Milestone 4 description

Deadline - Monday , 18th of December @11.59pm.

Overview -

For this milestone you are required to orchestrate the tasks performed in milestone 1 and 2 using Airflow in Docker. For this milestone we will primarily work on the green dataset and pre-process using pandas only for simplicity.

The tasks you have performed in milestone 1 and 2 were as follows.

Read csv(green_taxis) file >> clean and transform >> load to csv(both the cleaned dataset and the lookup table) >> extract additional resources(GPS coordinates) >> integrate with the cleaned dataset and load back to csv >> load both csv files(lookup and cleaned dataset) to postgres database as 2 separate tables.

Important note: table names MUST be as following `M4_green_taxis_month_year` and `lookup_table`. Replace month_year with the the month and year of the dataset.

In addition to orchestrating these tasks in Airflow. You will perform 1 additional task : create a dashboard and present it in a web interface (using dash package in Python).

Therefore, your workflow(DAG) should ideally be as follows. Read, transform and load to csv (t1) >> extract additional resources(t2) >> integrate and load to both postgres and csv >> create dashboard (t4). Your dashboard should preview at least 3 graphs that are properly labeled and represented (You are free to reuse the graphs created in milestone 1).

How to get started -

I STRONGLY advise you work on the trimmed version of your dataset(same month and year) rather than the full dataset as this will make the runtime much lower and consume less resources on your machine.
You can find the trimmed version of the green taxis dataset *here*, make sure you download the same month and year dataset as the one you worked on in milestone 1.

The trimmed version is approximately 20% of the original dataset and thus your script and analyses is still valid and should work fine.

To create your dags, you should use the python scripts you created in milestone 2 and call the appropriate functions for each task in your airflow python script(where you initialise the dag). You will also create a new function for creating a dashboard.

You should have 2 yaml (docker-compose) files for this milestone(1 in each folder). 1 for the PostgreSQL image(that holds the tables in a PostgreSQL database) and 1 for airflow(where you will execute your dag).

Note that you cannot place your PostgreSQL container, which will hold the tables,within the same yaml file of airflow, because airflow already uses a PostgreSQL image internally and it is only reserved for internal airflow services(to hold logs,metadata and other info regarding airflow in a database) (you can but it is somewhat complicated as you will need to further modify the yaml file of Airflow).

For that you will need to connect both yaml files as shown in lab 10. Remember to use named volumes and not binding volumes in your postgres container.

Commands to get started :

- download the image - `docker pull apache/airflow:2.7.3`
- create dir for milestone - `mkdir DE_M4_id_Major_month_year`
- under this dir create the directories needed and follow same hierarchy as lab 10 (refer to lab 10)
- inside the same directory, set airflow uid to be same as localhost id and save this variable in .env file (this file is read when airflow-init runs) : `echo -e "AIRFLOW_UID=$(id -u)" > .env`

Important notes

1. You are more than free to create your DAG as you see fit as long as all the tasks aforementioned have been performed. However, your DAG should orchestrate the tasks and NOT your data flow. In other words, you should keep your data flow encapsulated within a single task and not orchestrate

the data transformation steps. For instance, note that I have placed all of the pre-processing methods/tasks in 1 task and have not separated the tasks (cleaning and transformation) into separate tasks in my DAG as this is simply a flow of data and not tasks. Examples of tasks could be extracting from 3rd party, loading to a database,loading to a cloud service,etc.

2. Remember to use named volumes and not binding volumes in your postgres container.
3. You MUST separate the dag script from your function definitions for the tasks. That is, your code for executing the tasks (functions) must not be included in the same file where you initialise the dag and execute the tasks. Similar to the structure in lab 10. Advisable to separate each task in a separate python script.
4. You must also separate the csv files in a separate folder than the dags folder.
5. Throughout the milestone you might need to reset everything and start fresh, you could use the following command(from where where your airflow's yaml file is) `docker-compose down --volumes --remove-orphans` Note that this removes all volumes and containers created previously by airflow, you can refer to the *doc* for more info. I also strongly suggest you read the documentation in-case you run into any issues.
6. For the dashboards, you will be graded on the complexity of the insights derived and proper representation(appropriate graphs and labels) similar to milestone 1. Therefore, if you have been deducted marks in the 'complexity of the questions' in milestone 1 you should think of more complex insights to derive, otherwise you we will be deducted points again.
7. When creating the dashboard You MUST read the cleaned dataset and not uncleaned. Creating the dashboard from the uncleaned dataset will result in marks deduction.
8. In your tasks make sure to check if it has been performed previously running the code (i.e check if cleaned file exists or data already in db) similar to what you did in m2.

Weight distribution

- Logical flow of tasks (orchestration of tasks and not data) - 25%
- Scripts for the tasks separated from the main dag file, that is the main dag file only initialises the dag and runs the tasks. - 10%

- Dashboard is clean, using proper graphs that is properly labeled to convey the insigh derived. In essence, by just looking at the dashboard I should understand some of the insights you have derived from the dataset. - 20%
- Volumes mounted properly on both the airflow container and PostgreSQL container. - 10%
- Correct ports exposed to the host machine - 5%
- Folder hierarchy - 10%
- Overall correctness, that is when running the dag, cleaned dataset is created and integrated with the GPS coordinates and a dashboard is running and can be viewed through port x(port x being the port mapped to your machine) - 20%

Deliverables and submission guidelines.

Upload a zip folder to your drive folder under Milestone 4 folder. You MUST upload your submission in your drive folder as a zip folder (don't just upload the folder).

The zip folder AND MUST be named `DE_M4_id_Major_month_year`. Major is either NETW,COMM,MET or DMET. Replace id with your id and month_year with the dataset you are working on.

Your zip folder must include 3 folders.

Folder A MUST be named postgres_taxis_datasets must contain the following(but not limited to).

- yaml file for the postgres image where you load your database
- Any additional folders mounted from your host machine to the container must be included in the folder you submit.

Folder B MUST be named airflow_milestone4.

- All files required to run docker-compose up. Such as: additional dockerfile created, requirements.txt (the dependencies for your dag), all folders mounted from your host. Similar to milestone 2 , DO NOT include your cleaned dataset.

Folder C MUST be named 'dashboard_ss'

- This folder should contain screenshot(s) of your dashboard, and the title(centered) of your dashboard must be your name, id and major. For this you will just need to add a simple html tag with title centered at the top.

In addition to the zip folder, upload a short 2-3 minute video of your pipeline being run, starting from running 'docker compose up' for both yaml files upto running the dag and showing the dashboard.

- DO NOT include the video in your zipped folder, simply upload on the drive folder under milestone 4 alongside the zipped folder.
- Make sure the video extension is .mkv so it can be viewed directly from the google drive without having to download it.
- Also advisable to have the dag/container been executed before so that when the task is executed, it should not take anytime. Since in your code, it should automatically check if the file has been created before or if it has been uploaded to the db before(depending on the task) (similar to m2).

IMPORTANT: Not following these instructions and naming restrictions mentioned in the description will result in marks deduction.