databricks OLAP Analytics Queries Using SparkSQL

(https://databricks.com)

## Importing necessary libraries

```
import warnings
import pandas as pd
import numpy as np
import seaborn as sns

from pyspark.sql import SparkSession
# Import SparkSession and types
from pyspark.sql import SparkSession
from pyspark.sql.types import *

# Plot the bar chart using Matplotlib
import matplotlib.pyplot as plt
```

## Build Spark Session and connect to MongoDb

```
from pyspark.sql import SparkSession  spark = SparkSession.builder \     .appNam ...
```

Show cell

## Mounting ADLS Gen2 Storage Account

```
# Check if mount point exists and mount if not exist
mount_point = '/mnt/input'

if not any(mount_point in mp for mp in [mp.mountPoint for mp in dbutils.fs.mounts()]):
  STORAGE_ACCOUNT_NAME = 'sabds1'
  STORAGE_CONTAINER_NAME = 'aiml'
  STORAGE_ACCOUNT_ACCESS_KEY = 'gdaaI567mbPzS+cZgkuvH5gvmo3N8WVUTVLbX5WJXIVuAzXqkCCgkpUHYAX5DWOI5KIjefQsHSt++ASt8iuTYA=='

  dbutils.fs.mount(
    source=f'wasbs://{STORAGE_CONTAINER_NAME}@{STORAGE_ACCOUNT_NAME}.blob.core.windows.net',
    mount_point=mount_point,
    extra_configs={
      # 'fs.azure.account.key.{STORAGE_ACCOUNT_NAME}.blob.core.windows.net': STORAGE_ACCOUNT_ACCESS_KEY
      'fs.azure.account.key.{0}.blob.core.windows.net'.format(STORAGE_ACCOUNT_NAME): STORAGE_ACCOUNT_ACCESS_KEY
    }
  )
```

## check avaialable mount points

```
display(dbutils.fs.mounts())
```

Table

| | mountPoint | source | encryptionType |
|---|---|---|---|
| 1 | /databricks-datasets | databricks-datasets | |

| | | | |
|---|---|---|---|
| 2 | /Volumes | UnityCatalogVolumes | |
| 3 | /databricks/mlflow-tracking | databricks/mlflow-tracking | |
| 4 | /databricks-results | databricks-results | |
| 5 | /databricks/mlflow-registry | databricks/mlflow-registry | |
| 6 | /mnt/input | wasbs://aiml@sabds1.blob.core.windows.net | |
| 7 | /Volume | DbfsReserved | |

10 rows

```
# copy data from mountPoint to /tmp
dbutils.fs.cp("dbfs:/mnt/input/train.csv", "/tmp/train.csv")
```

```
True
```

## Load the Spotify track genre dataset into a Spark DataFrame.

```
# Create a SparkSession
spark = SparkSession.builder.appName("Spotify Data Analysis").getOrCreate()

# Define the schema for the DataFrame
# schema = StructType([
#   StructField("Unnamed: 0", IntegerType(), True),
#   StructField("track_id", StringType(), True),
#   StructField("artists", StringType(), True),
#   StructField("album_name", StringType(), True),
#   StructField("track_name", StringType(), True),
#   StructField("popularity", IntegerType(), True),
#   StructField("duration_ms", IntegerType(), True),
#   StructField("explicit", BooleanType(), True),
#   StructField("danceability", DoubleType(), True),
#   StructField("energy", DoubleType(), True),
#   StructField("key", IntegerType(), True),
#   StructField("loudness", DoubleType(), True),
#   StructField("mode", IntegerType(), True),
#   StructField("speechiness", DoubleType(), True),
#   StructField("acousticness", DoubleType(), True),
#   StructField("instrumentalness", DoubleType(), True),
#   StructField("liveness", DoubleType(), True),
#   StructField("valence", DoubleType(), True),
#   StructField("tempo", DoubleType(), True),
#   StructField("time_signature", IntegerType(), True),
#   StructField("track_genre", StringType(), True)
# ])

# Read the CSV file and create a DataFrame
# df = spark.read.csv("/tmp/train.csv", header=True,encoding='utf-8')
# handling some special characters and unstructered data while reading the CSV file
df = spark.read.format('csv').option('header',True).option('multiLine',
True).option('quote','"').option('escape','"').load('/tmp/train.csv')

display(df)
```

Table

| | Unnamed: 0 | track_id | artists |
|---|---|---|---|
| 1 | 0 | 5SuOikwiRyPMVoIQDJUgSV | Gen Hoshino |
| 2 | 1 | 4qPNDBW1i3p13qLCt0Ki3A | Ben Woodward |
| 3 | 2 | 1iJBSr7s7jYXzM8EGcbK5b | Ingrid Michaelson;ZAYN |
| 4 | 3 | 6lfxq3CG4xtTiEg7opyCyx | Kina Grannis |
| 5 | 4 | 5vjLSffimiIP26QG5WcN2K | Chord Overstreet |

| | | | |
|---|---|---|---|
| 6 | 5 | 01MVOl9KtVTNfFiBU9I7dc | Tyrone Wells |
| 7 | 6 | 6Vc5wAMmXdKIAM7WUoEb7N | A Great Big World;Christina Aguilera |

9,568 rows | Truncated data

```
print(df.select('track_genre').distinct().count())
df.select('track_genre').distinct().show(200)
```

```
114
+-----------------+
|      track_genre|
+-----------------+
|            anime|
|singer-songwriter|
|             folk|
|        hardstyle|
|              pop|
|      alternative|
|      death-metal|
|   detroit-techno|
|              idm|
|            k-pop|
|          j-dance|
|          ambient|
|           guitar|
|             goth|
|         cantopop|
|            blues|
|            study|
```

## Perform some OLAP Analytics queries on the DataFrame using Spark SQL or the DataFrame API

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

```
# 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()
```

```
+-----------------+--------------+------------+------------------+
|      track_genre|avg_popularity|avg_duration|         avg_tempo|
+-----------------+--------------+------------+------------------+
|         pop-film|        59.283|  279657.084|117.25747799999994|
|            k-pop|        56.896|  251277.169|119.24396899999998|
|            chill|        53.651|  169009.967|115.47937600000013|
|              sad|        52.379|   153800.88|119.06494999999995|
|           grunge|        49.594|  235579.061|129.34920799999992|
|           indian|        49.539|  245473.096|116.14295399999986|
|            anime|        48.772|  210204.076|123.52961600000015|
|              emo|        48.128|   189690.33| 126.9926430000002|
|        sertanejo|        47.866|  204583.551|127.05219699999991|
```

```
|             pop|        47.576|  220672.776|120.92707299999994|
|progressive-house|        46.615|  206842.186|125.09363600000009|
|           piano|        45.273|  203966.541|118.08467400000006|
|        mandopop|        45.025|   251550.05|123.47590599999992|
|       deep-house|        44.808|  219344.579|120.87490400000013|
|          brazil|         44.67|  274230.482|121.93564399999985|
|       electronic|        44.325|  233013.326|122.94783900000006|
|          pagode|        44.298|  228321.435|126.85930799999979|
|         ambient|        44.191|  237059.038|111.11312900000003|
```

## Query 2: Find the top 10 artists with the most tracks in the dataset and their genres.

```
# 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()
```

```
+-----------------+-----------+-----------+
|          artists|track_genre|track_count|
+-----------------+-----------+-----------+
|     George Jones| honky-tonk|        271|
|my little airport|   cantopop|        171|
|      The Beatles| psych-rock|        149|
|              BTS|      k-pop|        143|
|    Hank Williams| honky-tonk|        140|
|  Håkan Hellström|       goth|        139|
|        Glee Cast|       club|        139|
|      Linkin Park|     grunge|        131|
|          Scooter|      happy|        129|
|      The Beatles|    british|        127|
+-----------------+-----------+-----------+
```


```
+-----------------+-----------+-----------+
|          artists|track_genre|track_count|
+-----------------+-----------+-----------+
|     George Jones| honky-tonk|        271|
|my little airport|   cantopop|        171|
|      The Beatles| psych-rock|        149|
```


## Query 3: Find the average danceability, energy, and valence of each track genre and plot them as a bar chart.

```
from pyspark.sql.functions import col

# 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

N=20
df_avg_topN = df_avg.sort_values("avg_danceability", ascending=False).head(N)

display(df_avg_topN)

df_avg_lowestN_danceability = df_avg.sort_values("avg_danceability", ascending=True).head(N)
df_avg_lowestN_energy = df_avg.sort_values("avg_energy", ascending=True).head(N)
df_avg_lowestN_valence = df_avg.sort_values("avg_valence", ascending=True).head(N)

display(df_avg_lowestN_danceability)
display(df_avg_lowestN_energy)
display(df_avg_lowestN_valence)

# Add a new column 'avg_features' that is the average of the three features
df = df.withColumn('avg_features', (col('danceability') + col('energy') + col('valence')) / 3)

# Find the genre with the lowest average feature value
lowest_avg_genre = df.groupBy('track_genre').agg(F.avg('avg_features').alias('avg_features')).orderBy('avg_features').first()

print("The genre with the lowest average feature value is: ", lowest_avg_genre['track_genre'])


# Plot the bar chart using Matplotlib
import matplotlib.pyplot as plt
df_avg_topN.plot(x="track_genre", y=["avg_danceability", "avg_energy", "avg_valence"], kind="bar", figsize=(10, 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()
```

**Table**

| | track_genre | avg_valence | avg_danceability | avg_energy |
|---|---|---|---|---|
| 1 | kids | 0.6808638000000004 | 0.7789059999999993 | 0.6131285999999997 |
| 2 | chicago-house | 0.5865411999999995 | 0.7661760000000005 | 0.7332149999999983 |
| 3 | reggaeton | 0.642753499999999 | 0.7585209999999992 | 0.7387279999999978 |
| 4 | latino | 0.6302008 | 0.7570569999999999 | 0.7317964999999989 |
| 5 | reggae | 0.6475289999999999 | 0.7453309999999995 | 0.7257909999999984 |
| 6 | hip-hop | 0.5512478000000002 | 0.7361539999999978 | 0.682530000000002 |
| 7 | dancehall | 0.6290086000000006 | 0.7341689999999997 | 0.6852619999999996 |

20 rows

**Table**

| | track_genre | avg_valence | avg_danceability | avg_energy |
|---|---|---|---|---|
| 1 | sleep | 0.058187871000000196 | 0.1679225000000003 | 0.3420717167000004 |
| 2 | grindcore | 0.21643359999999978 | 0.2718537000000001 | 0.9242010000000016 |
| 3 | black-metal | 0.19173639999999972 | 0.2964108999999997 | 0.8748973000000019 |

| | | | | |
|---|---|---|---|---|
| 4 | iranian | 0.15353640000000002 | 0.3006860000000008 | 0.5458459026000002 |
| 5 | opera | 0.2152252000000001 | 0.31356309999999993 | 0.31705399999999995 |
| 6 | new-age | 0.18316719999999986 | 0.3484546000000001 | 0.21450061999999998 |
| 7 | ambient | 0.16749819999999987 | 0.36786679999999977 | 0.2371617900000003 |

20 rows

Table

| | track_genre | avg_valence | avg_danceability | avg_energy |
|---|---|---|---|---|
| 1 | classical | 0.38104999999999994 | 0.38192280000000006 | 0.18982732600000018 |
| 2 | new-age | 0.18316719999999986 | 0.3484546000000001 | 0.21450061999999998 |
| 3 | ambient | 0.16749819999999987 | 0.36786679999999977 | 0.2371617900000003 |
| 4 | romance | 0.3934146999999995 | 0.4321332000000002 | 0.2943038000000002 |
| 5 | disney | 0.36855740000000037 | 0.4628741999999997 | 0.30251885999999956 |
| 6 | opera | 0.2152252000000001 | 0.31356309999999993 | 0.31705399999999995 |
| 7 | piano | 0.3132662 | 0.45509829999999896 | 0.32010256399999965 |

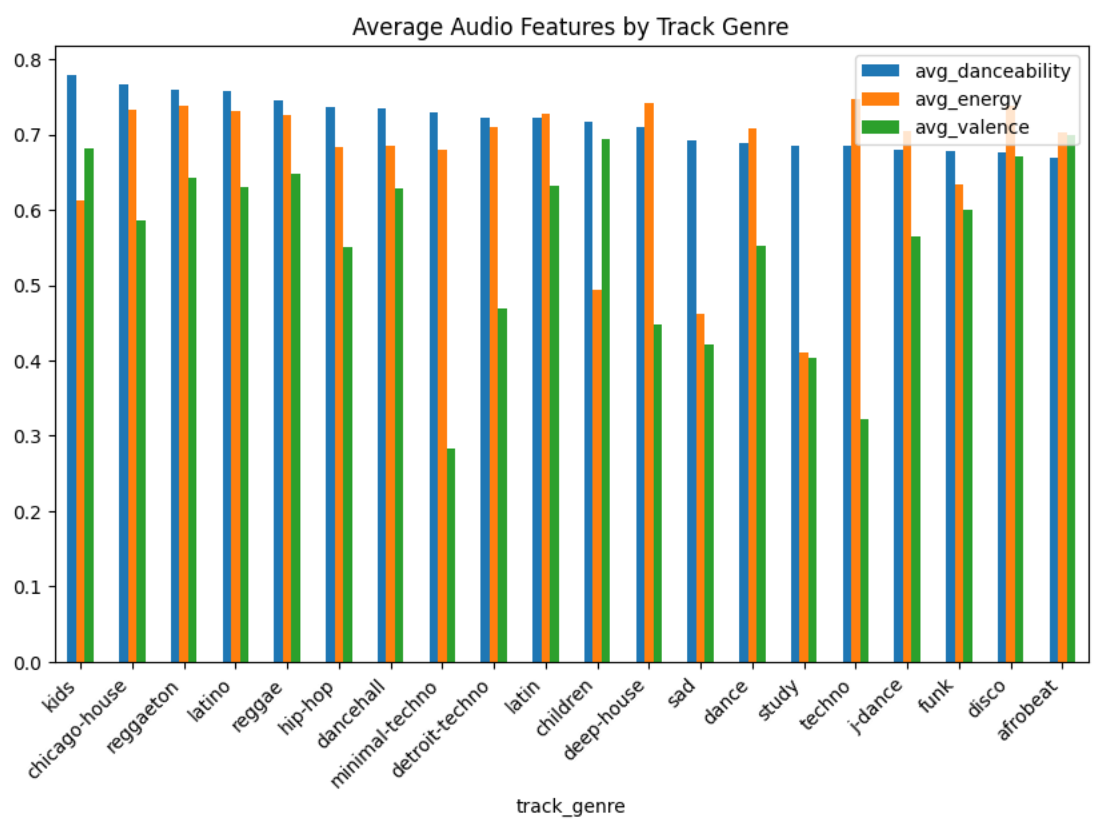
20 rows

Table

| | track_genre | avg_valence | avg_danceability | avg_energy |
|---|---|---|---|---|
| 1 | sleep | 0.058187871000000196 | 0.1679225000000003 | 0.3420717167000004 |
| 2 | iranian | 0.15353640000000002 | 0.3006860000000008 | 0.5458459026000002 |
| 3 | ambient | 0.16749819999999987 | 0.36786679999999977 | 0.2371617900000003 |
| 4 | new-age | 0.18316719999999986 | 0.3484546000000001 | 0.21450061999999998 |
| 5 | black-metal | 0.19173639999999972 | 0.2964108999999997 | 0.8748973000000019 |
| 6 | opera | 0.2152252000000001 | 0.31356309999999993 | 0.31705399999999995 |
| 7 | grindcore | 0.21643359999999978 | 0.2718537000000001 | 0.9242010000000016 |

20 rows

```
The genre with the lowest average feature value is:  sleep
```




**Some observations from the bar chart are:**

Some observations from the bar chart are:

- Kids has the highest average danceability, followed by Chicago-house and reggaeton.
- death-metal has the highest average energy, followed by grindcore and metalcore.
- salsa has the highest average valence, followed by forro and rockabilly.
- sleep has the lowest average values for all three features, followed by classical.

### Query 4: Find the average popularity of each track genre and plot them as a bar chart.

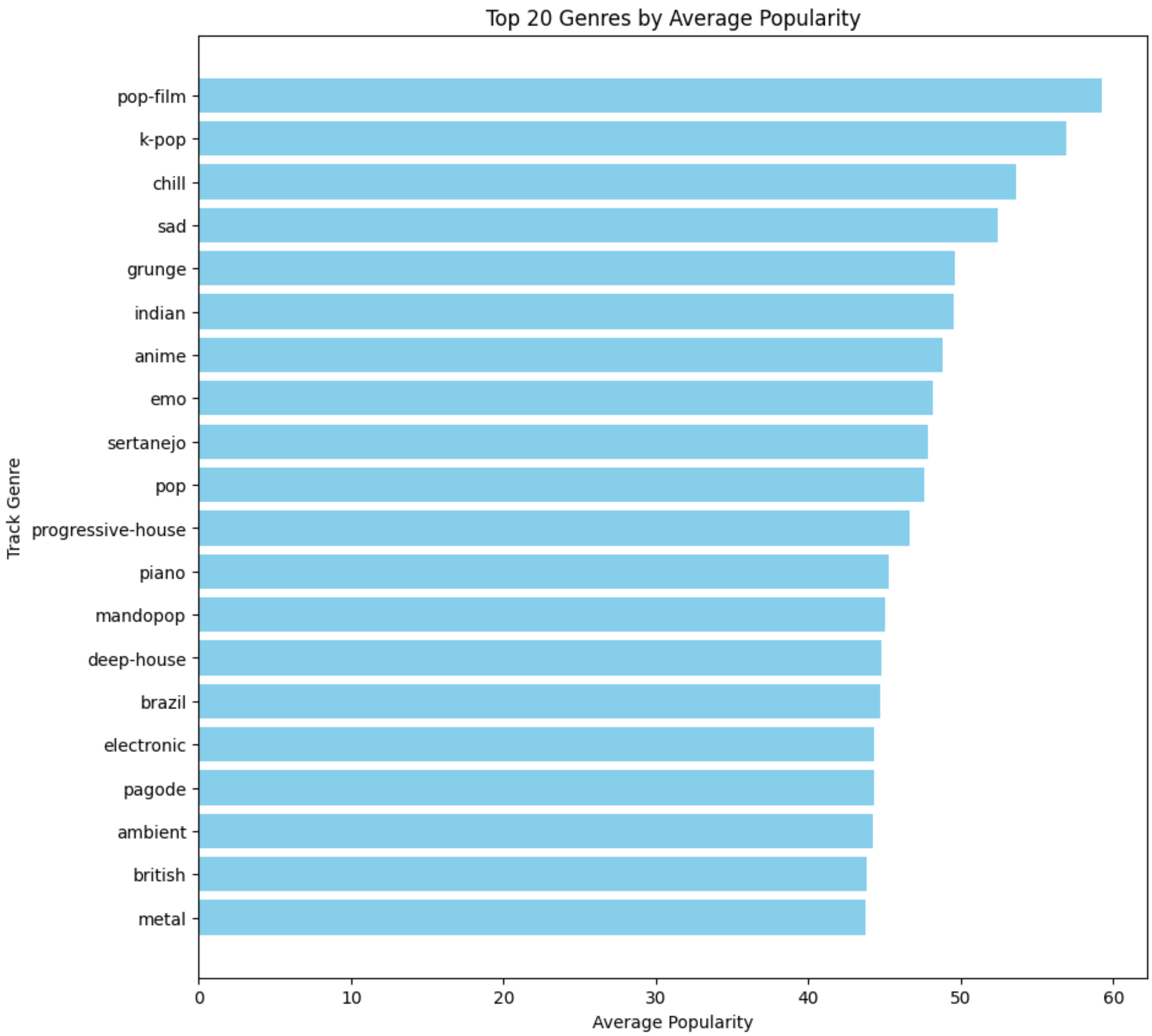
```
from pyspark.sql import functions as F
df.groupBy('track_genre').agg(F.avg('popularity').alias('avg_popularity')).show()

avg_track_popularity = df.groupBy('track_genre').agg(F.avg('popularity').alias('avg_popularity')).orderBy('avg_popularity',
ascending=False)

# Convert the result to Pandas DataFrame
avg_track_popularity_pd = avg_track_popularity.toPandas()
```

```
+-----------------+--------------+
|      track_genre|avg_popularity|
+-----------------+--------------+
|            anime|        48.772|
|singer-songwriter|        37.813|
|             folk|        38.006|
|        hardstyle|        26.623|
|              pop|        47.576|
|      alternative|        24.337|
|      death-metal|        32.169|
|   detroit-techno|        11.174|
|              idm|        15.766|
|            k-pop|        56.896|
|          j-dance|        26.656|
|          ambient|        44.191|
|           guitar|        29.526|
|             goth|        28.913|
|         cantopop|        34.739|
|            blues|        31.188|
|            study|        26.108|
|            malay|        30.358|
```

```
N = 20  # Replace with your desired number
top_N = avg_track_popularity_pd.sort_values('avg_popularity', ascending=False).head(N)
plt.figure(figsize=(10,10))  # Adjust the size as needed
plt.barh(top_N['track_genre'], top_N['avg_popularity'], color='skyblue')
plt.xlabel('Average Popularity')
plt.ylabel('Track Genre')
plt.title('Top {} Genres by Average Popularity'.format(N))
plt.gca().invert_yaxis()  # This will show the genre with highest popularity at the top
plt.show()
```




**Query 5: Find the maximum and minimum duration of tracks for each genre:**

```
df.groupBy('track_genre').agg(F.max('duration_ms').alias('max_duration'), F.min('duration_ms').alias('min_duration')).show()
```

```
+-----------------+------------+------------+
|      track_genre|max_duration|min_duration|
+-----------------+------------+------------+
|            anime|       99079|      100506|
|singer-songwriter|       99752|      102986|
|             folk|       99752|      100000|
|        hardstyle|       95163|      105711|
|              pop|       99583|      102315|
|      alternative|       90960|      109714|
|      death-metal|       98826|      101266|
|   detroit-techno|      960000|      100009|
|              idm|       97133|      100824|
|            k-pop|      946552|           0|
|          j-dance|       99713|      102773|
|          ambient|       99610|      100013|
|           guitar|       99893|     1013410|
|             goth|       83707|      100500|
```

```
|         cantopop|       98495|      102293|
|            blues|       95480|      110093|
|            study|       99317|      100000|
```

### Query 6: Find the number of explicit tracks for each genre:

```
df.filter(df['explicit'] == True).groupBy('track_genre').count().orderBy('count', ascending=False).show()
```

```
+-----------+-----+
|track_genre|count|
+-----------+-----+
|     comedy|  656|
|        emo|  465|
|        sad|  450|
|    j-dance|  391|
|   hardcore|  325|
|    hip-hop|  319|
|       funk|  304|
|  dancehall|  302|
|  metalcore|  291|
|death-metal|  251|
|     latino|  249|
| industrial|  236|
|     french|  219|
|    turkish|  218|
|  reggaeton|  212|
|      dance|  174|
|      chill|  171|
|     reggae|  167|
```

### Query 7: Find the top 10 most popular tracks:

```
df.orderBy(df['popularity'].desc()).select('track_name', 'popularity').show(10)
```

```
+--------------------+----------+
|          track_name|popularity|
+--------------------+----------+
|Quevedo: Bzrp Mus...|        99|
|          La Bachata|        98|
|          La Bachata|        98|
|     I'm Good (Blue)|        98|
|     I'm Good (Blue)|        98|
|          La Bachata|        98|
|     I'm Good (Blue)|        98|
|          La Bachata|        98|
|    Tití Me Preguntó|        97|
|     Me Porto Bonito|        97|
+--------------------+----------+
only showing top 10 rows
```

### Query 8: Find the artist with the most tracks:

```
df.groupBy('artists').count().orderBy(desc('count')).show(1)
```

```
+-----------+-----+
|    artists|count|
+-----------+-----+
|The Beatles|  279|
```

```
+-----------+-----+
only showing top 1 row
```

### Quert 9: Find the album with the most tracks:

```
df.groupBy('album_name').count().orderBy(desc('count')).show(1)
```

```
+--------------------+-----+
|          album_name|count|
+--------------------+-----+
|Alternative Chris...|  195|
+--------------------+-----+
only showing top 1 row
```

### Query 10: Find the most common words in track names:

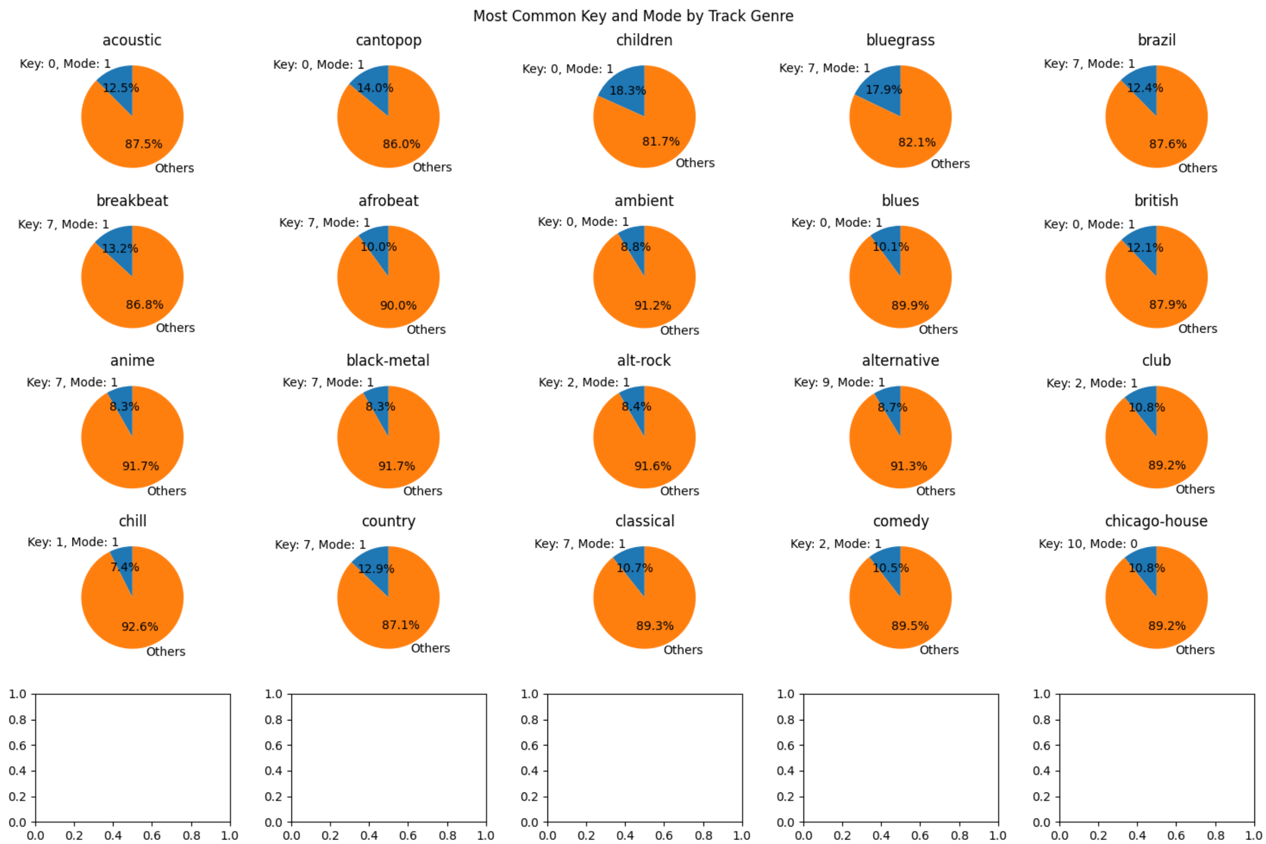
```
from pyspark.sql.functions import split, explode

# Split the track_name into words
df_words = df.select(explode(split(df['track_name'], ' ')).alias('word'))

# Count the occurrence of each word and show the most common words
df_words.groupBy('word').count().orderBy(desc('count')).show()
```

```
+---------+-----+
|     word|count|
+---------+-----+
|        -|16160|
|      The| 6247|
|      You| 3948|
|       Me| 3422|
|        I| 3022|
|      the| 2935|
|     Vivo| 2691|
|       Ao| 2610|
|    Remix| 2601|
|   (feat.| 2374|
|     Love| 2238|
|        A| 2166|
|       of| 2120|
|       My| 1909|
|Christmas| 1838|
|        /| 1741|
|       de| 1714|
|       It| 1617|
```

### Query 11: Find the most common key and mode for each genre and plot them as a pie chart.


Most Common Key and Mode by Track Genre
acoustic
Key: 0, Mode: 1
12.5%
87.5%
Others
cantopop
Key: 0, Mode: 1
14.0%
86.0%
Others
children
Key: 0, Mode: 1
18.3%
81.7%
Others
bluegrass
Key: 7, Mode: 1
17.9%
82.1%
Others
brazil
Key: 7, Mode: 1
12.4%
87.6%
Others
breakbeat
Key: 7, Mode: 1
13.2%
86.8%
Others
afrobeat
Key: 7, Mode: 1
10.0%
90.0%
Others
ambient
Key: 0, Mode: 1
8.8%
91.2%
Others
blues
Key: 0, Mode: 1
10.1%
89.9%
Others
british
Key: 0, Mode: 1
12.1%
87.9%
Others
anime
Key: 7, Mode: 1
8.3%
91.7%
Others
black-metal
Key: 7, Mode: 1
8.3%
91.7%
Others
alt-rock
Key: 2, Mode: 1
8.4%
91.6%
Others
alternative
Key: 9, Mode: 1
8.7%
91.3%
Others
club
Key: 2, Mode: 1
10.8%
89.2%
Others
chill
Key: 1, Mode: 1
7.4%
92.6%
Others
country
Key: 7, Mode: 1
12.9%
87.1%
Others
classical
Key: 7, Mode: 1
10.7%
89.3%
Others
comedy
Key: 2, Mode: 1
10.5%
89.5%
Others
chicago-house
Key: 10, Mode: 0
10.8%
89.2%
Others
0.0
0.2
0.4
0.6
0.8
1.0