Big Data Project

TEAM MEMBERS

ASMAU AMINU 62709

YUVASRI BALU 62780

Table of Content

1. INTRODUCTION

The Big Data Project is an ambitious undertaking that aims to demonstrate the utilization of Apache Airflow, a powerful workflow management platform, in conjunction with other Big Data technologies to create an end-to-end data pipeline. This project involves extracting data from external sources, preparing and formatting the data, performing data analysis using Apache Spark, and ultimately generating valuable insights.
The project will be divided into several episodes, each focusing on specific aspects of the overall data pipeline. The episodes will cover the fundamentals of Airflow, hands-on exercises to set up and familiarize oneself with Airflow, creating custom Directed Acyclic Graphs (DAGs) to orchestrate tasks, and integrating the project with Git for version control.

In later episodes, the project will delve into extracting data from mediaStack and exchangerate APIs, preparing and formatting the data into structured layers, and combining multiple datasets for comprehensive analysis. Apache Spark, a distributed computing system, will be utilized to process and analyze the data, generating meaningful insights and output.
hroughout the report, we will provide detailed documentation, step-by-step instructions, and code samples to guide you through each episode of the project.

Data Source and API

Free foreign exchange, crypto rates &EU VAT Rates API

https://exchangerate.host/#/#docs

It shows the exchange rate of many currencies with euro as a base.

News API

https://mediastack.com/documentation

It contains daily news with author, tittle, description, date.

Structure of the Datalake

`datalake/`: The root directory of the data lake.
`raw/`: Contains the raw data obtained directly from the APIs.
`exchangerates/`: Stores the raw data from the Exchangerates API.
`money/`: Stores the exchange rate data obtained from the Exchangerates API.
`<current_day>/`: Subdirectory for each day the data is fetched.
`<raw_data_files>`: Raw data files in JSON format (e.g., **`money.json`**).
`mediastack/`: Stores the raw data from the Mediastack API.
`news/`: Stores the news data obtained from the Mediastack API.
`<current_day>/`: Subdirectory for each day the data is fetched.
`<raw_data_files>`: Raw data files in JSON format (e.g., **`news.json`**).
`formatted/`: Contains the processed and formatted data.
`exchangerates/`: Stores the formatted data of exchange rates.
`money/`: Stores the formatted exchange rate data.
`<current_day>/`: Subdirectory for each day the data is processed.
`<formatted_data_files>`: Formatted data files in Parquet format (e.g., **`money.snappy.parquet`**).
`mediastack/`: Stores the formatted data of news.

`news/`: Stores the formatted news data.
`<current_day>/`: Subdirectory for each day the data is processed.
`<formatted_data_files>`: Formatted data files in Parquet format (e.g., `news.snappy.parquet`).
`usage/`: Contains the analyzed and indexed data for usage analysis.
`exchangeAnalysis/`: Stores the analyzed data related to exchange rates.
`ExchangedStatistics/`: Stores the analyzed exchange rate statistics.
`<current_day>/`: Subdirectory for each day the data is analyzed.
`<analyzed_data_files>`: Analyzed data files in Parquet format (e.g., `res.snappy.parquet`).
`newsAnalysis/`: Stores the analyzed data related to news.
`NewsAnalysis/`: Stores the analyzed news data.
`<current_day>/`: Subdirectory for each day the data is analyzed.
`<analyzed_data_files>`: Analyzed data files in Parquet format (e.g., `res.snappy.parquet`).

Pipeline

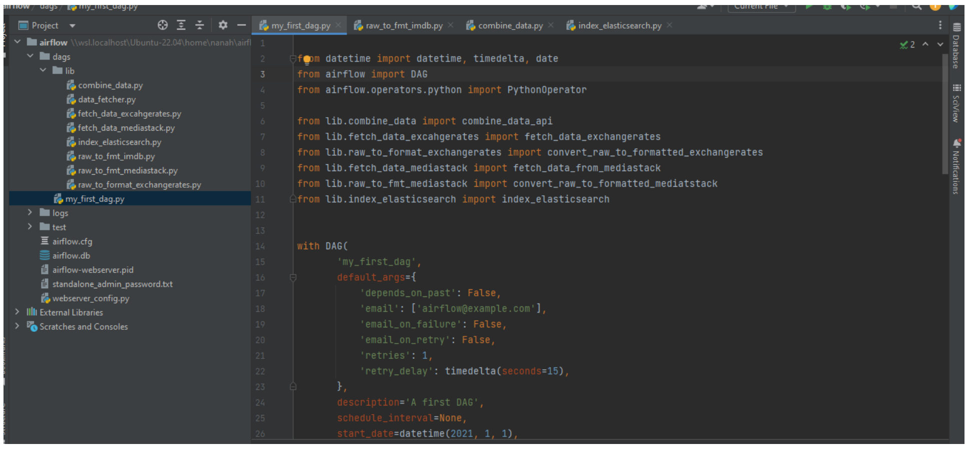
- The pipeline of this project involves several steps to collect, process, and analyze data. Let's walk through the pipeline step by step:
- **Fetching Data from Exchangerates API:**
- The fetch_data_exchangerates function is executed as a task in the pipeline.
- This function makes an HTTP request to the Exchangerates API to retrieve the latest exchange rate data.
- The data is downloaded and stored in the raw data folder of the data lake.
- **Converting Raw Exchangerates Data to Formatted Data:**
- The convert_raw_to_formatted_exchangerates function is executed as a task in the pipeline.
- It takes the raw data obtained from the Exchangerates API and processes it to create a formatted representation.
- The raw data in JSON format is converted to a Pandas DataFrame and then saved as a Parquet file in the formatted data folder of the data lake.
- **Fetching Data from Mediastack API:**

- The fetch_data_from_mediastack function is executed as a task in the pipeline.
- It makes an HTTP request to the Mediastack API to retrieve news articles and headlines.
- The data is downloaded and stored in the raw data folder of the data lake.
- **Converting Raw Mediastack Data to Formatted Data:**
- The convert_raw_to_formatted_mediatstack function is executed as a task in the pipeline.
- It processes the raw news data obtained from the Mediastack API and converts it to a formatted representation.
- The raw JSON data is transformed into a Pandas DataFrame and then saved as a Parquet file in the formatted data folder of the data lake.
- **Combining and Analyzing Data:**
- The combine_data_api function is executed as a task in the pipeline.
- It combines the formatted exchange rate data and news data from the data lake to perform data analysis.
- The Spark framework is used to analyze the data, extract relevant information, and generate insights.

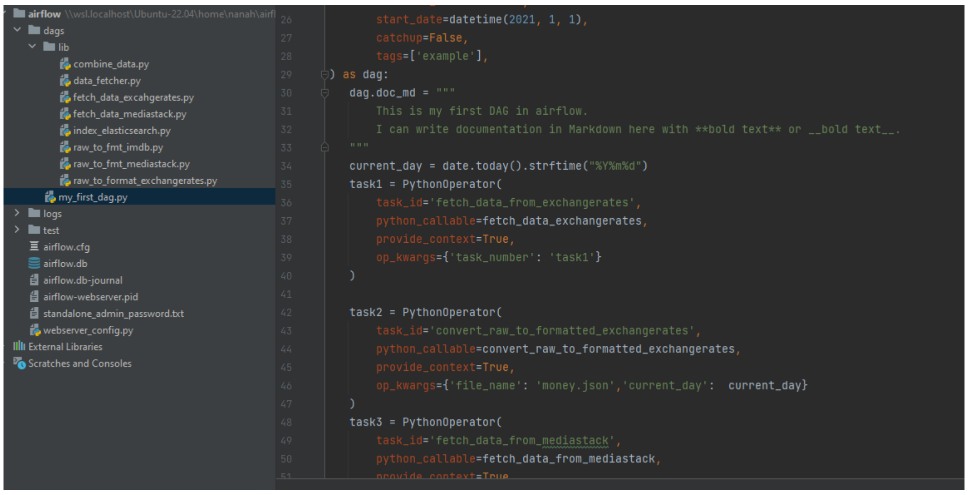
Indexing Data into Elasticsearch:

- The index_elasticsearch function is executed as a task in the pipeline.
- It connects to Elasticsearch and indexes the analyzed data into appropriate indices.
- The data is transformed into JSON records and bulk indexed into Elasticsearch for efficient searching and querying.

2. Create our first DAGS

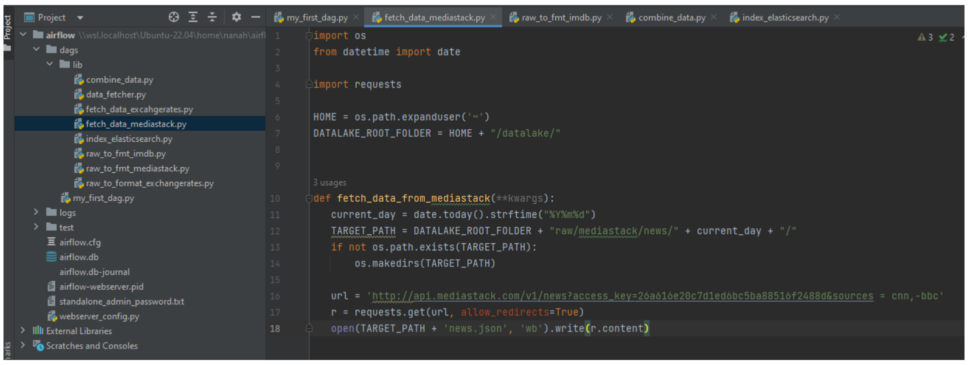


```python

from datetime import datetime, timedelta, date
from airflow import DAG
from airflow.operators.python import PythonOperator

from lib.combine_data import combine_data_api
from lib.fetch_data_excahgerates import fetch_data_exchangerates
from lib.raw_to_format_exchangerates import convert_raw_to_formatted_exchangerates
from lib.fetch_data_mediastack import fetch_data_from_mediastack
from lib.raw_to_fmt_mediastack import convert_raw_to_formatted_mediatstack
from lib.index_elasticsearch import index_elasticsearch


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(seconds=15),
        },
        description='A first DAG',
        schedule_interval=None,
        start_date=datetime(2021, 1, 1),
```



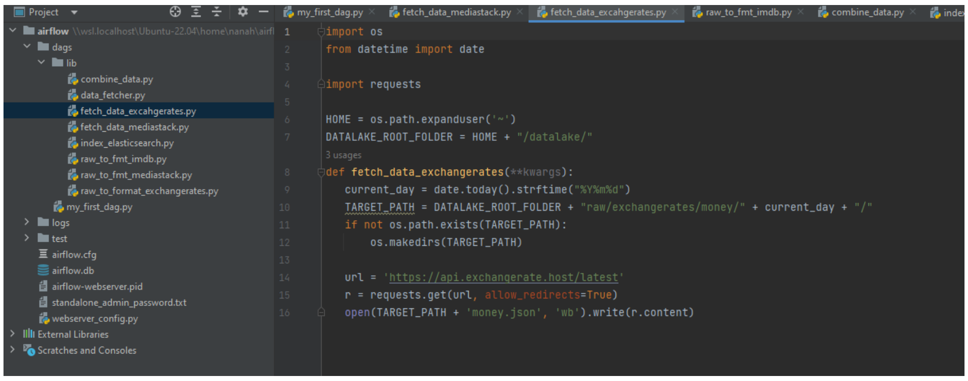
```python
        start_date=datetime(2021, 1, 1),
        catchup=False,
        tags=['example'],
) as dag:
    dag.doc_md = """
        This is my first DAG in airflow.
        I can write documentation in Markdown here with **bold text** or __bold text__.
    """
    current_day = date.today().strftime("%Y%m%d")
    task1 = PythonOperator(
        task_id='fetch_data_from_exchangerates',
        python_callable=fetch_data_exchangerates,
        provide_context=True,
        op_kwargs={'task_number': 'task1'}
    )

    task2 = PythonOperator(
        task_id='convert_raw_to_formatted_exchangerates',
        python_callable=convert_raw_to_formatted_exchangerates,
        provide_context=True,
        op_kwargs={'file_name': 'money.json','current_day':  current_day}
    )
    task3 = PythonOperator(
        task_id='fetch_data_from_mediastack',
        python_callable=fetch_data_from_mediastack,
        provide_context=True
```

3. Big Data Project - Extract source from Media Stack



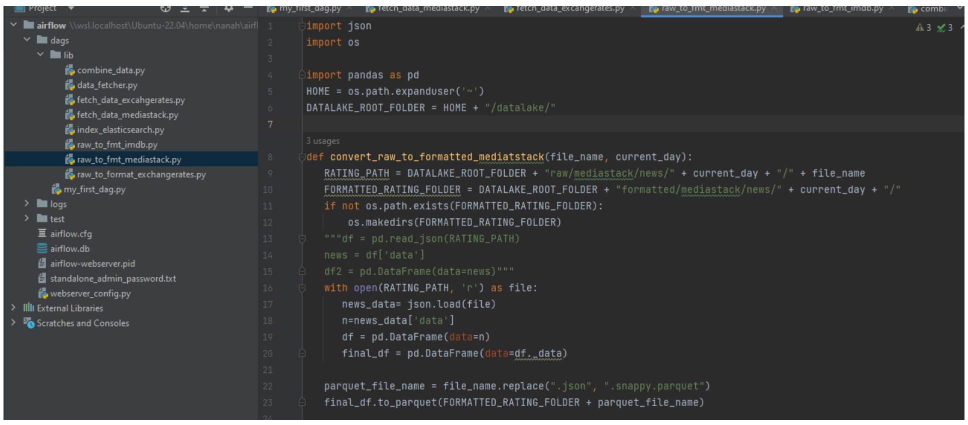
```python
import os
from datetime import date

import requests

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



3 usages
def fetch_data_from_mediastack(**kwargs):
   current_day = date.today().strftime("%Y%m%d")
   TARGET_PATH = DATALAKE_ROOT_FOLDER + "raw/mediastack/news/" + current_day + "/"
   if not os.path.exists(TARGET_PATH):
       os.makedirs(TARGET_PATH)

   url = 'http://api.mediastack.com/v1/news?access_key=26a616e20c7d1ed6bc5ba88516f2488d&sources = cnn,-bbc'
   r = requests.get(url, allow_redirects=True)
   open(TARGET_PATH + 'news.json', 'wb').write(r.content)
```

4. Big Data Project - Extract source from Exchange rate

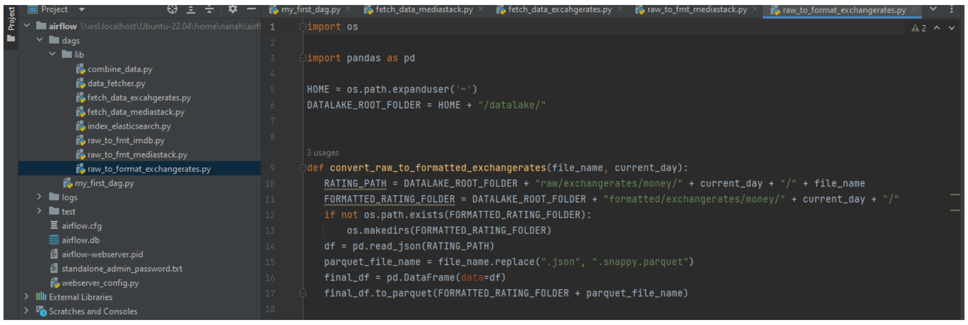


```python
import os
from datetime import date

import requests

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

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

    url = 'https://api.exchangerate.host/latest'
    r = requests.get(url, allow_redirects=True)
    open(TARGET_PATH + 'money.json', 'wb').write(r.content)
```

5. Big Data Project - Prepare formatted data

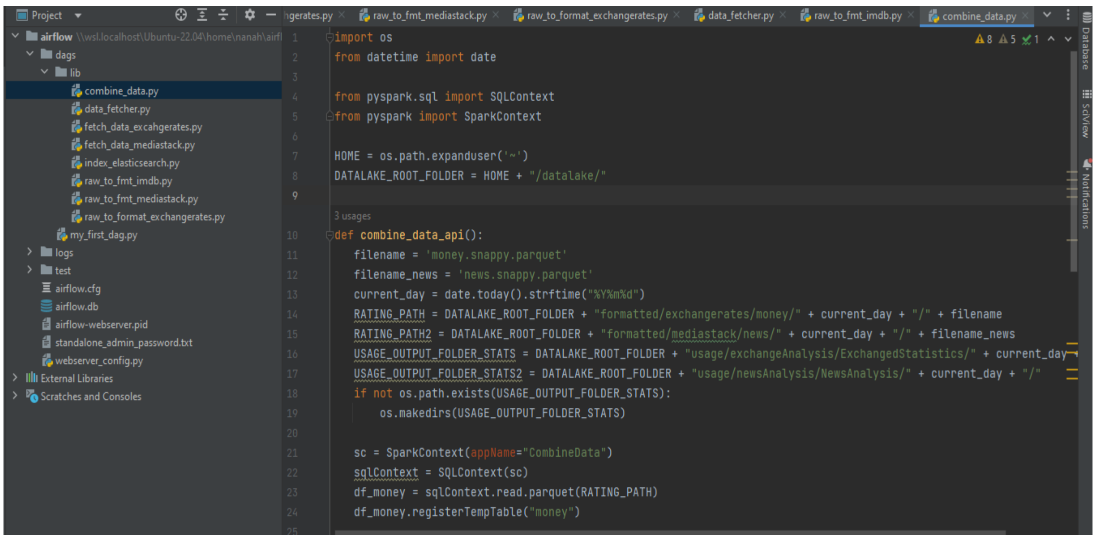


```python
import json
import os

import pandas as pd
HOME = os.path.expanduser('~')
DATALAKE_ROOT_FOLDER = HOME + "/datalake/"

def convert_raw_to_formatted_mediatstack(file_name, current_day):
    RATING_PATH = DATALAKE_ROOT_FOLDER + "raw/mediastack/news/" + current_day + "/" + file_name
    FORMATTED_RATING_FOLDER = DATALAKE_ROOT_FOLDER + "formatted/mediastack/news/" + current_day + "/"
    if not os.path.exists(FORMATTED_RATING_FOLDER):
        os.makedirs(FORMATTED_RATING_FOLDER)
    """df = pd.read_json(RATING_PATH)
    news = df['data']
    df2 = pd.DataFrame(data=news)"""
    with open(RATING_PATH, 'r') as file:
        news_data= json.load(file)
        n=news_data['data']
        df = pd.DataFrame(data=n)
        final_df = pd.DataFrame(data=df._data)

    parquet_file_name = file_name.replace(".json", ".snappy.parquet")
    final_df.to_parquet(FORMATTED_RATING_FOLDER + parquet_file_name)
```

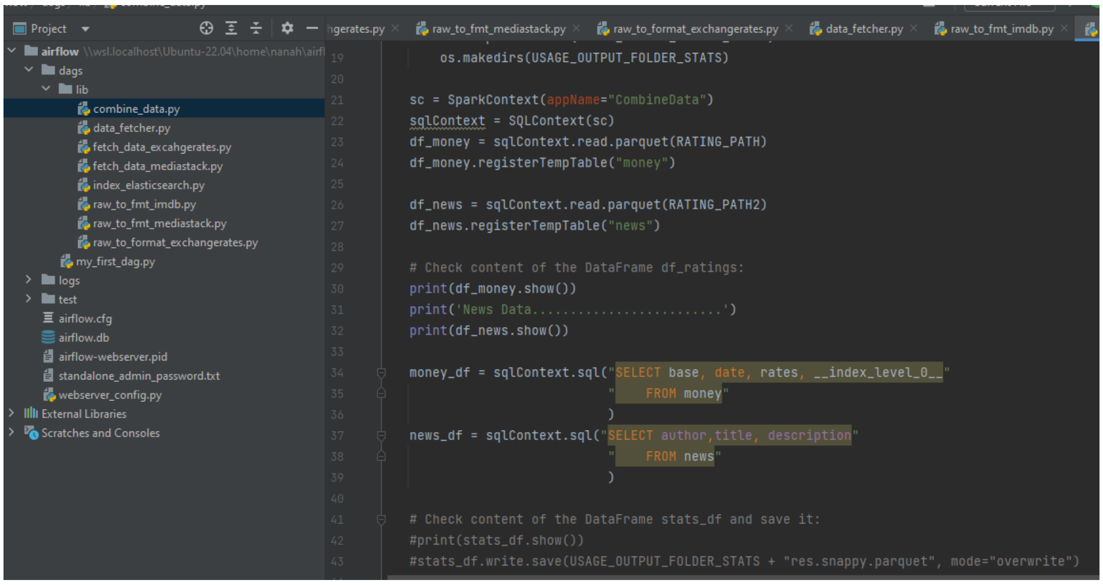


```python
import os

import pandas as pd

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


def convert_raw_to_formatted_exchangerates(file_name, current_day):
    RATING_PATH = DATALAKE_ROOT_FOLDER + "raw/exchangerates/money/" + current_day + "/" + file_name
    FORMATTED_RATING_FOLDER = DATALAKE_ROOT_FOLDER + "formatted/exchangerates/money/" + current_day + "/"
    if not os.path.exists(FORMATTED_RATING_FOLDER):
        os.makedirs(FORMATTED_RATING_FOLDER)
    df = pd.read_json(RATING_PATH)
    parquet_file_name = file_name.replace(".json", ".snappy.parquet")
    final_df = pd.DataFrame(data=df)
    final_df.to_parquet(FORMATTED_RATING_FOLDER + parquet_file_name)
```

6. Big Data Project - Prepare combined data

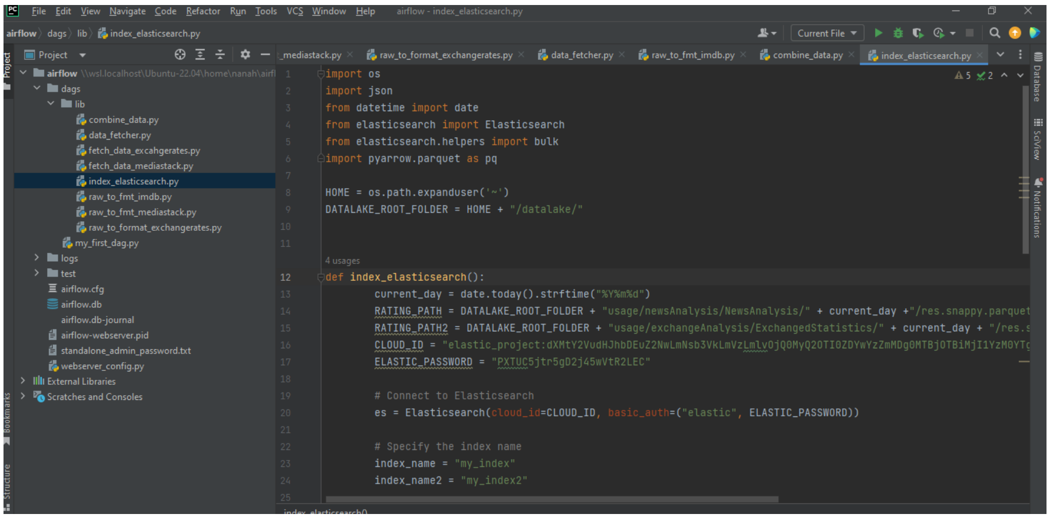


```python
import os
from datetime import date

from pyspark.sql import SQLContext
from pyspark import SparkContext

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

def combine_data_api():
    filename = 'money.snappy.parquet'
    filename_news = 'news.snappy.parquet'
    current_day = date.today().strftime("%Y%m%d")
    RATING_PATH = DATALAKE_ROOT_FOLDER + "formatted/exchangerates/money/" + current_day + "/" + filename
    RATING_PATH2 = DATALAKE_ROOT_FOLDER + "formatted/mediastack/news/" + current_day + "/" + filename_news
    USAGE_OUTPUT_FOLDER_STATS = DATALAKE_ROOT_FOLDER + "usage/exchangeAnalysis/ExchangedStatistics/" + current_day
    USAGE_OUTPUT_FOLDER_STATS2 = DATALAKE_ROOT_FOLDER + "usage/newsAnalysis/NewsAnalysis/" + current_day + "/"
    if not os.path.exists(USAGE_OUTPUT_FOLDER_STATS):
        os.makedirs(USAGE_OUTPUT_FOLDER_STATS)

    sc = SparkContext(appName="CombineData")
    sqlContext = SQLContext(sc)
    df_money = sqlContext.read.parquet(RATING_PATH)
    df_money.registerTempTable("money")
```

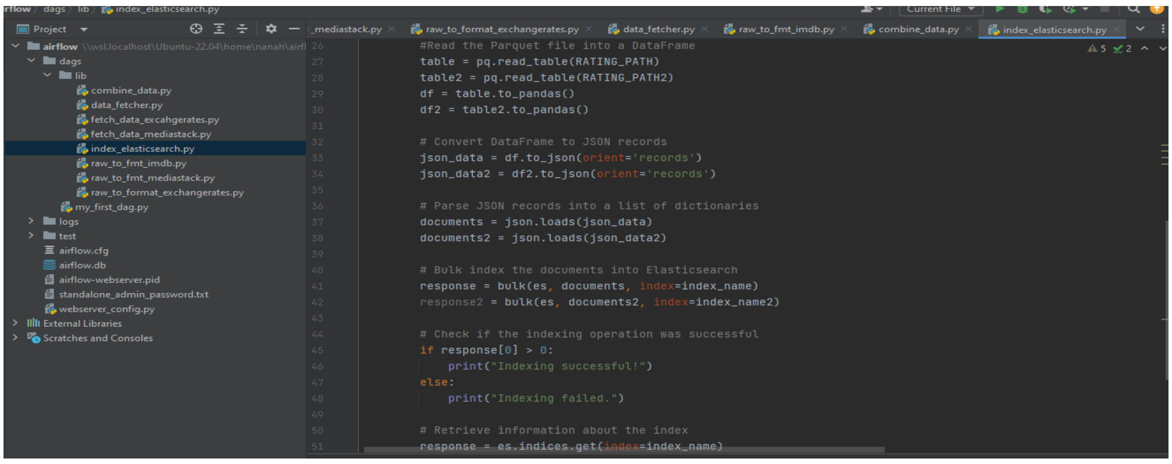


```
        os.makedirs(USAGE_OUTPUT_FOLDER_STATS)

    sc = SparkContext(appName="CombineData")
    sqlContext = SQLContext(sc)
    df_money = sqlContext.read.parquet(RATING_PATH)
    df_money.registerTempTable("money")

    df_news = sqlContext.read.parquet(RATING_PATH2)
    df_news.registerTempTable("news")

    # Check content of the DataFrame df_ratings:
    print(df_money.show())
    print('News Data.........................')
    print(df_news.show())

    money_df = sqlContext.sql("SELECT base, date, rates, __index_level_0__"
                              "    FROM money"
                              )

    news_df = sqlContext.sql("SELECT author,title, description"
                             "    FROM news"
                             )

    # Check content of the DataFrame stats_df and save it:
    #print(stats_df.show())
    #stats_df.write.save(USAGE_OUTPUT_FOLDER_STATS + "res.snappy.parquet", mode="overwrite")
```

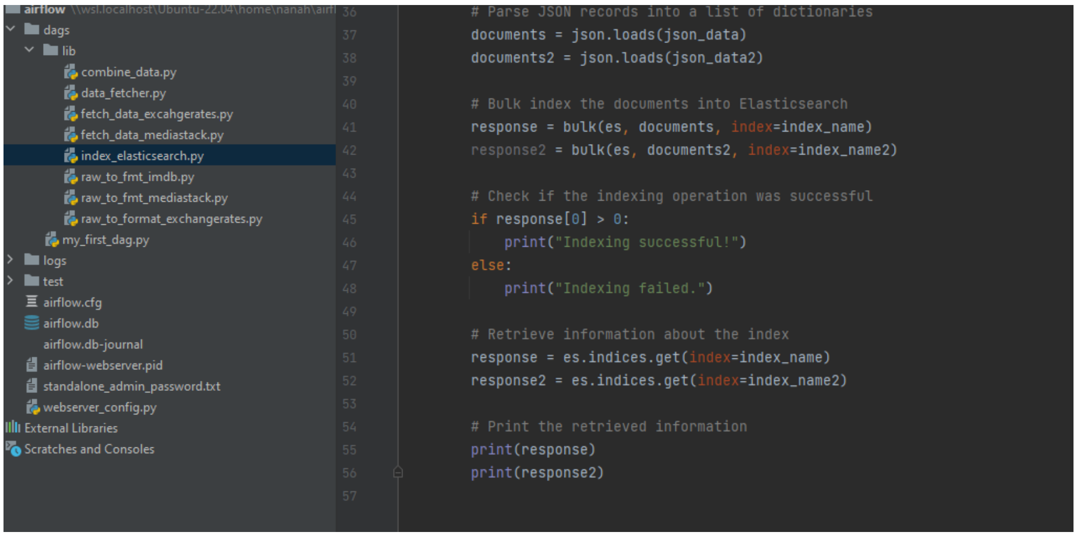
7. Big Data Project - Ingest into EK



```
import os
import json
from datetime import date
from elasticsearch import Elasticsearch
from elasticsearch.helpers import bulk
import pyarrow.parquet as pq

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


def index_elasticsearch():
        current_day = date.today().strftime("%Y%m%d")
        RATING_PATH = DATALAKE_ROOT_FOLDER + "usage/newsAnalysis/NewsAnalysis/" + current_day +"/res.snappy.parquet
        RATING_PATH2 = DATALAKE_ROOT_FOLDER + "usage/exchangeAnalysis/ExchangedStatistics/" + current_day + "/res.s
        CLOUD_ID = "elastic_project:dXMtY2VudHJhbDEuZ2NwLmNsb3VkLmVzLmlvOjQ0MyQ2OTI0ZDYwYzZmMDg0MTBjOTBiMjI1YzM0YTg
        ELASTIC_PASSWORD = "PXTUC5jtr5gD2j45wVtR2LEC"

        # Connect to Elasticsearch
        es = Elasticsearch(cloud_id=CLOUD_ID, basic_auth=("elastic", ELASTIC_PASSWORD))

        # Specify the index name
        index_name = "my_index"
        index_name2 = "my_index2"
```

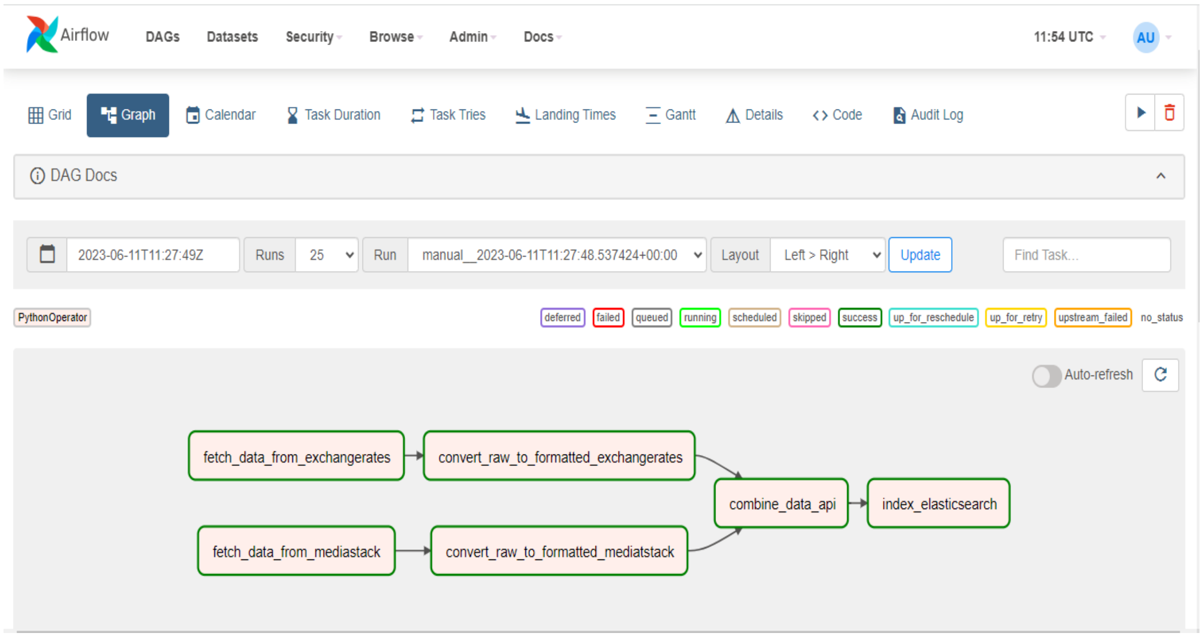


```
    #Read the Parquet file into a DataFrame
    table = pq.read_table(RATING_PATH)
    table2 = pq.read_table(RATING_PATH2)
    df = table.to_pandas()
    df2 = table2.to_pandas()

    # Convert DataFrame to JSON records
    json_data = df.to_json(orient='records')
    json_data2 = df2.to_json(orient='records')

    # Parse JSON records into a list of dictionaries
    documents = json.loads(json_data)
    documents2 = json.loads(json_data2)

    # Bulk index the documents into Elasticsearch
    response = bulk(es, documents, index=index_name)
    response2 = bulk(es, documents2, index=index_name2)

    # Check if the indexing operation was successful
    if response[0] > 0:
        print("Indexing successful!")
    else:
        print("Indexing failed.")

    # Retrieve information about the index
    response = es.indices.get(index=index_name)
```



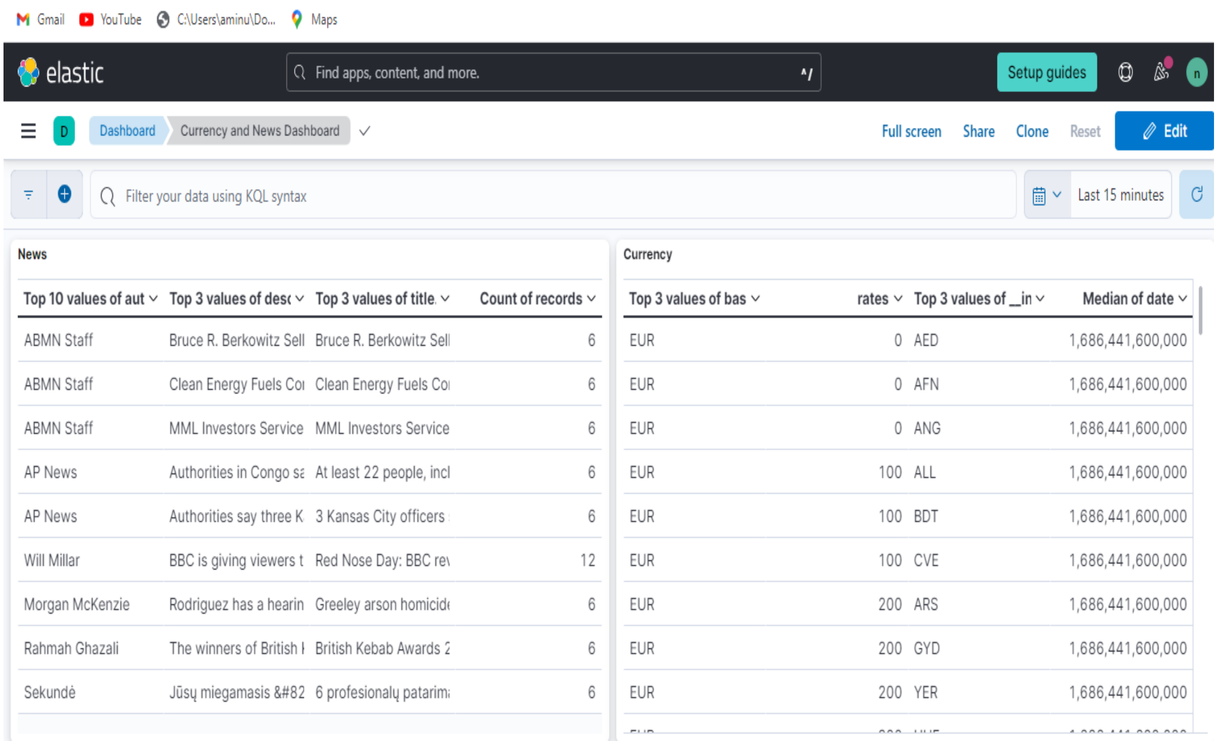
```
    # Parse JSON records into a list of dictionaries
    documents = json.loads(json_data)
    documents2 = json.loads(json_data2)

    # Bulk index the documents into Elasticsearch
    response = bulk(es, documents, index=index_name)
    response2 = bulk(es, documents2, index=index_name2)

    # Check if the indexing operation was successful
    if response[0] > 0:
        print("Indexing successful!")
    else:
        print("Indexing failed.")

    # Retrieve information about the index
    response = es.indices.get(index=index_name)
    response2 = es.indices.get(index=index_name2)

    # Print the retrieved information
    print(response)
    print(response2)
```

VIsualization



2023-06-11T11:27:49Z Runs 25 Run manual__2023-06-11T11:27:48.537424+00:00 Layout Left > Right

DASHBOARD

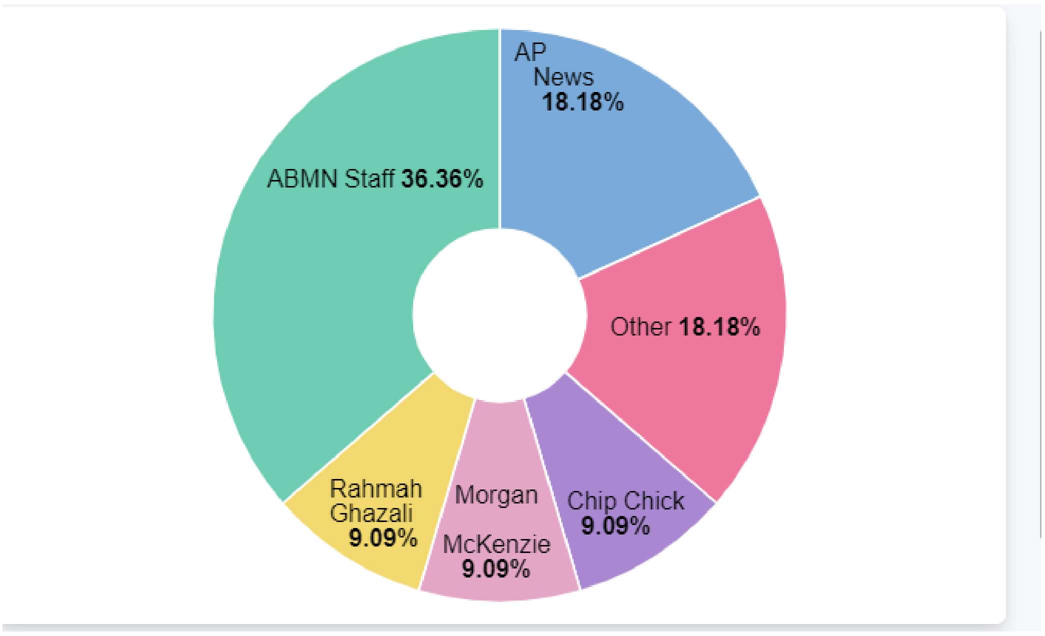


News

Top 10 values of aut	Top 3 values of desc	Top 3 values of title.	Count of records
ABMN Staff	Bruce R. Berkowitz Sell	Bruce R. Berkowitz Sell	6
ABMN Staff	Clean Energy Fuels Co	Clean Energy Fuels Co	6
ABMN Staff	MML Investors Service	MML Investors Service	6
AP News	Authorities in Congo sa	At least 22 people, incl	6
AP News	Authorities say three K	3 Kansas City officers	6
Will Millar	BBC is giving viewers t	Red Nose Day: BBC re	12
Morgan McKenzie	Rodriguez has a hearin	Greeley arson homicid	6
Rahmah Ghazali	The winners of British I	British Kebab Awards 2	6
Sekundė	Jūsų miegamasis &#82	6 profesionalų patarim	6

Currency

Top 3 values of bas	rates	Top 3 values of __in	Median of date
EUR	0	AED	1,686,441,600,000
EUR	0	AFN	1,686,441,600,000
EUR	0	ANG	1,686,441,600,000
EUR	100	ALL	1,686,441,600,000
EUR	100	BDT	1,686,441,600,000
EUR	100	CVE	1,686,441,600,000
EUR	200	ARS	1,686,441,600,000
EUR	200	GYD	1,686,441,600,000
EUR	200	YER	1,686,441,600,000



Conclusion

In conclusion, this project demonstrates a data pipeline for collecting, processing, analyzing, and indexing data from two different sources: the Exchangerates API and the Mediastack API. The pipeline is designed to run periodically, fetching the latest data from the APIs and performing various data processing tasks.

In conclusion, this project demonstrates a data pipeline for collecting, processing, analyzing, and indexing data from two different sources: the Exchangerates API and the Mediastack API. The pipeline is designed to run periodically, fetching the latest data from the APIs and performing various data processing tasks.