Milestone 2 Description

> *Deadline Friday 15/11 @ 11:59PM*

Make sure to read all the description before starting the milestone

Objectives:

1. Package your milestone in a docker image.
2. After cleaning the dataset (from milestone 1) save the output & the lookup table in a Postgres Database and a csv/parquet file.
3. Receive a data stream using Kafka and process the message then save it to the database.

Deliverables

1. Docker compose file `docker-compose.yaml`
2. Dockerfile `Dockerfile`
3. Source code: `cleaning.py` , `main.py` , `db.py` , `consumer.py` (details mentioned later)
4. `requirements.txt`

Note: All these files should reside in a folder for milestone 2, inside the root drive folder created previously in milestone 1.

Requirements

Part 0: functions:

In this section you need to make sure that all the preprocessing steps are converted to functions and called from within a main function called `clean` .

Note: you only need to convert functions that change the dataset, i.e. **YOU DON'T NEED TO** convert the EDA into functions.

Example:

```python
# cleaning.py
# function(s) to handle outliers
def handling_outliers(df):
    .....

# function(s) to impute missing values
def imputation(df):
    .....

# function(s) to apply transformations
def transformation(df):
    .....

def clean(df):
    no_outliers = handling_outliers(df)
    imputed = imputation(no_outliers)
    transformed = transformation(imputed)
    return transformed
```

```
# main.py
# main methods that calls all the other functions
from cleaning import clean

def main(df):
    df = load(...)
    cleaned = clean()
    save(cleaned)
    ...
```

> ***Note*** *that this is just and example, your code can be totally different as long as you have the main cleaning function.*

Part 1: Package your code in a dockerfile to create a docker image

- The base image must be `python:3.11-slim`
- You should have 4 different files
  - `db.py` : python file that handles all database connection and saving.
  - `consumer.py` : a file that initiates the kafka consumer
  - `clean.py` : a file containing all then functions for cleaning
  - `main.py` : the main file that imports functions from the other files and execute the whole process (this is the file to run from inside the dockerfile)
- In the main function you need to check if a cleaned dataset is already saved as a `csv` or `parquet`, you can use `os.path.exists(path)` to check if a file exists or not, if it exists just add it to the database and not run the full cleaning pipeline again.
- Remeber to add 2 tables to the database one for the cleaned dataset and one for the lookuptable, (optionally add other table(s) to save the used means and values), the names of the tables should be the following:
  - fintech: `fintech_data_{MAJOR}_{GROUP}_{ID}_clean`
  - lookup: `lookup_fintech_data_{MAJOR}_{GROUP}_{ID}`

    > *You need to separate the code in folder `src` and the dataset in folder `data`*

Part 2: Create a docker compose file to run the following images

1. Cleaning image (your own)
2. Postgres image : `postgres:13`
3. PgAdmin: `dpage/pgadmin4`
4. Kafka: `confluentinc/cp-kafka:7.4.0`
5. Zookeeper: `confluentinc/cp-zookeeper:7.4.0`

Part 3: Consume & Process the Stream

> *In order to be able to use kafka in python you need to* `pip install kafka-python`

In this part you will consume a stream produced by a docker image that you need to pull using the following command `docker pull mmedhat1910/dew24_streaming_producer`

Afterwards, to be to run the producer you need to do the following:

1. Add the `run_producer.py` file to your directory (this file located in the milestone zip folder) and import `start_producer(...)` and `stop_container(...)` from the file using `from run_producer import start_producer, stop_container`

2. Add a call to this function after the section of the cleaning **before** starting the consumer, then stop the container afterwards. Make sure to pass the correct parameter to the start_producer function like this:

   ```
   # ... cleaning & saving
   id = start_producer(id, kafka_url, topic_name)
   # ... consumer
   stop_container(id)
   ```

   such that

   - id : your id with underscore, eg. 52_XXXX

   - kafka_url: your **external** kafka url, by default `localhost:9092`

   - topic_name: the topic name that you created in the docker compose file

     What this function does in the backgroud is to start a container from the producer image which starts a producer.

Regarding the consumer, it should keep listening until it receives a message with value 'EOF', otherwise it keeps listening forever.

Part 4: PgAdmin

Finally after performing all the tasks, check that everything is saved to the postgres database using pgadmin.

Submission guidelines

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

Files that should be inside the folder:

1. `app` folder containing `src` and `data`, `src` folder containing the scripts mentioned earlier: `main.py`, `consumer.py`, `db.py`, `clean.py`
2. `docker-compose.yaml` file
3. `Dockerfile`
4. `requirements.txt`

EXTREMELY IMPORTANT

In your zip folder **DO NOT INCLUDE** the following:

- Database data/volumes info that is mounted in the volumes folder
- The clean csv folder **AGAIN, DO NOT UPLOAD THOSE FILES PLEASE**

.