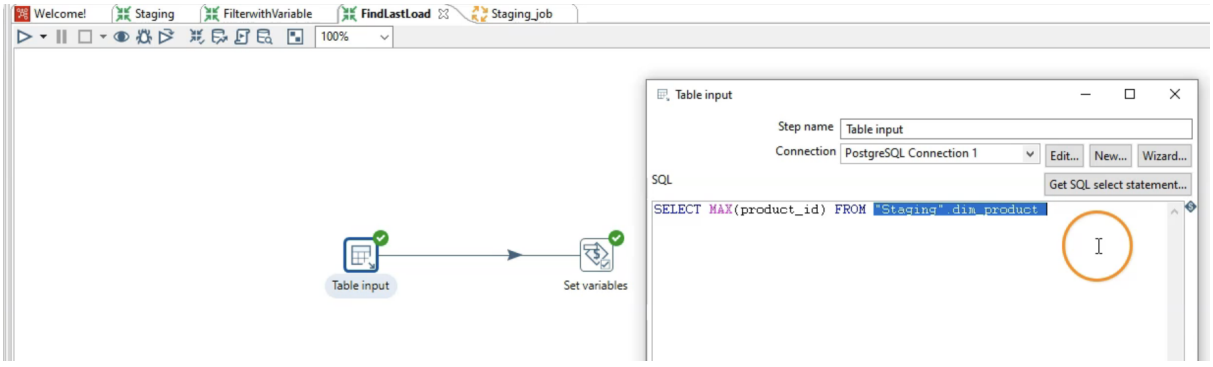




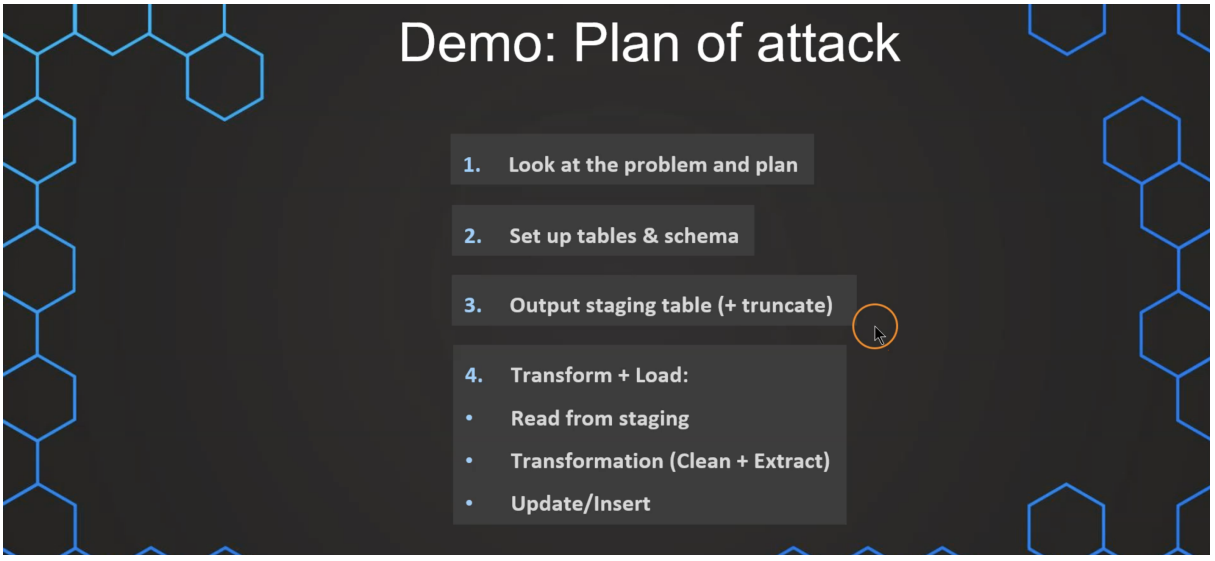
- But we have 1 problem with current setup



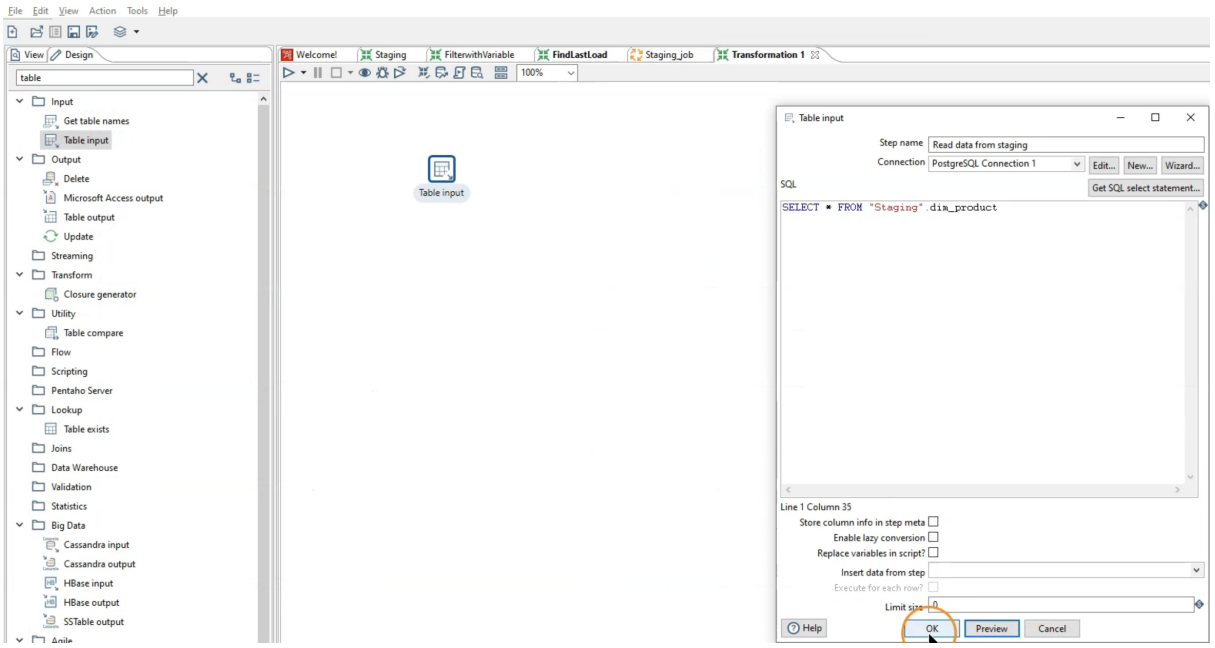
- We were getting max value from the previous load and using that to keep track of the max(product_id) but in next run if we do not have any new data then there will be no new max product_id as it will be set to null and then it will become full load
- So we should not be getting the max(product_id) from our staging layer but from our core layer



Demo: Transformation



- We want to do step 4, for this create a new transformation. Go to File>New>Transformation

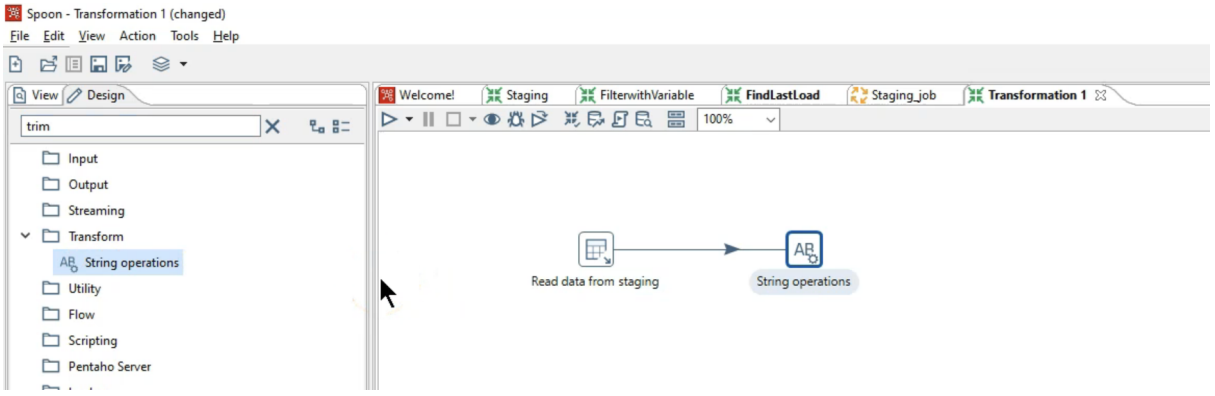


- Now we have set up reading the data from Staging tables, next step is to clean the data as we have bad data from the staging table

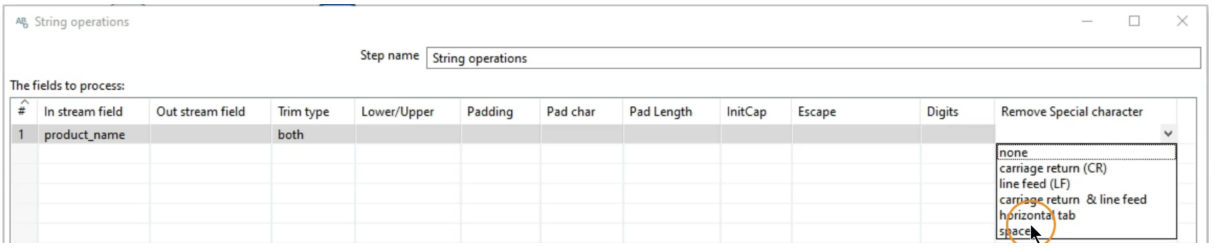
Data Output | Explain | Messages | Notifications

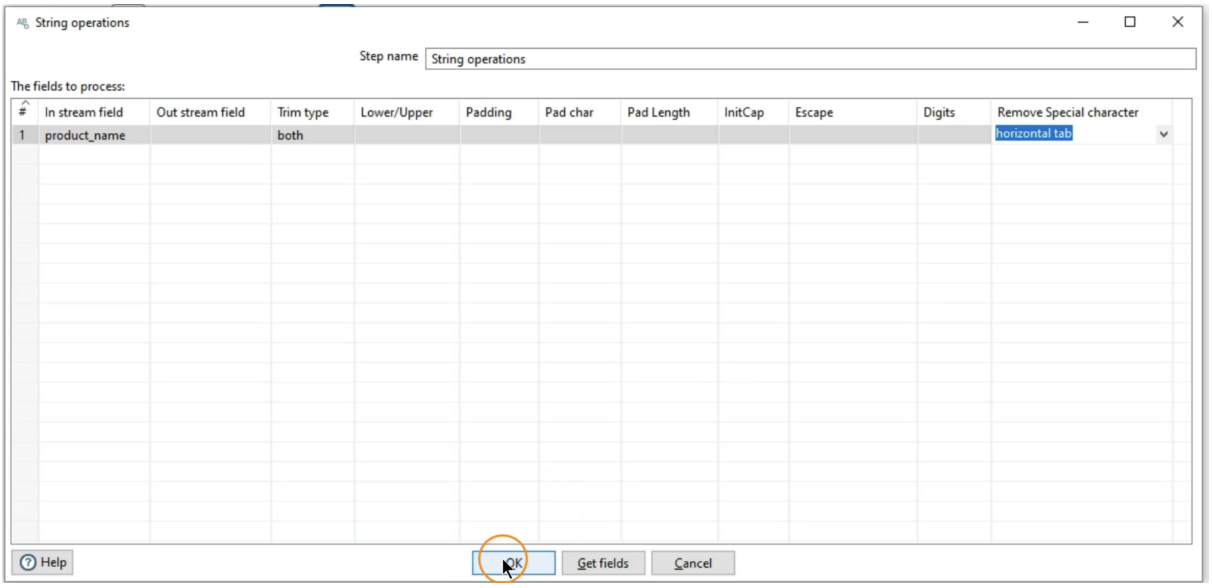
	Product_PK [PK] integer	product_id character varying	product_name character varying	category character varying	subcategory character varying
1	12358	P0000	[...] serum (Livon)	Fruits & Vegetables	Herbs & Seasonings
2	12359	P0001	hand wash - moisture Shield (Savlon)	Beauty & Hygiene	Hair Care
3	12360	P0002	good day butter Cookies (Britannia)	Beauty & Hygiene	Bath & Hand Wash
4	12361	P0004	Happy Happy Choco-Chip Cookies (Parle)	Snacks & Branded Foods	Biscuits & Cookies
5	12362	P0005	[...] 50-50 Timepass salted biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
6	12363	P0006	[...] tiger Elaichi Cream Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
7	12364	P0007	bounce Biscuits - Choco Creme (Sunfeast)	Snacks & Branded Foods	Biscuits & Cookies
8	12365	P0008	50-50 Timepass Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
9	12366	P0009	Tiger Chocolate Cream Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
10	12367	P0010	Biscuits - Magix Kreams Choc (Parle)	Snacks & Branded Foods	Biscuits & Cookies
11	12368	P0011	Dreams Cup Cake - Choco (Elite)	Snacks & Branded Foods	Biscuits & Cookies
12	12369	P0012	Layer Cake - Chocolate (Winkies)	Bakery, Cakes & Dairy	Cakes & Pastries
13	12370	P0013	Layer Cake - Orange (Winkies)	Bakery, Cakes & Dairy	Cakes & Pastries

- We want to remove unwanted characters, so search for string operations to do trimming

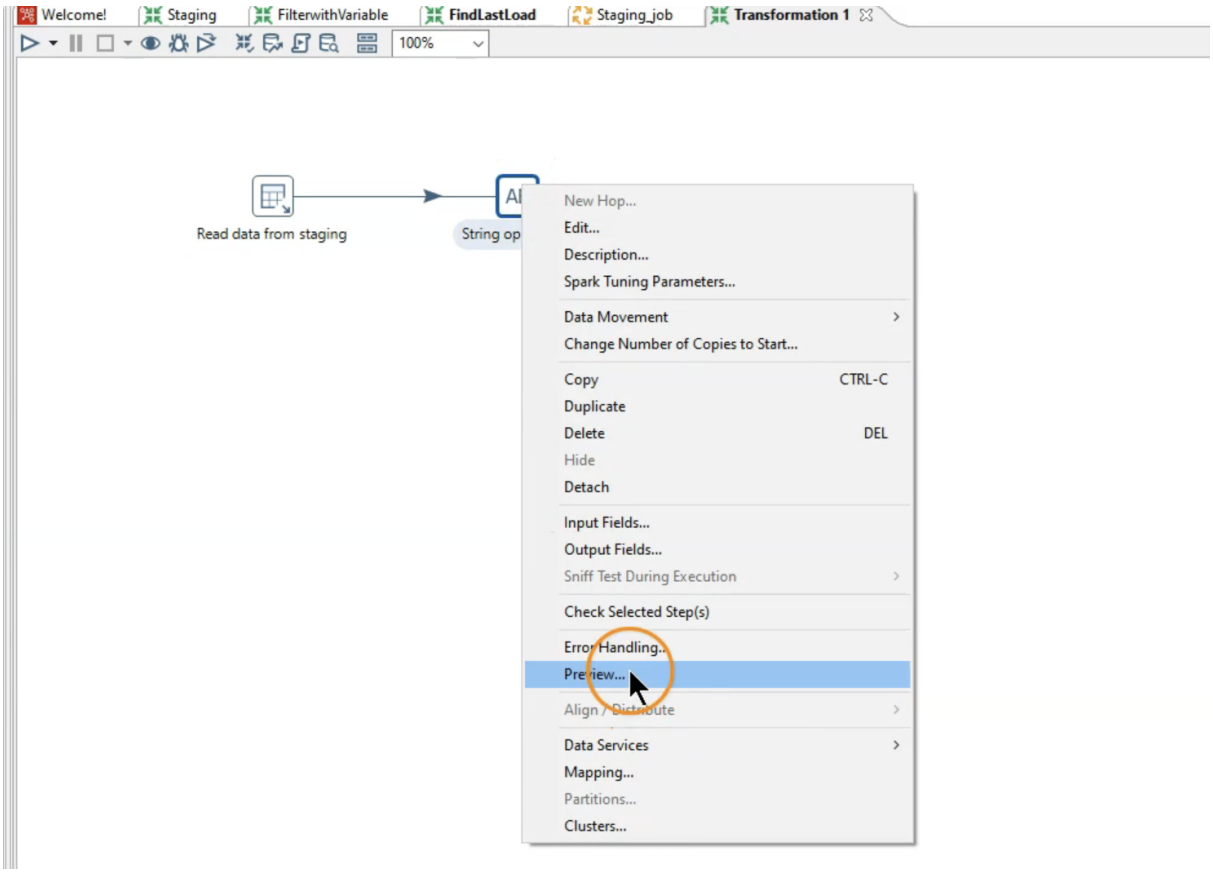


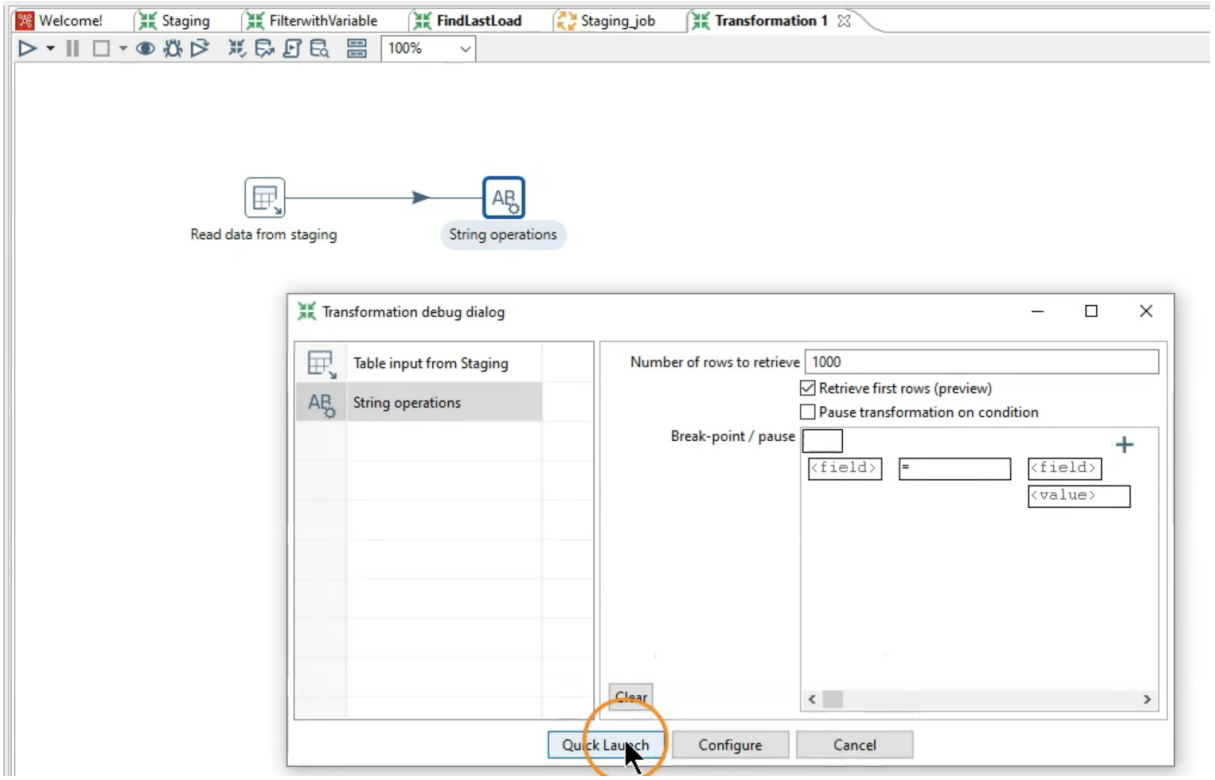
- Default trim (set to both) removes white spaces from beginning and end
- But in our data we also have Tab character in between our data, in spoon we have option to select a special character to remove

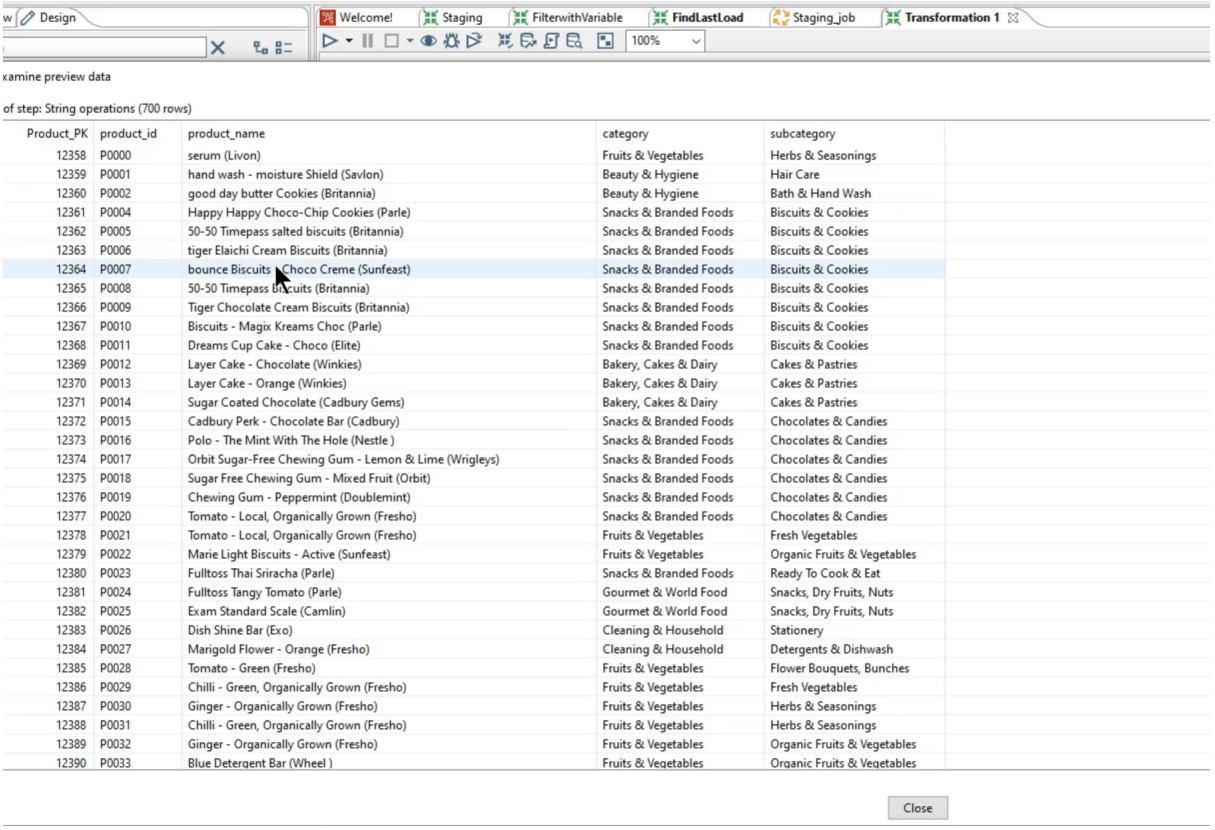




- We can preview the transformation with below steps

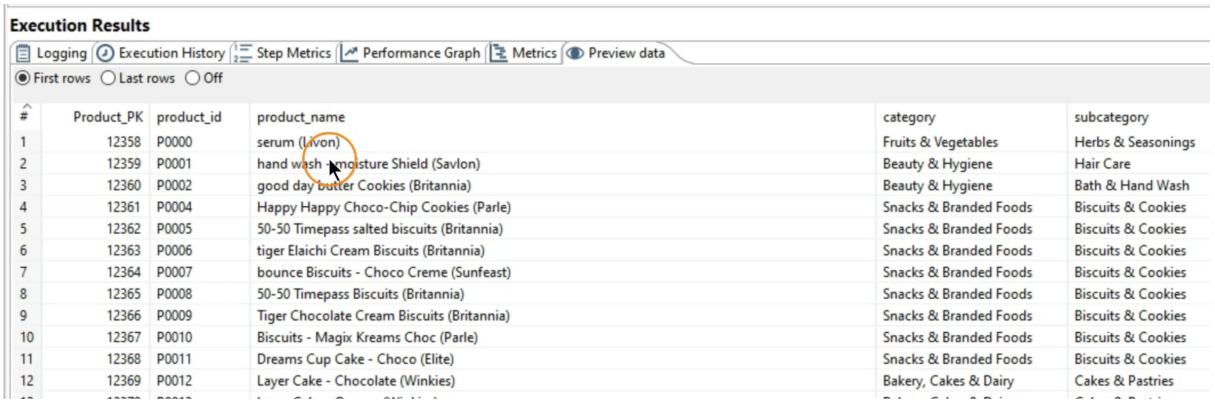






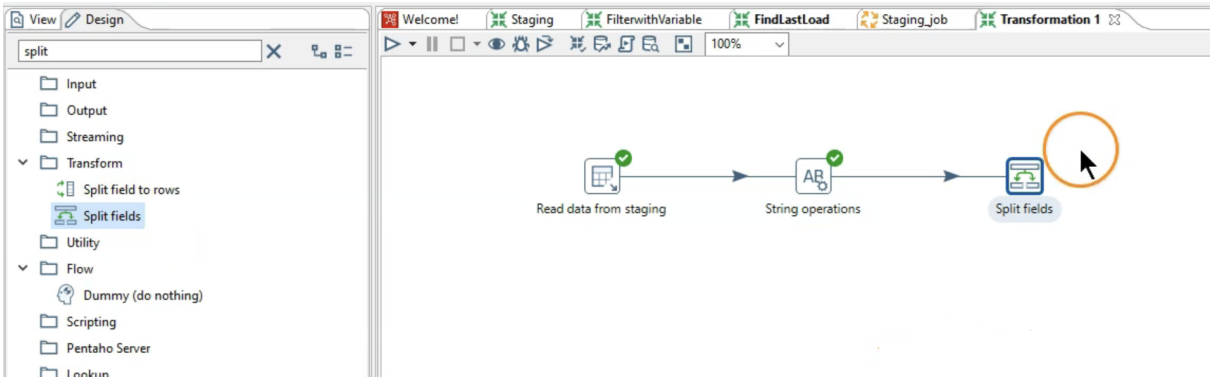
Product_PK	product_id	product_name	category	subcategory
12358	P0000	serum (Livon)	Fruits & Vegetables	Herbs & Seasonings
12359	P0001	hand wash - moisture Shield (Savlon)	Beauty & Hygiene	Hair Care
12360	P0002	good day butter Cookies (Britannia)	Beauty & Hygiene	Bath & Hand Wash
12361	P0004	Happy Happy Choco-Chip Cookies (Parle)	Snacks & Branded Foods	Biscuits & Cookies
12362	P0005	50-50 Timepass salted biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
12363	P0006	tiger Elaichi Cream Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
12364	P0007	bounce Biscuits - Choco Creme (Sunfeast)	Snacks & Branded Foods	Biscuits & Cookies
12365	P0008	50-50 Timepass Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
12366	P0009	Tiger Chocolate Cream Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
12367	P0010	Biscuits - Magix Kreams Choc (Parle)	Snacks & Branded Foods	Biscuits & Cookies
12368	P0011	Dreams Cup Cake - Choco (Elite)	Snacks & Branded Foods	Biscuits & Cookies
12369	P0012	Layer Cake - Chocolate (Winkies)	Bakery, Cakes & Dairy	Cakes & Pastries
12370	P0013	Layer Cake - Orange (Winkies)	Bakery, Cakes & Dairy	Cakes & Pastries
12371	P0014	Sugar Coated Chocolate (Cadbury Gems)	Bakery, Cakes & Dairy	Cakes & Pastries
12372	P0015	Cadbury Perk - Chocolate Bar (Cadbury)	Snacks & Branded Foods	Chocolates & Candies
12373	P0016	Polo - The Mint With The Hole (Nestle)	Snacks & Branded Foods	Chocolates & Candies
12374	P0017	Orbit Sugar-Free Chewing Gum - Lemon & Lime (Wrigleys)	Snacks & Branded Foods	Chocolates & Candies
12375	P0018	Sugar Free Chewing Gum - Mixed Fruit (Orbit)	Snacks & Branded Foods	Chocolates & Candies
12376	P0019	Chewing Gum - Peppermint (Doublemint)	Snacks & Branded Foods	Chocolates & Candies
12377	P0020	Tomato - Local, Organically Grown (Fresho)	Snacks & Branded Foods	Chocolates & Candies
12378	P0021	Tomato - Local, Organically Grown (Fresho)	Fruits & Vegetables	Fresh Vegetables
12379	P0022	Marie Light Biscuits - Active (Sunfeast)	Fruits & Vegetables	Organic Fruits & Vegetables
12380	P0023	Fulltoss Thai Sriracha (Parle)	Snacks & Branded Foods	Ready To Cook & Eat
12381	P0024	Fulltoss Tangy Tomato (Parle)	Gourmet & World Food	Snacks, Dry Fruits, Nuts
12382	P0025	Exam Standard Scale (Camlin)	Gourmet & World Food	Snacks, Dry Fruits, Nuts
12383	P0026	Dish Shine Bar (Exo)	Cleaning & Household	Stationery
12384	P0027	Marigold Flower - Orange (Fresho)	Cleaning & Household	Detergents & Dishwash
12385	P0028	Tomato - Green (Fresho)	Fruits & Vegetables	Flower Bouquets, Bunches
12386	P0029	Chilli - Green, Organically Grown (Fresho)	Fruits & Vegetables	Fresh Vegetables
12387	P0030	Ginger - Organically Grown (Fresho)	Fruits & Vegetables	Herbs & Seasonings
12388	P0031	Chilli - Green, Organically Grown (Fresho)	Fruits & Vegetables	Herbs & Seasonings
12389	P0032	Ginger - Organically Grown (Fresho)	Fruits & Vegetables	Organic Fruits & Vegetables
12390	P0033	Blue Detergent Bar (Wheel)	Fruits & Vegetables	Organic Fruits & Vegetables

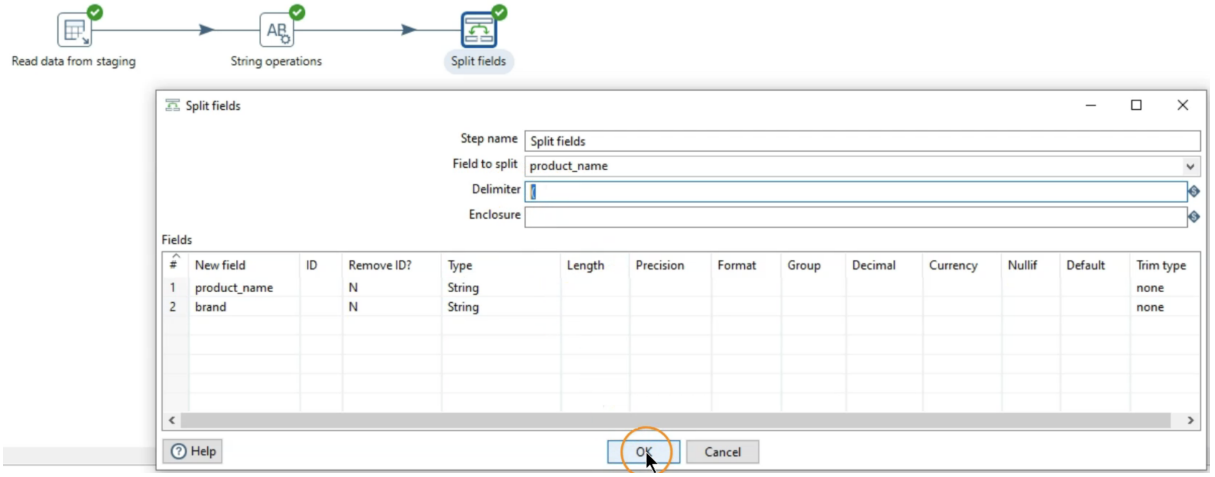
- This data looks more cleaned coming from staging layer
- Next step is to extract the brand value present within paranthesis



#	Product_PK	product_id	product_name	category	subcategory
1	12358	P0000	serum (Lavon)	Fruits & Vegetables	Herbs & Seasonings
2	12359	P0001	hand wash - Moisture Shield (Savlon)	Beauty & Hygiene	Hair Care
3	12360	P0002	good day butter Cookies (Britannia)	Beauty & Hygiene	Bath & Hand Wash
4	12361	P0004	Happy Happy Choco-Chip Cookies (Parle)	Snacks & Branded Foods	Biscuits & Cookies
5	12362	P0005	50-50 Timepass salted biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
6	12363	P0006	tiger Elaichi Cream Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
7	12364	P0007	bounce Biscuits - Choco Creme (Sunfeast)	Snacks & Branded Foods	Biscuits & Cookies
8	12365	P0008	50-50 Timepass Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
9	12366	P0009	Tiger Chocolate Cream Biscuits (Britannia)	Snacks & Branded Foods	Biscuits & Cookies
10	12367	P0010	Biscuits - Magix Kreams Choc (Parle)	Snacks & Branded Foods	Biscuits & Cookies
11	12368	P0011	Dreams Cup Cake - Choco (Elite)	Snacks & Branded Foods	Biscuits & Cookies
12	12369	P0012	Layer Cake - Chocolate (Winkies)	Bakery, Cakes & Dairy	Cakes & Pastries

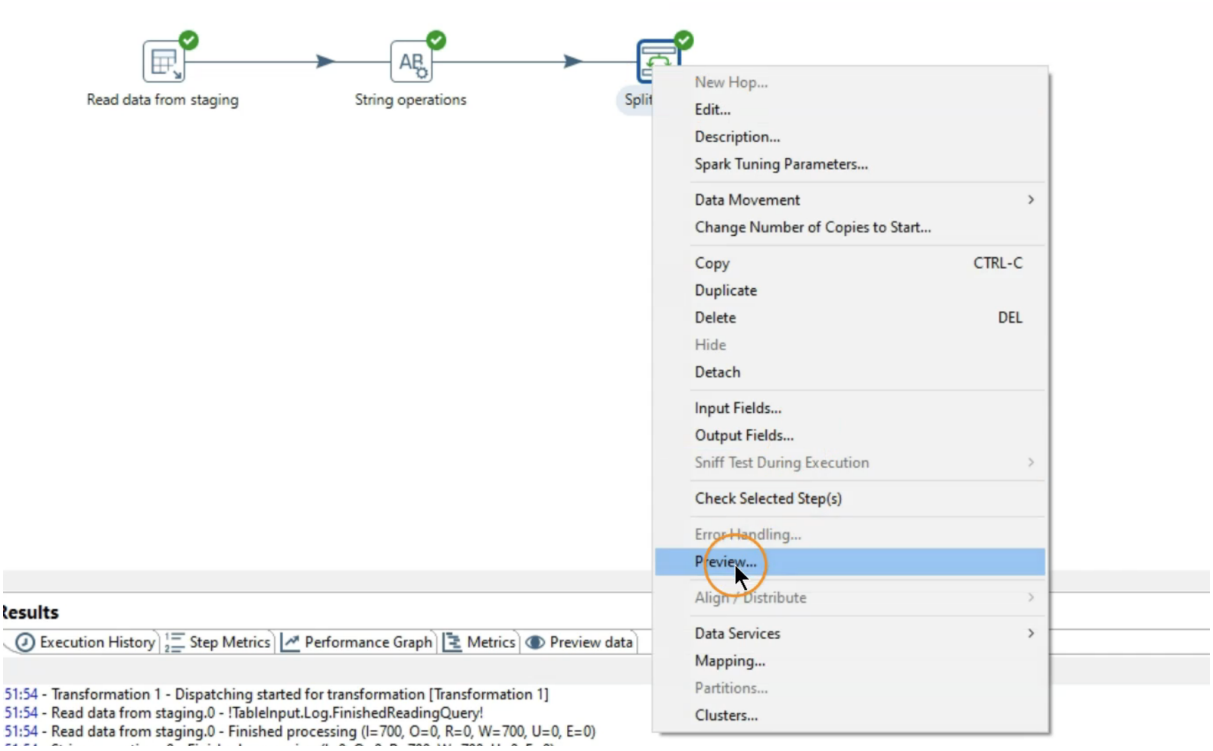
- To extract the brand value, search for split transformation

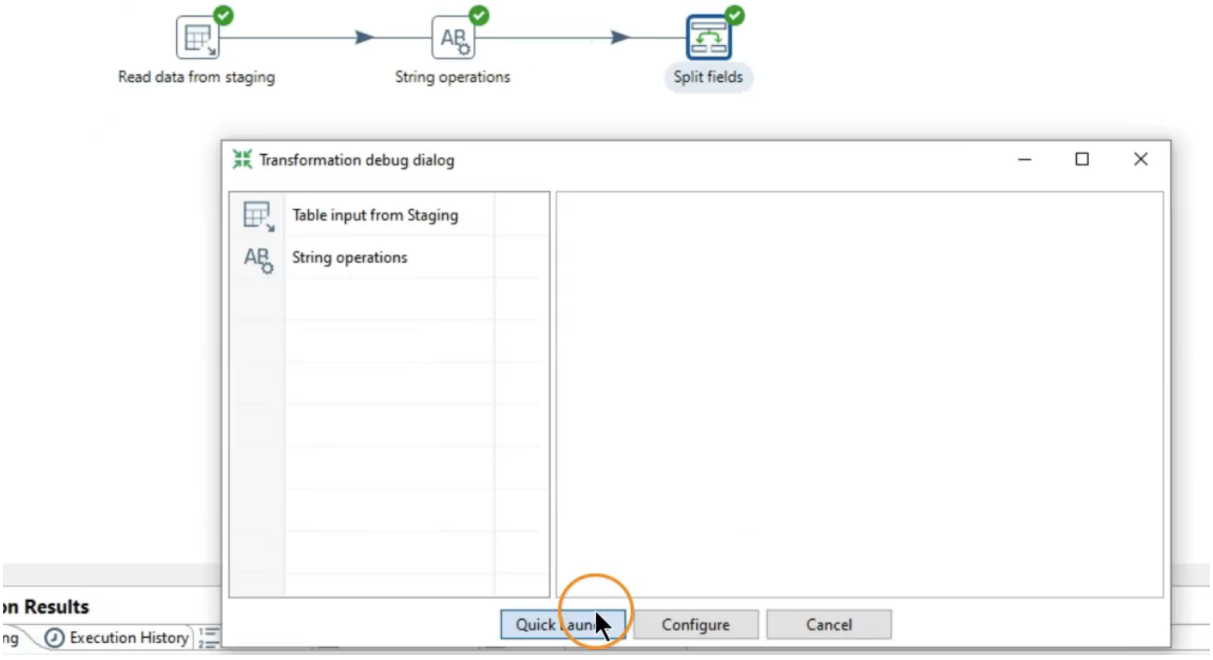


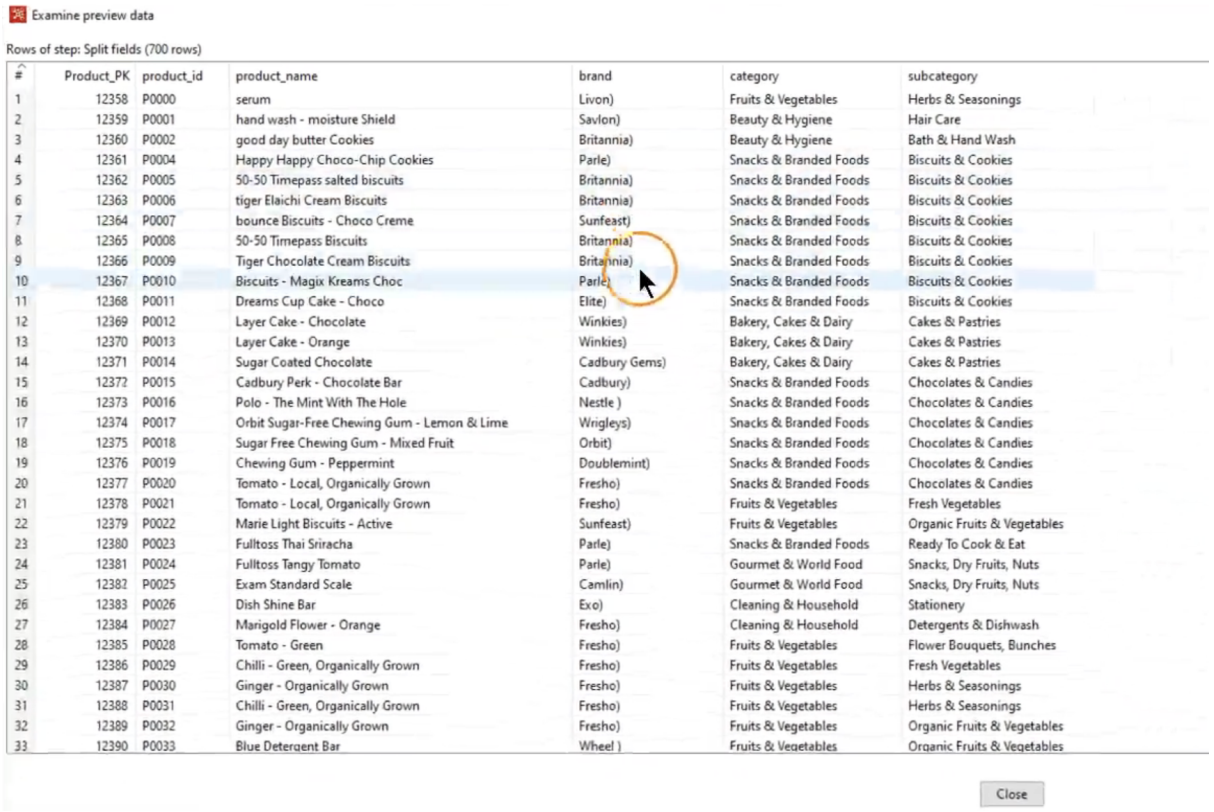


#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal	Currency	Nullif	Default	Trim type
1	product_name		N	String									none
2	brand		N	String									none

- Set the delimiter as (and the two columns the values to go into after split
- Now preview the data

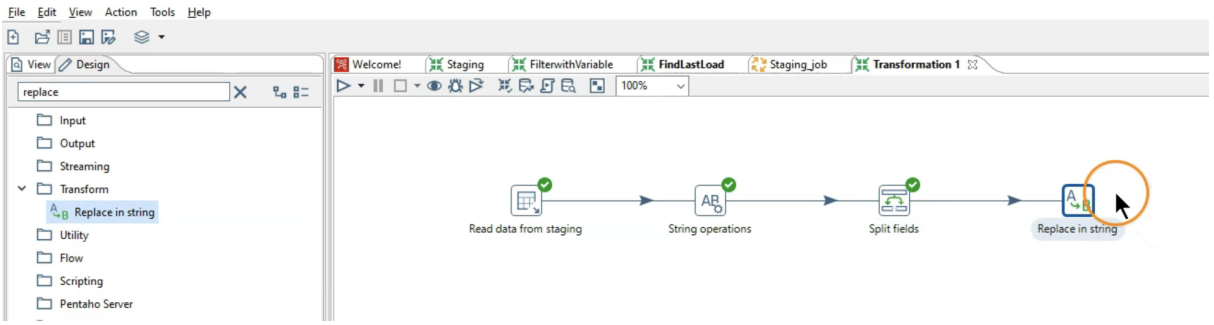

Read data from staging
String operations
New Hop...
Edit...
Description...
Spark Tuning Parameters...
Data Movement
Change Number of Copies to Start...
Copy CTRL-C
Duplicate
Delete DEL
Hide
Detach
Input Fields...
Output Fields...
Sniff Test During Execution
Check Selected Step(s)
Error Handling...
Preview...
Align / Distribute
Data Services
Mapping...
Partitions...
Clusters...
Execution History
Step Metrics
Performance Graph
Metrics
Preview data
51:54 - Transformation 1 - Dispatching started for transformation [Transformation 1]
51:54 - Read data from staging.0 - !TableInput.Log.FinishedReadingQuery!
51:54 - Read data from staging.0 - Finished processing (I=700, O=0, R=0, W=700, U=0, E=0)

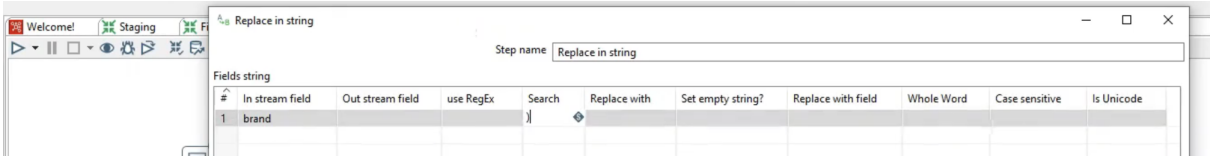

Read data from staging
String operations
Split fields
Transformation debug dialog
Table input from Staging
String operations
Quick Launch
Configure
Cancel
Execution History



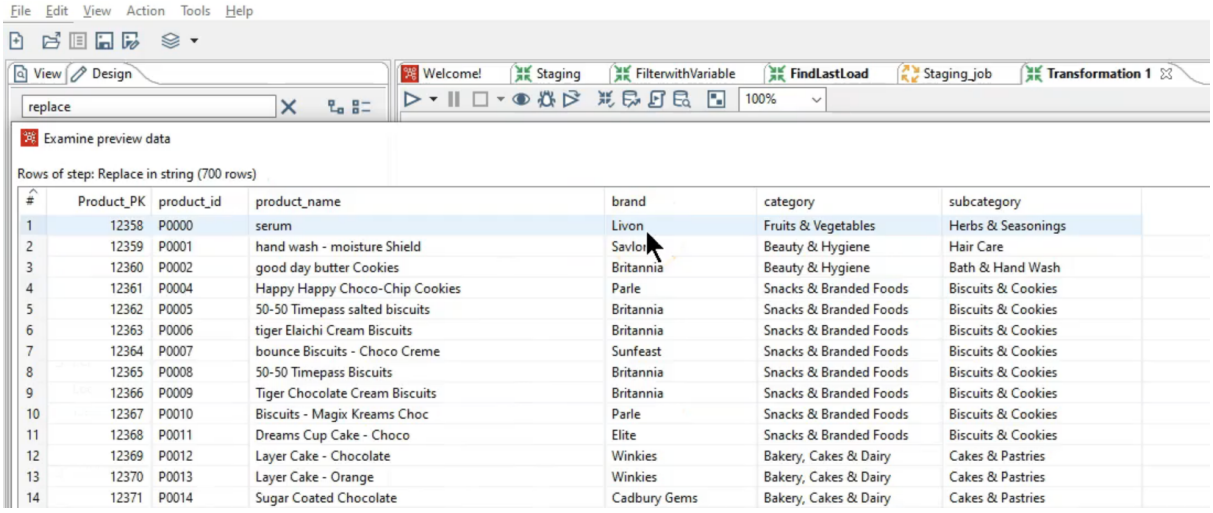
#	Product_PK	product_id	product_name	brand	category	subcategory
1	12358	P0000	serum	Livon)	Fruits & Vegetables	Herbs & Seasonings
2	12359	P0001	hand wash - moisture Shield	Savlon)	Beauty & Hygiene	Hair Care
3	12360	P0002	good day butter Cookies	Britannia)	Beauty & Hygiene	Bath & Hand Wash
4	12361	P0004	Happy Happy Choco-Chip Cookies	Parle)	Snacks & Branded Foods	Biscuits & Cookies
5	12362	P0005	50-50 Timepass salted biscuits	Britannia)	Snacks & Branded Foods	Biscuits & Cookies
6	12363	P0006	tiger Elaichi Cream Biscuits	Britannia)	Snacks & Branded Foods	Biscuits & Cookies
7	12364	P0007	bounce Biscuits - Choco Creme	Sunfeast)	Snacks & Branded Foods	Biscuits & Cookies
8	12365	P0008	50-50 Timepass Biscuits	Britannia)	Snacks & Branded Foods	Biscuits & Cookies
9	12366	P0009	Tiger Chocolate Cream Biscuits	Britannia)	Snacks & Branded Foods	Biscuits & Cookies
10	12367	P0010	Biscuits - Magix Kreams Choc	Parle)	Snacks & Branded Foods	Biscuits & Cookies
11	12368	P0011	Dreams Cup Cake - Choco	Elite)	Snacks & Branded Foods	Biscuits & Cookies
12	12369	P0012	Layer Cake - Chocolate	Winkies)	Bakery, Cakes & Dairy	Cakes & Pastries
13	12370	P0013	Layer Cake - Orange	Winkies)	Bakery, Cakes & Dairy	Cakes & Pastries
14	12371	P0014	Sugar Coated Chocolate	Cadbury Gems)	Bakery, Cakes & Dairy	Cakes & Pastries
15	12372	P0015	Cadbury Perk - Chocolate Bar	Cadbury)	Snacks & Branded Foods	Chocolates & Candies
16	12373	P0016	Polo - The Mint With The Hole	Nestle)	Snacks & Branded Foods	Chocolates & Candies
17	12374	P0017	Orbit Sugar-Free Chewing Gum - Lemon & Lime	Wrigleys)	Snacks & Branded Foods	Chocolates & Candies
18	12375	P0018	Sugar Free Chewing Gum - Mixed Fruit	Orbit)	Snacks & Branded Foods	Chocolates & Candies
19	12376	P0019	Chewing Gum - Peppermint	Doublemint)	Snacks & Branded Foods	Chocolates & Candies
20	12377	P0020	Tomato - Local, Organically Grown	Fresho)	Snacks & Branded Foods	Chocolates & Candies
21	12378	P0021	Tomato - Local, Organically Grown	Fresho)	Fruits & Vegetables	Fresh Vegetables
22	12379	P0022	Marie Light Biscuits - Active	Sunfeast)	Fruits & Vegetables	Organic Fruits & Vegetables
23	12380	P0023	Fulltoss Thai Sriracha	Parle)	Snacks & Branded Foods	Ready To Cook & Eat
24	12381	P0024	Fulltoss Tangy Tomato	Parle)	Gourmet & World Food	Snacks, Dry Fruits, Nuts
25	12382	P0025	Exam Standard Scale	Camlin)	Gourmet & World Food	Snacks, Dry Fruits, Nuts
26	12383	P0026	Dish Shine Bar	Exo)	Cleaning & Household	Stationery
27	12384	P0027	Marigold Flower - Orange	Fresho)	Cleaning & Household	Detergents & Dishwash
28	12385	P0028	Tomato - Green	Fresho)	Fruits & Vegetables	Flower Bouquets, Bunches
29	12386	P0029	Chilli - Green, Organically Grown	Fresho)	Fruits & Vegetables	Fresh Vegetables
30	12387	P0030	Ginger - Organically Grown	Fresho)	Fruits & Vegetables	Herbs & Seasonings
31	12388	P0031	Chilli - Green, Organically Grown	Fresho)	Fruits & Vegetables	Herbs & Seasonings
32	12389	P0032	Ginger - Organically Grown	Fresho)	Fruits & Vegetables	Organic Fruits & Vegetables
33	12390	P0033	Blue Detergent Bar	Wheel)	Fruits & Vegetables	Organic Fruits & Vegetables

- Now we can see the brand names coming in, but we have to remove the) character for that we can use the Replace in a string





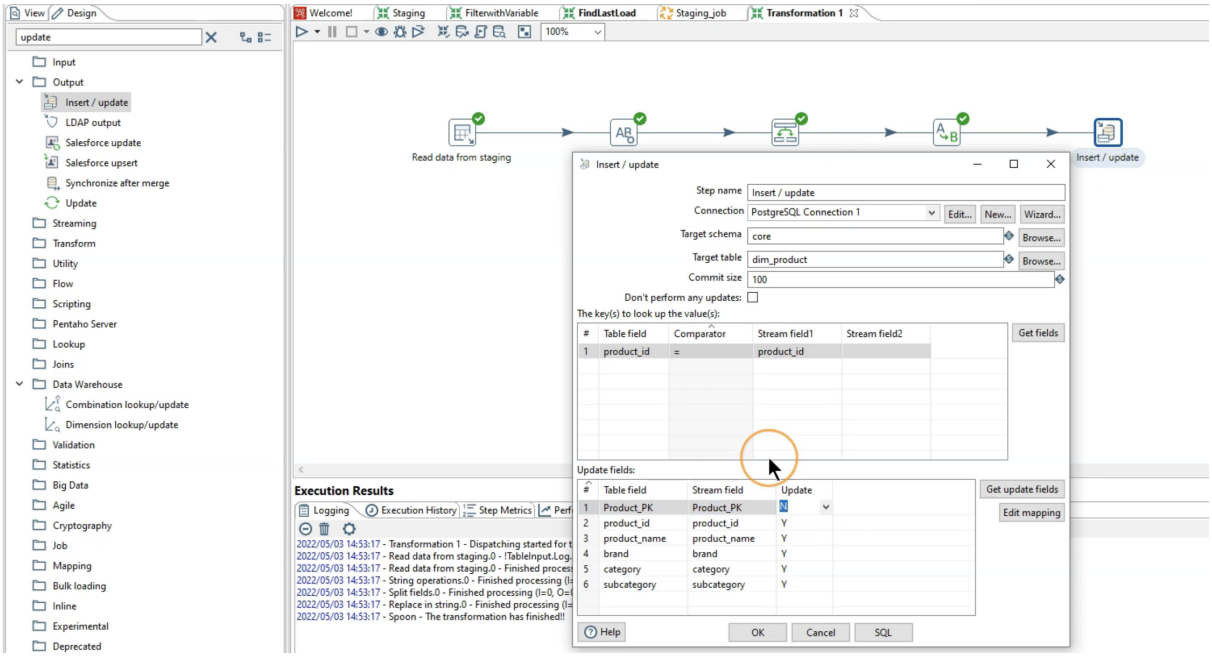
- Preview the data again



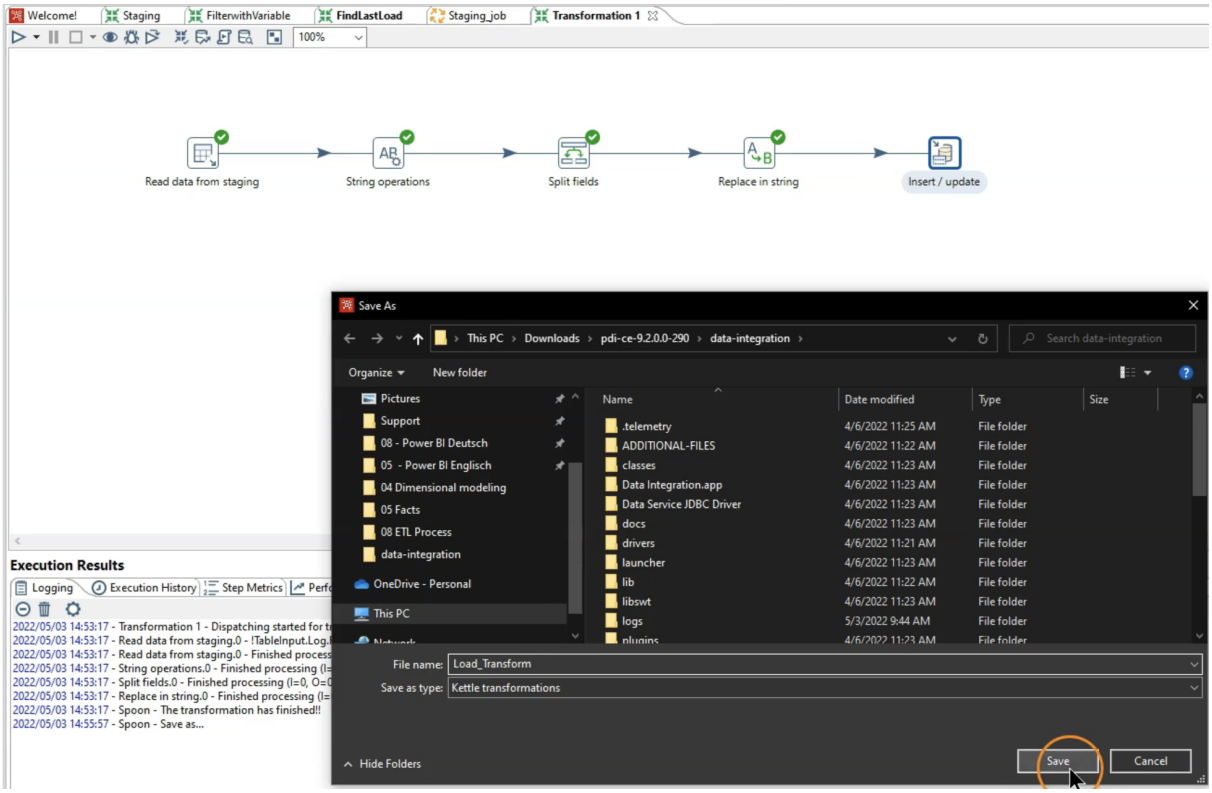
#	Product_PK	product_id	product_name	brand	category	subcategory
1	12358	P0000	serum	Livon	Fruits & Vegetables	Herbs & Seasonings
2	12359	P0001	hand wash - moisture Shield	Savlon	Beauty & Hygiene	Hair Care
3	12360	P0002	good day butter Cookies	Britannia	Beauty & Hygiene	Bath & Hand Wash
4	12361	P0004	Happy Happy Choco-Chip Cookies	Parle	Snacks & Branded Foods	Biscuits & Cookies
5	12362	P0005	50-50 Timepass salted biscuits	Britannia	Snacks & Branded Foods	Biscuits & Cookies
6	12363	P0006	tiger Elaichi Cream Biscuits	Britannia	Snacks & Branded Foods	Biscuits & Cookies
7	12364	P0007	bounce Biscuits - Choco Creme	Sunfeast	Snacks & Branded Foods	Biscuits & Cookies
8	12365	P0008	50-50 Timepass Biscuits	Britannia	Snacks & Branded Foods	Biscuits & Cookies
9	12366	P0009	Tiger Chocolate Cream Biscuits	Britannia	Snacks & Branded Foods	Biscuits & Cookies
10	12367	P0010	Biscuits - Magix Kreams Choc	Parle	Snacks & Branded Foods	Biscuits & Cookies
11	12368	P0011	Dreams Cup Cake - Choco	Elite	Snacks & Branded Foods	Biscuits & Cookies
12	12369	P0012	Layer Cake - Chocolate	Winkies	Bakery, Cakes & Dairy	Cakes & Pastries
13	12370	P0013	Layer Cake - Orange	Winkies	Bakery, Cakes & Dairy	Cakes & Pastries
14	12371	P0014	Sugar Coated Chocolate	Cadbury Gems	Bakery, Cakes & Dairy	Cakes & Pastries

Demo: Load and validate results

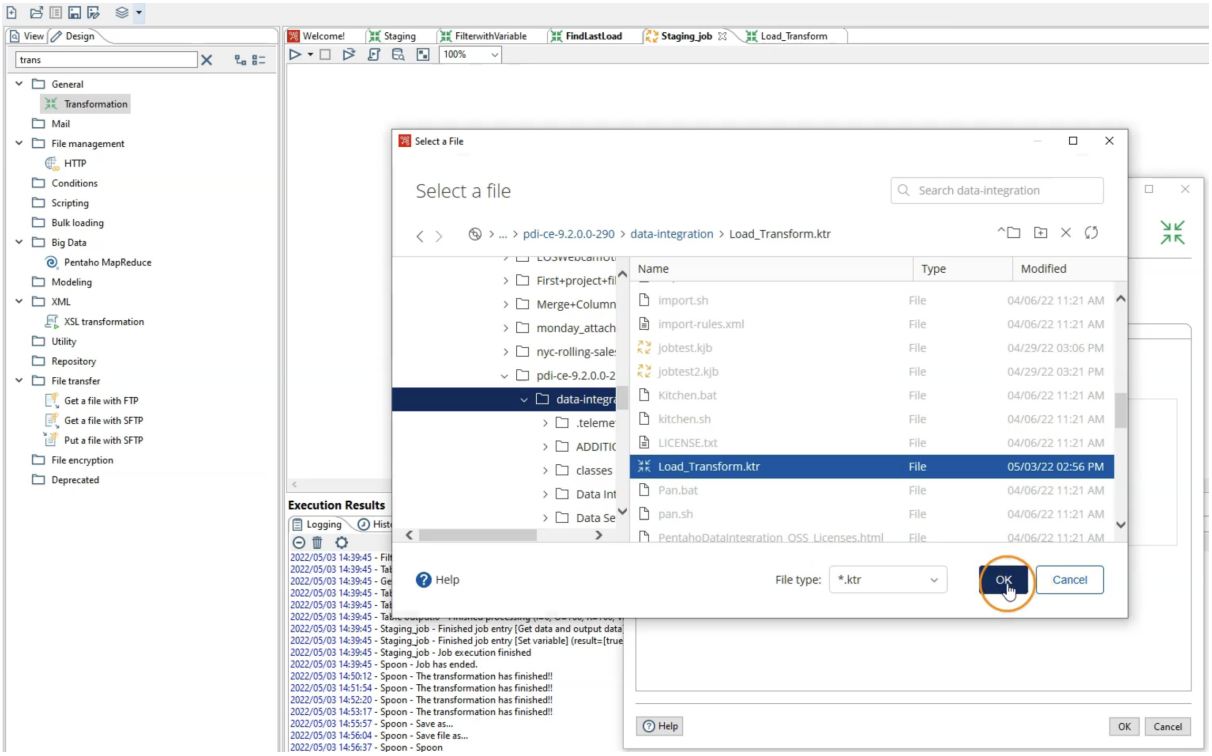
- We can setup insert/update, using the insert/update transformation
- We can specify the key column to identify if a record/row already exists and also specify if the row already exists what fileds should be updated

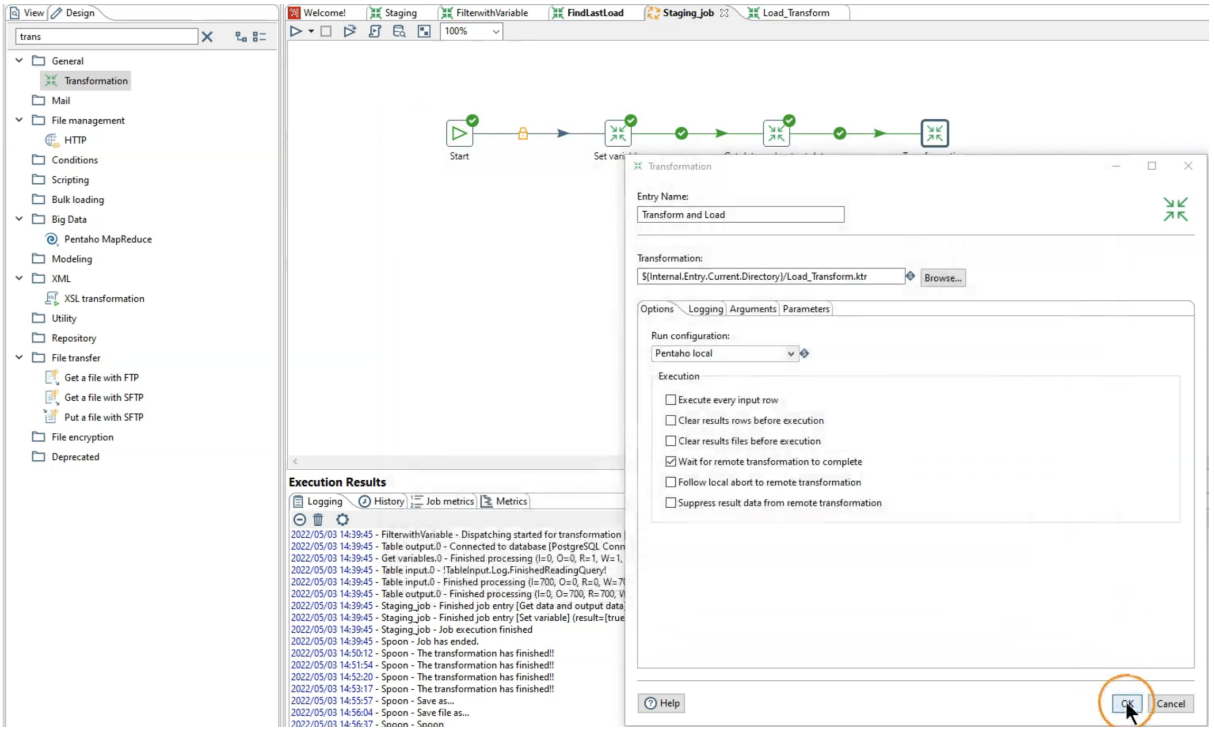


- Save this transformation

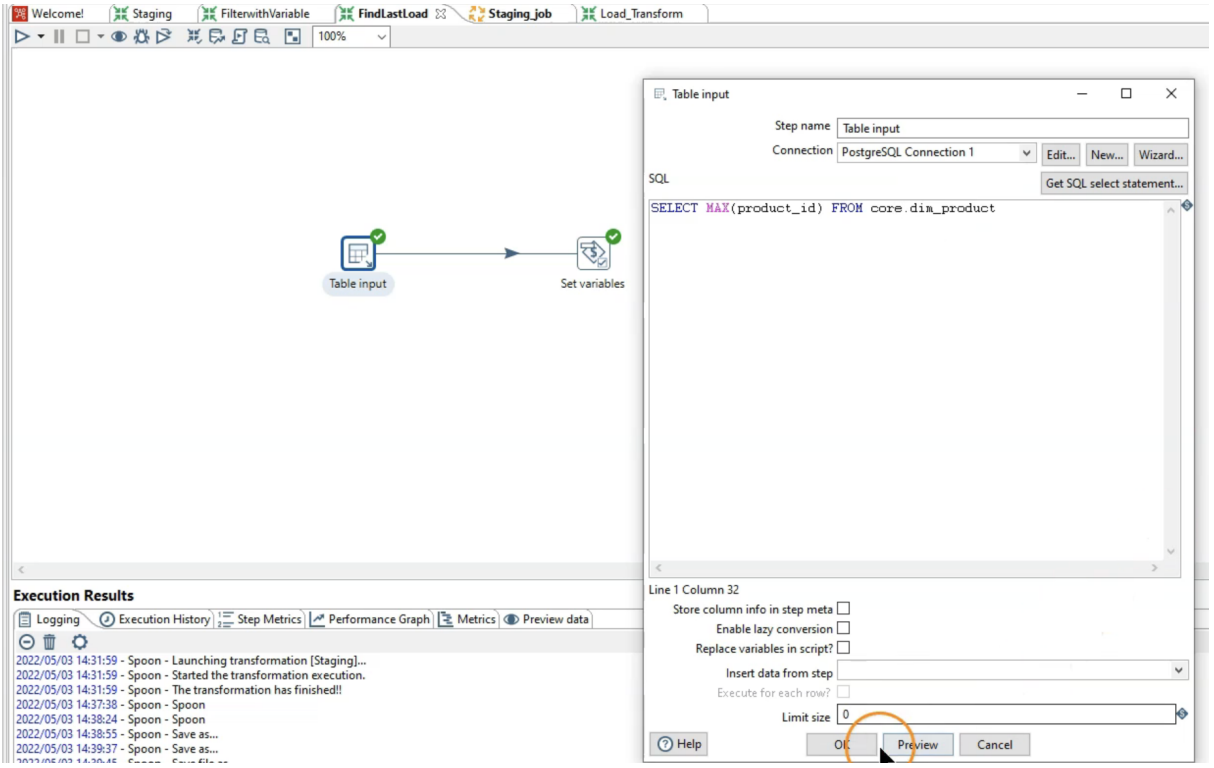


- Add load and transform step to the Job

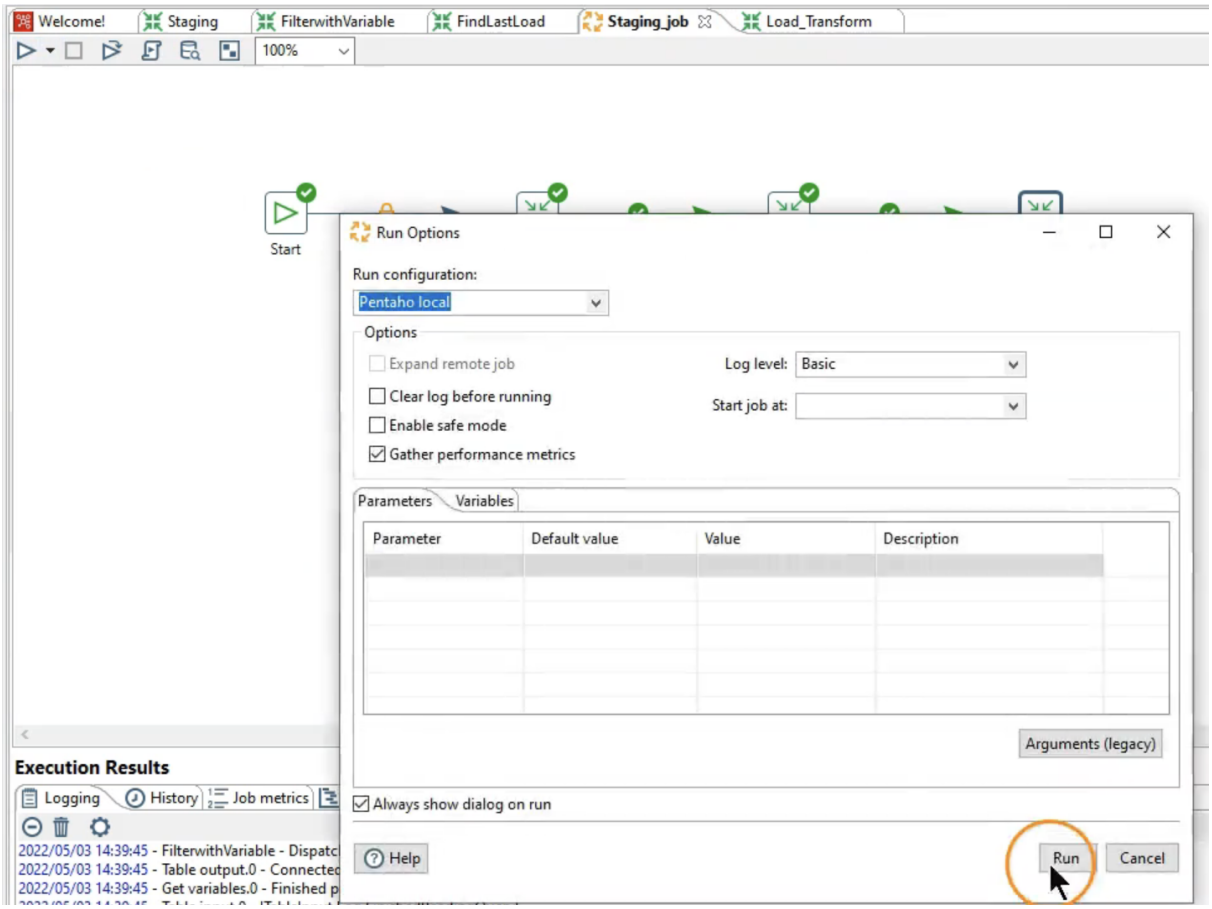




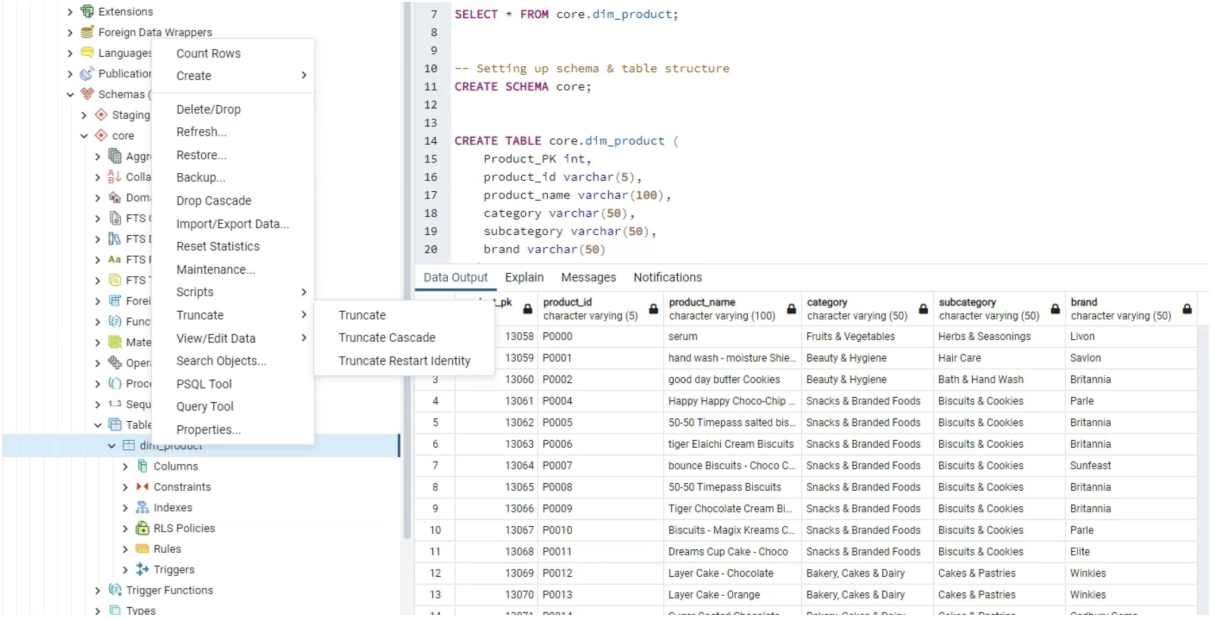
- As we identified the problem earlier we have to get the max(product_id) from the core layer instead of Staging layer



- We can now run the job and test it



- Now we have transformed data in the core layer



- We can alos Insert new record and run the job again to see the newly inserted record in staging table will flow into core layer with transofmration

```sql
INSERT INTO public.products(product_id, product_name, category, subcategory)
VALUES ('P0751', 'Red Apple (Milstaso)','Fruits & Vegetables','Fruits');

-- Looking at the problem
SELECT * FROM "Staging".dim_product;


SELECT * FROM core.dim_product;


 Setting up schema & table structure
EATE SCHEMA core;


EATE TABLE core.dim_product (
  Product_PK int,
  product_id varchar(5),
  product_name varchar(100),
  category varchar(50),
  subcategory varchar(50),
```

Data Output | Explain | Messages | Notifications

	product_pk integer	product_id character varying (5)	product_name character varying (100)	category character varying (50)	subcategory character varying (50)	brand character varying (50)
688	13745	P0736	Organic Seeds - Black Mus...	Snacks & Branded Foods	Chocolates & Candies	Organic Tattva
689	13746	P0737	Organic Seeds - Brown Mu...	Foodgrains, Oil & Masala	Organic Staples	Organic Tattva
690	13747	P0738	Fevicol MR Angular Tip	Foodgrains, Oil & Masala	Organic Staples	Pidilite
691	13748	P0739	Haldi Milk	Cleaning & Household	Stationery	mother dairy
692	13749	P0740	Sterilised Double Toned Fl...	Bakery, Cakes & Dairy	Dairy	THIRUMALA
693	13750	P0741	Dark Fantasy - Choco Fills ...	Bakery, Cakes & Dairy	Dairy	Sunfeast
694	13751	P0742	Coke Zero Soft Drink - No ...	Snacks & Branded Foods	Biscuits & Cookies	Coca-Cola
695	13752	P0744	Bleaching Powder - Strong	Beverages	Energy & Soft Drinks	Zermisol
696	13753	P0745	Cornflour	Cleaning & Household	All Purpose Cleaners	Weikfield
697	13754	P0746	Haldi Chandan Soap	Foodgrains, Oil & Masala	Atta, Flours & Sooji	Jiva Ayurveda
698	13755	P0747	Pure Magic Deuce Vanilla ...	Beauty & Hygiene	Bath & Hand Wash	Britannia
699	13756	P0748	[null]	Snacks & Branded Foods	Biscuits & Cookies	[null]
700	13757	P0750	test	testcategory	test	[null]
701	13758	P0751	Red Apple	Fruits & Vegetables	Fruits	Milstaso