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
```