**Query : 29**

SELECT candidate\_id

FROM candidates

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

GROUP BY candidate\_id

HAVING count(\*) = 3;

**Query : 30**

SELECT page\_id

FROM pages

WHERE page\_id NOT IN (

SELECT page\_id

FROM page\_likes

)

ORDER BY page\_id;

**Query : 31**

SELECT

part, assembly\_step

FROM parts\_assembly

WHERE finish\_date IS NULL;

**Query : 32**

WITH cte AS(SELECT user\_id, COUNT(\*)

FROM tweets

WHERE EXTRACT(YEAR FROM tweet\_date) = 2022

GROUP BY user\_id)

SELECT count, COUNT(\*)

FROM cte

GROUP BY count;

**Query : 33**

SELECT

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

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

FROM viewership;

**Query : 34**

SELECT COUNT(DISTINCT t1.company\_id) AS co\_w\_duplicate\_jobs FROM job\_listings t1

JOIN job\_listings t2

ON t1.company\_id = t2.company\_id AND t1.title = t2.title AND

t1.description = t2.description AND t1.job\_id <> t2.job\_id;

**Query 35:**

SELECT

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

FROM posts

WHERE EXTRACT(YEAR FROM post\_date::DATE) = 2021

GROUP BY user\_id

HAVING COUNT(user\_id) > 1

**Query : 36**

SELECT

sender\_id, COUNT(\*) AS message\_count

FROM messages

WHERE EXTRACT(YEAR FROM sent\_date) = 2022 AND EXTRACT(MONTH FROM sent\_date) = 8

GROUP BY sender\_id

ORDER BY message\_count DESC

LIMIT 2;

**Query : 37**

SELECT city, COUNT(\*) AS total\_orders

FROM trades t

JOIN users u

ON u.user\_id = t.user\_id

WHERE t.status = 'Completed'

GROUP BY city

ORDER BY total\_orders DESC, city

LIMIT 3;

**Query : 38**

SELECT

EXTRACT(MONTH FROM submit\_date) AS mth, product\_id AS product, ROUND(AVG(stars), 2) AS avg\_stars

FROM reviews

GROUP BY EXTRACT(MONTH FROM submit\_date), product\_id

ORDER BY mth, product;

**Query : 39**

SELECT

app\_id,

ROUND(100.0\*SUM(CASE WHEN event\_type = 'click' THEN 1END)/SUM(CASE WHEN event\_type = 'impression' THEN 1 END), 2) AS ctr

FROM events

WHERE timestamp >= '2022-01-01'

AND timestamp < '2023-01-01'

GROUP BY app\_id;

**Query : 40**

SELECT user\_id

FROM emails e

WHERE email\_id IN (

SELECT email\_id

FROM texts t

WHERE t.signup\_action = 'Confirmed'

AND EXTRACT(DAY FROM t.action\_date-e.signup\_date) = 1)

**Query : 41**

SELECT

card\_name, MAX(issued\_amount) - MIN(issued\_amount) AS difference

FROM monthly\_cards\_issued

GROUP BY card\_name

ORDER BY difference DESC;

**Query : 42**

SELECT

ROUND(

SUM(item\_count\*order\_occurrences)\*1.0/SUM(order\_occurrences)

, 1)

FROM items\_per\_order;

**Query : 43**

SELECT

drug, total\_sales - cogs total\_profit

FROM pharmacy\_sales

ORDER BY total\_profit DESC

LIMIT 3;

**Query : 44**

SELECT

manufacturer, COUNT(\*) drug\_count, SUM(cogs-total\_sales) AS total\_loss

FROM pharmacy\_sales

WHERE cogs > total\_sales

GROUP BY manufacturer

ORDER BY total\_loss DESC;

**Query : 45**

SELECT

manufacturer, CONCAT('$', ROUND(SUM(total\_sales)/1000000,0), ' million') AS sales\_mil

FROM pharmacy\_sales

GROUP BY manufacturer

ORDER BY ROUND(SUM(total\_sales)/1000000,0) DESC, manufacturer DESC;

**Query : 46**

SELECT COUNT(\*)

FROM (

SELECT policy\_holder\_id

FROM callers

GROUP BY policy\_holder\_id

HAVING COUNT(\*) >= 3

) a;

**Query : 47**

SELECT

ROUND(SUM(CASE WHEN call\_category IS NULL OR call\_category = 'n/a' THEN 1 ELSE 0 END)\*100.0/COUNT(\*), 1) call\_percentage

FROM callers;

**Query : 48**

WITH cte AS(

SELECT

user\_id, spend, transaction\_date,

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

FROM transactions

)

SELECT user\_id, spend, transaction\_date

FROM cte

WHERE row\_number = 3;

**Query : 49**

WITH cte AS(

SELECT

b.age\_bucket,

SUM(CASE WHEN activity\_type = 'send' THEN time\_spent ELSE 0 END) AS send\_perc,

SUM(CASE WHEN activity\_type = 'open' THEN time\_spent ELSE 0 END) AS open\_perc,

SUM(CASE WHEN activity\_type <> 'chat' THEN time\_spent ELSE 0 END) AS total

FROM activities a

JOIN age\_breakdown b

ON a.user\_id = b.user\_id

GROUP BY age\_bucket

)

SELECT

age\_bucket,

ROUND(send\_perc \* 100.0 / total, 2) AS send\_perc,

ROUND(open\_perc \* 100.0 /total, 2) AS open\_perc

FROM cte;

**Query : 50**

SELECT

user\_id, tweet\_date,

ROUND(AVG(tweet\_count) OVER(PARTITION BY user\_id ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),2) rolling\_avg\_3d

FROM tweets;

**Query : 51**

with cte as(

SELECT

category,

product,

sum(spend) total\_spent

FROM

product\_spend

where

EXTRACT(

year

from

transaction\_date

) = '2022'

group by

category,

product

),

cte2 as (

select

\*,

rank() over(PARTITION BY category

order by

total\_spent desc

) rnk

from

cte

)

select

category,

product,

total\_spent

from

cte2

where

rnk <= 2;

**Query : 52**

WITH cte1 AS(

SELECT a.artist\_name, COUNT(\*)

FROM global\_song\_rank g

JOIN songs s

ON g.song\_id = s.song\_id

JOIN artists a

ON a.artist\_id = s.artist\_id

WHERE rank <= 10

GROUP BY a.artist\_name

ORDER BY count DESC

),

cte2 AS(

SELECT artist\_name, count,

DENSE\_RANK() OVER(ORDER BY count DESC) rnk

FROM cte1

)

SELECT artist\_name, rnk AS artist\_rank

FROM cte2

WHERE rnk <= 5;

**Query : 53**

SELECT ROUND(COUNT(\*)\*1.0 / (SELECT COUNT(\*) FROM emails), 2) AS confirm\_rate

FROM texts

WHERE signup\_action = 'Confirmed';

**Query : 54**

SELECT customer\_id FROM customer\_contracts cc

JOIN products p

on cc.product\_id = p.product\_id

GROUP BY customer\_id

HAVING(COUNT(DISTINCT p.product\_category) = 3) ;

**Query : 55**

WITH cte1 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(CASE WHEN measurement\_num % 2 != 0 THEN measurement\_value ELSE 0 END) AS odd\_sum,

SUM(CASE WHEN measurement\_num % 2 = 0 THEN measurement\_value ELSE 0 END) AS even\_sum

FROM cte1

GROUP BY measurement\_day;

**Query : 56**

SELECT

MAX(transaction\_date) AS transaction\_date, user\_id, COUNT(\*) AS purchase\_count

FROM user\_transactions ut

WHERE transaction\_date =

(SELECT max(transaction\_date) FROM user\_transactions WHERE user\_id = ut.user\_id)

GROUP BY user\_id

ORDER BY transaction\_date;

**Query : 57**

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)

**Query : 58**

WITH cte AS (SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY card\_name ORDER BY issue\_year, issue\_month)

FROM

monthly\_cards\_issued

)

SELECT

card\_name, issued\_amount

FROM

cte

WHERE row\_number = 1

ORDER BY issued\_amount DESC

**Query : 59**

SELECT

ROUND(COUNT(\*)\*100.0/(SELECT COUNT(\*) FROM phone\_calls), 1)

FROM

phone\_calls p

JOIN phone\_info i1

ON p.caller\_id = i1.caller\_id

JOIN phone\_info i2

ON p.receiver\_id = i2.caller\_id

WHERE i1.country\_id <> i2.country\_id

**Query : 60**

SELECT

EXTRACT(MONTH FROM event\_date) AS month,

COUNT(DISTINCT user\_id) AS monthly\_active\_users

FROM

user\_actions t1

WHERE user\_id IN

(

SELECT user\_id

FROM user\_actions

WHERE user\_id = t1.user\_id

AND EXTRACT(MONTH FROM event\_date) =

EXTRACT(MONTH FROM t1.event\_date - interval '1 month')

)

AND EXTRACT(MONTH FROM event\_date) = 7

AND EXTRACT(YEAR FROM event\_date) = 2022

GROUP BY EXTRACT(MONTH FROM event\_date);

**Query : 61**

WITH cte AS (

SELECT

EXTRACT(YEAR FROM transaction\_date) AS year,

product\_id,

SUM(spend) AS curr\_year\_spend

FROM

user\_transactions

GROUP BY EXTRACT(YEAR FROM transaction\_date), product\_id)

SELECT

year,

product\_id,

curr\_year\_spend,

LAG(curr\_year\_spend) OVER(ORDER BY year) AS prev\_year\_spend,

(curr\_year\_spend - LAG(curr\_year\_spend) OVER(ORDER BY year))\*100.0/curr\_year\_spend AS yoy\_rate

FROM cte

**Query : 62**

WITH cte AS (  
    SELECT item\_type, SUM(square\_footage) AS total\_sqft,  
        COUNT(\*) AS total\_items  
      FROM inventory  
      GROUP BY item\_type  
      ),  
cte\_2 AS (  
     SELECT item\_type, total\_sqft,  
            FLOOR(500000/total\_sqft) AS prime\_item\_combination\_occurence,  
            FLOOR(500000/total\_sqft) \* total\_items AS no\_of\_prime\_items  
      FROM cte  
      WHERE item\_type = 'prime\_eligible'  
      )  
SELECT item\_type,  
      CASE  
            WHEN item\_type = 'prime\_eligible'  
                  THEN (FLOOR(500000/total\_sqft) \* total\_items)  
            WHEN item\_type = 'not\_prime'  
                  THEN FLOOR((500000 - (SELECT FLOOR(500000/total\_sqft) \* total\_sqft FROM cte\_2)) / total\_sqft) \* total\_items  
      END AS total\_items  
FROM cte  
ORDER BY item\_type DESC;

**Query : 63**

WITH RECURSIVE cte AS(

SELECT

searches, num\_users,1 ctn

FROM search\_frequency

UNION

SELECT

searches, num\_users-1,ctn+1 AS num\_users

FROM

cte

WHERE ctn < num\_users

)

SELECT PERCENTILE\_CONT(0.5) WITHIN GROUP(ORDER BY searches) AS median FROM cte;

**Query : 64**

SELECT user\_id,

CASE

WHEN paid is NULL THEN 'CHURN'

WHEN paid is not NULL AND status in ('NEW','EXISTING','RESURRECT') THEN 'EXISTING'

WHEN paid is not NULL AND status = 'CHURN' THEN 'RESURRECT'

WHEN paid is not NULL AND status is NULL THEN 'NEW'

END AS new\_status

FROM advertiser a

FULL OUTER JOIN daily\_pay dp using(user\_id)

ORDER BY user\_id

**Query : 65**

SELECT

CONCAT(p1.topping\_name, ',', p2.topping\_name, ',', p3.topping\_name) AS pizza,

p1.ingredient\_cost + p2.ingredient\_cost + p3.ingredient\_cost AS total\_cost

FROM pizza\_toppings p1

CROSS JOIN pizza\_toppings p2

CROSS JOIN pizza\_toppings p3

WHERE p1.topping\_name < p2.topping\_name AND p2.topping\_name < p3.topping\_name

ORDER BY total\_cost DESC, pizza;

**Query : 66**

SELECT COUNT(\*)

FROM (

SELECT policy\_holder\_id

FROM callers

GROUP BY policy\_holder\_id

HAVING COUNT(\*) >= 3

) a;

**Query : 67**

WITH cte AS

(SELECT

policy\_holder\_id, case\_id, call\_received,

COUNT(policy\_holder\_id)

OVER(PARTITION BY policy\_holder\_id ORDER BY call\_received RANGE INTERVAL '7 DAYS' PRECEDING)

FROM callers)

SELECT COUNT(DISTINCT policy\_holder\_id) AS patient\_count

FROM cte

WHERE count = 2;

**Query : 68**

WITH cte AS

(SELECT

merchant\_id,

COUNT(transaction\_id)

OVER(PARTITION BY credit\_card\_id

ORDER BY transaction\_timestamp RANGE INTERVAL '10 MINUTE' PRECEDING)

FROM transactions)

SELECT COUNT(DISTINCT merchant\_id) AS payment\_count

FROM cte

WHERE count = 2;

**Query : 69**

WITH cte AS (

SELECT id, DENSE\_RANK() OVER(ORDER BY salary DESC)

FROM

employee\_salary)

SELECT \*

FROM cte

WGERE dense\_rank = 3;

**Query :70**

SELECT

e.enam, e.empid, m.ename AS manager, e.manager\_id

FROM emp e, emp m

WHERE e.manager\_id = m.empid