**TASK 1 – Obtain and Clean Data**

Clone the Repository

1. Install the project dependencies according to the requirements.txt file
2. Execute the **getfiles.py** to fetch/download the datafiles to the infiles directory path. This is the location we the application expects to load files from. Files are expected in .xlsx formats. A scheduled job will run daily and would process any files in the directory.
3. Log on to your demo environment, then execute the database scripts in the directory, to prepare the database and warehouse for the rest of the project.
4. When steps 1, 2 and 3 are completed. You can open the **mergecleanfile.py.** Update/Populate your DB credentials and connection properties (MSSQL Database was used for this project).

The python file, merges all the excel files in the directory to a single file, loads it into a dataframe, cleanses and separates the file into different components and model as required by the project description. A breakdown of the various functions and code is also presented in a jupyter notebook *(mergefilesup.ipynb*)

**TASK 2 - Create a Data Model**

The data files downloaded are in an unnormalized state. To normalize the data, I applied the 1st to 3rd data normalization rules, which involves removing repeating groups, creating tables for each sets of related data, creating primary keys, ensuring no partial dependencies on the keys and also no transitive dependencies on non-prime attributes.

I was able to create the below data model, mapping entities, their attributes and relationships. The excel link below opens up the ERD of the tables.

**Reasons behind the choices.**

To satisfy **First normal form**, each column of a table must have a single value. Columns which contain sets of values or nested records are not allowed – The dataset is already in this form.

The Transaction table has one candidate key Batch-ID (which is therefore the primary key). To conform to **Second Normal Form** and remove duplicities, every non candidate-key attribute must depend on the whole candidate key, not just part of it. – All the columns on the dataset depend on the candidate key Batch ID that uniquely identifies the transaction.

To satisfy the **Third Normal Form,** there shall be no transitive dependences on non-prime attributes, in this case the table or data violates this form {Merchant Type, Merchant Description} depend on the *Merchant Name*, {GL Description} on GL Account, {Cost Center Description} *on Cost Center No.* We take care of this by placing the sets in their own respective tables and have a foreign key relationship on them.

**Graphical user interface, application

Description automatically generated**

**Task 3 - Data Pipeline and Warehouse**

Using a data ETL/pipeline and Orchestrator tool – Airflow and SSIS, I defined two (2) tasks with different intervals (unfortunately I could not meet the deadline to complete the design”

1. ***fetchClean*** – will execute different functions to download the files from the online repository as in the demo case or otherwise, extract and move the files to the ***/****infiles* directory path, cleanse, save the resulting datafiles to the output directory - */outfiles*. It will also create, update and load accordingly the **pcard** Database Tables in line with the model above. Schedule Interval is **daily** at a set time.
2. ***loadDW*** – will extract and load the data from the various databases into a raw-data environment, stage and transform the data into the dimension and facts table of the pcard Data Warehouse. Schedule Interval is **weekly** at a set time.

To preserve history, we will implement a slowly changing dimension (SCD – Type 2) across the dimensions. Using the DimDivision (Division Dimension) table on the warehouse. The primary key *division\_id* which is also the surrogate key is an identity property which will generate a new value for every row. Let’s imagine we’d like to keep history on the purchases authorized by the division supervisor (represented by the *division\_sup\_id*). Any time a new supervisor is assigned to the division, we don’t update the row, but insert a new record. Using the start\_date and end\_date fields we would track when a record was valid in time. A new surrogate key (division\_id) is generated, but the business key – division\_name remains the same. when a fact table is loaded, a lookup will be done on the DimDivision table, depending on the flag column (*IsCurrent*) the current attributes of the row are returned. Hence, we are able to track data lineage and history. For this data warehouse, the schema we have implemented is a STAR – Schema to keep the structure simple and flexible.

Graphical user interface, diagram

Description automatically generated

**Task 4 – Visualization**

For the visualizations I created a simple and robust semantic model that supports self-service analytics. When published, users can easily explore and interact with the dataset. The key take-aways can be found on the first tab called Summary and a weekly spend view is also provided where the user can see the spend per division.

The visualization tells a visual story across five (5) reports. Each of the reports makes it easy to analyze, slice, drill down and up across multiple indices. A quick breakdown of the visualizations:

1. Page1 - a Summary report or a quick overview of the numbers in the dataset,
2. Page2 – is about expenses across different categories – divisions, GLs, Cost Centers, Merchants etc.
3. Page3 - that displays the weekly purchases/expenses by divisions, one can also filter by merchants, by months and other date dimensions
4. Page4 – shows spending across different merchants by year. We can also drill down by divisions, and time
5. Page5 – shows a view over the period of the dataset.