

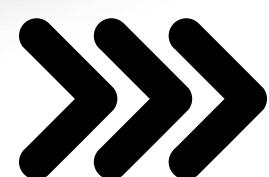


Tajamul Khan

50 SQL Interview Queries



@Tajamulkhan





1. Find duplicate records in a table



```
SELECT column1, column2, COUNT(*)  
FROM your_table  
GROUP BY column1, column2  
HAVING COUNT(*) > 1;
```

@Tajamulkhan



@Tajamulkhan





2. Retrieve the second highest salary from the Employee table



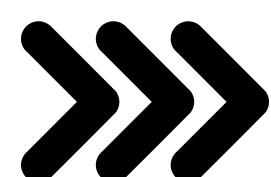
```
SELECT MAX(salary) AS  
SecondHighestSalary  
FROM Employee  
WHERE salary < (SELECT MAX(salary)  
FROM Employee);
```



Microsoft



@Tajamulkhan





3. Find employees without department (Left Join usage)



```
SELECT e.*  
FROM Employee e  
LEFT JOIN Department d  
ON e.department_id =  
d.department_id  
WHERE d.department_id IS NULL;
```

Uber



@Tajamulkhan





4. Calculate the total revenue per product

```
SELECT product_id,  
SUM(quantity * price) AS  
total_revenue  
FROM Sales  
GROUP BY product_id;
```



@Tajamulkhann





5. Get the top 3 highest-paid employees.

```
SELECT TOP 3 *  
FROM Employee  
ORDER BY salary DESC;
```

Google



@Tajamulkhan





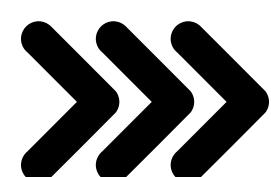
6. Customers who made purchases but never returned products.



```
SELECT DISTINCT c.customer_id  
FROM Customers c  
JOIN Orders o ON c.customer_id =  
o.customer_id  
WHERE c.customer_id NOT IN (  
    SELECT customer_id FROM Returns  
);
```



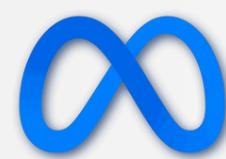
@Tajamulkhann





7. Show the count of orders per customer.

```
SELECT customer_id,  
COUNT(*) AS order_count  
FROM Orders  
GROUP BY customer_id;
```

 Meta



@Tajamulkhann





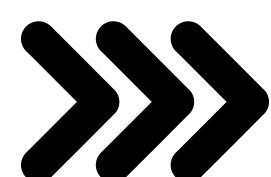
8. Retrieve all employees who joined in 2023.



```
SELECT *
FROM Employee
WHERE YEAR(hire_date) = 2023;
```



@Tajamulkhann





9. Calculate the average order value per customer.



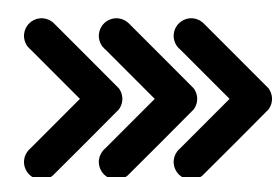
```
SELECT customer_id,  
AVG(total_amount) AS  
avg_order_value  
FROM Orders  
GROUP BY customer_id;
```



Microsoft



@Tajamulkhann





10. Get the latest order placed by each customer.

```
SELECT customer_id,  
MAX(order_date) AS  
latest_order_date  
FROM Orders  
GROUP BY customer_id;
```

Uber



@Tajamulkhann





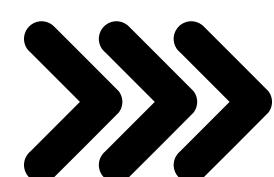
11. Find products that were never sold.



```
SELECT p.product_id  
FROM Products p  
LEFT JOIN Sales s  
ON p.product_id = s.product_id  
WHERE s.product_id IS NULL;
```



@Tajamulkhan





12. Identify the most selling product.



```
SELECT TOP 1 product_id,  
SUM(quantity) AS total_qty  
FROM Sales  
GROUP BY product_id  
ORDER BY total_qty DESC;
```



@Tajamulkhan





13. Get the total revenue and the number of orders per region.



```
SELECT region,  
SUM(total_amount) AS total_revenue,  
COUNT(*) AS order_count  
FROM Orders  
GROUP BY region;
```



@Tajamulkhann





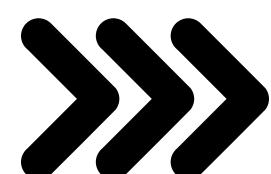
14. Count how many customers placed more than 5 orders.



```
SELECT COUNT(*) AS customer_count
FROM (
    SELECT customer_id FROM Orders
    GROUP BY customer_id
    HAVING COUNT(*) > 5
) AS subquery;
```



@Tajamulkhan



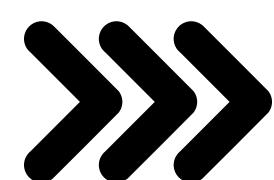


15. Retrieve customers with orders above the average order value.

```
SELECT *
FROM Orders
WHERE total_amount >
(SELECT AVG(total_amount)
FROM Orders);
```



@Tajamulkhan





16. Find all employees hired on weekends.

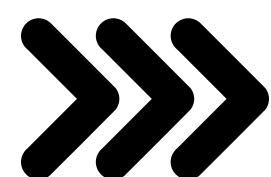


```
SELECT *
FROM Employee
WHERE DATENAME(WEEKDAY, hire_date)
IN ('Saturday', 'Sunday');
```

Google



@Tajamulkhann





17. Find all employees hired on weekends.

```
SELECT *
FROM Employee
WHERE EXTRACT(DOW FROM hire_date)
IN (0, 6);
```



@Tajamulkhann





18. Get monthly sales revenue and order count.

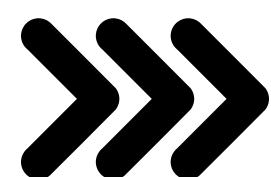


SELECT

```
FORMAT(date, 'yyyy-MM') AS month,  
SUM(amount) AS total_revenue,  
COUNT(order_id) AS order_count  
FROM Orders  
GROUP BY  
FORMAT(date, 'yyyy-MM');
```



@Tajamulkhann





19. Rank employees by salary within each department.



```
SELECT employee_id, department_id,  
salary, RANK() OVER (PARTITION BY  
department_id  
ORDER BY salary DESC) AS salary_rk  
FROM Employee;
```



@Tajamulkhann





20. Find customers who placed orders every month in 2023.



```
SELECT customer_id  
FROM Orders  
WHERE YEAR(order_date) = 2023  
GROUP BY customer_id  
HAVING COUNT(DISTINCT  
FORMAT(order_date, 'yyyy-MM')) = 12
```



@Tajamulkhann





21. Find moving average of sales over the last 3 days.



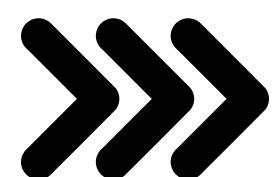
```
SELECT order_date,  
AVG(total_amount) OVER (ORDER BY  
order_date ROWS BETWEEN 2 PRECEDING  
AND CURRENT ROW) AS moving_avg  
FROM Orders;
```



Microsoft



@Tajamulkhan





22. Identify the first and last order date for each customer.



```
SELECT customer_id,  
MIN(order_date) AS first_order,  
MAX(order_date) AS last_order  
FROM Orders  
GROUP BY customer_id;
```

Uber



@Tajamulkhann





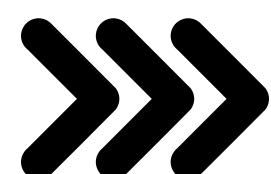
23. Show product sales distribution (percent of total revenue).



```
WITH TotalRevenue AS (
  SELECT
    SUM(quantity * price) AS total FROM Sales)
  SELECT s.product_id,
    SUM(s.quantity * s.price) AS revenue,
    SUM(s.quantity * s.price) * 100/ t.total
    AS revenue_pct
  FROM Sales s
  CROSS JOIN TotalRevenue t
  GROUP BY s.product_id, t.total;
```



@Tajamulkhann





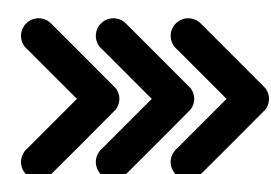
24. Retrieve customers who made consecutive purchases (2 Days)



```
WITH cte AS (
    SELECT id, order_date,
    LAG(order_date) OVER (PARTITION BY id
    ORDER BY order_date) AS prev_order_date
    FROM Orders)
SELECT id, order_date, prev_odate
FROM cte
WHERE
DATEDIFF(DAY, prev_oate, order_date) = 1;
```



@Tajamulkhann





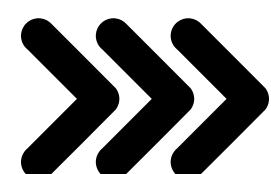
25. Find churned customers (no orders in the last 6 months).



```
SELECT customer_id
FROM Orders
GROUP BY customer_id
HAVING
MAX(order_date) <
DATEADD(MONTH, -6, GETDATE());
```



@Tajamulkhan





26. Calculate cumulative revenue by day.

```
SELECT order_date,  
SUM(total_amount) OVER  
(ORDER BY order_date) AS  
cumulative_revenue  
FROM Orders;
```



@Tajamulkhann





27. Identify top-performing departments by average salary.

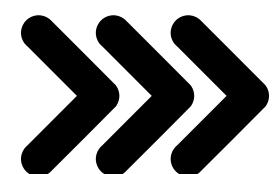


```
SELECT department_id,  
AVG(salary) AS avg_salary  
FROM Employee  
GROUP BY department_id  
ORDER BY avg_salary DESC;
```

Google



@Tajamulkhan





28. Find customers who ordered more than the average number of orders per customer.



```
WITH customer_orders AS (
  SELECT customer_id, COUNT(*) AS order_count
  FROM Orders
  GROUP BY customer_id)
SELECT * FROM customer_orders
WHERE order_count > (SELECT
  AVG(order_count) FROM customer_orders);
```



@Tajamulkhann





29. Calculate revenue generated from new customers (first-time orders).

```
WITH first_orders AS (
    SELECT customer_id, MIN(order_date) AS
        first_order_date FROM Orders
    GROUP BY customer_id)
SELECT SUM(o.total_amount) AS new_revenue
    FROM Orders o JOIN first_orders f
    ON o.customer_id = f.customer_id
    WHERE o.order_date = f.first_order_date;
```





30. Find the percentage of employees in each department.

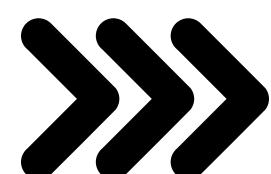


```
SELECT  
    department_id,  
    COUNT(*) AS emp_count,  
    COUNT(*) * 100.0 / (SELECT  
        COUNT(*) FROM Employee)  
    AS pct FROM Employee  
    GROUP BY department_id;
```

Uber



@Tajamulkhan





31. Retrieve the maximum salary difference within each department.

SELECT

```
department_id,  
MAX(salary) - MIN(salary) AS  
salary_diff  
FROM Employee  
GROUP BY department_id;
```



@Tajamulkhann





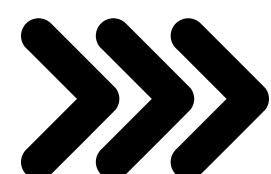
32. Find products that contribute to 80% of the revenue (Pareto Principle).



```
WITH sales_cte AS (
    SELECT product_id, SUM(qty * price) AS revenue
    FROM Sales GROUP BY product_id),
total_revenue AS (
    SELECT SUM(revenue) AS total FROM sales_cte)
SELECT s.product_id, s.revenue,
SUM(s.revenue) OVER
(ORDER BY s.revenue DESC ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW) AS running_total
FROM sales_cte s, total_revenue t
WHERE SUM(s.revenue) OVER (ORDER BY s.revenue DESC
ROWS BETWEEN UNBOUNDED PRECEDING AND
CURRENT ROW) <= t.total * 0.8;
```



@Tajamulkhan





33. Calculate average time between two purchases for each customer.



```
WITH cte AS (
    SELECT customer_id, order_date,
    LAG(order_date) OVER (PARTITION BY
customer_id
    ORDER BY order_date) AS prev_date
    FROM Orders)
SELECT customer_id,
AVG(DATEDIFF(DAY, prev_date, order_date))
AS avg_gap_days FROM cte
WHERE prev_date IS NOT NULL
GROUP BY customer_id;
```



@Tajamulkhann



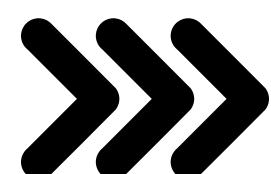


34. Show last purchase for each customer along with order amount.

```
WITH ranked_orders AS
(SELECT customer_id, order_id,
total_amount, ROW_NUMBER() OVER
(PARTITION BY customer_id ORDER BY
order_date DESC) AS rn FROM Orders)
SELECT customer_id, order_id,
total_amount
FROM ranked_orders
WHERE rn = 1;
```



@Tajamulkhan





35. Calculate year-over-year growth in revenue.



```
SELECT FORMAT(order_date, 'yyyy') AS year,  
SUM(total_amount) AS revenue,  
SUM(total_amount) - LAG(SUM(total_amount))  
OVER (ORDER BY FORMAT(order_date, 'yyyy'))  
AS yoy_growth  
FROM Orders  
GROUP BY FORMAT(order_date, 'yyyy');
```



Microsoft



@Tajamulkhann





36. Detect customers whose purchase amount is higher than their historical 90th percentile.

```
WITH ranked_orders AS (
  SELECT customer_id, order_id,
  total_amount,
  NTILE(10) OVER (PARTITION BY customer_id
  ORDER BY total_amount) AS decile
  FROM Orders)
SELECT customer_id, order_id, total_amount
FROM ranked_orders
WHERE decile = 10;
```





37. Find continuous login streaks (e.g., users who logged in 3 or more consecutive days).

```
WITH cte AS (
    SELECT user_id, login_date,
    DATEDIFF(DAY, ROW_NUMBER() OVER
    (PARTITION BY user_id ORDER BY login_date),
    login_date) AS grp FROM Logins)
SELECT user_id, MIN(login_date) AS
streak_start, MAX(login_date) AS streak_end,
COUNT(*) AS streak_length FROM cte
GROUP BY user_id, grp
HAVING COUNT(*) >= 3;
```





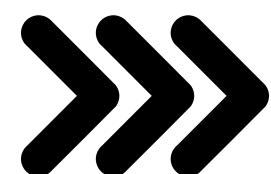
38. Calculate customer retention by month (Cohort analysis).



```
WITH Cohorts AS ( SELECT customer_id,  
MIN(DATEFROMPARTS(YEAR(order_date),  
MONTH(order_date), 1)) AS cohort_month FROM Orders  
GROUP BY customer_id),  
OrdersByMonth AS (  
SELECT customer_id, DATEFROMPARTS(YEAR(order_date),  
MONTH(order_date), 1)  
AS order_month FROM Orders)  
SELECT c.cohort_month, o.order_month,  
COUNT(DISTINCT o.customer_id) AS active_customers  
FROM Cohorts c  
JOIN OrdersByMonth o ON c.customer_id= o.customer_id  
GROUP BY c.cohort_month, o.order_month;
```



@Tajamulkhann



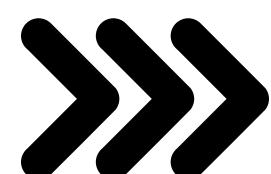


39. Find products that are always sold together (Market basket analysis).

```
SELECT A.product_id AS product_A,  
       B.product_id AS product_B,  
       COUNT(*) AS count_together  
  FROM Order_Details A  
  JOIN Order_Details B  
    ON A.order_id = B.order_id  
   AND  
A.product_id < B.product_id  
 GROUP BY A.product_id, B.product_id  
 HAVING COUNT(*) > 10;
```



@Tajamulkhan





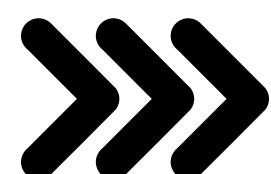
40. Calculate income inequality (Gini coefficient).

```
WITH income_cte AS (
    SELECT salary,
        SUM(salary) OVER (ORDER BY salary) AS cum_incom,
        COUNT(*) OVER() AS n,
        ROW_NUMBER() OVER (ORDER BY salary) AS r
    FROM Employee)
SELECT 1 - (2 * SUM((cum_income) / (SUM(salary)
OVER ()) * (1.0 / n))) ) AS gini_coefficient
FROM income_cte;
```

Uber



@Tajamulkhan





41. Compute the day when cumulative revenue first exceeded 50% of total revenue (median sales day).



```
WITH cte AS ( SELECT order_date,
SUM(total_amount) AS daily_rev
FROM Orders GROUP BY order_date),
cum_cte AS (
SELECT order_date, daily_rev, SUM(daily_rev) OVER
(ORDER BY order_date) AS cum_rev, SUM(daily_rev)
OVER() AS total_rev FROM cte)
SELECT TOP 1 order_date FROM cum_cte
WHERE cum_rev >= total_rev / 2
ORDER BY order_date;
```



@Tajamulkhan





42. Find percentiles (25th, 50th, 75th) of employee salaries.



```
SELECT
(SELECT PERCENTILE_CONT(0.25) WITHIN GROUP
(ORDER BY salary) OVER () FROM Employee) AS p25,
(SELECT PERCENTILE_CONT(0.50) WITHIN GROUP
(ORDER BY salary) OVER () FROM Employee) AS p50,
(SELECT PERCENTILE_CONT(0.75) WITHIN GROUP
(ORDER BY salary) OVER () FROM Employee) AS p75;
```



@Tajamulkhan





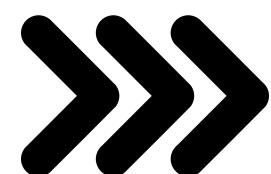
43. Retrieve customers with increasing order amounts over their last 3 orders.



```
WITH cte AS (
    SELECT customer_id, order_date, total_amount,
    LAG(total_amount, 2) OVER (PARTITION BY
    customer_id ORDER BY order_date) AS amt_t_minus_2,
    LAG(total_amount, 1) OVER (PARTITION BY
    customer_id ORDER BY order_date) AS amt_t_minus_1
    FROM Orders)
    SELECT customer_id, order_date, total_amount
    FROM cte
    WHERE amt_t_minus_2 < amt_t_minus_1
    AND amt_t_minus_1 < total_amount;
```



@Tajamulkhan





44. Calculate conversion funnel between different stages (e.g., visits → signups → purchases).



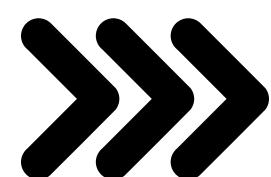
```
SELECT  
    SUM(CASE WHEN stage = 'visit' THEN 1  
    ELSE 0 END) AS visits,  
    SUM(CASE WHEN stage = 'sign_up' THEN 1  
    ELSE 0 END) AS sign_ups,  
    SUM(CASE WHEN stage = 'purchase' THEN 1  
    ELSE 0 END) AS purchases  
FROM Funnel;
```



Microsoft



@Tajamulkhann





45. Find the percentage of total sales contributed by top 10% of customers.

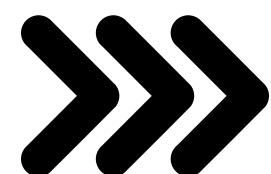


```
WITH cte AS (SELECT customer_id,  
SUM(total_amount) AS revenue  
FROM Orders GROUP BY customer_id),  
ranked AS (SELECT *, NTILE(10) OVER  
(ORDER BY revenue DESC) AS decile FROM cte)  
SELECT  
SUM(revenue) * 100.0 / (SELECT SUM(revenue)  
FROM cte) AS pct_top_10  
FROM ranked  
WHERE decile = 1;
```

Google



@Tajamulkhan





46. Calculate weekly active users

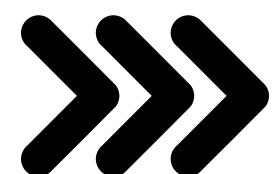


```
SELECT DATEPART(YEAR, login_date) AS year,  
DATEPART(WEEK, login_date) AS week,  
COUNT(DISTINCT user_id) AS wau  
FROM Logins  
GROUP BY DATEPART(YEAR, login_date),  
DATEPART(WEEK, login_date);
```

Uber



@Tajamulkhann





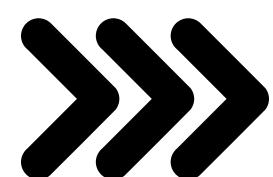
47. Find employees with salary higher than department average.



```
WITH dept_avg AS (
  SELECT department_id, AVG(salary) AS
    avg_salary
  FROM Employee
  GROUP BY department_id)
SELECT e.* FROM Employee e JOIN dept_avg d
ON e.department_id = d.department_id
WHERE e.salary > d.avg_salary;
```



@Tajamulkhann





48. Calculate time between user signup and their first purchase.



```
WITH first_purchase AS (
  SELECT user_id, MIN(purchase_date) AS
  first_purchase_date FROM Purchases
  GROUP BY user_id)
  SELECT u.user_id,
  DATEDIFF(DAY, u.signup_date,
  f.first_purchase_date) AS days_to_purchase
  FROM Users u JOIN first_purchase f
  ON u.user_id = f.user_id;
```



@Tajamulkhann



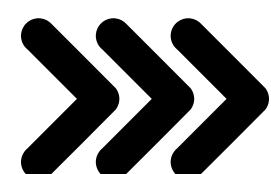


49. Retrieve the longest gap between orders for each customer.

```
WITH cte AS (
    SELECT customer_id, order_date,
    LAG(order_date) OVER (PARTITION BY
customer_id ORDER BY order_date) AS
prev_order_date FROM Orders)
SELECT customer_id, MAX(DATEDIFF(DAY,
prev_order_date, order_date)) AS max_gap
FROM cte
WHERE prev_order_date IS NOT NULL
GROUP BY customer_id;
```



@Tajamulkhan





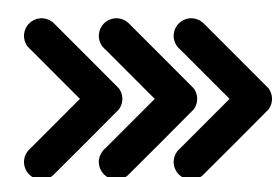
50. Identify customers with revenue below the 10th percentile.



```
WITH cte AS (
    SELECT customer_id, SUM(total_amount) AS total_revenue
    FROM Orders
    GROUP BY customer_id)
SELECT customer_id, total_revenue
FROM cte
WHERE total_revenue <
(SELECT PERCENTILE_CONT(0.1) WITHIN GROUP
(ORDER BY total_revenue) FROM cte);
```



@Tajamulkhan



Found Helpful?

Repost



Follow for more!

