Spotify Data Analysis

```
--Schema---
CREATE TABLE spotify
(
  artist  VARCHAR(50),
  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(100),
  views BIGINT,
  likes BIGINT,
  comments BIGINT,
  licensed BOOLEAN,
  official_video BOOLEAN,
  stream BIGINT,
```

```
  energyliveness FLOAT,
  most_playedon VARCHAR(50)
);

SELECT * FROM spotify;
```

```
-----EDA------

SELECT COUNT(*) FROM spotify;
SELECT COUNT(DISTINCT artist) FROM spotify;
SELECT COUNT(DISTINCT album) FROM spotify;

SELECT DISTINCT album_type FROM spotify;

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

SELECT * FROM spotify
WHERE duration_min = 0;

DELETE FROM spotify
WHERE duration_min = 0;

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

```
SELECT DISTINCT most_playedon FROM spotify;
```

-----Data Analysis Easy Categories-------

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

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

Q.2 List all album along with their respective artist.

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

Q.3 Get the total number of comments for track where licensed = true.

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

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

```
SELECT
        track,
```

```
        album_type
FROM spotify
WHERE album_type ILIKE '%single%';
```

Q.5 Count the total number of tracks by each artists.

```
SELECT
        artist,
        COUNT(track) AS total_track
FROM spotify
GROUP BY 1
ORDER BY 2 DESC;
```

---Medium Level---------

Q.6 Calculate the average danceability of tracks in each album.

```
SELECT
        album,
        track,
    AVG(danceability) AS avg_danceability
FROM spotify
GROUP BY 1,2;
```

Q.7 Find the top 5 track with the highest energy value.

```
SELECT
        track,
```

```
        MAX(energy) AS highest_energy
FROM spotify
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

Q.8 List all tracks along with their views and likes where official_video = TRUE.

```
SELECT
        track,
        views,
        likes,
        official_video
FROM spotify
WHERE official_video IS true;
```

Q.9 For each album, calculate the total views of all associated tracks.

```
SELECT
        album,
        track,
        SUM(views) AS total_views
FROM spotify
GROUP BY 1,2
ORDER BY 3 DESC;
```

Q.10 Retrieve the track names that have been streamed on spotify more than youtube.

```
WITH CTE
 AS
(
SELECT
        track,
        COALESCE(SUM(CASE WHEN most_playedon ='Youtube' THEN stream END),0) AS streamed_on_youtube,
        COALESCE(SUM(CASE WHEN most_playedon = 'Spotify' THEN stream END),0) AS streamed_on_spotify
FROM spotify
GROUP BY 1
)
SELECT * FROM CTE
WHERE
    streamed_on_spotify > streamed_on_youtube
        AND
        streamed_on_youtube <> 0;
```

Advanced Query

Q11. Find the top 3 most-viewed tracks for each artist using window functions.

```
WITH CTE AS
(
SELECT
        artist,
        track,
        SUM(views) AS total_view,
        DENSE_RANK() OVER(PARTITION BY artist ORDER BY SUM(views) DESC) AS rank
FROM spotify
GROUP BY 1,2
)
 SELECT * FROM CTE
 WHERE rank <= 3;
```

Q12. Write a query to find tracks where the liveness score is above the average.

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

Q13. Use a with clause to 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 1
)
 SELECT
        album,
        highest_energy - lowest_energy AS energy_diff
 FROM CTE
 ORDER BY 2 DESC;
```

Q14. Find tracks where energy-to-liveness ratio is greater than 1.2

```
WITH CTE
AS
(
SELECT
```

```
        track,
        energy,
        liveness,
        (energy/liveness) AS energy_to_liveness_ratio
FROM spotify
)
SELECT
        track,
        energy_to_liveness_ratio
FROM CTE
WHERE energy_to_liveness > 1.2
        AND liveness <> 0;
```

Q15. Calculate the cummulative sum of likes for tracks ordered by the number of views, using windows function.

```
SELECT
        track,
        views,
        likes,
        SUM(likes) OVER(ORDER BY views DESC) AS cummulative_likes
FROM spotify
GROUP BY 1,2,3;
```
