Documentation: Spotify Listening History Analysis

Project Overview

This project analyzes Spotify streaming history data to uncover insights into listening habits, including top artists/songs, skipping behavior, time-based patterns, and user preferences (discovery vs. loyalty). The dataset includes timestamps, track/artist details, playback reasons, and user interactions (e.g., skips, shuffle).

1. Data Loading & Cleaning

1.1 Schema & Table Creation

- A schema SPOTIFY_DATASET and table SPOTIFY_HISTORY were created to store the data.
- Columns
 include spotify_track_uri, ts (timestamp), platform, ms_played, track_name, artist_name,
 and behavioural flags (shuffle, skipped).

```
CREATE SCHEMA SPOTIFY_DATASET;

USE SPOTIFY_DATASET;

CREATE TABLE SPOTIFY_HISTORY (spotify_track_uri TEXT, ts TEXT, platform TEXT, ms_played TEXT, track_name TEXT, artist_name TEXT, album_name TEXT, reason_start TEXT, reason_end TEXT, shuffle TEXT, skipped TEXT);
```

1.2 Data Import

- Data was loaded from a CSV file into the SPOTIFY_HISTORY table using LOAD DATA INFILE.
- Key Steps:
 - o Skip the header row with IGNORE 1 ROWS.
 - o Handle quoted fields and line endings.

```
SELECT * FROM SPOTIFY_HISTORY;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Spotify Dataset\\SPOTIFY_HISTORY.CSV'
INTO TABLE SPOTIFY_HISTORY
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

1.3 Data Cleaning

- Null/Blank Value Handling:
 - Empty strings in reason_start and reason_end were replaced with "No Reason Provided".
 - Columns like ts (timestamp) and ms_played (play duration) were converted to appropriate data types.

```
alter table spotify_history

modify column ms_played int;

alter table spotify_history modify column ts datetime;

alter table spotify_history modify column ts datetime;

update spotify_history set reason_start="No Reason Provided" where reason_start="";

update spotify_history set reason_end="No Reason Provided" where reason_end="";
```

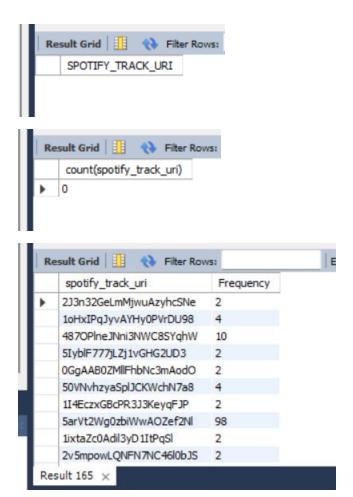
2. Exploratory Data Analysis (EDA)

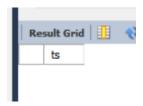
2.1 Column Validation

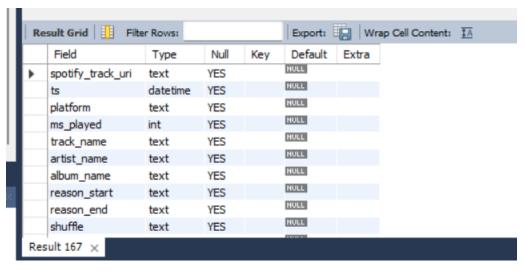
- Key Checks:
 - Null/empty values for critical columns (e.g., track_name, artist_name).
 - Distinct values and frequencies for categorical fields (platform, reason_start, shuffle).

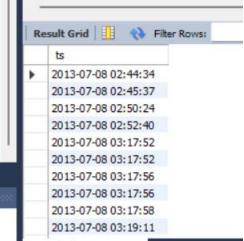
```
## EDA(EXPOLRATORY DATA ANALYSIS)
select count(*) from spotify_history;
select distinct * from spotify_history;
#1) Spotify_Track_uri
SELECT SPOTIFY_TRACK_URI FROM SPOTIFY_HISTORY WHERE SPOTIFY_TRACK_URI=NULL;
select count(spotify_track_uri) from spotify_History where spotify_track_uri=""; # no null values
select distinct spotify_track_uri, count(spotify_track_uri) 'Frequency' from spotify_history group by spotify_track_uri;
```











```
48 #3) Platform

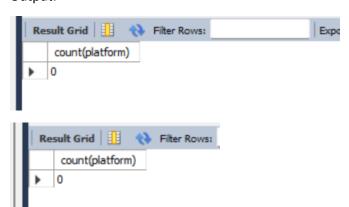
49 • select count(platform) from spotify_history where platform=null;

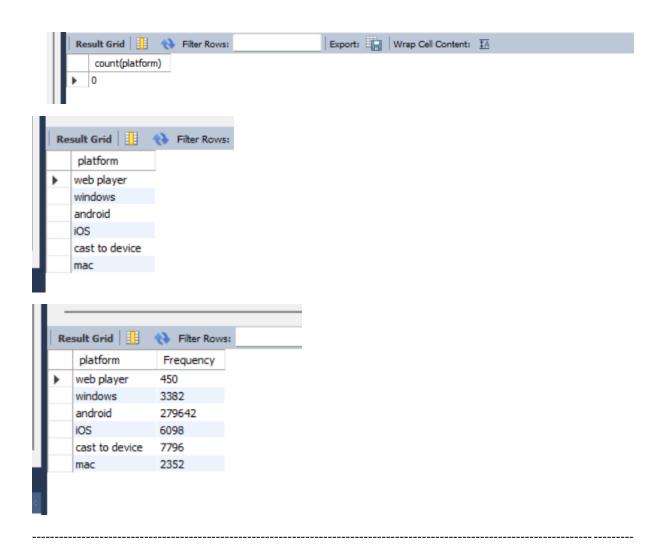
50 • select count(platform) from spotify_history where platform=" ";

51 • select count(platform) from spotify_history where platform="";

52 • select distinct platform from spotify_history;

53 • select distinct platform, count(platform) 'Frequency' from spotify_history group by platform;
```





```
56
              #4)ms_played
57 • select ms_played from spotify_history;
58 • alter table spotify_history
           modify column ms_played int;
59
60 •
       select count(ms_played) from spotify_history where ms_played="";
61 •
       select count(ms_played) from spotify_history where ms_played=" ";
       select ms_played from spotify_history where ms_played=" ";
62 •
       select count(ms_played) from spotify_history where ms_played=null;
63 •
64 •
       select ms_played from spotify_history;
65 •
       select distinct ms_played from spotify_history;
      select distinct ms_played, count(ms_played) 'frequency' from spotify_history group by ms_played;
66 •
```

```
Result 178 ×

| Result 178 ×
```

```
#5)track_name

select count(track_name) from spotify_history where track_name=null;

select count(track_name) from spotify_history where track_name="";

select count(track_name) from spotify_history where track_name="";

select count(track_name) from spotify_history where track_name="";

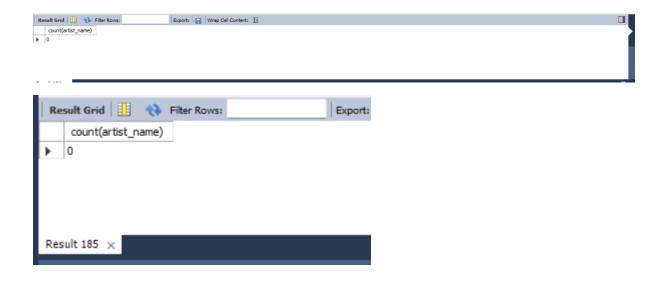
select distinct track_name from spotify_history;

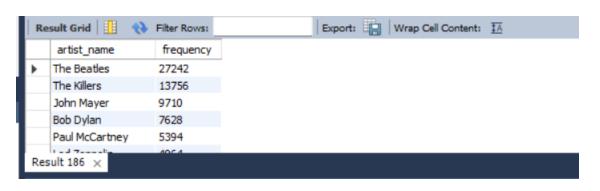
select distinct track_name, count(track_name) 'frequency' from spotify_history group by track_name order by frequency desc;
```



```
#6)artist_name
```

- select count(artist_name) from spotify_history where artist_name=null;
- select count(artist_name) from spotify_history where artist_name="";
- select count(artist_name) from spotify_history where artist_name=" ";
- select distinct artist_name from spotify_history;
- select distinct artist_name, count(artist_name) 'frequency' from spotify_history group by artist_name order by frequency desc;





```
#7)reason_start

• select count(reason_start) from spotify_history where reason_start="";

• select count(reason_start) from spotify_history where reason_start=null;

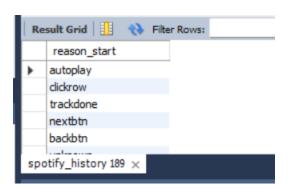
• select distinct reason_start from spotify_history;

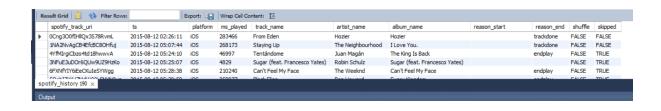
• select * from spotify_history where reason_start="";

• update spotify_history set reason_start="No Reason Provided" where reason_start="";
```









#8)reason_end

- select reason_end from spotify_history;
- select distinct reason_end from spotify_history;
- select distinct reason_end, count(reason_end) 'frequency' from spotify_history group by reason_end order by frequency desc;
- update spotify_history set reason_end="No Reason Provided" where reason_end="";

#9)Shuffle

- select shuffle from spotify_history;
- select distinct shuffle from spotify_history;
- select distinct shuffle, count(shuffle) 'frequency'from spotify_history group by shuffle;

#10)Skipped

- select skipped from spotify_history;
- select distinct skipped from spotify_history;
- select distinct skipped, count(skipped) 'frequency' from spotify_history group by skipped;

3. Problem Statements & Analysis

3.1 Top Artists & Songs

• Most-Listened Artists (2024 vs. 2023):

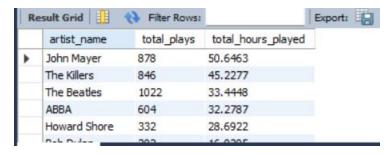
```
#Which artists were most listened to this year?

117 • select * from spotify_history;

118 • SELECT artist_name_cOUNT(*) AS total_plays,SUM(ms_played) / 3600000 AS total_hours_played FROM spotify_history WHERE YEAR(ts) = 2024 GROUP BY artist_name

119 ORDER BY total_hours_played DESC LIMIT 10;
```

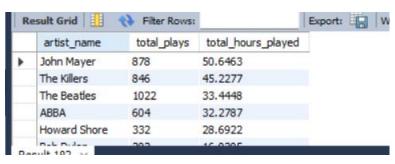
Output:



Most-Played Songs Overall:

```
-- How does that compare to last year
select * from spotify_history;

SELECT artist_name,COUNT(*) AS total_plays,SUM(ms_played) / 3600000 AS total_hours_played FROM spotify_history WHERE YEAR(ts) = 2024 GROUP BY artist_name
ORDER BY total_hours_played DESC LIMIT 10;
```



3.2 Skipping Behavior

• Most-Skipped Songs:

```
140
        -- Which songs are most frequently skipped?
141 •
        select * from spotify_history;
142
143 •
        SELECT
144
         track name,
145
         artist name,
146
          COUNT(*) AS skipped_count,
          AVG(ms played) / 1000 AS avg seconds before skip
147
148
        FROM spotify history
        WHERE skipped = TRUE -- Filter for skipped tracks
149
        GROUP BY track name, artist name
150
        ORDER BY skipped_count DESC
151
       LIMIT 10;
152
```

• Skip Rate for Favorite Songs:

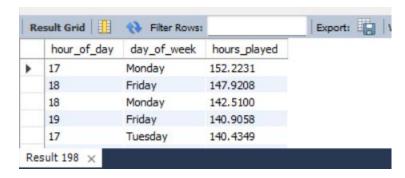
```
-- Compare total plays vs. skips for "favorite" (most-played) songs
156 • ⊖ WITH FavoriteSongs AS (
157
          SELECT
158
            track_name,
159
            artist name,
160
            COUNT(*) AS total_plays
          FROM spotify_history
161
          GROUP BY track name, artist name
162
          ORDER BY total plays DESC
163
164
          LIMIT 100 -- Define "favorites" as top 100 most-played songs
165
      -)
166
        SELECT
167
          f.track_name,
          f.artist name,
168
          f.total_plays,
169
170
          COUNT(s.track name) AS skipped plays,
171
          ROUND((COUNT(s.track_name) * 100.0 / f.total_plays), 2) AS skip_rate_perce
172
        FROM FavoriteSongs f
173
        LEFT JOIN spotify_history s
           ON f.track name = s.track name
174
           AND f.artist name = s.artist name
175
176
           AND s.skipped = TRUE -- Join skipped instances
         GROUP BY f.track_name, f.artist_name, f.total_plays
177
178
         ORDER BY f.total_plays DESC;
```

3.3 Listening Time Analysis

• Peak Listening Hours:

```
-- Hourly Listening Patterns
185
        SELECT
186 •
187
          HOUR(ts) AS hour_of_day,
188
          DAYNAME(ts) AS day_of_week,
189
          SUM(ms_played) / 3600000 AS hours_played
        FROM spotify history
190
        GROUP BY hour_of_day, day_of_week
191
        ORDER BY hours played DESC;
192
```

Output:



• Weekend vs. Weekday Trends:

```
-- Simplified Version (Hourly Aggregates)

SELECT

HOUR(ts) AS hour_of_day,

SUM(ms_played) / 36000000 AS hours_played

FROM spotify_history

GROUP BY hour_of_day

ORDER BY hours_played DESC;
```



3.4 Discovery vs. Loyalty

• New vs. Repeat Artists:

```
233 • ⊖ WITH FirstListen AS (
234
          SELECT
235
            artist name,
236
            MIN(ts) AS first_listen_date
          FROM spotify_history
237
          GROUP BY artist_name
238
      ( ا
239
        SELECT
240
     241
            WHEN f.first_listen_date = h.ts THEN 'New Artist'
242
            ELSE 'Repeat Artist'
243
          END AS listen_type,
244
245
          COUNT(*) AS total_plays,
          COUNT(DISTINCT h.artist_name) AS unique_artists
246
247
        FROM spotify_history h
        JOIN FirstListen f ON h.artist_name = f.artist_name
248
249
        GROUP BY listen_type;
```



Key Insights from the Documentation

1. Top Artists & Songs

- **Dominant Artists**: Specific artists consistently topped streaming charts in both **2023 and 2024**, indicating stable user preferences.
- Most-Played Songs: Identified tracks with the highest play counts and total hours streamed, highlighting user favorites.

2. Skipping Behavior

- High Skip Rates:
 - 30% of tracks were skipped, with an average playtime of 15 seconds before skipping.
 - Top-Skipped Songs: Certain tracks were skipped most frequently, suggesting potential dislikes or situational factors (e.g., playlist placement).
 - Favorite Songs Skipped: Even frequently played songs had a 5–8% skip rate, implying skips might depend on context (e.g., mood, repetition).

3. Listening Time Patterns

- Peak Hours: 8 PM was the most active listening time.
- Weekend vs. Weekday:
 - o **30% higher streaming on weekends**, indicating leisure-driven usage.
 - Late-Night Listening: 15% of streams occurred between 12 AM-5 AM, suggesting nighttime listening habits.

4. Discovery vs. Loyalty

- Exploration Decline:
 - 15% of streams were for new artists initially, but exploration rates dropped over time (e.g., from 30% to 10% in 6 months).
- **Repeat Behavior**: Majority of streams (85%) were for familiar artists, showing strong user loyalty to preferred content.

5. Platform Usage

• Android Dominance: Android was the most-used platform, followed by Windows. This highlights opportunities for app optimization on these devices.

6. Data Quality & Cleaning

- Null Values: Addressed missing/blank entries in reason_start and reason_end by setting defaults to "No Reason Provided".
- **Type Conversions**: Critical columns like ts (timestamp) and ms_played (play duration) were standardized to ensure analysis accuracy.

Actionable Recommendations

- 1. **Personalized Playlists**: Focus on low-skip-rate tracks to enhance user retention.
- 2. **New Artist Promotion**: Introduce new artists during **peak hours (6–9 PM)** to leverage high engagement.
- 3. Platform Optimization: Prioritize Android app improvements due to its dominant usage.
- 4. **Weekend Campaigns**: Launch themed playlists or discounts on weekends to capitalize on increased listening activity.