**Scenario 1: Employee Management System**

A company maintains an **Employee** database with the following tables:

**Tables & Structure:**

1. **Employees**
   * EmployeeID (Primary Key)
   * Name
   * DepartmentID (Foreign Key)
   * Salary
   * JoiningDate
2. **Departments**
   * DepartmentID (Primary Key)
   * DepartmentName

**Questions:**

1. **DDL - Table Creation & Modification:**
   * Write a query to create both tables with appropriate constraints.
   * Add a new column **Email** to the **Employees** table.
   * Remove the **JoiningDate** column from the **Employees** table.
2. **DML - Data Insertion & Modification:**
   * Insert at least **5 employees** and **3 departments** into the tables.
   * Update the salary of employees in the "HR" department by **10%**.
   * Delete an employee who has the lowest salary.
3. **Joins - Retrieving Data:**
   * Retrieve the **employee name, department name, and salary** using a join.
   * Get the **highest-paid employee** in each department.
   * Find departments that **do not have any employees**.
4. **Aggregate Functions & Built-in Functions:**
   * Find the **average salary** in each department.
   * Get the total number of employees in each department.
   * Retrieve employees whose **name starts with 'A'** (using a built-in function).
5. **Stored Procedure & Trigger:**
   * Write a **stored procedure** to retrieve all employees with a salary **above a given amount**.
   * Create a **trigger** that automatically updates a log table whenever an employee’s salary is updated.

**Scenario 2: Online Shopping System**

An e-commerce platform maintains two tables:

**Tables & Structure:**

1. **Orders**
   * OrderID (Primary Key)
   * CustomerName
   * OrderDate
   * TotalAmount
2. **OrderDetails**
   * OrderDetailID (Primary Key)
   * OrderID (Foreign Key)
   * ProductName
   * Quantity
   * Price

**Questions:**

1. **DDL - Table Creation & Modification:**
   * Create both tables with appropriate constraints.
   * Modify the **Orders** table to add a new column **PaymentStatus (Paid/Pending)**.
2. **DML - Insert & Update:**
   * Insert **at least 3 orders** and their corresponding order details.
   * Update the **total amount** for a specific order by adding an extra charge.
3. **Joins - Data Retrieval:**
   * Retrieve all orders along with their order details.
   * Find the total number of products ordered in each order.
4. **Aggregate & Built-in Functions:**
   * Get the **total revenue** for all orders.
   * Find the **most expensive product** ordered.
   * Extract the **month from OrderDate** using a built-in function.
5. **Stored Procedure & Trigger:**
   * Create a **stored procedure** to get all orders placed in a specific date range.
   * Write a **trigger** that prevents orders from being placed if TotalAmount is **zero**.

**Scenario 3: Student Result System**

A university maintains a **student result** database with the following tables:

**Tables & Structure:**

1. **Students**
   * StudentID (Primary Key)
   * Name
   * Course
   * JoinYear
2. **Results**
   * ResultID (Primary Key)
   * StudentID (Foreign Key)
   * Subject
   * Marks

**Questions:**

1. **DDL - Table Creation & Modification:**
   * Create both tables with constraints.
   * Modify the **Results** table to add a **Grade** column.
2. **DML - Insert & Update:**
   * Insert **5 students** and their corresponding results.
   * Update grades based on marks (**A: 90+, B: 75-89, C: 50-74, F: <50**).
3. **Joins - Retrieving Data:**
   * Retrieve the **student names along with their subjects and marks**.
   * Find students who have **not received any marks**.
4. **Aggregate & Built-in Functions:**
   * Find the **highest and lowest marks** in each subject.
   * Get the **average marks** for each student.
   * Display the **year from JoinYear** using a built-in function.
5. **Stored Procedure & Trigger:**
   * Write a **stored procedure** to get students who have failed in at least one subject.
   * Create a **trigger** that prevents marks from being inserted if they are greater than **100**.