7 JANUARY 2024

PROJECT REPORT

INTRODUCTION TO BIGDATA

PROJECT:

BATCH ANALYSIS ON YOUTUBE DATA

MEMBERS:
NOMIR NASIR 11330
AFNAN ANSARI 11378

SUMMARY:

The YouTube Videos Analysis project aimed to conduct a comprehensive analysis of YouTube video data across different regions using PySpark and Hive. The project involved Docker container deployment, data importation, preprocessing, and visualization. The key objectives were to understand trends in views over time, analyze likes vs. dislikes for selected videos, examine overall engagement for the entire region, and present detailed video information.

PROJECT STEPS

1. DOCKER CONTAINER SETUP:

The project began by deploying Docker containers, including the installation of Docker, downloading the Mini Lake folder, and constructing a Docker image. Essential services such as History Server, Data Node, Master Node, Jupyter, and Spark Node were started and monitored. Jupyter was launched to establish a Spark session and context.

2. DATA IMPORTATION AND PREPROCESSING:

YouTube video statistics in CSV format were imported into Jupyter. A directory was created in HDFS, and CSV files were systematically added. Predefined schemas were set up to accommodate the structure of the YouTube video data, and individual tables were created in Hive for each CSV file. Additional preprocessing steps included handling missing values, extracting date components, and introducing a "group_column" to categorize videos.

3. DATA TRANSFORMATION:

The processing code iterated through each table, using Spark SQL to read the original data and implementing a series of transformations. Missing values in key columns were addressed, and new columns were created for date components. The introduction of the "group_column" categorized videos based on specific criteria, such as views, likes, dislikes, and comment_count.

4. SAVING PREPROCESSED DATA:

The preprocessed data for each table was saved as a new table, prefixed by "preprocessed_table_," and appended to the pre_tables list. The code concluded by displaying the list of tables in the Spark session.

5.VISUALIZATION AND REPORTING PHASE:

A. VIEWS OVER TIME (BAR CHART)

- **Visualization:** Bar chart (views-bar-chart) showing the number of views over time.

- **Data Source:** Time series data from 'trending_date' and 'views' columns of the selected YouTube table.

- **Purpose:** Provides an overview of how the number of views changes over time for the selected country.

B. LIKES VS DISLIKES FOR SELECTED VIDEO (PIE CHART)

- **Visualization:** Pie chart (likes-dislikes-pie-chart) illustrating the proportion of likes and dislikes for a selected video.

- **Data Source:** Information about the selected video, including 'likes' and 'dislikes' columns.

- **Purpose:** Offers insights into the engagement of a specific video by visualizing the ratio of likes to dislikes.

C. Total Likes vs. Dislikes for the Entire Region (Pie Chart)

- **Visualization:** Pie chart (likes-dislikes-pie-chart-total) showing the total number of likes and dislikes for all videos in the selected region.

- **Data Source:** Aggregated data from 'likes' and 'dislikes' columns of all videos in the selected YouTube table.

- **Purpose:** Presents an overall summary of likes and dislikes across all videos in the chosen region.

D. Video Information Table

- **Visualization:** Dash Data Table (video-table) displaying key information about each video, including video ID, title, and channel title.

- **Data Source:** DataFrame containing relevant video details.

- **Purpose:** Allows users to explore and filter video information in a tabular format.

E. Selected Video Information (Table and Pie Chart)

- **Visualization:** Additional information (selected-video-info) about the video selected in the Data Table, including a table with various video details and a pie chart illustrating the likes vs. dislikes for the selected video.

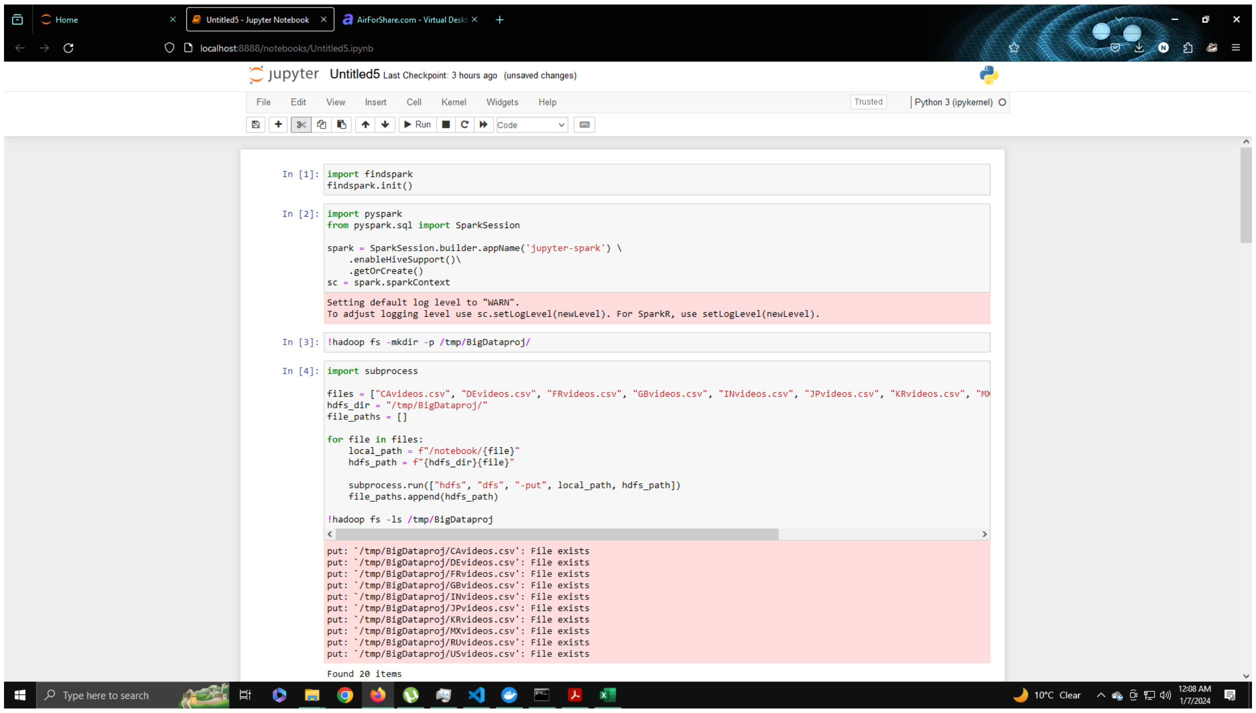
- **Data Source:** Information about the selected video from the DataFrame.

- **Purpose:** Provides a detailed overview of the selected video, enhancing the user's understanding of its characteristics.

CONCLUSION

The YouTube Videos Analysis project successfully leveraged PySpark and Hive to analyze and visualize YouTube video data. The detailed steps in Docker container setup, data importation, preprocessing, and visualization were outlined. The project's visualizations provide valuable insights into views, engagement metrics, and video details, empowering users to make informed decisions based on the presented data.

CODE SNAPSHOTS



```
In [1]: import findspark
        findspark.init()
```

```
In [2]: import pyspark
        from pyspark.sql import SparkSession

        spark = SparkSession.builder.appName('jupyter-spark') \
            .enableHiveSupport()\
            .getOrCreate()
        sc = spark.sparkContext
```

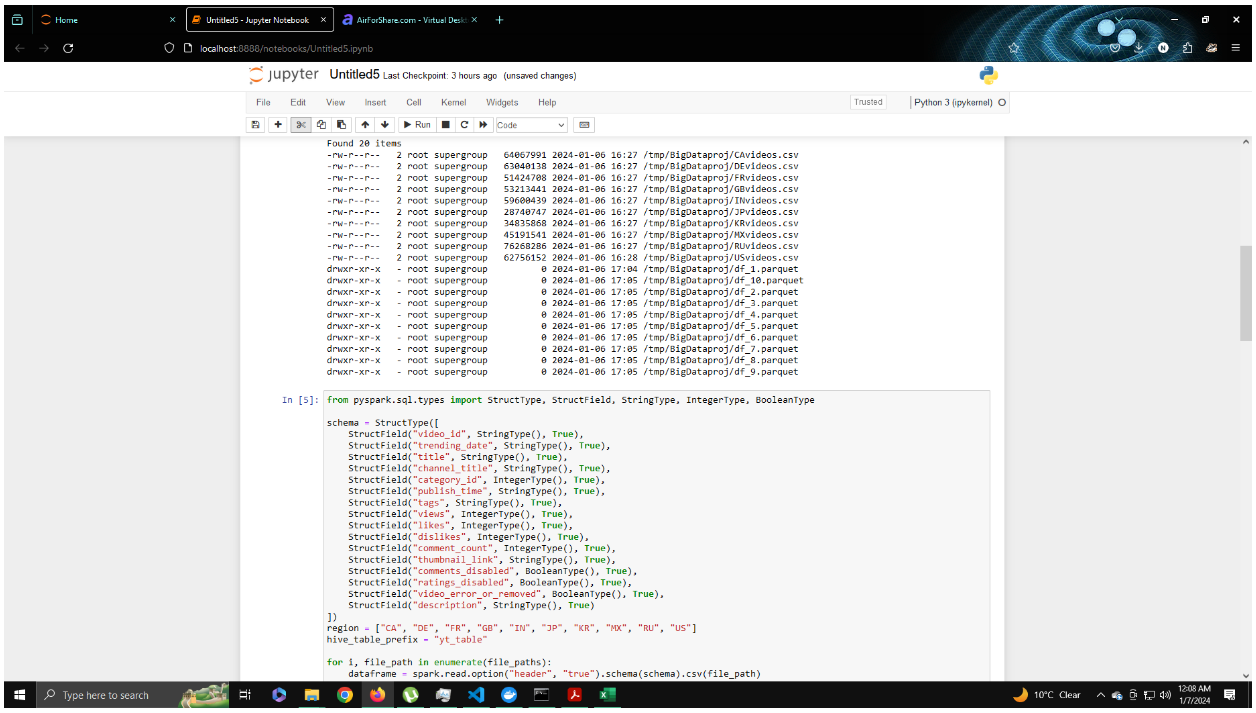
```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

```
In [3]: !hadoop fs -mkdir -p /tmp/BigDataproj/
```

```
In [4]: import subprocess

        files = ["CAvideos.csv", "DEvideos.csv", "FRvideos.csv", "GBvideos.csv", "INvideos.csv", "JPvideos.csv", "KRvideos.csv", "MX
        hdfs_dir = "/tmp/BigDataproj/"
        file_paths = []

        for file in files:
            local_path = f"/notebook/{file}"
            hdfs_path = f"{hdfs_dir}{file}"

            subprocess.run(["hdfs", "dfs", "-put", local_path, hdfs_path])
            file_paths.append(hdfs_path)

        !hadoop fs -ls /tmp/BigDataproj
```

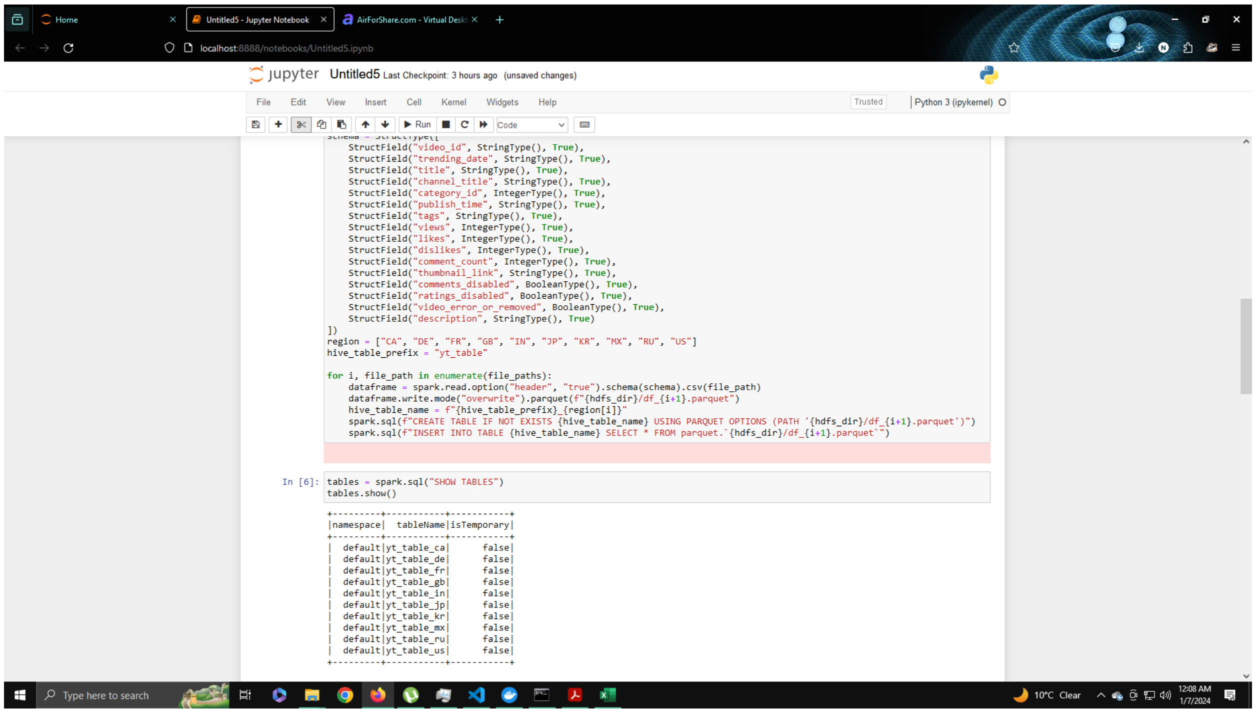
```
put: `/tmp/BigDataproj/CAvideos.csv': File exists
put: `/tmp/BigDataproj/DEvideos.csv': File exists
put: `/tmp/BigDataproj/FRvideos.csv': File exists
put: `/tmp/BigDataproj/GBvideos.csv': File exists
put: `/tmp/BigDataproj/INvideos.csv': File exists
put: `/tmp/BigDataproj/JPvideos.csv': File exists
put: `/tmp/BigDataproj/KRvideos.csv': File exists
put: `/tmp/BigDataproj/MXvideos.csv': File exists
put: `/tmp/BigDataproj/RUvideos.csv': File exists
put: `/tmp/BigDataproj/USvideos.csv': File exists
```

```
Found 20 items
```



```
Found 20 items
-rw-r--r--   2 root supergroup   64067991 2024-01-06 16:27 /tmp/BigDataproj/CAvideos.csv
-rw-r--r--   2 root supergroup   63040138 2024-01-06 16:27 /tmp/BigDataproj/DEvideos.csv
-rw-r--r--   2 root supergroup   51424708 2024-01-06 16:27 /tmp/BigDataproj/FRvideos.csv
-rw-r--r--   2 root supergroup   53213441 2024-01-06 16:27 /tmp/BigDataproj/GBvideos.csv
-rw-r--r--   2 root supergroup   59600439 2024-01-06 16:27 /tmp/BigDataproj/INvideos.csv
-rw-r--r--   2 root supergroup   28740747 2024-01-06 16:27 /tmp/BigDataproj/JPvideos.csv
-rw-r--r--   2 root supergroup   34835868 2024-01-06 16:27 /tmp/BigDataproj/KRvideos.csv
-rw-r--r--   2 root supergroup   45191541 2024-01-06 16:27 /tmp/BigDataproj/MXvideos.csv
-rw-r--r--   2 root supergroup   76268286 2024-01-06 16:27 /tmp/BigDataproj/RUvideos.csv
-rw-r--r--   2 root supergroup   62756152 2024-01-06 16:28 /tmp/BigDataproj/USvideos.csv
drwxr-xr-x   - root supergroup          0 2024-01-06 17:04 /tmp/BigDataproj/df_1.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_10.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_2.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_3.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_4.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_5.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_6.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_7.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_8.parquet
drwxr-xr-x   - root supergroup          0 2024-01-06 17:05 /tmp/BigDataproj/df_9.parquet
```

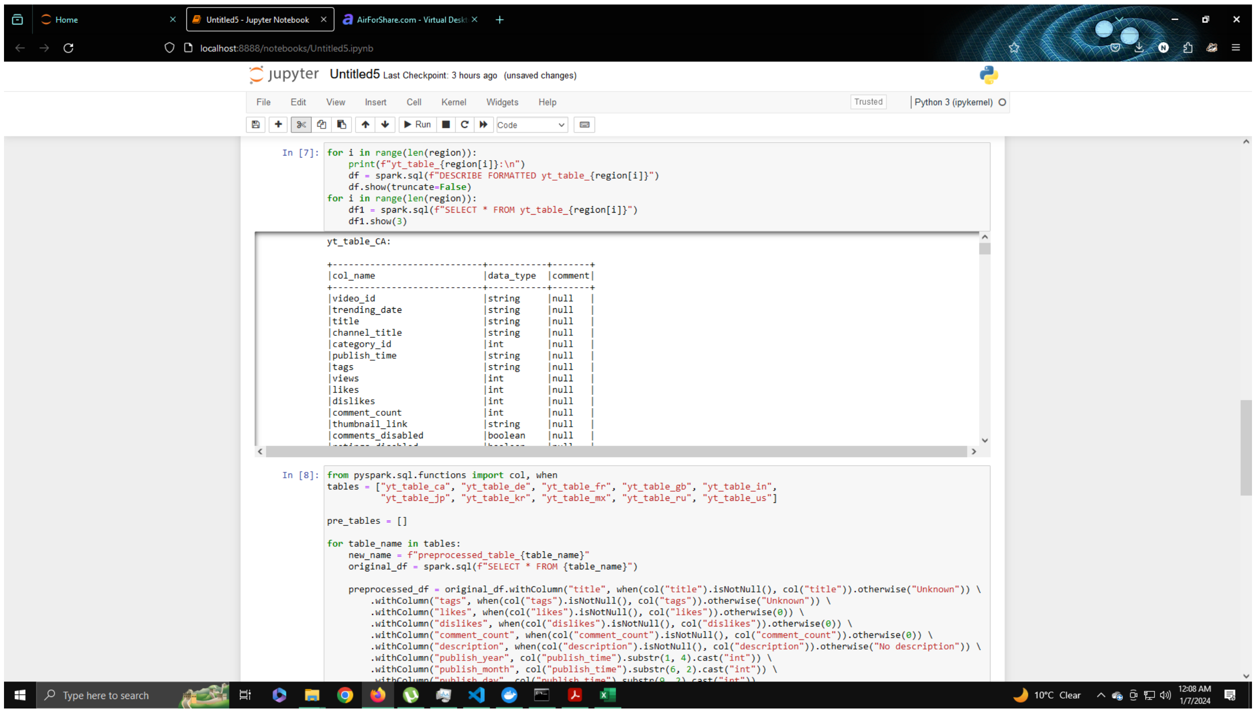
```
In [5]: from pyspark.sql.types import StructType, StructField, StringType, IntegerType, BooleanType

        schema = StructType([
            StructField("video_id", StringType(), True),
            StructField("trending_date", StringType(), True),
            StructField("title", StringType(), True),
            StructField("channel_title", StringType(), True),
            StructField("category_id", IntegerType(), True),
            StructField("publish_time", StringType(), True),
            StructField("tags", StringType(), True),
            StructField("views", IntegerType(), True),
            StructField("likes", IntegerType(), True),
            StructField("dislikes", IntegerType(), True),
            StructField("comment_count", IntegerType(), True),
            StructField("thumbnail_link", StringType(), True),
            StructField("comments_disabled", BooleanType(), True),
            StructField("ratings_disabled", BooleanType(), True),
            StructField("video_error_or_removed", BooleanType(), True),
            StructField("description", StringType(), True)
        ])
        region = ["CA", "DE", "FR", "GB", "IN", "JP", "KR", "MX", "RU", "US"]
        hive_table_prefix = "yt_table"

        for i, file_path in enumerate(file_paths):
            dataframe = spark.read.option("header", "true").schema(schema).csv(file_path)
```



```
    StructField("video_id", StringType(), True),
    StructField("trending_date", StringType(), True),
    StructField("title", StringType(), True),
    StructField("channel_title", StringType(), True),
    StructField("category_id", IntegerType(), True),
    StructField("publish_time", StringType(), True),
    StructField("tags", StringType(), True),
    StructField("views", IntegerType(), True),
    StructField("likes", IntegerType(), True),
    StructField("dislikes", IntegerType(), True),
    StructField("comment_count", IntegerType(), True),
    StructField("thumbnail_link", StringType(), True),
    StructField("comments_disabled", BooleanType(), True),
    StructField("ratings_disabled", BooleanType(), True),
    StructField("video_error_or_removed", BooleanType(), True),
    StructField("description", StringType(), True)
])
region = ["CA", "DE", "FR", "GB", "IN", "JP", "KR", "MX", "RU", "US"]
hive_table_prefix = "yt_table"

for i, file_path in enumerate(file_paths):
    dataframe = spark.read.option("header", "true").schema(schema).csv(file_path)
    dataframe.write.mode("overwrite").parquet(f"{hdfs_dir}/df_{i+1}.parquet")
    hive_table_name = f"{hive_table_prefix}_{region[i]}"
    spark.sql(f"CREATE TABLE IF NOT EXISTS {hive_table_name} USING PARQUET OPTIONS (PATH '{hdfs_dir}/df_{i+1}.parquet')")
    spark.sql(f"INSERT INTO TABLE {hive_table_name} SELECT * FROM parquet.`{hdfs_dir}/df_{i+1}.parquet`")
```

In [6]:
```
tables = spark.sql("SHOW TABLES")
tables.show()
```

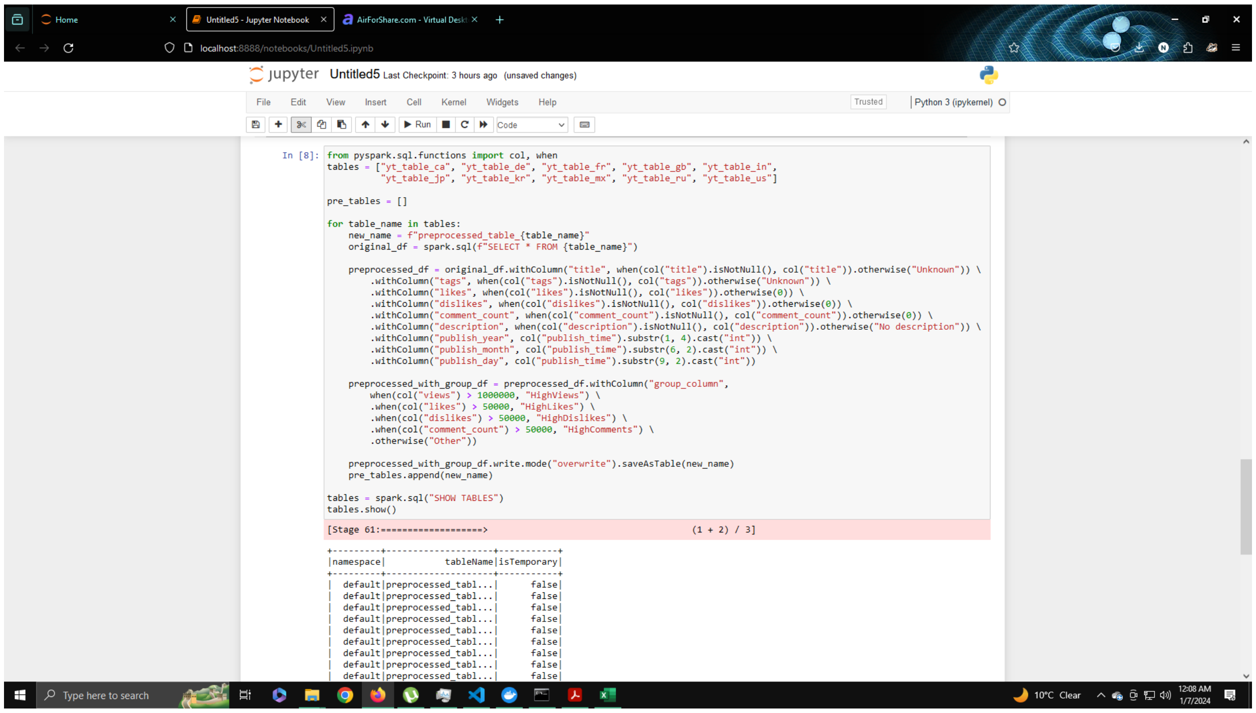
```
+---------+-----------+-----------+
|namespace|  tableName|isTemporary|
+---------+-----------+-----------+
|  default|yt_table_ca|      false|
|  default|yt_table_de|      false|
|  default|yt_table_fr|      false|
|  default|yt_table_gb|      false|
|  default|yt_table_in|      false|
|  default|yt_table_jp|      false|
|  default|yt_table_kr|      false|
|  default|yt_table_mx|      false|
|  default|yt_table_ru|      false|
|  default|yt_table_us|      false|
+---------+-----------+-----------+
```



In [7]:
```
for i in range(len(region)):
    print(f"yt_table_{region[i]}:\n")
    df = spark.sql(f"DESCRIBE FORMATTED yt_table_{region[i]}")
    df.show(truncate=False)
for i in range(len(region)):
    df1 = spark.sql(f"SELECT * FROM yt_table_{region[i]}")
    df1.show(3)
```

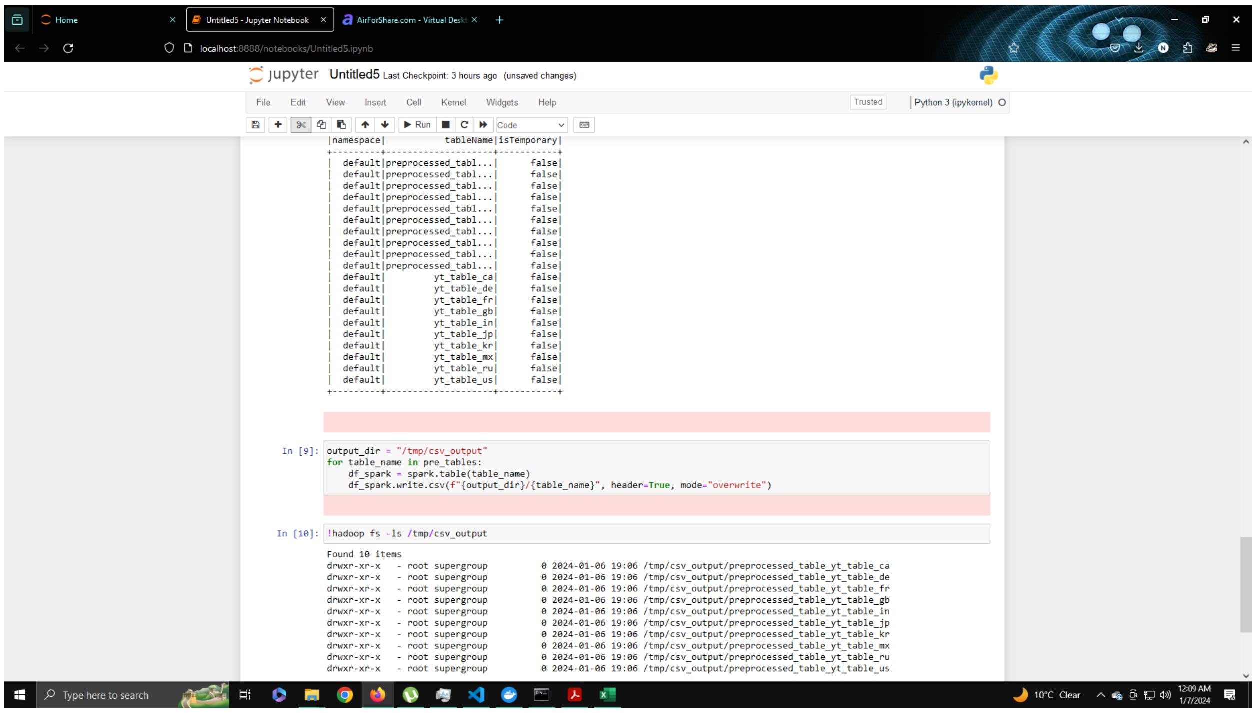
```
yt_table_CA:

+----------------------------+----------+-------+
|col_name                    |data_type |comment|
+----------------------------+----------+-------+
|video_id                    |string    |null   |
|trending_date               |string    |null   |
|title                       |string    |null   |
|channel_title               |string    |null   |
|category_id                 |int       |null   |
|publish_time                |string    |null   |
|tags                        |string    |null   |
|views                       |int       |null   |
|likes                       |int       |null   |
|dislikes                    |int       |null   |
|comment_count               |int       |null   |
|thumbnail_link              |string    |null   |
|comments_disabled           |boolean   |null   |
```

In [8]:
```
from pyspark.sql.functions import col, when
tables = ["yt_table_ca", "yt_table_de", "yt_table_fr", "yt_table_gb", "yt_table_in",
          "yt_table_jp", "yt_table_kr", "yt_table_mx", "yt_table_ru", "yt_table_us"]

pre_tables = []

for table_name in tables:
    new_name = f"preprocessed_table_{table_name}"
    original_df = spark.sql(f"SELECT * FROM {table_name}")

    preprocessed_df = original_df.withColumn("title", when(col("title").isNotNull(), col("title")).otherwise("Unknown")) \
        .withColumn("tags", when(col("tags").isNotNull(), col("tags")).otherwise("Unknown")) \
        .withColumn("likes", when(col("likes").isNotNull(), col("likes")).otherwise(0)) \
        .withColumn("dislikes", when(col("dislikes").isNotNull(), col("dislikes")).otherwise(0)) \
        .withColumn("comment_count", when(col("comment_count").isNotNull(), col("comment_count")).otherwise(0)) \
        .withColumn("description", when(col("description").isNotNull(), col("description")).otherwise("No description")) \
        .withColumn("publish_year", col("publish_time").substr(1, 4).cast("int")) \
        .withColumn("publish_month", col("publish_time").substr(6, 2).cast("int")) \
```


Home
Untitled5 - Jupyter Notebook
AirForShare.com - Virtual Deskt
localhost:8888/notebooks/Untitled5.ipynb
jupyter Untitled5 Last Checkpoint: 3 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Type here to search
10°C Clear
12:08 AM
1/7/2024

In [8]:
```
from pyspark.sql.functions import col, when
tables = ["yt_table_ca", "yt_table_de", "yt_table_fr", "yt_table_gb", "yt_table_in",
          "yt_table_jp", "yt_table_kr", "yt_table_mx", "yt_table_ru", "yt_table_us"]

pre_tables = []

for table_name in tables:
    new_name = f"preprocessed_table_{table_name}"
    original_df = spark.sql(f"SELECT * FROM {table_name}")

    preprocessed_df = original_df.withColumn("title", when(col("title").isNotNull(), col("title")).otherwise("Unknown")) \
        .withColumn("tags", when(col("tags").isNotNull(), col("tags")).otherwise("Unknown")) \
        .withColumn("likes", when(col("likes").isNotNull(), col("likes")).otherwise(0)) \
        .withColumn("dislikes", when(col("dislikes").isNotNull(), col("dislikes")).otherwise(0)) \
        .withColumn("comment_count", when(col("comment_count").isNotNull(), col("comment_count")).otherwise(0)) \
        .withColumn("description", when(col("description").isNotNull(), col("description")).otherwise("No description")) \
        .withColumn("publish_year", col("publish_time").substr(1, 4).cast("int")) \
        .withColumn("publish_month", col("publish_time").substr(6, 2).cast("int")) \
        .withColumn("publish_day", col("publish_time").substr(9, 2).cast("int"))

    preprocessed_with_group_df = preprocessed_df.withColumn("group_column",
        when(col("views") > 1000000, "HighViews") \
        .when(col("likes") > 50000, "HighLikes") \
        .when(col("dislikes") > 50000, "HighDislikes") \
        .when(col("comment_count") > 50000, "HighComments") \
        .otherwise("Other"))

    preprocessed_with_group_df.write.mode("overwrite").saveAsTable(new_name)
    pre_tables.append(new_name)

tables = spark.sql("SHOW TABLES")
tables.show()
```

```
[Stage 61:===================>                                  (1 + 2) / 3]

+---------+--------------------+-----------+
|namespace|           tableName|isTemporary|
+---------+--------------------+-----------+
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
```

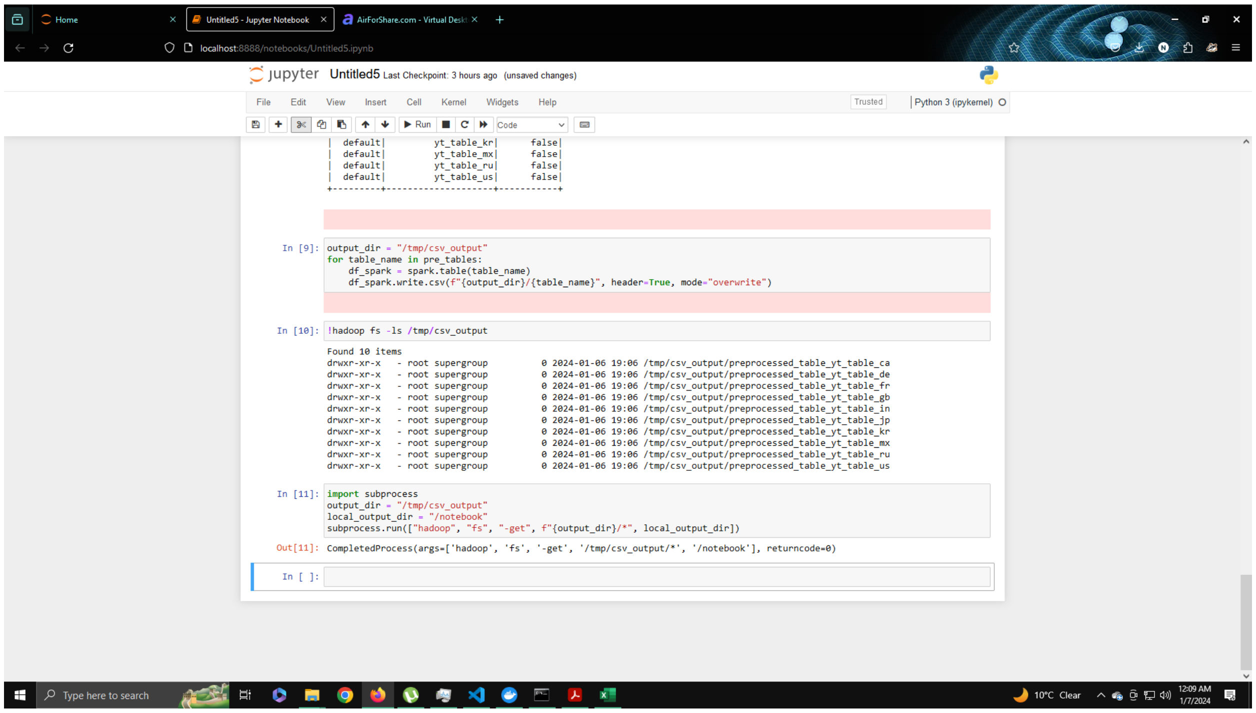

Home
Untitled5 - Jupyter Notebook
AirForShare.com - Virtual Deskt
localhost:8888/notebooks/Untitled5.ipynb
jupyter Untitled5 Last Checkpoint: 3 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Type here to search
10°C Clear
12:09 AM
1/7/2024

```
|namespace|           tableName|isTemporary|
+---------+--------------------+-----------+
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|         yt_table_ca|      false|
|  default|         yt_table_de|      false|
|  default|         yt_table_fr|      false|
|  default|         yt_table_gb|      false|
|  default|         yt_table_in|      false|
|  default|         yt_table_jp|      false|
|  default|         yt_table_kr|      false|
|  default|         yt_table_mx|      false|
|  default|         yt_table_ru|      false|
|  default|         yt_table_us|      false|
+---------+--------------------+-----------+
```

In [9]:
```
output_dir = "/tmp/csv_output"
for table_name in pre_tables:
    df_spark = spark.table(table_name)
    df_spark.write.csv(f"{output_dir}/{table_name}", header=True, mode="overwrite")
```

In [10]:
```
!hadoop fs -ls /tmp/csv_output
```

```
Found 10 items
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_ca
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_de
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_fr
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_gb
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_in
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_jp
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_kr
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_mx
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_ru
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_us
```


localhost:8888/notebooks/Untitled5.ipynb

Untitled5 Last Checkpoint: 3 hours ago (unsaved changes)

```
|  default|         yt_table_kr|      false|
|  default|         yt_table_mx|      false|
|  default|         yt_table_ru|      false|
|  default|         yt_table_us|      false|
+---------+-------------------+-----------+
```

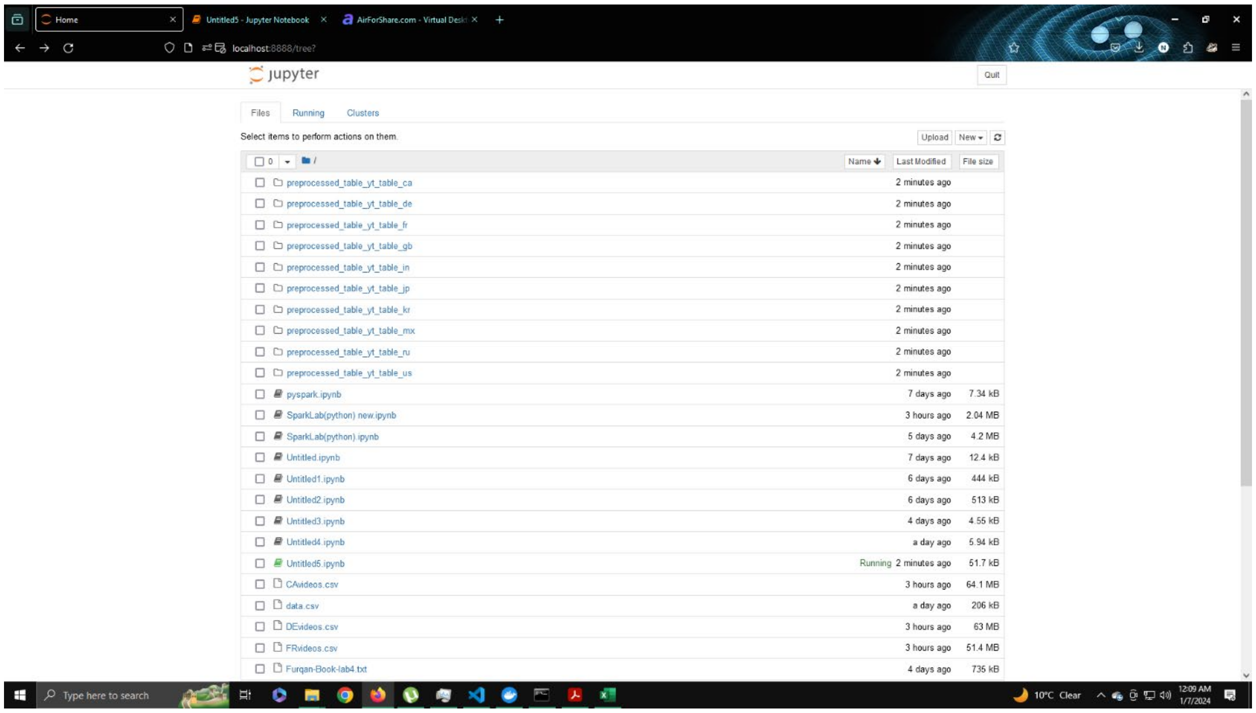
In [9]:
```
output_dir = "/tmp/csv_output"
for table_name in pre_tables:
    df_spark = spark.table(table_name)
    df_spark.write.csv(f"{output_dir}/{table_name}", header=True, mode="overwrite")
```

In [10]:
```
!hadoop fs -ls /tmp/csv_output
```

```
Found 10 items
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_ca
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_de
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_fr
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_gb
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_in
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_jp
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_kr
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_mx
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_ru
drwxr-xr-x   - root supergroup          0 2024-01-06 19:06 /tmp/csv_output/preprocessed_table_yt_table_us
```

In [11]:
```
import subprocess
output_dir = "/tmp/csv_output"
local_output_dir = "/notebook"
subprocess.run(["hadoop", "fs", "-get", f"{output_dir}/*", local_output_dir])
```

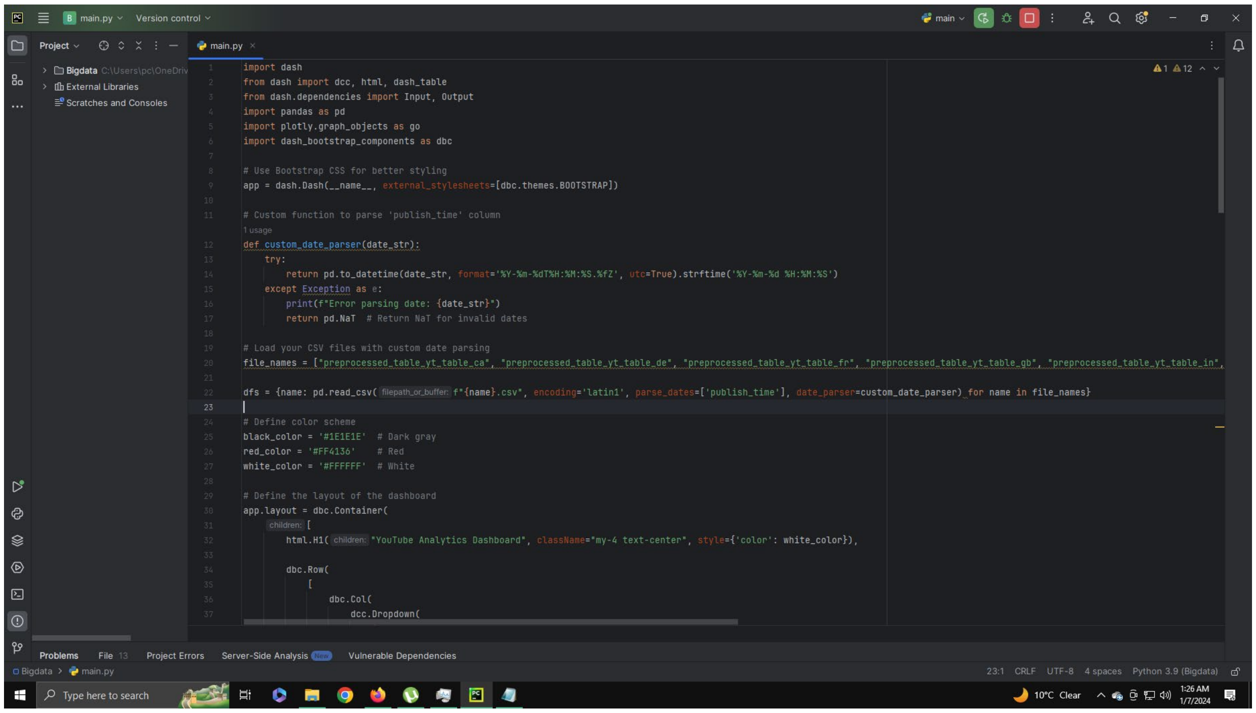
Out[11]:
```
CompletedProcess(args=['hadoop', 'fs', '-get', '/tmp/csv_output/*', '/notebook'], returncode=0)
```


localhost:8888/tree?

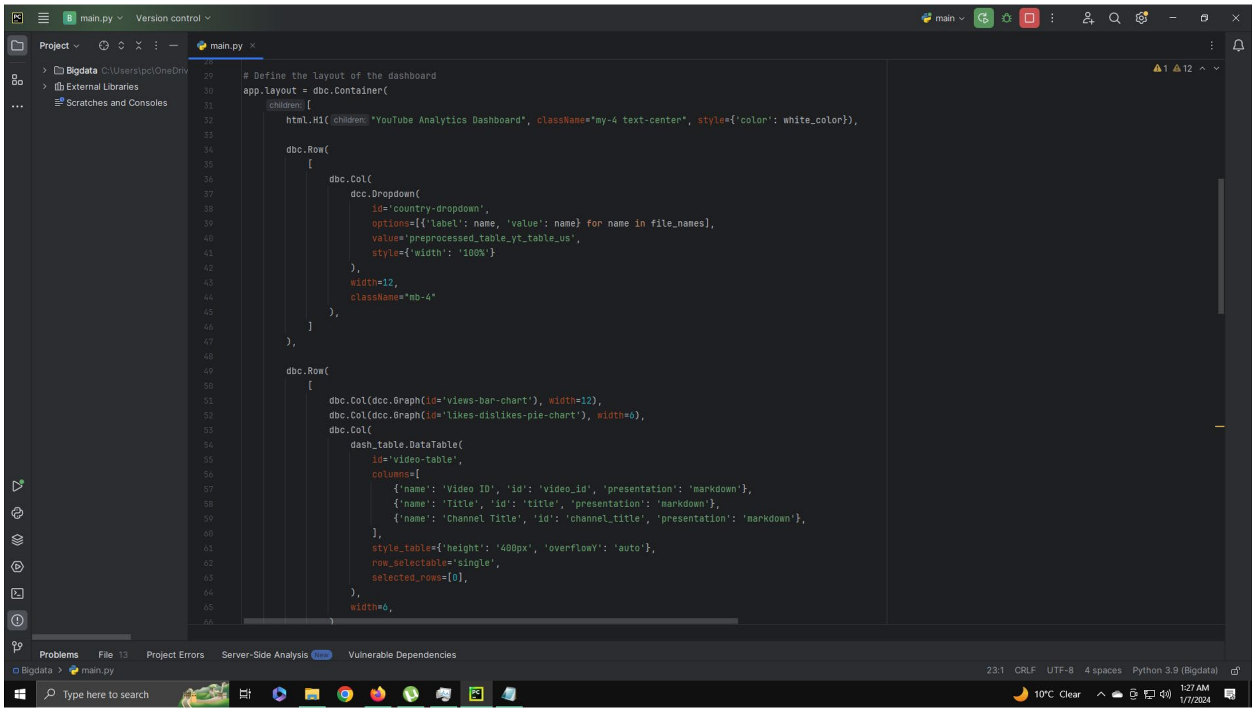
Select items to perform actions on them.

Name	Last Modified	File size
preprocessed_table_yt_table_ca	2 minutes ago	
preprocessed_table_yt_table_de	2 minutes ago	
preprocessed_table_yt_table_fr	2 minutes ago	
preprocessed_table_yt_table_gb	2 minutes ago	
preprocessed_table_yt_table_in	2 minutes ago	
preprocessed_table_yt_table_jp	2 minutes ago	
preprocessed_table_yt_table_kr	2 minutes ago	
preprocessed_table_yt_table_mx	2 minutes ago	
preprocessed_table_yt_table_ru	2 minutes ago	
preprocessed_table_yt_table_us	2 minutes ago	
pyspark.ipynb	7 days ago	7.34 kB
SparkLab(python) new.ipynb	3 hours ago	2.04 MB
SparkLab(python).ipynb	5 days ago	4.2 MB
Untitled.ipynb	7 days ago	12.4 kB
Untitled1.ipynb	6 days ago	444 kB
Untitled2.ipynb	6 days ago	513 kB
Untitled3.ipynb	4 days ago	4.55 kB
Untitled4.ipynb	a day ago	5.94 kB
Untitled5.ipynb	Running 2 minutes ago	51.7 kB
CAvideos.csv	3 hours ago	64.1 MB
data.csv	a day ago	206 kB
DEvideos.csv	3 hours ago	63 MB
FRvideos.csv	3 hours ago	51.4 MB
Furqan-Book-lab4.txt	4 days ago	735 kB

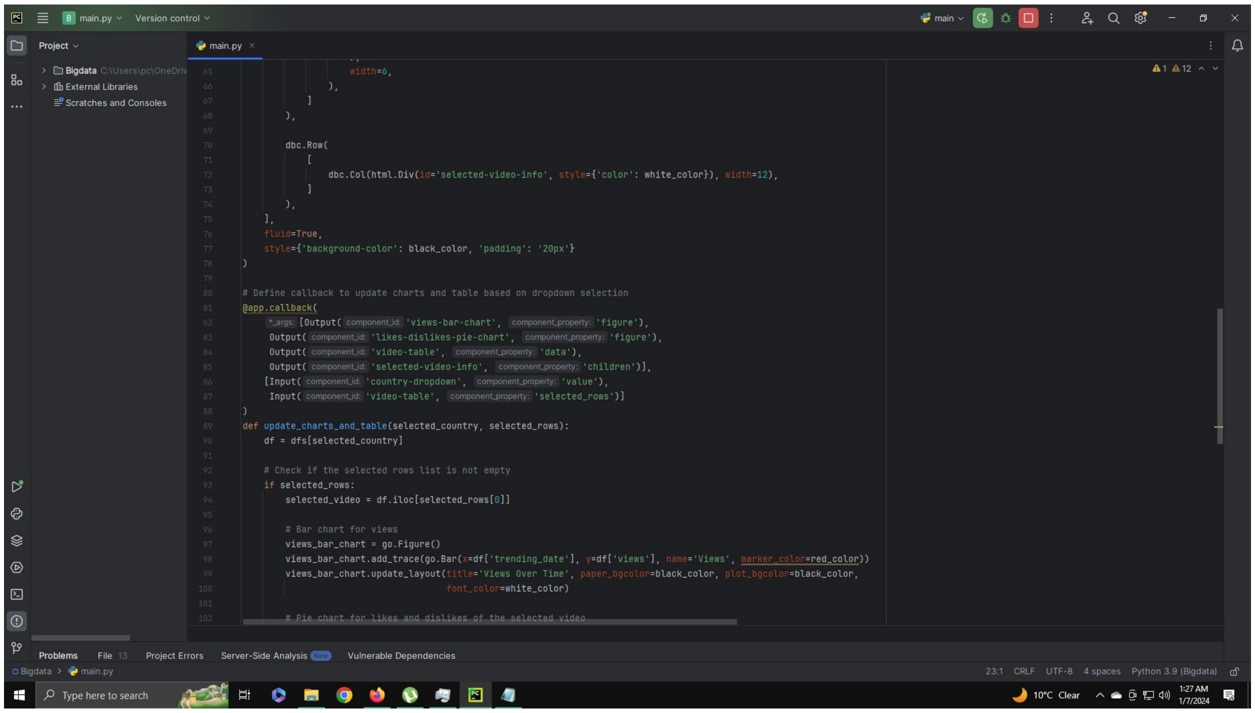
DASHBOARD CODE SNAPSHOT



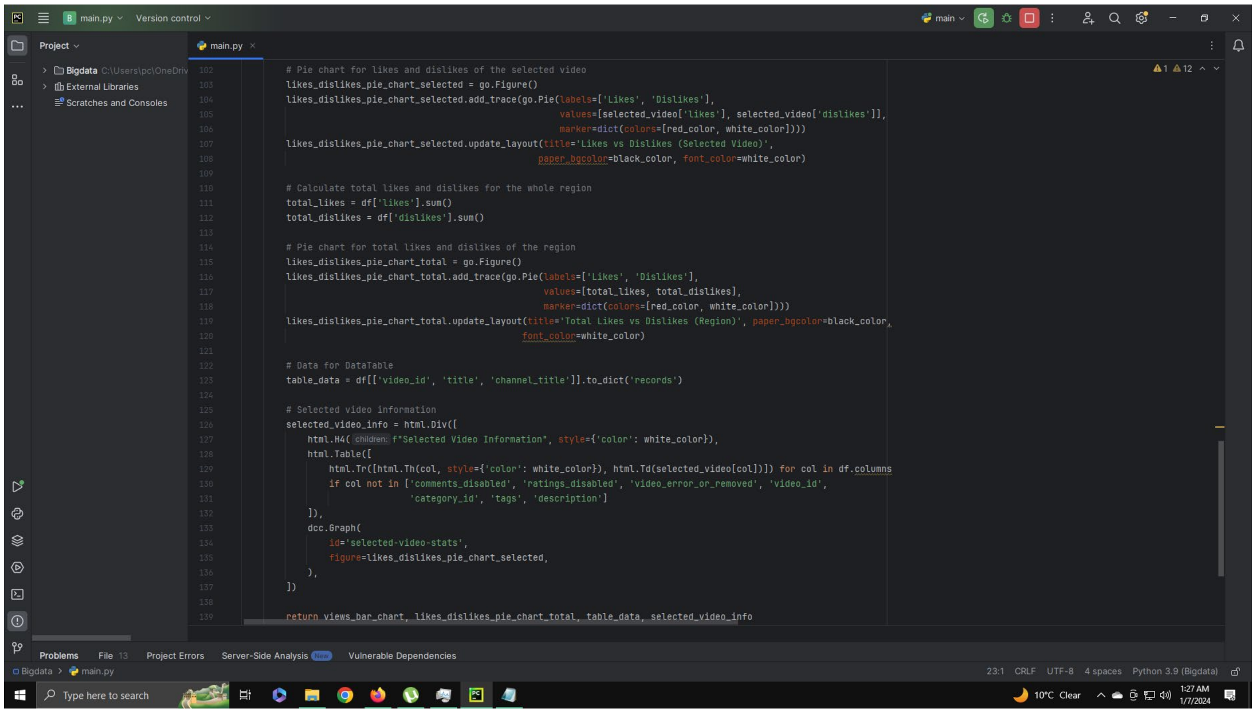
```python
import dash
from dash import dcc, html, dash_table
from dash.dependencies import Input, Output
import pandas as pd
import plotly.graph_objects as go
import dash_bootstrap_components as dbc

# Use Bootstrap CSS for better styling
app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])

# Custom function to parse 'publish_time' column
1 usage
def custom_date_parser(date_str):
    try:
        return pd.to_datetime(date_str, format='%Y-%m-%dT%H:%M:%S.%fZ', utc=True).strftime('%Y-%m-%d %H:%M:%S')
    except Exception as e:
        print(f"Error parsing date: {date_str}")
        return pd.NaT  # Return NaT for invalid dates

# Load your CSV files with custom date parsing
file_names = ["preprocessed_table_yt_table_ca", "preprocessed_table_yt_table_de", "preprocessed_table_yt_table_fr", "preprocessed_table_yt_table_gb", "preprocessed_table_yt_table_in",

dfs = {name: pd.read_csv( filepath_or_buffer: f"{name}.csv", encoding='latin1', parse_dates=['publish_time'], date_parser=custom_date_parser) for name in file_names}

# Define color scheme
black_color = '#1E1E1E'  # Dark gray
red_color = '#FF4136'    # Red
white_color = '#FFFFFF'  # White

# Define the layout of the dashboard
app.layout = dbc.Container(
    children: [
        html.H1( children: "YouTube Analytics Dashboard", className="my-4 text-center", style={'color': white_color}),

        dbc.Row(
            [
                dbc.Col(
                    dcc.Dropdown(
```



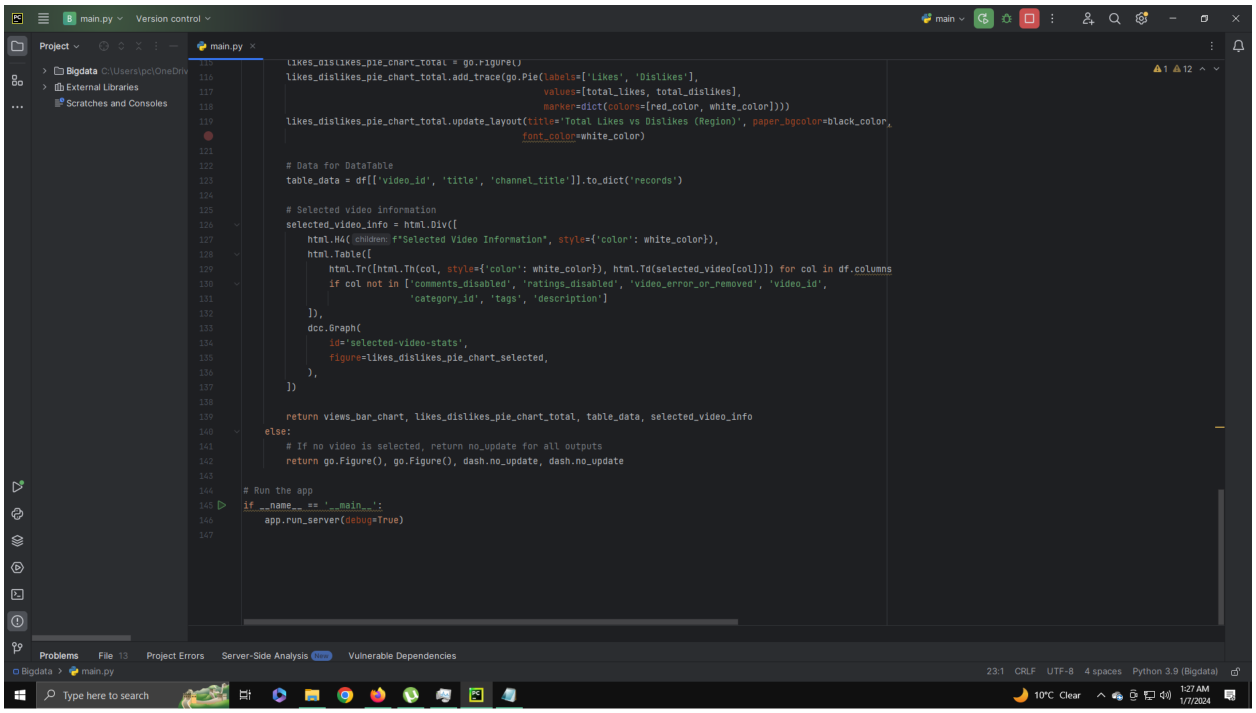
```python
# Define the layout of the dashboard
app.layout = dbc.Container(
    children: [
        html.H1( children: "YouTube Analytics Dashboard", className="my-4 text-center", style={'color': white_color}),

        dbc.Row(
            [
                dbc.Col(
                    dcc.Dropdown(
                        id='country-dropdown',
                        options=[{'label': name, 'value': name} for name in file_names],
                        value='preprocessed_table_yt_table_us',
                        style={'width': '100%'}
                    ),
                    width=12,
                    className="mb-4"
                ),
            ]
        ),

        dbc.Row(
            [
                dbc.Col(dcc.Graph(id='views-bar-chart'), width=12),
                dbc.Col(dcc.Graph(id='likes-dislikes-pie-chart'), width=6),
                dbc.Col(
                    dash_table.DataTable(
                        id='video-table',
                        columns=[
                            {'name': 'Video ID', 'id': 'video_id', 'presentation': 'markdown'},
                            {'name': 'Title', 'id': 'title', 'presentation': 'markdown'},
                            {'name': 'Channel Title', 'id': 'channel_title', 'presentation': 'markdown'},
                        ],
                        style_table={'height': '400px', 'overflowY': 'auto'},
                        row_selectable='single',
                        selected_rows=[0],
                    ),
                    width=6,
                )
```



```python
                    width=6,
                ),
            ]
        ),

        dbc.Row(
            [
                dbc.Col(html.Div(id='selected-video-info', style={'color': white_color}), width=12),
            ]
        ),
    ],
    fluid=True,
    style={'background-color': black_color, 'padding': '20px'}
)

# Define callback to update charts and table based on dropdown selection
@app.callback(
    *_args: [Output( component_id: 'views-bar-chart',  component_property: 'figure'),
     Output( component_id: 'likes-dislikes-pie-chart',  component_property: 'figure'),
     Output( component_id: 'video-table',  component_property: 'data'),
     Output( component_id: 'selected-video-info',  component_property: 'children')],
    [Input( component_id: 'country-dropdown',  component_property: 'value'),
     Input( component_id: 'video-table',  component_property: 'selected_rows')]
)
def update_charts_and_table(selected_country, selected_rows):
    df = dfs[selected_country]

    # Check if the selected rows list is not empty
    if selected_rows:
        selected_video = df.iloc[selected_rows[0]]

        # Bar chart for views
        views_bar_chart = go.Figure()
        views_bar_chart.add_trace(go.Bar(x=df['trending_date'], y=df['views'], name='Views', marker_color=red_color))
        views_bar_chart.update_layout(title='Views Over Time', paper_bgcolor=black_color, plot_bgcolor=black_color,
                                      font_color=white_color)

        # Pie chart for likes and dislikes of the selected video
```

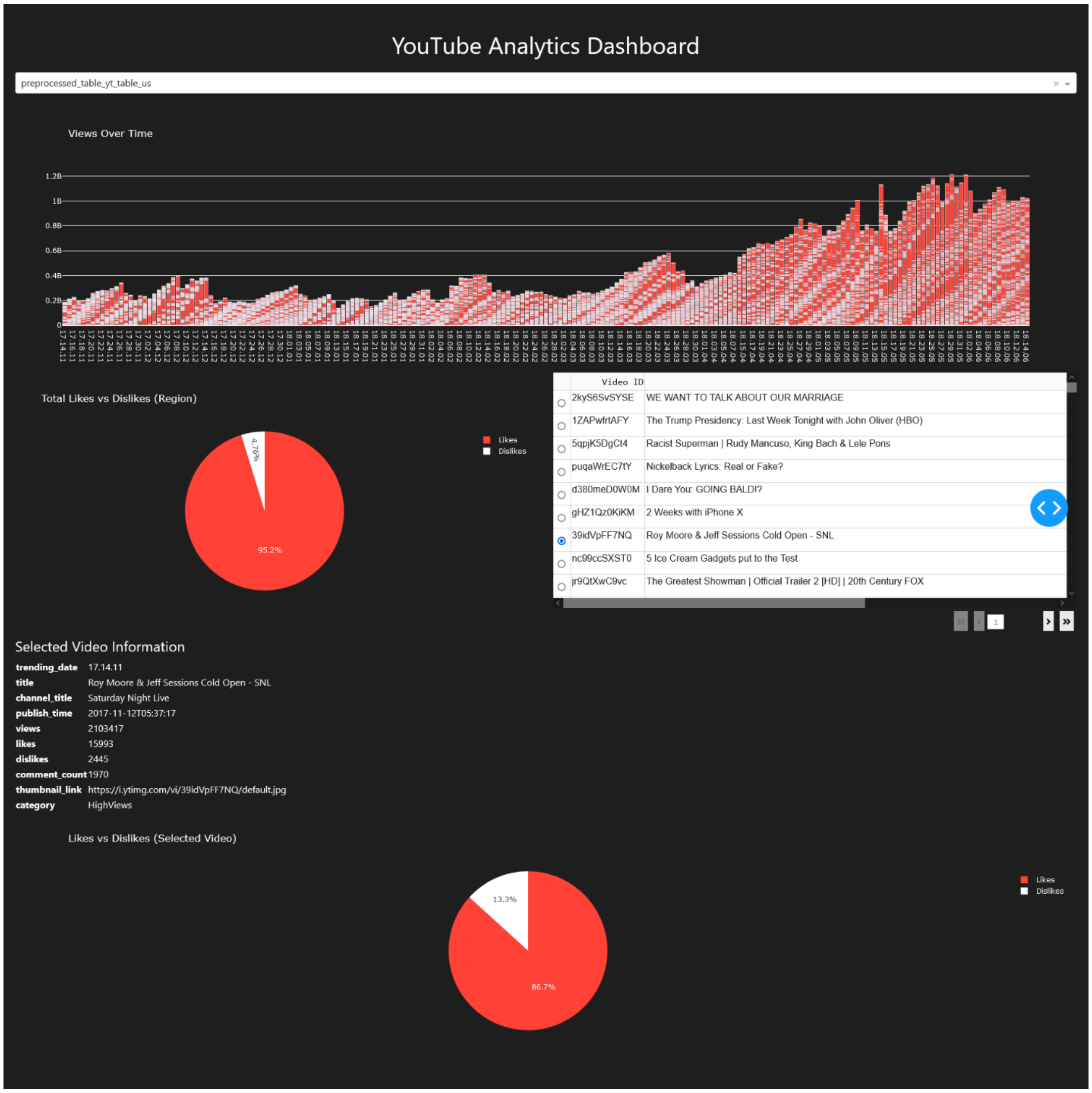


```python
        # Pie chart for likes and dislikes of the selected video
        likes_dislikes_pie_chart_selected = go.Figure()
        likes_dislikes_pie_chart_selected.add_trace(go.Pie(labels=['Likes', 'Dislikes'],
                                                           values=[selected_video['likes'], selected_video['dislikes']],
                                                           marker=dict(colors=[red_color, white_color])))
        likes_dislikes_pie_chart_selected.update_layout(title='Likes vs Dislikes (Selected Video)',
                                                        paper_bgcolor=black_color, font_color=white_color)

        # Calculate total likes and dislikes for the whole region
        total_likes = df['likes'].sum()
        total_dislikes = df['dislikes'].sum()

        # Pie chart for total likes and dislikes of the region
        likes_dislikes_pie_chart_total = go.Figure()
        likes_dislikes_pie_chart_total.add_trace(go.Pie(labels=['Likes', 'Dislikes'],
                                                        values=[total_likes, total_dislikes],
                                                        marker=dict(colors=[red_color, white_color])))
        likes_dislikes_pie_chart_total.update_layout(title='Total Likes vs Dislikes (Region)', paper_bgcolor=black_color,
                                                     font_color=white_color)

        # Data for DataTable
        table_data = df[['video_id', 'title', 'channel_title']].to_dict('records')

        # Selected video information
        selected_video_info = html.Div([
            html.H4( children: f"Selected Video Information", style={'color': white_color}),
            html.Table([
                html.Tr([html.Th(col, style={'color': white_color}), html.Td(selected_video[col])]) for col in df.columns
                if col not in ['comments_disabled', 'ratings_disabled', 'video_error_or_removed', 'video_id',
                               'category_id', 'tags', 'description']
            ]),
            dcc.Graph(
                id='selected-video-stats',
                figure=likes_dislikes_pie_chart_selected,
            ),
        ])

        return views_bar_chart, likes_dislikes_pie_chart_total, table_data, selected_video_info
```



```
        likes_dislikes_pie_chart_total = go.Figure()
        likes_dislikes_pie_chart_total.add_trace(go.Pie(labels=['Likes', 'Dislikes'],
                                                        values=[total_likes, total_dislikes],
                                                        marker=dict(colors=[red_color, white_color])))
        likes_dislikes_pie_chart_total.update_layout(title='Total Likes vs Dislikes (Region)', paper_bgcolor=black_color,
                                                    font_color=white_color)

        # Data for DataTable
        table_data = df[['video_id', 'title', 'channel_title']].to_dict('records')

        # Selected video information
        selected_video_info = html.Div([
            html.H4( children: f"Selected Video Information", style={'color': white_color}),
            html.Table([
                html.Tr([html.Th(col, style={'color': white_color}), html.Td(selected_video[col])]) for col in df.columns
                if col not in ['comments_disabled', 'ratings_disabled', 'video_error_or_removed', 'video_id',
                               'category_id', 'tags', 'description']
            ]),
            dcc.Graph(
                id='selected-video-stats',
                figure=likes_dislikes_pie_chart_selected,
            ),
        ])

        return views_bar_chart, likes_dislikes_pie_chart_total, table_data, selected_video_info
    else:
        # If no video is selected, return no_update for all outputs
        return go.Figure(), go.Figure(), dash.no_update, dash.no_update

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

DASHBOARD OUTPUT

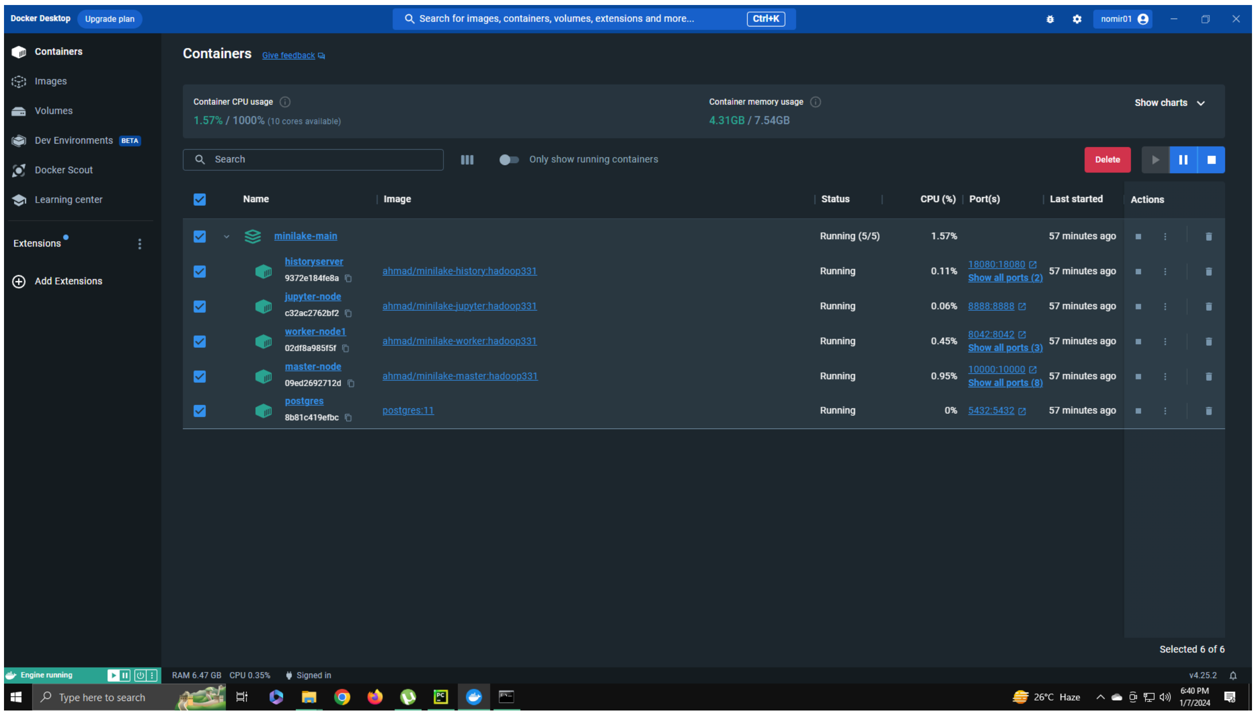

YouTube Analytics Dashboard
preprocessed_table_yt_table_us
Views Over Time
1.2B
1B
0.8B
0.6B
0.4B
0.2B
0
Total Likes vs Dislikes (Region)
4.76%
95.2%
Likes
Dislikes

	Video ID	
○	2kyS6SvSYSE	WE WANT TO TALK ABOUT OUR MARRIAGE
○	1ZAPwfrtAFY	The Trump Presidency: Last Week Tonight with John Oliver (HBO)
○	5qpjK5DgCt4	Racist Superman | Rudy Mancuso, King Bach & Lele Pons
○	puqaWrEC7tY	Nickelback Lyrics: Real or Fake?
○	d380meD0W0M	I Dare You: GOING BALD!?
○	gHZ1Qz0KiKM	2 Weeks with iPhone X
●	39idVpFF7NQ	Roy Moore & Jeff Sessions Cold Open - SNL
○	nc99ccSXST0	5 Ice Cream Gadgets put to the Test
○	jr9QtXwC9vc	The Greatest Showman | Official Trailer 2 [HD] | 20th Century FOX

Selected Video Information

trending_date	17.14.11
title	Roy Moore & Jeff Sessions Cold Open - SNL
channel_title	Saturday Night Live
publish_time	2017-11-12T05:37:17
views	2103417
likes	15993
dislikes	2445
comment_count	1970
thumbnail_link	https://i.ytimg.com/vi/39idVpFF7NQ/default.jpg
category	HighViews

DOCKER SNAPSHOT



Containers

Container CPU usage: 1.57% / 1000% (10 cores available)

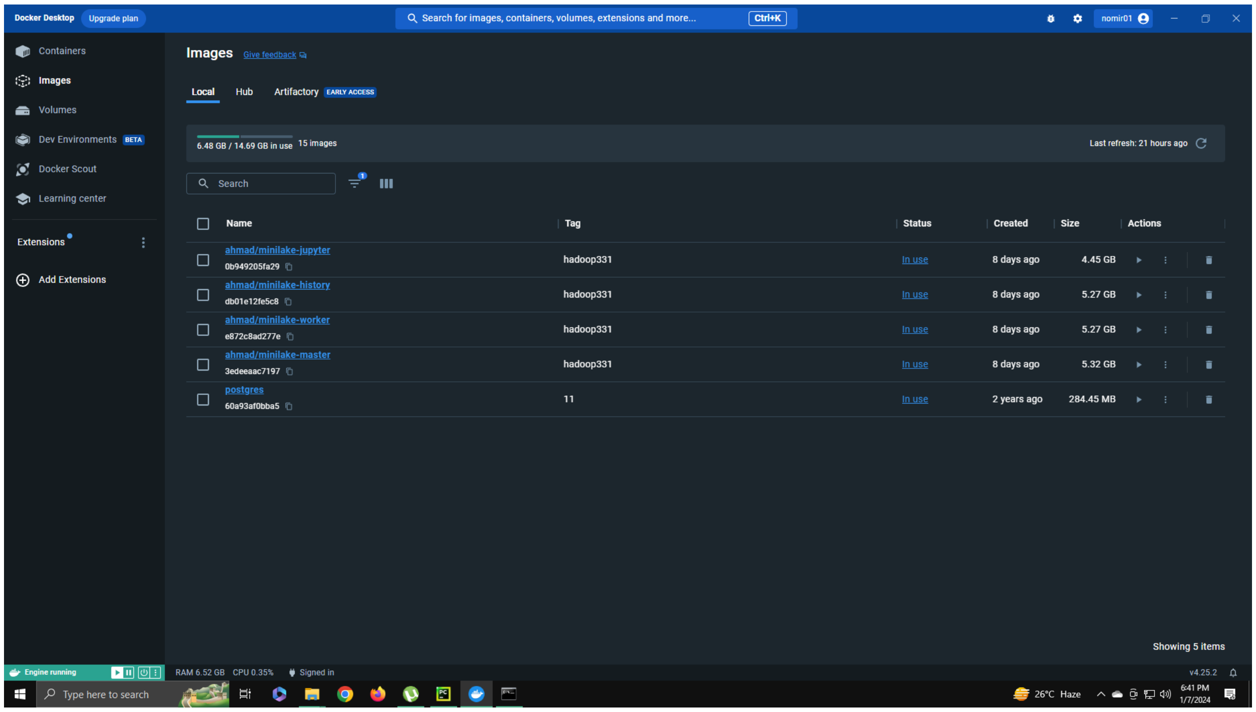
Container memory usage: 4.31GB / 7.54GB

Name	Image	Status	CPU (%)	Port(s)	Last started
minilake-main		Running (5/5)	1.57%		57 minutes ago
historyserver 9372e184fe8a	ahmad/minilake-history:hadoop331	Running	0.11%	18080:18080 Show all ports (2)	57 minutes ago
jupyter-node c32ac2762bf2	ahmad/minilake-jupyter:hadoop331	Running	0.06%	8888:8888	57 minutes ago
worker-node1 02df8a985f5f	ahmad/minilake-worker:hadoop331	Running	0.45%	8042:8042 Show all ports (3)	57 minutes ago
master-node 09ed2692712d	ahmad/minilake-master:hadoop331	Running	0.95%	10000:10000 Show all ports (8)	57 minutes ago
postgres 8b81c419efbc	postgres:11	Running	0%	5432:5432	57 minutes ago

Selected 6 of 6

Engine running · RAM 6.47 GB · CPU 0.35% · Signed in · v4.25.2

26°C Haze · 6:40 PM 1/7/2024



Images

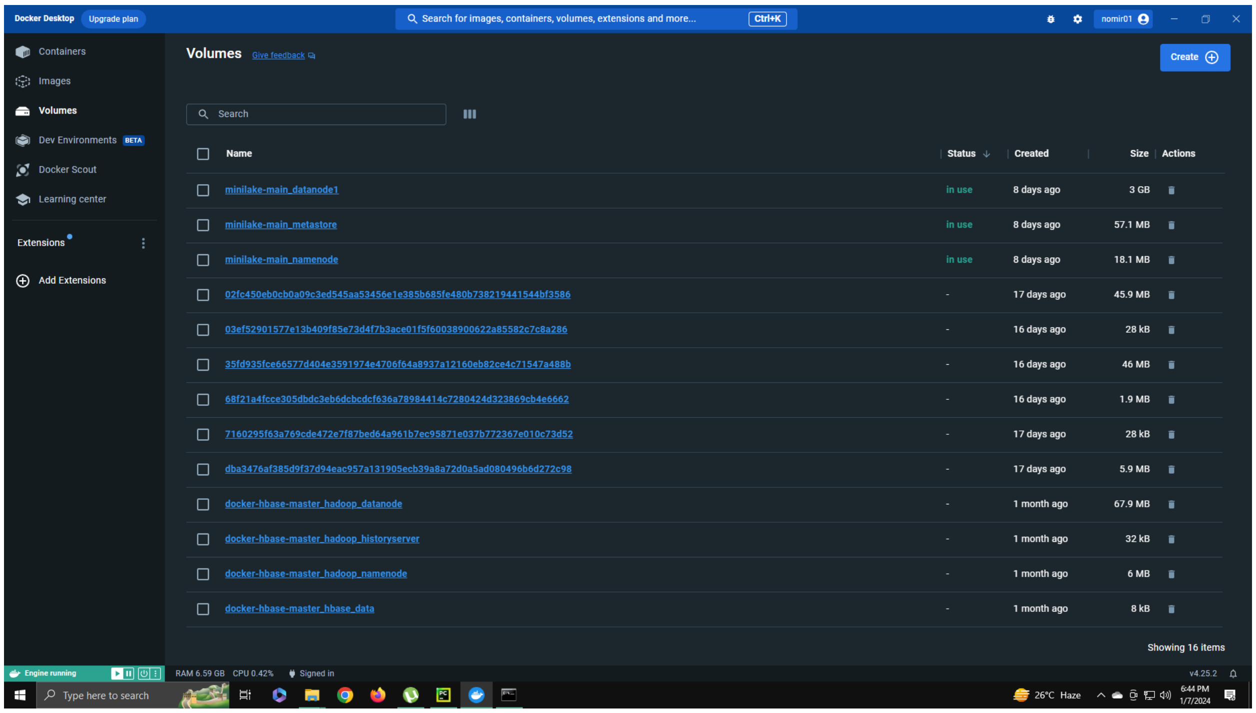
6.48 GB / 14.69 GB in use · 15 images · Last refresh: 21 hours ago

Name	Tag	Status	Created	Size
ahmad/minilake-jupyter 0b949205fa29	hadoop331	In use	8 days ago	4.45 GB
ahmad/minilake-history db01e12fe5c8	hadoop331	In use	8 days ago	5.27 GB
ahmad/minilake-worker e872c8ad277e	hadoop331	In use	8 days ago	5.27 GB
ahmad/minilake-master 3edeeaac7197	hadoop331	In use	8 days ago	5.32 GB
postgres 60a93af0bba5	11	In use	2 years ago	284.45 MB

Showing 5 items

Engine running · RAM 6.52 GB · CPU 0.35% · Signed in · v4.25.2

26°C Haze · 6:41 PM 1/7/2024

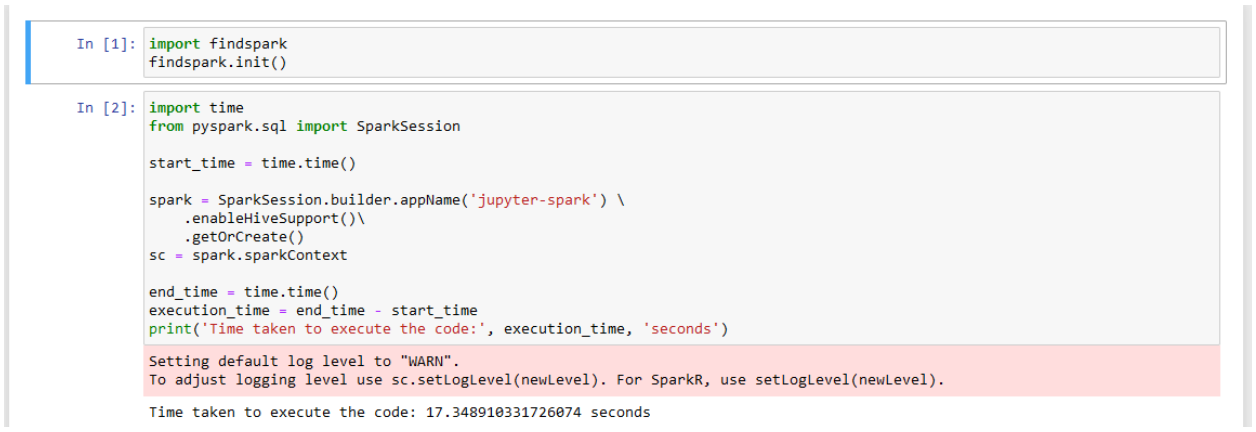


Volumes

Name	Status	Created	Size	Actions
minilake-main_datanode1	in use	8 days ago	3 GB	
minilake-main_metastore	in use	8 days ago	57.1 MB	
minilake-main_namenode	in use	8 days ago	18.1 MB	
02fc450eb0cb0a09c3ed545aa53456e1e385b685fe480b738219441544bf3586	-	17 days ago	45.9 MB	
03ef52901577e13b409f85e73d4f7b3ace01f5f60038900622a85582c7c8a286	-	16 days ago	28 kB	
35fd935fce66577d404e3591974e4706f64a8937a12160eb82ce4c71547a488b	-	16 days ago	46 MB	
68f21a4fcce305dbdc3eb6dcbcdcf636a78984414c7280424d323869cb4e6662	-	16 days ago	1.9 MB	
7160295f63a769cde472e7f87bed64a961b7ec95871e037b772367e010c73d52	-	17 days ago	28 kB	
dba3476af385d9f37d94eac957a131905ecb39a8a72d0a5ad080496b6d272c98	-	17 days ago	5.9 MB	
docker-hbase-master_hadoop_datanode	-	1 month ago	67.9 MB	
docker-hbase-master_hadoop_historyserver	-	1 month ago	32 kB	
docker-hbase-master_hadoop_namenode	-	1 month ago	6 MB	
docker-hbase-master_hbase_data	-	1 month ago	8 kB	

Showing 16 items

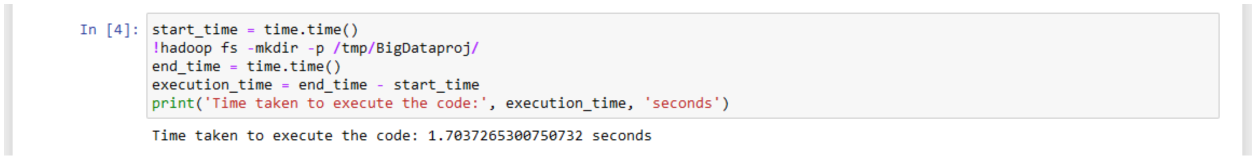
EXECUTION TIME



```
In [1]: import findspark
        findspark.init()
```

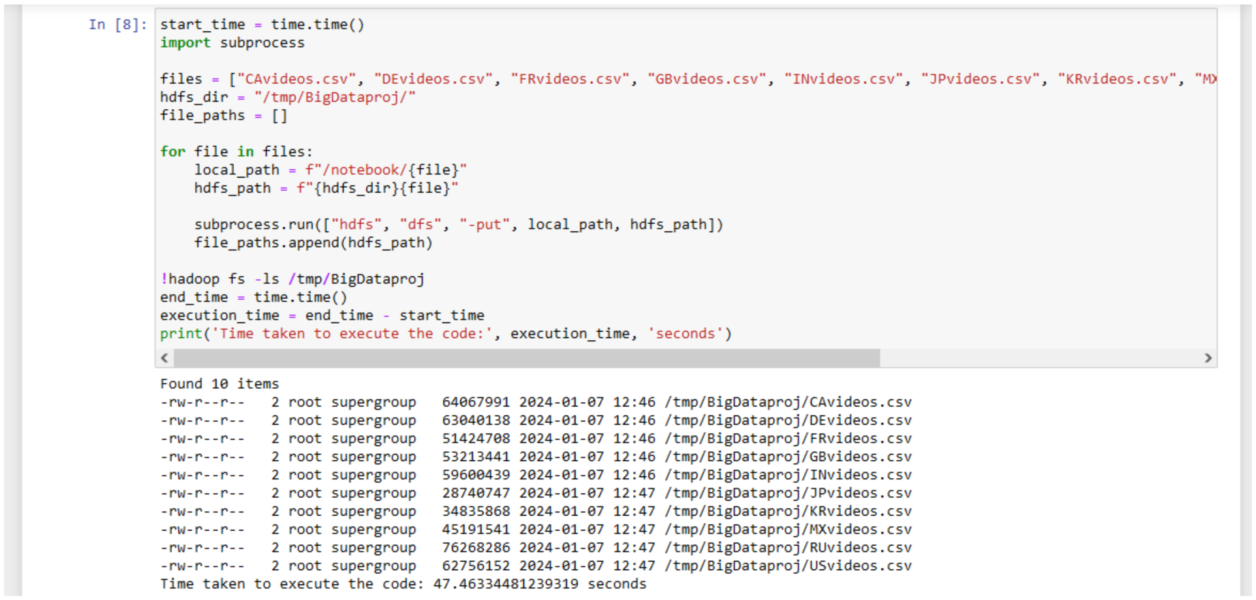
```
In [2]: import time
        from pyspark.sql import SparkSession

        start_time = time.time()

        spark = SparkSession.builder.appName('jupyter-spark') \
            .enableHiveSupport()\
            .getOrCreate()
        sc = spark.sparkContext

        end_time = time.time()
        execution_time = end_time - start_time
        print('Time taken to execute the code:', execution_time, 'seconds')
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

Time taken to execute the code: 17.348910331726074 seconds
```



```
In [4]: start_time = time.time()
        !hadoop fs -mkdir -p /tmp/BigDataproj/
        end_time = time.time()
        execution_time = end_time - start_time
        print('Time taken to execute the code:', execution_time, 'seconds')
```

```
Time taken to execute the code: 1.7037265300750732 seconds
```



```
In [8]: start_time = time.time()
        import subprocess

        files = ["CAvideos.csv", "DEvideos.csv", "FRvideos.csv", "GBvideos.csv", "INvideos.csv", "JPvideos.csv", "KRvideos.csv", "MX
        hdfs_dir = "/tmp/BigDataproj/"
        file_paths = []

        for file in files:
            local_path = f"/notebook/{file}"
            hdfs_path = f"{hdfs_dir}{file}"

            subprocess.run(["hdfs", "dfs", "-put", local_path, hdfs_path])
            file_paths.append(hdfs_path)

        !hadoop fs -ls /tmp/BigDataproj
        end_time = time.time()
        execution_time = end_time - start_time
        print('Time taken to execute the code:', execution_time, 'seconds')
```

```
Found 10 items
-rw-r--r--   2 root supergroup   64067991 2024-01-07 12:46 /tmp/BigDataproj/CAvideos.csv
-rw-r--r--   2 root supergroup   63040138 2024-01-07 12:46 /tmp/BigDataproj/DEvideos.csv
-rw-r--r--   2 root supergroup   51424708 2024-01-07 12:46 /tmp/BigDataproj/FRvideos.csv
-rw-r--r--   2 root supergroup   53213441 2024-01-07 12:46 /tmp/BigDataproj/GBvideos.csv
-rw-r--r--   2 root supergroup   59600439 2024-01-07 12:46 /tmp/BigDataproj/INvideos.csv
-rw-r--r--   2 root supergroup   28740747 2024-01-07 12:47 /tmp/BigDataproj/JPvideos.csv
-rw-r--r--   2 root supergroup   34835868 2024-01-07 12:47 /tmp/BigDataproj/KRvideos.csv
-rw-r--r--   2 root supergroup   45191541 2024-01-07 12:47 /tmp/BigDataproj/MXvideos.csv
-rw-r--r--   2 root supergroup   76268286 2024-01-07 12:47 /tmp/BigDataproj/RUvideos.csv
-rw-r--r--   2 root supergroup   62756152 2024-01-07 12:47 /tmp/BigDataproj/USvideos.csv
Time taken to execute the code: 47.46334481239319 seconds
```

In [34]:

```python
start_time = time.time()
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, BooleanType

schema = StructType([
    StructField("video_id", StringType(), True),
    StructField("trending_date", StringType(), True),
    StructField("title", StringType(), True),
    StructField("channel_title", StringType(), True),
    StructField("category_id", IntegerType(), True),
    StructField("publish_time", StringType(), True),
    StructField("tags", StringType(), True),
    StructField("views", IntegerType(), True),
    StructField("likes", IntegerType(), True),
    StructField("dislikes", IntegerType(), True),
    StructField("comment_count", IntegerType(), True),
    StructField("thumbnail_link", StringType(), True),
    StructField("comments_disabled", BooleanType(), True),
    StructField("ratings_disabled", BooleanType(), True),
    StructField("video_error_or_removed", BooleanType(), True),
    StructField("description", StringType(), True)
])
region = ["CA", "DE", "FR", "GB", "IN", "JP", "KR", "MX", "RU", "US"]
hive_table_prefix = "yt_table"

for i, file_path in enumerate(file_paths):
    dataframe = spark.read.option("header", "true").schema(schema).csv(file_path)
    dataframe.write.mode("overwrite").parquet(f"{hdfs_dir}/df_{i+1}.parquet")
    hive_table_name = f"{hive_table_prefix}_{region[i]}"
    spark.sql(f"CREATE TABLE IF NOT EXISTS {hive_table_name} USING PARQUET OPTIONS (PATH '{hdfs_dir}/df_{i+1}.parquet')")
    spark.sql(f"INSERT INTO TABLE {hive_table_name} SELECT * FROM parquet.`{hdfs_dir}/df_{i+1}.parquet`")

end_time = time.time()
execution_time = end_time - start_time
print('Time taken to execute the code:', execution_time, 'seconds')
```

```
[Stage 84:=============================>                            (1 + 1) / 2]
```

```
Time taken to execute the code: 26.581573247909546 seconds
```

In [35]:

```python
start_time = time.time()
tables = spark.sql("SHOW TABLES")
tables.show()
end_time = time.time()
execution_time = end_time - start_time
print('Time taken to execute the code:', execution_time, 'seconds')
```

```
+---------+-----------+-----------+
|namespace|  tableName|isTemporary|
+---------+-----------+-----------+
|  default|yt_table_ca|      false|
|  default|yt_table_de|      false|
|  default|yt_table_fr|      false|
|  default|yt_table_gb|      false|
|  default|yt_table_in|      false|
|  default|yt_table_jp|      false|
|  default|yt_table_kr|      false|
|  default|yt_table_mx|      false|
|  default|yt_table_ru|      false|
|  default|yt_table_us|      false|
+---------+-----------+-----------+

Time taken to execute the code: 0.08134651184082031 seconds
```

```
In [36]: start_time = time.time()
         for i in range(len(region)):
             df = spark.sql(f"DESCRIBE FORMATTED yt_table_{region[i]}")
             df.show(truncate=False)

         end_time = time.time()
         execution_time = end_time - start_time
         print('Time taken to execute the code:', execution_time, 'seconds')
```

```
|publish_time                |string     |null   |
|tags                        |string     |null   |
|views                       |int        |null   |
|likes                       |int        |null   |
|dislikes                    |int        |null   |
|comment_count               |int        |null   |
|thumbnail_link              |string     |null   |
|comments_disabled           |boolean    |null   |
|ratings_disabled            |boolean    |null   |
|video_error_or_removed      |boolean    |null   |
|description                 |string     |null   |
|                            |           |       |
|# Detailed Table Information|           |       |
|Database                    |default    |       |
|Table                       |yt_table_us|       |
+----------------------------+-----------+-------+
only showing top 20 rows

Time taken to execute the code: 0.6112241744995117 seconds
```

```
In [37]: start_time = time.time()
         from pyspark.sql.functions import col, when
         tables = ["yt_table_ca", "yt_table_de", "yt_table_fr", "yt_table_gb", "yt_table_in",
                   "yt_table_jp", "yt_table_kr", "yt_table_mx", "yt_table_ru", "yt_table_us"]

         pre_tables = []

         for table_name in tables:
             new_name = f"preprocessed_table_{table_name}"
             original_df = spark.sql(f"SELECT * FROM {table_name}")

             preprocessed_df = original_df.withColumn("title", when(col("title").isNotNull(), col("title")).otherwise("Unknown")) \
                 .withColumn("tags", when(col("tags").isNotNull(), col("tags")).otherwise("Unknown")) \
                 .withColumn("likes", when(col("likes").isNotNull(), col("likes")).otherwise(0)) \
                 .withColumn("dislikes", when(col("dislikes").isNotNull(), col("dislikes")).otherwise(0)) \
                 .withColumn("comment_count", when(col("comment_count").isNotNull(), col("comment_count")).otherwise(0)) \
                 .withColumn("description", when(col("description").isNotNull(), col("description")).otherwise("No description")) \
                 .withColumn("publish_year", col("publish_time").substr(1, 4).cast("int")) \
                 .withColumn("publish_month", col("publish_time").substr(6, 2).cast("int")) \
                 .withColumn("publish_day", col("publish_time").substr(9, 2).cast("int"))

             preprocessed_with_group_df = preprocessed_df.withColumn("group_column",
                 when(col("views") > 1000000, "HighViews") \
                 .when(col("likes") > 50000, "HighLikes") \
                 .when(col("dislikes") > 50000, "HighDislikes") \
                 .when(col("comment_count") > 50000, "HighComments") \
                 .otherwise("Other"))

             preprocessed_with_group_df.write.mode("overwrite").saveAsTable(new_name)
             pre_tables.append(new_name)

         tables = spark.sql("SHOW TABLES")
         tables.show()
         end_time = time.time()
         execution_time = end_time - start_time
         print('Time taken to execute the code:', execution_time, 'seconds')
```

```
[Stage 96:=====================================>                  (2 + 1) / 3]
```

```
[Stage 96:=======================================>                    (2 + 1) / 3]
```

```
+---------+--------------------+-----------+
|namespace|           tableName|isTemporary|
+---------+--------------------+-----------+
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|preprocessed_tabl...|      false|
|  default|         yt_table_ca|      false|
|  default|         yt_table_de|      false|
|  default|         yt_table_fr|      false|
|  default|         yt_table_gb|      false|
|  default|         yt_table_in|      false|
|  default|         yt_table_jp|      false|
|  default|         yt_table_kr|      false|
|  default|         yt_table_mx|      false|
|  default|         yt_table_ru|      false|
|  default|         yt_table_us|      false|
+---------+--------------------+-----------+

Time taken to execute the code: 17.15162086486816 4 seconds
```

In [38]:

```python
start_time = time.time()
output_dir = "/tmp/csv_output"
for table_name in pre_tables:
    df_spark = spark.table(table_name)
    df_spark.write.csv(f"{output_dir}/{table_name}", header=True, mode="overwrite")

!hadoop fs -ls /tmp/csv_output
end_time = time.time()
execution_time = end_time - start_time
print('Time taken to execute the code:', execution_time, 'seconds')
```

```
Found 10 items
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_ca
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_de
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_fr
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_gb
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_in
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_jp
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_kr
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_mx
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_ru
drwxr-xr-x   - root supergroup          0 2024-01-07 12:53 /tmp/csv_output/preprocessed_table_yt_table_us
Time taken to execute the code: 18.515202283859253 seconds
```

In [39]:

```python
start_time = time.time()
import subprocess
output_dir = "/tmp/csv_output"
local_output_dir = "/notebook"
subprocess.run(["hadoop", "fs", "-get", f"{output_dir}/*", local_output_dir])
end_time = time.time()
execution_time = end_time - start_time
print('Time taken to execute the code:', execution_time, 'seconds')
```

```
Time taken to execute the code: 37.85867238044739 seconds
```

In [40]:

```python
start_time = time.time()
spark.stop()
end_time = time.time()
execution_time = end_time - start_time
print('Time taken to execute the code:', execution_time, 'seconds')
```

```
Time taken to execute the code: 1.1115782260894775 seconds
```

API USED

No API was used in the project.

MACHINE CONFIGURATION

- **Processor:** AMD Ryzen 5 3600
- **Motherboard:** Gigabyte b450m ds3h v2
- **RAM:** Heatsink 16gb 3600mhz
- **GPU:** RTX 2060 super
- **SSD:** Adata 256gb
- **OS:** Windows 10 Pro

ATTACHMENTS IN FOLDER:

- **Docker:** NO additional docker files were used aside from "minilake", So no files are attached.

- **Data:** This is the link to original data

https://www.kaggle.com/datasets/datasnaek/youtube-new/data

Although an example csv file, named "USvideos", for the above link is saved in "Data" folder.

- **Code Base:** Notebook for the jupyter is in "Code" folder with the name of "Untitled5.ipynb", and the py file for the dashboard is also in the "Code" folder with the name of "main.py".

- **Video:** A video of our project is also attached.