Spotify Data Analysis

Final Project – Data System Architecture

Introduction:

In this project, our team is venturing into the expansive field of big data analytics. We aim to explore the intricate world of data, moving through the complexities of tasks like data cleaning, processing, analysis, and visualization.

Each command and operation in our toolkit play a vital role in constructing a strong data pipeline. Every step contributes to shaping a complete structure, reflecting the practical challenges and solutions found in the real world of big data analytics.

Approach:

For our data analytics journey, we adopted a two-tiered approach, leveraging Hive for efficient data cleaning and PySpark for a comprehensive exploration of data insights, manipulation, and visualization.

Hive, with its SQL-like interface, proved instrumental in ensuring the cleanliness and uniformity of our dataset. On the other hand, PySpark, a robust and versatile tool, served as the powerhouse for deriving meaningful insights, manipulating data with agility, and creating visual representations that enhance the interpretability of our findings.

This dual-tool strategy allowed us to harness the strengths of each platform, creating a dynamic and effective workflow for our big data analysis.

Methodology:

Data Cleaning with Hive:

- Dataset Understanding: Conducted a thorough exploration of the raw data to identify inconsistencies, missing values, and anomalies.
- HiveQL Operations: Utilized Hive's SQL-like capabilities to execute operations for data cleaning, ensuring a standardized and reliable dataset.

Data Analysis with PySpark:

- Exploratory Data Analysis (EDA): Applied PySpark for EDA, investigating patterns, trends, and distributions within the dataset.
- Insightful Queries: Formulated PySpark queries to extract key insights, focusing on specific artists and relevant chart metrics.

Data Manipulation and Visualization:

- Transformation Operations: Used PySpark for agile data manipulation, transforming the dataset for further analysis.
- Visualization Techniques: Employed PySpark's capabilities for creating visualizations to enhance the presentation and interpretation of our findings.

Challenges Faced:

Our journey through the Spotify charts dataset presented several challenges, each contributing to our growth and refinement of the analytical process. Key challenges encountered include:

Data Quality and Consistency:

Challenge: Inconsistent data formats and quality issues within the raw dataset.

Solution: Implemented rigorous validation checks and preprocessing steps in Hive to enhance data quality and ensure uniformity.

Scalability Issues with PySpark:

Challenge: PySpark's scalability limitations encountered during intensive data processing tasks.

Solution: Employed optimization techniques, distributed computing strategies, and explored alternative PySpark configurations to enhance scalability.

Complex Data Manipulation Requirements:

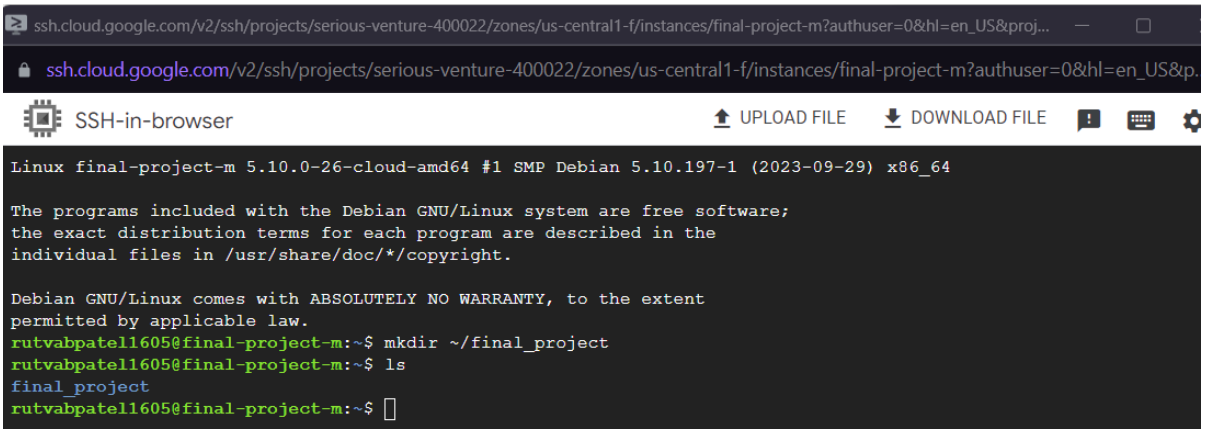
Challenge: Sophisticated data manipulation requirements, especially when filtering specific artists and chart metrics.

Solution: Leveraged PySpark's expressive capabilities and functions to efficiently handle complex data manipulation tasks.

Code Review:

Directory Creation

Initialized the project directory on the local machine and Hadoop Distributed File System (HDFS).



Hadoop Directory Structure

Verified the Hadoop directory structure and file presence.

```
ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?authuser=0&hl=en_US&proj...
ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?authuser=0&hl=en_US&p...
SSH-in-browser                                                    UPLOAD FILE   DOWNLOAD FILE

the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
rutvabpatel1605@final-project-m:~$ mkdir ~/final_project
rutvabpatel1605@final-project-m:~$ ls
final_project
rutvabpatel1605@final-project-m:~$ hadoop fs -ls /user
Found 11 items
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/dataproc
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/hbase
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/hdfs
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/hive
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/mapred
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/pig
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/solr
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/spark
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/yarn
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/zeppelin
drwxrwxrwt   - hdfs hadoop          0 2023-12-06 03:57 /user/zookeeper
rutvabpatel1605@final-project-m:~$ hadoop fs -mkdir /user/final_project
rutvabpatel1605@final-project-m:~$ hadoop fs -ls /user
Found 12 items
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/dataproc
drwxr-xr-x   - rutvabpatel1605 hadoop          0 2023-12-06 04:58 /user/final_project
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/hbase
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/hdfs
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/hive
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/mapred
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/pig
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/solr
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/spark
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/yarn
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/zeppelin
drwxrwxrwt   - hdfs            hadoop          0 2023-12-06 03:57 /user/zookeeper
rutvabpatel1605@final-project-m:~$
```

Data Acquisition

Copied Spotify charts data (charts.csv) from Google Storage to the local project directory.

```
ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?authuser=0&hl=en_US&projec...
ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?authuser=0&hl=en_US&proj...
SSH-in-browser                                                    UPLOAD FILE   DOWNLOAD FILE

rutvabpatel1605@final-project-m:~$ gsutil cp gs://data-system-architecture-final-project/charts.csv ~/final_proje
ct/spotify_data.csv
Copying gs://data-system-architecture-final-project/charts.csv...
\ [1 files][  3.2 GiB/  3.2 GiB]   51.7 MiB/s
Operation completed over 1 objects/3.2 GiB.
rutvabpatel1605@final-project-m:~$ ls ~/final_project
spotify_data.csv
rutvabpatel1605@final-project-m:~$ head -n 5 ~/final_project/spotify_data.csv
Title,Rank,Event_Date,Artist,Url,Region,Chart,Trend,Streams
Chantaje (feat. Maluma),1,2017-01-01,Shakira,https://open.spotify.com/track/6mICuAdrwEjh6Y6lroV2Kg,Argentina,top2
00,SAME_POSITION,253019
Vente Pa' Ca (feat. Maluma),2,2017-01-01,Ricky Martin,https://open.spotify.com/track/7DM4BPaS7uofFul3ywMe46,Argen
tina,top200,MOVE_UP,223988
Reggaetón Lento (Bailemos),3,2017-01-01,CNCO,https://open.spotify.com/track/3AEZUABDXNtecAOSC1qTfo,Argentina,top2
00,MOVE_DOWN,210943
Safari,4,2017-01-01,"J Balvin, Pharrell Williams, BIA, Sky",https://open.spotify.com/track/6rQSrBHf7HlZjtcMZ4S4bO
,Argentina,top200,SAME_POSITION,173865
rutvabpatel1605@final-project-m:~$
```

Data Upload to Hadoop & Data Inspection

Uploaded the Spotify charts data to Hadoop for distributed processing and checked the first 5 lines of the uploaded CSV file.

```
ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?authuser=0&hl=en_US&projec...
ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?authuser=0&hl=en_US&proj...
SSH-in-browser                                   UPLOAD FILE   DOWNLOAD FILE

rutvabpatel1605@final-project-m:~$ hadoop fs -copyFromLocal ~/final_project/spotify_data.csv /user/final_project/
.
rutvabpatel1605@final-project-m:~$ hadoop fs -ls /user/final_project
Found 1 items
-rw-r--r--   1 rutvabpatel1605 hadoop 3483476394 2023-12-06 05:03 /user/final_project/spotify_data.csv
rutvabpatel1605@final-project-m:~$ hadoop fs -cat /user/final_project/spotify_data.csv | head -n 5
Title,Rank,Event_Date,Artist,Url,Region,Chart,Trend,Streams
Chantaje (feat. Maluma),1,2017-01-01,Shakira,https://open.spotify.com/track/6mICuAdrwEjh6Y6lroV2Kg,Argentina,top2
00,SAME_POSITION,253019
Vente Pa' Ca (feat. Maluma),2,2017-01-01,Ricky Martin,https://open.spotify.com/track/7DM4BPaS7uofFul3ywMe46,Argen
tina,top200,MOVE_UP,223988
Reggaetón Lento (Bailemos),3,2017-01-01,CNCO,https://open.spotify.com/track/3AEZUABDXNtecAOSC1qTfo,Argentina,top2
00,MOVE_DOWN,210943
Safari,4,2017-01-01,"J Balvin, Pharrell Williams, BIA, Sky",https://open.spotify.com/track/6rQSrBHf7HlZjtcMZ4S4bO
,Argentina,top200,SAME_POSITION,173865
cat: Unable to write to output stream.
rutvabpatel1605@final-project-m:~$
```

Hive Database and Table Creation

Database Creation

Created a Hive database named 'final_project'.

```
der.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
Hive Session ID = 59619bb0-b468-43fd-99c7-19069acf8ca5

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.pr
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.hive.common.StringInternUt
ive-common-3.1.3.jar) to field java.net.URI.string
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.hi
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflectiv
WARNING: All illegal access operations will be denied in a future release
Hive Session ID = 701cce6f-5af5-4bbf-82c9-18f1270aacac
hive> show databases;
OK
default
fina
final_project
Time taken: 1.948 seconds, Fetched: 3 row(s)
hive>
```

Table Creation

Created an external table named ‘spotify_table’ in the ‘final_project’ database to store Spotify charts data.

```
WARNING: All illegal access operations will be denied in a future release
Hive Session ID = 7f6e4065-6b5b-4a52-9edd-b0dbc063d135
hive> use final_project;
OK
Time taken: 1.394 seconds
hive> Create external table if not exists spotify_table (Title STRING, Rank INT, Event_Date DATE, Artist STRING
,Url STRING, Region STRING, Chart STRING, Trend STRING, Streams INT)ROW FORMAT DELIMITED FIELDS TERMINATED BY '
,' stored as textfile location "/user/final_project" TBLPROPERTIES("skip.header.line.count"="1");
OK
Time taken: 0.478 seconds
hive>
```

Data Analysis with HiveQL

Initial Data Exploration

Executed queries to explore the contents of the ‘spotify_table’.

```
,' stored as textfile location "/user/final_project" TBLPROPERTIES("skip.header.line.count"="1");
OK
Time taken: 0.478 seconds
hive> select * from spotify_table limit 10;
OK
Chantaje (feat. Maluma) 1       2017-01-01      Shakira https://open.spotify.com/track/6mICuAdrwEjh6Y6lroV2Kg A
rgentina        top200  SAME_POSITION   253019
Vente Pa' Ca (feat. Maluma)     2       2017-01-01      Ricky Martin    https://open.spotify.com/track/7DM4BPaS
7uofFul3ywMe46  Argentina       top200  MOVE_UP 223988
Reggaetón Lento (Bailemos)      3       2017-01-01      CNCO    https://open.spotify.com/track/3AEZUABDXNtecAOS
C1qTfo  Argentina       top200  MOVE_DOWN       210943
Safari  4       2017-01-01      "J Balvin        Pharrell Williams       BIA     Sky"    https://open.spotify.co
m/track/6rQSrBHf7HlZjtcMZ4S4bO  NULL
Shaky Shaky     5       2017-01-01      Daddy Yankee    https://open.spotify.com/track/58IL315gMSTD37DOZPJ2hf A
rgentina        top200  MOVE_UP 153956
Traicionera     6       2017-01-01      Sebastian Yatra https://open.spotify.com/track/5J1c3M4EldCfNxXwrwt8mT A
rgentina        top200  MOVE_DOWN       151140
Cuando Se Pone a Bailar 7       2017-01-01      Rombai  https://open.spotify.com/track/1MpKZi1zTXpERKwxmOu1PH A
rgentina        top200  MOVE_DOWN       148369
Otra vez (feat. J Balvin)       8       2017-01-01      Zion & Lennox   https://open.spotify.com/track/3QwBODjS
EzelZyVjxPOHdq  Argentina       top200  MOVE_DOWN       143004
La Bicicleta    9       2017-01-01      "Carlos Vives    Shakira"       https://open.spotify.com/track/0sXvAOmX
gjR2QUqLK1M1tU  Argentina       top200  NULL
Dile Que Tu Me Quieres  10      2017-01-01      Ozuna   https://open.spotify.com/track/20ZAJdsKB5IGbGj4ilRt2o A
rgentina        top200  MOVE_DOWN       112012
Time taken: 2.803 seconds, Fetched: 10 row(s)
hive>
```

Extracting Year from Date

Utilized the SUBSTR function to extract the year from the 'Event_Date' column.

```
hive> SELECT Event_Date, SUBSTR(Event_Date, 1, 4) AS Event_Year FROM spotify_table limit 10;
Query ID = rutvabpatel1605_20231207172857_9d737e3e-cf87-470e-9c3b-14c3c9b44ed3
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1701841694457_0023)

--------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 13.31 s
--------------------------------------------------------------------------------------------
OK
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
2017-01-01      2017
Time taken: 16.69 seconds, Fetched: 10 row(s)
hive>
```

Creating Filtered Table

Created a new table named 'fp_data_table' with selected columns and artists.

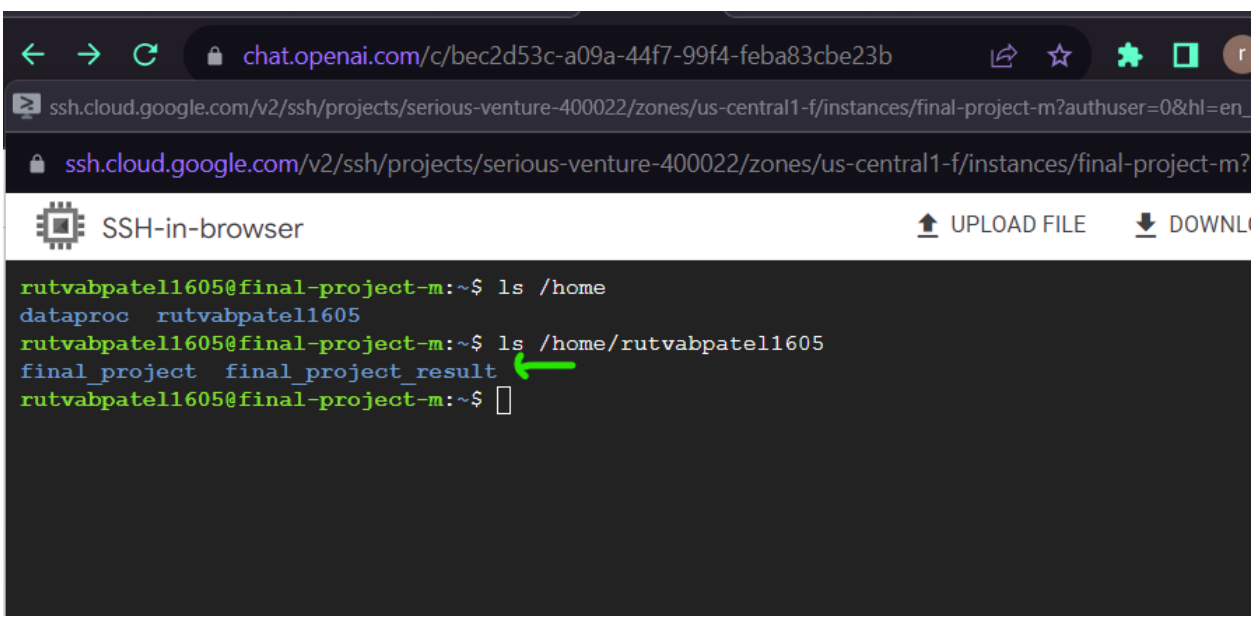
```
hive> use final_project;
OK
Time taken: 0.589 seconds
hive> select * from fp_data_table limit 10;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'fp_data_table'
hive> Create table if not exists fp_data_table AS
    > select Title, Rank, SUBSTR(Event_Date, 1, 4) AS Event_Year,
    > Artist, Region, Chart, Trend, Streams from spotify_table
    > WHERE Artist IN ('One Direction', 'The Weeknd', 'Drake','ZAYN','Post Malone','Coldplay',
'Ed Sheeran', 'Bruno Mars', 'Twenty One Pilots', 'Rihanna');
Query ID = rutvabpatel1605_20231206222346_86e314d8-122d-4769-91c6-86316f523466
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1701841694457_0013)

--------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 111.40 s
--------------------------------------------------------------------------------------------
Moving data to directory hdfs://final-project-m/user/hive/warehouse/final_project.db/fp_data_t
able
OK
Time taken: 298.201 seconds
hive>
```

```
Moving data to directory hdfs://final-project-m/user/hive/warehouse/final_project.db/fp_data_t
able
OK
Time taken: 298.201 seconds
hive> select * from fp_data_table limit 20;
OK
24K Magic       34      2017    Bruno Mars      Argentina       top200  MOVE_DOWN       58801
Ride    112     2017    Twenty One Pilots       Argentina       top200  MOVE_DOWN       22015
Heathens        134     2017    Twenty One Pilots       Argentina       top200  MOVE_DOWN    1
9334
Hymn for the Weekend - Seeb Remix       140     2017    Coldplay        Argentina       top200
MOVE_DOWN       18823
Stressed Out    164     2017    Twenty One Pilots       Argentina       top200  MOVE_DOWN    1
6845
Adventure of a Lifetime 188     2017    Coldplay        Argentina       top200  MOVE_DOWN    1
4834
24K Magic       12      2017    Bruno Mars      Australia       top200  SAME_POSITION   72054
Fake Love       20      2017    Drake   Australia       top200  MOVE_DOWN       62524
Heathens        54      2017    Twenty One Pilots       Australia       top200  MOVE_UP 28821
Party Monster   66      2017    The Weeknd      Australia       top200  MOVE_DOWN       24466
Hymn for the Weekend - Seeb Remix       80      2017    Coldplay        Australia       top200
MOVE_UP 20557
Ride    91      2017    Twenty One Pilots       Australia       top200  MOVE_UP 19163
Stressed Out    100     2017    Twenty One Pilots       Australia       top200  MOVE_UP 17845
Needed Me       105     2017    Rihanna Australia       top200  MOVE_UP 16772
Rockin' 107     2017    The Weeknd      Australia       top200  MOVE_UP 16651
Thinking out Loud       109     2017    Ed Sheeran      Australia       top200  MOVE_UP 16061
Controlla       111     2017    Drake   Australia       top200  MOVE_UP 15990
PILLOWTALK      124     2017    ZAYN    Australia       top200  MOVE_UP 14087
Versace on the Floor    127     2017    Bruno Mars      Australia       top200  NEW_ENTRY    1
3938
Can't Feel My Face      130     2017    The Weeknd      Australia       top200  MOVE_DOWN    1
3682
Time taken: 0.491 seconds, Fetched: 20 row(s)
hive>
```

Listing Home Directory Contents

Verified the existing directory structure before proceeding.

'final_project_result' directory will be used to store the results and outputs of the final project



Data Export Using Hive

Exporting Data to Local Directory

This HiveQL command exports data from the Hive table 'fp_data_table' to a local directory named 'final_project_result'. The data is stored in delimited format, separated by commas.

```
hive> insert overwrite local directory '/home/rutvabpatel1605/final_project_result'
    > row format delimited
    > fields terminated by ','
    > select * from fp_data_table;
Query ID = rutvabpatel1605_20231207175629_f2337954-7fb1-47ed-8451-31e66cfebf44
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1701841694457_0024)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 15.58 s
--------------------------------------------------------------------------------
Moving data to local directory /home/rutvabpatel1605/final_project_result
OK
Time taken: 22.025 seconds
hive>
```

Listing Contents of the Exported Directory

This command lists the contents of the 'final_project_result' directory to verify that the export process was successful. The output typically includes a file named '000000_0' containing the exported data.

```
rutvabpatel1605@final-project-m:~$ ls ~/final_project_result
000000_0
rutvabpatel1605@final-project-m:~$
```

Renaming the Exported File

This command renames the exported file '000000_0' to 'final_dataset.csv' for clarity and consistency in naming conventions.

ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?authuser=0&hl=en_US&

ssh.cloud.google.com/v2/ssh/projects/serious-venture-400022/zones/us-central1-f/instances/final-project-m?aut

SSH-in-browser UPLOAD FILE DOWNLOAD

```
rutvabpatel1605@final-project-m:~$ ls ~/final_project_result
000000_0
rutvabpatel1605@final-project-m:~$ cd ~/final_project_result
rutvabpatel1605@final-project-m:~/final_project_result$ mv 000000_0 final_dataset.csv;
rutvabpatel1605@final-project-m:~/final_project_result$ ls
final_dataset.csv
rutvabpatel1605@final-project-m:~/final_project_result$
```

Uploading to Google Storage

Moving the File to Google Storage

This command moves the finalized dataset file ('final_dataset.csv') from the local environment to Google Storage under the specified destination ('gs://final-project-dsa/'). The dataset is now securely stored in the cloud for broader accessibility.

```
rutvabpatel1605@final-project-m:~/final_project_result$ gsutil mv final_dataset.csv gs://final-project-dsa/fina
l_dataset.csv
Copying file://final_dataset.csv [Content-Type=text/csv]...
Removing file://final_dataset.csv...

Operation completed over 1 objects/71.2 MiB.
rutvabpatel1605@final-project-m:~/final_project_result$
```

charts.csv	3.2 GB	text/csv	Dec 6, 2023, 1:01:57 AM	Standard	Dec 6, 2023, 1:01:57 AM	Not public	–	Go
final_dataset.csv	71.2 MB	text/csv	Dec 7, 2023, 1:05:04 PM	Standard	Dec 7, 2023, 1:05:04 PM	Not public	–	Go

Data Processing:

Spark Session & Data Loading:

The code initializes a Spark session named "final-project" using PySpark. This session serves as the entry point for leveraging Apache Spark functionalities, such as distributed data processing and DataFrame operations, in a Python script or application.

Loads a CSV file named "Final_Dataset_Result.csv" from a cloud storage location ('gs://final-project-dsa/') into a PySpark DataFrame named df. It specifies that the schema should be inferred, considers the first row as the header, and uses a comma (',') as the delimiter for parsing the CSV data.

```
In [1]:  #Starting the Spark Session
         from pyspark.sql import SparkSession

         spark = SparkSession.builder.appName("final-project").getOrCreate()

         23/12/07 06:00:08 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.

In [2]:  #Loading the data from the cloud:

         path = 'gs://final-project-dsa/Final_Dataset_Result.csv'
         file_type = "csv"

         #CSV
         infer_schema = 'true'
         first_row_is_header = 'true'
         delimiter = ','


         #Import the csv file data
         df = spark.read.format(file_type)\
             .option("inferSchema",infer_schema)\
             .option("header", first_row_is_header) \
             .option("sep", delimiter)\
             .load(path)
```

Data Manipulation:

The code calculates and prints the total row count of the DataFrame df.

The code filters out rows in the DataFrame df where the 'Stream' column is not equal to '\N', removes any remaining null values in the 'Stream' column.

```
In [3]:  #Checking the total row count present in the dataset
         row_count = df.count()
         print("Total Row Count: ", row_count)

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

         Total Row Count:  1048575

In [4]:  df = df[df['Stream'] != '\\N'].dropna(subset=['Stream'])

In [5]:  row_count = df.count()
         print("Total Row Count after removing null values: ", row_count)

         [Stage 5:>                                                      (0 + 2) / 2]

         Total Row Count after removing null values:  981185

In [6]:  df.printSchema()

         root
          |-- Title: string (nullable = true)
          |-- Rank: integer (nullable = true)
          |-- Year: integer (nullable = true)
          |-- Artist: string (nullable = true)
          |-- Region: string (nullable = true)
          |-- Chart: string (nullable = true)
          |-- Trend: string (nullable = true)
          |-- Stream: string (nullable = true)
```

Prints the updated schema and the first 10 rows of the DataFrame.

In [7]:
```
df.show(10)
```

```
+--------------------+----+----+-----------------+---------+------+-------------+------+
|               Title|Rank|Year|           Artist|   Region| Chart|        Trend|Stream|
+--------------------+----+----+-----------------+---------+------+-------------+------+
|                Ride| 112|2017|Twenty One Pilots|Argentina|top200|    MOVE_DOWN| 22015|
|            Heathens| 134|2017|Twenty One Pilots|Argentina|top200|    MOVE_DOWN| 19334|
|Hymn for the Week...| 140|2017|         Coldplay|Argentina|top200|    MOVE_DOWN| 18823|
|        Stressed Out| 164|2017|Twenty One Pilots|Argentina|top200|    MOVE_DOWN| 16845|
|Adventure of a Li...| 188|2017|         Coldplay|Argentina|top200|    MOVE_DOWN| 14834|
|           24K Magic|  12|2017|       Bruno Mars|Australia|top200|SAME_POSITION| 72054|
|           Fake Love|  20|2017|            Drake|Australia|top200|    MOVE_DOWN| 62524|
|            Heathens|  54|2017|Twenty One Pilots|Australia|top200|      MOVE_UP| 28821|
|       Party Monster|  66|2017|       The Weeknd|Australia|top200|    MOVE_DOWN| 24466|
|Hymn for the Week...|  80|2017|         Coldplay|Australia|top200|      MOVE_UP| 20557|
+--------------------+----+----+-----------------+---------+------+-------------+------+
only showing top 10 rows
```

Insight Discovery:

Aggregation by Year and Region: Total Streams

The code utilizes PySpark's DataFrame API to aggregate the total streams for each combination of year and region from the DataFrame df. The results are displayed, showing the summed streams as "TotalStreams" in a new DataFrame named “total_streams_by_year_region”.

In [8]:
```
#Aggregation by Year and Region
from pyspark.sql.functions import sum

#The total streams for each combination of year and region
total_streams_by_year_region = df.groupBy("Year", "Region").agg(sum("Stream").alias("TotalStreams"))
total_streams_by_year_region.show()
```

```
[Stage 9:============================>                            (1 + 1) / 2]

+----+-------------+------------+
|Year|       Region|TotalStreams|
+----+-------------+------------+
|2019|      Estonia|   3687917.0|
|2017|     Honduras|   5526259.0|
|2017|    Lithuania|   2487193.0|
|2019|       Canada|4.36392581E8|
|2019|     Paraguay|   4153911.0|
|2019|United States|2.69949996E9|
|2018|      Romania|   6976044.0|
|2019|        Spain| 7.4312117E7|
|2020|      Hungary| 1.9060842E7|
|2018|      Bolivia|   4168105.0|
|2020|       Israel| 1.4914685E7|
|2020|       Latvia|   4309045.0|
|2018|       Norway|1.41605217E8|
|2020|      Vietnam|   8321781.0|
|2019|      Belgium| 5.1379833E7|
|2019|     Malaysia| 6.4653862E7|
|2019|       Panama|   3610330.0|
|2019|       Norway|1.18586776E8|
|2018|      Denmark|1.14516952E8|
|2018|      Finland| 5.0443215E7|
+----+-------------+------------+
```

Filtering Data by Year and Region: 2019, Spain

The code filters the DataFrame df to include only data from the year 2019 and the region "Spain" using PySpark's DataFrame API. The results are displayed in a new DataFrame named “updated_data_by_year_region”.

In [10]:

```
#Filter the DataFrame to include only data from a specific year and region
from pyspark.sql.functions import col

updated_data_by_year_region = df.filter((col("Year") == 2019) & (col("Region") == "Spain"))
updated_data_by_year_region.show()
```

```
[Stage 12:>                                                         (0 + 1) / 1]

+------------+----+----+-----------+------+------+-------------+------+
|       Title|Rank|Year|     Artist|Region| Chart|        Trend|Stream|
+------------+----+----+-----------+------+------+-------------+------+
|Shape of You| 129|2019| Ed Sheeran| Spain|top200|      MOVE_UP| 26523|
|        Wow.| 142|2019|Post Malone| Spain|top200|      MOVE_UP| 25478|
|  God's Plan| 167|2019|      Drake| Spain|top200|    NEW_ENTRY| 22711|
|     Perfect| 179|2019| Ed Sheeran| Spain|top200|      MOVE_UP| 21588|
|        Wow.| 129|2019|Post Malone| Spain|top200|      MOVE_UP| 38448|
|Shape of You| 136|2019| Ed Sheeran| Spain|top200|    MOVE_DOWN| 36794|
|     Perfect| 174|2019| Ed Sheeran| Spain|top200|      MOVE_UP| 30356|
|  God's Plan| 189|2019|      Drake| Spain|top200|    MOVE_DOWN| 27920|
|        Wow.| 127|2019|Post Malone| Spain|top200|      MOVE_UP| 38490|
|Shape of You| 133|2019| Ed Sheeran| Spain|top200|      MOVE_UP| 37510|
|     Perfect| 151|2019| Ed Sheeran| Spain|top200|      MOVE_UP| 33714|
|  God's Plan| 173|2019|      Drake| Spain|top200|      MOVE_UP| 31095|
|        Wow.| 127|2019|Post Malone| Spain|top200|SAME_POSITION| 39026|
|Shape of You| 134|2019| Ed Sheeran| Spain|top200|    MOVE_DOWN| 37120|
|     Perfect| 170|2019| Ed Sheeran| Spain|top200|    MOVE_DOWN| 31600|
|  God's Plan| 185|2019|      Drake| Spain|top200|    MOVE_DOWN| 29397|
|  God's Plan| 194|2019|      Drake| Spain|top200|    MOVE_DOWN| 25955|
|        Wow.| 129|2019|Post Malone| Spain|top200|    MOVE_DOWN| 35237|
|Shape of You| 139|2019| Ed Sheeran| Spain|top200|    MOVE_DOWN| 32790|
|     Perfect| 171|2019| Ed Sheeran| Spain|top200|    MOVE_DOWN| 28251|
+------------+----+----+-----------+------+------+-------------+------+
only showing top 20 rows
```

FINER Questions:

1) **How does the distribution of trending music tracks vary throughout the year?**

2) **Which five artists have the highest number of songs in the dataset, and how does the distribution of their songs compare to each other?**

3) **For a specific artist (e.g., “Ed Shareen"), how does the total stream count vary throughout the years?**

Data Visualization - 1

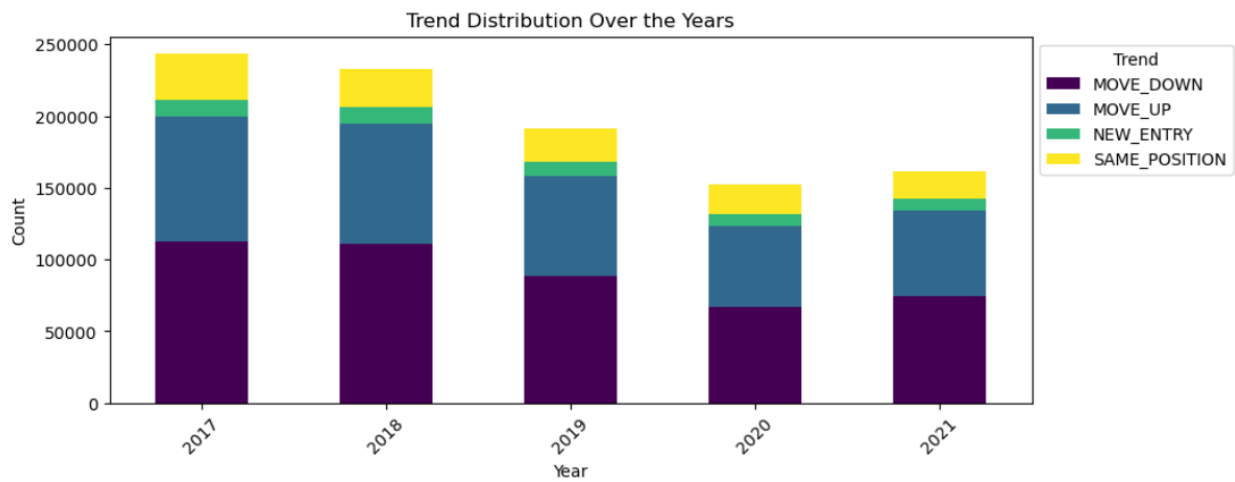
Yearly Trends in Music Patterns

The code analyzes and visualizes the distribution of music trends over the years. It groups the DataFrame df by "Year" and "Trend," calculates the count of each trend, and plots the trend distribution using a stacked bar chart. The resulting visualization provides insights into the yearly patterns of music trends.

In [20]:

```
from pyspark.sql.functions import count
import matplotlib.pyplot as plt

# Group by Year and Trend, calculate the count of each trend
trend_counts_by_year = df.groupBy("Year", "Trend").agg(count("Trend").alias("TrendCount")).orderBy("Year", "Trend")

# Convert Spark DataFrame to Pandas for easier plotting
pandas_df = trend_counts_by_year.toPandas()

# Pivot the data for better visualization
pivoted_df = pandas_df.pivot(index="Year", columns="Trend", values="TrendCount").fillna(0)

# Plotting the trend distribution over the years
plt.figure(figsize=(10, 4))
pivoted_df.plot(kind='bar', stacked=True, colormap='viridis', figsize=(10, 4))
plt.title("Trend Distribution Over the Years")
plt.xlabel("Year")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.legend(title="Trend", bbox_to_anchor=(1, 1))
plt.show()
```

```
<Figure size 1000x400 with 0 Axes>
```



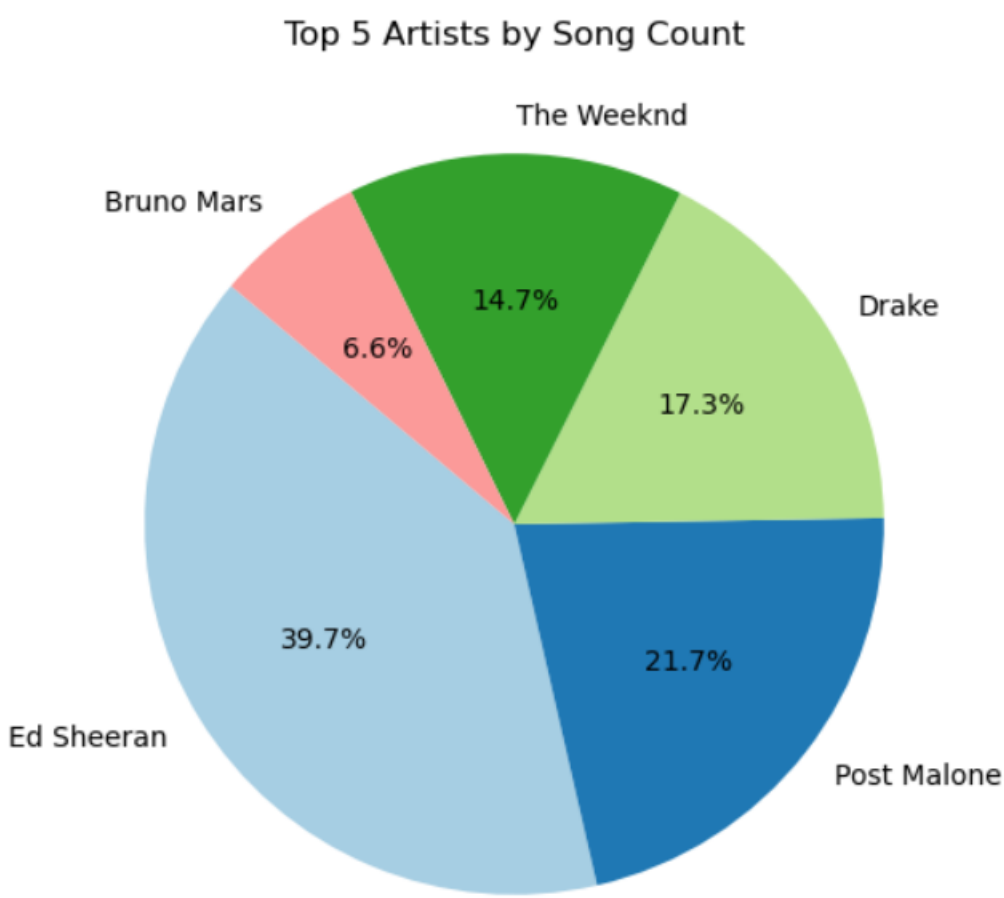
Data Visualization - 2

Top 5 Artists by Song Count

The code groups the DataFrame df by "Artist," counts the number of songs for each artist, selects the top 5 artists, and visualizes the distribution using a pie chart. The resulting chart provides a snapshot of the top artists based on song count in the dataset.

In [21]:

```
# Group by Artist and count the number of songs for each artist
artist_counts = df.groupBy("Artist").agg(count("*").alias("Count")).orderBy("Count", ascending=False)

# Select the top 5 artists
top_artists = artist_counts.limit(5)

# Plotting the pie chart for top artists
plt.figure(figsize=(10, 6))
plt.pie(top_artists.toPandas()["Count"], labels=top_artists.toPandas()["Artist"], autopct="%1.1f%%", startangle=140, colors=p
plt.title("Top 5 Artists by Song Count")
plt.show()
```



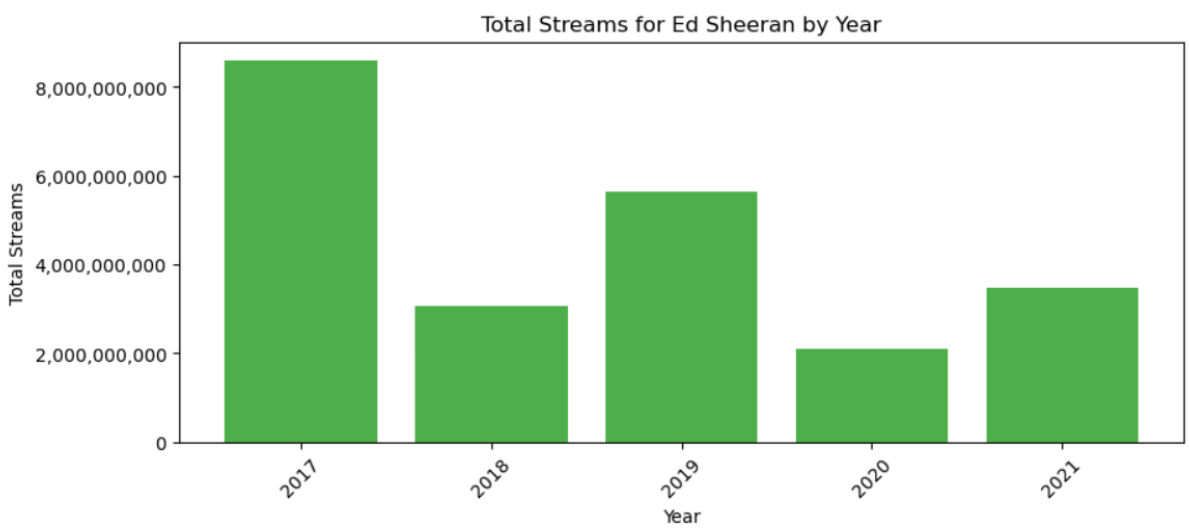
Data Visualization - 3

Total Streams for Ed Sheeran by Year

The code filters the DataFrame df to include only data for the artist 'Ed Sheeran'. It then groups the data by year, calculates the total streams for each year, and visualizes the results using a bar chart. This chart illustrates the yearly streaming patterns for the specified artist, providing insights into their popularity over time.

In [23]:

```
from matplotlib.ticker import StrMethodFormatter

specific_artist = 'Ed Sheeran'
artist_df = df.filter(col('Artist') == specific_artist)

# Grouping by 'Year' and summing the 'Stream' values
grouped_data = artist_df.groupBy('Year').agg({'Stream': 'sum'}).orderBy('Year')

# Convert PySpark DataFrame to Pandas for plotting
pandas_grouped_data = grouped_data.toPandas()

# Plotting the bar chart
plt.figure(figsize=(10, 4))
plt.bar(pandas_grouped_data['Year'], pandas_grouped_data['sum(Stream)'],color=plt.cm.Set1.colors[2])
plt.title(f'Total Streams for {specific_artist} by Year')
plt.xlabel('Year')
plt.ylabel('Total Streams')
plt.gca().get_yaxis().set_major_formatter(StrMethodFormatter('{x:,.0f}'))
plt.xticks(rotation=45)
plt.show()
```



Learning Outcomes:

Data Processing with Hive:

- Understand the basics of Hive and how to write queries in HQL.
- Learn about data types, tables, and joins in Hive.
- Practice using Hive to analyze and transform large datasets.

Distributed Computing with PySpark:

- Learn to use Spark for data processing, machine learning, and graph processing.
- Explore Spark's Python API (PySpark) for working with Spark using Python.

Data Visualization in PySpark:

- Master the basics of Matplotlib and Seaborn for static visualizations.
- Explore Plotly for interactive visualizations and dashboards.
- Practice creating a variety of charts and plots to effectively communicate data insights.

Conclusion:

This project has successfully explored and analyzed a music streaming dataset using PySpark. By examining trends over the years, identifying top artists, and specifically delving into Ed Sheeran's streaming patterns, we gained valuable insights into the dynamic landscape of music consumption. The visualizations provide a comprehensive understanding of music trends, artist popularity, and audience preferences, offering actionable insights for stakeholders in the music industry.