OLAP Query Execution Along with Snap Shots

- **Anupriya Sen(2022OG04013), Kumar Ashutosh(2022OG04019)**

We have used DataBrick Environment to connect with SparkSQL to execute queries focusing on Sales and Growth.

Configuration Update:

- spark.mongodb.input.uri=mongodb+srv://<username>:<password>@cluster0.m6hscve.mongodb.net/?retryWrites=true&w=majority/spotify.spotify?ssl=true
- "spark.mongodb.output.uri=mongodb+srv://<username>:<password>@cluster0.m6hscve.mongodb.net/?retryWrites=true&w=majority/spotify.spotify?ssl=true
- Database=spotify
- Collection=spotify

Git Hub URL :

Python Notebook HTML:

SparkSQL Python Notebook Executed · ashutosh-kumar10/BDS-Assignment@297059b (github.com)

Loom Video Link:

https://www.loom.com/share/1fda2cc6799b45588d29508dd6599060?sid=07675d91-d301-428a-a7e2-759e8978610f

Execution Snapshot :

Query 1: Find the average popularity, duration, and tempo of each track genre and sort them by popularity in descending order.

Cmd 8

Python

```
# Using Spark SQL
df.createOrReplaceTempView("tracks") # Register the DataFrame as a temporary view
spark.sql("""
SELECT track_genre, AVG(popularity) AS avg_popularity, AVG(duration_ms) AS avg_duration, AVG(tempo) AS avg_tempo
FROM tracks
GROUP BY track_genre
ORDER BY avg_popularity DESC
""").show()

# Using DataFrame API
df.groupBy("track_genre").agg(
  {"popularity": "avg", "duration_ms": "avg", "tempo": "avg"}
).withColumnRenamed("avg(popularity)", "avg_popularity").withColumnRenamed("avg(duration_ms)", "avg_duration").withColumnRenamed("avg(tempo)", "avg_tempo").orderBy("avg_popularity", ascending=False).show()
```

(4) Spark Jobs

```
+-----------+--------------+------------+---------+
|track_genre|avg_popularity|avg_duration|avg_tempo|
+-----------+--------------+------------+---------+
|   acoustic|          67.6|    198377.6| 108.6854|
+-----------+--------------+------------+---------+

+-----------+------------+--------------+---------+
|track_genre|avg_duration|avg_popularity|avg_tempo|
+-----------+------------+--------------+---------+
|   acoustic|    198377.6|          67.6| 108.6854|
+-----------+------------+--------------+---------+
```

Query 2: Top Artists with Highest Average Popularity:

Cmd 10

```
spark.sql("""SELECT artists, AVG(popularity) AS avg_popularity
FROM tracks
GROUP BY artists
ORDER BY avg_popularity DESC
LIMIT 10
""").show()
# Using DataFrame API
df.groupBy("artists", "track_genre").count().withColumnRenamed("count", "track_count").orderBy("track_count", ascending=False).limit(10).show()
```

▸ (4) Spark Jobs

```
+--------------------+--------------+
|             artists|avg_popularity|
+--------------------+--------------+
|    Chord Overstreet|          82.0|
|         Gen Hoshino|          73.0|
|        Kina Grannis|          71.0|
|Ingrid Michaelson...|          57.0|
|        Ben Woodward|          55.0|
+--------------------+--------------+

+--------------------+-----------+-----------+
|             artists|track_genre|track_count|
+--------------------+-----------+-----------+
|         Gen Hoshino|   acoustic|          1|
|        Ben Woodward|   acoustic|          1|
|Ingrid Michaelson...|   acoustic|          1|
|        Kina Grannis|   acoustic|          1|
|    Chord Overstreet|   acoustic|          1|
+--------------------+-----------+-----------+
```

Command took 5.59 seconds -- by anupriya1322@yahoo.com at 12/24/2023, 2:51:11 AM on BDA

Query 3: Top Artists with their famous track:

Cmd 12

```
# Using Spark SQL
spark.sql("""
SELECT artists, track_genre, COUNT(*) AS track_count
FROM tracks
GROUP BY artists, track_genre
ORDER BY track_count DESC
LIMIT 10
""").show()

# Using DataFrame API
df.groupBy("artists", "track_genre").count().withColumnRenamed("count", "track_count").orderBy("track_count", ascending=False).limit(10).show()
```

▸ (4) Spark Jobs

```
+--------------------+-----------+-----------+
|             artists|track_genre|track_count|
+--------------------+-----------+-----------+
|         Gen Hoshino|   acoustic|          1|
|        Ben Woodward|   acoustic|          1|
|Ingrid Michaelson...|   acoustic|          1|
|        Kina Grannis|   acoustic|          1|
|    Chord Overstreet|   acoustic|          1|
+--------------------+-----------+-----------+

+--------------------+-----------+-----------+
|             artists|track_genre|track_count|
+--------------------+-----------+-----------+
|         Gen Hoshino|   acoustic|          1|
|        Ben Woodward|   acoustic|          1|
|Ingrid Michaelson...|   acoustic|          1|
|        Kina Grannis|   acoustic|          1|
|    Chord Overstreet|   acoustic|          1|
+--------------------+-----------+-----------+
```

Query 4:Find the average danceability, energy, and valence of each track genre and plot them as a bar chart

Cmd 14

```
# Using Spark SQL
df_avg = spark.sql("""
SELECT track_genre, AVG(danceability) AS avg_danceability, AVG(energy) AS avg_energy, AVG(valence) AS avg_valence
FROM tracks
GROUP BY track_genre
""").toPandas() # Convert the Spark DataFrame to a Pandas DataFrame

# Using DataFrame API
df_avg = df.groupBy("track_genre").agg(
  {"danceability": "avg", "energy": "avg", "valence": "avg"}
).withColumnRenamed("avg(danceability)", "avg_danceability").withColumnRenamed("avg(energy)", "avg_energy").withColumnRenamed("avg(valence)", "avg_valence").toPandas() # Convert the Spark DataFrame to a Pandas DataFrame

display(df_avg)

# Plot the bar chart using Matplotlib
import matplotlib.pyplot as plt
df_avg.plot(x="track_genre", y=["avg_danceability", "avg_energy", "avg_valence"], kind="bar", figsize=(100, 6), title="Average Audio Features by Track Genre")
ax = plt.gca() # Get the current axes
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha="right") # Rotate and align the labels
plt.show()
```

▸ (4) Spark Jobs

Table +

	track_genre	avg_valence	avg_danceability	avg_energy
1	acoustic	0.2824	0.48360000000000003	0.29772000000000004

1 row | 5.77 seconds runtime

Command took 5.77 seconds -- by anupriya1322@yahoo.com at 12/24/2023, 2:51:50 AM on BDA

Query 5: Find the most common key and mode for each genre and plot them as a pie chart.

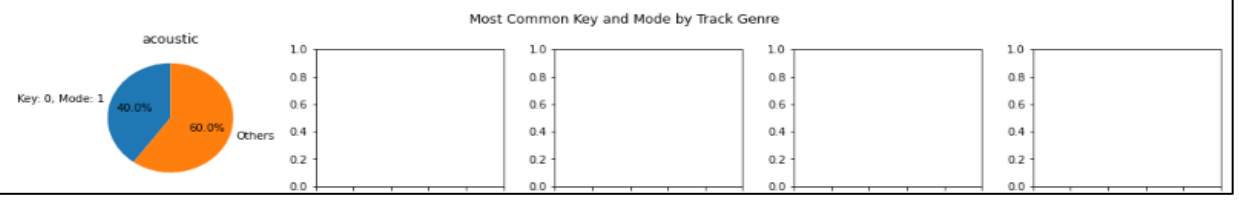
Cmd 16

```
# Using Spark SQL
df_key_mode = spark.sql("""
SELECT track_genre, key, mode, COUNT(*) AS count
FROM tracks
GROUP BY track_genre, key, mode
""").toPandas() # Convert the Spark DataFrame to a Pandas DataFrame

# Using DataFrame API
df_key_mode = df.groupBy("track_genre", "key", "mode").count().toPandas() # Convert the Spark DataFrame to a Pandas DataFrame

# Plot the pie chart using Matplotlib
import matplotlib.pyplot as plt
genres = df_key_mode["track_genre"].unique() # Get the unique genres
fig, axes = plt.subplots(4, 5, figsize=(15, 10)) # Create a figure with 4 rows and 5 columns of subplots
fig.suptitle("Most Common Key and Mode by Track Genre") # Set the figure title
for i, genre in enumerate(genres): # Loop through the genres
  df_genre = df_key_mode[df_key_mode["track_genre"] == genre] # Filter the data by genre
  max_count = df_genre["count"].max() # Get the maximum count
  df_max = df_genre[df_genre["count"] == max_count] # Get the row with the maximum count
  key = df_max["key"].iloc[0] # Get the key
  mode = df_max["mode"].iloc[0] # Get the mode
  labels = [f"Key: {key}, Mode: {mode}", "Others"] # Create the labels
  sizes = [max_count, df_genre["count"].sum() - max_count] # Create the sizes
  ax = axes[i // 5, i % 5] # Get the current subplot
  ax.pie(sizes, labels=labels, autopct="%1.1f%%", startangle=90) # Plot the pie chart
  ax.set_title(genre) # Set the subplot title
plt.tight_layout() # Adjust the layout
plt.show()
```

▸ (4) Spark Jobs



Recommendations to boost Sales/growth as future capability area –

These OLAP queries focus on different aspects of the dataset to gain insights into the factors influencing popularity and engagement. They cover features like track popularity, genre analysis, valence, energy, explicit content, duration, artist popularity, danceability, acoustic characteristics, temporal trends, instrumentalness, and key distribution.

Analyzing these aspects can guide decisions for better curation, marketing, and strategic planning to drive future sales and growth for Spotify as shared below -

Subscriber Base Expansion: Spotify's growth is closely tied to its ability to attract and retain subscribers. The company may focus on expanding its user base through targeted marketing, partnerships, and international expansion.

Content Strategy: The availability of exclusive and high-quality content, such as podcasts and music, can be a significant driver for subscriber acquisition and retention. Spotify may continue to invest in content creation and licensing agreements to differentiate itself from competitors.

Technological Innovations: Advances in technology, such as improved recommendation algorithms, personalized playlists, and better user interfaces, can enhance the user experience and drive user engagement. Spotify's ability to leverage emerging technologies may impact its sales growth.

Partnerships and Collaborations: Strategic partnerships with device manufacturers, telecom companies, and other platforms can help Spotify reach new audiences and increase its market share.

Monetization Diversification: Spotify may explore new revenue streams beyond subscription fees, such as advertising, exclusive content deals, or innovative features. Diversifying revenue sources can contribute to overall sales growth.

Global Market Trends: Changes in consumer behavior, preferences, and global market trends can impact Spotify's growth. Staying adaptable to shifts in the industry landscape is crucial for sustained success.

The End