Spotify_Project_Solution


```sql
create database spotify_1;
use spotify_1;

 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 BIT,
     official_video BIT,
     stream BIGINT,
     energy_liveness FLOAT,
     most_played_on VARCHAR(50)
 );


 delete from spotify
 where artist IS NULL or
     track IS NULL or
     album IS NULL or
     album_type IS NULL or
     danceability IS NULL or
     energy IS NULL or
     loudness IS NULL or
     speechiness IS NULL or
     acousticness IS NULL or
     instrumentalness IS NULL or
     liveness  IS  NULL  or
     valence IS NULL or
     tempo IS NULL or
     duration_min IS NULL or
     title IS NULL or
     channel IS NULL or
     views IS NULL or
     likes IS NULL or
     comments IS NULL or
     licensed IS NULL or
```

```
    official_video IS NULL or
    stream IS NULL or
    EnergyLiveness IS NULL or
    most_playedon IS NULL;

UPDATE spotify
SET Licensed = 'True'
WHERE  Licensed = 1;

UPDATE spotify
SET Licensed = 'False'
WHERE  Licensed = 0;



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  duration_min 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 most_playedon from spotify;
```

Spotify_Project_Solution


```
---- Data analysis - easy category


-- Q1.Retrieve the names of all tracks that have more than 1 billion streams.

select Track from spotify
where stream > 1000000000;


-- Q2. List all albums along with their respective artists.


select DISTINCT Album,Artist from spotify order by 1;


--- Q3. Get the total number of comments for tracks where licensed = TRUE.

select Count(Comments) from spotify
where Licensed = 1;


--- Q4. Find all tracks that belong to the album type single.

select * from spotify
WHERE Album_type = 'single';


--- Q5. Count the total number of tracks by each artist.
 select
 artist,
 count(*) as total_no_songs
 from spotify
 group by artist
 order by 2 ;




  --- Medium Level


   ---Q1. Calculate the average danceability of tracks in each album.

select
    Album,
   avg(Danceability) as avg_Danceability
   from spotify
   GROUP BY Album
   ORDER BY 2 DESC;
```

```sql
   ---Q2. Find the top 5 tracks with the highest energy values.
SELECT  TOP 5
    Track,
    MAX(EnergyLiveness) AS EnergyLiveness
FROM
    spotify
GROUP BY
    Track
ORDER BY
    EnergyLiveness DESC;

   ---Q3. List all tracks along with their views and likes where official_video =
     TRUE.

   select *from spotify;

   SELECT
    Track,
    SUM(Views) AS Total_Views,
    SUM(Likes) AS Total_Likes
FROM
    spotify
where official_video = 1
GROUP BY
    Track
ORDER BY
    2 DESC;


   ---Q4. For each album, calculate the total views of all associated tracks.

   select
   Album,
   Track,
   sum(Views) as Total_Views
   from spotify
   GROUP BY Album , Track
   ORDER BY 3 desc;

   ---Q5. Retrieve the track names that have been streamed on Spotify more than
     YouTube.

select * from
(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
```

```
    Track) as t1
    where Streamed_on_spotify> Streamed_on_youtube
    AND
    Streamed_on_youtube<>0;

--- Advanced Level

--- Q1. Find the top 3 most-viewed tracks for each artist using window functions.
--- each artist and total views for each track
--- track with highest view for each artist(we need top)
--- dense rank
--- cte and filder rank <=3


WITH rank_artist 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
    rank_artist
WHERE
    Rank <= 3
ORDER BY
    Artist, Total_Views DESC;


---Q2. Write a query to find tracks where the liveness score is above the average.

select * from spotify;

select
     Track,
     Artist,
     Liveness
from spotify
where liveness>(select avg(Liveness) from spotify);


--- Q3. Use a WITH clause to calculate the difference between the highest and lowest
  energy values for tracks in each album.

With cte
AS
```

```sql
(select
        Album,
        MAX(Energy) as Highest_Energy,
        MIN(Energy) as Lowest_Energy
from spotify
Group by Album
)
Select
        Album,
        Highest_Energy,
        Lowest_Energy,
        (Highest_Energy-Lowest_Energy) as energy_difference
from cte;



---Q4. Find tracks where the energy-to-liveness ratio is greater than 1.2.

SELECT
    Track,
    Energy,
    Liveness,
    (Energy / Liveness) AS EnergyToLivenessRatio
FROM
    spotify
WHERE
    (Energy / Liveness) > 1.2;


--- Q5.Calculate the cumulative sum of likes for tracks ordered by the number of views, using window functions.

SELECT
    Track,
    Views,
    Likes,
    SUM(Likes) OVER (ORDER BY Views) AS CumulativeLikes
FROM
    spotify
ORDER BY
    Views;
```