75 BASIC SQL PRACTICAL QUESTIONS

SECTION 1: SIMPLE SELECT + FILTER

Q1. List all employees
Hint: SELECT *

```
SELECT * FROM employees;
```

Q2. Show the employee's first name and email
Hint: column selection

```
SELECT first_name, email FROM employees;
```

Q3. Find all employees from department_id = 3
Hint: WHERE

```
SELECT * FROM employees WHERE department_id = 3;
```

Q4. Get employees hired after 2023-01-01
Hint: DATE filter

```
SELECT * FROM employees WHERE hire_date > '2023-01-01';
```

Q5. List all active employees
Hint: status filter

```
SELECT * FROM employees WHERE status = 'active';
```

Q6. Show all departments located in Delhi
Hint: WHERE

```
SELECT * FROM departments WHERE location = 'Delhi';
```

Q7. Find all users with gmail accounts
Hint: LIKE '%gmail%'

```
SELECT * FROM users WHERE email LIKE '%gmail%';
```

Q8. List products with price > 1000
Hint: WHERE

```
SELECT * FROM products WHERE price > 1000;
```

Q9. Show orders with status = 'completed'
Hint: WHERE

```
SELECT * FROM orders WHERE status = 'completed';
```

Q10. Get all posts with more than 500 views
Hint: WHERE

```
SELECT * FROM posts WHERE views > 500;
```

SECTION 2: ORDER BY + LIMIT

Q11. Get the top 5 highest-paid employees
Hint: ORDER BY DESC, LIMIT

```
SELECT * FROM salaries
ORDER BY base_salary DESC
LIMIT 5;
```

Q12. List 10 cheapest products
Hint: ORDER BY ASC

```
SELECT * FROM products
ORDER BY price ASC
LIMIT 10;
```

Q13. Show latest 5 orders
Hint: ORDER BY date

```
SELECT * FROM orders
ORDER BY order_date DESC
LIMIT 5;
```

Q14. Get employees sorted by hire_date
Hint: ORDER BY

```
SELECT * FROM employees
ORDER BY hire_date;
```

Q15. Show top 3 most viewed posts
Hint: ORDER BY views

```
SELECT * FROM posts
ORDER BY views DESC LIMIT 3;
```

50 INTERMEDIATE SQL QUESTIONS

SECTION 1: MULTI-TABLE JOINS

Q1. Find employee name, department name, and manager name
Hint: self join + multiple joins

```sql
SELECT
  e.first_name AS employee_name,
  d.department_name,
  m.first_name AS manager_name
FROM employees e
  LEFT JOIN departments d ON e.department_id = d.department_id
  LEFT JOIN employees m ON e.manager_id = m.employee_id;
```

Q2. Get total salary (base + variable) per employee with department name
Hint: JOIN + SUM

```sql
SELECT
  e.employee_id,
  e.first_name,
  d.department_name,
  SUM(s.base_salary + IFNULL(s.variable_salary, 0)) AS total_salary
FROM employees e
  JOIN departments d ON e.department_id = d.department_id
  JOIN salaries s ON e.employee_id = s.employee_id
GROUP BY e.employee_id, e.first_name, d.department_name;
```

Q3. List all orders with user name and total items count
Hint: JOIN + COUNT

```sql
SELECT
  o.order_id,
  u.name,
  COUNT(oi.order_item_id) AS total_items
FROM orders o
  JOIN users u ON o.user_id = u.user_id
  LEFT JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY o.order_id, u.name;
```

Q4. Show products with category names (multiple categories per product)
Hint: many-to-many JOIN

```sql
SELECT
  p.product_id,
  p.name,
  c.category_name
FROM products p
  JOIN product_categories pc ON p.product_id = pc.product_id
  JOIN categories c ON pc.category_id = c.category_id;
```

Q5. List posts with author name and category names
Hint: JOIN + M:N

```sql
SELECT
  p.post_id,
  p.title,
  a.name AS author_name,
  bc.name AS category_name
FROM posts p
  JOIN authors a ON p.author_id = a.author_id
  JOIN post_categories pc ON p.post_id = pc.post_id
  JOIN blog_categories bc ON pc.category_id = bc.category_id;
```

SECTION 2: CONDITIONAL AGGREGATION

Q6. Count present vs absent days per employee
Hint: CASE + COUNT

```sql
SELECT
  employee_id,
  COUNT(CASE WHEN status = 'present' THEN 1 END) AS present_days,
  COUNT(CASE WHEN status = 'absent' THEN 1 END) AS absent_days
FROM attendance
GROUP BY employee_id;
```

100 HARD SQL INTERVIEW PROBLEMS

SECTION 1: COMPLEX JOINS + LOGIC

Q1. Find employees whose salary increased compared to previous period
Hint: self join on salaries

```
SELECT s1.employee_id
FROM salaries s1
  JOIN salaries s2
    ON s1.employee_id = s2.employee_id
    AND s1.salary_from > s2.salary_from
WHERE s1.base_salary > s2.base_salary;
```

Q2. Find employees who never took leave
Hint: NOT EXISTS

```
SELECT * FROM employees e
WHERE NOT EXISTS (
    SELECT 1
    FROM attendance a
    WHERE a.employee_id = e.employee_id
    AND a.status = 'leave'
);
```

Q3. Find departments where all employees are active
Hint: HAVING

```
SELECT department_id
FROM employees
GROUP BY department_id
  HAVING COUNT(*) = COUNT(CASE WHEN status = 'active' THEN 1 END);
```

Q4. Find users who placed orders in every month of 2024
Hint: GROUP BY + COUNT DISTINCT

```
SELECT user_id
FROM orders
WHERE YEAR(order_date) = 2024
GROUP BY user_id
  HAVING COUNT(DISTINCT MONTH(order_date)) = 12;
```

Q5. Find products never sold but reviewed
Hint: JOIN + NOT EXISTS

```
SELECT p.*
FROM products p
    JOIN reviews r ON p.product_id = r.product_id
WHERE
    NOT EXISTS (
        SELECT 1
        FROM order_items oi
        WHERE
            oi.product_id = p.product_id
    );
```

Q6. Find employees with highest salary in their department
Hint: subquery

```
SELECT e.employee_id, e.first_name, e.department_id, s.base_salary
FROM employees e
    JOIN salaries s ON e.employee_id = s.employee_id
WHERE
    s.base_salary = (
        SELECT MAX(s2.base_salary)
        FROM employees e2
            JOIN salaries s2 ON e2.employee_id = s2.employee_id
        WHERE
            e2.department_id = e.department_id
    );
```

Q7. Find orders where payment is missing
Hint: LEFT JOIN

```
SELECT o.* FROM orders o
  LEFT JOIN payments p ON o.order_id = p.order_id
WHERE p.payment_id IS NULL;
```

Q8. Find products with more reviews than orders
Hint: COUNT compare

```
SELECT p.product_id
FROM products p
  LEFT JOIN reviews r ON p.product_id = r.product_id
  LEFT JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id
  HAVING COUNT(DISTINCT r.review_id) > COUNT(DISTINCT oi.order_item_id);
```

30 WINDOW FUNCTION QUESTIONS

SECTION 1: RANKING BASICS

Q1. Rank employees by salary (highest first)
Hint: RANK() OVER (ORDER BY)

```
SELECT e.employee_id, e.first_name, s.base_salary,
    RANK() OVER (ORDER BY s.base_salary DESC) AS salary_rank
FROM employees e
JOIN salaries s
    ON e.employee_id = s.employee_id;
```

Q2. Dense rank employees by salary
Hint: DENSE_RANK()

```
SELECT e.employee_id,e.first_name,s.base_salary,
DENSE_RANK() OVER (ORDER BY s.base_salary DESC) AS salary_rank
FROM employees e
JOIN salaries s
ON e.employee_id = s.employee_id;
```

Q3. Row number for employees by hire date
Hint: ROW_NUMBER()

```
SELECT employee_id, first_name,hire_date,
ROW_NUMBER() OVER (ORDER BY hire_date) AS row_num
FROM employees;
```

Q4. Rank products by price within each category
Hint: PARTITION BY category

```
SELECT p.product_id,p.name,pc.category_id,p.price,
RANK() OVER (PARTITION BY pc.category_id ORDER BY p.price DESC) AS price_rank
FROM products p
JOIN product_categories pc
ON p.product_id = pc.product_id;
```

Q5. Top 3 highest paid employees per department
Hint: PARTITION + RANK

```
SELECT *
FROM ( SELECT e.employee_id,e.first_name,e.department_id,s.base_salary,
```