ROYAUME DU MAROC
--*-*-*
HAUT COMMISSARIAT AU PLAN
--*-*-*-*-*-*
INSTITUT NATIONAL
DE STATISTIQUE ET D'ECONOMIE APPLIQUEE

INSEA

MOROCCAN BANKS & REVIEWS
- Report -

BY

Mohamed Nijadi

Anass El Basraoui

2022 - 2023

TABLE OF CONTENTS

CHAPTER 1: PROJECT SETUP

Before diving into the detailed implementation of the different steps of this project, we need to define the context first, the tools we are using, and the structure of our work.

CONTEXT

In this project, we are trying to perform an analysis of the user experience within banking agencies in Morocco.

Here is a broad view of the expected work:

- Scraping Google Map's Moroccan bank agencies data related to reviews;
- Cleaning and transforming the data into a meaningful dataset with relevant features;
- Applying NLP models for sentiment analysis on the reviews;
- Modeling and Creating a Data Warehouse for storing this data;
- Feeding Dashboarding software with this data and create a meaningful dashboard;

TOOLS

Here are the different tools and technologies we are using for reaching our end goals:

Apify - a great and easy to use platform for scraping data from various sources including Google Maps, it provides the ability to extract data in different formats, and

also has a wide range of options to choose from (scraping reviews, table management…).

Python & Jupyter Notebooks - All the coding will be done with Python in a Jupyter notebooks environment, so we can better test the different functionalities and look at the different possibilities for cleaning and data transformation.

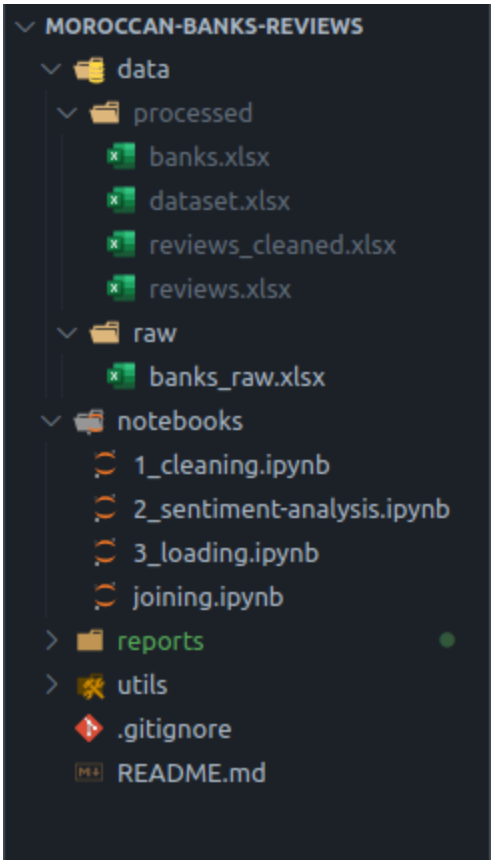
HuggingFace Transformers - HuggingFace is a great platform where useful ML models are deployed and published frequently. In our case, we are going to use a model named nlptown/bert-base-multilingual-uncased-sentiment for performing a sentiment analysis on the comments that come with the reviews.

PostgreSQL - Postgres is a RDBMS for working with transactional data, e.g. the kind of data that comes with data warehouses, that's why it is considered a great tool for our use case.

Power BI (web version) - I was planning on using Tableau but because I switched from Windows to Linux in the middle of working on this project, I had to find a quick alternative and ended up working with Power BI the web version which is very limited unfortunately and didn't come with the variety of options as with Tableau or the desktop version of Power BI.

FILE STRUCTURE

Here's a view of the file structure.

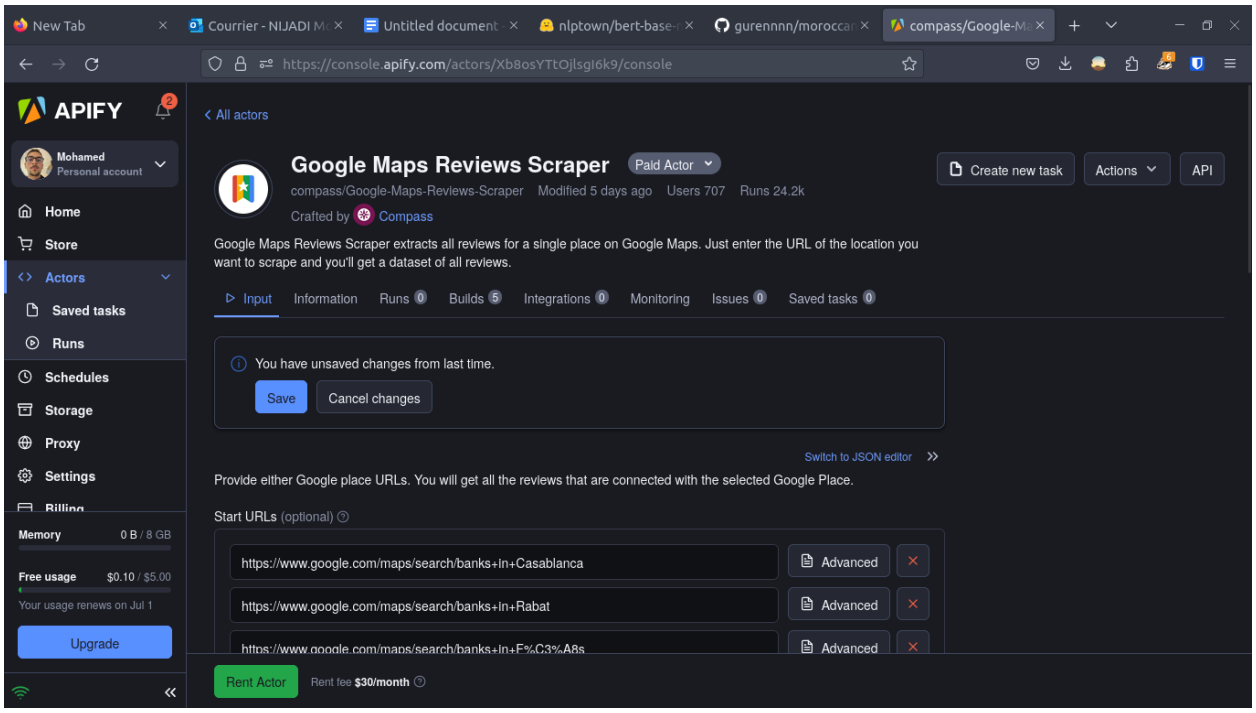


- Data is the folder where both our raw and processed data files are put.
- Notebooks is the folder where we have the different notebooks used for implementing the different tasks.
- Reports is the folder where we exported the dashboard, along with this report.
- Utils is a report where we implemented some functions and utilities we are going to use in our notebooks.

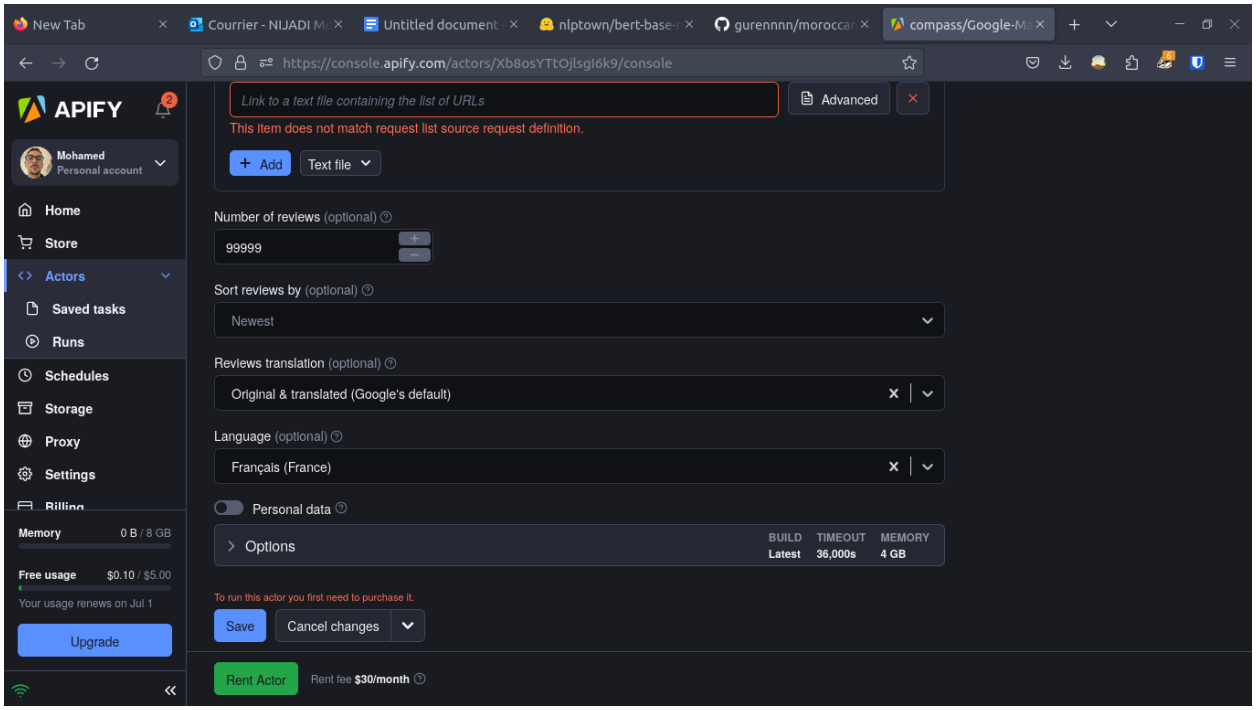
CHAPTER 2 : DATA EXTRACTION AND WEB SCRAPING

The first task is to extract the data, for this we used the apify platform.

APIFY



We are using the Google Maps Reviews Scraper, we provided it with different links to bank agencies (one link for each city),



Here are the next options for extraction, and at the end we exported the results in an Excel format.

FEATURES

Here are the features we picked while extracting the data and exporting in the Excel file:

- Bank and review IDs
- Banks information
    - Title
    - Category
    - City
    - Address
    - GSM
    - Score

    - Latitude / Longitude
- Reviews information
    - Text and translation
    - Review # stars
    - Reviewer

CHAPTER 3 : DATA TRANSFORMATION AND CLEANING

CLEANING

The most important steps we took on our way to cleaning this data were:

- Loading the raw Excel file into the notebook, then we started looking at the data frame, we made sure to drop duplicates and to encode the columns in the right data type.
- While looking at the categories' column, we found that some of the reviews concern some agencies that aren't related to banks, we had to filter them out and keep only those that refer to banks.

```
df['bank_category'].unique()
```

Python

```
array(['Banque', "Service de transfert d'argent",
       "Service de transfert d'argent par mandat", "Banque d'épargne",
       'Poste', 'Institution financière', 'Établissement de crédit',
       'Magasin de téléphonie mobile', 'Conseiller financier',
       'Bibliothèque municipale', 'Bureau de change',
       'Distributeur de billets', nan, 'Attraction touristique',
       'Pizzeria', 'Dépôt-vente'], dtype=object)
```

We see that here, we got some categories that are not really banks, we need to get rid of them.

```
df = df.loc[df['bank_category'].isin(['Banque', "Banque d'épargne", 'Institution financière', 'Établissement de
crédit', np.nan])]
df['bank_category'].unique()
```

Python

```
array(['Banque', "Banque d'épargne", 'Institution financière',
       'Établissement de crédit', nan], dtype=object)
```

- While looking at the bank_title (name of the agency), we saw that the names were very inconsistent, we had to establish a naming convention that we would apply to this column. For this we defined a function for our utils package that takes the title and as a result returns the same title but following the conventions.

```python
# import the function from utils/functions.py
import sys
sys.path.append('../utils')
import functions

# define a list of bank labels
labels = df['bank_title'].unique()

# create a dictionnary that maps each bank label to its corresponding bank
bank_dict = functions.map_labels_to_banks(labels)
```

Python

```python
# create a new column that contains the bank name
df['bank_title'] = df['bank_title'].apply(lambda x: bank_dict[x])

# drop the rows that contain unknown banks
df = df.loc[df['bank_title'] != 'unknown']
```

Python

- Moving on to cleaning the columns that are related to reviews, one of the things we did was to ensure the comments are all in the same language, which in our case is French, and made sure our dates are in the correct format. At the end we split the data frame into two data frames, one is related to banks and the other for reviews, then exported the new data frames in the Excel format and stored them in the processed data folder.

SENTIMENT ANALYSIS

The next transformation step is to take in the reviews' dataset, prepare the comments for analysis, load the model, and perform sentiment analysis on the comments:

```python
from transformers import pipeline
```

```python
sentiment_pipeline = pipeline('sentiment-analysis', model='nlptown/bert-base-multilingual-uncased-sentiment')
```

```python
# testing the model
analyse_text("j'adore ce produit")
```

```
5
```

We see that the output of the model is an int referring the sore of the input text, ranging from 1 being the most Negative and 5 being the most Positive.

```python
# define a function for performing the model on the reviews texts
def analyse_text(text):
  # the case of inexistant comment
  if text == 'no text':
    return 0

  # take the first 512 chars from the comment because it's the maximum size the model can handle
  text = text[:512]

  # apply model pipeline on text
  result = sentiment_pipeline(text)
  return int(result[0]['label'][0])
```

After performing this function on the text column, we now have 2 scores, the score of the comment and the score that the user joined with his comment (number of stars), the final review score was the average of these two values.

```python
def get_review_score(row):
  star1 = row['review_stars']
  star2 = row['stars_from_comment']
  if star1 == 0:
    return star2
  if star2 == 0:
    return star1
  return (star1 + star2) / 2
```

```python
# create review_score column
reviews['review_score'] = reviews.apply(get_review_score, axis=1)
```

Based on this final review score, we defined a function that would classify the review as NEGATIVE, NEUTRAL or POSITIVE.

```python
def get_sentiment(score):
  if score <= 2:
    return 'NEGATIVE'
  elif score >= 4:
    return 'POSITIVE'
  return 'NEUTRAL'
```

Python Python

```python
# create review_sentiment column
reviews['review_sentiment'] = reviews['review_score'].apply(get_sentiment)
```

Python

Finally, we have dropped the temporary created columns, and saved the new data frame as the final Excel file in the processed data folder.

CHAPTER 4 : DATA WAREHOUSE

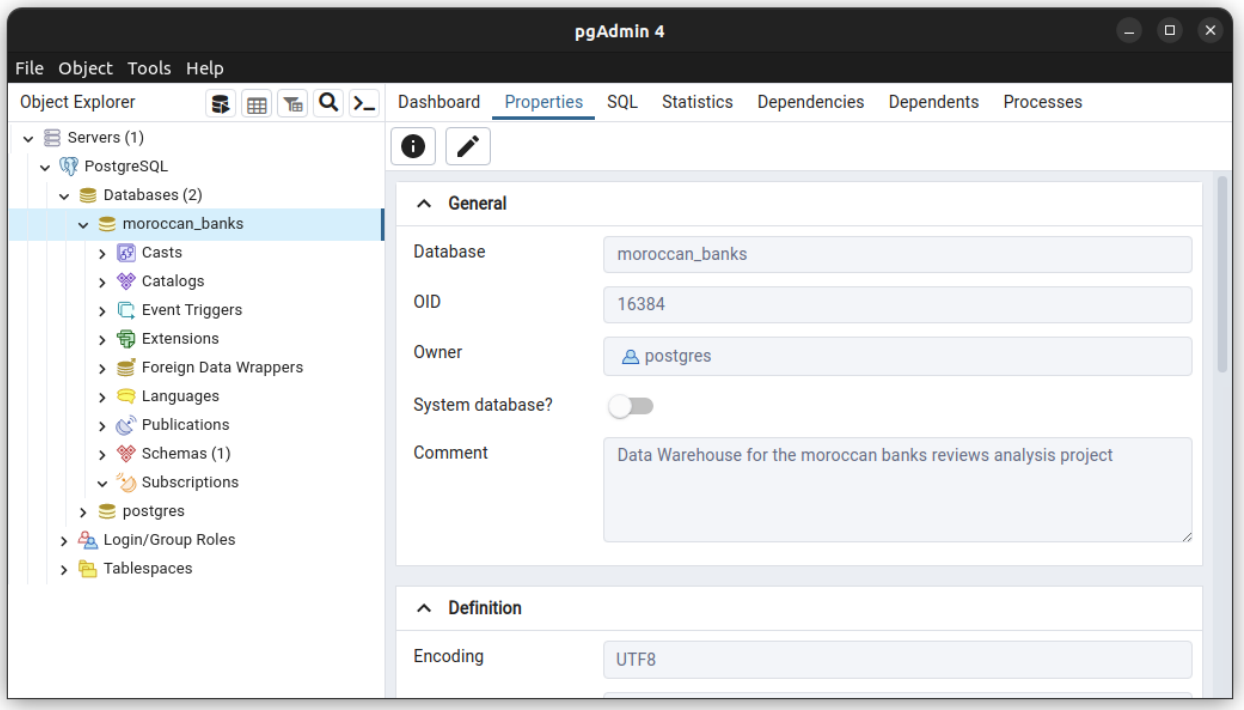
After data transformation, the next step in the ETL process is to load data in the data destination, which is a data warehouse in our case. We need to create this data warehouse and then feeding with data from our transformation phase.

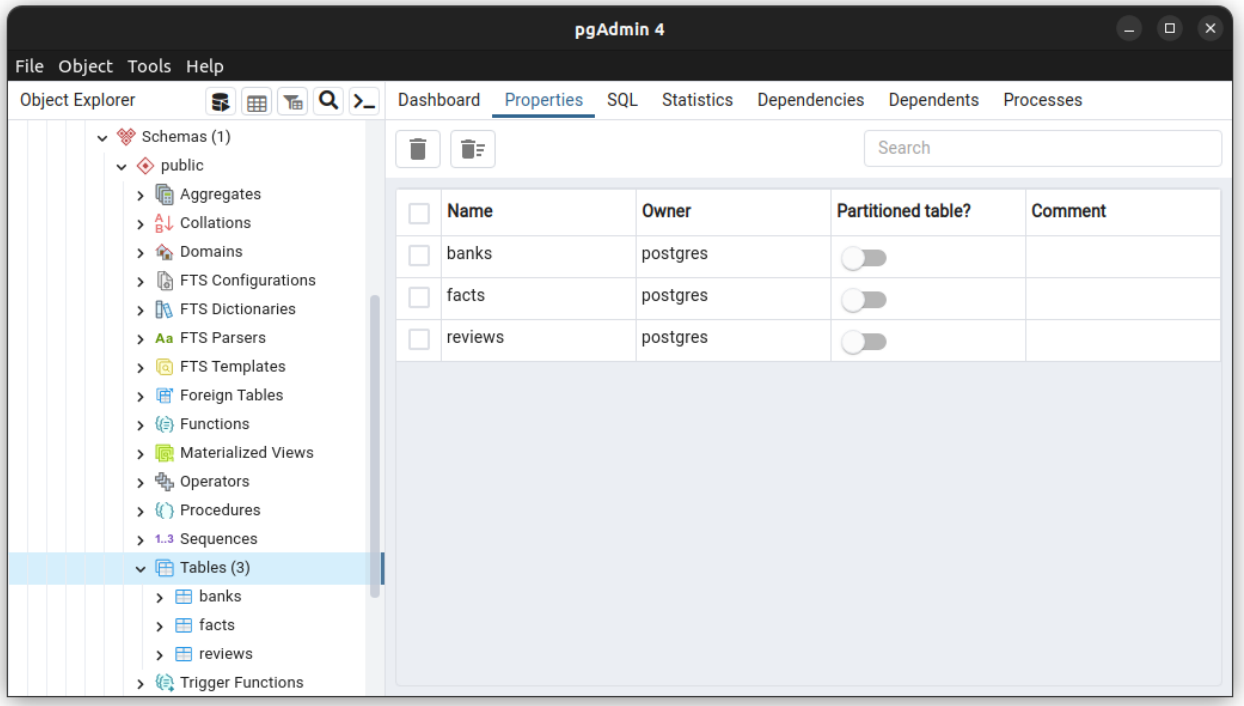
MODELING

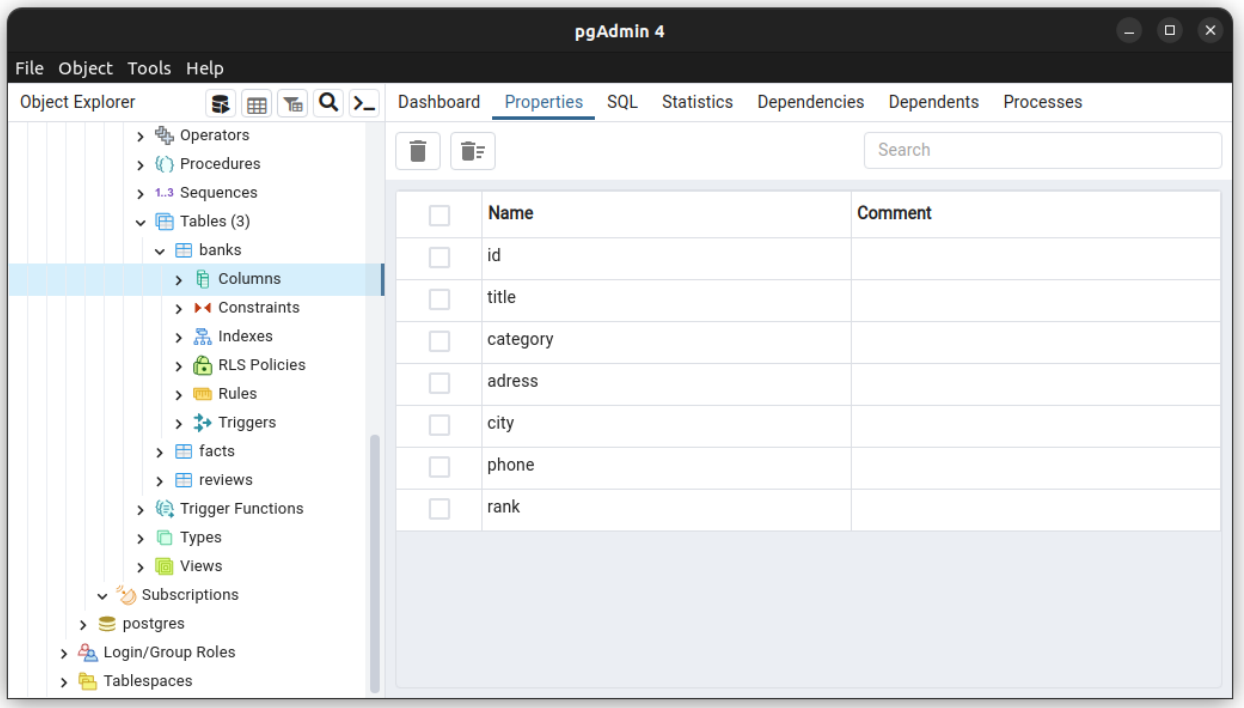
The model of our data warehouse would be a star schema, where the central table is the facts tables, with 2 foreign keys linking to the dimension tables (reviews and banks), date, and facts which are the bank score, review stars, bank reviews count, bank score, bank latitude and longitude. The reviews' table holds the ID of the review, the text and sentiment, while the banks' table holds the bank title, city, address, phone, etc.

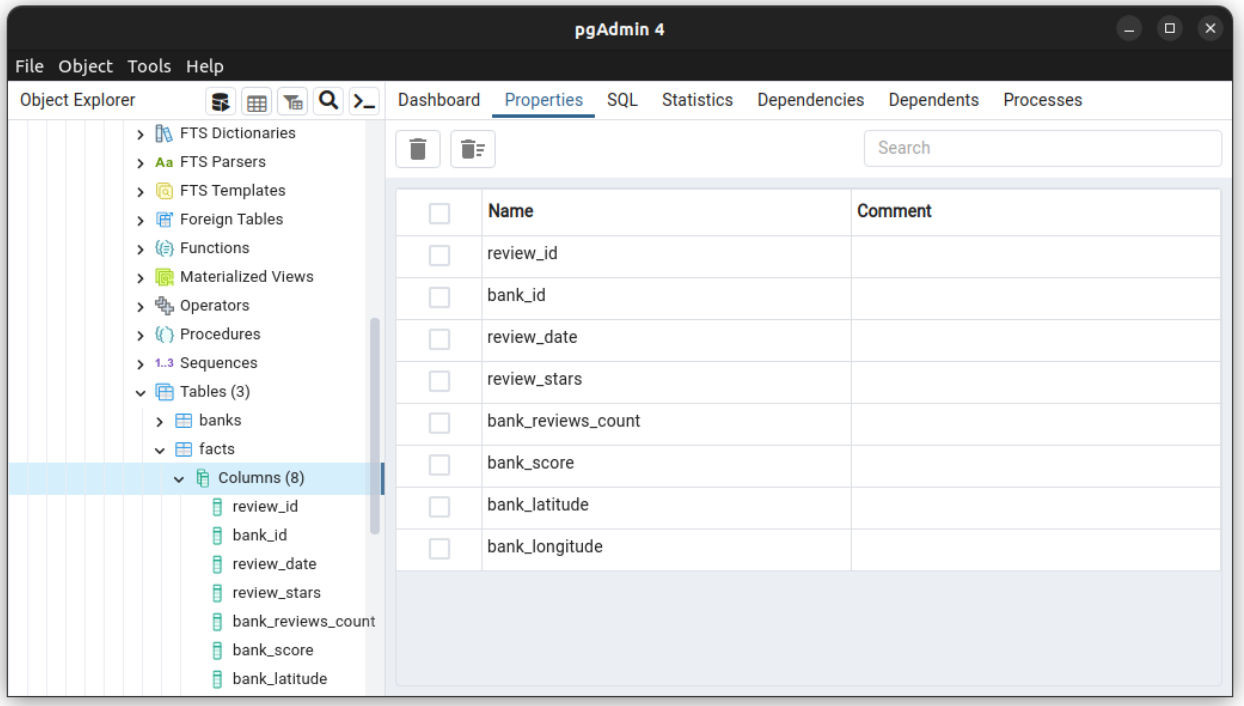
POSTGRES AND PGADMIN

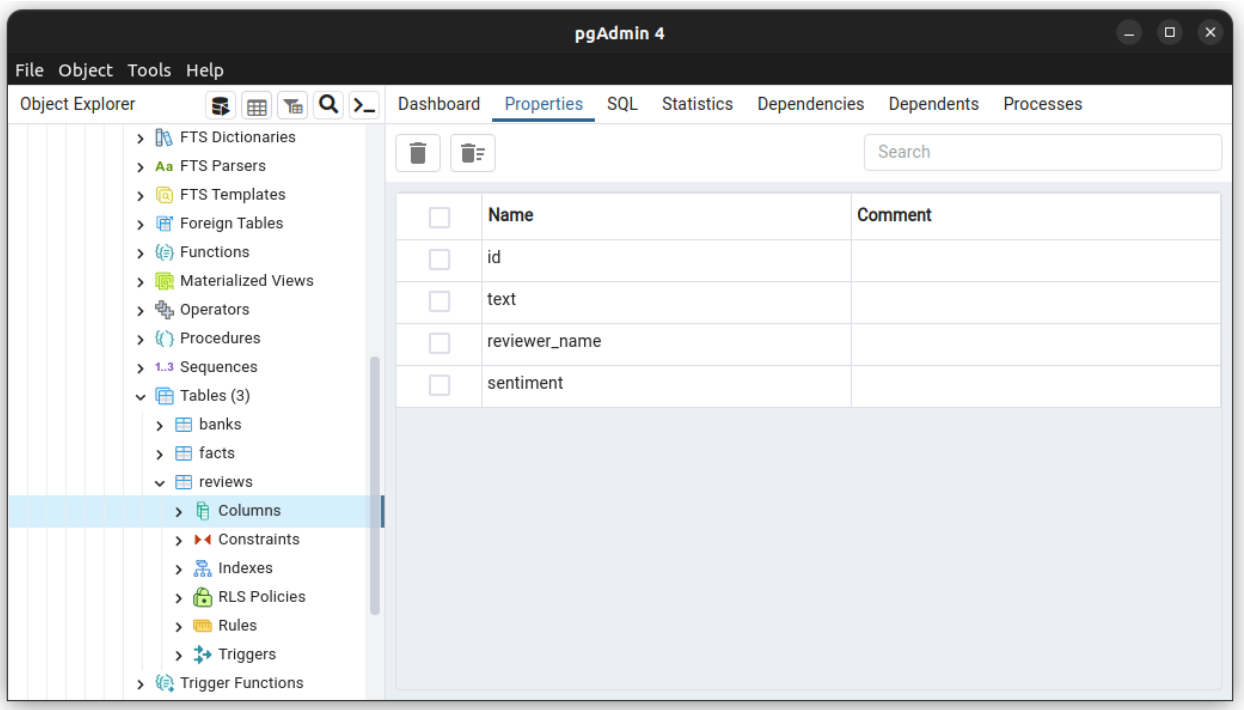
After Modeling the data warehouse, we are going to implement it in Postgres using PGAdmin, since it is considered an easy-to-use interface for creating the database, table schemas, and testing the whole package.


pgAdmin 4
File Object Tools Help
Object Explorer
Servers (1)
PostgreSQL
Databases (2)
moroccan_banks
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
Subscriptions
postgres
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
General
Database
moroccan_banks
OID
16384
Owner
postgres
System database?
Comment
Data Warehouse for the moroccan banks reviews analysis project
Definition
Encoding
UTF8


pgAdmin 4
File Object Tools Help
Object Explorer
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (3)
banks
facts
reviews
Trigger Functions
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
Search
Name
Owner
Partitioned table?
Comment
banks
postgres
facts
postgres
reviews
postgres


pgAdmin 4
File Object Tools Help
Object Explorer
Operators
Procedures
Sequences
Tables (3)
banks
Columns
Constraints
Indexes
RLS Policies
Rules
Triggers
facts
reviews
Trigger Functions
Types
Views
Subscriptions
postgres
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
Search
Name
Comment
id
title
category
adress
city
phone
rank


pgAdmin 4
File Object Tools Help
Object Explorer
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (3)
banks
facts
Columns (8)
review_id
bank_id
review_date
review_stars
bank_reviews_count
bank_score
bank_latitude
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
Search
Name
Comment
review_id
bank_id
review_date
review_stars
bank_reviews_count
bank_score
bank_latitude
bank_longitude



DATA LOADING

Now that we have an operational data warehouse, we need to feed it with data, we used python and the psycopg2 module, here are the important steps we took:

- Creating utility functions for inserting rows into the tables

```python
def insert_rows(table, rows):
    """ insert multiple rows into the table  """
    sql = "INSERT INTO " + table + " VALUES (%s" + ", %s" * (len(rows[0])-1) + ");"
    conn = None
    try:
        # connect to the PostgreSQL database
        conn = psycopg2.connect(database="moroccan_banks", user="postgres" , password="postgres" ,
        host="localhost")
        # create a new cursor
        cur = conn.cursor()
        # execute the INSERT statement
        cur.executemany(sql, rows)
        # commit the changes to the database
        conn.commit()
        # close communication with the database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()
```

Python

We had to test this function before using it for feeding our data warehouse.

```python
# test the function for 1 row
# insert_row("fact_table" ,(1, 1, '2021-04-29', 4.5, 7, 3.4, 32, 18))

# test the function for multiple rows
# insert_rows('reviews', [(2, 'test', 'test', 'test'), (3, 'test', 'test', 'test')])
```

Python

- Then we loaded the datasets and prepared them to match the database schema.

```python
# get data for each table
banks_table = banks[['bank_id', 'bank_title', 'bank_category', 'bank_adress', 'bank_city', 'bank_phone',
'bank_rank']]
```

Python

```python
reviews_table = reviews[['review_id', 'review_text', 'reviewer_name', 'review_sentiment']]
```

Python

```python
facts_table = pd.merge(reviews, banks, on='bank_id', how='inner')[['review_id', 'bank_id', 'review_date',
'review_stars', 'bank_reviews_count', 'bank_score','bank_latitude', 'bank_longitude']]
```

Python

- The last thing was to apply the function on these data frames in order to have a data warehouse fed with data and ready to use for creating the dashboard.

```python
# load data into the database
insert_rows('banks', banks_table.values.tolist())
insert_rows('reviews', reviews_table.values.tolist())
insert_rows('facts', facts_table.values.tolist())
```

Python

CHAPTER 5 : DASHBOARDING

POWER BI (Web Version)

The plan here was to connect the data warehouse to the dashboarding software, however, as I mentioned earlier, we ended up using Power BI on the web because of the short time we had and because of the switch to Linux mid-project.

The limitations we have found were the inability to connect to any data source such as postgres, we had to enter data manually, so we had to join our datasets and paste them in the Excel format into the power bi web editor in order to start creating the dashboard.

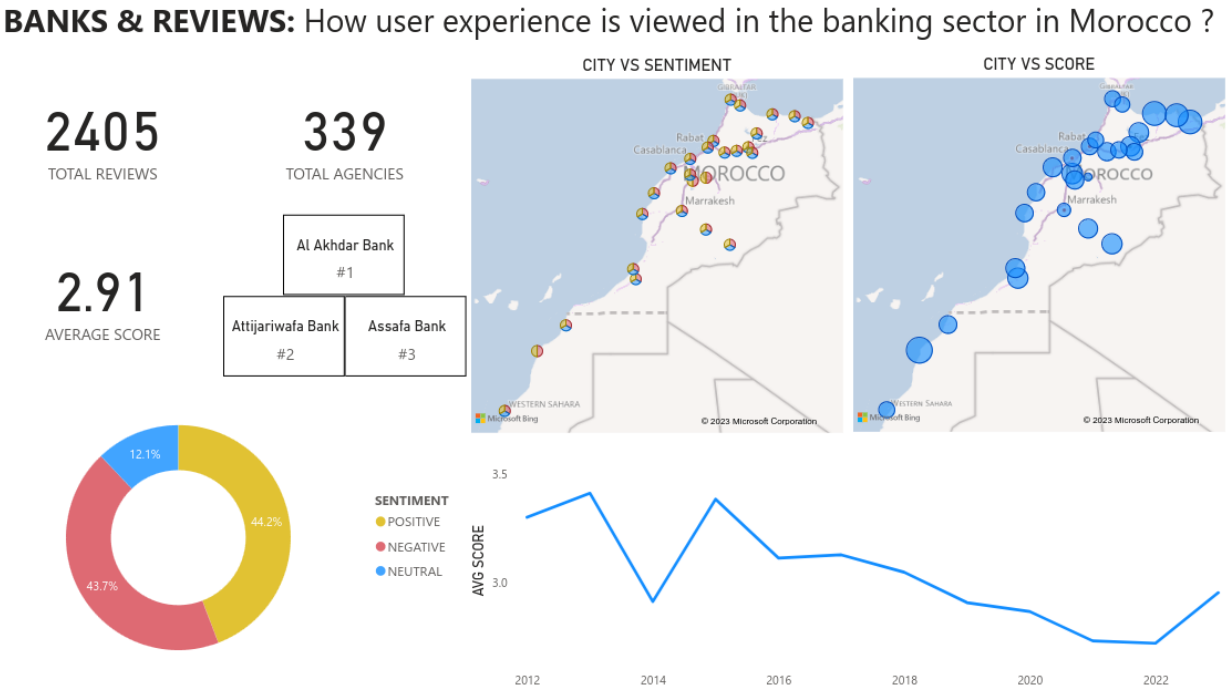
```python
# load the two tables
table1 = pd.read_excel('../data/processed/reviews_cleaned.xlsx')
table2 = pd.read_excel('../data/processed/banks.xlsx')
```

Python

```python
# join the tables on the bank_id column
joined = table1.merge(table2, on='bank_id')
joined.head()
```

Python

Here’s an overview of the final dashboard:



CONCLUSION

We have learned many things throughout working on this project, even if it took wrong turns at the end, and we didn't manage to orchestrate, automate and schedule these tasks with airflow. We learned about the process of ETL, various cleaning technics, connecting Python with Postgres, building and sketching a dashboard, etc.

I think that it would be a total W if we retake these steps, and add more to them, and finding a better alternative to Tableau for the ability of connecting with our data warehouse without doing things manually.