1. Identify duplicates
   Method 1:-
   ```
   Select emp_id, count(*) from emp
   Group by emp_id
   having count(*)>1;
   ```
   Method 2:-
   ```
   With cte as(
   Select *, row_number()over(order by emp_id) as rn from emp
   )
   Select from cte where rn>1;
   ```

2. Delete duplicates
   ```
   With cte as(
   Select *, row_number()over(order by emp_id) as rn from emp
   )
   Delete from cte where rn>1;
   ```

3. Nth highest salary
   - General
     ```
     Select salary from emp order by salary desc limit 1 offset 2;
     ```
   - In a dept
     ```
     With cte as(
     Select *, dense_rank()over(partition by dept order by salary desc) as rnk)
     Select name from cte where rnk=2;
     ```
   - 2nd highest salary
     ```
     Select max(salary) from emp where salary < (select max(salary) from employee)
     ```

4. Employee salary > manager salary
   ```
   SELECT
     e.name AS employee_name,
     e.salary AS employee_salary,
     m.name AS manager_name,
     m.salary AS manager_salary
   FROM Employee e
   JOIN Employee m
     ON e.manager_id = m.emp_id
   WHERE e.salary > m.salary;
   ```

5. How many resultant rows for each type of join?

   A B
   1 1
   1 1
   2 1
   2 3
   2
   Inner join :- 2*3(1)+2*1(2)=8
   Left Join :- 2*3(1)+2*1(2)=8
   Right Join :- 2*3(1)+1(3)+2*1(2)=9
   Outer Join :- 2*3(1)+1(3)+2*1(2)=9

6. Calculate mode in sql

   Method 1 :-

```
SELECT product_id AS mode
FROM sales
GROUP BY product_id
ORDER BY COUNT(*) DESC
LIMIT 1;
```

   Method 2 :-

```
With cte as(
Select product_id, count(*) as freq from sales group by product)
Select * from cte where freq=(select max(freq) from cte)
```

   Method 3:-

```
With freq_cte as(
Select product_id, count(*) as freq from sales group by product
)
With rnk as(
Select *, rank() over(order by freq desc) as rk from freq_cte)
Select * from rnk where rk=1;
```

7. Custom sort - happiness index india should have its original ranking shown but should be at the top

```
Select * from happiness_index
Order by case when country=”India” then 1
```

Else 0 ,
Hapiness_index;

8. Difference btw all count
   Count(*) all value cout including nulls
   Count(1),count(-1),count(0), count("vanshika") are all same as count(*)
   Count(col_name) gives the count of all Non Null values

9. Find business days btw create_date & resolved_date excluding weekends
   Select datediff(day,create_date, resolved_date)-2*(datediff(week, create_date, resolved_date))

10. Convert comma separated values into separate rows
    SELECT
      e.id,
      s.value AS skill
    FROM Employee e
    CROSS APPLY STRING_SPLIT(e.skills, ',') s;

11. Top 5 products in category be sales
    With rank_cte as(
    Select *, dense_rank() over(partition by category order by sales desc) as rnk from sales)
    Select * from rank_cte where rk<=5;

12. Pattern Matching
    A[np]% :- n or p can come in 2nd
    A[^np}% :- anything apart from n and p can come in 2nd
    A[b-k]% :- anything between b to k can come in 2nd place

13. Find 3 month rolling average of sales
    Sales Table :- product_id amount date
    With year_month_sales as(
    Select * ,year(date) as year, month(date) as month
    from sales
    Group by year, month
)

Select *, sum(amount) over(partition by year, month rows between 3 preceding and current row) as rolling sum from year_month_sales;

14. Top 25% customers by sales
    Sales table :- cust_id product_id amount order_date
    With customer_sales as(
    Select *, sum(amount) as cust_sales from sales group by cust_id order by cust_sales desc)

    With percentile as(
    Select * , ntile(4) over(order by cust_sales desc) as cust_grouping from customer_sales)
    Select cust_id from percentile where cust_grouping=1;

15. Sales_data :- order_id product_id sales_amount country (us, uk, india)
    Output :- Product_id us_sale_amount uk_sale_amount india_sale _amount

    Select product_id,
    Sum(case when country=”us” then sales end) as us_sale_amount,
    Sum(case when country=”uk” then sales end) as uk_sale_amount,
    Sum(case when country=”india” then sales end) as india_sale_amount
    From sales
    Group by product_id;

    Now if they were as different sheets i.e one sales sheet for each country
    Each sale sheet has – order_id, product_id, amount
    - We will need to combine them into one sheet now

    With cte as(
    Select *, ‘us’ as country from orders_uk
    Union all
    Select *, ‘uk’ as country from orders_uk
    Union all
    Select *, ‘india’ as country from orders_uk
)
    Select product_id,

Sum(case when country=”us” then sales end) as us_sale_amount,
Sum(case when country=”uk” then sales end) as uk_sale_amount,
Sum(case when country=”india” then sales end) as india_sale_amount
From sales
Group by product_id;

16. Find students who know only sql and python

Table	student_id	skill
	1	Python
	1	SQL
	1	Tableau
	2	SQL
	2	Python
	3	SQL
	3	Tableau

- Distinct number of skills should be 2 and they should only be sql python
- Output :- 2

Select student_id from skill

Group by student_id

Where count(distinct(skill))=2

And sum(case when skill=”sql” or skill=”python” then 1 else 0)=2

17. Source table

Id	name
1	A
2	B
3	C
4	D

Target table

Id	Name
1	A
2	B
4	X
5	F

Output table :-

ID	Comment
3	New in source
4	New in Target
5	Mismatch

Select coalesce(s.id, t.id) as id
Case when t.name is null then “New in source” ,
Case when s.name is null then “New in target” ,
Else ‘mismatch’ end as comment
From source full join target t on s.id=t.id
Where s.name!=t.name or s.name is null or t.name is null

18. How can you generate all possible combinations of grouping ?
Suppose you want to find which route took least time and want to enter data for all possible combinations to make a master table
supplychain
---
product_id | src | warehouse | driver_id | route_taken | target_time
1. GROUP BY ROLLUP – Hierarchical Aggregation (Order Matters)

```
SELECT
  product_id,
  src,
  warehouse,
  driver_id,
  route_taken,
  MIN(target_time) AS min_time
FROM supplychain
GROUP BY
  product_id,
  src,
  ROLLUP (warehouse, driver_id, route_taken);
```

Key Point

- Aggregations follow a **hierarchy**
- **Order matters**
- Best for **drill-down / roll-up analysis**

2. GROUP BY CUBE – All Possible Combinations (Power Set)

```
SELECT
  product_id,
  src,
```

```
    warehouse,
    driver_id,
    route_taken,
    MIN(target_time) AS min_time
FROM supplychain
GROUP BY
    product_id,
    src,
    CUBE (warehouse, driver_id, route_taken);
```

Key Point

- Generates **all possible groupings**
- Order **does not matter**
- Best for **exploratory analysis**

3. `GROUP BY GROUPING SETS` – Explicit & Controlled Combinations

```
SELECT
    product_id,
    src,
    warehouse,
    driver_id,
    route_taken,
    MIN(target_time) AS min_time
FROM supplychain
GROUP BY
    product_id,
    src,
    GROUPING SETS (
        (warehouse, driver_id),
        (warehouse, route_taken),
        (route_taken)
    );
```

Key Point

- No extra combinations
- Full control
- Best for **production / reporting tables**

Feature	ROLLUP	CUBE	GROUPING SETS
Order sensitive	✅	❌	❌
Generates all combinations	❌	✅	❌
Custom combinations	❌	❌	✅
Best use case	Hierarchical totals	Full analysis	Targeted reporting

19. YOY

Data :- product_id , order_id, amount , order_date

WITH monthly_sales AS (

SELECT YEAR(order_date) AS year, MONTH(order_date) AS month,

SUM(amount) AS total_sales

FROM sales

GROUP BY YEAR(order_date), MONTH(order_date))

SELECT year, month, total_sales,

```
-- MoM
total_sales
    - LAG(total_sales, 1) OVER (ORDER BY year, month) AS mom_sales,
ROUND(
    (total_sales - LAG(total_sales, 1) OVER (ORDER BY year, month))
    / LAG(total_sales, 1) OVER (ORDER BY year, month) * 100,
    2
) AS mom_growth_percent,

-- YoY
```

```
total_sales
    - LAG(total_sales, 12) OVER (ORDER BY year, month) AS yoy_sales,
ROUND(
    (total_sales - LAG(total_sales, 12) OVER (ORDER BY year, month))
    / LAG(total_sales, 12) OVER (ORDER BY year, month) * 100,
    2
) AS yoy_growth_percent


FROM monthly_sales ORDER BY year, month;
```

20. Pivoting
    Data:- order_id, order_date, product_id, category
    Select year(order_date) as year,
    Sum(case when category=”Furniture” then sales else 0 end) as furniture_sales,
    Sum(case when category=”Office Supplies” then sales else 0 end) as
    Office_supplies_sales,
    Sum(case when category=”Technology” then sales else 0 end) as tecnology_sales,
    From orders
    Group by year(order_date)

21. Qualify keyword
    SELECT order_id,customer_id,order_date,amount
    FROM orders
    QUALIFY
      ROW_NUMBER() OVER (
        PARTITION BY customer_id
        ORDER BY order_date DESC
) = 1;

22. Median in sql
    Method 1:-
    With cte as(
    Select *, row_number() over(order by emp_age) as rn_asc
    Select *, row_number() over(order by emp_age desc) as rn_desc
    From emp
)
    Select average(emp_age) as median
    from cte

where abs(rn_asc – rn_desc)<=1;

Method 2:-

Select percentile_cont(0.5) within group(order by emp_age) over() as median from emp;

23. Pivot rows to column

DATA :-

Name	Value	Id
name	Adam	1
gender	Male	1
salary	5000	1

Method 1:-

```
Select id,
Max(case when name=”name” then value else “” end) as name ,
Max(case when name=”gender” then value else “” end) as gender ,
Max(case when name=”salary” then value else “” end) as salary
Group by dbo.emp_id
```

Method 2:-

```
Select id, [name], [gender], [salary] from
( Select id , name as ename, value from emp) as src_table
PIVOT
(max(value) for ename in ([name],[gender],[salary]) as pivot_table
```

24. How to remove tab spaces, \n etc

```
Select *, replace(address, char(a). ‘’) from emp
```

ASCII- char(a) for tab char(B) char(10)

25. Year to date / Month to date sales

```
SELECT order_id,order_date,amount,
  SUM(amount) OVER (
    PARTITION BY YEAR(order_date), MONTH(order_date)
    ORDER BY order_date
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS mtd_sales,
  SUM(amount) OVER (
    PARTITION BY YEAR(order_date)
    ORDER BY order_date
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

```
  ) AS ytd_sales
FROM sales
ORDER BY order_date;



SELECT
  product_id,
  SUM(CASE WHEN YEAR(order_date) = YEAR(CURDATE())
           AND MONTH(order_date) = MONTH(CURDATE())
      THEN amount ELSE 0 END) AS mtd_sales,
  SUM(CASE WHEN YEAR(order_date) = YEAR(CURDATE())
      THEN amount ELSE 0 END) AS ytd_sales
FROM sales
GROUP BY product_id;
```

26. Find employees with salary greater than department average

```
SELECT

  emp_id,emp_name,department_id,salary,
  AVG(salary) OVER (PARTITION BY department_id) AS dept_avg_salary
FROM employees
WHERE salary > AVG(salary) OVER (PARTITION BY department_id);
```

27. Find employees with salary less than company average

```
SELECT
  emp_id,
  emp_name,
  department_id,
  salary,
  AVG(salary) OVER () AS company_avg_salary
FROM employees
WHERE salary < AVG(salary) OVER ();
```

28. Find employees with salary less than thier own department average but higher than other department average

```
WITH dept_avg AS
```

```
( SELECT department_id,

AVG(salary) AS dept_avg_salary FROM employees GROUP BY department_id ),

 -- 2. Overall company average per department (excluding own)

other_dept_avg AS ( SELECT e1.emp_id, AVG(e2.salary) AS other_avg_salary

FROM employees e1 JOIN employees e2 ON e1.department_id <> e2.department_id
GROUP BY e1.emp_id )

SELECT e.emp_id, e.emp_name, e.department_id, e.salary, d.dept_avg_salary,
o.other_avg_salary

FROM employees e JOIN dept_avg d ON e.department_id = d.department_id

JOIN other_dept_avg o ON e.emp_id = o.emp_id

WHERE e.salary < d.dept_avg_salary

AND e.salary > o.other_avg_salary;
```

29. How to find departments having only male / female employees?

```
Select dept from hr

 group by dept

Having sum(case when gender="M" then 1 else 0)=0 (for female only)
```

30. Identify No sales days

```
With cte_gap as(

Select *,

datediff(day,Order_date, Lead(order_date,1) over(order by order_Date) as lead_day) as
gap

From sales

)

Select * from cte_gap where gap>1;
```

31. Compare monthly sales with previous month, same month previous year

```
WITH monthly_sales AS (
SELECT YEAR(order_date) AS yr, MONTH(order_date) AS mn,
SUM(amount) AS monthly_sales
FROM orders
 GROUP BY YEAR(order_date), MONTH(order_date) ),
same_month_prev_year_cte AS (
SELECT yr, mn, monthly_sales,
 LAG(monthly_sales) OVER ( PARTITION BY mn ORDER BY yr ) AS
same_month_prev_year FROM monthly_sales ),
previous_month_cte AS(
SELECT yr, mn, monthly_sales,
LAG(monthly_sales) OVER ( ORDER BY yr, mn ) AS prev_month_sales
FROM monthly_sales )
```

32. Employee with closest salary to average salary in a department

```
WITH dept_calc AS (
   SELECT emp_id,emp_name, department_id,salary,
      AVG(salary) OVER (PARTITION BY department_id) AS dept_avg_salary,
      ABS(
         salary - AVG(salary) OVER (PARTITION BY department_id)
      ) AS diff_from_avg
   FROM employees
),
```

```
ranked AS (
   SELECT *,
        DENSE_RANK() OVER (PARTITION BY department_id ORDER BY diff_from_avg
        ) AS rnk
   FROM dept_calc
)
SELECT emp_id, emp_name, department_id, salary, dept_avg_salary
FROM ranked
WHERE rnk = 1;
```

33. Convert data from rows into single concatenated and delimited string

```
Select name, STRING_AGG(Project_Name,' | ')
Within group ( order by Project_Name desc)
From projects
Group by name;
```

34. Find consecutive numbers

```
SELECT distinct(t1.num) as ConsecutiveNums
FROM logs t1, logs t2 , logs t3
WHERE t1.id=t2.id+1 AND t2.id=t3.id+1 AND t1.num=t2.num AND t2.num=t3.num
```

35. Customers whose revenue increases MOM basis

```sql
With CTE_tot_rev as
(Select customer_id,
YEAR(order_date) as YR, MONTH(order_date) as MTH,
SUM(revenue) as total_revenue
FROM orders
GROUP BY customer_id, YEAR(order_date), MONTH(order_date))

, CTE_prev_rev as
(Select *,
LAG(total_revenue) OVER (partition by customer_id ORDER BY YR, MTH) as prev_rev
FROM CTE_tot_rev)

Select * FROM CTE_prev_rev
WHERE total_revenue > prev_rev
```

36. Find only those days when user breaks his personal record

```sql
With CTE_Scores as
(Select *,
MAX(score) OVER (Partition by user_id ORDER BY score_date
ROWS BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING) as max_prev_score
FROM scores)


Select * FROM CTE_Scores
WHERE score = max_prev_score
```

Results Messages

	user_id	score_date	score
1	1	2024-01-01	10
2	1	2024-01-02	15
3	1	2024-01-03	12
4	1	2024-01-04	20
5	1	2024-01-05	18
6	1	2024-01-06	22
7	1	2024-01-07	21
8	1	2024-01-08	22

37. If pause is less than 5 min combine it to 1 viewing session

```sql
With CTESessions As
(Select * ,
LAG(EndTime) OVER (Partition by userid ORDER BY starttime) as prev_end_time
FROM StreamingSessions)

Select * ,
SUM(CASE WHEN prev_end_time is NULL THEN 1
WHEN DATEDIFF(MINUTE, prev_end_time, Starttime) <= 5 THEN 0
ELSE 1 END) OVER (Partition by userid ORDER BY Starttime) AS GrpNum
FROM CTESessions
```

Results Messages

UserID	SessionID	StartTime	EndTime	prev_end_time	GrpNum
1	101	2025-12-0...	2025-12-0...	NULL	1
1	102	2025-12-0...	2025-12-0...	2025-12-02 10:30:00.000	1
1	103	2025-12-0...	2025-12-0...	2025-12-02 11:00:00.000	1
1	104	2025-12-0...	2025-12-0...	2025-12-02 11:20:00.000	2
2	201	2025-12-0...	2025-12-0...	NULL	1
2	202	2025-12-0...	2025-12-0...	2025-12-02 09:45:00.000	1

```sql
Select UserID,  GrpNum,
MIN(Starttime) as StTime,
MAX(EndTime) as EndTime
FROM CTEGrp
GROUP BY UserID, GrpNum
```

Results Messages

UserID	GrpNum	StTime	EndTime
1	1	2025-12-02 10:00:00.000	2025-12-02 11:20:00.000
2	1	2025-12-02 09:00:00.000	2025-12-02 10:15:00.000