For creating the pipeline follow the below mentioned instructions:

1. Go to the **Assignment/01\_data\_pipeline**/scripts folder.

2. First we will break down **all the data cleaning tasks into functions in the utils.py file** you have been provided with functions level breakdown of all the tasks. You can do that by **simply taking appropriate code from the data cleaning notebook named “data\_cleaning\_template.ipynb”** that has been provided in the “Assignment/01\_data\_pipeline/notebooks” folder.

3.Recall that while reducing the level of “first\_platform\_c”, “first\_utm\_medium\_c” and “first\_utm\_source\_c” we mapped the bottom 10 percentile of these columns to ‘others’. In our notebook we had created a list of significant levels of these 3 columns. We will store this list in the “significant\_categorical\_level.py” file here. **Now import this in utils.py and use the variables instead of hardcoded lists.**

4. Once you have converted the code into the respective functions. **You will modify the function to read their input data and write their output to the database present in file 'utils\_output.db'.** The function will be executed in the following order so make sure to read the input for the functions and write the outputs of the function to the above db appropriately. Please ensure that you follow the naming convention specified in the docstring of the function, while naming the tables to which your function will read and write the data from.

**The order of the functions is:**

build\_dbs -> load\_data\_into\_db -> map\_city\_tier -> map\_categorical\_vars -> interactions\_mapping.

NOTE:  **‘utils\_output.db’** will be created once you execute the build\_dbs function.This **db should be created in /Assignment/01\_data\_pipeline/scripts folder**

5. Once you have modified the functions, create a test notebook in the same folder and import utils file in it and run all the functions in the proper order mentioned above. This will help you **debug your code and create the ‘utils\_output.db’ database** in the specified path.

6. Once you have ensured that the functions in utils.py file are working as intended, **store all the constant values in constants.py** and repeat step 4 to ensure that the code is working properly. Some of the variables’ name are mentioned in the constants file which you can use. You can also create your own additional variable if required.

NOTE: utils.py and constants.py must be present in the same directory always

7. Once you have ensured that the code is working properly, **create a folder named 'unit\_test' and copy paste all the files except for the data folder** in the “/Assignment/01\_data\_pipeline/scripts” folder in it. Please do not delete the files in the “/Assignment/01\_data\_pipeline/scripts” rather just copy-paste it to the “unit\_test” folder.

NOTE: The 'leadscoring\_test.csv' present in the scripts folder has a sample input which will be given to the “load\_data\_into\_db” function. Since other functions will work on the output of the previous function input so we do not need input for them. However **we need to compare their output to the expected output so we will need the expected output for all the functions that are present in test\_with\_pytest.py' file.** These outputs are already provided to you in the “unit\_test\_cases.db” file. The name ot the table for each of.

7.Now go to the 'unit\_test' folder and open the **'test\_with\_pytest.py' file. Here you will write unit test cases** to verify that the functions are actually giving their intended outputs. For this you will use write tests that check that the output for all the functions **match against** the expected output provided in **'unit\_test\_cases.db'** . The sample output for each of the function is provided in the following

load\_data\_into\_db : “loaded\_data\_test\_case”.

map\_city\_tier : “city\_tier\_mapped\_test\_case”

map\_categorical\_vars : “categorical\_variables\_mapped\_test\_case”

Interactions\_mapping : “interactions\_mapped\_test\_case”

8.Once all the test cases are passed go back to the 'scripts' folder and **create a folder named 'mapping'. Save all the mapping files namely “ city\_tier\_mapping.py”, interaction\_mapping.csv, significant\_categorical\_level.py in the ‘mapping’ folder**.

9.Now open **'data\_validation\_check.py' and write the mentioned code there**. The function mentioned in the 'data\_validation\_check.py' will run in the order mentioned below. Ensure that you read and write from the appropriate table from the db for these 2 functions in 'data\_validation\_check.py'.

build\_dbs -> raw\_data\_schema\_check -> load\_data\_into\_db -> map\_city\_tier -> map\_categorical\_vars -> interactions\_mapping -> model\_input\_schema\_check

NOTE: Note that there will not be any change in the functions present in utils.py as for the function present in “data\_validation\_checks.py” we are simply reading the data from the appropriate source and **checking if it is the same as the schema present in schema.py file**.

10. Now in same **test notebook ensure that these function are working properly**

11.Now **create a pipeline using airflow in the 'lead\_scoring\_data\_pipeline.py'**. Follow the instructions present in the file for creating the dag for airflow.

12.Now go to the '~/airflow/dags/' folder and create a folder named 'Lead\_scoring\_data\_pipeline' in it.

**The following file should be present in Lead\_scoring\_data\_pipeline**

├── data

│ ├── data/leadscoring.csv

├── mapping

│ ├── mapping/city\_tier\_mapping.py

│ ├── mapping/interaction\_mapping.csv

│ └── mapping/significant\_categorical\_level.py

├── schema.py

├── data\_validation\_checks.py

├── lead\_scoring\_data\_pipeline.py

├── constants.py

└── utils.py

13.After copying all the necessary files and folder from 'scripts' folder to 'Lead\_scoring\_data\_pipeline' and modify your constants, utils and lead\_scoring\_data\_pipeline.py as the paths have been changed when you have changed the directory. **Make changes in the file path names in constants.py file.**

Notice that we did not copy paste “utils\_output.db”, this is because we plan to create a new db named “lead\_scoring\_data\_cleaning.db”. **So you have to change the name of db to “lead\_scoring\_data\_cleaning.db” in the constants.py file and finally make changes in the import statements in utils.py and “lead\_scoring\_data\_pipeline.py” file.**

14.With this you should be able to run the data pipeline. Follow the instructions mentioned in the Airflow module to run the data pipeline and **take the screenshot of Airflow UI for submission.**