CASE STUDY MCQ

Case Study 1

RetailChain Inc. operates a network of retail stores and manages its data in a SQL-based relational database. The company has the following tables:

1. Customers
    - customer_id (PK)
    - name
    - city
2. Orders
    - order_id (PK)
    - customer_id (FK)
    - order_date
    - total_amount
3. Products
    - product_id (PK)
    - name
    - category
4. Order_Items
    - order_item_id (PK)
    - order_id (FK)
    - product_id (FK)
    - quantity
5. Inventory
    - store_id (PK)
    - product_id (PK)
    - stock

RetailChain wants insights from its data to improve customer targeting, product availability, and sales analysis.

Q1. Which query will return the names of customers who have placed at least one order?

A. SELECT c.name FROM Customers c LEFT JOIN Orders o ON c.customer_id = o.customer_id;
B. SELECT c.name FROM Customers c INNER JOIN Orders o ON c.customer_id = o.customer_id;
C. SELECT c.name FROM Customers c RIGHT JOIN Orders o ON c.customer_id = o.customer_id;
D. SELECT DISTINCT c.name FROM Customers c FULL OUTER JOIN Orders o ON c.customer_id = o.customer_id;

Correct Answer: B

Explanation: INNER JOIN returns only those customers who have matching orders. LEFT or FULL OUTER joins may include customers with no orders.

Q2. Which query lists all customers, including those who never placed an order?

A. SELECT c.name, o.order_id FROM Customers c LEFT JOIN Orders o ON c.customer_id = o.customer_id;
B. SELECT c.name, o.order_id FROM Customers c RIGHT JOIN Orders o ON c.customer_id = o.customer_id;
C. SELECT c.name, o.order_id FROM Customers c INNER JOIN Orders o ON c.customer_id = o.customer_id;
D. SELECT c.name FROM Orders o LEFT JOIN Customers c ON c.customer_id = o.customer_id;

Correct Answer: A

Explanation: LEFT JOIN ensures all customers are shown, with NULLs for those without matching orders.

Q3. Which query finds all customers and all orders, including those without matches on either side?

A.SELECT c.name, o.order_id FROM Customers c INNER JOIN Orders o ON c.customer_id = o.customer_id;

B.SELECT c.name, o.order_id FROM Customers c FULL OUTER JOIN Orders o ON c.customer_id = o.customer_id;

C.SELECT c.name, o.order_id FROM Customers c RIGHT JOIN Orders o ON c.customer_id = o.customer_id WHERE c.customer_id IS NULL;

D.SELECT c.name, o.order_id FROM Customers c LEFT JOIN Orders o ON c.customer_id = o.customer_id WHERE o.order_id IS NULL;

Correct Answer: B

Explanation: FULL OUTER JOIN includes all records from both tables whether there is a match or not.

Q4. Retail Chain wants to simulate all combinations of stores and products for inventory planning. Which query achieves this?

A.SELECT * FROM Inventory i INNER JOIN Products p ON i.product_id = p.product_id;

B.SELECT * FROM Products p LEFT JOIN Inventory i ON p.product_id = i.product_id;

C.SELECT * FROM Products p CROSS JOIN (SELECT DISTINCT store_id FROM Inventory) s;

D.SELECT * FROM Inventory i RIGHT JOIN Products p ON i.product_id = p.product_id;

Correct Answer: C

Explanation: CROSS JOIN produces a Cartesian product of stores and products — ideal for generating all combinations.

Q5. RetailChain wants to find customers living in the same city as other customers. Which query can achieve this?

A.SELECT a.name, b.name FROM Customers a INNER JOIN Customers b ON a.city = b.city WHERE a.customer_id = b.customer_id;

B.SELECT a.name, b.name FROM Customers a INNER JOIN Customers b ON a.city = b.city WHERE a.customer_id <> b.customer_id;

C.SELECT a.name FROM Customers a LEFT JOIN Orders o ON a.customer_id = o.customer_id;

D.SELECT name FROM Customers WHERE city IN (SELECT DISTINCT city FROM Customers);

Correct Answer: B

Explanation: This is a SELF JOIN — comparing rows within the same table to find customers sharing the same city, excluding matches with themselves.

Case Study 2

RetailMart is a chain of retail stores that sells a variety of products, including groceries, electronics, and household items. The company stores its data in a relational database with the following simplified schema:

Key Tables:

- **Customers**
  - customer_id (INT, PK)
  - name (VARCHAR)
  - city (VARCHAR)
- **Orders**
  - order_id (INT, PK)
  - customer_id (INT, FK to Customers)
  - order_date (DATE)
- **OrderItems**
  - item_id (INT, PK)
  - order_id (INT, FK to Orders)
  - product_id (INT, FK to Products)
  - quantity (INT)
  - price (DECIMAL)

- **Products**
  - product_id (INT, PK)
  - product_name (VARCHAR)
  - category (VARCHAR)

RetailMart's analytics team needs to generate insights on customer behavior and product performance. They use **subqueries** to:

1. Identify top customers by revenue.
2. Find products never ordered.
3. Compare customer purchases with the average.
4. List cities with above-average order frequency.
5. Track orders that included specific categories of products.

Q1. Which query correctly finds customers whose total order value is above the average order value of all customers?

A. SELECT name FROM Customers WHERE customer_id IN (SELECT customer_id FROM Orders GROUP BY customer_id HAVING SUM(order_total) > AVG(order_total);

B. SELECT name FROM Customers WHERE customer_id IN (SELECT customer_id FROM Orders GROUP BY customer_id HAVING SUM((SELECT SUM(quantity * price) FROM OrderItems oi WHERE oi.order_id = o.order_id)) > (SELECT AVG(order_total) FROM Orders));

C. SELECT name FROM Customers WHERE customer_id IN (SELECT customer_id FROM Orders JOIN OrderItems USING(order_id)GROUP BY customer_id HAVING SUM(quantity * price) > (SELECT AVG(order_total) FROM (SELECT order_id, SUM(quantity * price) AS order_total FROM OrderItems GROUP BY order_id) AS order_sums));

D. SELECT name FROM Customers WHERE customer_id IN (SELECT customer_id FROM Orders WHERE order_total > (SELECT AVG(order_total) FROM Orders));

Answer: C
Explanation: This query correctly joins Orders and OrderItems, calculates total spend per customer, and compares it to the average using a subquery. Options A and B are syntactically incorrect or misstructured.

Q2. What type of subquery is used when a subquery refers to a column from the outer query?

A. Scalar subquery
B. Correlated subquery
C. Inline subquery
D. Nested join

Answer: B
Explanation: A **correlated subquery** depends on a value from the outer query. It's evaluated once for every row processed by the outer query.

Q3. Which query returns cities with above-average number of orders per customer?

A.

```
SELECT city
FROM Customers
WHERE city IN (
  SELECT city
  FROM Customers
  GROUP BY city
  HAVING COUNT(customer_id) > (
    SELECT AVG(order_count) FROM (
      SELECT customer_id, COUNT(*) AS order_count
      FROM Orders
      GROUP BY customer_id
    ) AS cust_orders
  )
);
```

B.

```
SELECT city
FROM Customers
GROUP BY city
HAVING COUNT(*) > (
  SELECT AVG(cnt) FROM (
    SELECT city, COUNT(*) AS cnt
    FROM Customers
    GROUP BY city
  ) AS city_counts
);
```

C.

```
SELECT city
FROM Customers
```

```
WHERE EXISTS (
  SELECT 1
  FROM Orders
  WHERE Orders.customer_id = Customers.customer_id
  GROUP BY city
  HAVING COUNT(*) > (
    SELECT AVG(cnt) FROM Orders GROUP BY customer_id
  )
);
```

D.

```
SELECT city
FROM (
  SELECT city, COUNT(o.order_id) AS order_count
  FROM Customers c
  JOIN Orders o ON c.customer_id = o.customer_id
  GROUP BY city
) city_orders
WHERE order_count > (
  SELECT AVG(order_count) FROM (
    SELECT c.city, COUNT(o.order_id) AS order_count
    FROM Customers c
    JOIN Orders o ON c.customer_id = o.customer_id
    GROUP BY c.city
  ) AS city_avg
);
```

Answer: D
Explanation: This query accurately calculates orders per city and compares it with the average. It uses **derived tables (subqueries in FROM clause)** for aggregation comparison.

Q4. How can you list customers who bought every product in the 'Electronics' category?

A.

```
SELECT name
FROM Customers
WHERE customer_id IN (
  SELECT customer_id
  FROM Orders o
  JOIN OrderItems oi ON o.order_id = oi.order_id
  WHERE oi.product_id IN (
    SELECT product_id FROM Products WHERE category = 'Electronics'
  )
);
```

B.

```
SELECT name
FROM Customers c
WHERE NOT EXISTS (
   SELECT product_id
   FROM Products
   WHERE category = 'Electronics'
   AND product_id NOT IN (
      SELECT oi.product_id
      FROM Orders o
      JOIN OrderItems oi ON o.order_id = oi.order_id
      WHERE o.customer_id = c.customer_id
   )
);
```

C.

```
SELECT name
FROM Customers
WHERE EXISTS (
   SELECT * FROM Products WHERE category = 'Electronics'
);
```

D.

```
SELECT name
FROM Customers
WHERE customer_id IN (
   SELECT DISTINCT customer_id
   FROM Orders
   JOIN OrderItems USING(order_id)
   WHERE category = 'Electronics'
);
```

Answer: B
Explanation: This query checks for **absence** of any 'Electronics' product not purchased by the customer—i.e., it uses **double negation logic** with NOT EXISTS and a **correlated subquery**.

Case Study 4

You are working with a database that contains the following tables:

Table: Employees

emp_id	name	department_id	salary
1	Alice	101	70000
2	Bob	102	50000
3	Charlie	101	60000
4	David	103	55000
5	Eve	102	75000

Table: Departments

department_id	department_name
101	HR
102	IT
103	Finance

MCQs

Q1. Which SQL query returns the names of employees who earn **more than the average salary** of all employees?

- **A.** SELECT name FROM Employees WHERE salary > ALL (SELECT salary FROM Employees);
- **B.** SELECT name FROM Employees WHERE salary > (SELECT AVG(salary) FROM Employees);
- **C.** SELECT name FROM Employees WHERE salary > (SELECT MAX(salary) FROM Employees);
- **D.** SELECT name FROM Employee WHERE salary IN (SELECT AVG(salary) FROM Employees);

Answer: B
Explanation: Option B correctly uses a scalar subquery to compare each employee's salary with the average salary of all employees.

Q2. Which query returns the names of employees who work in the **same department as 'Alice'?**

- **A.** SELECT name FROM Employees WHERE department_id = 101;
- **B.** SELECT name FROM Employees WHERE name = 'Alice';
- **C.** SELECT name FROM Employees WHERE department_id = (SELECT department_id FROM Employees WHERE name = 'Alice');
- **D.** SELECT name FROM Employees WHERE department_id IN (SELECT emp_id FROM Employees WHERE name = 'Alice');

Answer: C
Explanation: Option C uses a subquery to first get Alice's department_id, then finds all employees in that department.

Q3. Which query retrieves the departments **with at least one employee** earning more than 70,000?

A. SELECT department_name FROM Departments WHERE department_id IN (

SELECT department_id FROM Employees WHERE salary > 70000);

B. SELECT department_name FROM Departments WHERE department_id = (

SELECT department_id FROM Employees WHERE salary > 70000);

C. SELECT department_name FROM Departments WHERE EXISTS (SELECT * FROM Employees WHERE salary > 70000);
D. SELECT department_name FROM Departments WHERE department_id = 102;

Answer: A
Explanation: Option A uses a subquery to find department IDs with employees earning more than 70,000, and matches those against the Departments table.

Case Study 5
A retail company uses a SQL Server database to manage orders. To streamline operations, they use a stored procedure called ProcessOrder that:

- Accepts an OrderID as input
- Updates inventory quantities
- Calculates total cost
- Inserts an order record in the billing table
- Uses transactions to ensure data consistency

Q1: What is the primary reason to use a stored procedure like ProcessOrder in a database system?

A) To display data to users
B) To reduce application size
C) To encapsulate business logic and ensure atomic transactions
D) To reduce the number of columns in a table

Correct Answer: C
Explanation: Stored procedures encapsulate logic and allow complex operations (like inventory update and billing) to occur in a single, atomic transaction, improving consistency and maintainability.

Q2: In the ProcessOrder stored procedure, a transaction is used. Which SQL clause ensures that the transaction is rolled back in case of an error?

A) COMMIT
B) SAVEPOINT
C) THROW
D) ROLLBACK

Correct Answer: D
Explanation: The ROLLBACK statement is used to undo all changes made by the current transaction if an error occurs, ensuring data integrity.

Q3: Which benefit is achieved by using input parameters (e.g., @OrderID) in the ProcessOrder procedure?

A) Reduces storage requirements
B) Makes the procedure dynamic and reusable
C) Limits the use of SQL joins
D) Increases execution time

Correct Answer: B
Explanation: Parameters like @OrderID make stored procedures flexible and reusable for different inputs, supporting dynamic processing of orders.

Q4: If the ProcessOrder stored procedure updates multiple tables (e.g., inventory and billing), what database concept ensures that all updates succeed or none are applied?

A) Views
B) Triggers
C) Transactions
D) Indexes

Correct Answer: C
Explanation: Transactions ensure atomicity. If any part of the process fails, the entire transaction is rolled back, preventing partial data updates.

Q5: What will happen if you forget to use the COMMIT statement at the end of a successful ProcessOrder transaction?

A) The data will be automatically saved
B) The transaction will be committed after a timeout
C) The transaction may remain open and lock resources
D) The system will auto-correct and commit changes

Correct Answer: C
Explanation: Without COMMIT, the transaction remains open, potentially causing locks that block other operations. It must be explicitly committed to finalize changes.

Case Study 6

Suppose you have a table Employees:

EmployeeID	FirstName	LastName	Salary	HireDate
101	John	Doe	60000	2018-04-15
102	Jane	Smith	75000	2016-07-20
103	Mike	Johnson	50000	2019-11-01
104	Emily	Davis	85000	2015-02-28

Q1: Which SQL function would you use to calculate the total sum of all employees' salaries?

A) COUNT(Salary)
B) SUM(Salary)
C) AVG(Salary)
D) MAX(Salary)

Answer: B) SUM(Salary)
Explanation: SUM() adds all numeric values in a column, so SUM(Salary) gives the total salary paid.

Q2: You want to find the employee who has the highest salary. Which SQL function helps you find this?

A) MIN(Salary)
B) MAX(Salary)
C) AVG(Salary)
D) TOP(Salary)

Answer: B) MAX(Salary)
Explanation: MAX() returns the maximum value from a column, here it returns the highest salary.

Q3: Which SQL function would help you find how many employees have been hired?

A) SUM(HireDate)
B) COUNT(EmployeeID)
C) AVG(EmployeeID)
D) LENGTH(EmployeeID)

Answer: B) COUNT(EmployeeID)
Explanation: COUNT() returns the number of rows or non-null values in a column. Counting EmployeeID gives the number of employees.

MCQ 4: To get the length of each employee's first name, which function should be used?

A) LENGTH(FirstName)
B) COUNT(FirstName)
C) SUM(FirstName)
D) MAX(FirstName)

Answer: A) LENGTH(FirstName)
Explanation: LENGTH() returns the number of characters in a string column.

Q5: To calculate how many years each employee has worked (assuming today is 2025-05-29), which function would be appropriate?

A) YEAR(HireDate)
B) DATEDIFF(year, HireDate, GETDATE())
C) DATEADD(year, HireDate, GETDATE())
D) MONTH(HireDate)

Answer: B) DATEDIFF(year, HireDate, GETDATE())
Explanation: DATEDIFF() calculates the difference between two dates, here in years from hire date to current date.

Case Study 7

Imagine a simple e-commerce database with the following two tables:

1. Customers

CustomerID	Name
1	Alice
2	Bob
3	Charlie

2. Orders

OrderID	CustomerID	Product
101	1	Laptop
102	2	Headphones
103	4	Monitor

Note: CustomerID 4 in the Orders table does not exist in the Customers table.

Q1. Which type of JOIN will return only the records that have matching values in both Customers and Orders tables?

A) LEFT JOIN
B) RIGHT JOIN
C) INNER JOIN
D) FULL OUTER JOIN

Correct Answer: C) INNER JOIN
Explanation: INNER JOIN returns rows where there is a match in both tables. So, customers who have placed valid orders will be returned, excluding unmatched rows like CustomerID 4.

Q2. What will a LEFT JOIN from Customers to Orders return?

A) All orders, regardless of customer
B) Only customers with orders
C) All customers and their orders, if any
D) Only unmatched customers

Correct Answer: C) All customers and their orders, if any
Explanation: LEFT JOIN returns all rows from the left table (Customers) and the matched rows from the right table (Orders). If there is no match, NULLs are shown for order data.

Q3. What is the result of a RIGHT JOIN from Customers to Orders?

A) All customers only
B) All orders and matching customers
C) Only orders that match customers
D) Only unmatched orders

Correct Answer: B) All orders and matching customers
Explanation: RIGHT JOIN returns all rows from the right table (Orders) and the matching rows from Customers. This includes the order with CustomerID 4, which will show NULL for customer info.

Q4. Which JOIN type will return all customers and all orders, regardless of whether they match?

A) INNER JOIN
B) FULL OUTER JOIN
C) LEFT JOIN
D) RIGHT JOIN

Correct Answer: B) FULL OUTER JOIN
Explanation: FULL OUTER JOIN returns all rows from both tables, showing NULLs where there is no match. This captures both unmatched customers and unmatched orders.

Q5. How can we find customers who have not placed any orders using a JOIN?

A) Use INNER JOIN and filter NULL orders
B) Use RIGHT JOIN and filter NULL customers
C) Use LEFT JOIN and filter NULL in OrderID
D) Use FULL OUTER JOIN and filter NULL customers

Correct Answer: C) Use LEFT JOIN and filter NULL in OrderID
Explanation: LEFT JOIN gets all customers. Filtering for NULLs in OrderID isolates customers who did not place any order.

Case Study 8

You are a database developer at **FreshMart**, a supermarket chain that manages inventory, sales, and employee data using a relational database. The company has created **views** to simplify access for reporting teams and ensure data abstraction for security and performance.

Q1. FreshMart has created a view named TopSellingProducts that shows only the top 10 selling products. Why would the company prefer to use a view instead of giving direct access to the sales table?

A. To increase disk space usage
B. To restrict access to sensitive or irrelevant data
C. To allow deletion of original sales data
D. To improve hardware performance

Answer: B. To restrict access to sensitive or irrelevant data

Explanation:
Views can simplify and secure data access by exposing only relevant columns and rows, preventing unauthorized users from seeing the full underlying table.

Q2. FreshMart has a view ActiveEmployees created as:

```
CREATE VIEW ActiveEmployees AS
SELECT emp_id, name, department FROM Employees WHERE status = 'Active';
```

Can this view be used to update an employee's department?

A. Yes, because it only filters rows, not columns
B. No, views cannot be updated
C. Yes, because it contains a GROUP BY clause
D. No, because it is a read-only view due to filters

Answer: A. Yes, because it only filters rows, not columns

Explanation:
Views that are based on a single table and contain no joins or aggregation can generally be updated, provided they don't include complex clauses like GROUP BY, DISTINCT, etc.

Q3. FreshMart noticed their regular view DailySalesSummary is slow to load during business hours. What kind of view could improve performance by storing the query result?

A. Standard View
B. Recursive View
C. Materialized View
D. Temporary View

Answer: C. Materialized View

Explanation:
A materialized view stores the result of a query physically and can be refreshed periodically, significantly improving read performance for heavy queries.

Q4. What will happen if FreshMart drops the underlying Products table that a view TopProducts depends on?

A. The view will automatically update its source
B. The view will continue to work with old data
C. The view will become invalid and raise an error when queried
D. The database will create a new table automatically

Answer: C. The view will become invalid and raise an error when queried

Explanation:
Views are dependent on their underlying tables. If a base table is dropped, any dependent views will be invalidated and fail when accessed.

Q5. Why might FreshMart grant report analysts access to a view rather than the original Employees table?

A. To reduce database size
B. To ensure analysts can change employee records
C. To prevent accidental data deletion
D. To provide controlled, read-only access to selected data

Answer: D. To provide controlled, read-only access to selected data

Explanation:
Views can expose only necessary data and prevent access to sensitive fields like salaries or personal details, especially when combined with read-only permissions.