Lab: Data Access and Preparation in Watson Studio

November 19, 2018

Author: Elena Lowery elowery@us.ibm.com

Table of contents

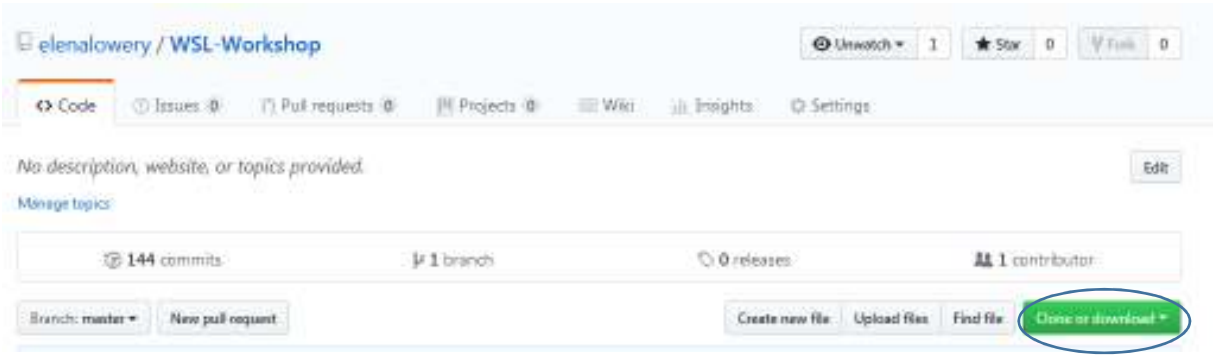
Contents

Overview

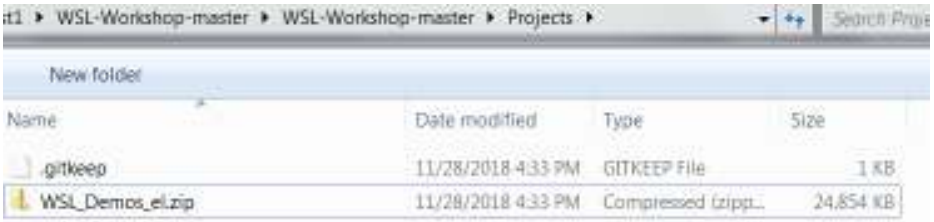
In this lab you will learn how to work with database data sources in Watson Studio. We will also review options for data preparation. The step-by-step instructions are provided for **Watson Studio Local** (WSL). We also provide high-level instructions for similar capabilities in **Watson Studio Cloud**.

Required software, access, and files

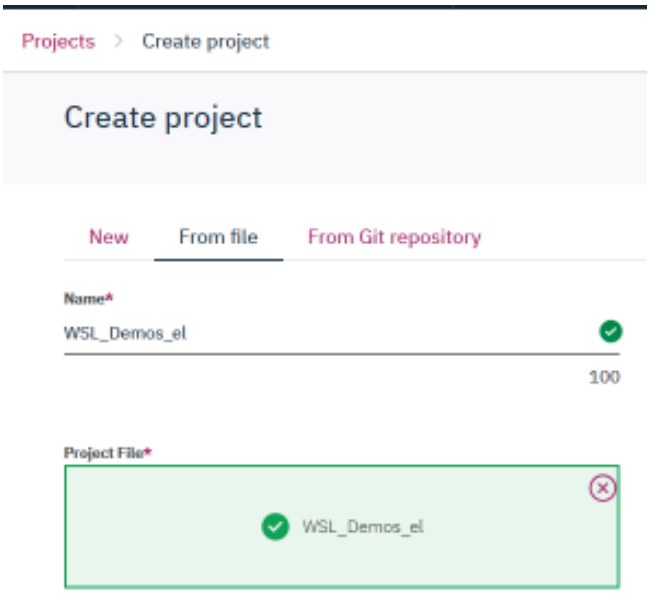
- To complete this lab, you will need access to a **Watson Studio Local** cluster.
- You you will also need to complete the following steps to import the sample project (if you have not previously loaded this project):
    - Download and unzip this GitHub repository: https://github.com/elenalowery/WSL-Workshop



    - In the **Projects** directory of the unzipped file, rename *WSL_Demos.zip* to a unique name, for example, add your initials.



    - Log in to **WSL** and create a project **From File**, using the *WSL_Demos.zip* file that you just renamed.



- Optional: an **IBM Cloud account** (only if you want to complete the lab in **Watson Studio Cloud**): https://dataplatform.cloud.ibm.com

Part 1: Configuring Data Access

CSV and other files

WSL supports working with *csv* files. Csv files are loaded into the project and stored in the file system. In general, csv files should be used for experimentation/testing, and not as a main data source for developing models and scoring.

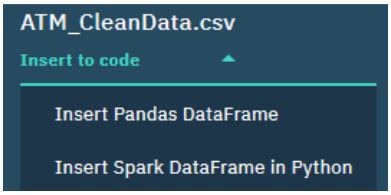
WSL replicates all project files across the cluster and creates a copy of the project for each collaborator. If a project has a large number of csv files or large csv files, this will affect overall system performance. If you have to include a large volume of csv files into a project, create a *Library* project type. WSL does not create copies of a *Library* project for each collaborator.

1. Open the demo project you've used in other labs and click on **Data sets.** We already have several .cvs files in the project. Click on the vertical ellipses to preview any file.

   Note: csv files are assumed to have headers and use a comma as a field separator. Malformed csv files can't be previewed.

2. Click **Add Data Set** to load new files. Notice that you're not restricted to load just *csv* files. You can load other file types, for example, .txt or .jpg. If you wish, you can import other file types.

3. Open the test notebook you created in one of the previous labs or create a new notebook. Click on the data icon and try the **Insert to code** functions for any *csv* file.




```
import os, pandas as pd
# Add asset from file system
df_data_1 = pd.read_csv(os.environ['DSX_PROJECT_DIR']+'/datasets/ATM_CleanData.csv')
df_data_1.head()
```

While **Insert to code** is not available for non-csv file types, these files are available in the */datasets* directory, and can be accessed programmatically.

If a large number of files has to be loaded into a project or if you need to update the files in the project on a regular basis, the upload process can be automated. See documentation for more information: https://content-WSLlocal.mybluemix.net/docs/content/local/sshd.html

Database data sources

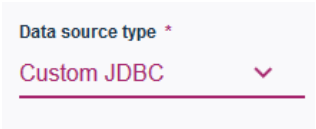
Note: Not every IBM environment has external database connectivity. Please check with the lab instructor if this section can be completed in your WSL environment.

In this section you will learn how to create a data source, datasets, and test connectivity in WSL.

1. In the **Project** view, click **Data Sources**, then **add data source**. Click on the **Data source type** dropdown.

   WSL includes *jdbc* drivers for every database data source that's shown in the dropdown. WSL also provides built-in connectivity for the listed Hadoop data sources.

   If the data source that you would like to use is not listed, then **Custom JDBC** option can be used. Custom jdbc option requires that you provide a jdbc driver for the data source.



   Instructions for adding a custom JDBC driver are available here: https://content-dsxlocal.mybluemix.net/docs/content/SSAS34_current/local/drivers.html

   While it's possible to connect to data directly from IDEs using Python, R or Scala code, defining a data source in the project has several benefits:

   - **Better organization**: we can see which data sources are used in a project.
   - **Security**: credentials that are used to access a data source are encrypted. Credentials are not shared between collaborators.
   - WSL provides **code generation** for datasets that are associated with data sources.

IBM compatibility reports list data sources that are officially supported by WSL (see **Reference**). Since the majority of IDEs in WSL are open source IDEs, there is no limitation for which data source can be used. The *supported* data sources are the ones that have been tested by IBM and will be supported by IBM tech support.

Set up a database

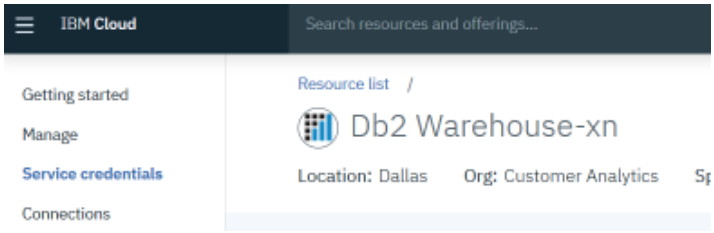
In this section we will set up a database in *IBM Cloud* so that we can test database access from WSL. If you already have an external database or service that you would like to use, you can skip this section.

Note: If you are not able to create the DB2 Warehouse service, please ask the lab instructor for a pre-configured instance.

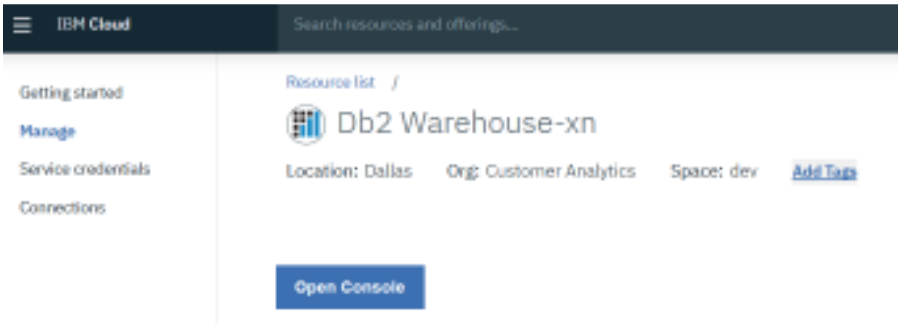
1. Create a *DB2 on Cloud* service in **IBM Cloud**.

   - Login to **IBM Cloud**: *https://www.ibm.com/cloud/*
   - Click on **Catalog**. Search for db2 warehouse" and create the service.

2. Generate service credentials in **IBM Cloud** and save them in a notepad.



3. Click **Manage** then **Open Console** to open the admin console.

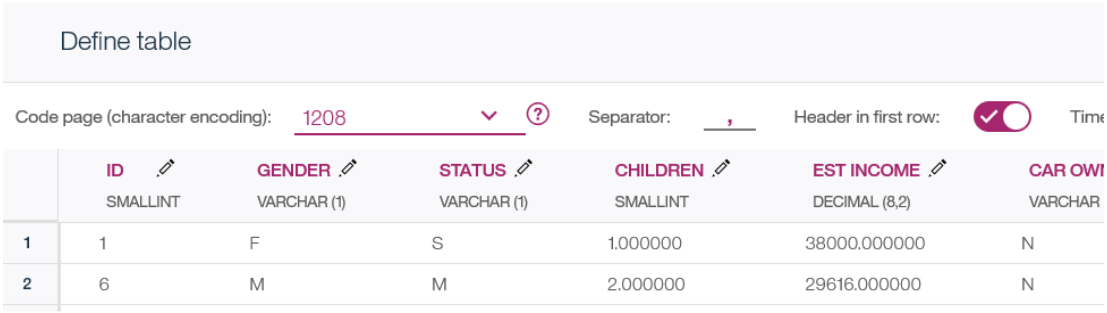


1. Click **Load.**

2. Select **browse files** and navigate to the *data* folder of the unzipped GitHub repository. Select *customer.csv*. Click **Next**.

3. Select *Schema* (which will be different than the screenshot in your instance) and click **New Table**. Enter table name *CUSTOMER* and click **Create**. Click **Next**.

4. Leave the default values on the *Define Table* screen. Click **Next**. Then **Begin Load**.



5. If you want to verify that data has been loaded successfully, click **View Table**.

6. Optionally, repeat the data load steps for all files in the */data* directory. If you upload all data files to database, you can change notebooks to use a database data source instead of csv files.

Configure database connection in WSL

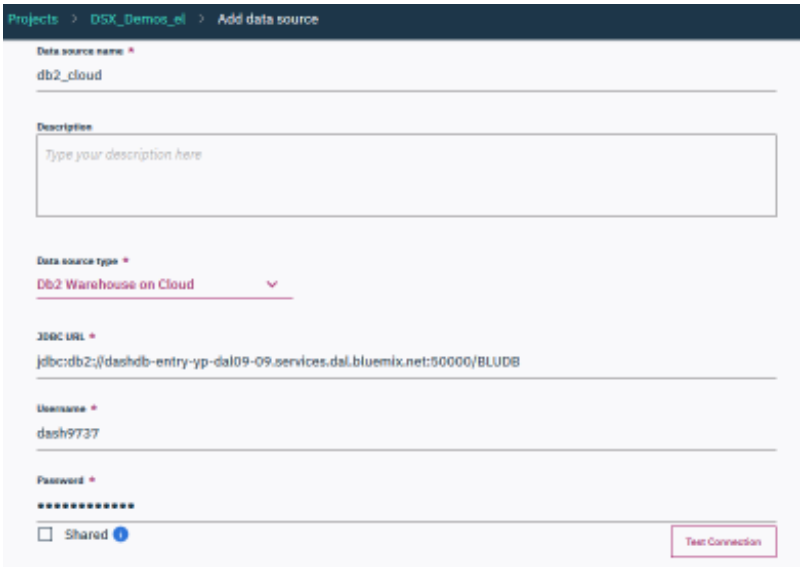
In this section we will define a database connection in the WSL UI and test it in a notebook.

1. Open your WSL Local project. Click on **Data Sources**, then **add data source.**

2. Enter data source name (for example, *db_cloud*) and fill out the required fields (which you saved from *Service Credentials* view in **IBM Cloud**).

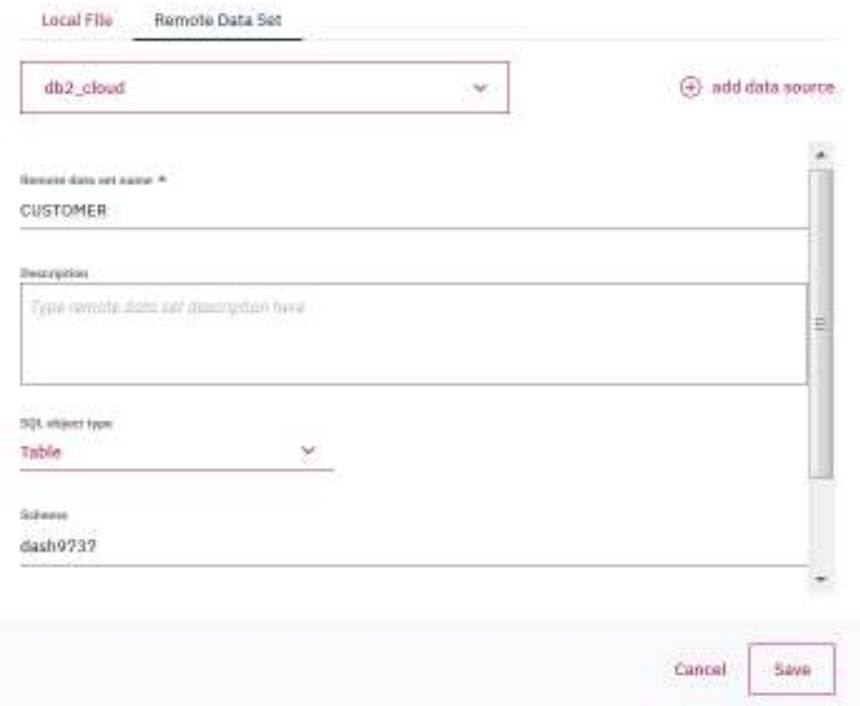
   Do not check the "Shared" checkbox. If you select it, then your credentials will be shared with collaborators on the project.

   Click **Test Connection**. After you see the success message, click **Create**.



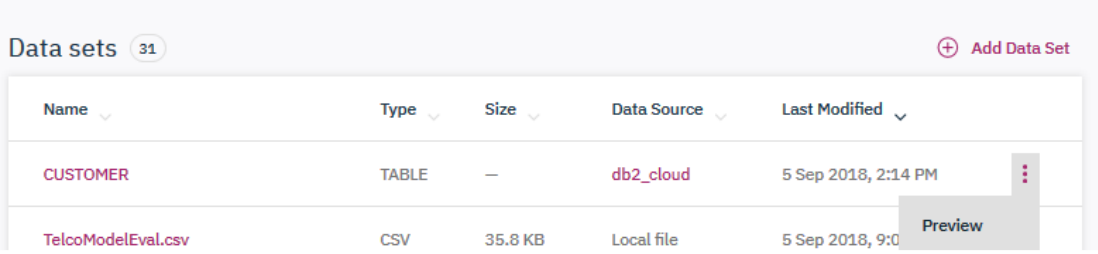
3. Switch to the **Assets** view and select **Data Sets**. Click **add data set**. Select **Remote Data Set** and provide the required fields.

   Click **Save**.



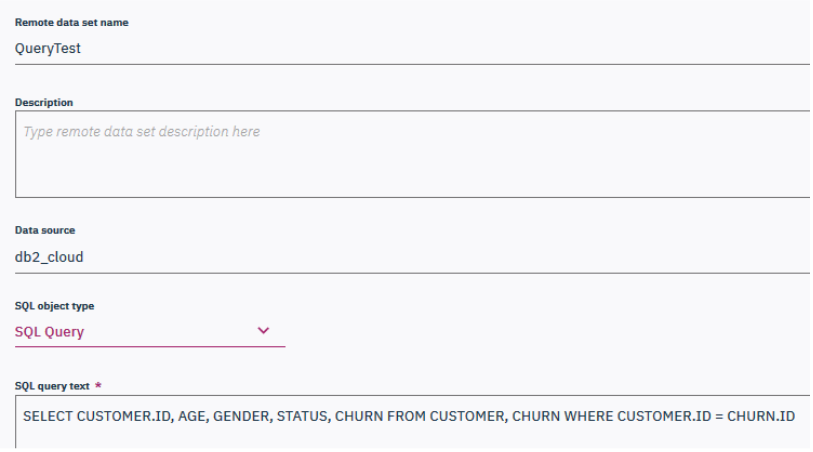
4. If you created tables from other *csv* files, create the **Remote Data Source** for each of them.

5. You can preview remote data sets similar to the way that you can preview csv files.



6. You may have noticed that you can create an *SQL Query* type of data set.

You can use this query: **SELECT CUSTOMER.ID, AGE, GENDER, STATUS, CHURN FROM CUSTOMER, CHURN WHERE CUSTOMER.ID = CHURN.ID**



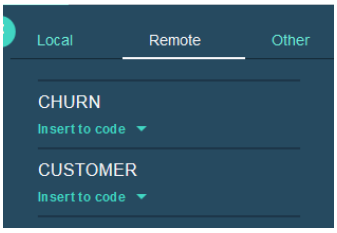
After the query data set has been created, you can test it with the **Preview** option.

ID	AGE	GENDER	STATUS	CHURN
3626	52.146667	M	M	T
3628	40.313333	F	S	T
3631	51.520000	M	M	F
3632	44.573333	M	M	T

Data access code generation

1. You have already tested code generation for local csv files. Now you can also test it for the database data sources that you created.

2. Open the test notebook and use **Insert to Code** to generate code to database data sources that we created earlier in the lab.



3. Run the code and make sure data is displayed.

```
df1.head()
```

	ID	GENDER	STATUS	CHILDREN	EST_INCOME	CAR_OWNER	AGE	LONGDISTANCE	INTERNATIONAL	LOCAL	DROPPED	PAYMETHOD	LOCALBILLTYPE	LONGDISTANCEBILLTYPE	USAGE	RATEPLAN
0	886	F	M	1	34555.0	N	23.000000	25.77	0.00	8.84	0	CH	FreeLocal	Standard	34.61	4
1	887	F	M	2	11234.7	Y	49.046667	0.69	0.00	112.72	0	CC	Budget	Standard	113.42	2
2	889	F	M	2	36660.9	Y	39.960000	0.00	0.00	3.36	1	CH	Budget	Intnl_discount	3.36	3
3	891	F	M	2	68462.8	N	34.400000	24.39	5.93	60.51	0	CC	FreeLocal	Standard	90.84	1
4	894	F	M	2	72841.3	N	47.020000	10.36	0.00	72.16	0	CC	Budget	Standard	82.53	2

4. If you wish, change the sample notebooks that use csv data sources to use a database data source.

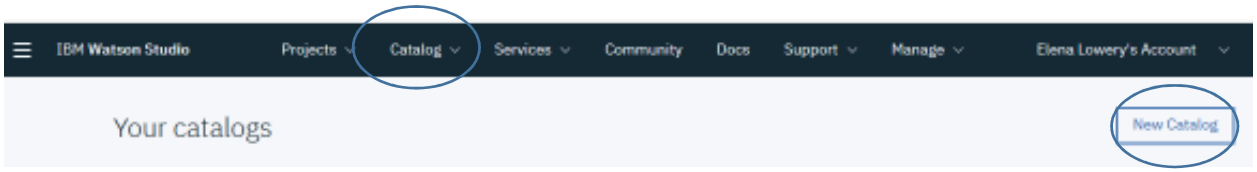
   Make sure to insert the correct data frame type. *TelcoChurn* notebook uses Spark data frames

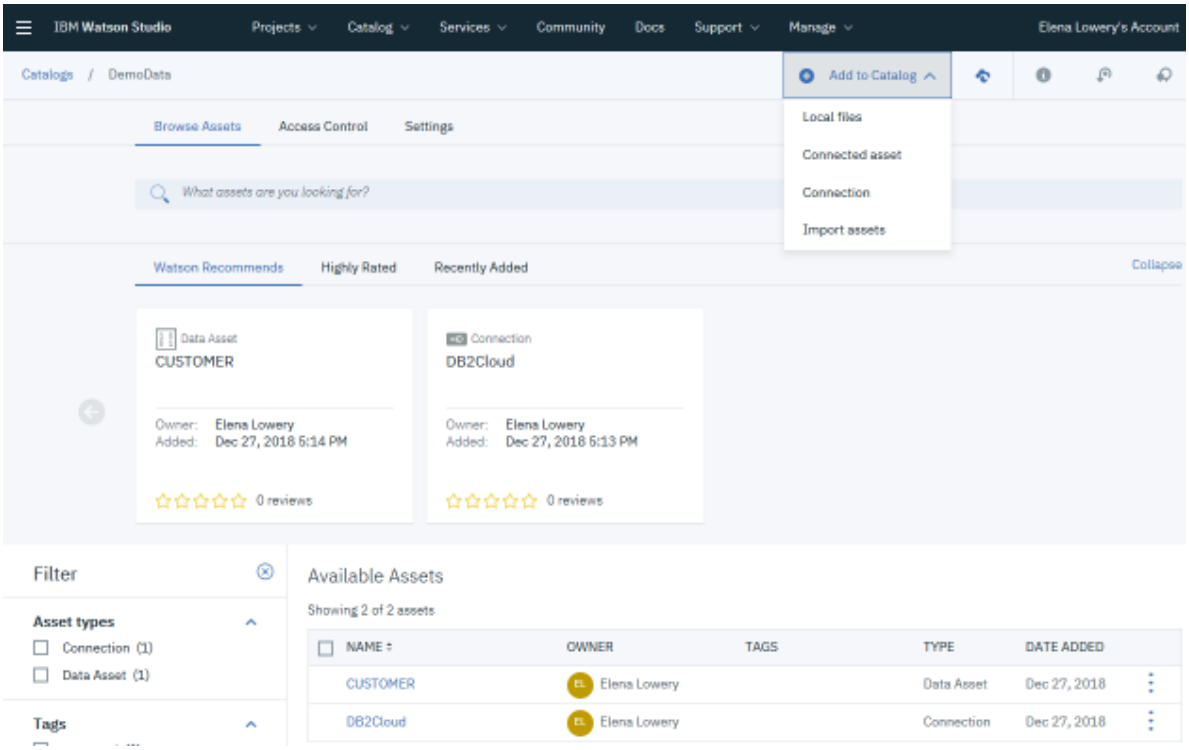
   In addition to generating the code, you may need to change variable names. Please check with the lab instructor if you need help with understanding how to modify the code.

Data Access in Watson Studio Cloud

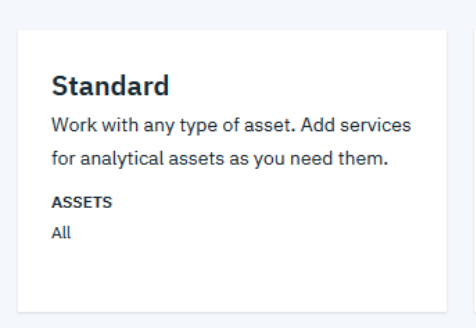
Before we can access data in **Watson Studio Cloud**, we need to set up a database that we can access. If you have not completed the **Set up a database** (set up the DB2 warehouse service) step earlier in the lab, make sure to complete it.

Optionally, you can set up a **Catalog** in your Watson Studio workspace. In the **Catalog**, configure the connection to *DB2 Warehouse service* and the *CUSTOMER* table (called *Connected dataset* in Watson Studio).



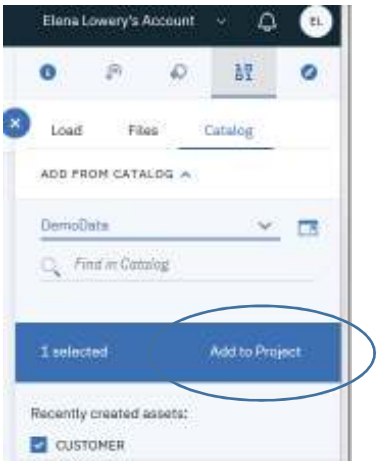


1. In Watson Studio create a new **Standard** project.

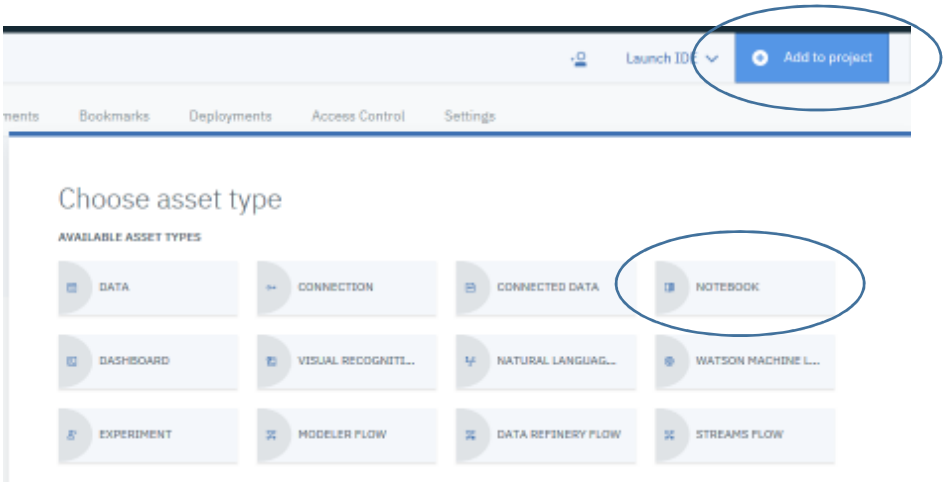


2. If you created a catalog, add the *CUSTOMER* data set to the project. If you didn't create a catalog, you will need to configure connections manually (in the following steps).

   In the **Project view** click on **Assets**. Click on Catalog, select the CUSTOMER data set and add it to the project.



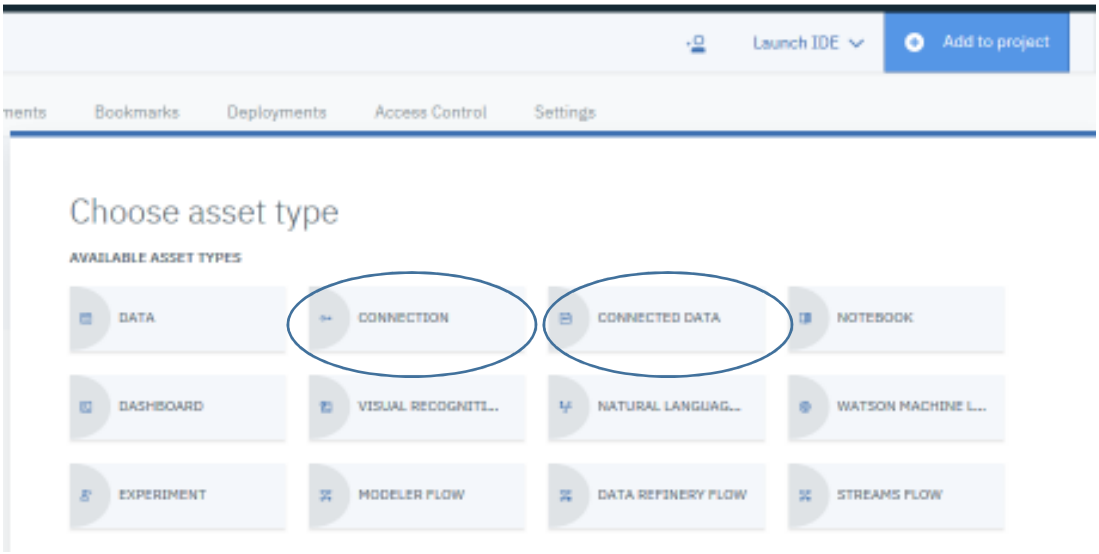
3. Add a notebook.



4. When the notebook opens, click on the *Data* icon in the top right corner.

   If you set up a Catalog, you will see the CUSTOMER data source, and you can use the Insert to Code option.

If you have not set up the Catalog, using Add to Project menu, first create a Connection, then Connected Data. After you complete these steps, the data set will be available in the Data menu.



Part 2: Data Preparation

Data Preparation options

WSL provides multiple options for data preparation:

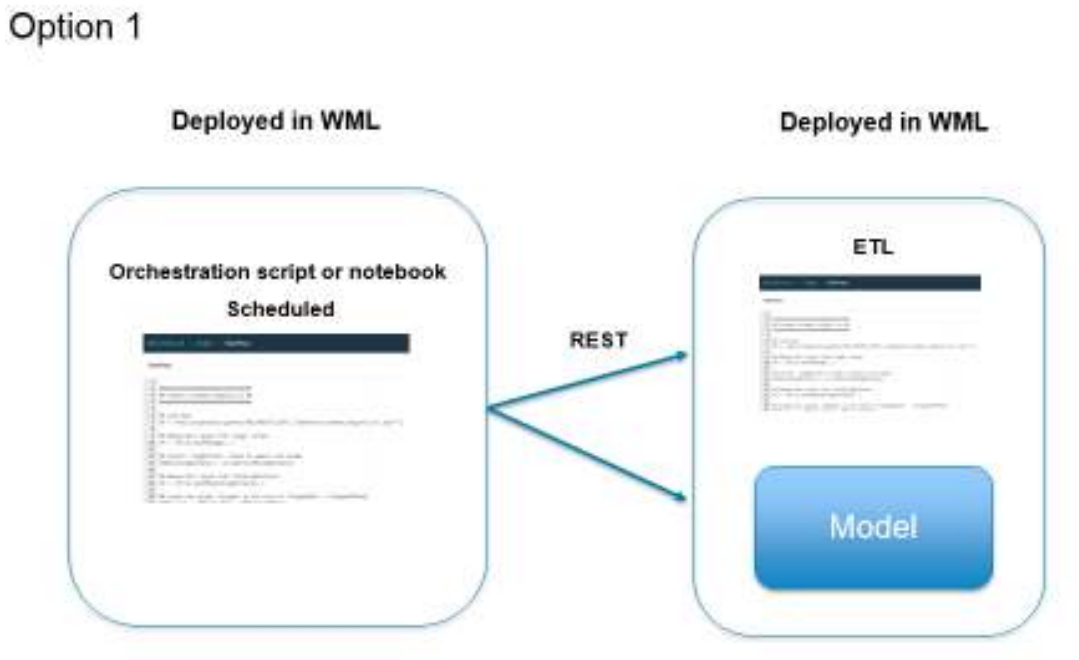
1. Import existing data preparation assets into WSL: Python/R scripts, notebooks
2. Create new Python/R scripts
3. Use data preparation tools included in WSL.

The data preparation tools that are included in WSL are **Data Refiner** and **Modeler**.

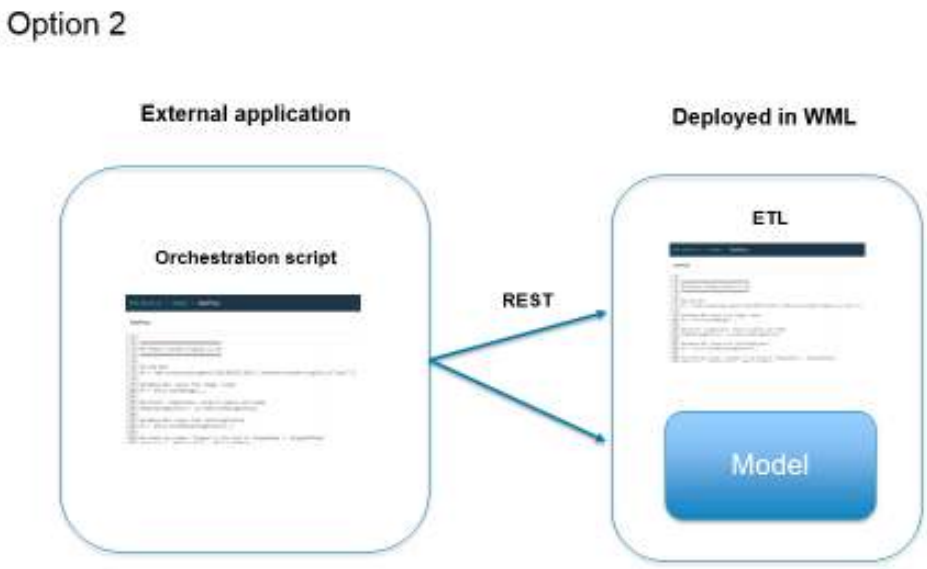
- **Data Refiner** is a data visualization and shaping tool which is included with WSL.
- **Modeler** is a comprehensive visual data mining workbench.

All data preparation assets can be invoked interactively or deployed for batch scoring. Batch deployments in WSL can be invoked with a scheduler or a REST API. With these capabilities an administrator can decide how to invoke and orchestrate data preparation assets. Some of the options for configuring data preparation jobs are:

- **Option 1:** Data preparation is "self-contained" in WSL: invoked with a WSL scheduler or a script/notebook in WSL



- **Option 2:** Data preparation assets are invoked by an external applications.

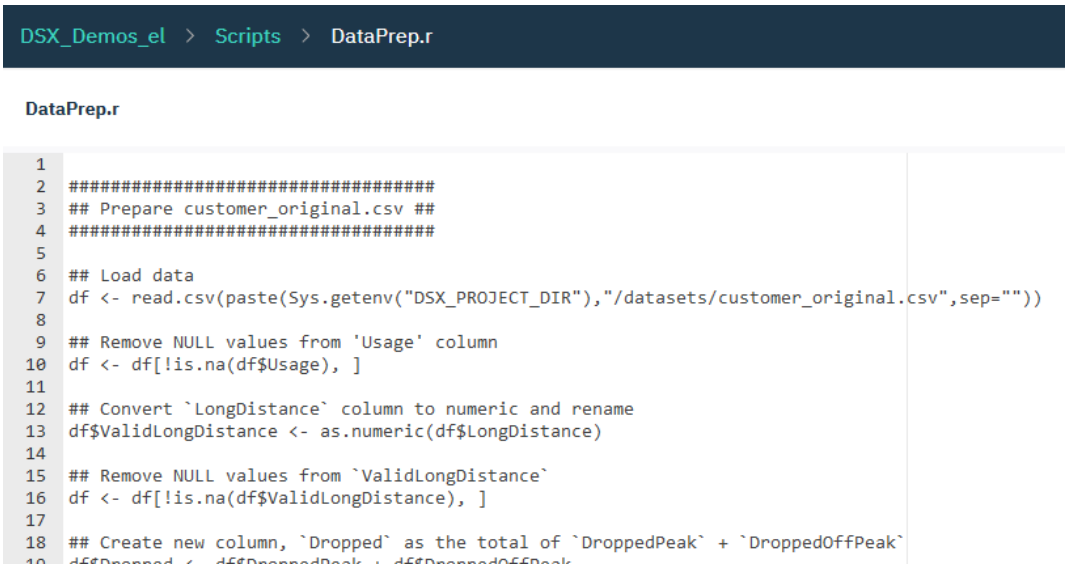


Option 1: Deploy existing data preparation scripts

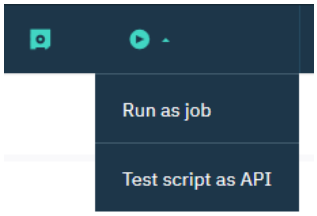
In this section we will walk through deploying an existing data preparation script.

1. In WSL open the *WSL_Demos* project and click on **Scripts**.
2. Click on *DataPrep.r* script. This script performs some data preparation tasks for the *Telco Churn* example that we've been using throughout the workshop.

   The script reads data from a csv file and writes the transformed data to a csv file, *customer_clean_r.csv*.



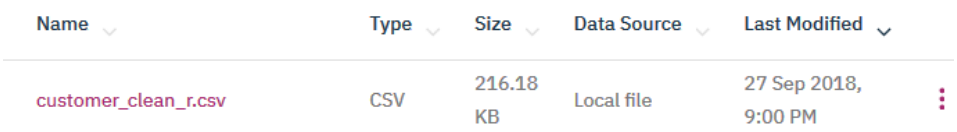
3. Select **Run as a job**.



4. Switch to the **Project -> Jobs** view.

5. Click on the *DataPrepr* job, and verify that the job was successful.

6. Navigate to **Data sets** view and preview the generated file, *customer_clean_r.csv*.



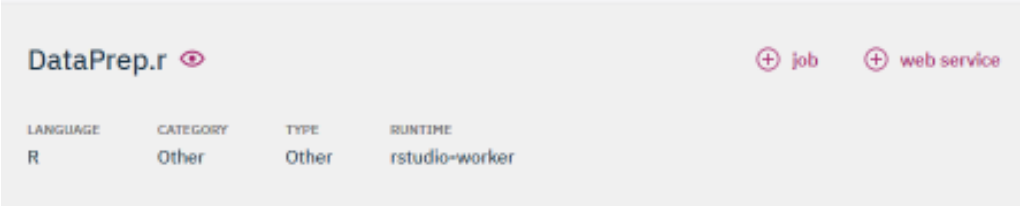
While it's possible to run the data preparation scripts interactively in the development environment (invoke **Run now** from the job definition), most customers will want to automate this process. In order to do that, we need to create a deployment for the script.

Since the script was already included in the *WSL_Demos* project, we can switch to **WML** view to configure deployment.

7. In **WML** click on the project release you created in one of the previous labs.

8. Click on **Assets**, then select **Scripts** from the dropdown.
9. Click on *DataPrep.r*. Then click **job**.

   Note: The Web service option applies to Python and R scripts that have a function that should be invoked "as a service". This function my implement scoring or business rules. The Web service option is typically not used for data preparation.



10. Provide required input fields, and if you wish, configure a schedule.
    - **Name**: *dataprep*
    - **Type**: *Script run*

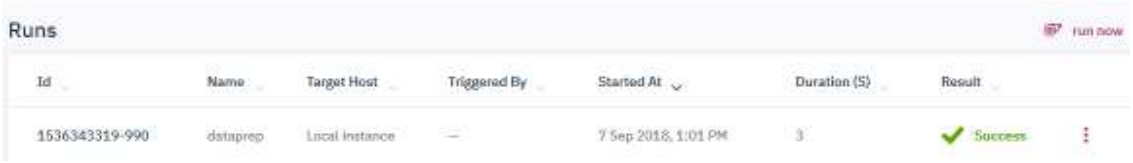
- **Worker**: *Jupyter with Python 2.7…*

Similar to batch and evaluation scripts that we configured in the **Deployment** lab, the R script has a REST API which can be used to invoke the script.

Because the project has already been launched, the new deployments become available immediately.

11. If you wish, you can test the script by invoking it via API. Click on **Deployments** tab, then *dataprep*.
12. Click on the **API** tab, then select **Start** from the dropdown and **Submit**.

13. Click on the **Overview** tab. Here you can see job invocation results.



Now that the data preparation job has been configured another script or notebook can invoke it prior to model building or model scoring.

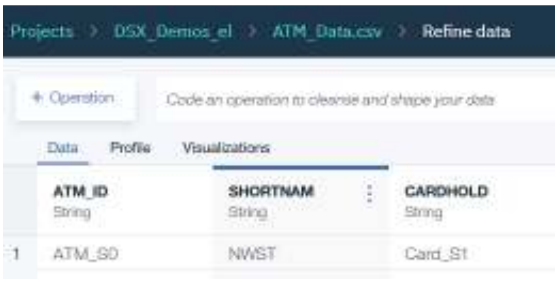
Option 2: Create Data Preparation Scripts with Data Refiner

In this section we will cover the basics of working in **Data Refiner.**

Data Refiner is a data visualization and shaping tool which is included with WSL. **Data Refiner** generates R scripts which can then be deployed similar to a custom script that we deployed in the previous section.

1. In the *WSL_Demos* project click on **Data sets**, then on *ATM_Data.csv*.

Clicking on the data set opens it in the **Data Refiner** tool.



2. Notice that by default the data type for all fields is *String*. Data type conversion is important for both visualization and transformations.

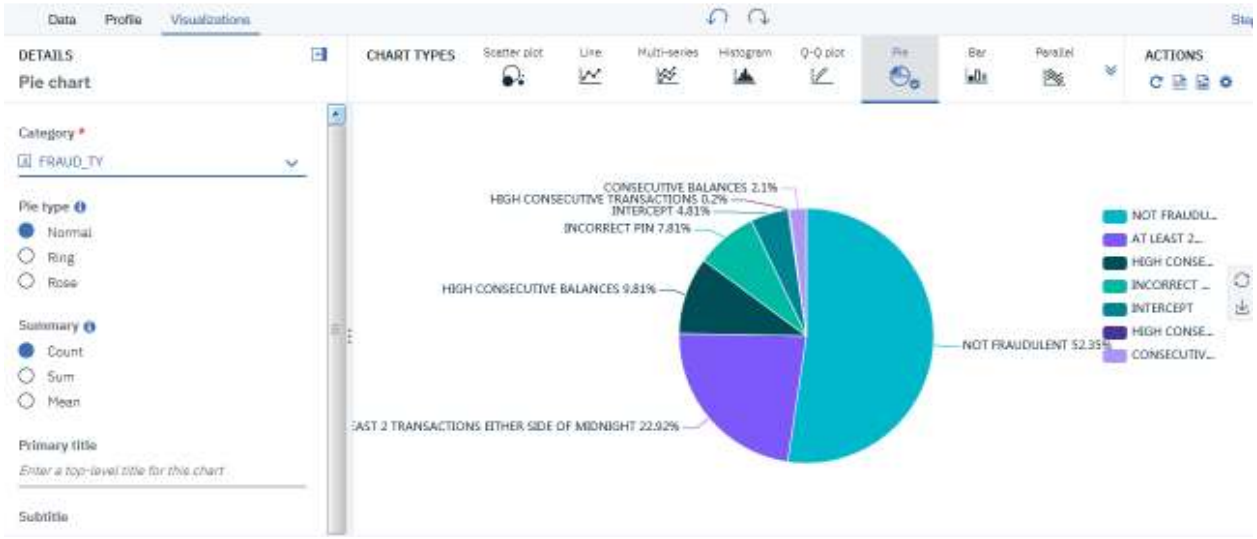
   Click on one of the columns that should be numeric, for example *Fraud*. Expand the menu (vertical ellipses) and select **Convert Column**. Notice that *Integer* type is recommended for this column.

   Click **Integer**.

If you wish, change data types for other numeric columns, for example, *Issuer_I* and *Time of Day*.

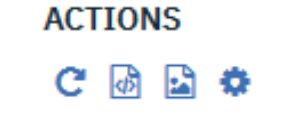
3. Click on **Visualizations** tab and experiment with creating different types of graphs.

   For example, create a pie chart for *FRAUD_TY* (fraud type)

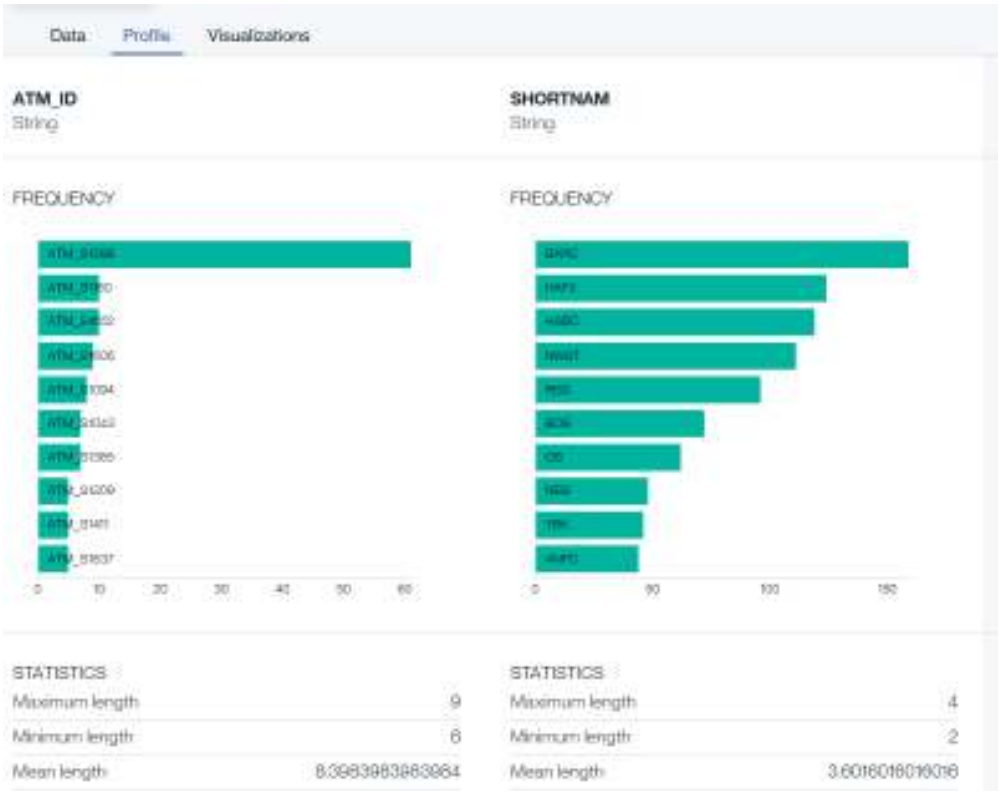


And a histogram for *Time of Day*

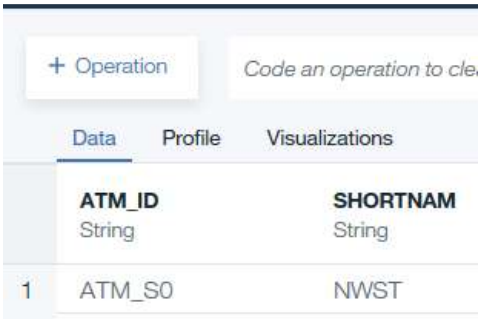
Notice that the **Actions** menu allows you to download charts.



4. Click on the **Profile** tab. Here you can get a quick view of statistics for each column.



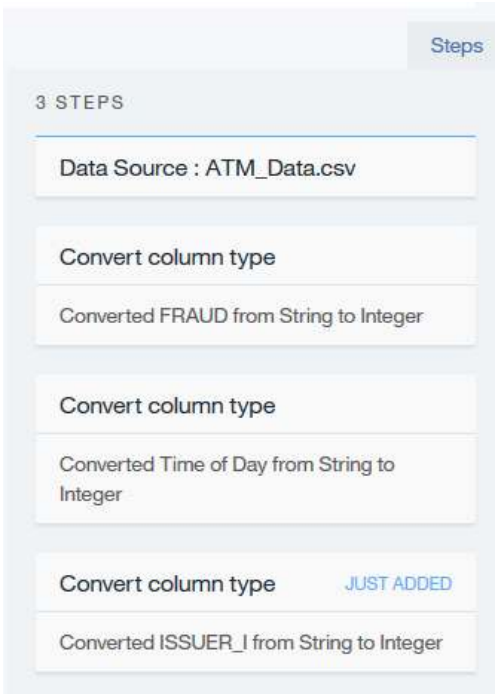
5. Click on the **Data** tab, then the **Operation** button.



6. Review the various operations you can apply to data.

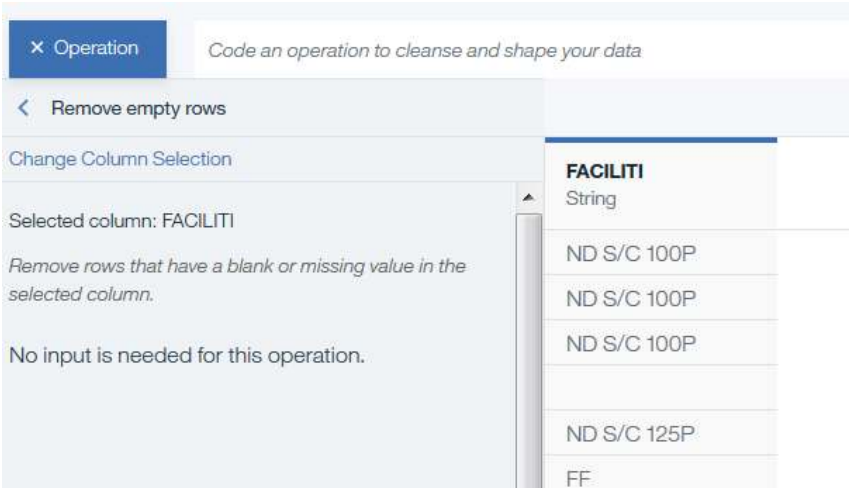
   When we select these operations, they are not applied to the original dataset. In the spreadsheet view we are seeing the "preview" of the operation that will be applied. The operations are applied only when we run a job, which we will do later in the lab.

   If you click on **Steps** in the top right corner, you'll see the transformations that have already been "recorded" and applied in the sample data.

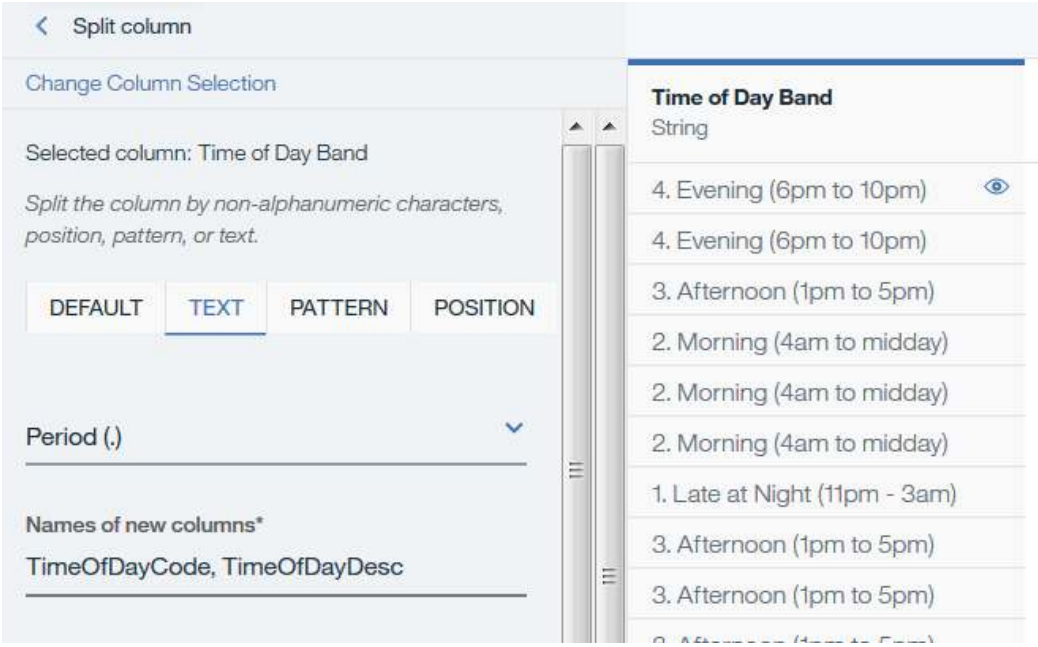


Let's apply a few cleansing and transformation operations before we save and run the data transformation flow.

7. Notice that the column *FACILITI* has a null value. Select the **Remove Empty rows** operation and apply it to this column.



8. Next, split the values in the *Time of Day Band* column. Select **Split**, enter new column names, and select *Period(.)* as the separator on the *TEXT* tab.

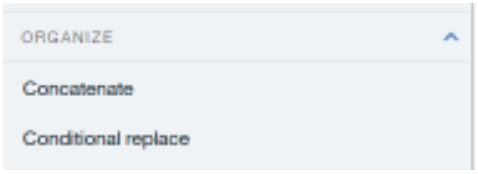


After you apply this transformation, you should see two new columns.

TimeOfDayCode String	TimeOfDayDesc String
4	Evening (6pm to 10pm)
4	Evening (6pm to 10pm)
3	Afternoon (1pm to 5pm)

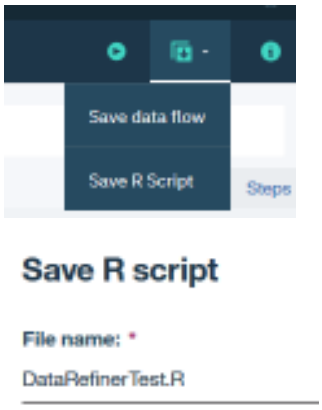
9. If you wish, reclassify the *ATM_POSI* (ATM position) string field to a numeric field using the **Conditional replace** operation.

   Hint: use visualization to look up possible values for ATM position.



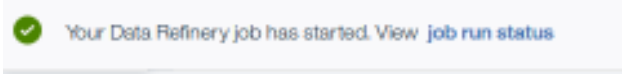
All the steps that we have completed are shown in the **Steps** view.

10. Click **Save R script.** You can use any name for the script, for example, *DataRefinerTest.R.*

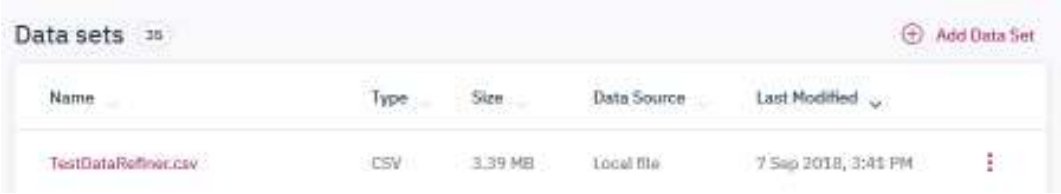


11. Click the arrow icon in the top right corner to run the script as a job. Provide the output file name and run the script.

12. You can check the job status by clicking on the status message or navigating to the **Jobs** view in the project.



13. After the job has finished, you can preview the generated output file in the **Data Sets** view.



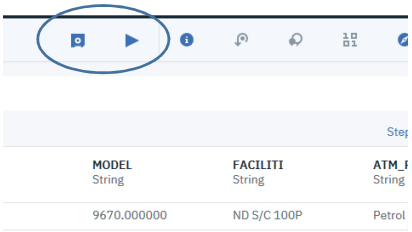
The R script generated by **Data Refiner** can be scheduled for batch scoring similar to R script that we reviewed in the previous section.

Option 2 (optional): Data Refiner – Cloud

In this section we will cover the basics of working in **Data Refiner** in **Watson Studio Cloud.**

1. In the *WSL_Demos* project (in **Watson Studio Local**) click on **Data sets**, then **Export** (from the ellipses menu) for the *ATM_Data.csv*.
2. Import *ATM_Data.csv* into your **Watson Studio Cloud** project.
3. To open the **Data Refiner**, select **Refine** from the **Assets -> Data Assets** menu.

4. Follow the same instructions in the **Watson Studio Local** section to review and modify the data set.
5. The options to *save* and *run* the saved flow are in the top menu bar.

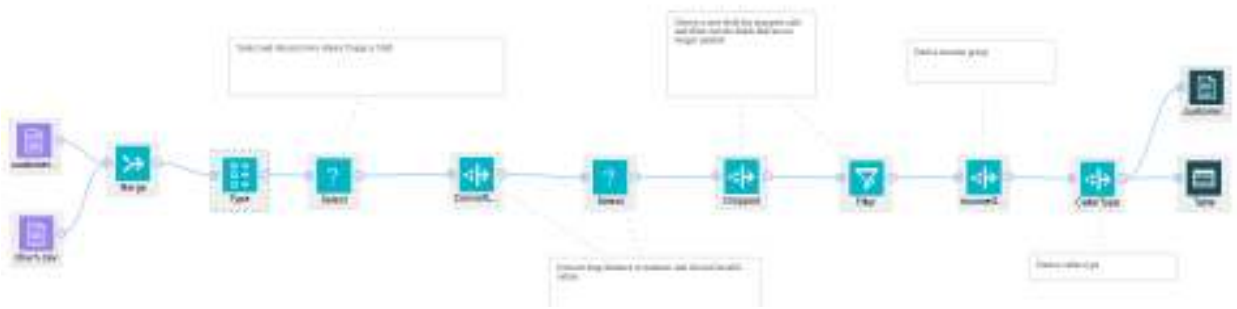


Option 3: Implement Data Preparation in Modeler

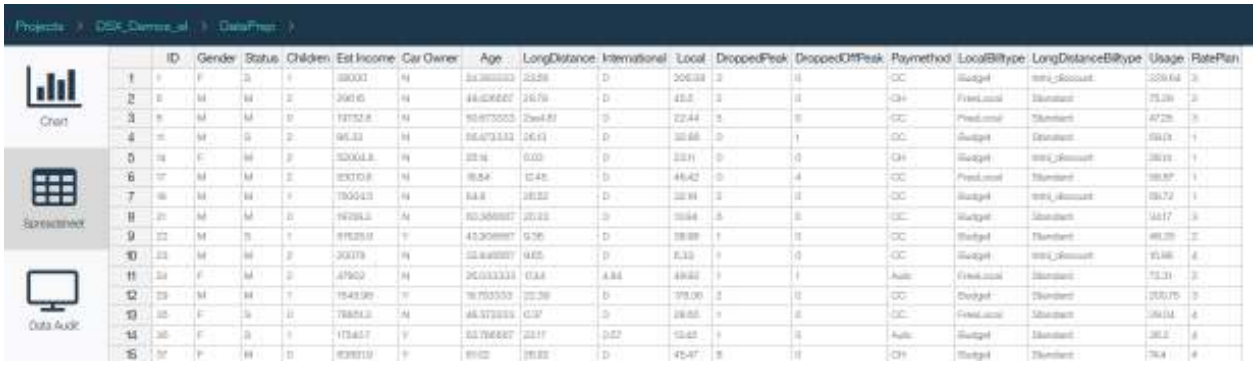
In this section we will review a pre-implemented data preparation flow and deploy it for batch scoring. **SPSS Modeler** is a comprehensive data mining workbench, and data preparation is just a subset of Modeler capabilities. More information about Modeler features is available in the **SPSS Modeler in WSL** lab.

1. In the **Project** view click on **Flows**, then open **DataPrep**.flow.

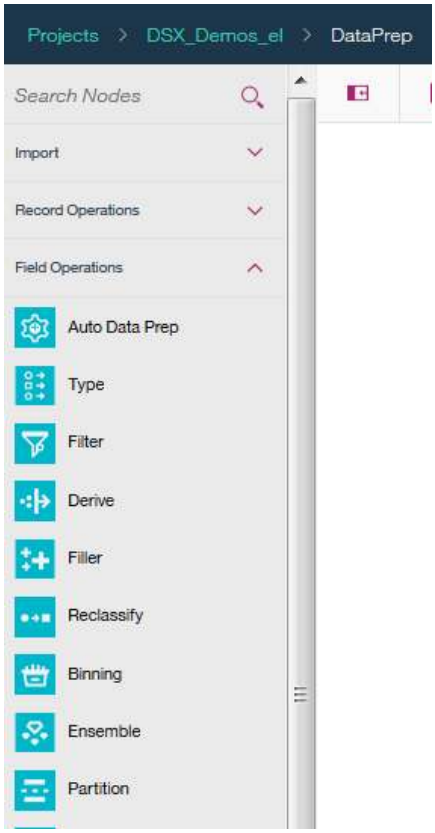
   Each Modeler flow starts with a data source. In our example the data sources are csv files which contains demographic/usage and churn information for telco customers.



2. You can preview the data sources by right clicking on the data source (purple icons) and selecting **View Data.**



3. Navigate back to the flow and click on the arrow icon to bring up the menu of visual nodes.

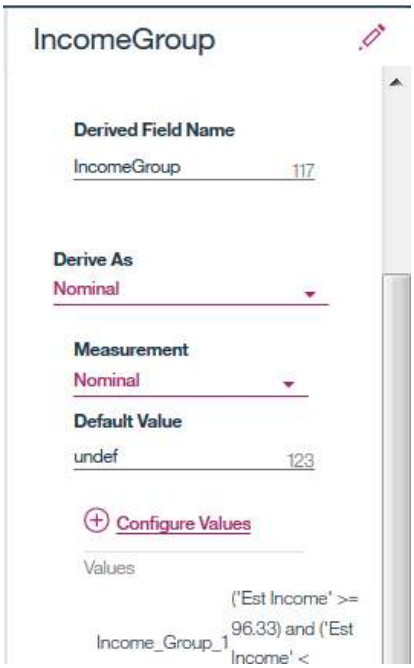


Most nodes that are used for data preparation are organized in the **Record Operators** and **Field Operators** tab. Notice that the flow uses several nodes from these steps.

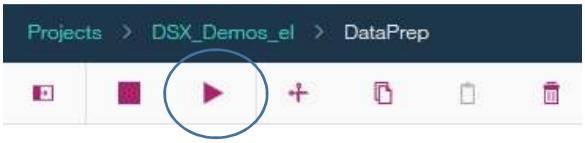
The flow performs the following steps:

- Reads data from data sources
- Merges customer demographic/usage and churn data
- Automatically determines data types (the **Type** node)
- Removes null values from field *Usage*
- Converts *Long_Distance* field to integer
- Discards invalid *Long_Distance* fileds
- Derives a new field, Dropped, which is sum of two other fields
- Filters out the variables that are no longer needed
- Derives *income group* and *caller type*
- Writes results to *spss_dataprep.csv* file and a table (visual output at runtime)

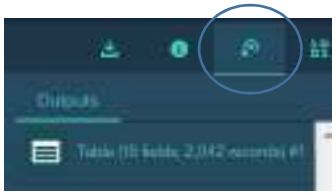
If you would like to see how each node is configured, you can double click on the node and review the values in the **Settings** panel on the right side of the canvas.

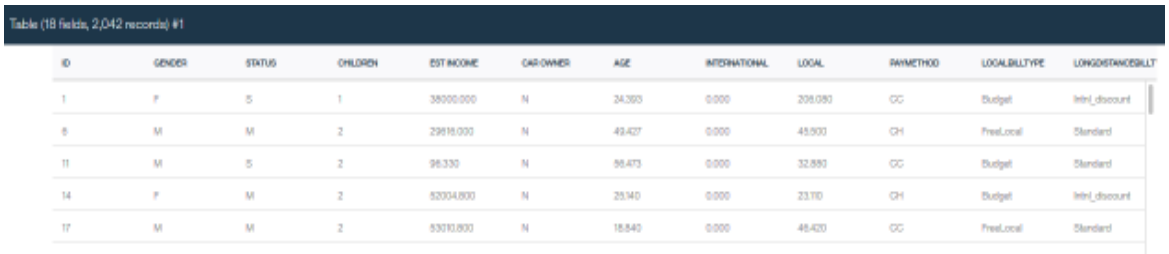


4. Run the stream by clicking the **run** icon.



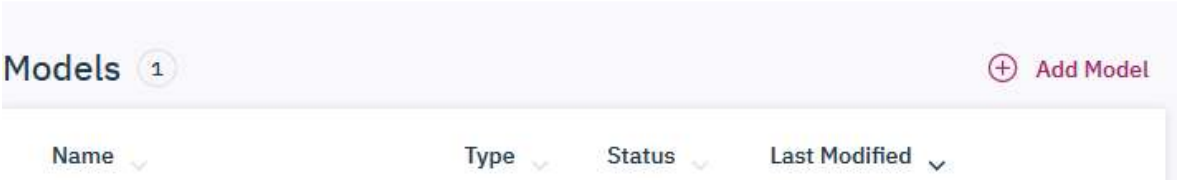
Click on the **Output** icon, then double click on the **Table**. The **Table** shows the same data that was written to the *spss_dataprep.csv* file. If you wish, you can preview the csv file in the project **Data Sets** view.





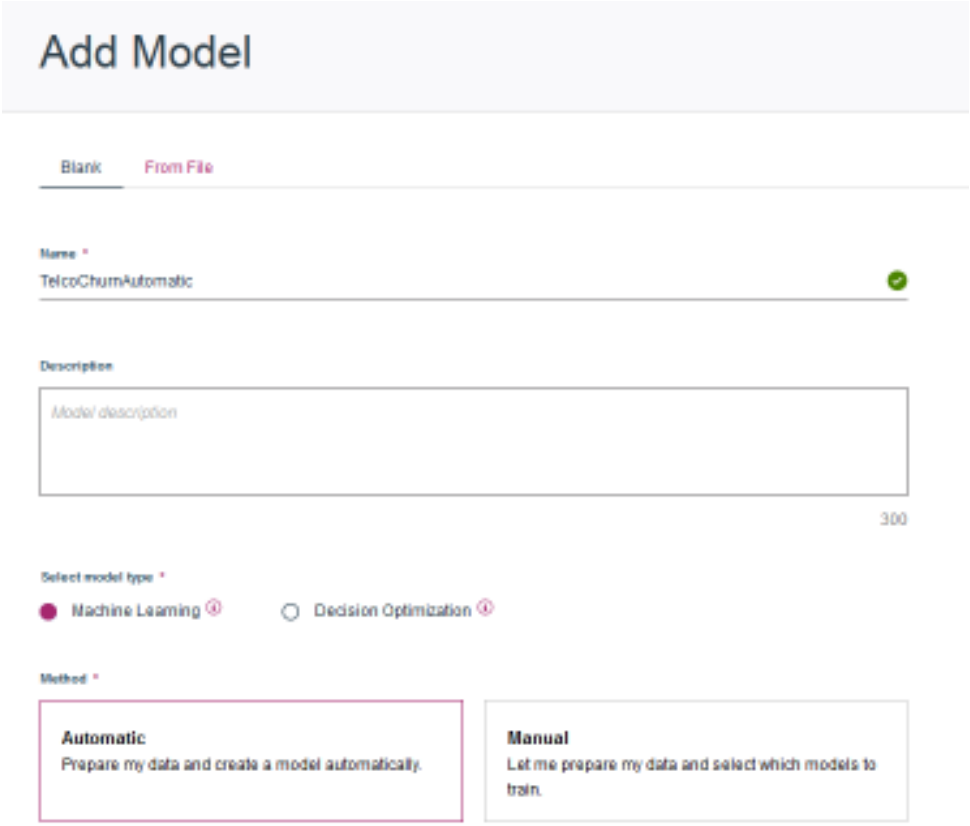
Next, we will build a *SparkML* model using a model wizard. We will use the transformed data set to train the model.

5. In the **Project** view select **Models** and click **Add Model**.



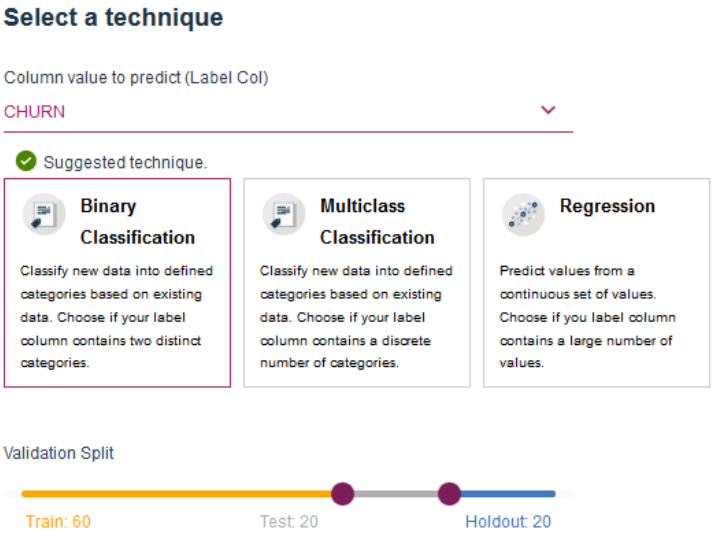
6. Provide the following input:
   - **Name**: *TelcoChurnAutomatic*
   - **Type**: *Machine Learning*
   - **Method**: *Automatic*

   Click **Create**.

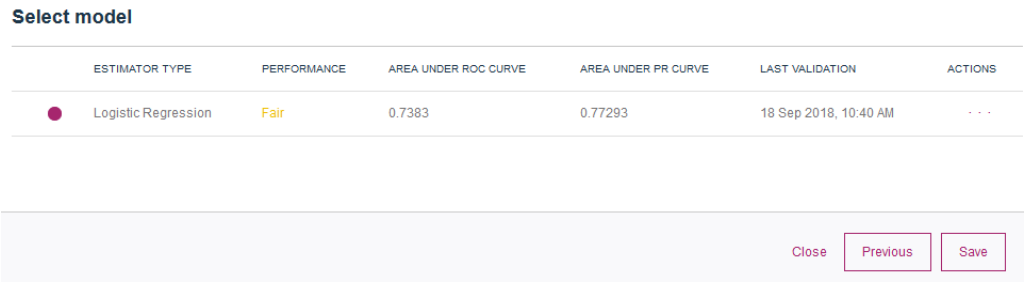


7. Select data set *spss_dataprep.csv*. Click **Next**.

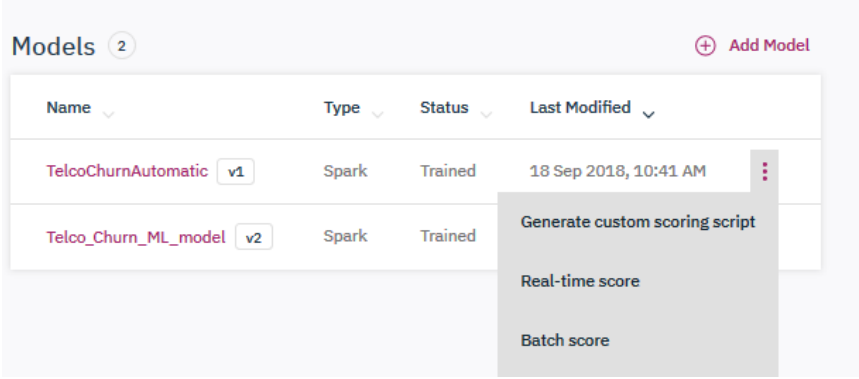
8. Select label column **CHURN**. By default *Binary Classification* model algorithm is selected. Click **Next**.



9. Click **Save** to save the generated model.

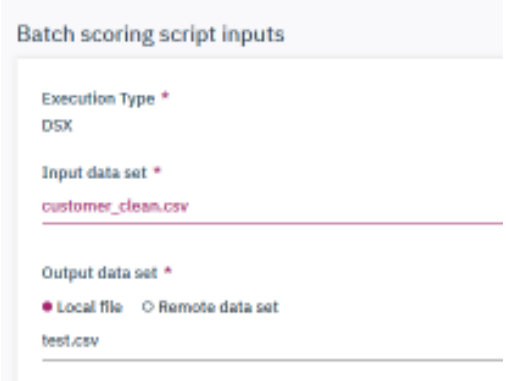


10. We need to configure batch scoring for this model in the development environment because we will need to deploy the batch scoring script.
    - Select **Batch Score** from the model menu

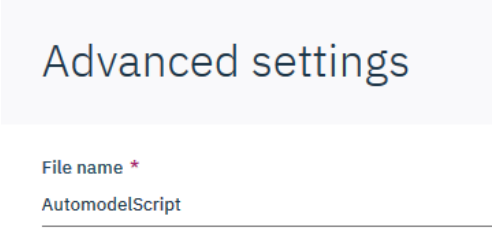


- Select data source *customer_clean.csv* and name the output file *test.csv*

  Note: in a production deployment customer_clean.csv is the file that should be created by the same data flow as was used for model building.



- Click on **Advanced Settings** and name the script *AutomodelScript*

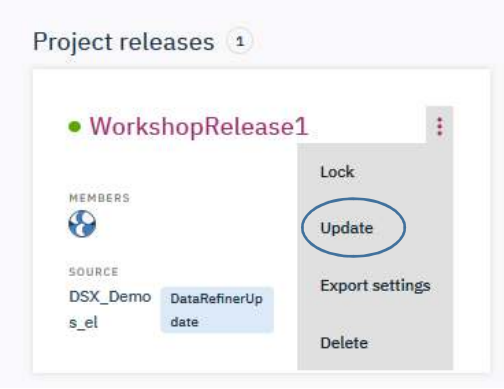


- Click **Save** on **Advanced Setting** page, then **Generate Script**, and **Run**.

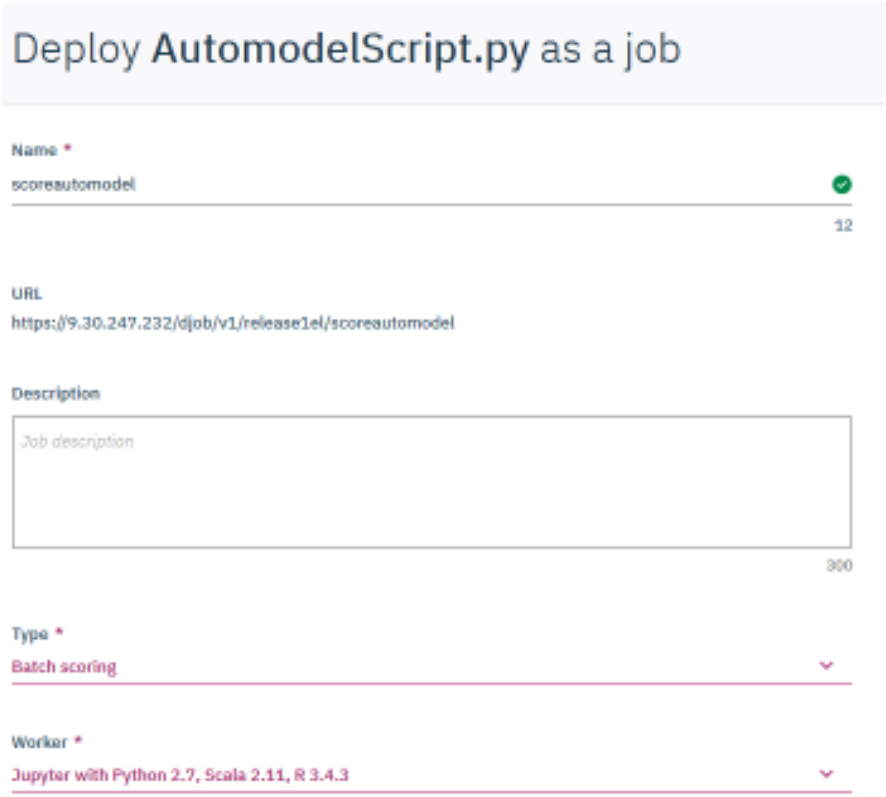
11. Next, we will deploy the model and SPSS flow as batch jobs, and invoke them via a REST API from a notebook.
12. Complete the same steps as you did in the **Deployment** lab to deploy the model as and the flow as the batch jobs.

    Hints:

    - **Commit** and **push** the project to the repository
    - In **Deployment Manager** use the update function in the **Project Release** to pull in the changes



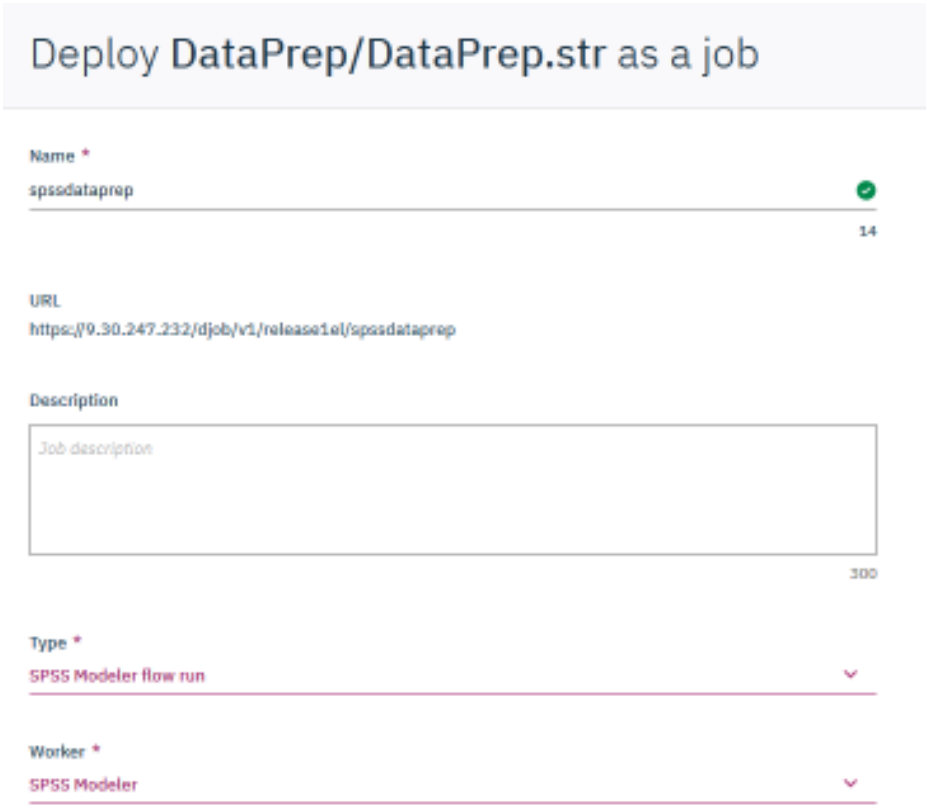
- Create a batch job for *AutomodelScript.py*. Name the deployment *scoreautomodel*. If you use another name, take a note of it. Make sure to select *Batch Scoring* as **Type** and *Jupyter with Python 2.7…* as **Worker**.



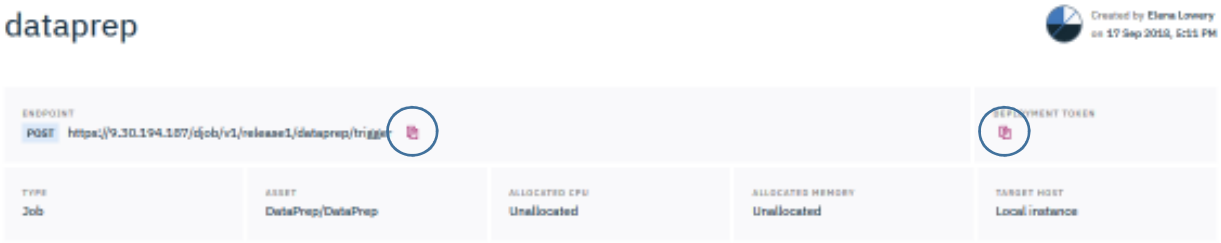
- From the job deployment page, copy the **REST URL** and the **Token** to the notepad. When you click on the copy icon, while the token is not displayed, it's copied to the clipboard.

- When configuring the batch job for the Modeler flow, make sure to select *SPSS Modeler Flow Run* as **Type** and **SPSS Modeler** as **Worker**.

  Take a note of the **route** that you provide, or use "*spssdataprep*".



- From the batch job deployment page, copy the **REST URL** and the **Token** to the notepad. When you click on the copy icon, while the token is not displayed, it's copied to the clipboard.



Now that we have configured the data preparation job and the model, we can invoke them from a notebook.

Invoking the data preparation job and scoring from the notebook is just one example of how to combine data preparation and scoring. A script or an external process can also be used.

13. From the **Projects** view open the *DataPrepAndScoring* notebook, make the required changes (documented in the notebook) and run it.

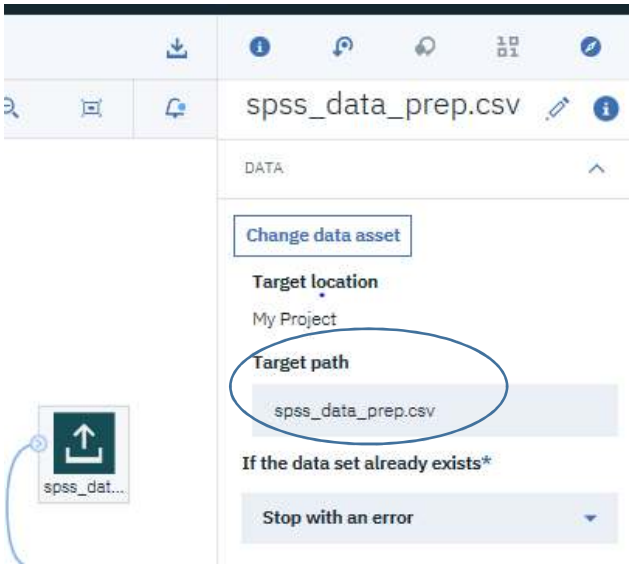
    In a production environment this notebook can be scheduled to run or invoked via a REST API.

Option 3 (optional): Implement Data Preparation in Modeler - Cloud

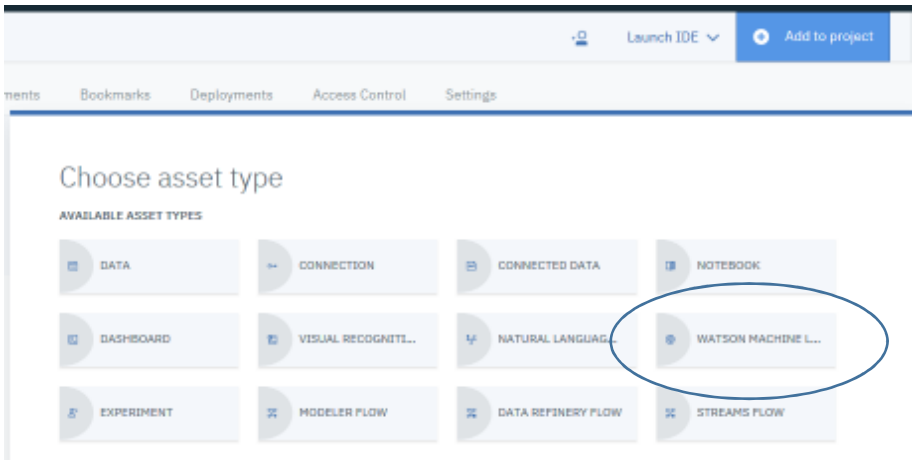
In this section we will review a pre-implemented data preparation SPSS flow in Watson Studio Cloud.

The same Modeler flow can be used in **Watson Studio Cloud**. Complete the following steps to set it up:

1. Import the DataPrep.str Modeler flow into your Watson Studio project from the *SPSS Modeler/streams* directory of the *WSL_Demos* git repo that you downloaded earlier in the lab.
2. Import *customer_original.csv* and *churn.csv* from the SPSS Modeler/data directory of the WSL_Demos git repo.
3. When opening the Modeler flow in Watson Studio, you will be prompted to migrated input data sources – migrate them to the corresponding csv files.
4. Delete the export node and add a new one. Make sure to specify the name of the file in the **Target path** field (in our example *spss_data_prep.csv*).



5. The process of creating a model with the WML wizard is similar to WSL. To start the wizard, click **Add to project**, then select **Watson Machine Learning**.



Summary

In this lab we reviewed how to use csv and database data sources in WSL.

We also reviewed different options for implementing data preparation in WSL, which include:

- Python or R Scripts (imported or created in WSL)
- Data Refiner (generates an R script)
- SPSS Modeler

Since every deployed WSL asset has a REST API, combining data preparation and other tasks (model building, model scoring) can be easily done by doing REST calls in a script, a notebook, or an external application.