Most Asked Question Answer in SQL

```
create database Testing;
create table customers
        (customer_id int, name varchar(255), gender varchar(50),
        status varchar(50), city varchar(255));
insert into customers (customer_id, name, gender, status, city)
values
(1,'Alice','Female','active','Mumbai'),
(2,'Bob','Male','inactive','Delhi'),
(3,'Charlie','Male','active','Bangalore'),
(4,'Alice','Female','active','Mumbai'),
(5,'Eva','Female','active','Mumbai');

create table orders
        (order_id int,customer_id int,
        order_date date,order_amount int);
insert into orders(order_id,customer_id,order_date,order_amount)
values
(101,1,'2023-12-01',500),
(102,1,'2023-12-05',800),
(103,2,'2023-12-03',1200),
(104,3,'2023-12-02',1000),
(105,3,'2023-12-10',2000),
(106,5,'2023-12-11',400);

create table employees(employee_id int, name varchar(255),department varchar(255),
        salary int,status varchar(50));
insert into employees
        (employee_id,name,department,salary,status)
values
        (201,'Ravi','Sales',40000,'active'),
```

```
(202,'Nisha','Marketing',50000,'active'),
(203,'Ali','Sales',45000,'active'),
(204,'Sanjay','IT',70000,'active'),
(205,'Ravi','Sales',40000,'active');

create table sales_data
        (date date, employee_id int, sales int);

insert into sales_data(date,employee_id,sales)
        values
        ('2023-12-01',201,2000),
('2023-12-02',201,2200),
('2023-12-03',201,1800),
('2023-12-04',201,2500),
('2023-12-05',201,3000),
('2023-12-06',201,2800);
-- Find Duplicate Customers
SELECT name, city, COUNT(*)
FROM customers
GROUP BY name, city
HAVING COUNT(*) > 1;
/* GROUP BY puts similar records together.
COUNT(*) counts how many times each group appears.
HAVING COUNT(*) > 1 shows only those that appear more than once.*/

-- Demonstrate Different Types of JOINs
-- INNER JOIN Only the rows where there's a match in both tables
SELECT * FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id;

-- LEFT JOIN All from the left (customers), plus orders info if available.
```

```
SELECT * FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;

-- RIGHT JOIN (if supported by your SQL) All from the right (orders), plus customers info if available.
SELECT * FROM customers c
RIGHT JOIN orders o ON c.customer_id = o.customer_id;

-- FULL OUTER JOIN (if supported) Everything from both tables, matched where possible, NULL where there's no match.
SELECT * FROM customers c
FULL JOIN orders o ON c.customer_id = o.customer_id;

-- Use Window Functions
SELECT
  employee_id,
  date,
  sales,
  SUM(sales) OVER (ORDER BY date) AS running_total,
  RANK() OVER (ORDER BY sales DESC) AS rank_by_sales
FROM sales_data;
/*Daily sales
A 7-day running total
A rank of each day by highest sales*/

--Use a CTE to Filter or Rank
/*(Common Table Expression)
A CTE is like a temporary view or a shortcut that makes your query easier to read and debug.
It starts with WITH, followed by a name and a subquery.*/
WITH ranked_employees AS (
  SELECT *, RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank_in_dept
  FROM employees
```

```
)
SELECT * FROM ranked_employees WHERE rank_in_dept = 1;

--Use Subquery to Filter
SELECT name FROM customers WHERE customer_id IN
        (SELECT customer_id FROM orders GROUP BY customer_id
  HAVING SUM(order_amount) > 1000);
/*The inner query finds the customers with total order amount > ₹1000.
The outer query gets their names.*/

--Calculate Rolling 7-Day Average Sales
SELECT date,sales,
  AVG(sales) OVER (ORDER BY date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS
rolling_avg
FROM sales_data;

--WHERE vs HAVING – What's the difference?
/*WHERE filters before grouping.
HAVING filters after you group.*/
SELECT name,COUNT(*),status FROM employees
WHERE status = 'active'
GROUP BY name,status
HAVING COUNT(*) >1;

--How do you pivot data in SQL?
/*Pivoting means turning rows into columns.*/
SELECT city,
  SUM(CASE WHEN gender = 'Male' THEN 1 ELSE 0 END) AS male_count,
  SUM(CASE WHEN gender = 'Female' THEN 1 ELSE 0 END) AS female_count
FROM customers
GROUP BY city;
```