Milestone 2 description

Deadline - 19/11 @11.59pm.

The objective of this milestone is to package your milestone 1 code in a docker image that could be run anywhere. In addition, you will load your cleaned and prepared dataset as well as your lookup table into a PostgreSQL database which would act as your data warehouse.

Deliverables

Before diving into the requirements it might help to envision what the deliverables would be. It should simply be a folder containing the required files/folders to build both images and run them(similar to lab 5). At the root of the folder there should be a `docker-compose.yaml` file such that by running `docker-compose up` the pipeline would be run (cleaning and preparing and then loading to PostgreSQL database). The root folder that you will create MUST be named `m2_{id}_{major}_{tutorialno}_{month}_{year}.`
Replace id,major and tut no. with your id,etc. replace month and year with the month year of your dataset. i.e if you are a COMM major that worked on the july dataset of 2017 your folder name must then be `m2_49-1234_COMM_p2_7_2017`.
The folder name must be as instructed, not doing so will result in marks deduction.

Requirements:

For this milestone your dependencies so far would be python(and the necessary packages) and PostgreSQL. Each of these dependencies should be a separate image that would run under the same network(docker compose).

1. Building the python image:

- You must use the python:3.11 image as your base image.
- Convert your notebook into a python script. Why? because the code you have written to clean the dataset must be executed inside the container and from the terminal. The format (ipynb/python notebook) is not an executable file and cannot be run from the terminal and thus you must convert your code into a python script that can be executed from the terminal.
- Important notes regarding converting from ipynb to py.

    - You should have at least two python scripts. One for listing the functions you created in M1 and the other would be your main script were you call these functions.
    - In your script you only need the functions/code that changes the dataset(I.e. cleaning,encoding,disc.,adding features). You DO NOT need to include code you have written to analyse the dataset to reach your decisions (i.e. you do not need the basic EDA, 5 questions,any visualizations or debugging code).
    - In your main.py, you should first check if the cleaned dataset exists, if it does then you do not need to call the functions (similar to what you did with when you extracted additional data from an API). Check the method from the OS package `os.path.exists(path)`
- Add the functions required to connect to the database and upload your cleaned dataframe AND the lookup table into the database(you should not re-upload the dataframe if the table already exists, check 'if_exists option' *here*)
    - table names MUST be `green_taxi_month_year` and `lookup_green_taxi_month_year`. replace replace month and year with your corresponding dataset month and year
- All in all your python script should check if the cleaned csv exist,
    - if it doesn't, read the csv file, prepare the dataset,save to new csv,connect to postgres and load the dataset
    - if it does, then you can load into PostgreSQL directly.
- For your python image you should separate your source code and datasets in different directories(check lab 5)
- Add your package dependencies(pandas,etc) in the image.
- DO NOT copy any files from host to container. Instead mount volumes.
    - Important: When mounting volumes make sure to have your relative host path and not the full path. i.e. the path relative to your docker file and to the absolute path (c:/users/...). The reason being that this way the path is agnostic to your host machine and anyone could run the image easily.
- When you run the image it should execute the `main.py` file.

2. PostgreSQL image:

- You must use postgres:13 as your image.
- make sure to mount volumes for the database data and expose the appropriate ports. Your directory at the host where the database data will reside MUST be named `green_taxi_month_year_postgres` replace month and year with your corresponding dataset month and year
- Place your SQL queries in a directory called 'm2_queries' and mount this directory as well into the PostgreSQL container at /var/lib.
    - directory name and path name MUST be as instructed, not doing so will result in marks deduction.

3. SQL queries:
   Write SQL queries to answer the following questions. Place the sql queries in a file called m2_queries.sql under the m2_queries folder.
   1. All trip info(location,tip amount,etc) for the 20 highest trip distances.
   2. What is the average fare amount per payment type.
   3. On average, which city tips the most.
   4. On average, which city tips the least.
   5. What is the most frequent destination on the weekend.
   6. On average, which trip type travels longer distances.
   7. between 4pm and 6pm what is the average fare amount.

Tip: You will need to join the results with your lookup table to understand what the output is. I.e. when you get the result for query 1, payment types will be 1,2,3,etc. That is because the feature is encoded. You can then join with the lookup table to derive the real values.

IMPORTANT : The SQL queries must be your own work and you have to write them without any assistance.

Take screenshots of the output (whether that be from the terminal or Pgadmin) and place them in another directory called queries_screenshots(under your root directory/folder (m2_docker)).

You are more than free to add more containers within your network such as Pgadmin to easily run your sql queries(check lab5 task).

Submission guidelines

Upload your folder as a zip folder in your Milestone 2 drive folder.

EXTREMELY IMPORTANT:

- In your uploaded zip folder DO NOT INCLUDE the following
  - database data info that is mounted on to the container (the `green_taxi_month_year_postgres` folder)
  - the cleaned CSV file
- Again, do NOT include your database data and the cleaned csv file in your uploaded zip folder.

The reason being is so that the pipeline could be run as an initial run (before creating the database and cleaning).