(https://databricks.com)

OLAP - Analysis of Spotify User Recomendation Data to improve user experience and Organizational Profitability

Creating park session with MongoDB connection

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("MongoDB Spark Connector") \
    .config("spark.mongodb.input.uri", "mongodb+srv://myAtlasDBUser:myatlas-001@myatlasclusteredu.xliqzij.mongodb.net/?retryWrite
    Music_Recomendation.user_songs_recomended") \
    .config("spark.mongodb.output.uri", "mongodb+srv://myAtlasDBUser:myatlas-001@myatlasclusteredu.xliqzij.mongodb.net/?retryWrit
    Music_Recomendation.user_songs_recomended") \
    .getOrCreate()
```

Saving the DB in Spark data Frame

```
df = spark.read.format("mongo") \
    .option("uri", "mongodb+srv://myAtlasDBUser:myatlas-001@myatlasclusteredu.xliqzij.mongodb.net/Music_Recomendation.user_songs_
    w=majority") \
    .load()

# Limit the display to only two records
df = df.limit(5)
display(df)
```

Table

	_id	acousticness	album_name	artists	danceability	duration_ms
1	{"oid": "65889ad5eec457234caa65f1"}	0.55	The Live Debut - 1990	Mariah Carey	0.548	181688
2	{"oid": "65889ad7eec457234caa65f2"}	0.876	Italian Love Songs	Dean Martin	0.106	158813
3	{"oid": "65889ad9eec457234caa65f3"}	0.203	Private Collection	Cliff Richard	0.67	204266
4	{"oid": "6588b7ffd27eacf4519e1f60"}	0.000167	Curb	Nickelback	0.437	240706
5	{"oid": "6588b803d27eacf4519e1f61"}	0.88	Affirmation	Savage Garden	0.55	230200

5 rows

Creating Temp View of Table to execute Spark SQL Queries

```
df.createOrReplaceTempView("songData")
```

1. Analyzing Popularity Distribution:

```
popularity_distribution = spark.sql("SELECT popularity, COUNT(*) AS count FROM songData GROUP BY popularity ORDER BY popularity")
popularity_distribution.show()
```

```
+----------+-----+
|popularity|count|
+----------+-----+
|        15|    1|
|        24|    1|
|        31|    1|
|        35|    1|
|        39|    1|
```

```
+----------+-----+
```

2. Average Valence by Genre:

```
from pyspark.sql.functions import explode
valence_by_genre = (
    df.select(explode("track_genre").alias("genre"), "valence")
    .groupBy("genre")
    .agg({"valence": "avg"})
    .withColumnRenamed("avg(valence)", "avg_valence")
)

valence_by_genre.show()
```

```
+------------------+-----------+
|             genre|avg_valence|
+------------------+-----------+
|         dance pop|     0.5835|
|               pop|      0.732|
|urban contemporary|      0.732|
|   adult standards|     0.4785|
|    easy listening|      0.114|
|            lounge|      0.114|
|        vocal jazz|      0.114|
|     rock-and-roll|      0.843|
| alternative metal|      0.589|
|     canadian rock|      0.589|
|       post-grunge|      0.589|
|          boy band|      0.435|
|          pop rock|      0.435|
+------------------+-----------+
```

3. Top Tracks by Energy:

```
# Query to find top tracks based on energy
top_tracks_by_energy = spark.sql("SELECT track_name, energy FROM songData ORDER BY energy DESC LIMIT 10")
top_tracks_by_energy.show()
```

```
+--------------------+------+
|          track_name|energy|
+--------------------+------+
|              Pusher| 0.892|
|Don't Play That Song| 0.889|
|    A Little In Love| 0.589|
|I Don't Know You ...| 0.195|
|       Hear My Heart| 0.179|
+--------------------+------+
```

4. Average Popularity Of Artists

```
top_artists_query = spark.sql("SELECT artists, AVG(popularity) AS avg_popularity FROM songData GROUP BY artists ORDER BY avg_popu
top_artists_query.show()
```

```
+-------------+--------------+
|      artists|avg_popularity|
+-------------+--------------+
|Savage Garden|          39.0|
|  Dean Martin|          35.0|
|   Nickelback|          31.0|
| Mariah Carey|          24.0|
|Cliff Richard|          15.0|
```

```
+-------------+--------------+
```

5.Track and Genre Analysis by Avg Dancebility and Avg Energy

```
genre_analysis_query = spark.sql("SELECT track_name, track_genre AS Genre, AVG(danceability) AS avg_danceability, AVG(energy) AS
GROUP BY track_genre,track_name ORDER BY avg_danceability DESC ,avg_energy DESC")
genre_analysis_query.show()
```

```
+--------------------+--------------------+----------------+----------+
|          track_name|               Genre|avg_danceability|avg_energy|
+--------------------+--------------------+----------------+----------+
|    A Little In Love|[adult standards,...|            0.67|     0.589|
|I Don't Know You ...|[boy band, dance ...|            0.55|     0.195|
|Don't Play That Song|[dance pop, pop, ...|           0.548|     0.889|
|              Pusher|[alternative meta...|           0.437|     0.892|
|       Hear My Heart|[adult standards,...|           0.106|     0.179|
+--------------------+--------------------+----------------+----------+
```

Likewise we can Execute multiple Spark SQL queries for Batch Analytics which will help us to improve user experience and Organizational Profits