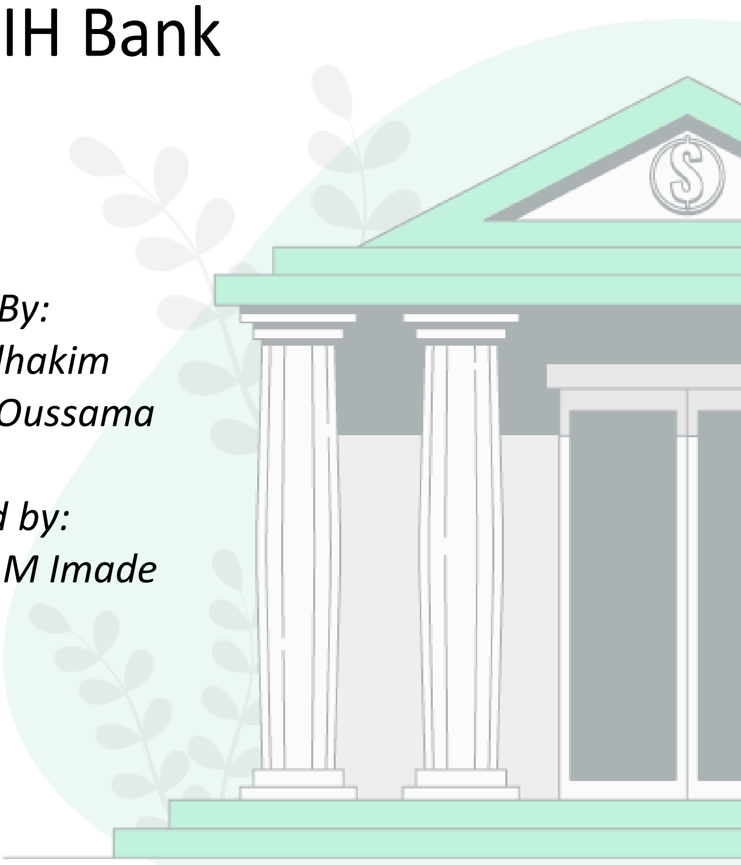
ROYAUME DU MAROC
HAUT-COMMISSARIAT AU PLAN

INSEA

Master of Information Systems and Intelligent Systems

DATA MINING & DATA WAREHOUSE PROJECT

Customer's Feedback Sentiment Analysis of CIH Bank



Realized By:
ZENIBI Abdelhakim
BOUKOUTAYA Oussama

Supervised by:
Mr. BENELALLAM Imade

Academic Year: 2022/2023

Table of Contents

Introduction

The goal of this project is to create a robust data warehouse solution that enables a thorough analysis of customer feedback for each branch of CIH Bank. In today's highly competitive banking sector, understanding customer sentiment and feedback is crucial for banks to improve their services, enhance customer satisfaction, and make data-driven decisions.

To achieve this objective, we will harness modern technologies and application programming interfaces (APIs). Our primary data source will be the Apify Google Maps Scraper API, which allows us to extract specific feedback related to CIH bank branches. This API provides us with a wealth of valuable information, including customer experiences, ratings, and reviews.

After extracting the data, we will employ a sentiment analysis model API from Hugging Face. This advanced model uses natural language processing techniques to categorize each review into one of three categories: positive, neutral, or negative. By conducting sentiment analysis on the extracted reviews, we gain valuable insights into the overall customer sentiment towards different CIH bank branches.

Moreover, our project goes beyond sentiment classification by calculating the total number of reviews for each CIH branch. This quantitative measure enables us to assess the volume of customer feedback and identify branches that may require special attention or improvement.

By combining the capabilities of the Apify Google Maps Scraper API, the Hugging Face sentiment analysis model API, and our custom data aggregation methods, we aim to offer a comprehensive and actionable analysis of CIH bank feedback. This analysis will equip CIH Bank with the necessary insights to make well-informed decisions, enhance customer satisfaction, and continuously improve their services.

I. General Notions:

1. What is Data Warehouse?

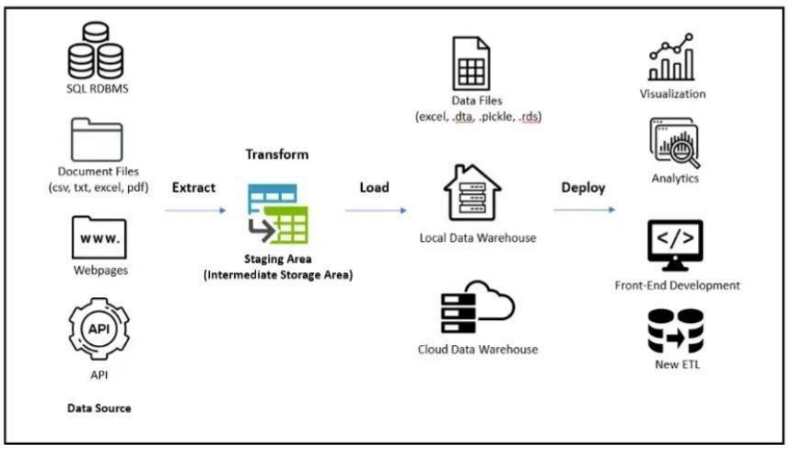
A data warehouse is a system used for reporting and data analysis, and is considered a core componentof business intelligence. It is a relational database that is designed for query and analysis rather than for transaction processing. It typically contains historical data derived from transaction data, but can also include data from other sources. Data is stored in a way that is optimized for reporting and analysis, rather than for transaction processing. The data is then used to create informative and actionable business reports and data visualizations for decision-making purposes.

2. What is ETL?

Extract, Transform & Load Process (ETL) is an important process that is used to populate a data warehouse with the data that is needed for reporting and analysis. It is used to extract data from various sources, clean and transform it into a format that is compatible with the data warehouse, and then load it into the data warehouse. This process is critical for ensuring that the data in the data warehouse is accurate, complete, and up-to-date, and that it can be used to create meaningful and actionable reports and data visualizations.

- Extract: The first step in the ETL process is to extract data from a variety of sources. These sources can be structured or unstructured, and can include databases, flat files, or even social media feeds.
- Transform: The extracted data is then transformed to fit the specific format and structure required by the target system. This can include tasks such as data cleaning, data validation, and data normalization.
- Load: The final step is to load the transformed data into the target system, such as a data warehouse or a data mart.

ETL Process in Detail



3. What is Data mining?

Data mining is the process of finding valuable patterns and insights in large sets of data. It involves using statistical and machine learning techniques to uncover hidden relationships and trends. Data mining helps businesses make informed decisions, predict future trends, and improve processes. It includes techniques like clustering, classification, and association rule mining. By analyzing data from different angles, data mining enables organizations to gain valuable knowledge and gain a competitiveedge.

II. Conception

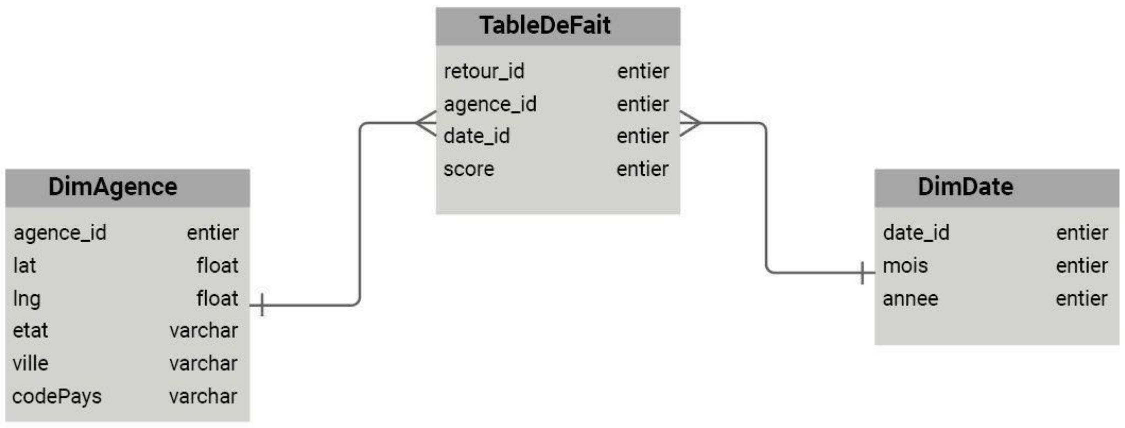
1. KPIs

The primary key performance indicator (KPI) analyzed in this data warehouse project is the Review Score, which serves as an indicator of the sentiment expressed in each review of CIH bank services, categorizing it as positive, neutral, or negative. The sentiment analysis model API from Hugging Face is responsible for this classification and assigns a corresponding score. A positive score of 1 signifies a positive sentiment, a score of 0 denotes a neutral sentiment, and a score of -1 indicates a negative sentiment.

The Review Score KPI will be presented in a visual format with a focus on two key dimensions: location and date. By aggregating scores for each agency across various locations and dates, we can create a comprehensive overview of customer sentiment. Heatmaps will be generated to illustrate the scores, highlighting the intensity of positive or negative sentiment across CIH bank agencies. If a particular location or timeframe shows a higher concentration of negative scores, it would signify areas that need attention and improvement.

Visualizing the score KPI based on location and date empowers decision-makers to identify patterns, trends, and potential areas of customer dissatisfaction. This data can guide targeted actions and strategies to address specific issues and enhance the overall customer experience. Additionally, monitoring changes in sentiment over time through score visualization allows for the evaluation of the effectiveness of any implemented improvements. In summary, this visualization component introduces a dynamic and spatial dimension to the Review Score KPI, providing actionable insights to drive customer-centric decision-making within the banking organization.

2. Fact Table and Dimensions Diagram



The data warehouse project's database schema comprises three interconnected tables. The Agency table contains details about CIH bank agencies, including agency ID and geographic location represented by latitude and longitude.

The fact table functions as the central repository for CIH bank feedback and sentiment analysis scores. It incorporates foreign keys that reference the Agency table and the Date table, creating links between reviews, specific agencies, and dates. The Date table is responsible for holding temporal data, including day, month, and year.

By establishing these relationships and applying foreign key constraints, the database schema ensures data integrity and enables efficient querying and analysis of CIH bank feedback for each agency. The provided database schema, with its relationships and constraints, serves as the foundation of the data warehouse project. It offers a structured and orderly framework for storing and managing the extracted and transformed data, ultimately facilitating effective analysis, visualization, and interpretation of customer sentiment for each CIH bank agency.

III. Realization

1. Airflow DAG Configuration for ETL Pipeline

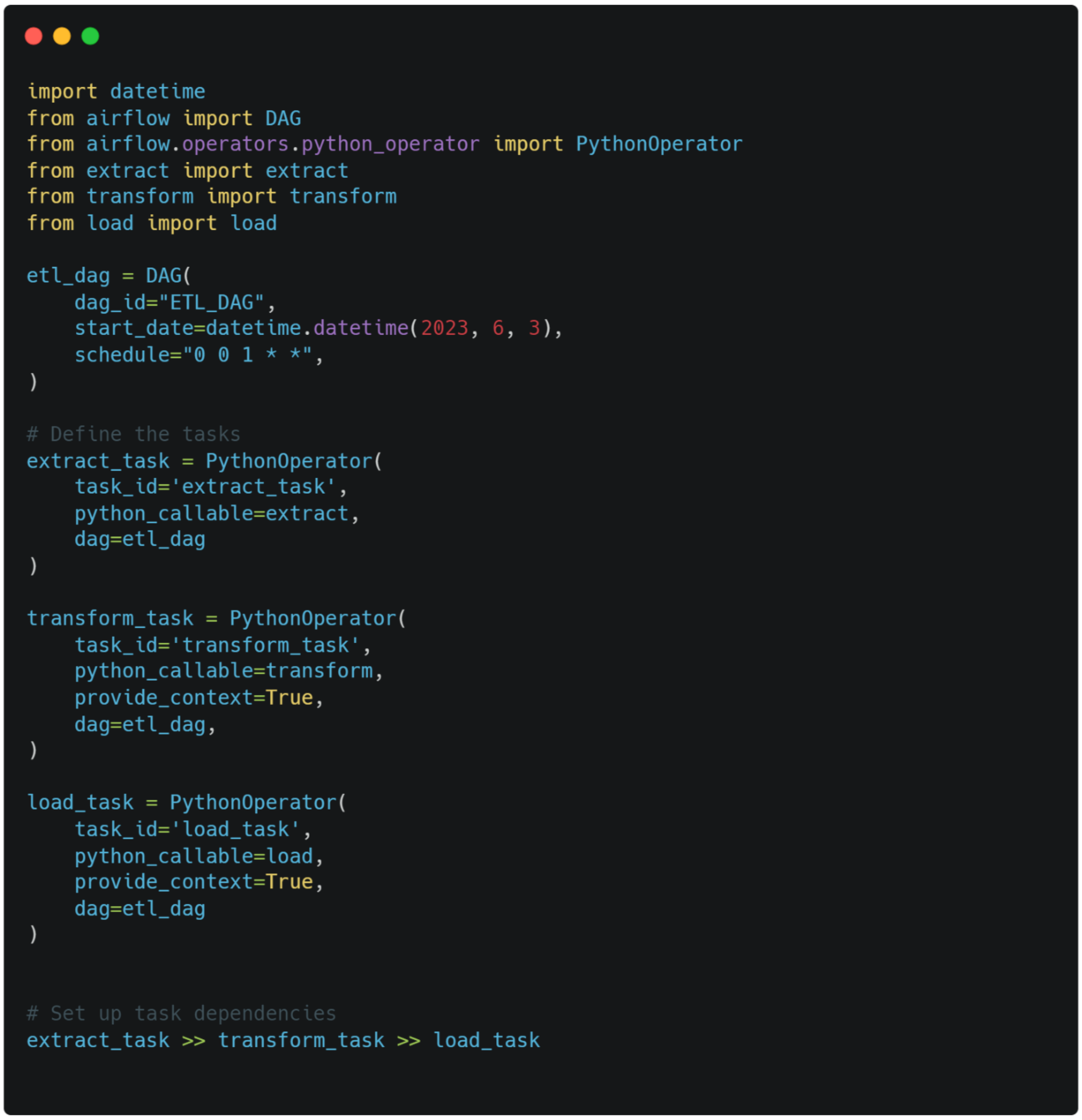
In our data warehouse project, Apache Airflow plays a pivotal role in orchestrating the Extract, Transform, and Load (ETL) pipeline. The provided code snippet offers a glimpse into the configuration of the Airflow Directed Acyclic Graph (DAG), showcasing how tasks are defined and their interdependencies.

The DAG, named "ETL_DAG," is scheduled to kick off on June 3, 2023, commencing at 1 AM on the first day of each month. This DAG comprises three core tasks: extract_task, transform_task, and load_task.

- The extract_task is realized as a PythonOperator, responsible for invoking the extract function, which, in turn, fetches vital data from the APIFY web application. This data forms the foundation for our multi-agency review analysis project.
- Similarly, the transform_task is another PythonOperator that takes on the execution of the transform function.
- The load_task is also a PythonOperator, and its purpose is to trigger the load function, responsible for injecting the transformed data into our data warehouse.

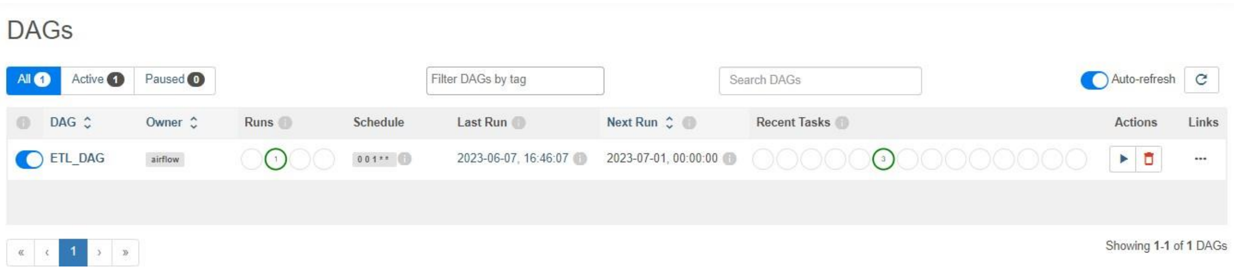
To maintain the task order and interdependencies, the code employs the >> operator. This symbolizes that the extract_task must successfully conclude before the transform_task can begin, and the transform_task must wrap up before the load_task kicks into action. This sequential execution guarantees the integrity and consistency of our ETL pipeline.

Through the meticulous configuration of the DAG and the definition of these task dependencies, we establish a dependable and automated ETL workflow. Apache Airflow adeptly handles task scheduling and execution, empowering us to efficiently pull, transform, and load data from the APIFY web application into our Postgres data warehouse.



```python
import datetime
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from extract import extract
from transform import transform
from load import load

etl_dag = DAG(
    dag_id="ETL_DAG",
    start_date=datetime.datetime(2023, 6, 3),
    schedule="0 0 1 * *",
)

# Define the tasks
extract_task = PythonOperator(
    task_id='extract_task',
    python_callable=extract,
    dag=etl_dag
)

transform_task = PythonOperator(
    task_id='transform_task',
    python_callable=transform,
    provide_context=True,
    dag=etl_dag,
)

load_task = PythonOperator(
    task_id='load_task',
    python_callable=load,
    provide_context=True,
    dag=etl_dag
)


# Set up task dependencies
extract_task >> transform_task >> load_task
```

After running the code, we are able to see our ETL_DAG within the Airflow user interface.



DAGs

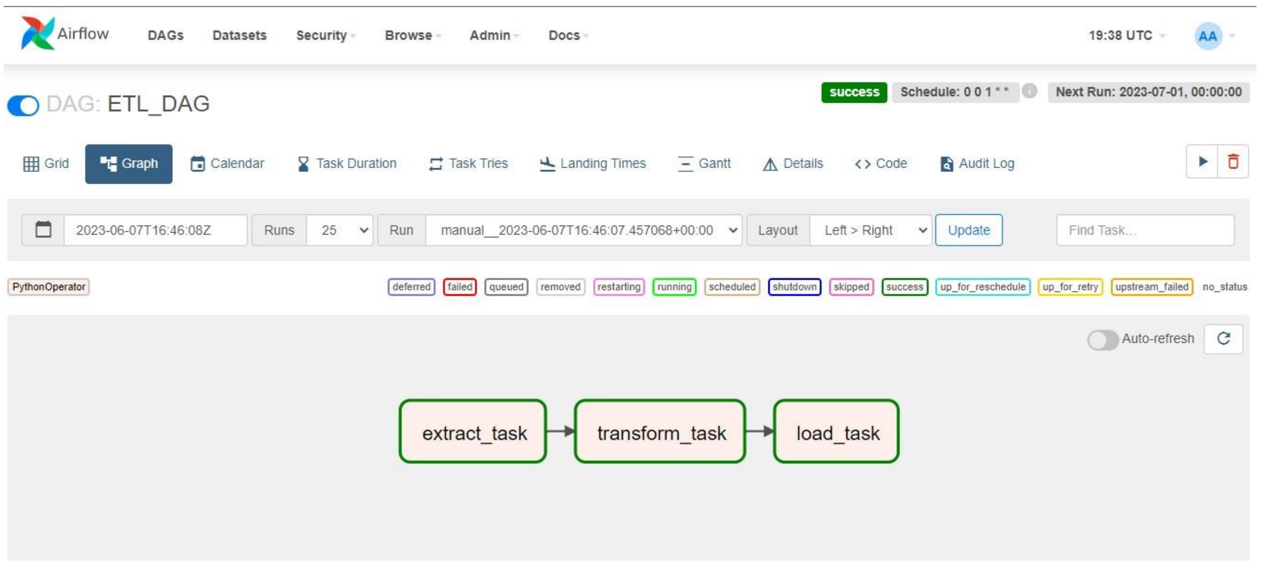
All 1 | Active 1 | Paused 0 | Filter DAGs by tag | Search DAGs | Auto-refresh

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
	ETL_DAG	airflow	1	0 0 1 * *	2023-06-07, 16:46:07	2023-07-01, 00:00:00	3		···

Showing 1-1 of 1 DAGs

- **Graph View**

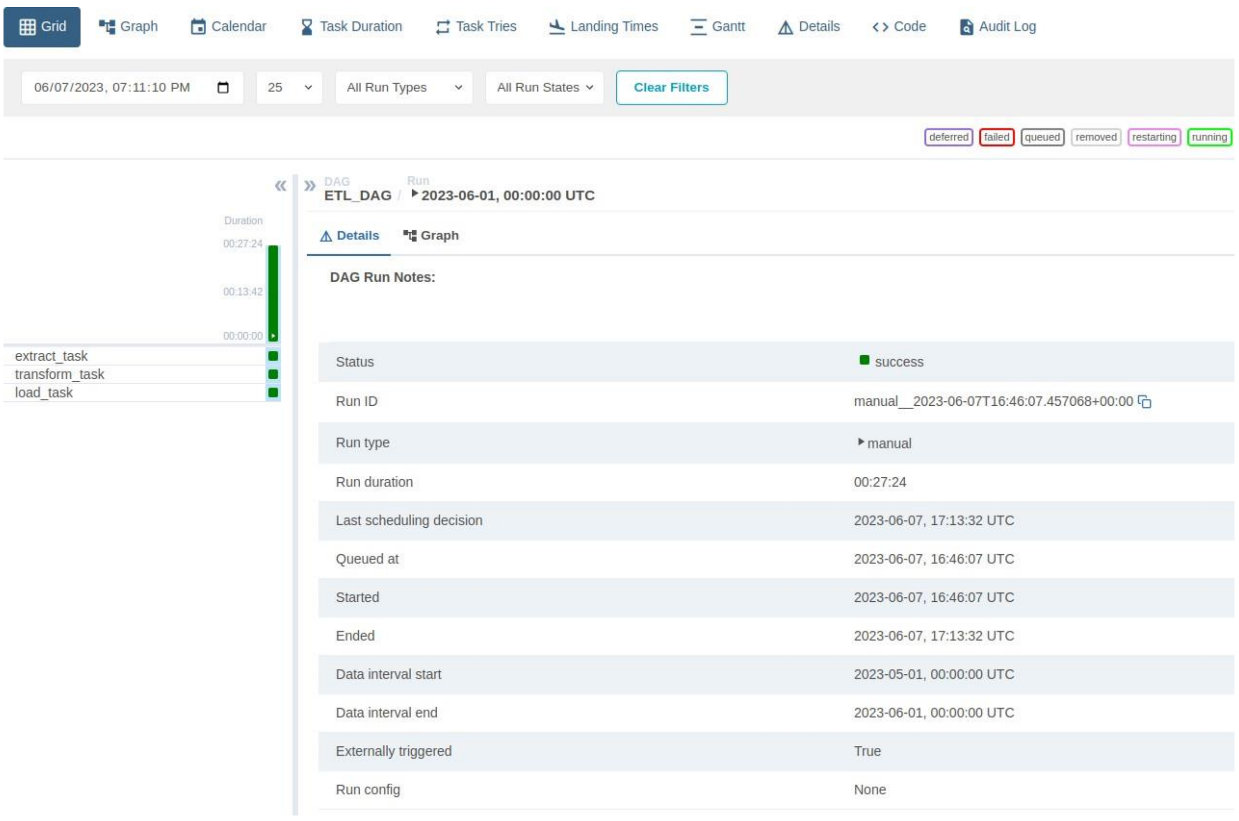
Within the Airflow platform's graphical interface, you'll encounter a visual portrayal of our ETL process. This graphical representation offers insight into the ETL pipeline in a visually intuitive manner. The graph illustrates a sequence of interlinked nodes or tasks, with each node symbolizing a distinct stage within the process. The layout of these nodes is thoughtfully organized to convey the data flow and task dependencies, providing a clear and concise overview of the entire workflow.



- **Grid View**

The grid view image provides a simplified representation of the ETL pipeline in our project. It visually depicts the different stages of the pipeline, including data extraction, transformation, and loading.

Each cell in the grid represents a specific task within a stage, and the connections between cells indicate the flow of data and task dependencies. This visual representation helps in understanding the overall workflow of the ETL process and allows for easy tracking of progress and identification of any issues or bottlenecks.



DAG Run Notes:

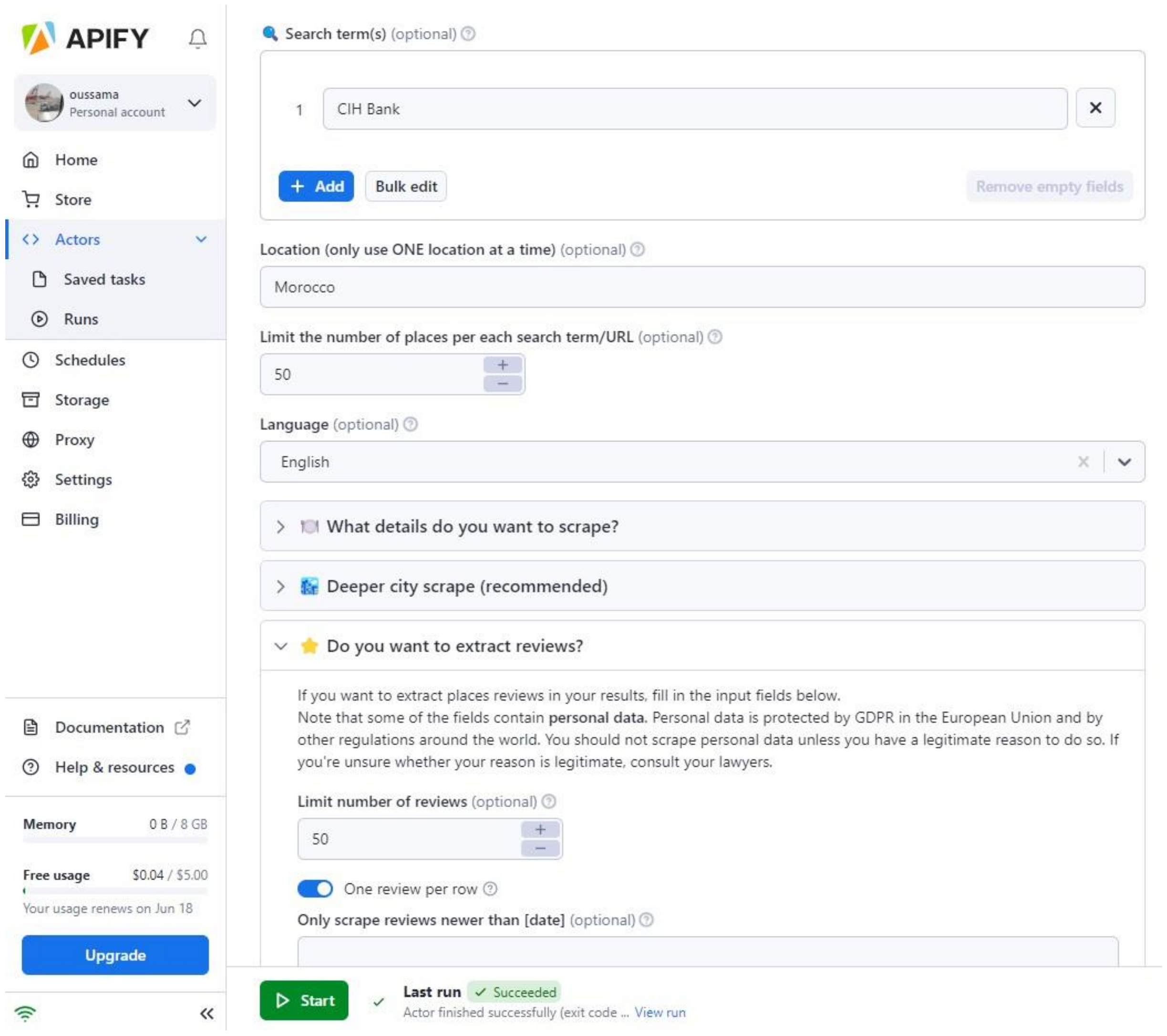
Status	🟩 success
Run ID	manual__2023-06-07T16:46:07.457068+00:00
Run type	manual
Run duration	00:27:24
Last scheduling decision	2023-06-07, 17:13:32 UTC
Queued at	2023-06-07, 16:46:07 UTC
Started	2023-06-07, 16:46:07 UTC
Ended	2023-06-07, 17:13:32 UTC
Data interval start	2023-05-01, 00:00:00 UTC
Data interval end	2023-06-01, 00:00:00 UTC
Externally triggered	True
Run config	None

2. Data Extraction from APIFY Web

Within our Airflow Directed Acyclic Graph (DAG), we establish an extraction function with the primary purpose of fetching data from the APIFY web application. This function plays a central role in the extraction of essential data required for our comprehensive multi-agency review analysis project. To accomplish this, we harness the capabilities of the Google Maps scraper API, a powerful resource that grants us access to a wealth of location-related data, reviews, and other pertinent information sourced from Google Maps.

a. Configuring Google Maps Scraper for CIH Bank

First, we configured the Google Maps scraper API to search for all agencies of CIH Bank. By specifying the search term as «CIH Bank," we ensure that the API retrieves data specifically related to CIH Bank branches and agencies. To narrow down the search results to English language reviews, we set the language parameter to "English." In order to retrieve a comprehensive set of data, we limit the number of places per each search term to 50. Additionally, we specify the country as Morocco to narrow down the search to CIH Bank agencies located within the Morocco



Memory 0 B / 8 GB

Free usage $0.04 / $5.00

Your usage renews on Jun 18

Search term(s) (optional)

1 CIH Bank

Location (only use ONE location at a time) (optional)

Morocco

Limit the number of places per each search term/URL (optional)

50

Language (optional)

English

Do you want to extract reviews?

If you want to extract places reviews in your results, fill in the input fields below.
Note that some of the fields contain **personal data**. Personal data is protected by GDPR in the European Union and by other regulations around the world. You should not scrape personal data unless you have a legitimate reason to do so. If you're unsure whether your reason is legitimate, consult your lawyers.

Limit number of reviews (optional)

50

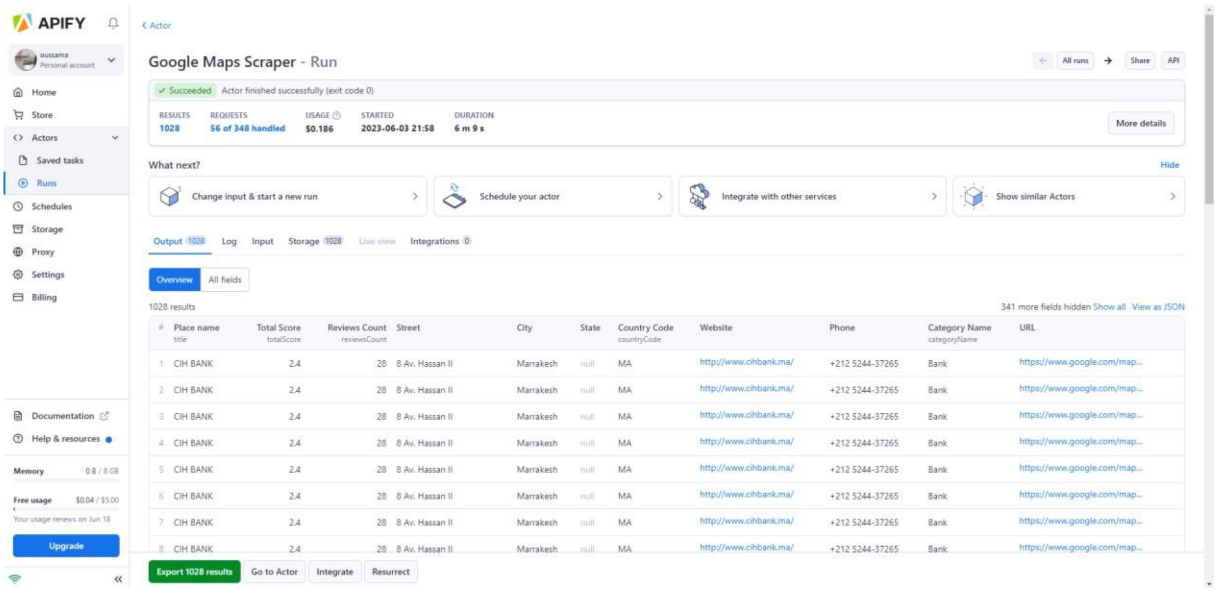
One review per row

Only scrape reviews newer than [date] (optional)

Last run ✓ Succeeded

Actor finished successfully (exit code ... View run

Through the configuration of the Google Maps scraper using these specific parameters, we take measures to guarantee that the extraction process targets the precise data we seek. This data is specifically associated with CIH Bank agencies and is retrieved in English. We also place a limit of 50 results per search term, all confined within the geographic boundary of Morocco. This tailored configuration is instrumental in our efforts to amass pertinent and all-encompassing data essential for our multi-agency review analysis project.



Google Maps Scraper - Run

✓ Succeeded Actor finished successfully (exit code 0)

RESULTS	REQUESTS	USAGE	STARTED	DURATION
1028	56 of 348 handled	$0.186	2023-06-03 21:58	6 m 9 s

What next?

1028 results

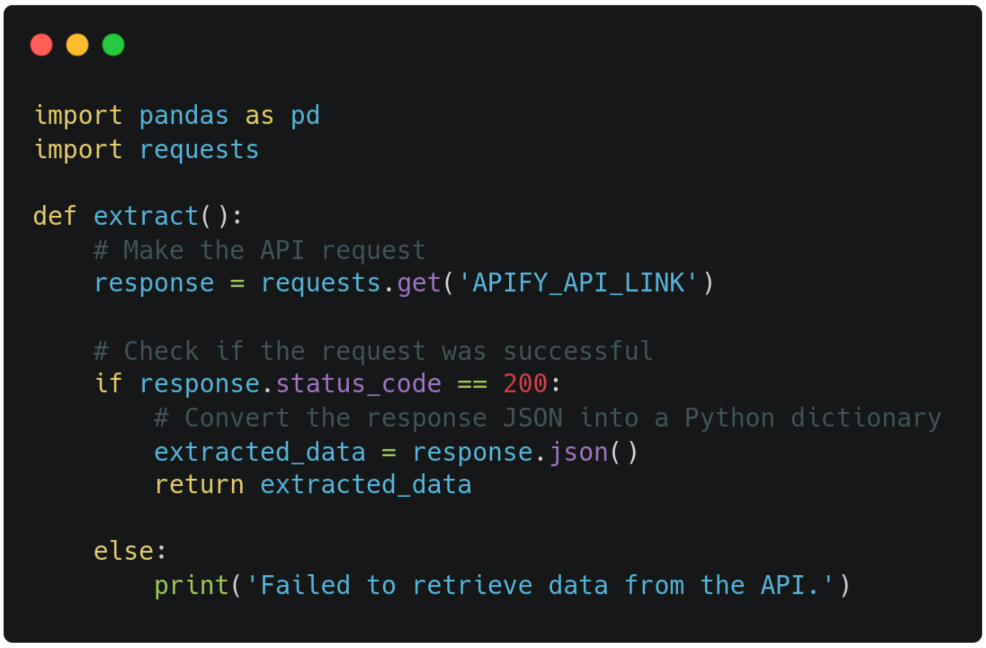
341 more fields hidden

#	Place name title	Total Score totalScore	Reviews Count reviewsCount	Street	City	State	Country Code countryCode	Website	Phone	Category Name categoryName	URL
1	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...
2	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...
3	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...
4	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...
5	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...
6	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...
7	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...
8	CIH BANK	2.4	28	8 Av. Hassan II	Marrakesh	null	MA	http://www.cihbank.ma/	+212 5244-37265	Bank	https://www.google.com/map...

b. Defining the Extraction Function

In this data warehouse project, we utilize an Apache Airflow DAG to orchestrate the extraction of data from the APIFY web application. As part of this process, we implement a specific function responsible for extracting our data about the CIH bank and the feedbacks from the Google Maps scraper API. This function plays a crucial role in retrieving the necessary data for our multi-agency review analysis project.

We accessed the data using an API in Python:


```python
import pandas as pd
import requests

def extract():
    # Make the API request
    response = requests.get('APIFY_API_LINK')

    # Check if the request was successful
    if response.status_code == 200:
        # Convert the response JSON into a Python dictionary
        extracted_data = response.json()
        return extracted_data

    else:
        print('Failed to retrieve data from the API.')
```

The code initiates its execution by sending an API request to a specified endpoint using the requests library. After obtaining the response, it assesses the success of the request by validating the status code. If the response code equals 200, signifying a successful request, the code transforms the response JSON into a Python dictionary, representing the data that was extracted.

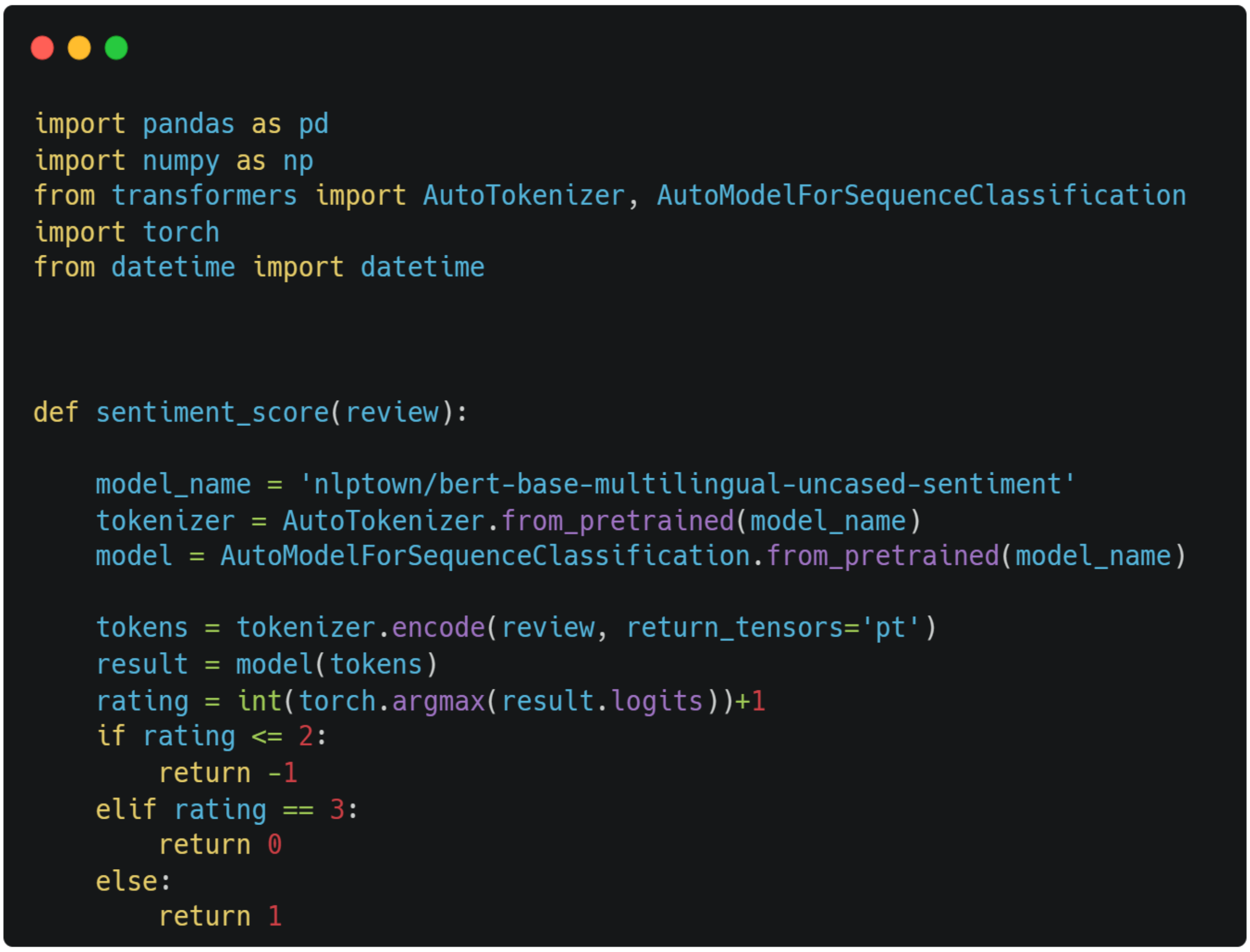
Ultimately, the extracted data is provided for further processing and transformation in subsequent tasks. In cases where the API request fails, the code generates an appropriate message to indicate the inability to fetch data from the API. This code assumes a crucial role in the data extraction phase of the project, ensuring the acquisition of bank feedbacks for each agency from the APIFY web application.

3. Data Transformation and Preprocessing

In the transformation task of our project, we implemented a series of data transformations using Python and various libraries such as pandas, numpy, and transformers. These transformations aimed to preprocess and enrich the data extracted from multiple agencies for further analysis and modeling.

a. Review analysis using Transformers

We start our transformation with the review analysis, where we leverage the power of the BERT model to perform sentiment analysis on the extracted reviews. The BERT (Bidirectional Encoder Representations from Transformers) model is a state-of-the-art pre-trained language model that excels at understanding the contextual relationships between words in a sentence. To conduct the sentiment analysis, we utilize the BERT-based model called **'nlptown/bert-base-multilingual-uncased-sentiment'**, which is specifically trained for multilingual sentiment classification. This model can handle text data in various languages, making it suitable for our multi-agency review analysis project. Next, we instantiate an AutoModelForSequenceClassification object using the same model name. This model has been pre-trained on a large corpus of text data and is capable of predicting sentiment labels based on the encoded sequences.



```python
import pandas as pd
import numpy as np
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
from datetime import datetime



def sentiment_score(review):

    model_name = 'nlptown/bert-base-multilingual-uncased-sentiment'
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSequenceClassification.from_pretrained(model_name)

    tokens = tokenizer.encode(review, return_tensors='pt')
    result = model(tokens)
    rating = int(torch.argmax(result.logits))+1
    if rating <= 2:
        return -1
    elif rating == 3:
        return 0
    else:
        return 1
```

In our sentiment_score function, we encode the input review into tokens using the tokenizer's encode() method. The tokens are returned as PyTorch tensors. These encoded tokens are then fed into the BERT model using **model(tokens)** to obtain the model's output.

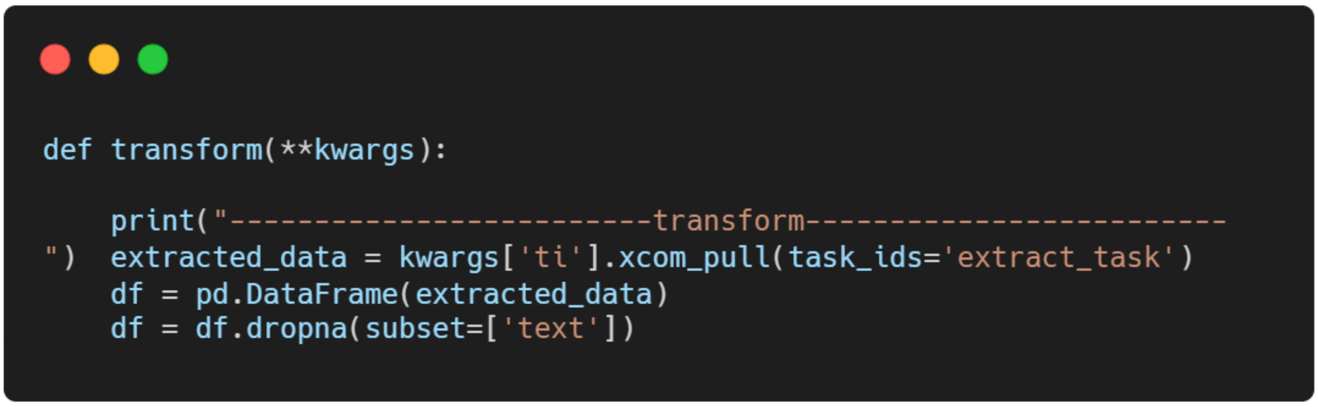
The output of the BERT model, **result**, consists of logits representing the predicted sentiment probabilities for each sentiment class. The torch.argmax() function is applied to find the index of the class with the highest probability. By adding 1 to the index, we obtain the sentiment rating associated with the review.

The BERT model predicts the sentiment of the review as a number of stars (between 1 and 5) .Based on the sentiment rating, we assign a sentiment score to the review. If the rating is less than or equal to 2, we consider it as a negative sentiment and assign a score of -1. If the rating is 3, we interpret it as a neutral sentiment and assign a score of 0. For ratings greater than 3, we categorize it as a positive sentiment and assign ascore of 1.

b. Treating Missing Values and Geographic Data

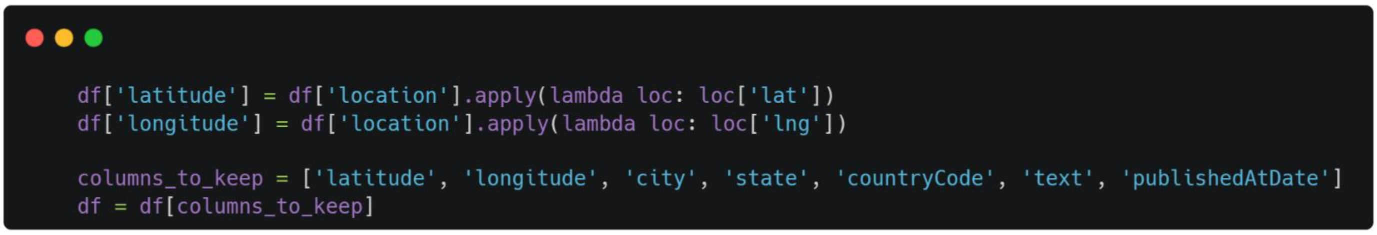
In our function **transform (**kwargs),** we begin by extracting the data from the previous task and converting it into a pandas DataFrame (df). This DataFrame serves as the basis for our data transformation and preprocessing.

To ensure the quality of the data, we start by treating missing values. Any rows with missing values in the 'text' column are dropped using **df.dropna(subset=['text']).** This step ensures that we work with complete textual data for accurate analysis.



```
def transform(**kwargs):

    print("-------------------------transform------------------------
")  extracted_data = kwargs['ti'].xcom_pull(task_ids='extract_task')
    df = pd.DataFrame(extracted_data)
    df = df.dropna(subset=['text'])
```

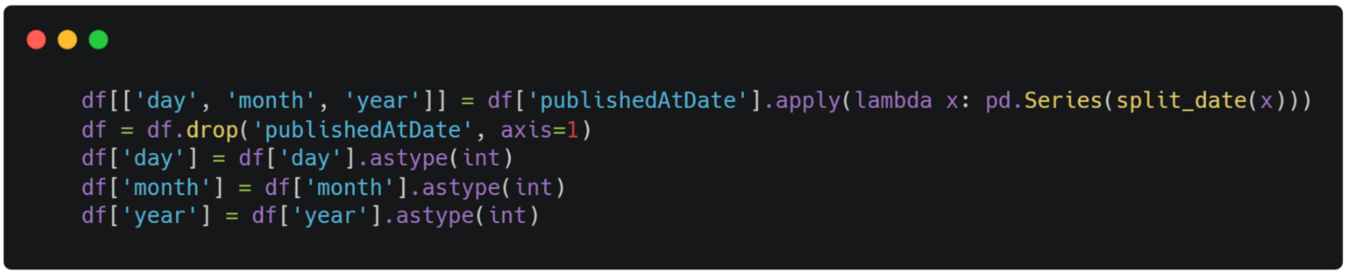
Next, we proceed to extract the latitude and longitude coordinates from the 'location' column. Using lambda functions, we assign these values to new columns in the DataFrame: 'latitude' and 'longitude'. These geographical coordinates will be useful for spatial analysis or visualizations. For further processing, we select a subset of columns that are relevant to our analysis. These include 'lat', 'lng', 'ville','CodePays', 'text', and 'publishedAtDate'. These columns contain essential information that we will leverage in our multi-agency review analysis.



```
df['latitude'] = df['location'].apply(lambda loc: loc['lat'])
df['longitude'] = df['location'].apply(lambda loc: loc['lng'])

columns_to_keep = ['latitude', 'longitude', 'city', 'state', 'countryCode', 'text', 'publishedAtDate']
df = df[columns_to_keep]
```

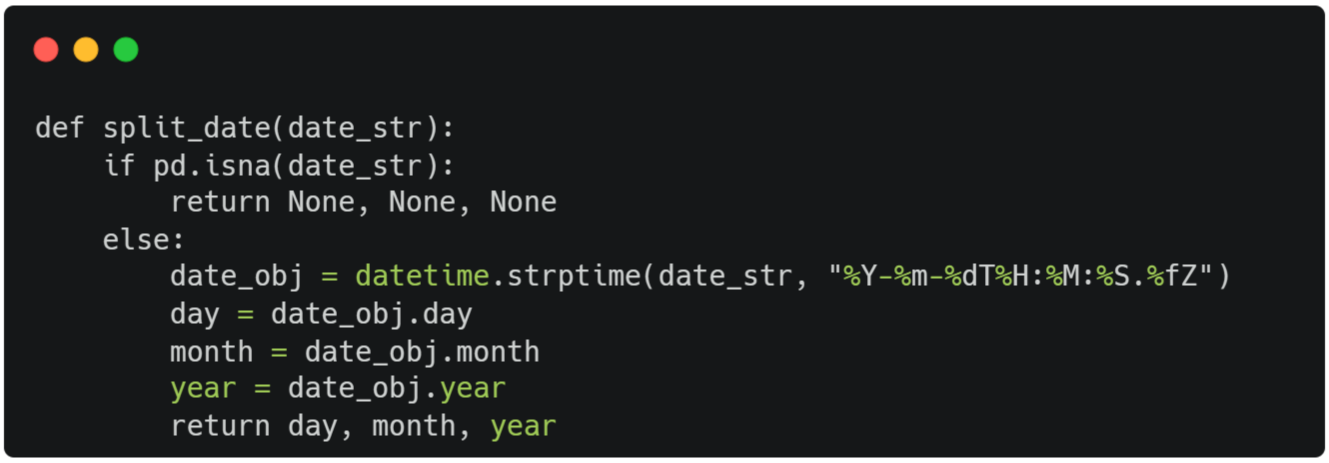
c. Transforming Date Format

We also perform a transformation on the date format as part of our data preprocessing. The date format provided in the dataset is in the form of a string, which needs to be converted into a more structured format for further analysis.



```python
df[['day', 'month', 'year']] = df['publishedAtDate'].apply(lambda x: pd.Series(split_date(x)))
df = df.drop('publishedAtDate', axis=1)
df['day'] = df['day'].astype(int)
df['month'] = df['month'].astype(int)
df['year'] = df['year'].astype(int)
```

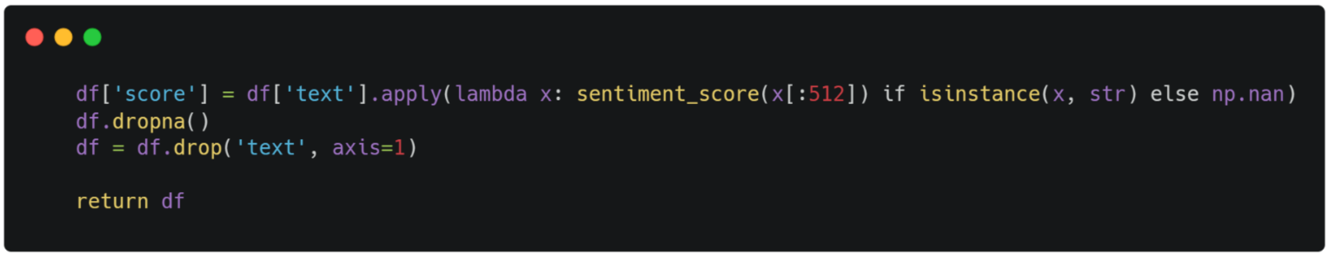
To accomplish this, we utilize the **split_date(date_str)** function. This function takes a date string as input and splits it into its day, month, and year components. If the date string is empty or missing, we assign None values to all components.



```python
def split_date(date_str):
    if pd.isna(date_str):
        return None, None, None
    else:
        date_obj = datetime.strptime(date_str, "%Y-%m-%dT%H:%M:%S.%fZ")
        day = date_obj.day
        month = date_obj.month
        year = date_obj.year
        return day, month, year
```

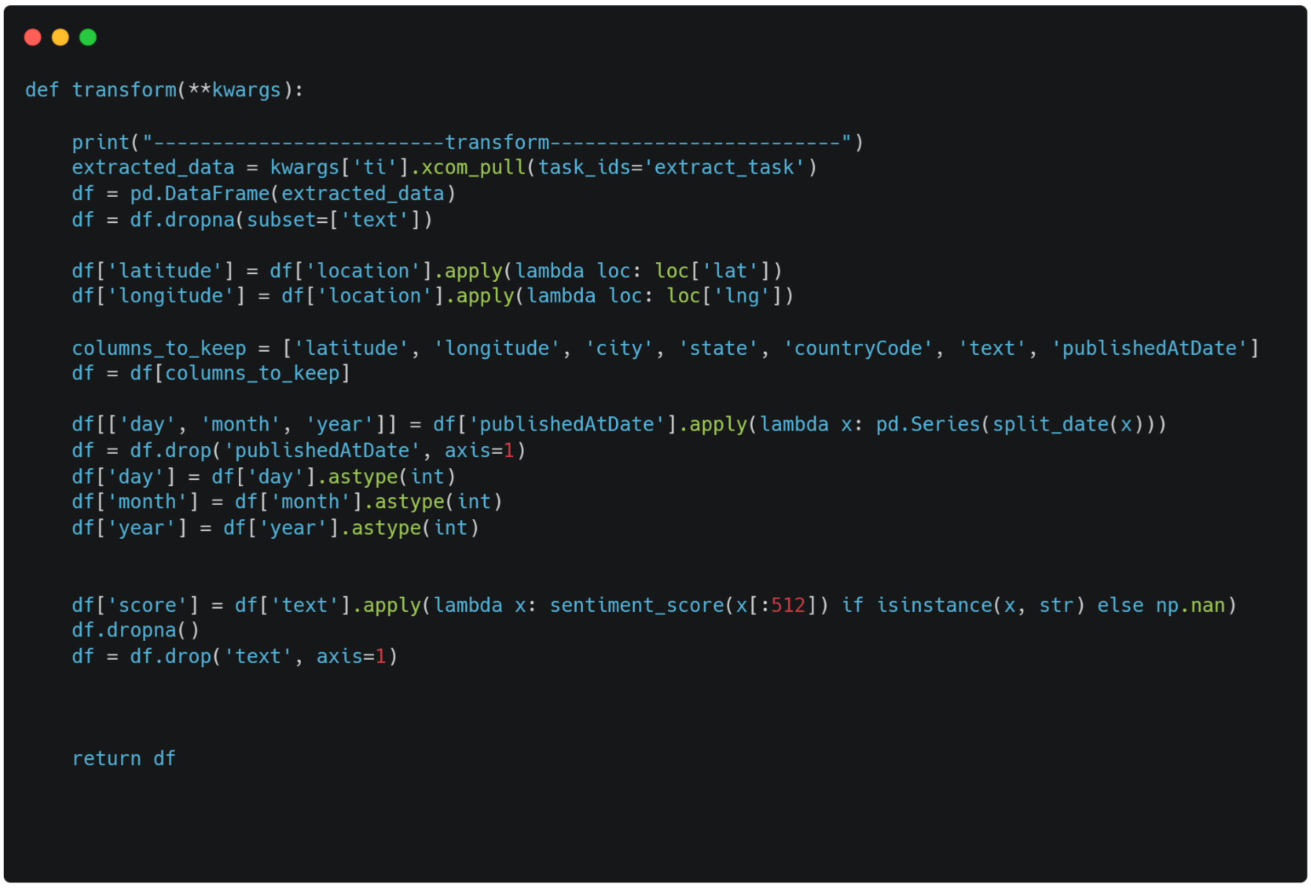
d. Calculating Sentiment Scores

To gauge the sentiment expressed in each review, we calculate a sentiment score using the **sentiment_score(review) function** that we explained before. This score provides an indication of the sentiment polarity of the review. Any rows with missing sentiment scores are then dropped from the DataFrame, ensuring that we work with reliable sentiment data. Finally, we remove the 'text' column from the DataFrame since we have already derived the sentiment scores from the text.



```python
df['score'] = df['text'].apply(lambda x: sentiment_score(x[:512]) if isinstance(x, str) else np.nan)
df.dropna()
df = df.drop('text', axis=1)

return df
```

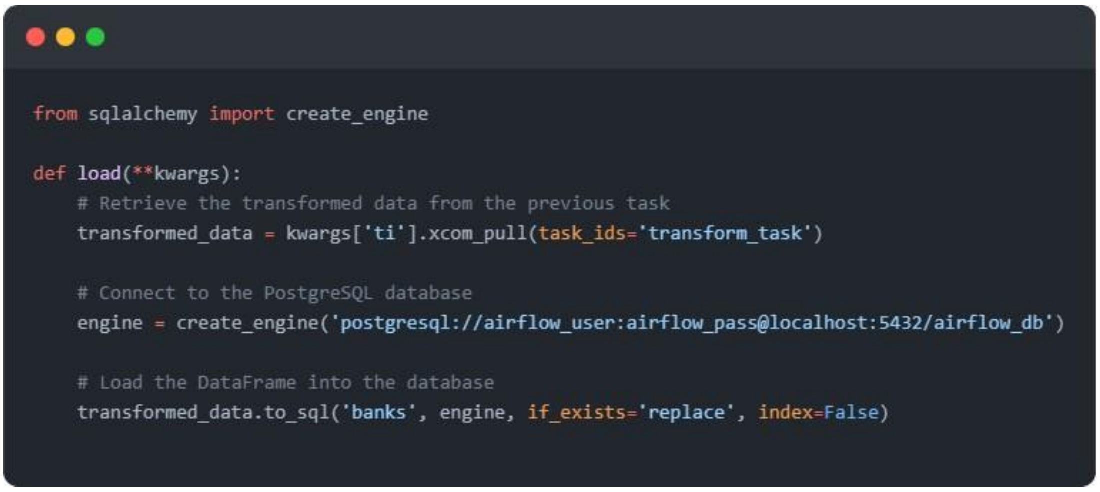
The resulting transformed DataFrame, df, contains the necessary columns for further analysis in our multi-agency review analysis project. By extracting data, handling missing values, and deriving sentiment scores, we lay the foundation for deeper insights and meaningful exploration of the data.



```python
def transform(**kwargs):

    print("-------------------------transform------------------------")
    extracted_data = kwargs['ti'].xcom_pull(task_ids='extract_task')
    df = pd.DataFrame(extracted_data)
    df = df.dropna(subset=['text'])

    df['latitude'] = df['location'].apply(lambda loc: loc['lat'])
    df['longitude'] = df['location'].apply(lambda loc: loc['lng'])

    columns_to_keep = ['latitude', 'longitude', 'city', 'state', 'countryCode', 'text', 'publishedAtDate']
    df = df[columns_to_keep]

    df[['day', 'month', 'year']] = df['publishedAtDate'].apply(lambda x: pd.Series(split_date(x)))
    df = df.drop('publishedAtDate', axis=1)
    df['day'] = df['day'].astype(int)
    df['month'] = df['month'].astype(int)
    df['year'] = df['year'].astype(int)


    df['score'] = df['text'].apply(lambda x: sentiment_score(x[:512]) if isinstance(x, str) else np.nan)
    df.dropna()
    df = df.drop('text', axis=1)



    return df
```

4. Data Loading into Postgres

In the context of our ETL (Extract, Transform, Load) DAG, we've introduced a data loading step within the load_task. The primary objective of this step is to take the transformed data and insert it into our PostgreSQL database. When we execute the load_task, it guarantees that the transformed data becomes securely and durably stored in the database. This, in turn, facilitates effortless data retrieval and enables subsequent analysis with convenience.



```python
from sqlalchemy import create_engine

def load(**kwargs):
    # Retrieve the transformed data from the previous task
    transformed_data = kwargs['ti'].xcom_pull(task_ids='transform_task')

    # Connect to the PostgreSQL database
    engine = create_engine('postgresql://airflow_user:airflow_pass@localhost:5432/airflow_db')

    # Load the DataFrame into the database
    transformed_data.to_sql('banks', engine, if_exists='replace', index=False)
```

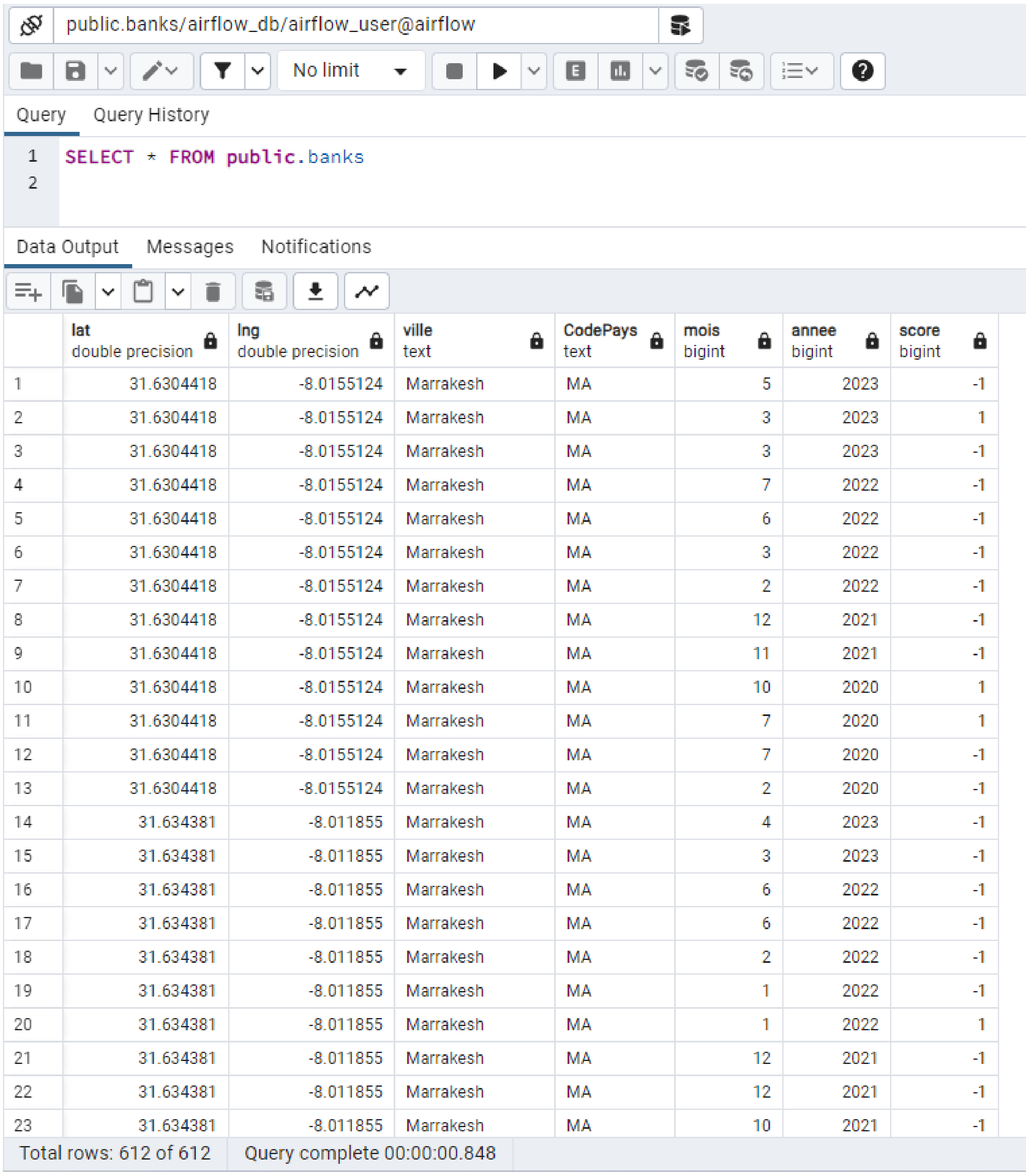
The load function employs the SQLAlchemy library to create a connection to the PostgreSQL database. It initiates by retrieving the transformed data from the preceding task using the xcom_pull method, ensuring that the correct data is available for database loading.

With the create_engine method, the load function generates an engine object, encapsulating all the essential connection details such as username, password, host, port, and database name. This engine object functions as the gateway for interactions with the database.

Once the database connection is established, the load function proceeds to insert the transformed data into the 'banks' table within the database. This operation is executed by invoking the **to_sql method** on the **transformed_data DataFrame**.

By integrating the load function into our ETL pipeline, we guarantee that the transformed data is securely stored within the PostgreSQL database, making it ready for subsequent analysis and reporting. This capability empowers us to efficiently manage and harness the data for gaining insights and making informed decisions.

Here is an example of the 'banks' table loaded in PostgreSQL after executing the load function:



public.banks/airflow_db/airflow_user@airflow

```sql
SELECT * FROM public.banks
```

	lat double precision	lng double precision	ville text	CodePays text	mois bigint	annee bigint	score bigint
1	31.6304418	-8.0155124	Marrakesh	MA	5	2023	-1
2	31.6304418	-8.0155124	Marrakesh	MA	3	2023	1
3	31.6304418	-8.0155124	Marrakesh	MA	3	2023	-1
4	31.6304418	-8.0155124	Marrakesh	MA	7	2022	-1
5	31.6304418	-8.0155124	Marrakesh	MA	6	2022	-1
6	31.6304418	-8.0155124	Marrakesh	MA	3	2022	-1
7	31.6304418	-8.0155124	Marrakesh	MA	2	2022	-1
8	31.6304418	-8.0155124	Marrakesh	MA	12	2021	-1
9	31.6304418	-8.0155124	Marrakesh	MA	11	2021	-1
10	31.6304418	-8.0155124	Marrakesh	MA	10	2020	1
11	31.6304418	-8.0155124	Marrakesh	MA	7	2020	1
12	31.6304418	-8.0155124	Marrakesh	MA	7	2020	-1
13	31.6304418	-8.0155124	Marrakesh	MA	2	2020	-1
14	31.634381	-8.011855	Marrakesh	MA	4	2023	-1
15	31.634381	-8.011855	Marrakesh	MA	3	2023	-1
16	31.634381	-8.011855	Marrakesh	MA	6	2022	-1
17	31.634381	-8.011855	Marrakesh	MA	6	2022	-1
18	31.634381	-8.011855	Marrakesh	MA	2	2022	-1
19	31.634381	-8.011855	Marrakesh	MA	1	2022	-1
20	31.634381	-8.011855	Marrakesh	MA	1	2022	1
21	31.634381	-8.011855	Marrakesh	MA	12	2021	-1
22	31.634381	-8.011855	Marrakesh	MA	12	2021	-1
23	31.634381	-8.011855	Marrakesh	MA	10	2021	-1

Total rows: 612 of 612 Query complete 00:00:00.848

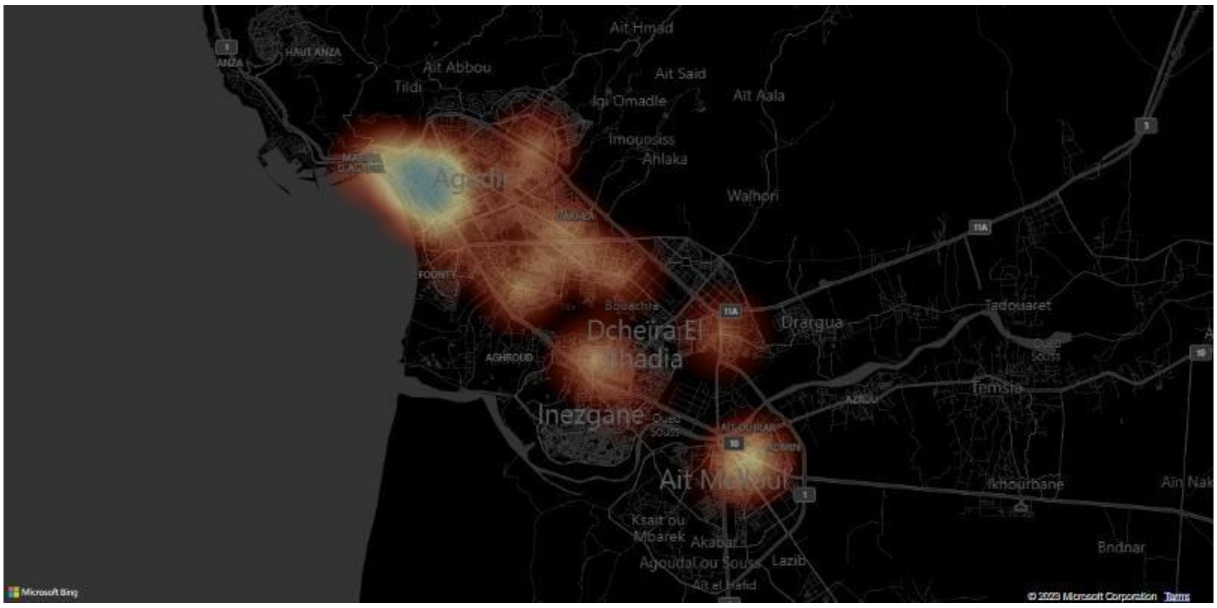
The 'banks' table is a combination of tables that contains the transformed data from our ETL process, including information such as latitude, longitude, cities, sentiment score , and other relevant attributes. Storing the data in a structured format allows for efficient querying and analysis, enabling us to derive meaningful insights and make informed business decisions based on the CIH bank feedback.

IV. Visualization and interpretation

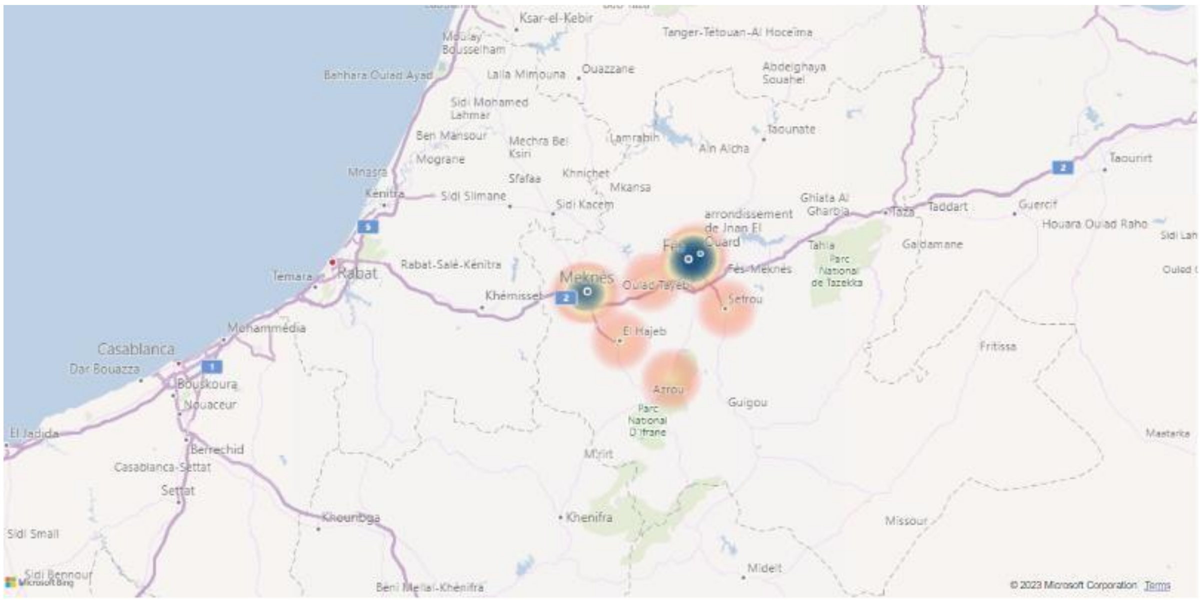
In this data warehouse project, we leverage a heatmap to provide a visual representation of the sentiment analysis outcomes for each CIH bank agency, taking into account their geographic positions. This visualization employs a color-coded scheme to illustrate various sentiment categories.

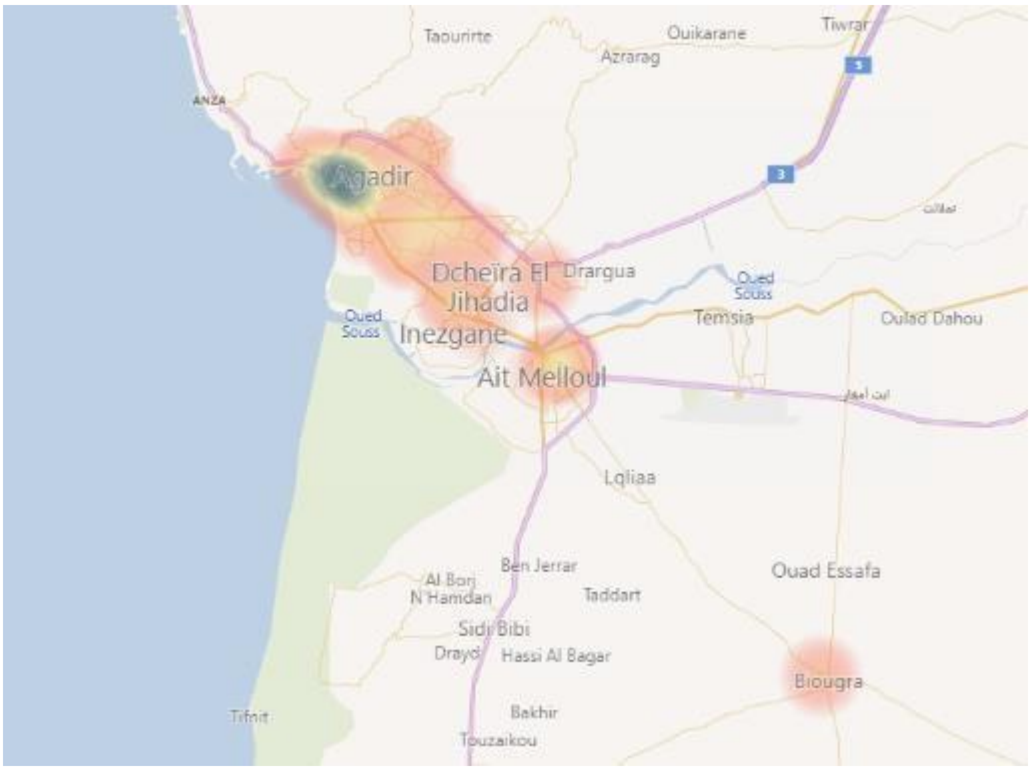
Locations associated with a preponderance of negative reviews are denoted by the color red, symbolizing regions where customer dissatisfaction is prevalent. Meanwhile, areas receiving predominantly neutral sentiment reviews are portrayed in a calming blue shade, indicating a balanced response. On the contrary, locales with an abundance of positive reviews are displayed in a refreshing green hue, underscoring high levels of customer satisfaction.

1. ArcGIS map



2. Various simple maps



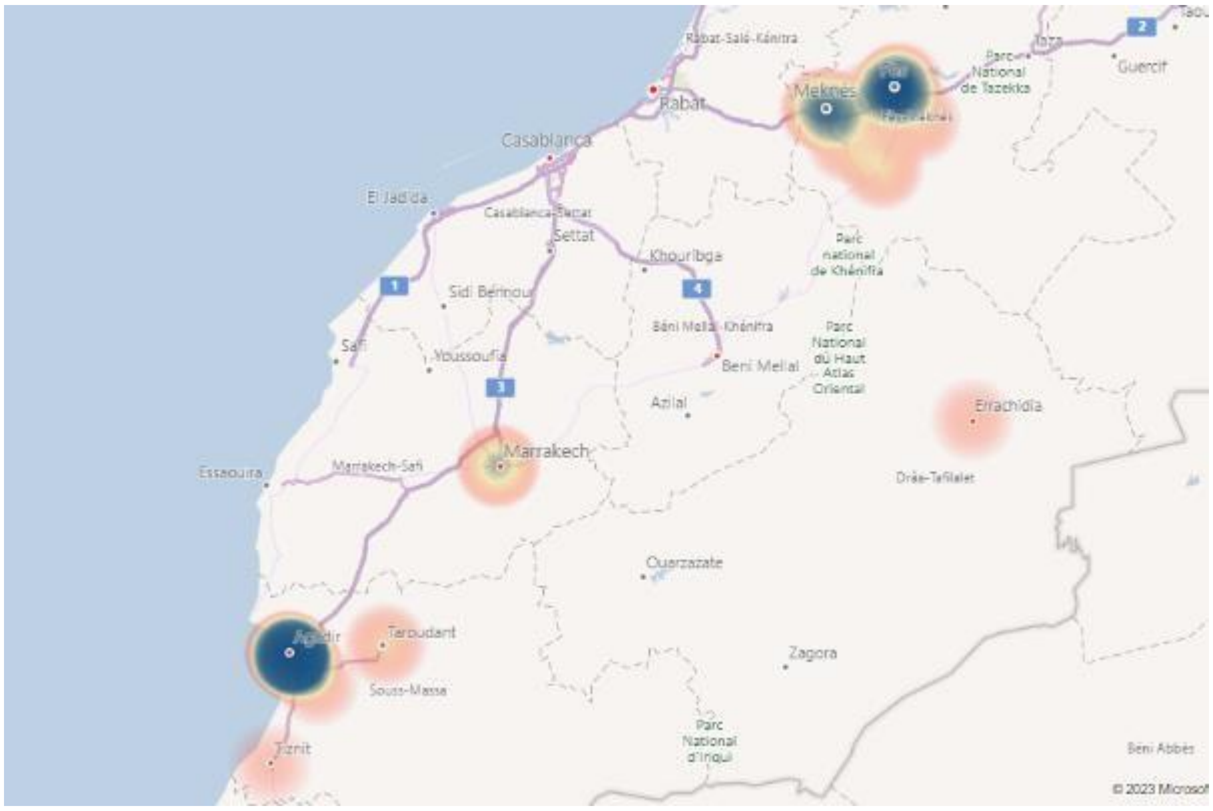


By incorporating this color-coded approach, the heatmap provides an intuitive and visually appealing representation of sentiment distribution across CIH agency locations.

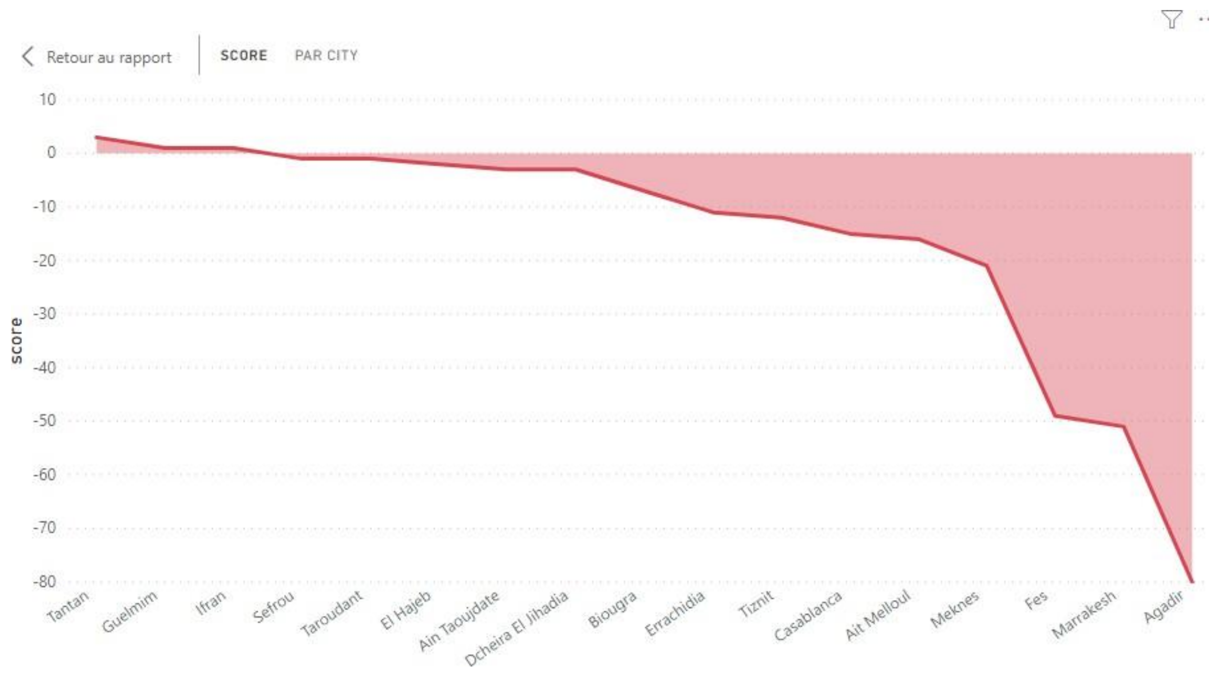
Stakeholders can readily identify and analyze regions with a concentration of negative, neutral, or positive sentiment, enabling them to pinpoint areas that require immediate attention or further improvement.

This enhanced visualization of sentiment scores per agency location in distinct colors empowers decision-makers to make data-driven actions and devise strategies to enhance customer satisfaction and address sentiment-related concerns more effectively.

Additionally, our data warehouse project includes two other visualizations that provide valuable insights into the CIH bank feedbacks, enhancing the analytical capabilities of the project. By leveraging these visualizations, stakeholders can gain actionable insights into customer sentiments, identify areas of improvement, and make informed decisions to enhance customer satisfaction and drive business growth.


Rabat-Salé-Kénitra
Rabat
Casablanca
Meknes
Fes
Taza
Guercif
Parc National de Tazekka
El Jadida
Casablanca-Settat
Settat
Khouribga
Parc national de Khénifra
Sidi Bennour
Safi
Youssoufia
Béni Mellal-Khénifra
Beni Mellal
Parc National du Haut Atlas Oriental
Azilal
Errachidia
Marrakech
Essaouira
Marrakech-Safi
Drâa-Tafilalet
Ouarzazate
Agadir
Taroudant
Souss-Massa
Zagora
Tiznit
Parc National d'Iriqui
Béni Abbès
© 2023 Microsoft

3. area char


Retour au rapport
SCORE
PAR CITY
score
10
0
-10
-20
-30
-40
-50
-60
-70
-80
Tantan
Guelmim
Ifran
Sefrou
Taroudant
El Hajeb
Ain Taoujdate
Dcheira El Jihadia
Biougra
Errachidia
Tiznit
Casablanca
Ait Melloul
Meknes
Fes
Marrakesh
Agadir

V. The technologies used:

1. Apache Airflow

Apache Airflow is an open-source platform that helps manage and schedule workflows and data pipelines. It allows users to define tasks and their dependencies as code, making it easy to create and monitor complex workflows. With a user-friendly interface and extensible features, Airflow simplifies the automation and management of data processing tasks, enabling efficient workflow orchestration.

2. Apify

Apify is a user-friendly platform for web scraping and data extraction. It automates the process of collecting data from websites, transforming it into a structured format, and storing it for analysis. With Apify, you can easily create scraping tasks, handle complex website interactions, and process the extracted data. It offers scalability, monitoring, and error handling features, making it a convenient tool for efficient web scraping.

3. PostgreSQL

PostgreSQL is a powerful and reliable open-source database system. It can store and manage structured data with flexibility. It supports advanced SQL queries and indexing for efficient data manipulation. PostgreSQL prioritizes data security and offers features like encryption and access control. It works on multiple platforms and has a strong community for support and development. Overall, PostgreSQL is a versatile and trusted choice for storing and managing data.

4. Power BI

Power BI is a user-friendly data visualization and business intelligence tool developed by Microsoft. It helps users connect to different data sources, transform data into visual representations, and create interactive reports and dashboards. With Power BI, you can import data, create visualizations, perform data analysis, collaborate with others, and share insights. It offers a range of features and integrations with other Microsoft tools for seamless data exploration and reporting. Overall, Power BI simplifies the process of turning data into meaningful insights for better decision-making.

Conclusion

In summary, our data warehouse project, which focused on analyzing bank feedback for each CIH agency, has delivered invaluable insights and actionable information to enhance customer satisfaction and support decision-making. We achieved this by harnessing the Apify Google Maps Scraper API to extract precise bank feedback, including customer reviews and ratings, from various agency locations.

We further categorized these extracted reviews into positive, neutral, and negative sentiments through the use of the Hugging Face sentiment analysis model API. This sentiment analysis empowered us to grasp the overall customer sentiment toward different agencies and pinpoint areas that necessitate attention and improvement.

Furthermore, by tallying the total number of reviews for each agency, we obtained a quantitative measure of customer feedback. This measure provided a comprehensive understanding of the feedback volume received by each agency, allowing CIH bank to efficiently prioritize efforts and allocate resources.

Our project exemplifies the transformative influence of data-driven analysis within the banking sector. Through the amalgamation of advanced technologies, including API integration, sentiment analysis models, and data aggregation techniques, we have furnished CIH bank with actionable insights into customer sentiment and feedback.

Looking ahead, the insights unearthed by our data warehouse project can serve as a cornerstone for further exploration, enabling CIH bank to delve deeper into specific facets of customer feedback, identify trends, and execute targeted improvements.

Overall, our data warehouse solution has equipped CIH bank with the essential tools to enhance their services, make informed decisions, and ultimately elevate customer satisfaction. Embracing data-driven methodologies, CIH bank can more effectively meet customer expectations, nurture stronger relationships, and maintain competitiveness within the dynamic banking landscape.