# SQL Assignment Questions and Answers

## Set 2

Q1 (10 marks): Create Orders and Customers tables and retrieve customer order history using different types of JOINS.

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100)
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- INNER JOIN Example
SELECT Customers.CustomerName, Orders.OrderDate
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Q2 (15 marks): Build a SalesRecord table and apply aggregate functions to summarize sales by region or category.

```
CREATE TABLE SalesRecord (
    SaleID INT PRIMARY KEY,
    Region VARCHAR(50),
    Category VARCHAR(50),
    Amount DECIMAL(10,2)
);

-- Sample aggregate query
SELECT Region, SUM(Amount) AS TotalSales
FROM SalesRecord
GROUP BY Region;
```

Q3 (10 marks): Perform UPDATE on a sales table to increase all sales values by 10%.

```
UPDATE SalesRecord
SET Amount = Amount * 1.10;
```

## Set 5

Q1 (15 marks): Create a student record system with columns for StudentID, Name, Marks, and Grade. Insert sample data and demonstrate SELECT queries with filtering.

```
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(100),
    Marks INT,
    Grade CHAR(1)
```

```
);

-- Sample SELECT with filtering
SELECT * FROM Students WHERE Marks > 70;
```

Q2 (15 marks): Create a table with a composite primary key and demonstrate how it enforces uniqueness across combined columns. Insert valid/invalid data to test.

```
CREATE TABLE CourseRegistrations (
    StudentID INT,
    CourseID INT,
    RegistrationDate DATE,
    PRIMARY KEY (StudentID, CourseID)
);

-- Insert valid record
INSERT INTO CourseRegistrations VALUES (1, 101, '2023-06-01');

-- Insert duplicate (will fail)
-- INSERT INTO CourseRegistrations VALUES (1, 101, '2023-06-02');
```

Q3 (10 marks): Demonstrate LEFT JOIN on Departments and Employees tables to list all departments with or without employees.

```
CREATE TABLE Departments (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100)
);

CREATE TABLE Employees (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(100),
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);

-- LEFT JOIN Example
SELECT Departments.DeptName, Employees.EmpName
FROM Departments
LEFT JOIN Employees ON Departments.DeptID = Employees.DeptID;
```

### Set 7

Q1 (15 marks): Build a database with EmployeeID, Name, Salary, Department, and ActiveStatus. Demonstrate use of WHERE clause in SELECT statements.

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    Salary DECIMAL(10,2),
```

```
    Department VARCHAR(50),
    ActiveStatus BOOLEAN
);

-- WHERE clause example
SELECT * FROM Employees WHERE ActiveStatus = TRUE;
```

Q2 (15 marks): Create a table using a primary key and a foreign key referencing another table. Show how to insert and reject invalid foreign keys.

```
CREATE TABLE Departments (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100)
);

CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);

-- Valid insert
INSERT INTO Departments VALUES (1, 'HR');
INSERT INTO Employees VALUES (101, 'John Doe', 1);

-- Invalid insert (DeptID does not exist)
-- INSERT INTO Employees VALUES (102, 'Jane Smith', 2);
```

Q3 (10 marks): Show usage of DEFAULT value by inserting new records into a table without specifying all fields.

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Stock INT DEFAULT 100
);

-- Insert with default
INSERT INTO Products (ProductID, ProductName) VALUES (1, 'Notebook');
```