Spotify Advanced SQL Project and Query Optimization

SQL Query Solutions

1. Retrieve the names of all tracks that have more than 1 billion streams.

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

2. List all albums along with their respective artists.

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

3. Get the total number of comments for tracks where licensed = TRUE.

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

4. Find all tracks that belong to the album type 'single'.

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

5. Count the total number of tracks by each artist.

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

1. Calculate the average danceability of tracks in each album.

```
SELECT album, AVG(danceability) AS avg_danceability
FROM spotify
GROUP BY album;
```

2. Find the top 5 tracks with the highest energy values.

```
SELECT track, energy
FROM spotify
ORDER BY energy DESC
LIMIT 5;
```

3. List all tracks along with their views and likes where official_video = TRUE.

```
SELECT track, views, likes
FROM spotify
WHERE official_video = TRUE;
```

4. For each album, calculate the total views of all associated tracks.

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

5. Retrieve the track names that have been streamed on Spotify more than YouTube.

```
SELECT track
FROM spotify
WHERE most_played_on = 'Spotify'
  AND stream > views;
```

1. Find the top 3 most-viewed tracks for each artist using window functions.

```
SELECT artist, track, views
FROM (
   SELECT artist, track, views,
        ROW_NUMBER() OVER (PARTITION BY artist ORDER BY views DESC) AS rn
   FROM spotify
) t
WHERE rn <= 3;
```

2. Write a query to find tracks where the liveness score is above the average.

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

3. Calculate the difference between the highest and lowest energy values for tracks in each album.

```
WITH cte AS (
   SELECT album, MAX(energy) AS highest_energy, MIN(energy) AS lowest_energy
   FROM spotify
   GROUP BY album
)
SELECT album, highest_energy - lowest_energy AS energy_diff
FROM cte
ORDER BY energy_diff DESC;
```

4. Find tracks where the energy-to-liveness ratio is greater than 1.2.

```
SELECT track, (energy / NULLIF(liveness, 0)) AS ratio
FROM spotify
WHERE (energy / NULLIF(liveness, 0)) > 1.2;
```

5. Calculate the cumulative sum of likes for tracks ordered by views.

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