Spotify SQL Project - Questions & Queries

Table Creation

```
CREATE TABLE spotify (
    artist VARCHAR(255),
    track VARCHAR(255),
    album VARCHAR(255),
    album_type VARCHAR(50),
    danceability FLOAT,
    energy FLOAT,
    loudness FLOAT,
    speechiness FLOAT,
    acousticness FLOAT,
    instrumentalness FLOAT,
    liveness FLOAT,
    valence FLOAT,
    tempo FLOAT,
    duration_min FLOAT,
    title VARCHAR(255),
    channel VARCHAR(255),
    views FLOAT,
    likes BIGINT,
    comments BIGINT,
    licensed BOOLEAN,
    official_video BOOLEAN,
    stream BIGINT,
    energy_liveness FLOAT,
    most_played_on VARCHAR(50)
);
```

1. Count total number of records in the dataset:

```
SELECT COUNT(*) FROM spotify;
```

2. Count total number of unique artists:

```
SELECT COUNT(DISTINCT artist) FROM spotify;
```

3. Count total number of unique albums:

```
SELECT COUNT(DISTINCT album) FROM spotify;
```

4. Find distinct album types:

```
SELECT DISTINCT album_type FROM spotify;
```

5. Find maximum and minimum track duration (in minutes):

```
SELECT MAX(duration_min) FROM spotify;
SELECT MIN(duration_min) FROM spotify;
```

6. Display tracks with duration 0:

```
SELECT * FROM spotify WHERE duration_min = 0;
```

7. Delete tracks with duration 0:

Spotify SQL Project - Questions & Queries

```
DELETE FROM spotify WHERE duration_min = 0;
```

8. List all distinct channels:

```
SELECT DISTINCT channel FROM spotify;
```

9. List all platforms where a track was most played:

```
SELECT DISTINCT most_played_on FROM spotify;
```

10. Retrieve names of tracks with more than 1 billion streams:

```
SELECT track FROM spotify WHERE stream > 1000000000;
```

11. List all albums along with their respective artists:

```
SELECT DISTINCT album, artist FROM spotify ORDER BY album;
```

12. Total comments on licensed tracks:

```
SELECT SUM(comments) AS total_comment FROM spotify WHERE licensed = TRUE;
```

13. List all tracks that are singles:

```
SELECT track, album_type FROM spotify WHERE album_type = 'single';
```

14. Count total number of tracks by each artist:

```
SELECT artist, COUNT(track) AS total_tracks FROM spotify GROUP BY artist;
```

15. Average danceability of tracks per album:

```
SELECT album, AVG(danceability) AS average_danceability FROM spotify GROUP BY album
ORDER BY average_danceability DESC;
```

16. Top 5 tracks with highest energy:

```
SELECT track, MAX(energy) FROM spotify GROUP BY track ORDER BY MAX(energy) DESC LIMIT 5;
```

17. Tracks with views and likes where official_video = TRUE:

```
SELECT track, SUM(views) AS total_views, SUM(likes) AS total_likes FROM spotify WHERE
official_video = TRUE GROUP BY track ORDER BY total_views DESC;
```

18. Total views for each album:

```
SELECT album, track, SUM(views) AS total_views FROM spotify GROUP BY album, track ORDER
BY total_views DESC;
```

19. Tracks streamed more on Spotify than on YouTube:

```
SELECT * FROM (
    SELECT track,
             COALESCE(SUM(CASE WHEN most_played_on = 'Youtube' THEN stream END), 0) AS
stream_on_youtube,
             COALESCE(SUM(CASE WHEN most_played_on = 'Spotify' THEN stream END), 0) AS
stream_on_spotify
    FROM spotify
    GROUP BY track
) AS s
WHERE stream_on_spotify > stream_on_youtube AND stream_on_youtube <> 0;
```

Spotify SQL Project - Questions & Queries

20. Top 3 most viewed tracks per artist using window functions:

```
WITH ranked_tracks AS (
    SELECT artist, track, SUM(views) AS total_views,
           DENSE_RANK() OVER (PARTITION BY artist ORDER BY SUM(views) DESC) AS rank
    FROM spotify
    GROUP BY artist, track
)
SELECT * FROM ranked_tracks WHERE rank <= 3;
```

21. Tracks with liveness above average:

```
SELECT track, liveness FROM spotify WHERE liveness > (SELECT AVG(liveness) FROM spotify);
```

22. Difference between highest and lowest energy per album:

```
WITH energy_stats AS (
      SELECT album, MAX(energy) AS max_energy, MIN(energy) AS min_energy FROM spotify GROUP BY album
)
SELECT album, max_energy - min_energy AS energy_difference FROM energy_stats ORDER BY energy_difference DESC;
```

23. Tracks where energy-to-liveness ratio is greater than 1.2:

```
SELECT track, energy_liveness FROM spotify WHERE energy_liveness > 1.2;
```

24. Cumulative likes for tracks ordered by views:

```
SELECT track, views, likes,
       SUM(likes) OVER (ORDER BY views DESC) AS cumulative_likes
FROM spotify;
```