Practical Work 1 in Cloud Computing

Gustavo Freitas Cunha

Federal University of Minas Gerais (UFMG)
Belo Horizonte - MG - Brazil

gustavocunha@dcc.ufmg.br

1 Installation of Dependencies

At first, the operational system was imported and then the Spark session was set on the notebook. Additionally, another libraries, such as PySpark and MatPlotLib, has been imported.

2 Task 1 - Statistics about songs duration

2.1 Generating a table containing the minimum, average and maximum duration, in milliseconds, of the songs in the dataset

For this subtask, after set playlists and tracks as datasets, named *df_playlists* and *df_tracks*, respectively, with *spark.read.json*, I used the Spark aggregate function, by the field *duration_ms*, to get the min, average and max numbers. The results are presented in Table 1.

Min Duration [ms]	**Average Duration [ms]**	**Max Duration [ms]**
0	234408.55	10435467

Table 1: Basic stats about the track's durations

From the table, we can see that there's some outliers, once the minimum duration is zero and the maximum is almost 3 hours, despite the 3.9 minutes average.

2.2 Computing the first and third quartiles as well as the interquartile range (IRQ)

For this subtask, I used the PySpark function *approxQuantile* to get the quartiles. The IRQ just follows, doing $Q_3 - Q_1$.

The results show that Q_1 is 198026 ms, that is approximately 3.3 minutes and Q_3 is 258133 ms, that is approximately 4.3 minutes. The IRQ, therefore, is something like 1 min. More precisely, we have IRQ = 60107 ms.

2.3 Computing the set of songs with durations that are not outliers, as defined by the IQRR methodology

To get the set of songs non outliers concerning it's durations, after made the calculus of the posterior and inferior limits, I used the PySpark filter function to generate another dataframe containing the desired data.

The results shows that non-outlier data totals 94% of the dataset, that is 10189635 songs.

2.4 Generating a new table containing the minimum, average and maximum duration of the non-outliers songs

Finally, for this subtask I just filtered the negation of the values obtained on the previous subtask and applied the aggregation function to generate the stats, that are presented in Table 2.

Min Duration [ms]	**Average Duration [ms]**	**Max Duration [ms]**
107866	226795.86	348293

Table 2: Basic stats of the the track's non-outliers

From the table, we can see that the outliers, such as that 0 ms duration song, was removed from the dataset. We can also see that the average duration has decreased, since the outliers with very high value was disregarded in this set.

3 Task 2 - Finding the most popular artists over time

For this task, at first, I have got the five most popular artists, by doing a count of tracks, grouping by the field *artist_name*. Then, I converted the *modified_at* field to get the year, and I did another group by, but now using the year and only considering the five most popular artists that I previously got. Finally, I plotted the graph with the data obtained. The results are shown in Figure 1.

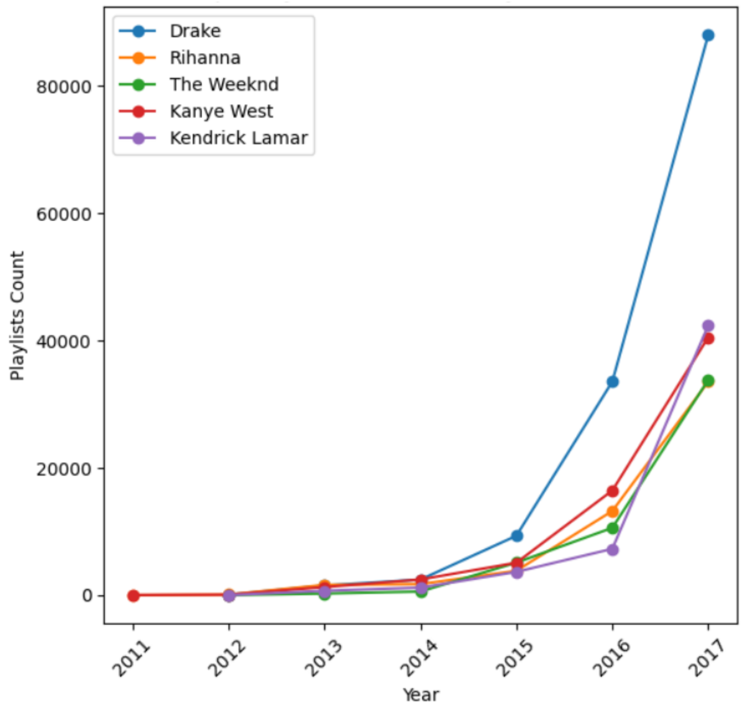

Figure 1: Artists Popularity Over the Years

By the graph we can see that Drake appears to lead the number of playlists where his musics shows off since 2011. From 2014 onwards, his leadership became even more isolated, with an increasingly significant difference with Kanye West, in second place. The three others singers. The other three artists have had a fierce dispute over the years, which has not shown major changes, except for Kendrick Lamar, who went straight from last to first place from 2016 to 2017.

4 Task 3 - Playlists's behavior

First, I grouped the data by the *pid* and *artist_name* fields to calculate the number of times each artist appears in each playlist, using the *count()* function. Then, I calculated the maximum appearance count for any artist within each playlist by grouping again by *pid* and applying *F.max("count")* to find the highest value. Next, I calculated the total number of songs in each playlist by grouping by *pid* and using *count()* once more.

To find the prevalence of the most frequent artist per playlist, I joined the tables with the most frequent artist count and the total song count, then divided the highest appearance count by the total number of songs in each playlist. This gave me a fraction representing how much of each playlist is attributed to its most frequent artist. I collected and sorted these prevalence values into a list, then calculated the CDF by creating an array from 1 to the length of the sorted values and dividing it by the total count.

Finally, I plotted the CDF shown in Figure 2, with artist prevalence on the x-axis and the cumulative probability on the y-axis.

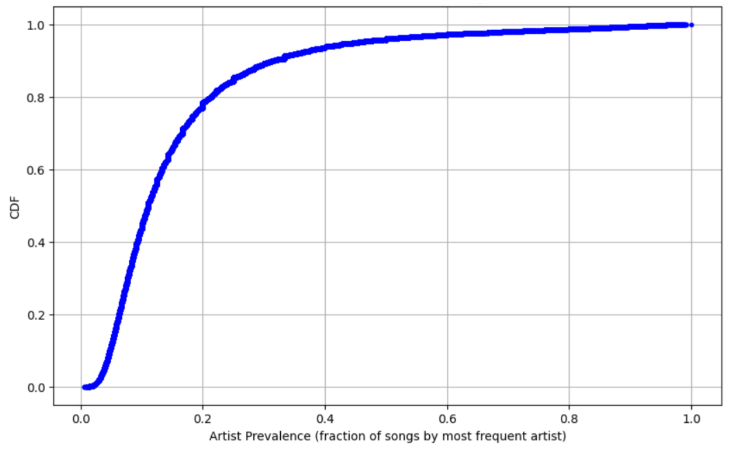

Figure 2: CDF of Artists Prevalence on Playlists

The CDF plot shows the distribution of artist prevalence across playlists, indicating how frequently the most represented artist appears relative to the total number of songs in each playlist. The curve rises steeply at lower prevalence values, meaning that in most playlists, the most frequent artist makes up a small fraction of the songs. This is clear when we look at the point (0.2, 0.8), which tells us that in 80% of playlists, there is only 20% prevalence of the most popular artist. As we move toward higher prevalence values, the curve flattens, showing that only a few playlists have high concentrations of a single artist. This suggests that most playlists tend to feature a diverse range of artists, with only a small percentage dominated by one artist.