INSTITUT SUPÉRIEUR D'ÉLECTRONIQUE DE PARIS (ISEP)

II.2414 ADVANCED DATABASES AND BIG DATA

BIG DATA PROJECT

(Data Lake and Data Pipeline)

BUHARI ALIYU

ABUBAKAR UMAR ELNAFATY

PROF. THIBAUT DE BROCA

JUNE 2023

Synopsis

The main goal of the 'Big Data project' is to create a simple end-to-end data architecture that spans the entire data lifecycle, encompassing data ingestion, transformation, and exposition. Within this project, you have the freedom to choose the specific data you wish to retrieve and determine the desired output that will generate value from the input data.

However, it is essential to adhere to a Datalake architecture, as it plays a crucial role in organizing the data effectively, ensuring a streamlined data pipeline, and facilitating data sharing among stakeholders. By following the Datalake architecture principles, we can achieve a well-structured and efficient data ecosystem

Theme for the Data

The central focus of this project revolves around gathering movies, weather data, and forecast information from multiple sources. The primary goal is to conduct a comprehensive analysis of the current top-rated, upcoming, and trending movies, considering their popularity, vote counts, release dates, and weather conditions. By incorporating weather forecast data, this analysis aims to provide valuable insights for individuals planning to visit a cinema.

By visualizing the cinema data alongside weather forecasts, stakeholders such as consumers, researchers, statisticians, and data scientists can make more informed decisions about their moviegoing experiences. The integration of weather information allows users to consider factors like temperature, precipitation, and other weather-related conditions that can impact their cinema experience.

For example, if someone prefers to enjoy outdoor activities before or after watching a movie, they can check the weather forecast to determine the best time to visit the cinema. If it's expected to rain heavily, they might opt for an indoor movie theater instead of an outdoor cinema. On the other hand, if the weather forecast indicates clear skies and pleasant temperatures, they may choose an outdoor cinema for a unique and enjoyable experience.

Moreover, the combination of cinema data and weather forecasts enables users to apply filters and preferences based on their specific moviegoing desires. They can filter movies based on genre, release date, popularity, and user ratings, and further refine their choices by considering weather conditions. For instance, individuals who prefer to watch horror movies during rainy days can easily identify suitable options by cross-referencing the cinema data with the weather forecast.

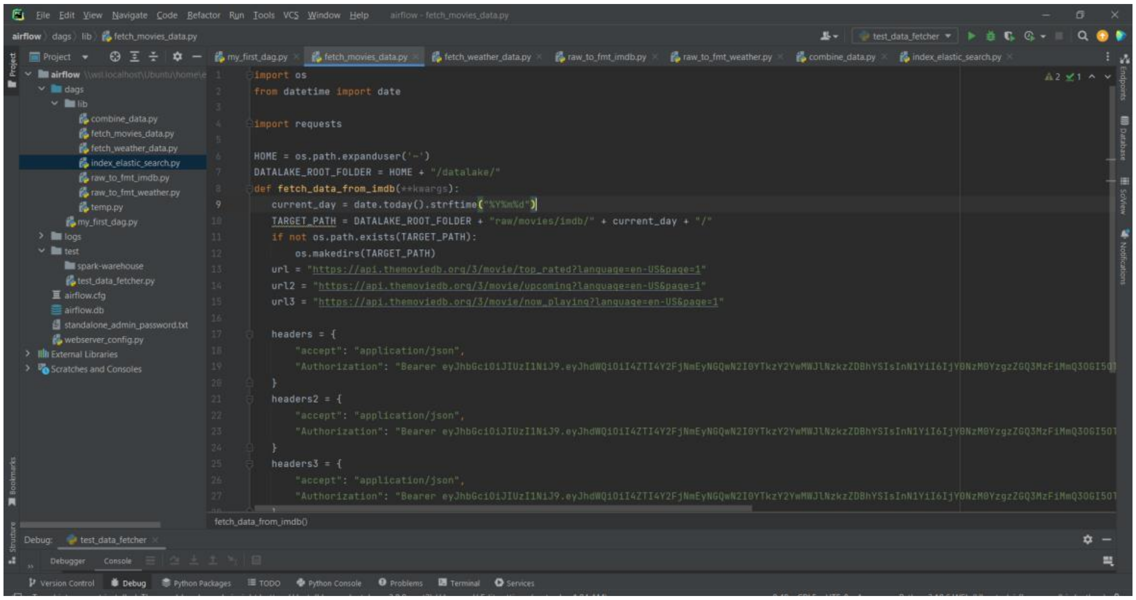
Furthermore, this enhanced visualization empowers individuals to plan their cinema outings more effectively by taking into account both movie-related factors and weather conditions. It provides a comprehensive understanding of the movie landscape and allows users to make informed decisions, ensuring an enjoyable and tailored cinema experience

Data Sources

The retrieved data would be gathered in the data lake as raw data. Data sources include:

1. The Movie Database TMDB API
2. Weather Api

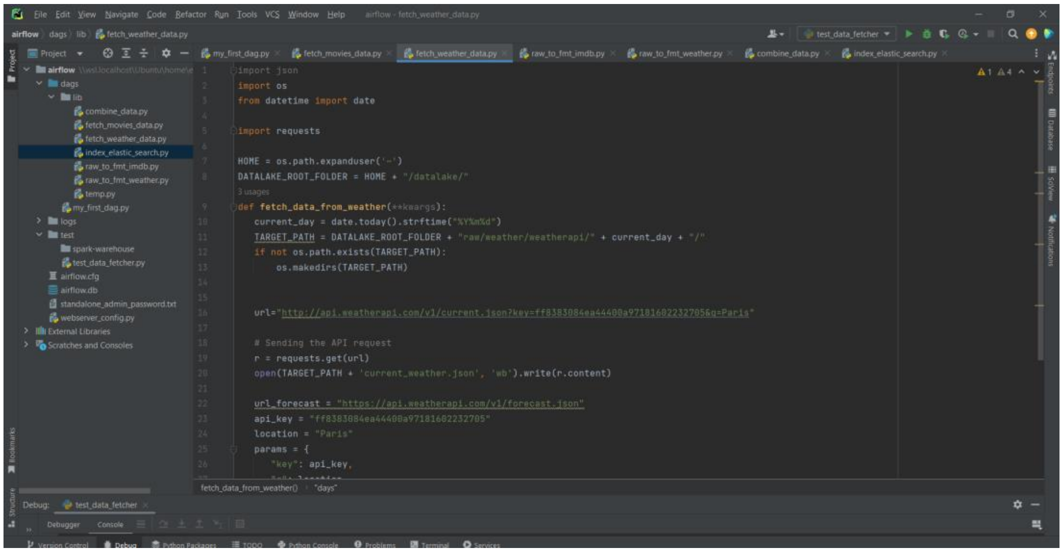
A community-built movie and television database is called The Movie Database (TMDB) API. The documentation for the API is available at https://developer.themoviedb.org/docs/getting-started , and it includes instructions on how to connect to our application and fetch data via the API. We have fetched 3 different dataset from the api, the top rated movies, the upcoming movies and trending movies in the cinema. We saved the data as a json file. While the weather api documentation can be access via https://www.weatherapi.com/docs, we fetched the current weather and forecast weather of Paris



```python
import os
from datetime import date

import requests

HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"
def fetch_data_from_imdb(**kwargs):
    current_day = date.today().strftime("%Y%m%d")
    TARGET_PATH = DATALAKE_ROOT_FOLDER + "raw/movies/imdb/" + current_day + "/"
    if not os.path.exists(TARGET_PATH):
        os.makedirs(TARGET_PATH)
    url = "https://api.themoviedb.org/3/movie/top_rated?language=en-US&page=1"
    url2 = "https://api.themoviedb.org/3/movie/upcoming?language=en-US&page=1"
    url3 = "https://api.themoviedb.org/3/movie/now_playing?language=en-US&page=1"

    headers = {
        "accept": "application/json",
        "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI4ZTI4Y2FjNmEyNGQwN2I0YTkzY2YwMWJlNzkzZDBhYSIsInN1YiI6IjY0NzM0YzgzZGQ3MzF1MmQ3OGI50
    }
    headers2 = {
        "accept": "application/json",
        "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI4ZTI4Y2FjNmEyNGQwN2I0YTkzY2YwMWJlNzkzZDBhYSIsInN1YiI6IjY0NzM0YzgzZGQ3MzF1MmQ3OGI50T
    }
    headers3 = {
        "accept": "application/json",
        "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI4ZTI4Y2FjNmEyNGQwN2I0YTkzY2YwMWJlNzkzZDBhYSIsInN1YiI6IjY0NzM0YzgzZGQ3MzF1MmQ3OGI50T
```

Figure 1 shows how to access the movie database API.

The Weather API serves as a valuable source of raw data for this project, providing real-time weather forecasts (for the next 7 days) and current weather information for Paris. By integrating this data into the project's data lake, stakeholders can explore the relationship between weather conditions and cinema trends, gaining insights into moviegoer behavior and enabling informed decision-making in Paris.



```python
import json
import os
from datetime import date

import requests

HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"

def fetch_data_from_weather(**kwargs):
    current_day = date.today().strftime("%Y%m%d")
    TARGET_PATH = DATALAKE_ROOT_FOLDER + "raw/weather/weatherapi/" + current_day + "/"
    if not os.path.exists(TARGET_PATH):
        os.makedirs(TARGET_PATH)


    url="http://api.weatherapi.com/v1/current.json?key=ff8383084ea44400a97181602232705&q=Paris"

    # Sending the API request
    r = requests.get(url)
    open(TARGET_PATH + 'current_weather.json', 'wb').write(r.content)

    url_forecast = "https://api.weatherapi.com/v1/forecast.json"
    api_key = "ff8383084ea44400a97181602232705"
    location = "Paris"
    params = {
        "key": api_key,
```

Figure 2: Connection to weather API

Data Lake Organization

We designed our data lake with a structure that adheres to the following naming convention: Each route will look like this:

/{layer}/{group}/{TableName}/{date}/filename

Layers can include: - **raw** which is the first phase of the data and is straight from the source system, the data is prepared for use by the Datalake in the second stage, which is ***formatted***, and ***usage*** is the final stage before

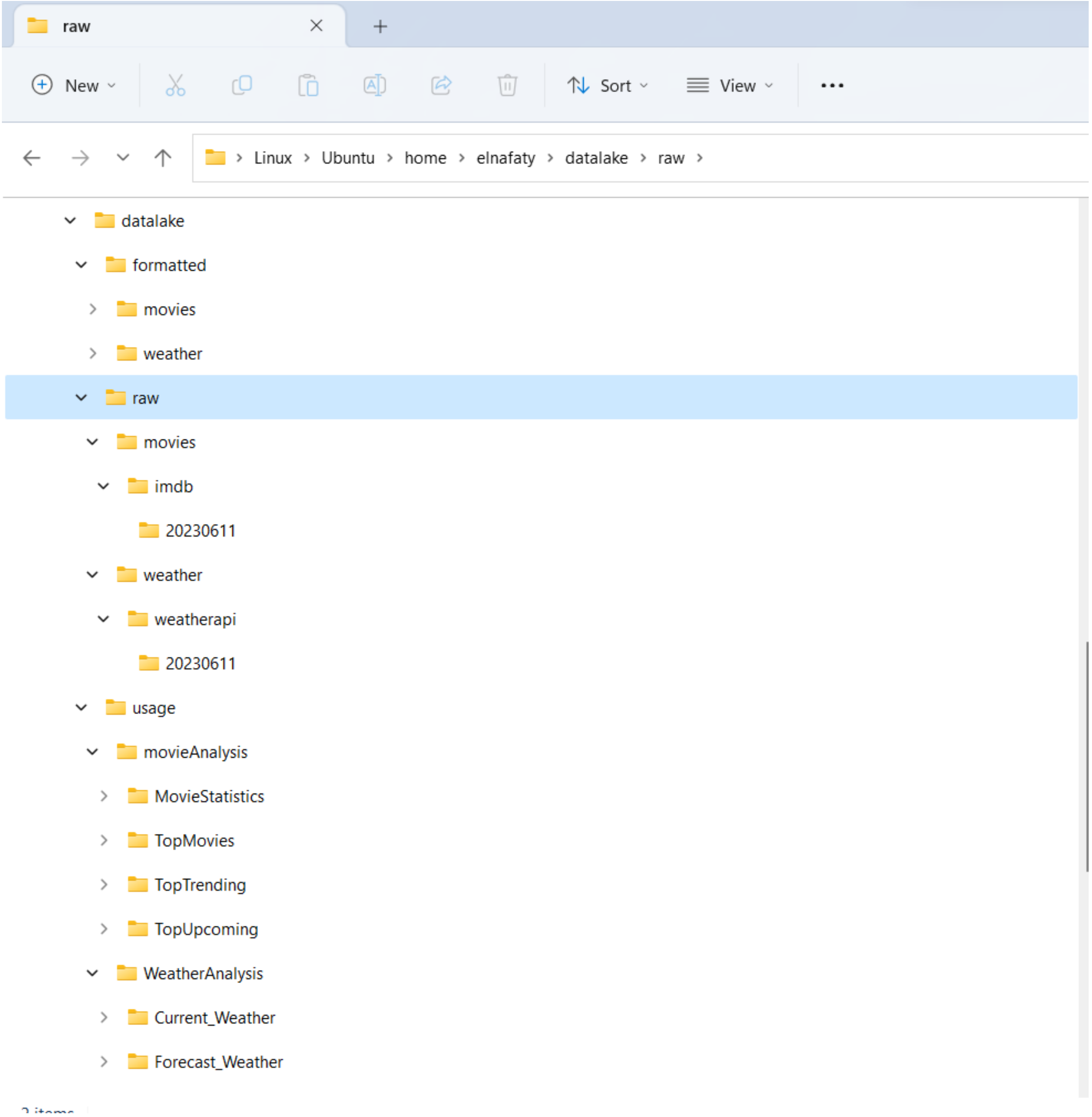
G**roup**: we use ***movies*** to store movies file and ***weather*** to save weather files

Table Name: An item in this folder will always have this name and a similar schema, in raw we use ***imdb*** for the movies and ***weatherapi*** for weather.

FileName: toprated_movies.json, upcoming_movies.json, trending_movies.json, current_weather.json and forecast.json are the file names for the raw data.

Date: should be formatted as YYYYMMDD, for instance, 20230611 (11th June 2022).

For formatted file: toprated_movies.snappy.parquet, trending_movies.snappy.parquet, upcoming_movies.snappy.parquet, current_weather.snappy.parquet and forecast.snappy.parquet.



```
datalake
  formatted
    movies
    weather
  raw
    movies
      imdb
        20230611
    weather
      weatherapi
        20230611
  usage
    movieAnalysis
      MovieStatistics
      TopMovies
      TopTrending
      TopUpcoming
    WeatherAnalysis
      Current_Weather
      Forecast_Weather
```

Figure 3: Datalake folder structure

Data Pipeline

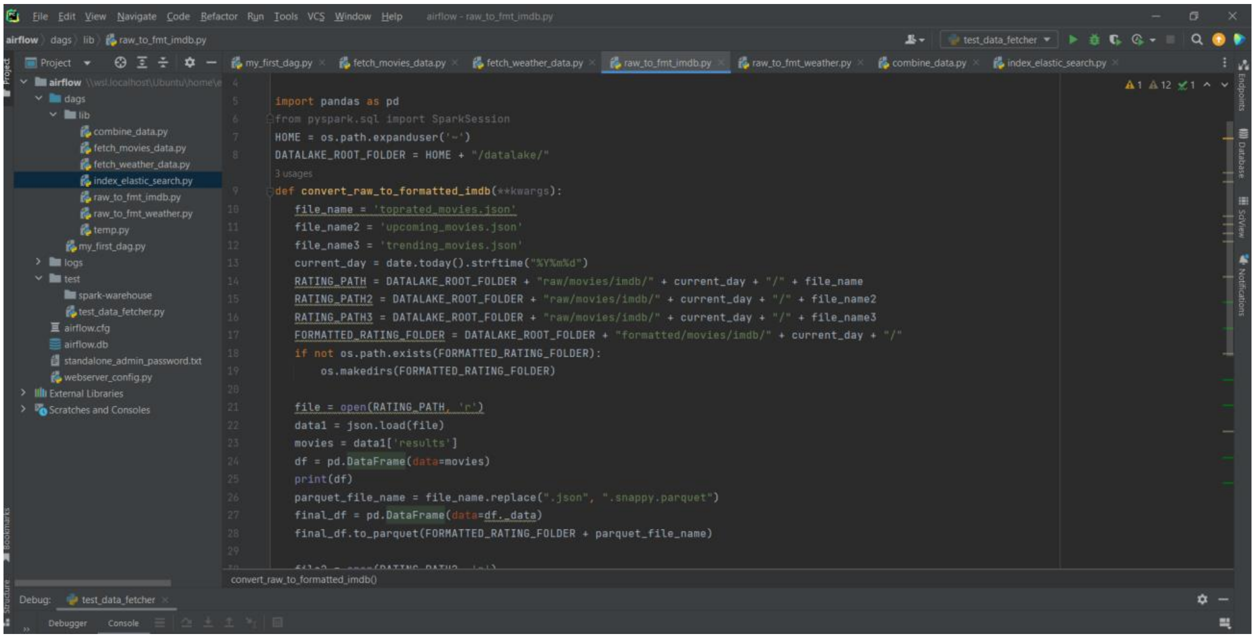
The data pipeline plays a crucial role in transferring data from its source to a designated destination, such as a data warehouse. Throughout this journey, the data undergoes transformations and optimizations, ultimately arriving in a state that allows for analysis and the development of valuable business insights. The data pipeline encompasses the essential steps of aggregating, organizing, and moving data effectively.

Data Pipeline Architecture

1. **Data Ingestion**: The source data is obtained through APIs. The data sources and ingestion process are detailed in Figure 1 and Figure 2 in the data source section. The collected data is then transferred to the raw folder in the data lake, where it can be further prepared for analysis. Python programming language is used for data ingestion in this project.

2. **Data Formatting**: In this step, the data is aggregated, cleansed, and manipulated to bring it to a standardized format suitable for analysis. Tools like Pandas are utilized to format the data like location, latitude and longitude for the weather. Figure 5 and Figure 6 illustrate the formatting process applied to data from different sources. The formatted data is then converted to the Parquet data format and compressed using snappy, which offers better processing and analysis capabilities.

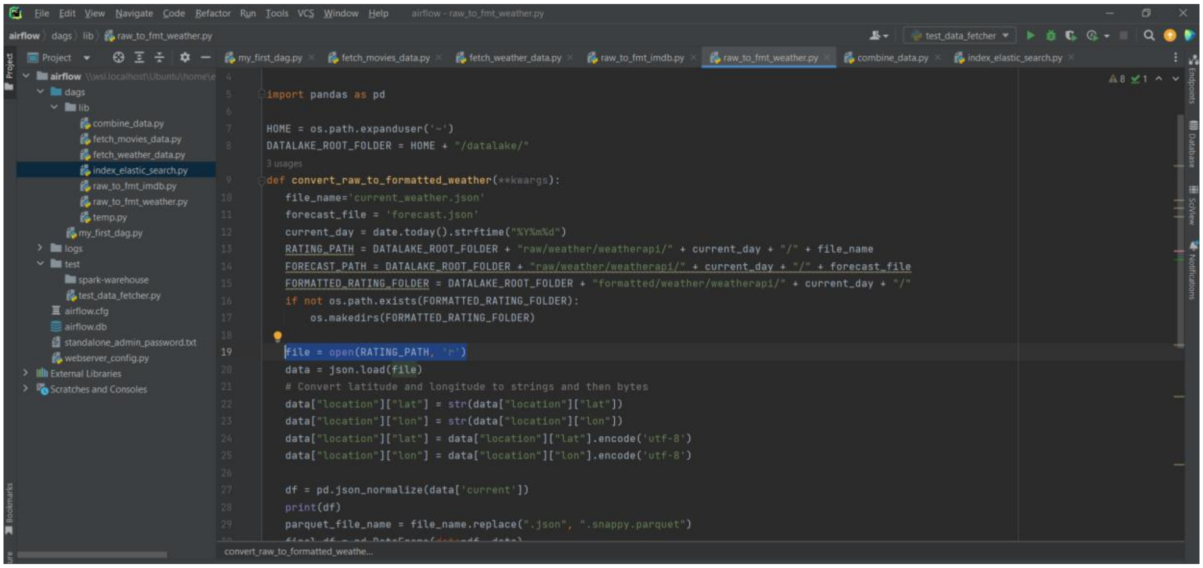
By following these steps, the data is transformed and prepared in a way that optimizes its usability for subsequent analysis and visualization.



```python
import pandas as pd
from pyspark.sql import SparkSession
HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"
3 usages
def convert_raw_to_formatted_imdb(**kwargs):
    file_name = 'toprated_movies.json'
    file_name2 = 'upcoming_movies.json'
    file_name3 = 'trending_movies.json'
    current_day = date.today().strftime("%Y%m%d")
    RATING_PATH = DATALAKE_ROOT_FOLDER + "raw/movies/imdb/" + current_day + "/" + file_name
    RATING_PATH2 = DATALAKE_ROOT_FOLDER + "raw/movies/imdb/" + current_day + "/" + file_name2
    RATING_PATH3 = DATALAKE_ROOT_FOLDER + "raw/movies/imdb/" + current_day + "/" + file_name3
    FORMATTED_RATING_FOLDER = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/"
    if not os.path.exists(FORMATTED_RATING_FOLDER):
        os.makedirs(FORMATTED_RATING_FOLDER)

    file = open(RATING_PATH, 'r')
    data1 = json.load(file)
    movies = data1['results']
    df = pd.DataFrame(data=movies)
    print(df)
    parquet_file_name = file_name.replace(".json", ".snappy.parquet")
    final_df = pd.DataFrame(data=df._data)
    final_df.to_parquet(FORMATTED_RATING_FOLDER + parquet_file_name)

    file2 = open(RATING_PATH2, 'r')
```

convert_raw_to_formatted_imdb()

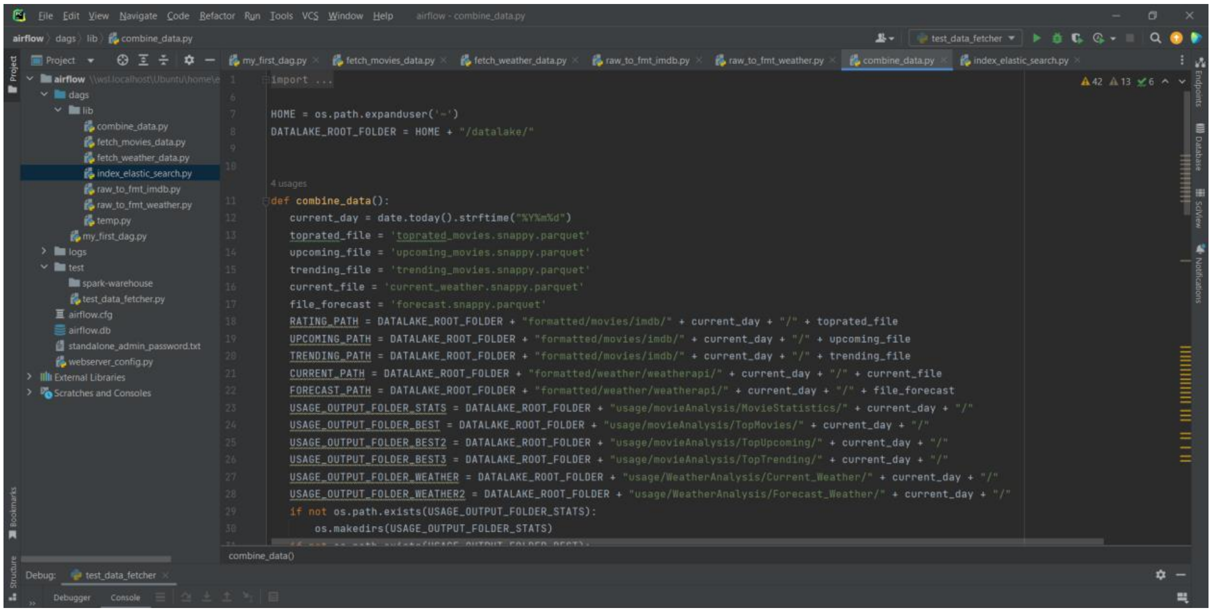
Figure 5: Formatting data from imdb with Panda



```python
import pandas as pd

HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"

def convert_raw_to_formatted_weather(**kwargs):
    file_name='current_weather.json'
    forecast_file = 'forecast.json'
    current_day = date.today().strftime("%Y%m%d")
    RATING_PATH = DATALAKE_ROOT_FOLDER + "raw/weather/weatherapi/" + current_day + "/" + file_name
    FORECAST_PATH = DATALAKE_ROOT_FOLDER + "raw/weather/weatherapi/" + current_day + "/" + forecast_file
    FORMATTED_RATING_FOLDER = DATALAKE_ROOT_FOLDER + "formatted/weather/weatherapi/" + current_day + "/"
    if not os.path.exists(FORMATTED_RATING_FOLDER):
        os.makedirs(FORMATTED_RATING_FOLDER)

    file = open(RATING_PATH, 'r')
    data = json.load(file)
    # Convert latitude and longitude to strings and then bytes
    data["location"]["lat"] = str(data["location"]["lat"])
    data["location"]["lon"] = str(data["location"]["lon"])
    data["location"]["lat"] = data["location"]["lat"].encode('utf-8')
    data["location"]["lon"] = data["location"]["lon"].encode('utf-8')

    df = pd.json_normalize(data['current'])
    print(df)
    parquet_file_name = file_name.replace(".json", ".snappy.parquet")
```

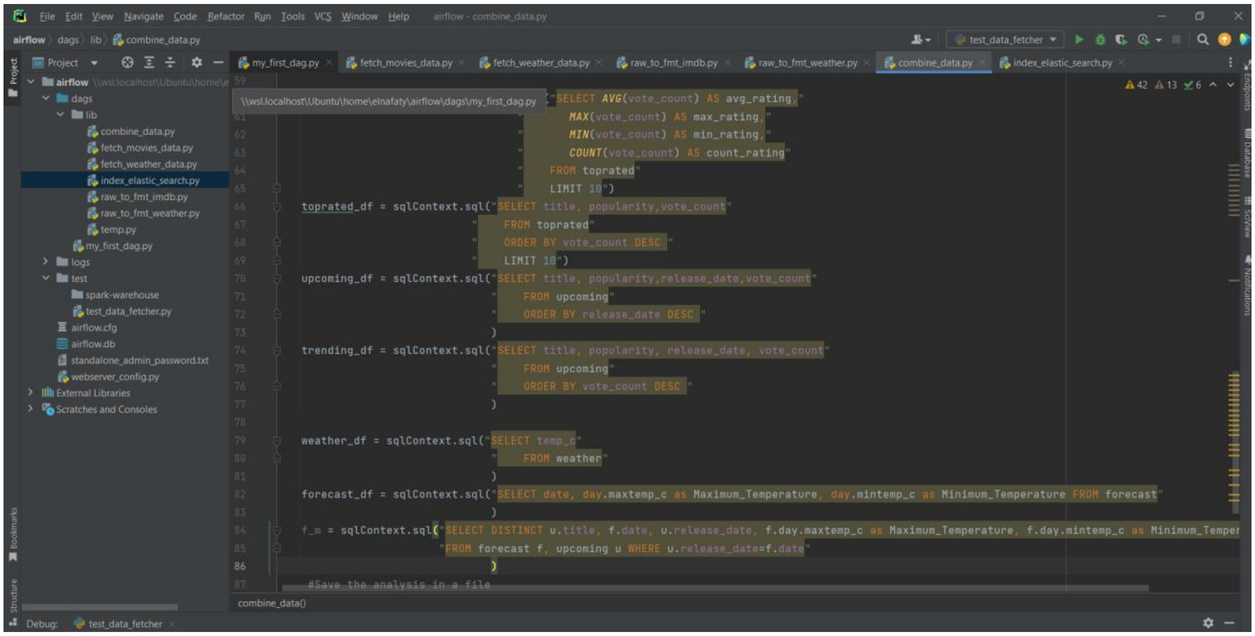
Figure 6: Formatting weather data with Panda

3. **Data Combination:** Apache Spark is utilized in this step to combine the data from different data sources and performed sql analysis. This combined data becomes the foundation for further analysis. It is indexed and made accessible through the Elasticsearch and Kibana to visualize the data, we use the cloud version of the elasticsearch and kibana



```python
import ...

HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"


def combine_data():
    current_day = date.today().strftime("%Y%m%d")
    toprated_file = 'toprated_movies.snappy.parquet'
    upcoming_file = 'upcoming_movies.snappy.parquet'
    trending_file = 'trending_movies.snappy.parquet'
    current_file = 'current_weather.snappy.parquet'
    file_forecast = 'forecast.snappy.parquet'
    RATING_PATH = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/" + toprated_file
    UPCOMING_PATH = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/" + upcoming_file
    TRENDING_PATH = DATALAKE_ROOT_FOLDER + "formatted/movies/imdb/" + current_day + "/" + trending_file
    CURRENT_PATH = DATALAKE_ROOT_FOLDER + "formatted/weather/weatherapi/" + current_day + "/" + current_file
    FORECAST_PATH = DATALAKE_ROOT_FOLDER + "formatted/weather/weatherapi/" + current_day + "/" + file_forecast
    USAGE_OUTPUT_FOLDER_STATS = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/MovieStatistics/" + current_day + "/"
    USAGE_OUTPUT_FOLDER_BEST = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/TopMovies/" + current_day + "/"
    USAGE_OUTPUT_FOLDER_BEST2 = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/TopUpcoming/" + current_day + "/"
    USAGE_OUTPUT_FOLDER_BEST3 = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/TopTrending/" + current_day + "/"
    USAGE_OUTPUT_FOLDER_WEATHER = DATALAKE_ROOT_FOLDER + "usage/WeatherAnalysis/Current_Weather/" + current_day + "/"
    USAGE_OUTPUT_FOLDER_WEATHER2 = DATALAKE_ROOT_FOLDER + "usage/WeatherAnalysis/Forecast_Weather/" + current_day + "/"
    if not os.path.exists(USAGE_OUTPUT_FOLDER_STATS):
        os.makedirs(USAGE_OUTPUT_FOLDER_STATS)
```

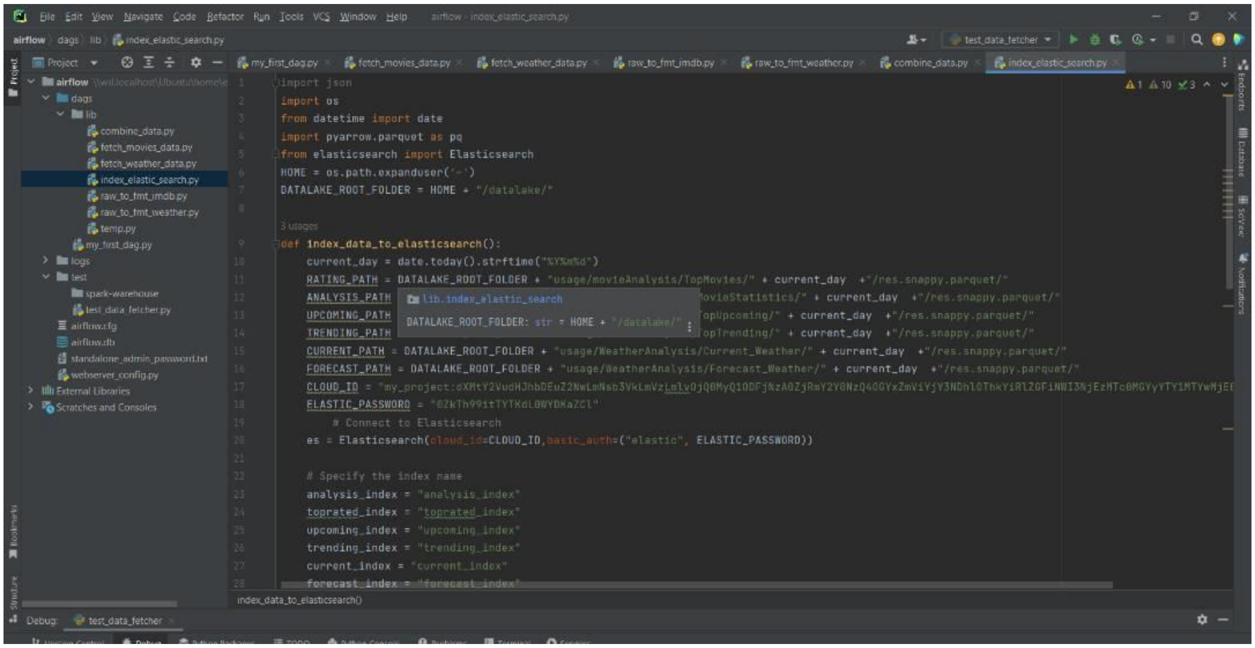
Figure 7: Data combination with Apache Spark



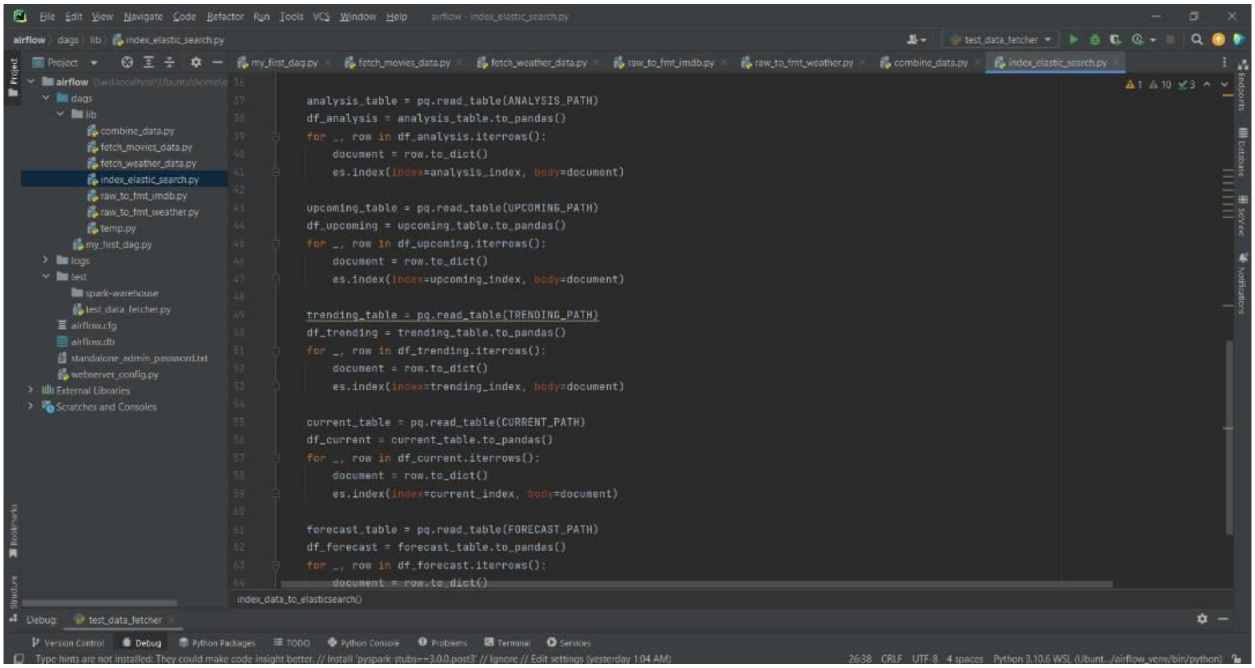
```
                                 "SELECT AVG(vote_count) AS avg_rating,"
                                 "    MAX(vote_count) AS max_rating,"
                                 "    MIN(vote_count) AS min_rating,"
                                 "    COUNT(vote_count) AS count_rating"
                                 "  FROM toprated"
                                 "  LIMIT 10")
    toprated_df = sqlContext.sql("SELECT title, popularity,vote_count"
                              "    FROM toprated"
                              "    ORDER BY vote_count DESC "
                              "    LIMIT 10")
    upcoming_df = sqlContext.sql("SELECT title, popularity,release_date,vote_count"
                                 "    FROM upcoming"
                                 "    ORDER BY release_date DESC "
                                 )
    trending_df = sqlContext.sql("SELECT title, popularity, release_date, vote_count"
                                 "    FROM upcoming"
                                 "    ORDER BY vote_count DESC "
                                 )

    weather_df = sqlContext.sql("SELECT temp_c"
                                 "    FROM weather"
                                 )
    forecast_df = sqlContext.sql("SELECT date, day.maxtemp_c as Maximum_Temperature, day.mintemp_c as Minimum_Temperature FROM forecast"
                                 )
    f_m = sqlContext.sql("SELECT DISTINCT u.title, f.date, u.release_date, f.day.maxtemp_c as Maximum_Temperature, f.day.mintemp_c as Minimum_Temper
                          "FROM forecast f, upcoming u WHERE u.release_date=f.date"
                                 )
     #Save the analysis in a file
```

Figure 8: Analysis

4. **Indexing**: In this project, the Elasticsearch Python library is utilized to facilitate the indexing process. The data is efficiently indexed and organized, enabling easy retrieval and exploration through Elasticsearch and visualization using Kibana.



```
import json
import os
from datetime import date
import pyarrow.parquet as pq
from elasticsearch import Elasticsearch
HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"

3 usages
def index_data_to_elasticsearch():
    current_day = date.today().strftime("%Y%m%d")
    RATING_PATH = DATALAKE_ROOT_FOLDER + "usage/movieAnalysis/TopMovies/" + current_day  +"/res.snappy.parquet/"
    ANALYSIS_PATH                                       ovieStatistics/" + current_day  +"/res.snappy.parquet/"
    UPCOMING_PATH                                       opUpcoming/" + current_day  +"/res.snappy.parquet/"
    TRENDING_PATH                                       opTrending/" + current_day  +"/res.snappy.parquet/"
    CURRENT_PATH = DATALAKE_ROOT_FOLDER + "usage/WeatherAnalysis/Current_Weather/" + current_day  +"/res.snappy.parquet/"
    FORECAST_PATH = DATALAKE_ROOT_FOLDER + "usage/WeatherAnalysis/Forecast_Weather/" + current_day  +"/res.snappy.parquet/"
    CLOUD_ID = "my_project:dXMtY2VudHJhbDEuZ2NwLmNsb3VkLmVzLmlvOjQ0MyQ1ODFjNzA0ZjRmY2Y0NzQ4OGYxZmViYjY3NDhlOThkYiRlZGFiNWI3NjEzMTc0MGYyYTY1MTYwMjE
    ELASTIC_PASSWORD = "0ZkTh99itTYTKdL0WYDKaZCl"
        # Connect to Elasticsearch
    es = Elasticsearch(cloud_id=CLOUD_ID,basic_auth=("elastic", ELASTIC_PASSWORD))

    # Specify the index name
    analysis_index = "analysis_index"
    toprated_index = "toprated_index"
    upcoming_index = "upcoming_index"
    trending_index = "trending_index"
    current_index = "current_index"
    forecast_index = "forecast_index"
```

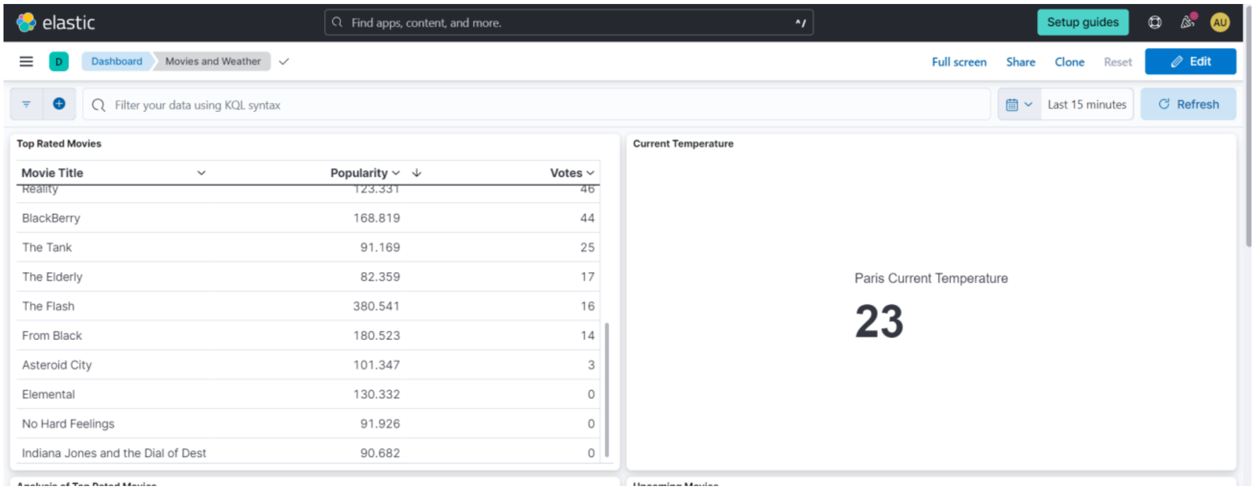


```
    analysis_table = pq.read_table(ANALYSIS_PATH)
    df_analysis = analysis_table.to_pandas()
    for _, row in df_analysis.iterrows():
        document = row.to_dict()
        es.index(index=analysis_index, body=document)

    upcoming_table = pq.read_table(UPCOMING_PATH)
    df_upcoming = upcoming_table.to_pandas()
    for _, row in df_upcoming.iterrows():
        document = row.to_dict()
        es.index(index=upcoming_index, body=document)

    trending_table = pq.read_table(TRENDING_PATH)
    df_trending = trending_table.to_pandas()
    for _, row in df_trending.iterrows():
        document = row.to_dict()
        es.index(index=trending_index, body=document)

    current_table = pq.read_table(CURRENT_PATH)
    df_current = current_table.to_pandas()
    for _, row in df_current.iterrows():
        document = row.to_dict()
        es.index(index=current_index, body=document)

    forecast_table = pq.read_table(FORECAST_PATH)
    df_forecast = forecast_table.to_pandas()
    for _, row in df_forecast.iterrows():
        document = row.to_dict()
```

```
index_data_to_elasticsearch()
```

```
Type hints are not installed: They could make code insight better. // Install 'pyspark-stubs==3.0.0.post3' // Ignore // Edit settings (yesterday 1:04 AM)
```

5. **Usage:** In this project, the prepared data is deployed to production systems, specifically Elasticsearch and Kibana. These tools serve as the backbone for analytics and visualization based on the data indexed. Multiple dashboards are created to cater to various analysis requirements and provide valuable insights. Users of the data, such as consumers, researchers, and statisticians can leverage the visually appealing and informative dashboards to perform a range of analyses. For example, if someone prefers to enjoy outdoor activities before or after watching a movie, they can check the weather forecast to determine the best time to visit the cinema. If it's expected to rain heavily, they might opt for an indoor movie theater instead of an outdoor cinema. On the other hand, if the weather forecast indicates clear skies and pleasant temperatures, they may choose an outdoor cinema for a unique and enjoyable experience. The versatility of Elasticsearch and Kibana empowers users to derive actionable insights from the data and make informed decisions across various domains.



Top Rated Movies

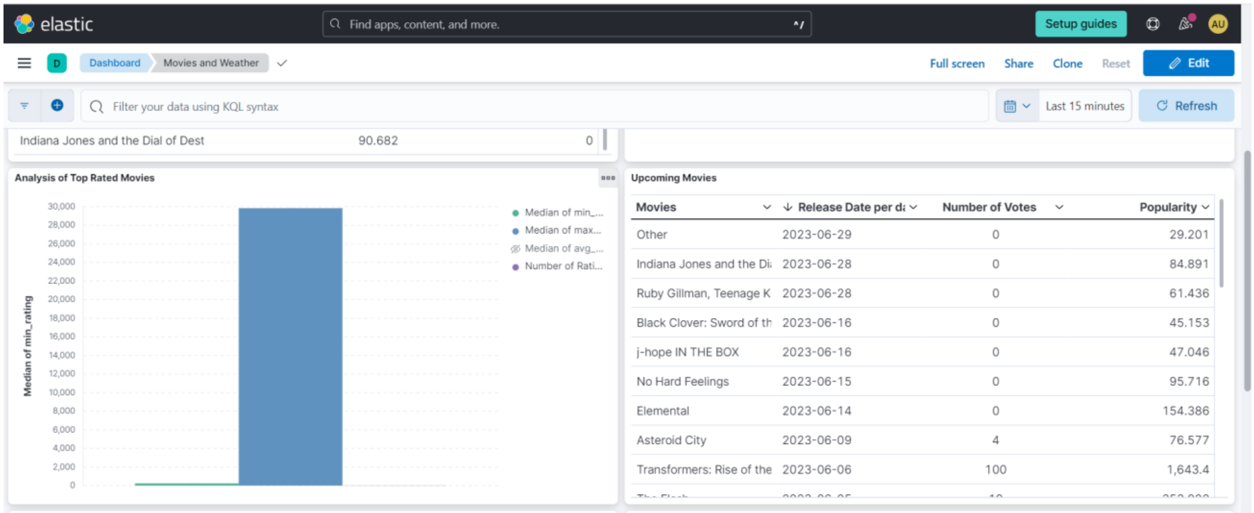
Movie Title	Popularity	Votes
Reality	123.331	46
BlackBerry	168.819	44
The Tank	91.169	25
The Elderly	82.359	17
The Flash	380.541	16
From Black	180.523	14
Asteroid City	101.347	3
Elemental	130.332	0
No Hard Feelings	91.926	0
Indiana Jones and the Dial of Dest	90.682	0

Current Temperature

Paris Current Temperature

23

Top Rated Movies & Current Temperature

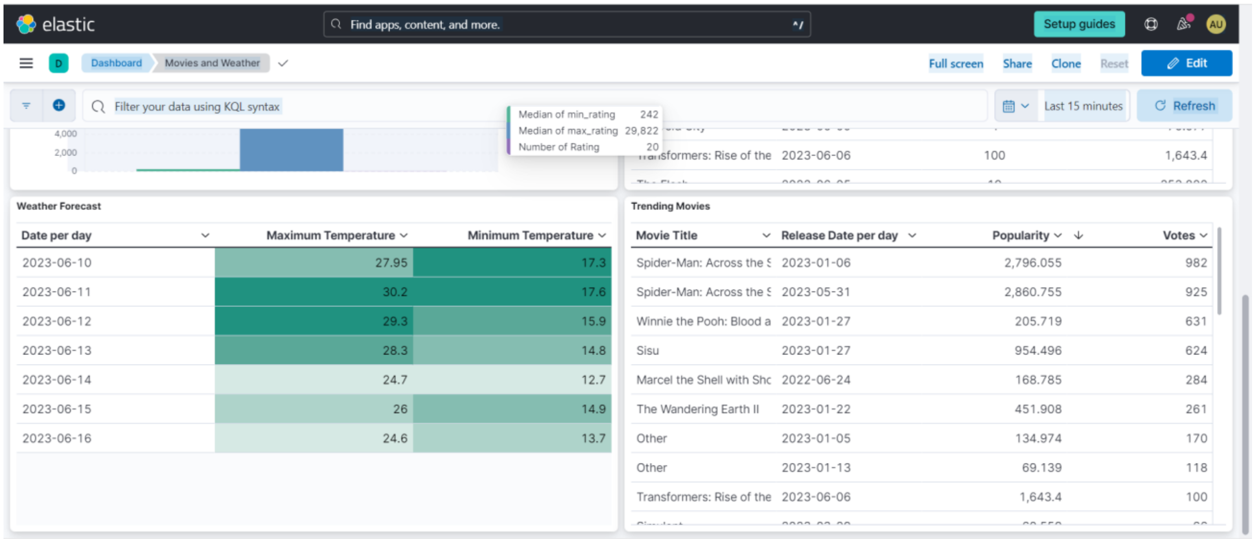


Indiana Jones and the Dial of Dest	90.682	0

Upcoming Movies

Movies	Release Date per da	Number of Votes	Popularity
Other	2023-06-29	0	29.201
Indiana Jones and the Di	2023-06-28	0	84.891
Ruby Gillman, Teenage K	2023-06-28	0	61.436
Black Clover: Sword of th	2023-06-16	0	45.153
j-hope IN THE BOX	2023-06-16	0	47.046
No Hard Feelings	2023-06-15	0	95.716
Elemental	2023-06-14	0	154.386
Asteroid City	2023-06-09	4	76.577
Transformers: Rise of the	2023-06-06	100	1,643.4

Analysis of Top Rated Movies & Upcoming Movies



Median of min_rating	242
Median of max_rating	29,822
Number of Rating	20

Transformers: Rise of the	2023-06-06	100	1,643.4

Weather Forecast

Date per day	Maximum Temperature	Minimum Temperature
2023-06-10	27.95	17.3
2023-06-11	30.2	17.6
2023-06-12	29.3	15.9
2023-06-13	28.3	14.8
2023-06-14	24.7	12.7
2023-06-15	26	14.9
2023-06-16	24.6	13.7

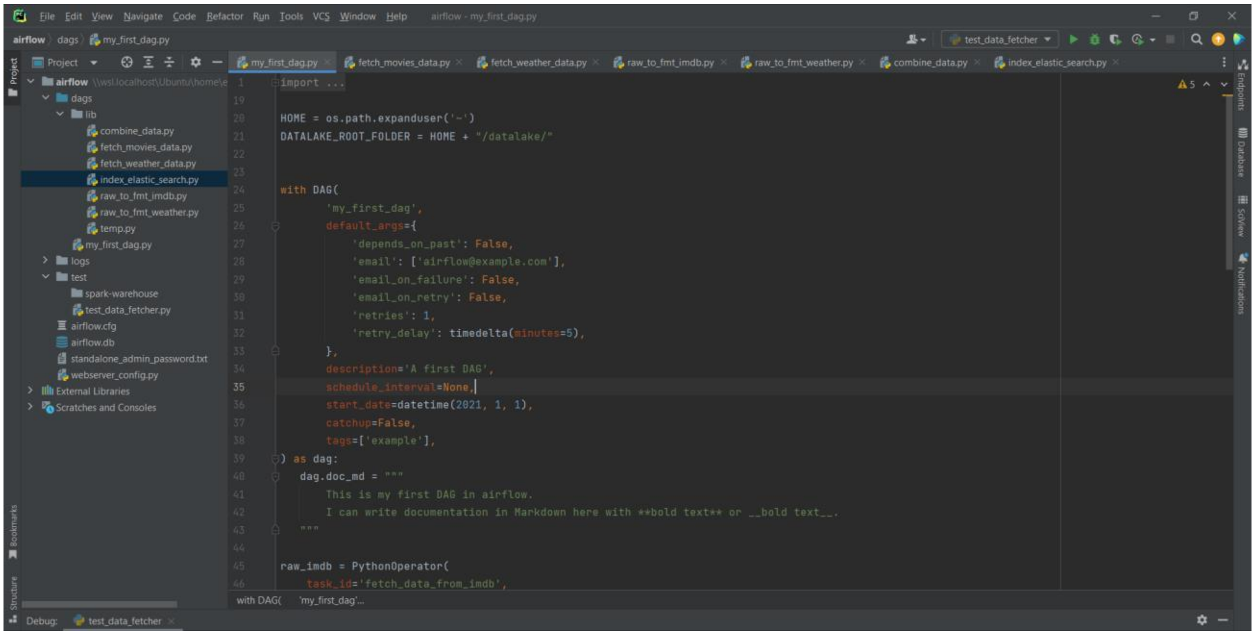
Trending Movies

Movie Title	Release Date per day	Popularity	Votes
Spider-Man: Across the S	2023-01-06	2,796.055	982
Spider-Man: Across the S	2023-05-31	2,860.755	925
Winnie the Pooh: Blood a	2023-01-27	205.719	631
Sisu	2023-01-27	954.496	624
Marcel the Shell with Shc	2022-06-24	168.785	284
The Wandering Earth II	2023-01-22	451.908	261
Other	2023-01-05	134.974	170
Other	2023-01-13	69.139	118
Transformers: Rise of the	2023-06-06	1,643.4	100

Weather Forecast & Trending Movies

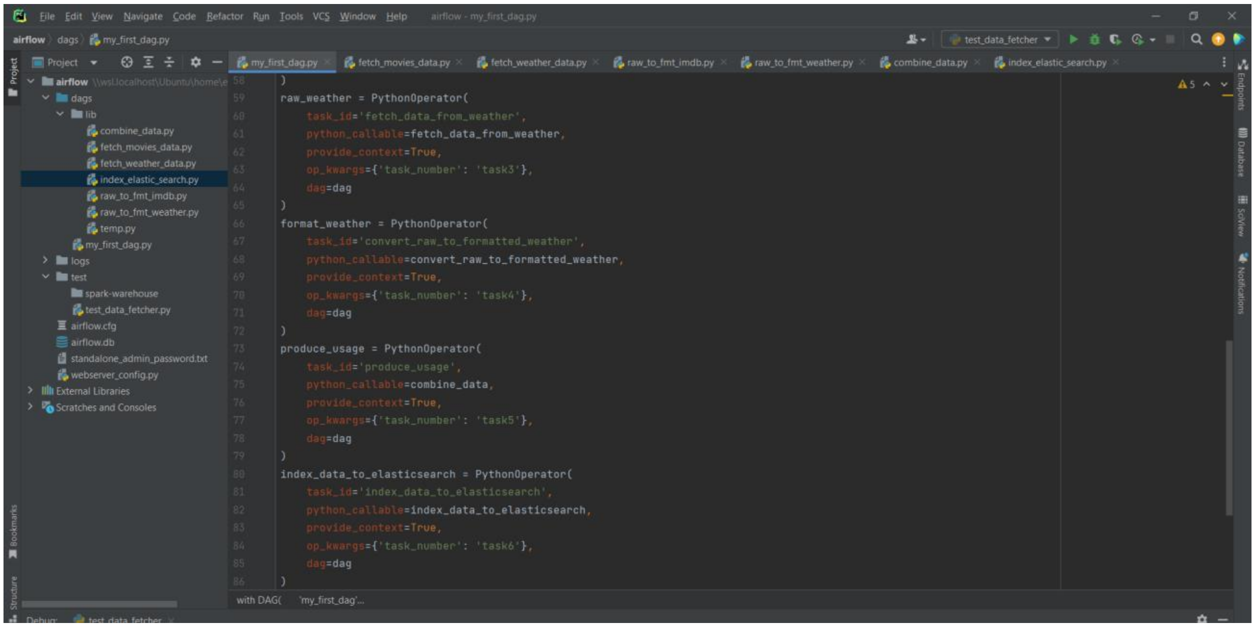
Directed Acyclic Graphs - DAGs

We use Apache Airflow to create DAGs (Directed Acyclic Graphs) to manage workflow orchestration of our data. The tasks and dependencies are defined in python (Figure 10 and 11) and Airflow manages the scheduling and execution. The DAG that reflects our pipeline architecture is shown in Figure 12.



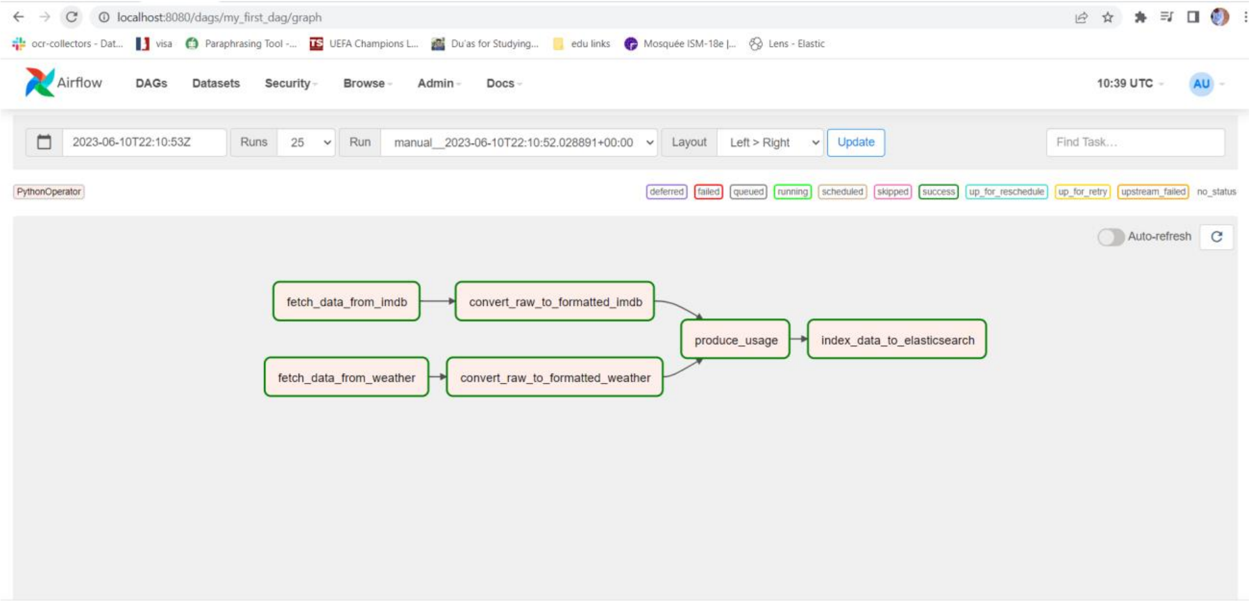
```
import ...

HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"


with DAG(
        'my_first_dag',
        default_args={
            'depends_on_past': False,
            'email': ['airflow@example.com'],
            'email_on_failure': False,
            'email_on_retry': False,
            'retries': 1,
            'retry_delay': timedelta(minutes=5),
        },
        description='A first DAG',
        schedule_interval=None,
        start_date=datetime(2021, 1, 1),
        catchup=False,
        tags=['example'],
) as dag:
    dag.doc_md = """
        This is my first DAG in airflow.
        I can write documentation in Markdown here with **bold text** or __bold text__.
    """

raw_imdb = PythonOperator(
    task_id='fetch_data_from_imdb',
```

Figure 10: DAG Phyton Code i



```
)
raw_weather = PythonOperator(
    task_id='fetch_data_from_weather',
    python_callable=fetch_data_from_weather,
    provide_context=True,
    op_kwargs={'task_number': 'task3'},
    dag=dag
)
format_weather = PythonOperator(
    task_id='convert_raw_to_formatted_weather',
    python_callable=convert_raw_to_formatted_weather,
    provide_context=True,
    op_kwargs={'task_number': 'task4'},
    dag=dag
)
produce_usage = PythonOperator(
    task_id='produce_usage',
    python_callable=combine_data,
    provide_context=True,
    op_kwargs={'task_number': 'task5'},
    dag=dag
)
index_data_to_elasticsearch = PythonOperator(
    task_id='index_data_to_elasticsearch',
    python_callable=index_data_to_elasticsearch,
    provide_context=True,
    op_kwargs={'task_number': 'task6'},
    dag=dag
)
```

Figure 11: DAG Phyton Code ii



2023-06-10T22:10:53Z Runs 25 Run manual__2023-06-10T22:10:52.028891+00:00 Layout Left > Right

Figure 12 : DAG Airflow Graph View

Conclusion

Throughout the implementation of this project, we have gained extensive knowledge and valuable experience in various technologies. Despite facing challenges along the way, the outcome has been highly successful, providing us with in-depth learnings and honing our skills.

The dynamic and comprehensive visualization of cinema and weather data offers significant benefits to consumers, researchers, statisticians, data scientists, and others, empowering them to make well-informed decisions, perform comparisons, and apply filters based on their specific needs. By considering trending movies, upcoming releases, top-rated films, user ratings, and weather forecasts for the upcoming week, stakeholders can leverage these insights for powerful recommendations.

This project has allowed us to broaden our understanding of key concepts such as big data, data lakes, data transformation, visualization, streaming, data analytics, and workflow. We have become proficient in utilizing various tools and technologies, including Apache Airflow, Apache Spark, cloud-based Elasticsearch and Kibana, Python, Panda, Parquet, and JSON, among others.