databricks BDS assignment - part 3 (Spark SQL and Map reduce from Databricks)

(https://databricks.com)

Query 1 - Getting average danceability of each album.

```
%python
from pyspark.sql import Window, DataFrame, SparkSession
import sys
from pyspark.sql import functions as f
from pyspark.sql.types import StringType, IntegerType

#Reading csv as a Pyspark dataframe
df = spark.read.options(header=True,).csv("dbfs:/FileStore/tables/users/pgoyal19/train.csv")

#Defining the window partition for group by on the basis album_name column
w = Window.partitionBy("album_name").rowsBetween(-sys.maxsize, sys.maxsize)

# Adding the dancibility for every album
rule_filtered = df.withColumn("dancibility_of_album",f.sum("danceability").over(w)).select("album_name", "dancibility_of_album").dropDuplicates().sort(f.col("dancibility_of_album").des

# Getting the count (Number of songs) for each album
rule_filtered1 = df.withColumn("no_of_songs_in_album",f.count("album_name").over(w)).select("album_name", "no_of_songs_in_album").dropDuplicates().sort(f.col("no_of_songs_in_album").de

#Joining dataframe1 (Having danceability for each album) and dataframe2 (Having count of songs in each album) and calculating the average dancibility.
final_output = rule_filtered.join(rule_filtered1, "album_name", "inner")
final_output=final_output.withColumn("average_danceability_of_any_album", final_output["dancibility_of_album"]/final_output["no_of_songs_in_album"]).filter(f.col("average_danceability_
isNotNull()).filter(f.col("average_danceability_of_any_album") < 1).sort(f.col("average_danceability_of_any_album").desc()).select("album_name", "average_danceability_of_any_album")
final_output.show(10, truncate=0)
```

```
+-----------------------------------+-------------------------------+
|album_name                         |average_danceability_of_any_album|
+-----------------------------------+-------------------------------+
|The Best of Quantic                |0.985                          |
|Fields Without Fences              |0.982                          |
|Dance Mania: Ghetto Madness        |0.981                          |
|Dancing in My Room                 |0.98                           |
|Mono:Disko, Vol. 12                |0.979                          |
|The Slide Album                    |0.978                          |
|Mentally Free                      |0.975                          |
|Jamie 3:26 Presents a Taste of Chicago|0.974                          |
|Electric Slide Dance Party         |0.974                          |
|Saci (Remix)                       |0.974                          |
+-----------------------------------+-------------------------------+
only showing top 10 rows
```

Query 2 - Getting top 8 popular albums in every genre on the basis of average popularity.

```
%python
from pyspark.sql import Window, DataFrame, SparkSession
import sys
from pyspark.sql import functions as f
from pyspark.sql.types import StringType, IntegerType

#Reading csv as a Pyspark dataframe
df = spark.read.options(header=True,).csv("dbfs:/FileStore/tables/users/pgoyal19/train.csv")

#Defining the window partition for group by on the basis track_genre, album_name column
w = Window.partitionBy("track_genre", "album_name").rowsBetween(-sys.maxsize, sys.maxsize)

# Adding the popularity for every track_genre, album_name
rule_filtered = df.withColumn("popularity_of_album",f.sum("popularity").over(w)).select("track_genre", "album_name",
"popularity_of_album").dropDuplicates().sort(f.col("popularity_of_album").desc())

# Getting the count (Number of songs) for each album
rule_filtered1 = df.withColumn("no_of_songs_in_album",f.count("album_name").over(w)).select("track_genre", "album_name",
"no_of_songs_in_album").dropDuplicates().sort(f.col("no_of_songs_in_album").desc())

#Joining dataframe1 (Having popularity for each album, genre) and dataframe2 (Having count of songs in each album) and calculating the average popularity.
final_output = rule_filtered.join(rule_filtered1, ["track_genre", "album_name"], "inner")
final_output=final_output.withColumn("average_popularity_of_any_album",
final_output["popularity_of_album"]/final_output["no_of_songs_in_album"]).filter(f.col("average_popularity_of_any_album").isNotNull()).sort(f.col("average_popularity_of_any_album
").desc())

#Defining window function to get the first 8 rows for each track_genre
w1 = Window.partitionBy("track_genre").orderBy(f.col("track_genre"), f.col("average_popularity_of_any_album").desc())
final_putput_first_ten_rows = final_output.withColumn("ln", f.row_number().over(w1)) \
    .filter(f.col("ln") <=8).select("track_genre", "album_name", "average_popularity_of_any_album")
final_putput_first_ten_rows.show(10, truncate=0)
```

```
+-----------+-----------------------------------------------------+-------------------------------+
|track_genre|album_name                                           |average_popularity_of_any_album|
+-----------+-----------------------------------------------------+-------------------------------+
|acoustic   |Pano                                                 |75.0                           |
|acoustic   |We Sing. We Dance. We Steal Things.                  |74.25                          |
|acoustic   |Asan Ka Na Ba                                        |73.0                           |
|acoustic   |Comedy                                               |73.0                           |
|acoustic   |Crazy Rich Asians (Original Motion Picture Soundtrack)|71.0                           |
|acoustic   |America Town                                         |70.0                           |
|acoustic   |Little Voice                                         |70.0                           |
|acoustic   |Nangangamba                                          |70.0                           |
|afrobeat   |Calle 13 (Explicit Version)                          |75.0                           |
|afrobeat   |Entren Los Que Quieran                               |68.33333333333333              |
+-----------+-----------------------------------------------------+-------------------------------+
```

only showing top 10 rows

Query 3 - Getting top 8 most live albums in every genre on the basis of liveness.

```
%python
from pyspark.sql import Window, DataFrame, SparkSession
import sys
from pyspark.sql import functions as f
from pyspark.sql.types import StringType, IntegerType

#Reading csv as a Pyspark dataframe
df = spark.read.options(header=True,).csv("dbfs:/FileStore/tables/users/pgoyal19/train.csv")

#Cleaning the dataset by removing irrelevant data from track_genre column
df = df.filter(~(f.col("track_genre").contains("1"))).filter(~(f.col("track_genre").contains("3"))).filter(~(f.col("track_genre").contains("4"))).filter(~
(f.col("track_genre").contains("5"))).filter(~(f.col("track_genre").contains("7"))).filter(~(f.col("track_genre").contains("8")))

#Selecting required columns and removing outliar data having null values.
df = df.select("track_genre", "track_name", "liveness").filter(f.col("liveness").isNotNull())

#Defining the window partition for group by on the basis track_genre and ordering them on the basis of liveness.
w = Window.partitionBy("track_genre").orderBy(f.col("track_genre").desc(), f.col("liveness").desc())

#Getting top 8 rows for each track_genre on the basis of liveness.
final_putput_first_eight_rows = df.withColumn("ln", f.row_number().over(w)) \
    .filter(f.col("ln") <=8).select("track_genre", "track_name", "liveness")

final_putput_first_eight_rows.show(10, truncate=0)
```

```
+-----------+-----------------------------------------------------+--------+
|track_genre|track_name                                           |liveness|
+-----------+-----------------------------------------------------+--------+
|alternative|Rodo cotidiano - Ao vivo                             |0.965   |
|alternative|Un Pacto - Live In Buenos Aires / 2001               |0.961   |
|alternative|Poli / Love - En Vivo                                |0.958   |
|alternative|Tendo A Lua - Ao Vivo                                |0.952   |
|alternative|Toloache Pa' Mi Negra - En Vivo                      |0.939   |
|alternative|Pescador de ilusões - Ao vivo acústico               |0.933   |
|alternative|Labios Rotos - En Vivo                               |0.931   |
|alternative|Rompecabezas De Amor... - Live In Buenos Aires / 2016|0.896   |
|ambient    |Coco                                                 |0.975   |
|ambient    |"Toxygene - 7"" Edit"                                |0.747   |
+-----------+-----------------------------------------------------+--------+
only showing top 10 rows
```