SQL Assignments - Answers with Questions

Set 9 - Q1

Create a sales table and calculate total and average sales.

```
CREATE TABLE Sales (
    ProductName VARCHAR(50),
    QuantitySold INT,
    UnitPrice DECIMAL(10, 2)
);

INSERT INTO Sales VALUES ('Pen', 10, 5.00), ('Notebook', 5, 20.00), ('Eraser', 15, 2.00);

SELECT ProductName, QuantitySold * UnitPrice AS TotalSales FROM Sales;
SELECT AVG(QuantitySold * UnitPrice) AS AverageSales FROM Sales;
```

Set 9 - Q2

Create a table of customers and orders with foreign key relationships and show INNER JOIN results.

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(50)
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    Product VARCHAR(50),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

INSERT INTO Customers VALUES (1, 'John'), (2, 'Alice');
INSERT INTO Orders VALUES (101, 1, 'Pen'), (102, 2, 'Notebook');

SELECT Customers.Name, Orders.Product
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Set 9 - Q3

Perform DELETE operations on rows in a customer table based on a condition and show the result.

```
DELETE FROM Customers WHERE Name = 'John';
SELECT * FROM Customers;
```

Set 10 - Q1

SQL Assignments - Answers with Questions

Create a Movies table and perform queries to update movie ratings and delete movies based on release year.

```
CREATE TABLE Movies (
    MovieID INT PRIMARY KEY,
    Title VARCHAR(100),
    Rating DECIMAL(3,1),
    ReleaseYear INT
);

INSERT INTO Movies VALUES (1, 'Movie A', 7.5, 2020), (2, 'Movie B', 6.0, 2015);

UPDATE Movies SET Rating = 8.5 WHERE Title = 'Movie A';
DELETE FROM Movies WHERE ReleaseYear < 2018;
```

Set 10 - Q2

Create an Inventory table with CHECK constraint on quantity and use SELECT to display products with low stock.

```
CREATE TABLE Inventory (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(50),
    Quantity INT CHECK (Quantity >= 0)
);

INSERT INTO Inventory VALUES (1, 'Pen', 3), (2, 'Book', 0), (3, 'Eraser', 5);

SELECT * FROM Inventory WHERE Quantity < 5;
```

Set 10 - Q3

Update marks in a student table where the score is below 40. Show updated table.

```
CREATE TABLE Students (
    StudentID INT,
    Name VARCHAR(50),
    Marks INT
);

INSERT INTO Students VALUES (1, 'Ravi', 35), (2, 'Meena', 45);
UPDATE Students SET Marks = 40 WHERE Marks < 40;
SELECT * FROM Students;
```

Set 16 - Q1

Create and drop a table using DDL. Insert, update and delete data using DML. Show SELECT queries after each operation to verify.

```
CREATE TABLE Sample (
    ID INT PRIMARY KEY,
    Name VARCHAR(50)
```

```
);

INSERT INTO Sample VALUES (1, 'A'), (2, 'B');
UPDATE Sample SET Name = 'Updated A' WHERE ID = 1;
DELETE FROM Sample WHERE ID = 2;
DROP TABLE Sample;
```

Set 16 - Q2

```
Create a Product table with constraints like UNIQUE and CHECK on price. Insert values to validate the constraints work correctly.
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(50) UNIQUE,
    Price DECIMAL(10,2) CHECK (Price > 0)
);

INSERT INTO Product VALUES (1, 'Pen', 10.0);
/* Invalid Inserts to test constraints:
INSERT INTO Product VALUES (2, 'Pen', 10.0); -- violates UNIQUE
INSERT INTO Product VALUES (3, 'Notebook', -5.0); -- violates CHECK */
```

Set 16 - Q3

```
Test CHECK constraint on a Grade column in a student table by inserting valid/invalid grades.
CREATE TABLE StudentGrades (
    StudentID INT,
    Name VARCHAR(50),
    Grade CHAR(1) CHECK (Grade IN ('A', 'B', 'C', 'D', 'F'))
);

INSERT INTO StudentGrades VALUES (1, 'John', 'A');
/* Invalid Insert:
INSERT INTO StudentGrades VALUES (2, 'Alice', 'Z'); -- violates CHECK */
```