SAS Data Quality Monitoring workshop exercises

Paul Van Mol

Contents

1. Read Breakfast_Items and Manufacturers with an import

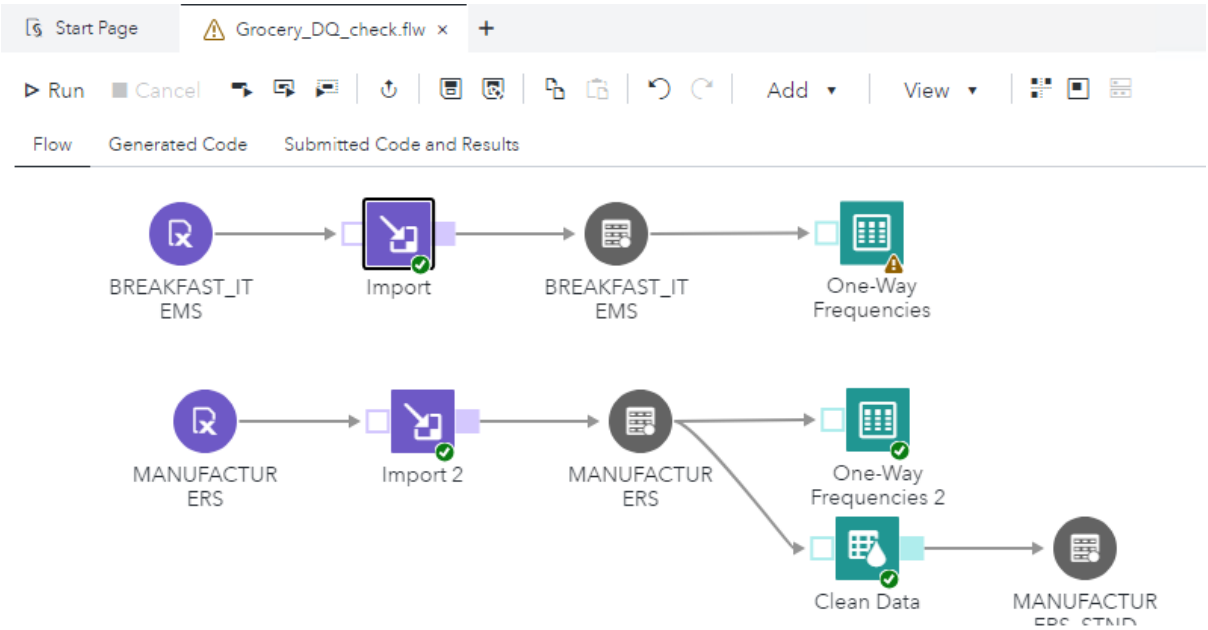
Navigate to the Folder: /gelcontent/dqdashboard/data/

Add the BREAKFAST_ITEMS.xlsx, then add an import step.

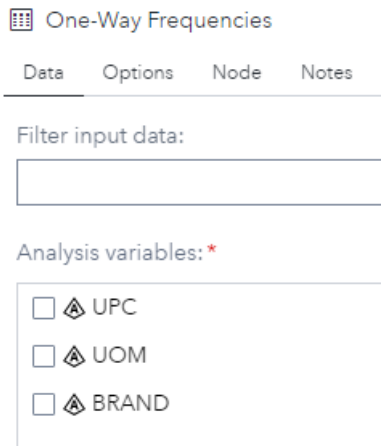
Analyze the columns and store the result in a table SASDM.BREAKFAST_ITEMS

Add the MANUFACTUERS.xlsx, then add an import step.

Analyze the columns and store the result in a table SASDM.MANUFACTURERS

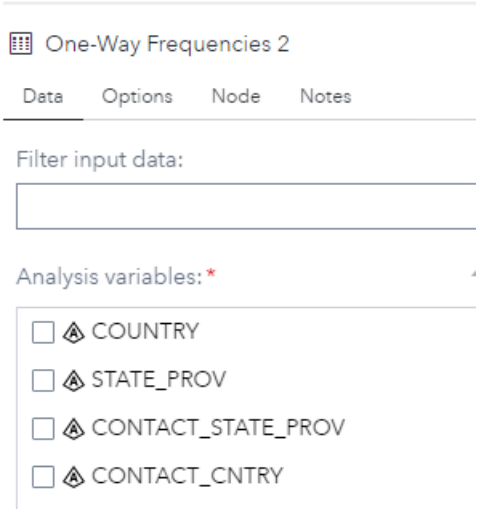


In order to check the values in BREAKFAST_ITEMS: UPC, UOM and BRAND, add a One-Way Frequencies Step to the Breakfast_Items table:



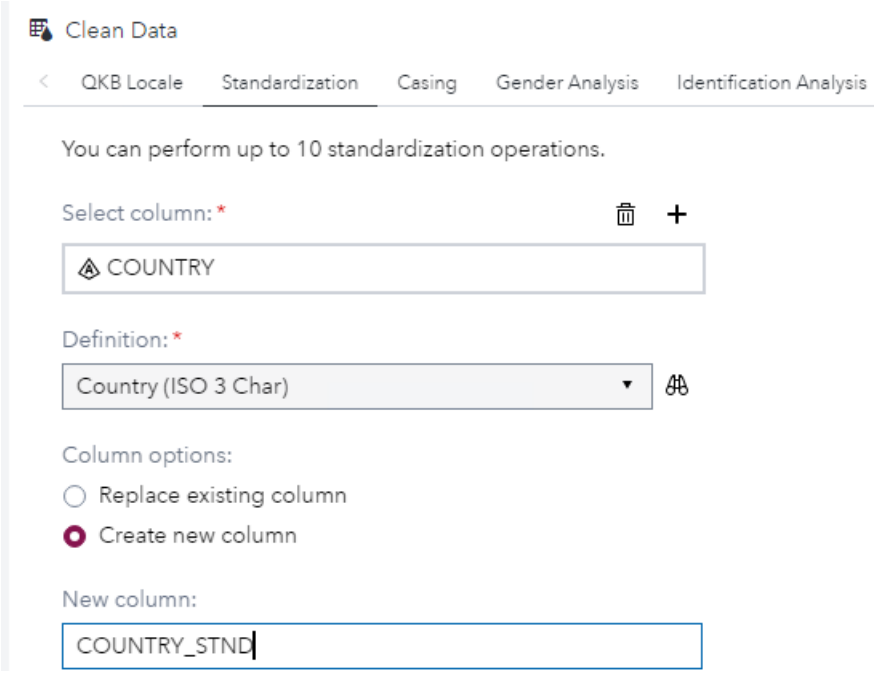
In options unselect include cumulative frequencies and percentages:

In order to check the values in MANUFACTURERS fields: COUNTRY, STATE_PROV, CONTACT_CNTRY, CONTACT_STATE_PROV add a One-Way Frequencies Step to the MANUFACTURERS table:



Finally, Standardize the COUNTRY and STATE_PROV fields in the MANUFACTURERS table:

Using a Clean Data step:

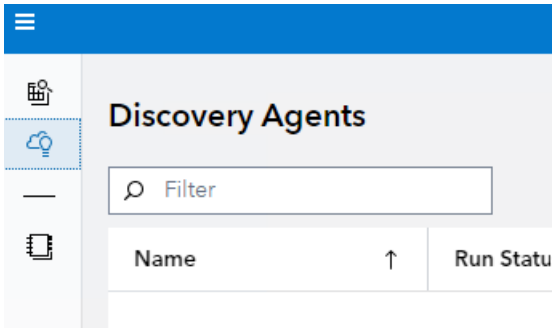


2. Add 2 discovery Agents as an SAS Viya Administrator for SASDM Compute and Public Caslib.

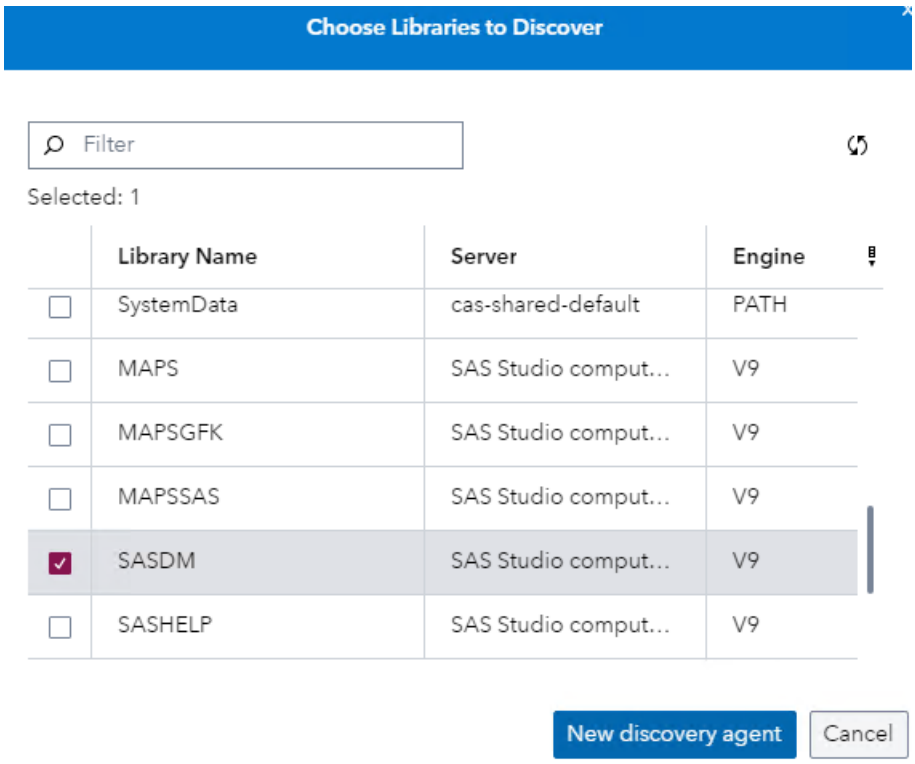
Login to SAS Viya as sasadm & lnxsas.

Go to Discover Information Assets

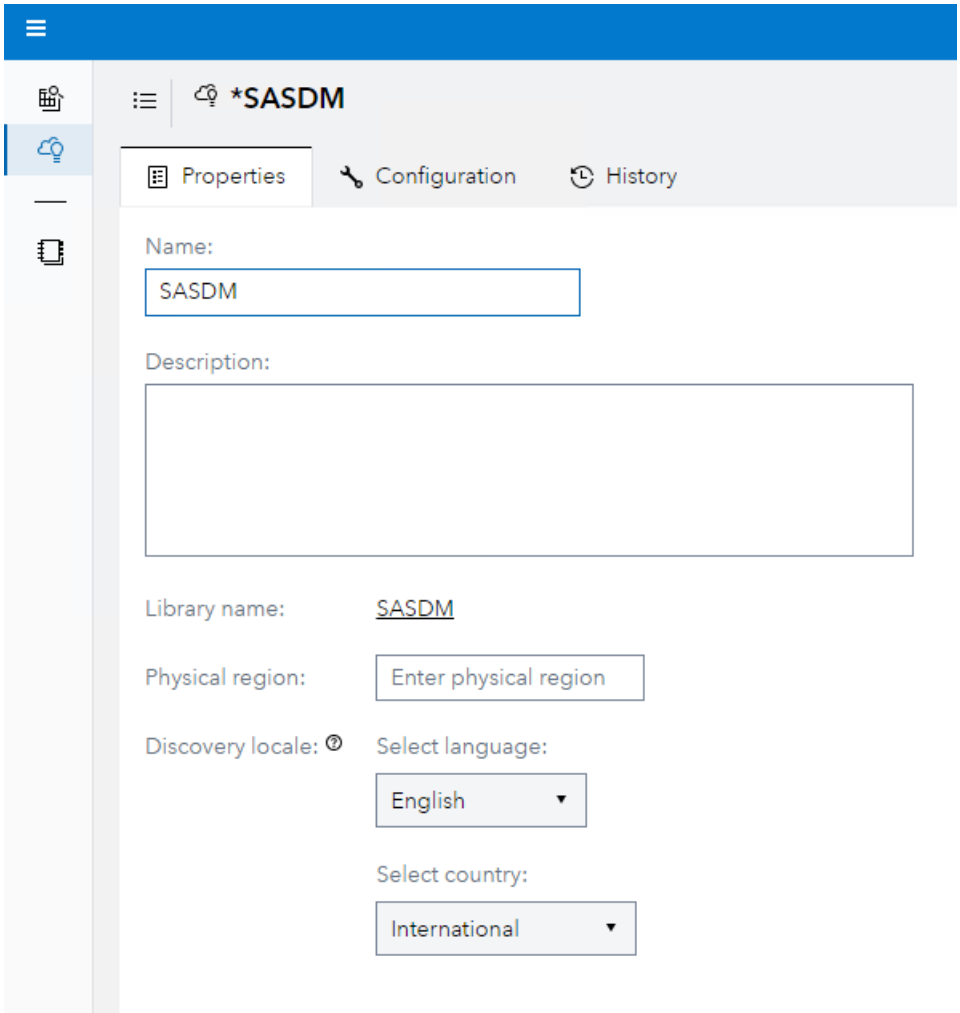
Select Discovery Agents



Select new Discovery Agent and choose the compute library SASDM.



No changes are required in the configuration:

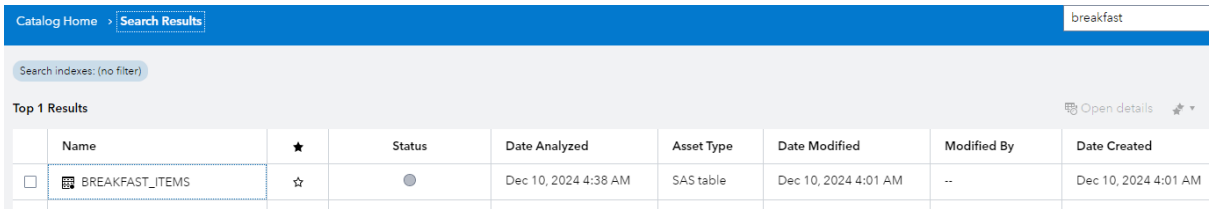


Select Run Now. This will save the discovery agent and run the Discovery Agent.

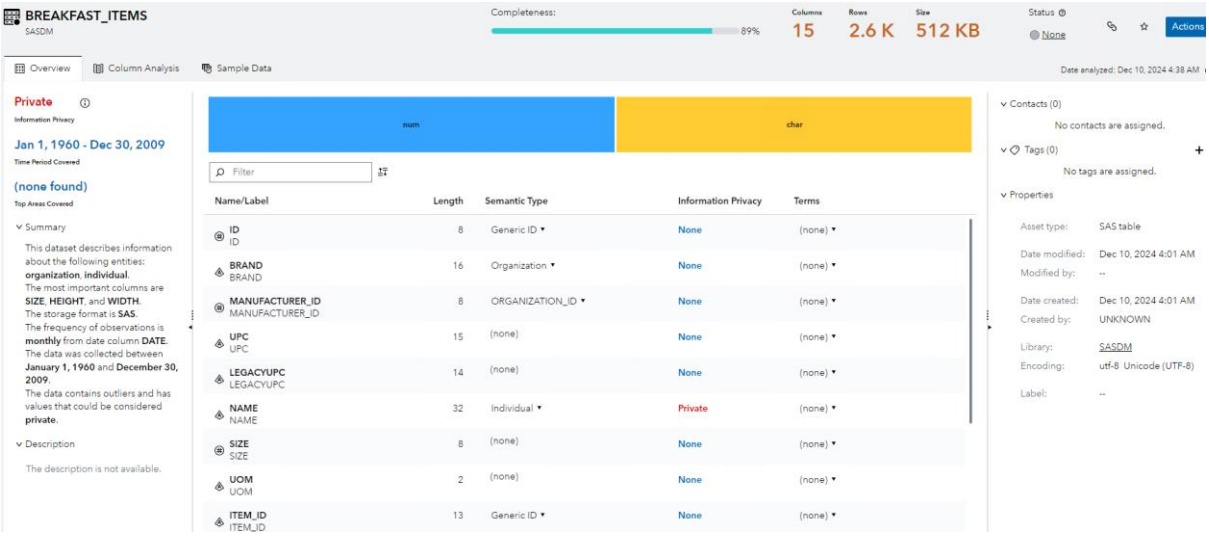
Same can be done for the PUBLIC Caslib.

3. Go to Discover Information Assets as Alex and search for the Breakfast_Items table.

Search for the Breakfast_Items table:



Select the Table and look at the Overview page:



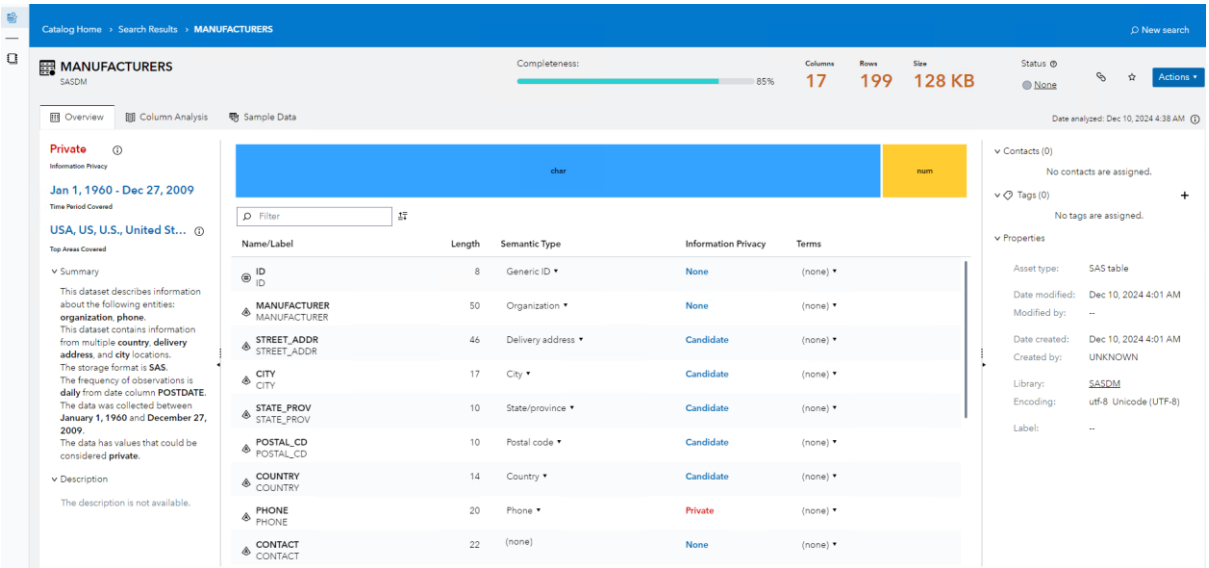
Select column analysis:

Investigate the columns UPC, UOM and BRAND.

- What is the completeness of each field?
- For UPC and UOM what is the most common Pattern?

Close the Breakfast_items table

Now search for the Manufacturers table:



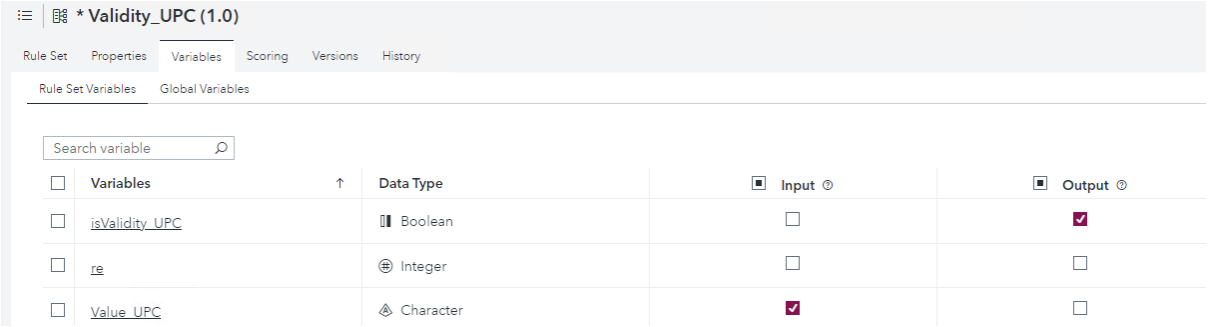
In the column analysis, investigate the completeness and Frequency Distribution of different fields:

What is the completeness of Street_addr, STATE_PROV, COUNTRY

4. Create Ruleset to check Validity of UPC (Uniform Product Code) in Breakfast_Items:

1. create Validity_UOM Ruleset:
   Duplicate the Ruleset Validity_Phone and save as Validity_UOM

- Rename Variables isAccuracy_UOM (Boolean), Value_UOM (string)



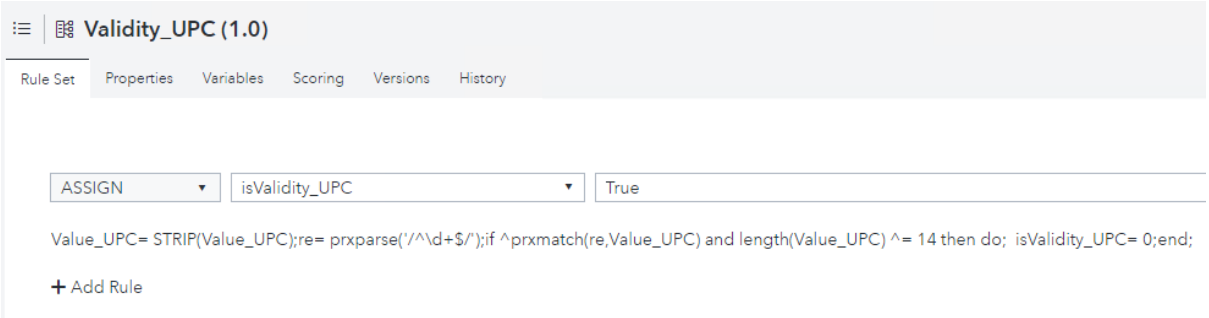
On the Rule Set tab:

- Rule: Validity_UPC
- Dimension: Validity
- Rule_Description: Checking type and length of UPC
- Expression:

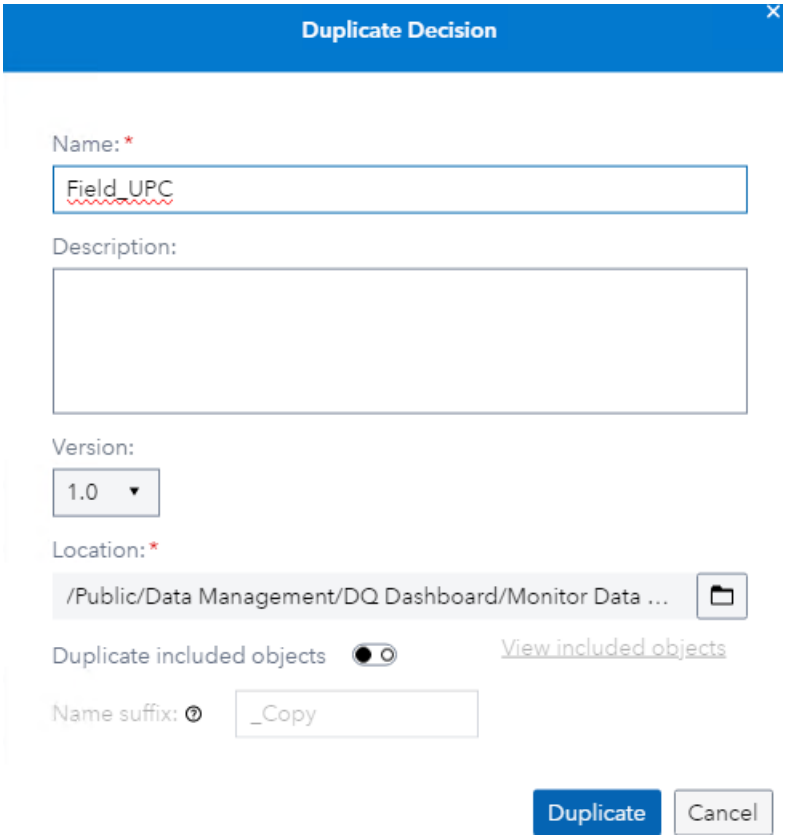
Assign: isValidity_UPC True

Change the custom expression to the expression below. It uses a Regex to check if not numbers are found in the Value_UPC and if the length is not 14 long.

```
Value_UPC= STRIP(Value_UPC)

re= prxparse('/^\d+$/');

if ^prxmatch(re,Value_UPC) and length(Value_UPC) ^= 14 then do;

  isValidity_UPC= 0;

end;
```

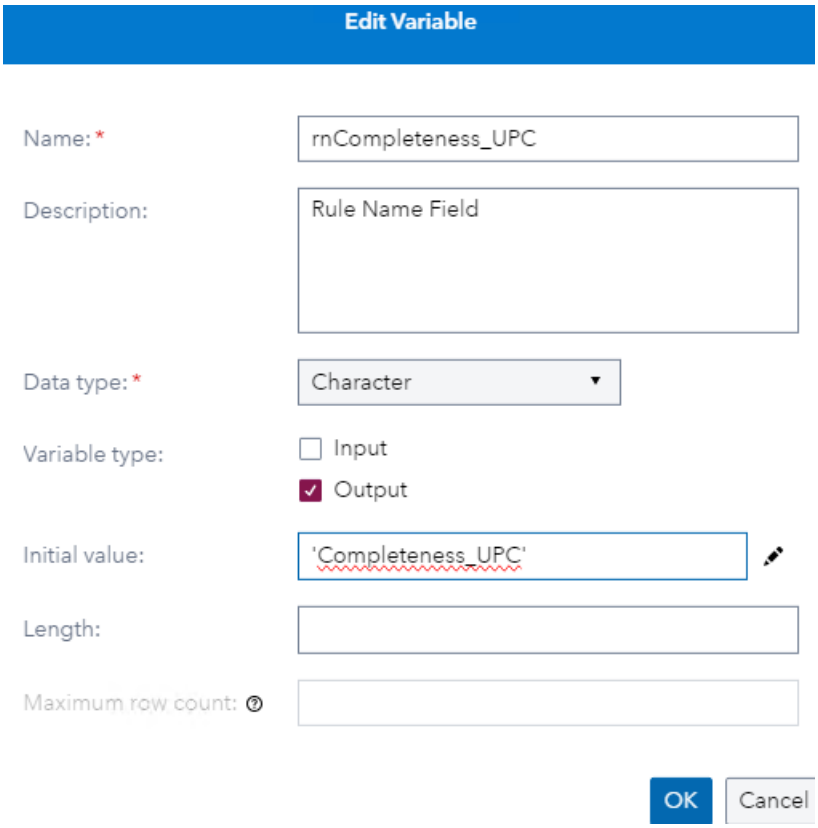


2. Create a Decision Field_UPC
   Select Decisions: select Field_Phone, check the checkbox, then top left menu: duplicate. Rename the Decision to Field_UPC

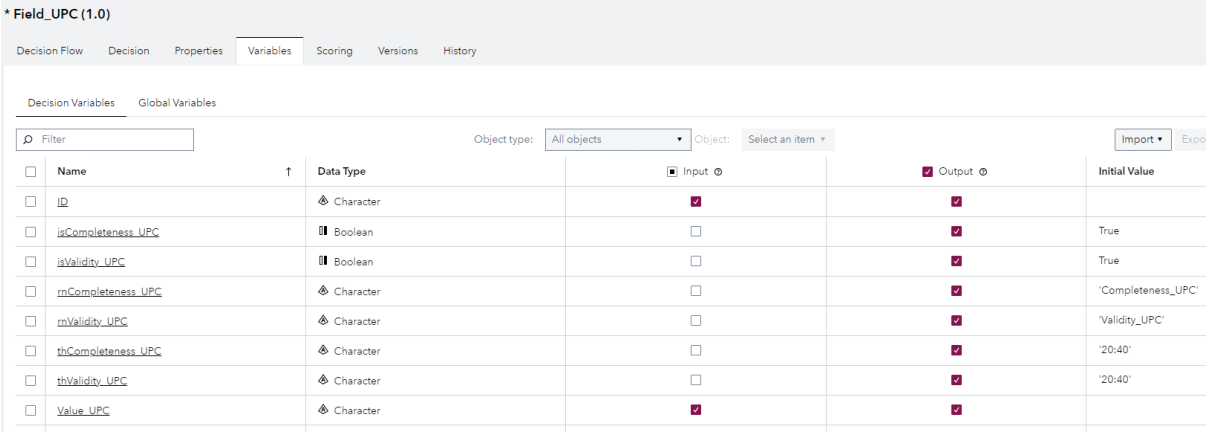


Select the variables tab, rename all variables and initial values from Phone to UPC

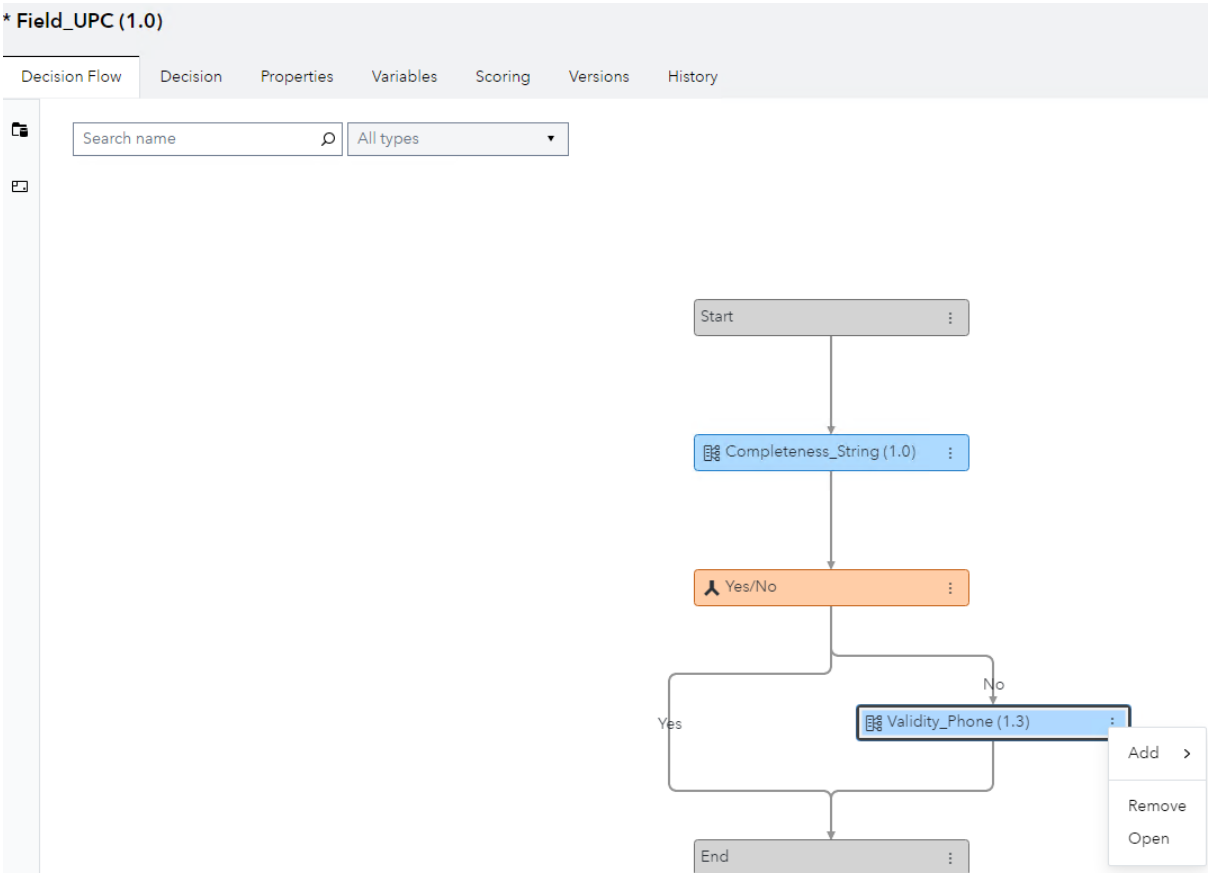
Remark: sometimes a variable rn also contains an initial value that needs to be renamed:



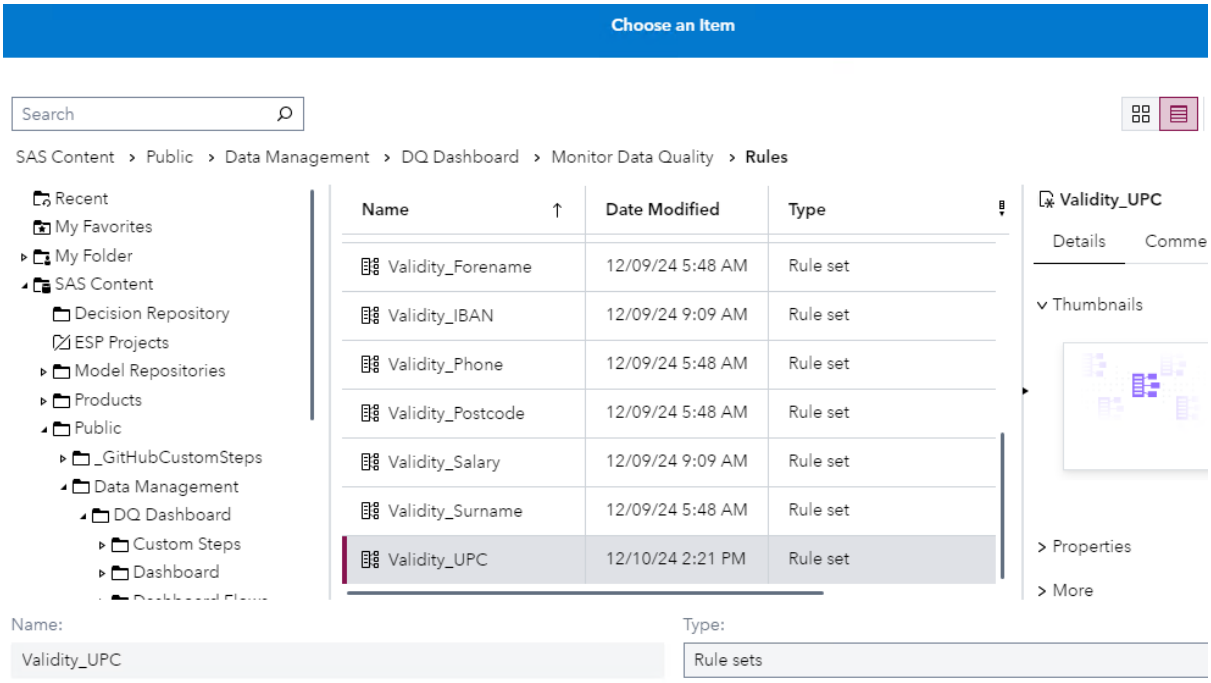
After the renaming, the variables tab should like like following screen:



On the Decision Flow, remove the Validity_Phone Ruleset and Add to the No Branch after the Completeness check, the Validity_UPC Ruleset



Then add to the No branch the Validity_UPC Ruleset:

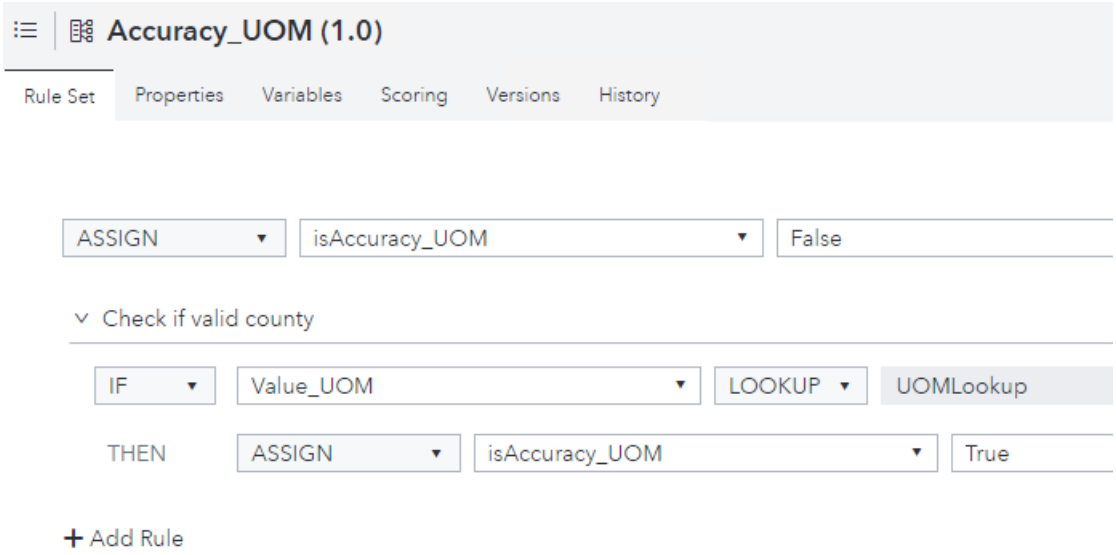


Check the mappings of the Input and Output variables. The Value_UPC in the decision should be mapped to the Value_UPC in the Ruleset. Same for the isValidity_UPC.

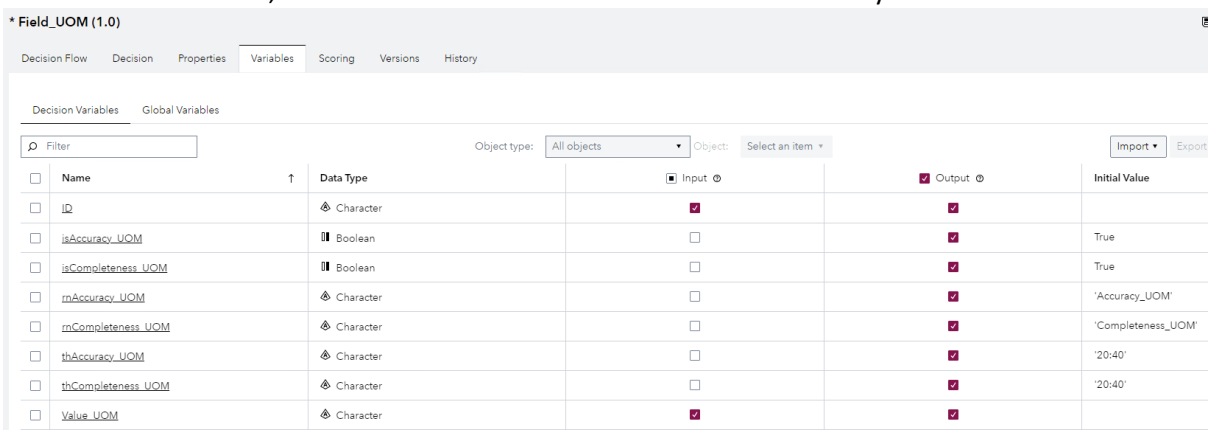
5. Create Ruleset to check Accuracy of UOM (Unit of Measure)

3. Create Lookup Table UOMLookup:
   CT=Carat, LB=Pound, OZ=Ounce, PK=Peck
   Store the lookup table and Activate it
4. create Accuracy_UOM Ruleset:
   Duplicate the Ruleset Accuracy_County and save as Accuracy_UOM

- Rename Variables isAccuracy_UOM (Boolean), Value_UOM (string)
- Rule: UOM Check
- Rule_Description: Checking the values for UOM
- Assign isAccuracy_UOM False
- Add Rule: if Value_UOM Lookup UOMLookup THEN then isAccuracy_UOM=true



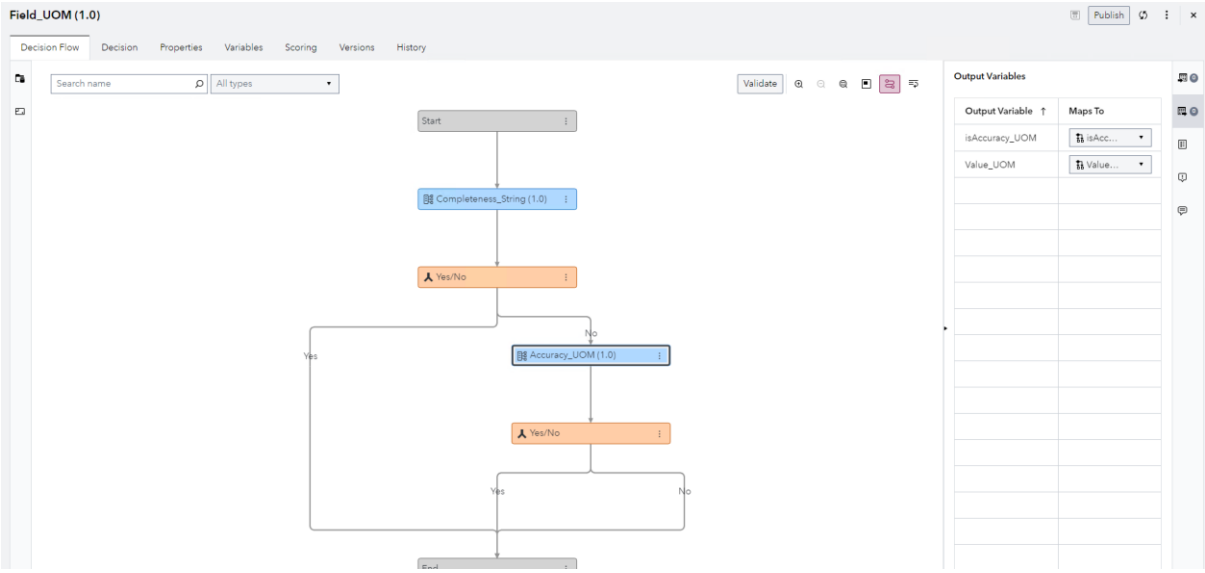
5. Create a Decision Field_UOM
   Select Decisions: select Field_County, check the checkbox, then top left menu: duplicate
   Rename the Decision to Field_UOM.
   Select the variable tab, rename all variables and initial values from County to UOM



Name	Data Type	Input	Output	Initial Value
ID	Character	☑	☑	
isAccuracy_UOM	Boolean	☐	☑	True
isCompleteness_UOM	Boolean	☐	☑	True
rnAccuracy_UOM	Character	☐	☑	'Accuracy_UOM'
rnCompleteness_UOM	Character	☐	☑	'Completeness_UOM'
thAccuracy_UOM	Character	☐	☑	'20:40'
thCompleteness_UOM	Character	☐	☑	'20:40'
Value_UOM	Character	☑	☑	

On the Decision Flow, remove the Accuracy_County Ruleset and Add to the No Branch after the Completeness check, the Accuracy_UOM Ruleset

Check the mappings of the Input and Output variables. The Value_UOM in the decision should be mapped to the Value_UOM in the Ruleset. Same for the isAccuracy_UOM.

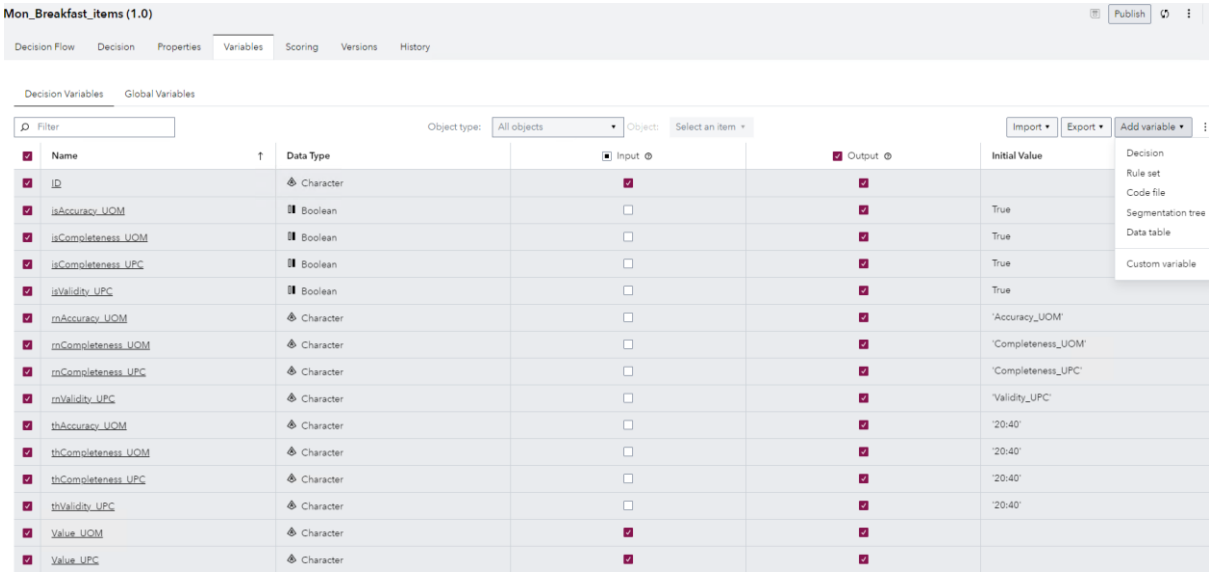

Field_UOM (1.0)
Publish
Decision Flow
Decision
Properties
Variables
Scoring
Versions
History
Search name
All types
Validate
Start
Completeness_String (1.0)
Yes/No
No
Yes
Accuracy_UOM (1.0)
Yes/No
Yes
No
Output Variables
Output Variable
Maps To
isAccuracy_UOM
isAcc...
Value_UOM
Value...

6. Create a Monitoring Task: Mon_Breakfast_items

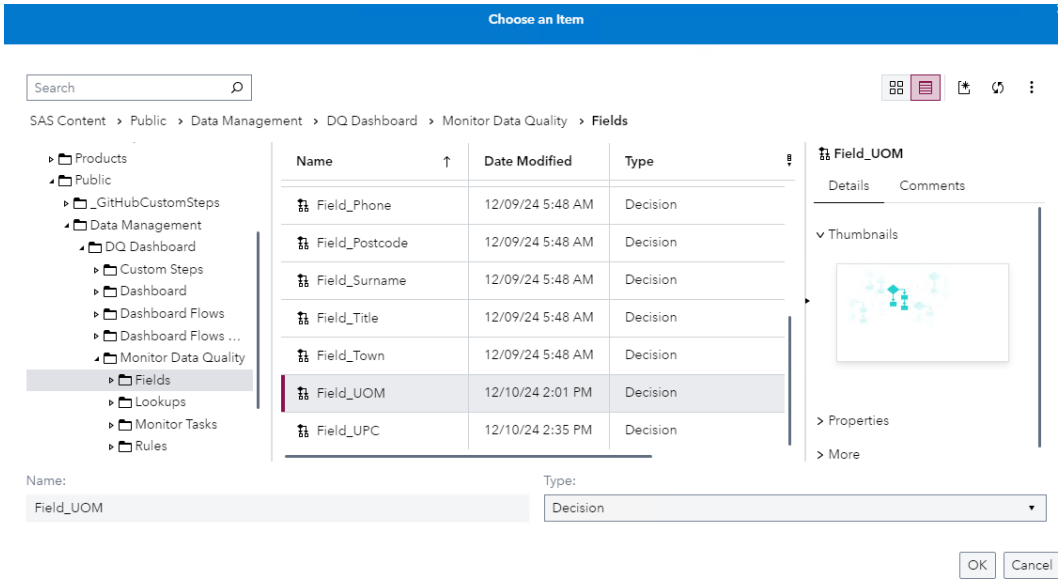
Start with a Duplicate of Mon_Person Decision.

Delete all the Field_ decisions from the flow, then delete the variables.

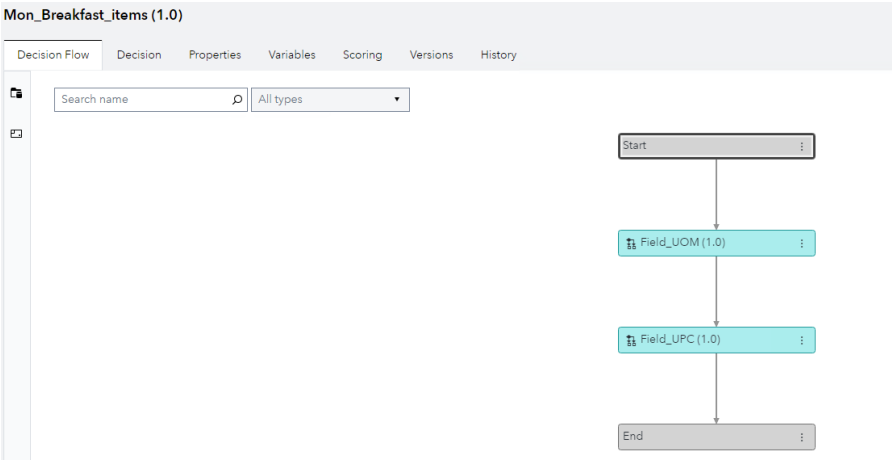
Add Variables from the Field_UOM and Field_UPC decisions from /Public/DataManagement/DQDashboard/Monitor Data Quality



On the Decision Flow, add the Field_UOM and Field_UPC Decisions from the folder



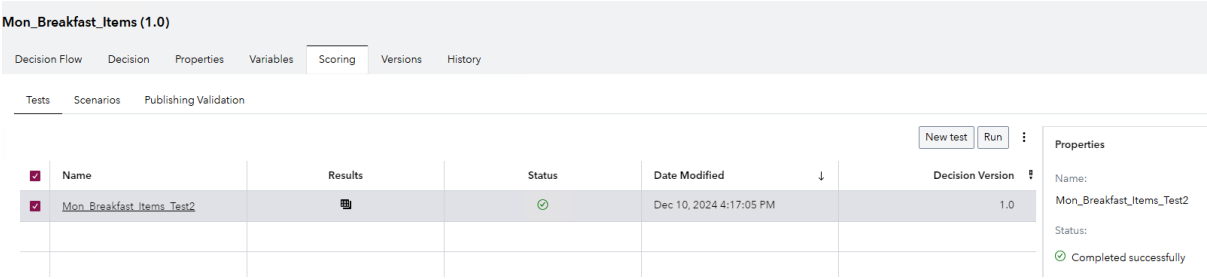
The resulting Monitor Task Decision flow should look like:



Select Scoring, add a new Test:

Select Breakfast_items and map the columns UPC and UOM to Value_UPC and Value_UOM.

Run the test, check if the test is successful.

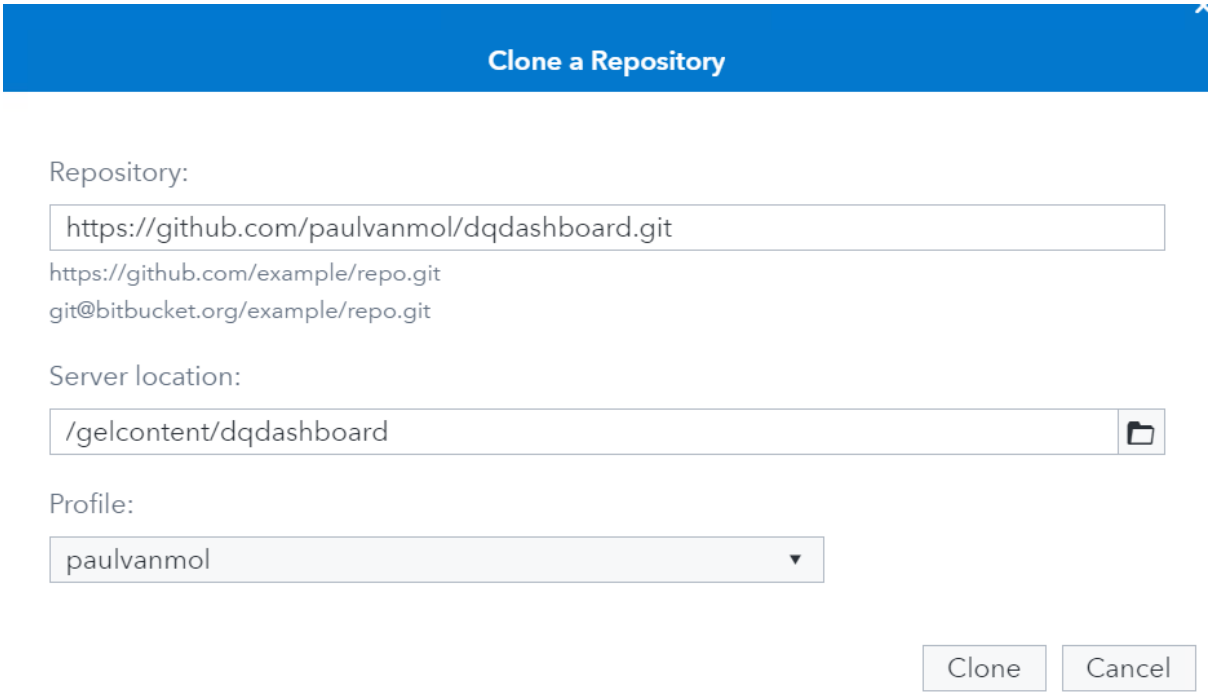


7. Appendix

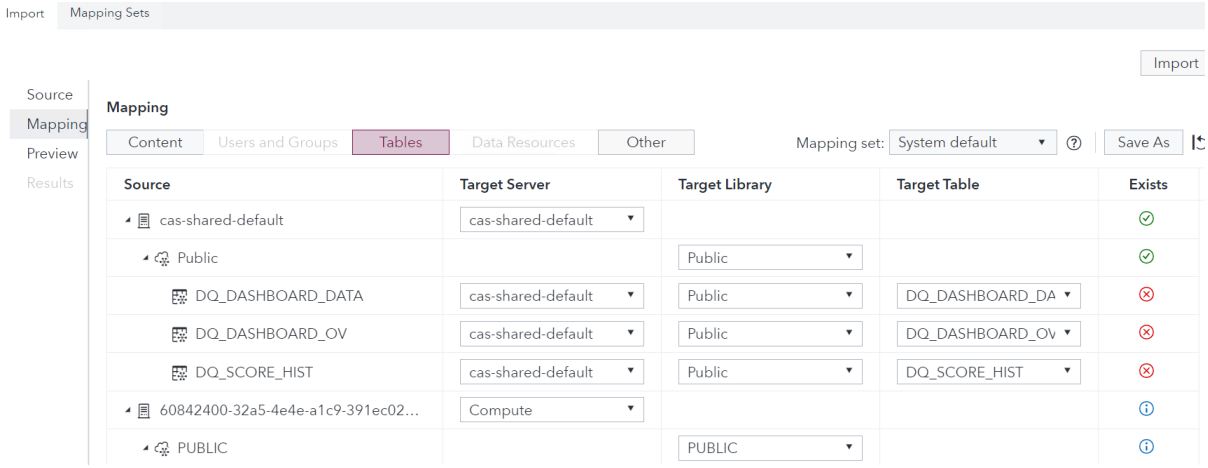
Prepare Demo on Training Virtuallab

To ensure that DQ Dashboard demo works as desired follow the steps below to load the data into memory and to run the Studio Flows to load the DQ Dashboard tables.

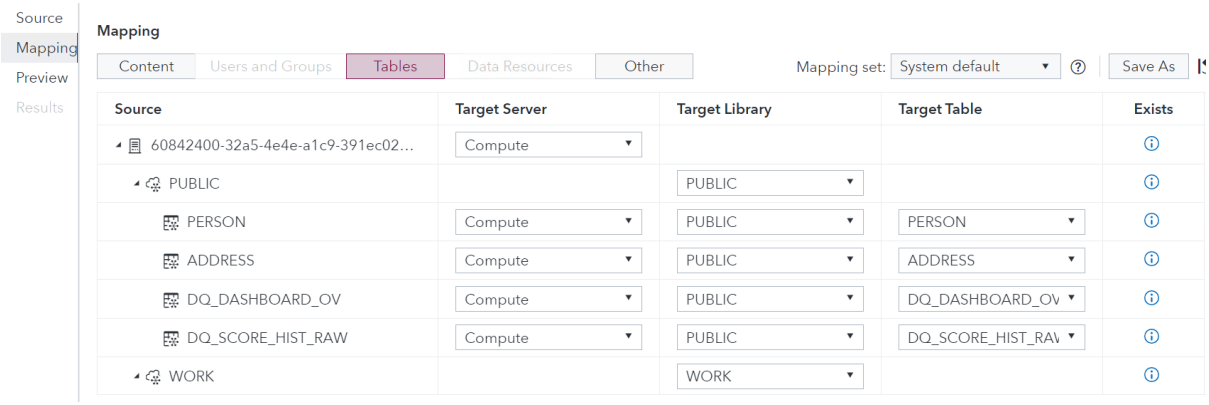
1. Fork the following repository: https://github.com/paulvanmol/dqdashboard.git to your personal git account
2. Clone the Repository from SAS Studio
   Create a new empty folder dqdashboard:
   Create a new git profile:



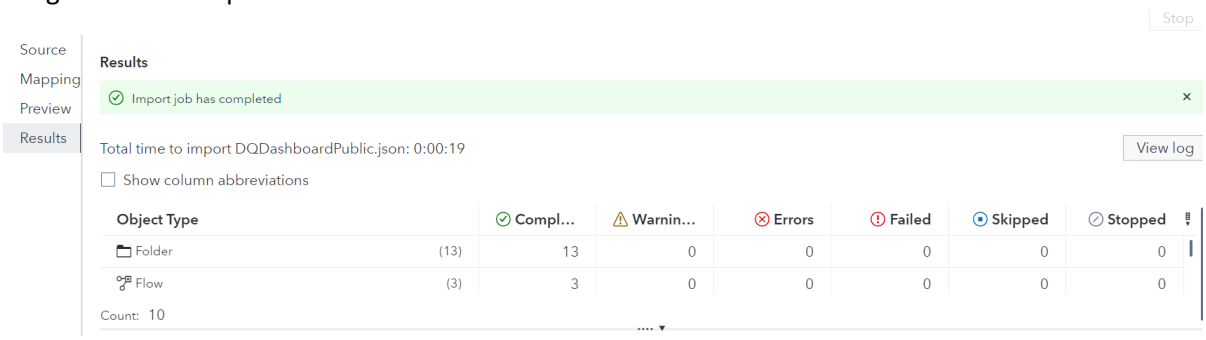
3. Download the required files from the dqdashboard folder:
   a. Packages/DQDashboard.json package from SAS Studio to your Downloads folder.
   b. Data/ADDRESS.xlsx, PERSON.xlsx, DQ_SCORE_HIST_RAW.xlsx
4. Import DQDashboard.json package in SAS Environment Manager
   a. Login as sasadm + lnxsas to have administrative privileges:
   b. Select Manage Environment:
   c. Select Import tab
   d. Import the DQDashboardPublic.json
   e. Map the Target Server to cas-shared-default, Map the Target Library PUBLIC, WORK CAS Library to the Public CAS Library:



f. Other tables should map to Compute, Public and WORK libraries:



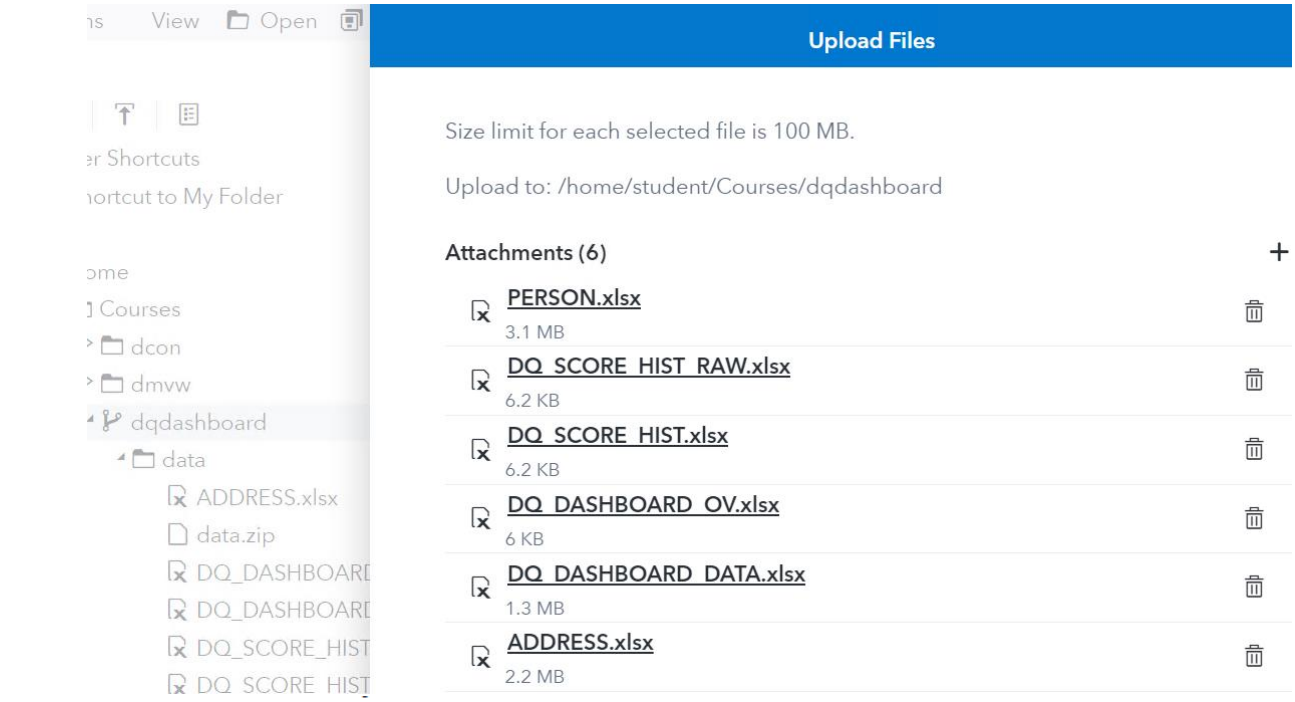
g. Check if Import is successful:

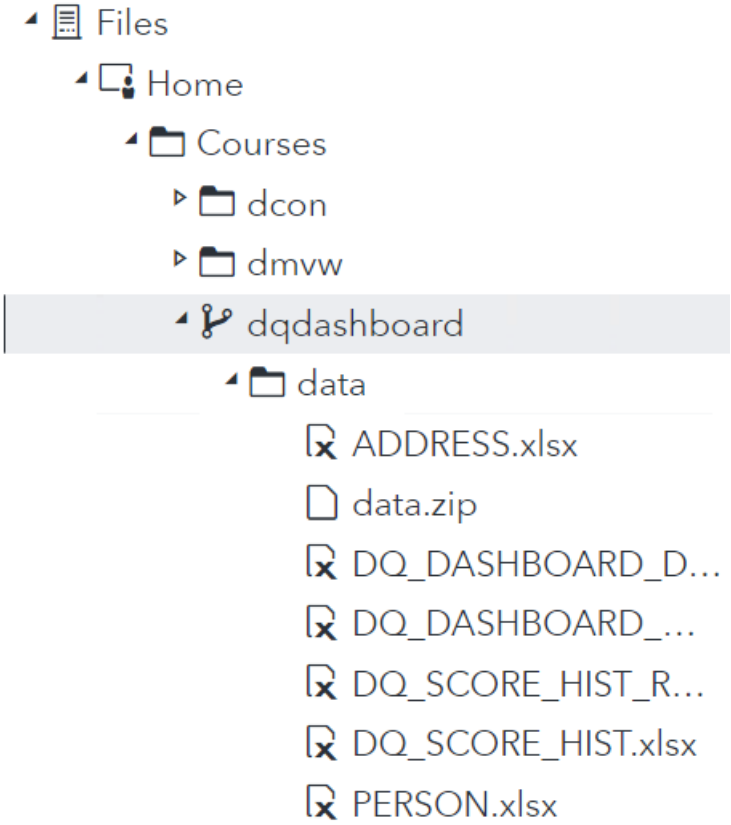


Upload the data to SAS Studio Explorer:

Upload the data of the dqdashboard repository to the home directory of the student:

- Either by doing a Git Clone in SAS Studio to the /gelcontent/dqdashboard folder
- Or by using the upload Button in the Explorer:


View
Open
Upload Files
Size limit for each selected file is 100 MB.
Upload to: /home/student/Courses/dqdashboard
Attachments (6)
PERSON.xlsx
3.1 MB
DQ_SCORE_HIST_RAW.xlsx
6.2 KB
DQ_SCORE_HIST.xlsx
6.2 KB
DQ_DASHBOARD_OV.xlsx
6 KB
DQ_DASHBOARD_DATA.xlsx
1.3 MB
ADDRESS.xlsx
2.2 MB
Courses
dcon
dmvw
dqdashboard
data
ADDRESS.xlsx
data.zip


Files
Home
Courses
dcon
dmvw
dqdashboard
data
ADDRESS.xlsx
data.zip
DQ_DASHBOARD_D...
DQ_DASHBOARD_...
DQ_SCORE_HIST_R...
DQ_SCORE_HIST.xlsx
PERSON.xlsx

Go to SAS Studio (Develop Code and Flows)

1.1. Go to location: SAS Content/GTPPub/Data Management/DQ Dashboard/SAS Programs

1.2. Open and run SAS job: Load tables into memory.sas

```
cas casauto;
%let outcaslib=PUBLIC;
%let incaslib=GTPPUB;
%let path=/gelcontent/dqdashboard;

proc cas;
     session casauto;
table.dropcaslib /caslib="&incaslib" quiet=true;
table.addCaslib /
     caslib="&incaslib"
     description="Monitor data"
     dataSource={srctype="path"}
     path="&path/data";

      table.dropTable  /
            caslib= "&outcaslib",
            name= "ADDRESS",
            quiet= True;
      run;
      table.loadTable /
            caslib= "&incaslib",
            path="ADDRESS.xlsx",
            casout={
                  caslib="&outcaslib",
                  promote= True
                  };
      run;
      table.dropTable  /
            caslib= "&outcaslib",
            name= "PERSON",
            quiet= True;
      run;
      table.loadTable /
            caslib= "&incaslib",
            path="PERSON.xlsx",
            casout={
                  caslib="&outcaslib",
                  promote= True
                  };
      run;
      table.dropTable  /
            caslib= "&outcaslib"
            name= "DQ_SCORE_HIST_RAW",
            quiet= True;
      run;
      table.loadTable /
```

```
            caslib= "&incaslib",
            path="DQ_SCORE_HIST_RAW.xlsx",
            casout={
                  caslib="&outcaslib",
                  promote= True
                  };
      run;
quit;
```

1.3. Go to location: SAS Content/Public/Data Management/DQ Dashboard/Dashboard Flows

1.4. Open and run flows:

- Monitor_Person.flw
- Monitor_Address.flw
- Write_Dashboard_Tables.flw

2. Go to SAS Drive (Share and Collaborate)

2.1. Open Dashboard: DQ Dashboard in location: SAS Content/Public/Data Management/DQ Dashboard/Dashboard to ensure the Dashboard data got generated correctly.