**WEEK 2 SQL CHALLENGE – LEVEL -Medium**

-- write a query to find the number of days between each user’s first post of the year and last post of the year in the year 2021. Output the user and number of the days between each user's first and last post.

SELECT user\_id,

MAX(post\_date::DATE) - MIN(post\_date::DATE) AS days\_between

FROM posts

WHERE DATE\_PART('year', post\_date::DATE) = 2021

GROUP BY user\_id

HAVING COUNT(post\_id)>1;

-- Output the tweet count per user as the bucket and the number of Twitter users who fall into that bucket.

SELECT tweet\_count\_per\_user AS tweet\_bucket,

COUNT(user\_id) AS users\_num

FROM (SELECT user\_id,

COUNT(tweet\_id) AS tweet\_count\_per\_user

FROM tweets

WHERE tweet\_date BETWEEN '2022-01-01'

AND '2022-12-31'

GROUP BY user\_id) AS total\_tweets

GROUP BY tweet\_count\_per\_user;

-- Write a query to list the candidates who possess all of the required skills for the job. Sort the output by candidate ID in ascending order.

SELECT candidate\_id

FROM candidates

where skill IN ('Python','Tableau','PostgreSQL')

GROUP BY candidate\_id

HAVING COUNT(skill) = 3

ORDER BY candidate\_id ASC

-- Write a query to return the IDs of the Facebook pages that have zero likes. The output should be sorted in ascending order based on the page IDs.

SELECT pages.page\_id

FROM pages

LEFT JOIN page\_likes

on pages.page\_id = page\_likes.page\_id

WHERE page\_likes.page\_id is NULL

ORDER BY pages.page\_id

-- Write a query to determine which parts have begun the assembly process but are not yet finished.

SELECT part,assembly\_step

FROM parts\_assembly

WHERE finish\_date is NULL

-- Write a query that calculates the total viewership for laptops and mobile devices where mobile is defined as the sum of tablet and phone viewership. Output the total viewership for laptops as laptop\_reviews and the total viewership for mobile devices as mobile\_views.

SELECT SUM(CASE WHEN device\_type = 'laptop' THEN 1 ELSE 0 END) AS laptop\_views,

SUM(CASE WHEN device\_type IN ('tablet', 'phone') THEN 1 ELSE 0 END) AS mobile\_views

FROM viewership;

**WEEK 2 SQL CHALLENGE – LEVEL -Medium**

--Output the user and number of the days between each user's first and last post.

SELECT user\_id,

MAX(post\_date::DATE) - MIN(post\_date::DATE) AS days\_between

FROM posts

WHERE DATE\_PART('year', post\_date::DATE) = 2021

GROUP BY user\_id

HAVING COUNT(post\_id)>1;

-- Write a query to identify the top 2 Power Users who sent the highest number of messages on Microsoft Teams in August 2022. Display the IDs of these 2 users along with the total number of messages they sent. Output the results in descending order based on the count of the messages.

SELECT sender\_id,COUNT(sender\_id) as message\_count

FROM messages

where EXTRACT(MONTH FROM sent\_date) = '8'

AND EXTRACT(YEAR FROM sent\_date) = '2022'

GROUP BY sender\_id

ORDER BY message\_count DESC

LIMIT 2

-- Write a query to retrieve the count of companies that have posted duplicate job listings.

SELECT COUNT(DISTINCT company\_id) AS duplicate\_companies

FROM (

SELECT company\_id,title,description,

COUNT(job\_id) AS job\_count

FROM job\_listings

GROUP BY company\_id, title, description

)AS job\_count\_cte

WHERE job\_count > 1;

-- Write a query to retrieve the top three cities that have the highest number of completed trade orders listed in descending order. Output the city name and the corresponding number of completed trade orders.

SELECT U.city,COUNT(status) as total\_orders

FROM trades T

Inner Join users U

on T.user\_id = U.user\_id

where T.status = 'Completed'

GROUP BY U.city

ORDER BY total\_orders DESC

LIMIT 3

-- write a query to retrieve the average star rating for each product, grouped by month. The output should display the month as a numerical value, product ID, and average star rating rounded to two decimal places. Sort the output first by month and then by product ID.

SELECT EXTRACT(MONTH FROM submit\_date) as mth,product\_id,

ROUND(AVG(stars),2) as avg\_stars

FROM reviews

GROUP BY mth,product\_id

ORDER BY mth,product\_id

**WEEK 2 SQL CHALLENGE – LEVEL -Medium**

-- Write a query to calculate the click-through rate (CTR) for the app in 2022 and round the results to 2 decimal places.

SELECT app\_id,

ROUND(100.0 \* SUM(CASE WHEN event\_type = 'click' THEN 1 ELSE 0 END) /

SUM(CASE WHEN event\_type = 'impression' THEN 1 ELSE 0 END), 2) AS ctr\_rate

FROM events

WHERE timestamp >= '2022-01-01' AND timestamp < '2023-01-01'

GROUP BY app\_id;

-- Write a query to display the user IDs of those who did not confirm their sign-up on the first day, but confirmed on the second day.

SELECT DISTINCT user\_id

FROM emails

INNER JOIN texts

ON emails.email\_id = texts.email\_id

WHERE texts.action\_date = emails.signup\_date + INTERVAL '1 day'

AND texts.signup\_action = 'Confirmed';

-- Write a query that outputs the name of each credit card and the difference in the number of issued cards between the month with the highest issuance cards and the lowest issuance. Arrange the results based on the largest disparity.

SELECT card\_name,

(MAX(issued\_amount) - MIN(issued\_amount)) as difference

FROM monthly\_cards\_issued

GROUP BY card\_name

ORDER BY difference DESC

-- You're trying to find the mean number of items per order on Alibaba, rounded to 1 decimal place using tables which includes information on the count of items in each order (item\_count table) and the corresponding number of orders for each item count (order\_occurrences table).

SELECT ROUND(SUM(item\_count::DECIMAL\*order\_occurrences)/SUM(order\_occurrences),1) AS mean

FROM items\_per\_order;

-- Write a query to find the top 3 most profitable drugs sold, and how much profit they made. Assume that there are no ties in the profits. Display the result from the highest to the lowest total profit.

SELECT drug,SUM(total\_sales - cogs) as profit

FROM pharmacy\_sales

GROUP BY drug

ORDER BY profit desc

LIMIT 3

-- Write a query to identify the manufacturers associated with the drugs that resulted in losses for CVS Health and calculate the total amount of losses incurred.

SELECT manufacturer,

COUNT(drug) AS NO\_OF\_drugs,abs(SUM(total\_sales - cogs)) as Total\_losses

FROM pharmacy\_sales

WHERE cogs > total\_sales

GROUP BY manufacturer

ORDER BY Total\_losses DESC

**WEEK 2 SQL CHALLENGE – LEVEL -Medium**

-- Write a query to calculate the total drug sales for each manufacturer. Round the answer to the nearest million and report your results in descending order of total sales. In case of any duplicates, sort them alphabetically by the manufacturer name.

SELECT manufacturer,

CONCAT( '$', ROUND(SUM(total\_sales) / 1000000), ' million') AS sales\_mil

FROM pharmacy\_sales

GROUP BY manufacturer

ORDER BY SUM(total\_sales) DESC, manufacturer;

-- Write a query to obtain the third transaction of every user. Output the user id, spend and transaction date.

SELECT user\_id,spend,transaction\_date

FROM (

SELECT user\_id,spend,transaction\_date ,

ROW\_NUMBER () over(PARTITION BY user\_id ORDER BY transaction\_date)

as Row\_num FROM transactions) as trans\_num

WHERE Row\_num = 3

--Write a query to obtain a breakdown of the time spent sending vs. opening snaps as a percentage of total time spent on these activities grouped by age group. Round the percentage to 2 decimal places in the output.

SELECT B.age\_bucket,

Round((SUM(a.time\_spent) FILTER (WHERE A.activity\_type = 'send')/SUM(a.time\_spent)) \* 100,2) as SEND\_Perc,

Round((SUM(a.time\_spent) FILTER (WHERE A.activity\_type = 'open')/SUM(a.time\_spent)) \* 100,2) as open\_Perc

FROM activities A

Inner join age\_breakdown B

on A.user\_id = B.user\_id

WHERE A.activity\_type in ('send','open')

GROUP BY B.age\_bucket

-- Given a table of tweet data over a specified time period, calculate the 3-day rolling average of tweets for each user. Output the user ID, tweet date, and rolling averages rounded to 2 decimal places.

SELECT user\_id,tweet\_date,

ROUND(AVG(tweet\_count) OVER (

PARTITION BY user\_id

ORDER BY tweet\_date

ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)

,2) AS rolling\_avg\_3d

FROM tweets;

-- write a query to identify the top two highest-grossing products within each category in the year 2022. The output should include the category, product, and total spend.

SELECT category,product,total\_spend

FROM (

SELECT category,product,SUM(spend) as total\_spend,

RANK() over(PARTITION BY category ORDER BY SUM(spend) DESC) as ranking

FROM product\_spend

WHERE EXTRACT(year FROM transaction\_date) = 2022

GROUP BY category,product) as ranked\_spending

WHERE ranking <=2

ORDER BY category ,ranking

**WEEK 2 SQL CHALLENGE – LEVEL -Medium**

-- Write a query to find the top 5 artists whose songs appear most frequently in the Top 10 of the global\_song\_rank table. Display the top 5 artist names in ascending order, along with their song appearance ranking.

WITH TOP\_10

AS (

SELECT a.artist\_name,

DENSE\_RANK() over(ORDER BY COUNT(s.song\_id)DESC) AS ARTIST\_RANK

FROM artists a

inner join songs s on a.artist\_id = s.artist\_id

inner join global\_song\_rank g on s.song\_id = g.song\_id

where g.rank <=10

GROUP BY a.artist\_name

)

SELECT artist\_name,ARTIST\_RANK

FROM TOP\_10

WHERE ARTIST\_RANK <=5

-- Write a query to find the activation rate. Round the percentage to 2 decimal places.

SELECT

ROUND(COUNT(t.email\_id)::DECIMAL/COUNT(DISTINCT e.email\_id),2) as activation\_rate

FROM emails E

LEFT JOIN texts T on e.email\_id = t.email\_id

and t.signup\_action = 'Confirmed'

-- Write a query that effectively identifies the company ID of such Supercloud customers.

WITH supercloud AS (

SELECT customers.customer\_id,

COUNT(DISTINCT products.product\_category) as unique\_count

FROM customer\_contracts AS customers

LEFT JOIN products ON customers.product\_id = products.product\_id

GROUP BY customers.customer\_id

)

SELECT customer\_id

FROM supercloud

WHERE unique\_count = (

SELECT COUNT(DISTINCT product\_category)

FROM products)

ORDER BY customer\_id;

-- Write a query to calculate the sum of odd-numbered and even-numbered measurements separately for a particular day and display the results in two different columns. Refer to the Example Output below for the desired format.

WITH ranked\_measurements AS (

SELECT CAST(measurement\_time AS DATE) AS measurement\_day,

measurement\_value,

ROW\_NUMBER() OVER (PARTITION BY CAST(measurement\_time AS DATE)

ORDER BY measurement\_time) AS measurement\_num

FROM measurements

)

SELECT measurement\_day,

SUM(measurement\_value) FILTER (WHERE measurement\_num % 2 != 0) AS odd\_sum,

SUM(measurement\_value) FILTER (WHERE measurement\_num % 2 = 0) AS even\_sum

FROM ranked\_measurements

GROUP BY measurement\_day;

**WEEK 2 SQL CHALLENGE – LEVEL -Medium**

-- Write a query to retrieve the mode of the order occurrences. Additionally, if there are multiple item counts with the same mode, the results should be sorted in ascending order.

SELECT item\_count AS mode

FROM items\_per\_order

WHERE order\_occurrences = (SELECT MODE() WITHIN GROUP (ORDER BY order\_occurrences DESC)

FROM items\_per\_order

)

ORDER BY item\_count;

--Write a query that outputs the name of the credit card, and how many cards were issued in its launch month. The launch month is the earliest record in the monthly\_cards\_issued table for a given card. Order the results starting from the biggest issued amount.

WITH card\_launch AS (

SELECT card\_name,issued\_amount,

MAKE\_DATE(issue\_year, issue\_month, 1) AS issue\_date,

MIN(MAKE\_DATE(issue\_year, issue\_month, 1))

OVER (PARTITION BY card\_name) AS launch\_date

FROM monthly\_cards\_issued

)

SELECT card\_name, issued\_amount

FROM card\_launch

WHERE issue\_date = launch\_date

ORDER BY issued\_amount DESC;

-- Find the 3 most profitable companies in the entire world.Output the result along with the corresponding company name.Sort the result based on profits in descending order.

select top 3 company,profits

from forbes\_global\_2010\_2014

order by profits DESC

-- You have been asked to find the job titles of the highest-paid employees.Your output should include the highest-paid title or multiple titles with the same salary.

select worker\_title AS best\_paid\_title

from worker w

join title t on w.worker\_id= t.worker\_ref\_id

where salary = (select max(salary) from worker)