Task1

The approach I took:

1. **Calculating basic duration statistics**: To obtain the minimum, average, and maximum song durations in the dataset, the agg function in PySpark was used on the tracks_df DataFrame. Aggregation functions min, avg, and max were applied to the duration_ms column, and the results were aliased for better readability.
2. **Computing quartiles and inter - quartile range**: The approxQuantile function in PySpark was utilized to estimate the first quartile (Q1, corresponding to the 25th percentile) and the third quartile (Q3, corresponding to the 75th percentile) of the song durations. The inter - quartile range (IQR) was then calculated as the difference between Q3 and Q1.
3. **Identifying non - outliers**: The Interquartile Range Rule (IQRR) was applied to filter out the outliers. A filter operation was performed on the tracks_df DataFrame, keeping only the rows where the song duration (duration_ms) was within the range defined by Q1 - 1.5 * IQR and Q3 + 1.5 * IQR.
4. **Analyzing remaining data after outlier removal**: The number of outlier records was determined by subtracting the count of non_outlier records from the total count of records in the tracks_df DataFrame. Then, the minimum, average, and maximum durations were recalculated for the non_outlier subset of the data.

Result I obtained:

```
+------------+------------------+------------+
|min_duration|      avg_duration|max_duration|
+------------+------------------+------------+
|           0|234408.54976216817|    10435467|
+------------+------------------+------------+


Q1: 198026.0,Q3: 258133.0,IQR:60107.0

Number of outliers: 570909
[Stage 131:================================================>      (21 + 3) / 24]
+------------+-----------------+------------+
|min_duration|     avg_duration|max_duration|
+------------+-----------------+------------+
|      107866|226795.8593433425|      348293|
+------------+-----------------+------------+
```

1. **Initial duration statistics**: The initial results showed a minimum duration of 0 milliseconds, an average duration of approximately 234,408.55 milliseconds, and a maximum duration of 10,435,467 milliseconds.
2. **Quartiles and IQR**: The first quartile (Q1) was 198,026.0 milliseconds, the third quartile (Q3) was 258,133.0 milliseconds, and the inter - quartile range (IQR) was 60,107.0 milliseconds. These values represent the spread of the middle 50% of the song durations in the dataset.
3. **Outlier detection and removal**: A total of 570,909 outliers were identified and removed from the dataset. This significant number of outliers indicates that the song duration data

has a long - tailed distribution with many extreme values.

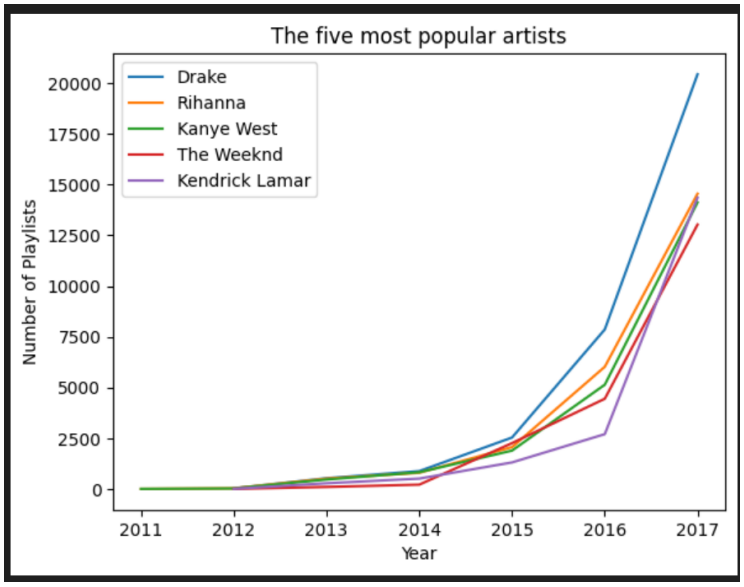
4. **Statistics after outlier removal**: After removing the outliers, the minimum duration increased to 107,866 milliseconds, the average duration decreased to approximately 226,795.86 milliseconds, and the maximum duration reduced to 348,293 milliseconds. The reduction in the average and maximum durations after outlier removal shows that the original average and maximum were heavily influenced by the extreme values. The new statistics provide a more representative view of the typical song durations in the dataset, excluding the extreme cases.

Task2

The approach I took:

1. **Identifying top - 5 popular artists**: First, the tracks_df DataFrame was grouped by the artist_name column. For each artist, the number of distinct playlists (pid) they appeared in was counted using the countDistinct function and aliased as playlist_count. The results were then ordered in descending order of playlist_count, and the top 5 artists were selected.
2. **Preparing time - related data**: The modified_at column in the playlists_df DataFrame, which likely represents the last - modification time of each playlist, was cast to a timestamp type and then the year was extracted using the year function from PySpark. This new year column was added to the playlists_df.
3. **Joining and aggregating data**:The tracks_df and playlists_df were joined on the pid (playlist ID) column. Then, the joined DataFrame was filtered to only include the top 5 artists identified earlier. The data was grouped by both artist_name and year, and for each group, the number of distinct playlists (playlist_count) was counted again. The results were ordered by artist_name and year.
4. **Plotting the results**: The aggregated data was collected to the local machine. For each of the top 5 artists, the years and the corresponding number of playlists they appeared in were extracted. These data were then used to plot line graphs for each artist on the same plot, with appropriate axis labels and a title.

Result I obtained:



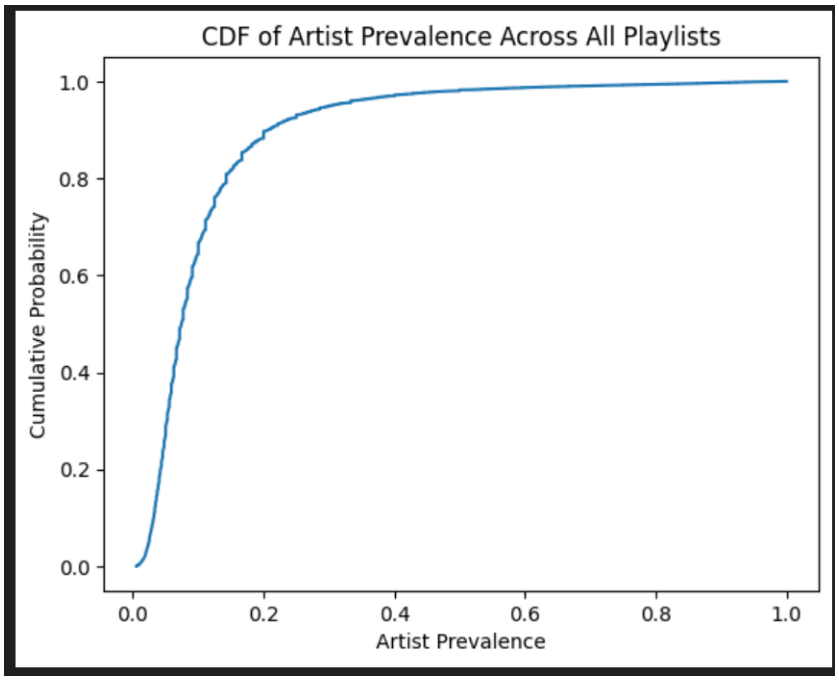
1. **Popular artists identification**: The top 5 artists identified were Drake, Rihanna, Kanye West, The Weeknd, and Kendrick Lamar. This indicates that these artists have a significant presence in the Spotify playlists within the dataset.
2. **Time - series trends**: The line graph shows an upward trend in the number of playlists containing these artists over the years from 2011 to 2017. This upward trend might suggest growing popularity for these artists over time, or it could also be related to the overall growth of the Spotify platform and the increasing number of users creating playlists.
3. **Comparative analysis**: By comparing the lines of different artists, we can observe differences in their growth rates and popularity levels. For instance, while all artists show growth, some like Drake and Rihanna seem to have a more rapid increase in playlist appearances compared to others like Kendrick Lamar in the later years.

Task3

The approach I took:

1. **Counting artist - specific song numbers per playlist**: The tracks_df DataFrame was first grouped by both pid (playlist ID) and artist_name. For each group, the number of distinct songs (track_uri) by that artist in the corresponding playlist was counted using the countDistinct function and aliased as song_count. This step aimed to determine how many songs of each artist were present in each playlist.
2. **Identifying the most - frequent artist per playlist:**
   The artist_count_per_playlist DataFrame was then grouped by pid again. For each playlist, the maximum value of song_count was found using the max function and aliased as max_song_count. This identified the highest number of songs by any single artist in each playlist.
3. **Filtering for the most - frequent artist:**
   The artist_count_per_playlist and max_artist_per_playlist DataFrames were joined on the pid column. Then, a filter was applied to keep only the rows where the song_count was equal to the max_song_count. This ensured that only the most - frequent artist (or artists in case of ties) in each playlist were selected.
4. **Calculating total songs per playlist**: The tracks_df was grouped by pid once more, and the total number of distinct songs in each playlist was counted using countDistinct and aliased as total_song_count.
5. **Computing artist prevalence**:
   The joined_max_artist and total_songs_per_playlist DataFrames were joined on pid. A new column named prevalence was added, which was calculated as the ratio of max_song_count to total_song_count. This represented the proportion of songs by the most - frequent artist in each playlist.
6. **Generating the CDF**: The prevalence data was collected to the local machine. It was sorted, and a cumulative distribution function (CDF) was calculated using NumPy. The CDF was then plotted to visualize the distribution of artist prevalence across all playlists.

Result I obtained:



1. **Shape of the CDF**: The CDF plot shows a rapidly increasing curve at the beginning, which then flattens out as it approaches a cumulative probability of 1. This indicates that a significant portion of playlists have a relatively low artist prevalence. In other words, many playlists have a diverse set of artists, with no single artist dominating the playlist.
2. **Interpretation of prevalence values**: A low prevalence value means that the most - frequent artist in a playlist does not have a large share of the songs. As the prevalence approaches 1, it implies that a single artist makes up almost all of the songs in the playlist. The fact that the curve rises steeply initially suggests that playlists with a more diverse selection of artists are quite common.
3. **Insight into playlist creation behavior**: Based on the CDF, it can be inferred that playlists with more diverse songs are more common than those with many songs by the same artist. This could reflect the listening habits of Spotify users, who might prefer to create playlists that offer a wide range of musical experiences rather than focusing on a single artist.