

**DEEZER – CASE STUDY**

**PART 1: SQL**

1. Write a single SQL query that provides the name of the top 10 artists per country. Note: You may decide which metric to use to rank the artists. Expected outcome:

|  |  |  |
| --- | --- | --- |
| country | ranking | artist\_name |
|  |  |  |

*/\*criteria for ranking by stream count and total listening time\*/*

SELECT country, rank, artist\_name

FROM (

SELECT artist\_id, country, COUNT(\*) as stream\_count, SUM(listening\_time) as stream\_time, RANK() OVER(PARTITION BY country ORDER BY COUNT(\*) DESC, SUM(listening\_time) DESC) as rank

FROM streams

GROUP BY artist\_id, country

) X

JOIN artists a

ON X.artist\_id = a.artist\_id

WHERE rank <= 10;

1. Write a *single* SQL query that provides the average number of streams per user and offer type. Expected outcome:

|  |  |
| --- | --- |
| offer\_type | average\_streams\_per\_user |
|  |  |

SELECT o.offer\_type, ROUND(AVG(stream\_count),2) AS average\_streams\_per\_user

FROM users u

JOIN offers o

ON u.offer\_id = o.offer\_id

JOIN (

SELECT user\_id, COUNT(\*) AS stream\_count

FROM streams s

GROUP BY user\_id

) X

ON u.user\_id = X.user\_id

GROUP BY 1;

1. Compute how many artists represent 80% of all streams made by users between 18 and 24 years old.

*/\* calculating the user streams where users are in the age range 18-24\*/*

WITH user\_streams AS

(

SELECT s.artist\_id, COUNT(\*) AS stream\_count

FROM Streams s

JOIN Users u ON s.user\_id = u.user\_id

WHERE u.user\_age BETWEEN 18 AND 24

GROUP BY s.artist\_id

),

*/\* calculating the total streams\*/*

total\_streams AS

(

SELECT SUM(stream\_count) AS total\_count

FROM user\_streams

)

*/\*calculating the cumulative stream count in order of descending streams to find the artists representing 80% of the streams\*/*

SELECT COUNT(\*) AS artist\_count

FROM (

SELECT artist\_id, stream\_count, SUM(stream\_count) OVER (ORDER BY stream\_count DESC) AS cumulative\_count, total\_count

FROM user\_streams, total\_streams

) X

WHERE cumulative\_count <= 0.8 \* total\_count;

1. A fan is defined as a user who does more than 30 streams on at least 10 different days and 5 distinct tracks on a single artist. Based on this definition, compute the percentage of the audience which qualifies as being fans.

*/\*CTE for criteria 1 for total streams more than 30 and stream days more than 10\*/*

WITH c1 AS

(SELECT user\_id, COUNT(DISTINCT d) as stream\_days, COUNT(\*) as total\_streams

FROM streams

GROUP BY user\_id

HAVING total\_streams >=30 AND stream\_days >=10

),

*/\* criteria 2 for distinct tracks more than 5 on a single artist \*/*

c2 AS

(SELECT user\_id, artist\_id, COUNT(DISTINCT song\_id) as distinct\_track

FROM streams

GROUP BY 1,2

HAVING distinct\_track >=5

)

SELECT (COUNT(DISTINCT c1.user\_id)\*100/(SELECT COUNT(DISTINCT s.user\_id)FROM streams s)) as fan\_percentage

FROM c1

JOIN c2

ON c1.user\_id = c2.user\_id;

1. (Bonus) We define a listening session as all consecutive streams made by a single user within 5 minutes of each other. Compute the average duration of a session for each age bracket.

NB: Age brackets: -18, 18-24, 25-34, 35-49, 50+



*/\*calculating the end of streaming time by adding the listening time to the timestamp at which the stream started\*/*

WITH stream\_time AS

(SELECT user\_id, ts\_listen, datetime(strftime('%s', substr(ts\_listen, 1, 19))+listening\_time, 'unixepoch') AS endtime

FROM streams

ORDER BY 2 DESC),

*/\*creating the column to have the endtime of previouse session next to the start time of the current stream\*/*

stream\_diff AS

(SELECT user\_id, ts\_listen, endtime, LAG(endtime, 1) OVER(PARTITION BY user\_id ORDER BY endtime DESC) as endtime\_2

FROM stream\_time),

*/\*calculating the difference between the end of previous stream time and the start of next stream time and filtering the one within 5 minutes of each other\*/*

stream\_session AS

(SELECT (strftime('%s', substr(endtime\_2,1,19)) - strftime('%s', substr(ts\_listen,1,19)))/60 AS stream\_t, user\_id

FROM stream\_diff

WHERE stream\_t <=5),

*/\*creating age brackets and calculating the duration of stream session\*/*

bracket AS (SELECT s.user\_id, listening\_time,

CASE WHEN stream\_t <=5 THEN listening\_time ELSE NULL

END AS session\_duration,

CASE

WHEN u.user\_age < 18 THEN '-18'

WHEN u.user\_age BETWEEN 18 AND 24 THEN '18-24'

WHEN u.user\_age BETWEEN 25 AND 34 THEN '25-34'

WHEN u.user\_age BETWEEN 35 AND 49 THEN '35-49'

ELSE '50+'

END AS age\_bracket

FROM stream\_session ss

JOIN users u

ON ss.user\_id = u.user\_id

JOIN streams s

ON s.user\_id = ss.user\_id)

*/\*calculating the avg session duration for each bracket\*/*

SELECT AVG(session\_duration) as avg\_session\_duration, age\_bracket

FROM bracket

GROUP BY age\_bracket;

Another method, where instead of calculating the end time of a stream session, I used LAG function to subtract the difference between the previous stream and check if it is under 5 minutes.

SELECT age\_bracket, AVG(session\_duration) AS average\_duration

FROM (

SELECT

CASE

WHEN u.user\_age < 18 THEN '-18'

WHEN u.user\_age BETWEEN 18 AND 24 THEN '18-24'

WHEN u.user\_age BETWEEN 25 AND 34 THEN '25-34'

WHEN u.user\_age BETWEEN 35 AND 49 THEN '35-49'

ELSE '50+'

END AS age\_bracket,

session\_duration

FROM (

SELECT

user\_id,

ts\_listen,

LAG(ts\_listen) OVER (PARTITION BY user\_id ORDER BY ts\_listen) AS prev\_ts\_listen,

ts\_listen - LAG(ts\_listen) OVER (PARTITION BY user\_id ORDER BY ts\_listen) AS time\_diff,

listening\_time,

CASE

WHEN ts\_listen - LAG(ts\_listen) OVER (PARTITION BY user\_id ORDER BY ts\_listen) <= 300 THEN NULL

ELSE listening\_time

END AS session\_duration

FROM Streams

) sessions

JOIN Users u ON sessions.user\_id = u.user\_id

WHERE session\_duration IS NOT NULL

) session\_summary

GROUP BY age\_bracket;