Perfect! Here are **25 Medium-Level SQL Interview Questions with Answers**, designed for hands-on practice on platforms like **HackerRank**:

🔹 1. Top 5 Customers by Total Purchase

```
SELECT * FROM customers
ORDER BY total_purchase DESC
LIMIT 5;
```

🔹 2. Employees Joined After 2020

```
SELECT * FROM employees
WHERE join_date > '2020-01-01';
```

🔹 3. Average Salary per Department

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department;
```

🔹 4. Employees in HR or IT Department

```
SELECT * FROM employees
WHERE department IN ('HR', 'IT');
```

🔹 5. Find Duplicate Emails

```
SELECT email, COUNT(*)
FROM users
GROUP BY email
HAVING COUNT(*) > 1;
```

🔹 6. Employees Whose Name Starts with 'A'

```
SELECT * FROM employees
WHERE name LIKE 'A%';
```

🔹 **7. Employees with Even ID**

```
SELECT * FROM employees
WHERE MOD(id, 2) = 0;
```

🔹 **8. Count of Employees per Department**

```
SELECT department, COUNT(*) AS total
FROM employees
GROUP BY department;
```

🔹 **9. Products Not Yet Sold**

```
SELECT * FROM products
WHERE id NOT IN (SELECT product_id FROM sales);
```

🔹 **10. Employees and Their Managers**

```
SELECT e.name AS employee, m.name AS manager
FROM employees e
JOIN employees m ON e.manager_id = m.id;
```

🔹 **11. Second Highest Salary**

```
SELECT MAX(salary)
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

🔹 **12. Departments with More Than 5 Employees**

```
SELECT department, COUNT(*) AS emp_count
FROM employees
GROUP BY department
HAVING COUNT(*) > 5;
```

🔹 **13. Cities with More Than 1 Customer**

```
SELECT city, COUNT(*)
FROM customers
GROUP BY city
HAVING COUNT(*) > 1;
```

🔹 **14. Customers Without Any Orders**

```
SELECT * FROM customers
WHERE id NOT IN (SELECT customer_id FROM orders);
```

🔹 **15. Names and Length of Each Name**

```
SELECT name, LENGTH(name) AS name_length
FROM users;
```

🔹 **16. Extract Join Year**

```
SELECT name, EXTRACT(YEAR FROM join_date) AS join_year
FROM employees;
```

🔹 **17. Orders in Last 30 Days**

```
SELECT * FROM orders
WHERE order_date >= CURRENT_DATE - INTERVAL '30 days';
```

🔹 **18. Products Priced Above Average**

```
SELECT * FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

🔹 **19. Salary with Department Average**

```
SELECT name, salary, department,
    AVG(salary) OVER (PARTITION BY department) AS avg_dept_salary
FROM employees;
```

🔹 20. Employees Without Managers

```
SELECT * FROM employees
WHERE manager_id IS NULL;
```

🔹 21. Users Logged in Today

```
SELECT * FROM logins
WHERE login_date = CURRENT_DATE;
```

🔹 22. Customers with More Than 3 Orders

```
SELECT customer_id, COUNT(*) AS order_count
FROM orders
GROUP BY customer_id
HAVING COUNT(*) > 3;
```

🔹 23. Rank Products by Price Within Category

```
SELECT product_name, category, price,
    RANK() OVER (PARTITION BY category ORDER BY price DESC) AS price_rank
FROM products;
```

🔹 24. Total Orders Per Day

```
SELECT order_date, COUNT(*) AS total_orders
FROM orders
GROUP BY order_date;
```

🔹 25. Median Salary (Workaround)

```
SELECT salary
FROM (
 SELECT salary, ROW_NUMBER() OVER (ORDER BY salary) AS rn
 FROM employees
) AS ranked
```

WHERE rn = (SELECT CEIL(COUNT(*)/2.0) FROM employees);

Would you like me to continue with the **25 Advanced-Level Questions** next?

Awesome! Here's a set of **25 Advanced SQL Interview Questions with Answers** to practice and strengthen your skills further:

🔶 1. Find Nth Highest Salary

```
SELECT DISTINCT salary
FROM employees e1
WHERE N - 1 = (
  SELECT COUNT(DISTINCT salary)
  FROM employees e2
  WHERE e2.salary > e1.salary
);
```

Replace N with the desired rank (e.g., 2 for second highest).

🔶 2. Running Total of Salaries

```
SELECT name, salary,
    SUM(salary) OVER (ORDER BY id) AS running_total
FROM employees;
```

🔶 3. Difference Between Consecutive Salaries

```
SELECT name, salary,
    salary - LAG(salary) OVER (ORDER BY salary) AS diff_from_prev
FROM employees;
```

🔶 4. Cumulative Count by Department

```
SELECT department, name,
    COUNT(*) OVER (PARTITION BY department ORDER BY name) AS running_count
FROM employees;
```

🔶 **5. Dense Rank Products by Sales Amount**

```
SELECT product_id, SUM(amount) AS total_sales,
    DENSE_RANK() OVER (ORDER BY SUM(amount) DESC) AS sales_rank
FROM sales
GROUP BY product_id;
```

🔶 **6. Count Nulls in a Column**

```
SELECT COUNT(*) - COUNT(email) AS null_emails
FROM users;
```

🔶 **7. Pivot Table: Total Sales by Region**

```
SELECT
 region,
 SUM(CASE WHEN year = 2023 THEN sales ELSE 0 END) AS sales_2023,
 SUM(CASE WHEN year = 2024 THEN sales ELSE 0 END) AS sales_2024
FROM sales
GROUP BY region;
```

🔶 **8. Reverse a String**

```
SELECT REVERSE('OpenAI') AS reversed;
```

🔶 **9. Find Employees with More Than Average Salary in Their Department**

```
SELECT *
FROM employees e
WHERE salary > (
 SELECT AVG(salary)
 FROM employees
 WHERE department = e.department
);
```

🔶 10. Window Function to Find Latest Login per User

```
SELECT *
FROM (
  SELECT user_id, login_time,
       ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY login_time DESC) AS rn
  FROM logins
) AS ranked
WHERE rn = 1;
```

🔶 11. Top Earning Employee per Department

```
SELECT *
FROM (
  SELECT *, RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rnk
  FROM employees
) ranked
WHERE rnk = 1;
```

🔶 12. Days Between Order and Delivery

```
SELECT order_id, DATEDIFF(delivery_date, order_date) AS days_to_deliver
FROM orders;
```

🔶 13. Self Join: Find Employee and Their Manager

```
SELECT e.name AS employee, m.name AS manager
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.id;
```

🔶 14. Find Consecutive Login Days

```
SELECT user_id, login_date,
     login_date - INTERVAL ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY login_date)
DAY AS grp
FROM logins;
```

Group by this expression to find streaks.

🔶 15. Find Employees with Same Salary

```
SELECT salary, COUNT(*) AS count
FROM employees
GROUP BY salary
HAVING COUNT(*) > 1;
```

🔶 16. Detect Gaps in Invoice Numbers

```
SELECT invoice_id
FROM (
  SELECT invoice_id, LAG(invoice_id) OVER (ORDER BY invoice_id) AS prev_id
  FROM invoices
) AS t
WHERE invoice_id != prev_id + 1;
```

🔶 17. Retrieve 3rd Row From Table

```
SELECT *
FROM (
  SELECT *, ROW_NUMBER() OVER (ORDER BY id) AS rn
  FROM employees
) AS temp
WHERE rn = 3;
```

🔶 18. Correlated Subquery Example

```
SELECT name, salary
FROM employees e
WHERE salary > (
  SELECT AVG(salary)
  FROM employees
  WHERE department = e.department
```

);

🔶 19. Rows with Max Salary Per Department

```
SELECT *
FROM employees e
WHERE salary = (
  SELECT MAX(salary)
  FROM employees
  WHERE department = e.department
);
```

🔶 20. Generate a Series of Dates

(Works in PostgreSQL)

```
SELECT generate_series('2024-01-01'::date, '2024-01-10'::date, '1 day') AS date;
```

🔶 21. Find All Employees Earning More Than Their Manager

```
SELECT e.name
FROM employees e
JOIN employees m ON e.manager_id = m.id
WHERE e.salary > m.salary;
```

🔶 22. Remove Duplicate Rows (Keep One)

```
DELETE FROM users
WHERE id NOT IN (
  SELECT MIN(id)
  FROM users
  GROUP BY email
);
```

🔶 23. Update with Join

```
UPDATE employees e
```

```
SET salary = e.salary * 1.1
FROM departments d
WHERE e.department = d.name AND d.bonus_eligible = TRUE;
```

🔶 24. Trigger to Audit Table Updates

```
CREATE TRIGGER audit_changes
AFTER UPDATE ON employees
FOR EACH ROW
INSERT INTO audit_log(emp_id, changed_at)
VALUES (OLD.id, CURRENT_TIMESTAMP);
```

🔶 25. Use of CTE to Chain Queries

```
WITH dept_avg AS (
  SELECT department, AVG(salary) AS avg_salary
  FROM employees
  GROUP BY department
)
SELECT e.name, e.salary, d.avg_salary
FROM employees e
JOIN dept_avg d ON e.department = d.department;
```

✅ These are perfect for **platforms like HackerRank, LeetCode, DataLemur, StrataScratch, and interviews**.

Would you like all **50 questions in a formatted PDF** now?