## **🧠 Example Interview Queries You Can Now Practice**

**🟢 EASY (Warm-Up)**

### **Q1. List all students who scored more than 75 marks.**

**Expected concepts:** WHERE, ORDER BY

SELECT student\_id, name, marks

FROM students

WHERE marks > 75

ORDER BY marks DESC;

### **Q2. Display distinct countries from the customers table.**

**Concept:** DISTINCT

SELECT DISTINCT country

FROM customers;

### **Q3. Find the total number of employees.**

**Concept:** COUNT

SELECT COUNT(\*) AS total\_employees

FROM employees;

### **Q4. Show all products with a price greater than 200.**

**Concept:** Filtering

SELECT product\_id, product\_name, price

FROM products

WHERE price > 200

ORDER BY price DESC;

# **🟡 MEDIUM (Core HackerRank Level)**

### **Q5. Find the average salary for each department.**

**Concepts:** JOIN, GROUP BY

SELECT d.department\_name,

AVG(e.salary) AS avg\_salary

FROM employees e

JOIN departments d

ON e.department\_id = d.department\_id

GROUP BY d.department\_name;

### **Q6. Count the number of employees in each department.**

SELECT d.department\_name,

COUNT(e.employee\_id) AS employee\_count

FROM departments d

LEFT JOIN employees e

ON d.department\_id = e.department\_id

GROUP BY d.department\_name;

### **Q7. Find customers who have placed more than 2 orders.**

**Concept:** HAVING

SELECT c.customer\_id,

c.customer\_name,

COUNT(o.order\_id) AS total\_orders

FROM customers c

JOIN orders o

ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.customer\_name

HAVING COUNT(o.order\_id) > 2;

### **Q8. Calculate total quantity sold per product.**

SELECT p.product\_name,

SUM(oi.quantity) AS total\_quantity

FROM products p

JOIN order\_items oi

ON p.product\_id = oi.product\_id

GROUP BY p.product\_name

ORDER BY total\_quantity DESC;

# **🔵 SUBQUERIES (Classic HackerRank)**

### **Q9. Find employees earning more than the average salary.**

SELECT employee\_id, employee\_name, salary

FROM employees

WHERE salary > (

SELECT AVG(salary)

FROM employees

);

### **Q10. Find products that have never been ordered.**

**Concept:** NOT EXISTS

SELECT p.product\_id, p.product\_name

FROM products p

WHERE NOT EXISTS (

SELECT 1

FROM order\_items oi

WHERE oi.product\_id = p.product\_id

);

# 

# **🔴 HARD (Advanced / Window Functions)**

**Q11. Rank employees by salary within each department.**

**Concept:** DENSE\_RANK()

SELECT employee\_id,

employee\_name,

department\_id,

salary,

DENSE\_RANK() OVER (

PARTITION BY department\_id

ORDER BY salary DESC

) AS salary\_rank

FROM employees;

### **Q12. Find the highest-paid employee in each department.**

WITH ranked\_employees AS (

SELECT employee\_id,

employee\_name,

department\_id,

salary,

ROW\_NUMBER() OVER (

PARTITION BY department\_id

ORDER BY salary DESC

) AS rn

FROM employees

)

SELECT employee\_id,

employee\_name,

department\_id,

salary

FROM ranked\_employees

WHERE rn = 1;

### 

### **Q13. Calculate running total of order quantities by order date.**

**Concept:** SUM() OVER

SELECT o.order\_date,

SUM(oi.quantity) AS daily\_quantity,

SUM(SUM(oi.quantity)) OVER (

ORDER BY o.order\_date

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

) AS running\_total

FROM orders o

JOIN order\_items oi

ON o.order\_id = oi.order\_id

GROUP BY o.order\_date

ORDER BY o.order\_date;

# **🔥 VERY HARD (HackerRank Final-Round Style)**

### **Q14. Find the top 3 customers by total quantity purchased.**

WITH customer\_totals AS (

SELECT c.customer\_id,

c.customer\_name,

SUM(oi.quantity) AS total\_quantity

FROM customers c

JOIN orders o

ON c.customer\_id = o.customer\_id

JOIN order\_items oi

ON o.order\_id = oi.order\_id

GROUP BY c.customer\_id, c.customer\_name

)

SELECT TOP 3 \*

FROM customer\_totals

ORDER BY total\_quantity DESC;

### 

### **Q15. For each department, show the salary difference between each employee and the department average.**

SELECT employee\_id,

employee\_name,

department\_id,

salary,

salary - AVG(salary) OVER (PARTITION BY department\_id)

AS salary\_vs\_department\_avg

FROM employees;

# **JOINS — HackerRank Style (T-SQL)**

## **🟢 BASIC JOINS**

### **Q1. List all employees with their department names.**

**Concept:** INNER JOIN

SELECT e.employee\_id,

e.employee\_name,

d.department\_name

FROM employees e

JOIN departments d

ON e.department\_id = d.department\_id;

### **Q2. Display all orders with the customer name and order date.**

SELECT o.order\_id,

c.customer\_name,

o.order\_date

FROM orders o

JOIN customers c

ON o.customer\_id = c.customer\_id

ORDER BY o.order\_date;

### 

### **Q3. Show product names and quantities ordered.**

SELECT p.product\_name,

oi.quantity

FROM order\_items oi

JOIN products p

ON oi.product\_id = p.product\_id;

## **🟡 INTERMEDIATE JOINS**

### **Q4. Find total quantity sold for each product.**

**Concept:** JOIN + GROUP BY

SELECT p.product\_name,

SUM(oi.quantity) AS total\_quantity

FROM products p

JOIN order\_items oi

ON p.product\_id = oi.product\_id

GROUP BY p.product\_name

ORDER BY total\_quantity DESC;

### **Q5. Find the number of orders placed by each customer.**

SELECT c.customer\_name,

COUNT(o.order\_id) AS total\_orders

FROM customers c

LEFT JOIN orders o

ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_name

ORDER BY total\_orders DESC;

🔑 **LEFT JOIN** ensures customers with **zero orders** still appear — very HackerRank-like.

### 

### **Q6. Find departments and the average salary of employees in each department.**

SELECT d.department\_name,

AVG(e.salary) AS avg\_salary

FROM departments d

JOIN employees e

ON d.department\_id = e.department\_id

GROUP BY d.department\_name;

## **🔵 MULTI-TABLE JOINS (Classic HackerRank)**

### **Q7. Show customer name, product name, and quantity purchased.**

**Concept:** 4-table JOIN

SELECT c.customer\_name,

p.product\_name,

oi.quantity

FROM customers c

JOIN orders o

ON c.customer\_id = o.customer\_id

JOIN order\_items oi

ON o.order\_id = oi.order\_id

JOIN products p

ON oi.product\_id = p.product\_id

ORDER BY c.customer\_name;

### 

### **Q8. Calculate total amount spent by each customer.**

**Concept:** JOIN + aggregation + arithmetic

SELECT c.customer\_name,

SUM(oi.quantity \* p.price) AS total\_spent

FROM customers c

JOIN orders o

ON c.customer\_id = o.customer\_id

JOIN order\_items oi

ON o.order\_id = oi.order\_id

JOIN products p

ON oi.product\_id = p.product\_id

GROUP BY c.customer\_name

ORDER BY total\_spent DESC;

## **🔴 ADVANCED JOINS (Interview Level)**

### **Q9. Find customers who have never placed an order.**

**Concept:** LEFT JOIN + NULL

SELECT c.customer\_id,

c.customer\_name

FROM customers c

LEFT JOIN orders o

ON c.customer\_id = o.customer\_id

WHERE o.order\_id IS NULL;

### 

### **Q10. Find employees whose salary is higher than their department average.**

**Concept:** JOIN + window function

SELECT e.employee\_id,

e.employee\_name,

e.salary,

d.department\_name

FROM employees e

JOIN departments d

ON e.department\_id = d.department\_id

WHERE e.salary > (

SELECT AVG(salary)

FROM employees

WHERE department\_id = e.department\_id

);

### 

### **Q11. Find the top-earning employee in each department.**

**Concept:** JOIN + window function

WITH ranked\_employees AS (

SELECT e.employee\_id,

e.employee\_name,

e.salary,

d.department\_name,

ROW\_NUMBER() OVER (

PARTITION BY e.department\_id

ORDER BY e.salary DESC

) AS rn

FROM employees e

JOIN departments d

ON e.department\_id = d.department\_id

)

SELECT employee\_id,

employee\_name,

salary,

department\_name

FROM ranked\_employees

WHERE rn = 1;

## **🟣 FULL OUTER JOIN (Rare but Tested)**

### **Q12. Show all departments and employees, including departments without employees and employees without departments.**

SELECT d.department\_name,

e.employee\_name

FROM departments d

FULL OUTER JOIN employees e

ON d.department\_id = e.department\_id;