SQL INTERVIEW PREPARATION PART 3.1

JOIN QUESTIONS

1. **Find Customers Who Made Purchases**

   Scenario: You have two tables:
   Customers with customer details.
   Orders with order details.
   Question: Write a query to find the names of customers who have made at least one purchase.
   Solution:

```
SELECT c.name
FROM customers c
INNER JOIN orders o
ON c.id = o.customer_id;
```

2. **Customers Without Orders**

   Scenario: Using the same Customers and Orders tables.
   Question: Find the names of customers who have not made any purchases.
   Solution:

```
SELECT c.name
FROM customers c
LEFT JOIN orders o
ON c.id = o.customer_id
WHERE o.customer_id IS NULL;
```

3. **Employees Reporting to Managers**

   Scenario: In a company, each employee reports to a manager. The table Employees has columns: EmployeeID, Name, and ManagerID.
   Question: Write a query to list each employee's name along with their manager's name.
   Solution:

```
SELECT e1.Name AS Employee_name,
       e2.Name AS Manager_name
FROM Employees e1
LEFT JOIN Employees e2
ON e1.ManagerID = e2.EmployeeID;
```

   Query Using SELF JOIN:

```
SELECT e1.Name AS Employee_name,
       e2.Name AS Manager_name
FROM Employees e1
JOIN Employees e2
ON e1.ManagerID = e2.EmployeeID;
```

Key Differences:

SELF JOIN:

- Retrieves only employees who have a manager (ManagerID is not NULL).
- Useful when you want to exclude top-level employees, like a CEO.

LEFT JOIN:

- Includes all employees, even if they don't have a manager (ManagerID IS NULL).
- Useful for hierarchical reporting when you need the full employee list.

4. **Combine Product Information**

Scenario: You have two tables:

Products with product IDs and names.

ProductDetails with product IDs and descriptions.

Question: Write a query to get a list of all products along with their descriptions, including products that don't have descriptions.

Solution:

```
SELECT p.product_name,
        pd.product_description
FROM Products p
LEFT JOIN ProductDetails pd
ON p.product_id = pd.product_id;
```

5. **Mismatched Data**

Scenario: You have two tables:

Sales with transaction details.

Inventory with product stock.

Question: Write a query to find products that were sold but are not in the inventory list.

Solution:

```
SELECT s.product_name
FROM Sales s
LEFT JOIN Inventory i
ON s.product_id = i.product_id
WHERE i.product_id IS NULL;
```

6. **Find Common Customers Across Platforms**

Scenario: You have two tables:

OnlineSales with customer IDs who bought online.

InStoreSales with customer IDs who bought in-store.

Question: Write a query to find customers who purchased both online and in-store.

Solution:

```
SELECT os.customer_name
FROM OnlineSales os
INNER JOIN InStoreSales is
```

```
ON os.customer_id = is.customer_id;
```

7. **Most Recent Orders by Customer**
   Scenario: You have a Customers table and an Orders table.
   Question: Write a query to get the most recent order (by order date) for each customer.
   Solution:
   1. Using ROW_NUMBER() with a CTE:

```
WITH RankedOrders AS (
  SELECT o.customer_id, o.order_date,
       ROW_NUMBER() OVER (PARTITION BY o.customer_id ORDER BY o.order_date
DESC) AS rn
  FROM Orders o
)
SELECT c.customer_id, o.order_date
FROM Customers c
INNER JOIN RankedOrders o
ON c.customer_id = o.customer_id
WHERE o.rn = 1;
```

   2. Using a Subquery:

```
SELECT c.customer_id, o.order_date
FROM Customers c
INNER JOIN Orders o
ON c.customer_id = o.customer_id
WHERE o.order_date = (
  SELECT MAX(order_date)
  FROM Orders
  WHERE Orders.customer_id = o.customer_id
);
```

8. **Total Sales by Region**
   Scenario: You have two tables:
   Orders with sales and region information.
   Regions with region names and IDs.
   Question: Write a query to calculate the total sales for each region.
   Solution:

```
SELECT r.region_name,
        COALESCE(SUM(o.sales), 0) AS total_sales
FROM Orders o
LEFT JOIN Regions r
```

```
ON o.region_id = r.region_id
GROUP BY r.region_name;
```

Key Notes:

Regions with no associated orders will show NULL or 0 (depending on the database implementation). To display 0 instead of NULL, you can use COALESCE.

9. **Missing Data Audit**

Scenario: You have two tables:
Invoices with invoice details.
Payments with payment details.
Question: Write a query to find invoices that have not been paid yet.
Solution:

```
SELECT i.invoice_id
FROM Invoices i
LEFT JOIN Payments p
ON i.invoice_id = p.invoice_id
WHERE p.invoice_id IS NULL;
```

10. **Department-Wise Employee Count**

Scenario: You have two tables:
Employees with employee details and DepartmentID.
Departments with department details.
Question: Write a query to list each department along with the total number of employees. Include departments with no employees.
Solution:

```
SELECT d.dept_name ,
        COUNT(e.emp_id) AS employee_count
FROM Departments d
LEFT JOIN Employees e
ON d.DepartmentID = e.DepartmentID
GROUP BY d.dept_name;
```

11. **Duplicate Orders**

Scenario: You have an Orders table.
Question: Write a query to find duplicate orders based on customer ID and order date.
Solution:

1. Using **SELF JOIN**

```
SELECT o1.order_id, o1.customer_id, o1.order_date
FROM Orders o1
```

```
INNER JOIN Orders o2
ON o1.customer_id = o2.customer_id
AND o1.order_date = o2.order_date
AND o1.order_id <> o2.order_id;
```

2. Using **Window Functions**

```
SELECT order_id, customer_id, order_date
FROM (
   SELECT order_id, customer_id, order_date,
        COUNT(*) OVER (PARTITION BY customer_id, order_date) AS order_count
   FROM Orders
) AS subquery
WHERE order_count > 1;
```

12. Top-Selling Products

Scenario: You have Products and Sales tables.

Question: Write a query to find the top 3 best-selling products.

Solution:

```
SELECT p.product_name,
          SUM(s.sales_amt) AS total_sales
 FROM Sales s
 INNER JOIN Products p
 ON s.product_id = p.product_id
 GROUP BY p.product_name
 ORDER BY total_sales DESC
 LIMIT 3;
```

Query using ROW_NUMBER():

```
WITH RankedProducts AS (
   SELECT p.product_name,
        SUM(s.sales_amt) AS total_sales,
        ROW_NUMBER() OVER (ORDER BY SUM(s.sales_amt) DESC) AS row_num
   FROM Sales s
   INNER JOIN Products p
   ON s.product_id = p.product_id
   GROUP BY p.product_name
)
SELECT product_name, total_sales
FROM RankedProducts
WHERE row_num <= 3;
```

13. Find Employees with and without Assigned Projects

Scenario: You have two tables:
Employees with employee details.
Projects with project assignments (EmployeeID and ProjectID).
Question: Write a query to list employees who are assigned to projects and those who are not.
Solution:

```
SELECT e.emp_name,
        CASE
            WHEN p.ProjectID IS NULL THEN 'Not Assigned To Project'
            ELSE 'Assigned To Project'
        END AS Project_Status
FROM Employees e
LEFT JOIN Projects p
ON e.EmployeeID = p.EmployeeID
```

14. Orders with Shipping Delays

Scenario:
Orders table includes order IDs and shipping dates.
Shipments table includes shipment IDs and actual delivery dates.
Question: Write a query to find orders where the delivery was delayed (actual delivery date is later than the shipping date).
Solution:

```
SELECT o.order_id
FROM Orders o
INNER JOIN Shipments s
ON o.order_id = s.order_id
WHERE s.delivery_date > o.shipping_date;
```

15. Find Best-Selling Product in Each Category

Scenario:
Products table includes product names and category IDs.
Sales table includes product IDs and the number of units sold.
Question: Write a query to find the best-selling product in each category.
Solution:

```
 WITH TopSellingCTE AS (
    SELECT
      p.category_id,
      p.product_name,
      SUM(s.units_sold) AS total_units_sold,
        ROW_NUMBER() OVER (PARTITION BY p.category_id ORDER BY
        SUM(s.units_sold) DESC) AS rank
    FROM Products p
    INNER JOIN Sales s
    ON p.product_id = s.product_id
```

```
    GROUP BY p.category_id, p.product_name
)
SELECT category_id, product_name, total_units_sold
FROM TopSellingCTE
WHERE rank = 1;
```