# SQL Solutions for All Question Sets

## Set 1 Solutions

### Q1: Employee Attendance Table with UPDATE Queries

```sql

-- Create Employee Attendance table

CREATE TABLE EmployeeAttendance (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

AttendanceDate DATE,

Status VARCHAR(20) CHECK (Status IN ('Present', 'Absent', 'Late', 'Leave'))

);

-- Insert sample data

INSERT INTO EmployeeAttendance VALUES

(1, 'John Doe', '2023-05-01', 'Present'),

(2, 'Jane Smith', '2023-05-01', 'Absent'),

(3, 'Mike Johnson', '2023-05-01', 'Late');

-- Update attendance based on conditions

UPDATE EmployeeAttendance

SET Status = 'Leave'

WHERE EmployeeID = 2 AND AttendanceDate = '2023-05-01';

-- Show final results

SELECT \* FROM EmployeeAttendance;

```

### Q2: Product Table with Constraints

```sql

-- Create Product table with constraints

CREATE TABLE Product (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100) UNIQUE,

Category VARCHAR(50),

Price DECIMAL(10,2) CHECK (Price > 0),

StockQuantity INT

);

-- Insert valid data

INSERT INTO Product VALUES

(1, 'Laptop', 'Electronics', 999.99, 10),

(2, 'Smartphone', 'Electronics', 699.99, 15);

-- This will fail due to CHECK constraint

INSERT INTO Product VALUES (3, 'Tablet', 'Electronics', -100, 5);

-- This will fail due to UNIQUE constraint

INSERT INTO Product VALUES (4, 'Laptop', 'Electronics', 899.99, 8);

```

### Q3: RIGHT JOIN on Customers and Orders

```sql

-- Create Customers and Orders tables

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

CustomerName VARCHAR(100),

Email VARCHAR(100)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

Amount DECIMAL(10,2)

);

-- Insert sample data

INSERT INTO Customers VALUES (1, 'Alice', 'alice@example.com'), (2, 'Bob', 'bob@example.com');

INSERT INTO Orders VALUES (101, 1, '2023-05-01', 150.00), (102, 3, '2023-05-02', 200.00);

-- RIGHT JOIN to show all orders including unknown customers

SELECT o.OrderID, o.OrderDate, o.Amount, c.CustomerName

FROM Customers c

RIGHT JOIN Orders o ON c.CustomerID = o.CustomerID;

```

## Set 2 Solutions

### Q1: Orders and Customers Tables with JOINs

```sql

-- Create Customers and Orders tables

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

Amount DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Insert sample data

INSERT INTO Customers VALUES

(1, 'John Smith', 'john@example.com'),

(2, 'Sarah Johnson', 'sarah@example.com');

INSERT INTO Orders VALUES

(101, 1, '2023-05-01', 99.99),

(102, 1, '2023-05-15', 149.99),

(103, 2, '2023-05-10', 79.99);

-- INNER JOIN

SELECT c.Name, o.OrderID, o.OrderDate, o.Amount

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID;

-- LEFT JOIN

SELECT c.Name, o.OrderID, o.OrderDate, o.Amount

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;

-- RIGHT JOIN

SELECT c.Name, o.OrderID, o.OrderDate, o.Amount

FROM Customers c

RIGHT JOIN Orders o ON c.CustomerID = o.CustomerID;

```

### Q2: SalesRecord Table with Aggregate Functions

```sql

-- Create SalesRecord table

CREATE TABLE SalesRecord (

SaleID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Region VARCHAR(50),

SaleAmount DECIMAL(10,2),

SaleDate DATE

);

-- Insert sample data

INSERT INTO SalesRecord VALUES

(1, 'Laptop', 'Electronics', 'North', 999.99, '2023-05-01'),

(2, 'Smartphone', 'Electronics', 'South', 699.99, '2023-05-02'),

(3, 'Desk', 'Furniture', 'North', 249.99, '2023-05-01'),

(4, 'Chair', 'Furniture', 'East', 149.99, '2023-05-03');

-- Summarize sales by region

SELECT Region, SUM(SaleAmount) AS TotalSales, AVG(SaleAmount) AS AverageSale

FROM SalesRecord

GROUP BY Region;

-- Summarize sales by category

SELECT Category, SUM(SaleAmount) AS TotalSales, COUNT(\*) AS NumberOfSales

FROM SalesRecord

GROUP BY Category;

```

### Q3: UPDATE Sales Table with 10% Increase

```sql

-- Create sales table

CREATE TABLE Sales (

SaleID INT PRIMARY KEY,

ProductName VARCHAR(100),

SaleAmount DECIMAL(10,2)

);

-- Insert sample data

INSERT INTO Sales VALUES

(1, 'Laptop', 1000.00),

(2, 'Phone', 500.00),

(3, 'Tablet', 300.00);

-- Update all sales values by 10%

UPDATE Sales

SET SaleAmount = SaleAmount \* 1.10;

-- Display results

SELECT \* FROM Sales;

```

## Set 3 Solutions

### Q1: Books Table with SELECT Queries

```sql

-- Create Books table

CREATE TABLE Books (

BookID INT PRIMARY KEY,

Title VARCHAR(100),

Author VARCHAR(100),

Genre VARCHAR(50),

Price DECIMAL(10,2),

PublishedYear INT

);

-- Insert sample data

INSERT INTO Books VALUES

(1, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Classic', 12.99, 1925),

(2, 'To Kill a Mockingbird', 'Harper Lee', 'Fiction', 10.99, 1960),

(3, '1984', 'George Orwell', 'Dystopian', 9.99, 1949),

(4, 'The Hobbit', 'J.R.R. Tolkien', 'Fantasy', 14.99, 1937);

-- SELECT with WHERE clause

SELECT \* FROM Books WHERE Genre = 'Fiction';

-- SELECT with ORDER BY

SELECT \* FROM Books ORDER BY PublishedYear DESC;

-- SELECT with LIMIT

SELECT \* FROM Books ORDER BY Price DESC LIMIT 2;

```

### Q2: Employee Attendance Table with UPDATE (Same as Set 1 Q1)

```sql

-- Same solution as Set 1 Q1

```

### Q3: SUM and COUNT on Product Prices and Quantities

```sql

-- Create Products table

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Price DECIMAL(10,2),

Quantity INT

);

-- Insert sample data

INSERT INTO Products VALUES

(1, 'Laptop', 999.99, 5),

(2, 'Phone', 699.99, 10),

(3, 'Tablet', 299.99, 8);

-- Calculate total value of inventory

SELECT SUM(Price \* Quantity) AS TotalInventoryValue FROM Products;

-- Count number of products

SELECT COUNT(\*) AS TotalProducts FROM Products;

```

## Set 5 Solutions

### Q1: Student Record System

```sql

-- Create student table

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(100),

Marks INT,

Grade VARCHAR(2)

);

-- Insert sample data

INSERT INTO Students VALUES

(1, 'Alice Johnson', 85, 'A'),

(2, 'Bob Smith', 72, 'B'),

(3, 'Charlie Brown', 55, 'D'),

(4, 'Diana Prince', 92, 'A');

-- SELECT with filtering

SELECT \* FROM Students WHERE Marks >= 70;

SELECT \* FROM Students WHERE Grade = 'A' ORDER BY Marks DESC;

SELECT Name, Marks FROM Students WHERE Marks BETWEEN 50 AND 75;

```

### Q2: Table with Composite Primary Key

```sql

-- Create table with composite primary key

CREATE TABLE OrderItems (

OrderID INT,

ProductID INT,

Quantity INT,

PRIMARY KEY (OrderID, ProductID)

);

-- Insert valid data

INSERT INTO OrderItems VALUES (101, 1, 2), (101, 2, 1), (102, 1, 3);

-- This will fail due to duplicate composite key

INSERT INTO OrderItems VALUES (101, 1, 5);

```

### Q3: LEFT JOIN on Departments and Employees

```sql

-- Create Departments and Employees tables

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)

);

-- Insert sample data

INSERT INTO Departments VALUES (1, 'HR'), (2, 'IT'), (3, 'Finance');

INSERT INTO Employees VALUES (101, 'John', 1), (102, 'Sarah', 2), (103, 'Mike', 1);

-- LEFT JOIN to list all departments with or without employees

SELECT d.DeptName, e.EmpName

FROM Departments d

LEFT JOIN Employees e ON d.DeptID = e.DeptID;

```

## Set 9 Solutions

### Q1: Sales Table with Aggregate Functions

```sql

-- Create sales table

CREATE TABLE Sales (

SaleID INT PRIMARY KEY,

ProductName VARCHAR(100),

QuantitySold INT,

UnitPrice DECIMAL(10,2)

);

-- Insert sample data

INSERT INTO Sales VALUES

(1, 'Laptop', 5, 999.99),

(2, 'Phone', 10, 699.99),

(3, 'Tablet', 8, 299.99);

-- Calculate total and average sales

SELECT

SUM(QuantitySold \* UnitPrice) AS TotalSales,

AVG(QuantitySold \* UnitPrice) AS AverageSale,

SUM(QuantitySold) AS TotalUnitsSold

FROM Sales;

```

### Q2: Customers and Orders with Foreign Key and INNER JOIN

```sql

-- Create Customers and Orders tables with foreign key

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

CustomerName VARCHAR(100),

Email VARCHAR(100)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

Amount DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Insert sample data

INSERT INTO Customers VALUES (1, 'Alice', 'alice@example.com'), (2, 'Bob', 'bob@example.com');

INSERT INTO Orders VALUES (101, 1, '2023-05-01', 150.00), (102, 2, '2023-05-02', 200.00);

-- INNER JOIN results

SELECT c.CustomerName, o.OrderID, o.OrderDate, o.Amount

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID;

```

### Q3: DELETE Operations on Customer Table

```sql

-- Create customer table

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

CustomerName VARCHAR(100),

LastPurchaseDate DATE,

IsActive BOOLEAN

);

-- Insert sample data

INSERT INTO Customers VALUES

(1, 'Alice', '2023-05-01', TRUE),

(2, 'Bob', '2022-12-15', TRUE),

(3, 'Charlie', '2021-08-20', FALSE);

-- DELETE inactive customers

DELETE FROM Customers WHERE IsActive = FALSE;

-- Show result

SELECT \* FROM Customers;

```

## Set 10 Solutions

### Q1: Movies Table with UPDATE and DELETE

```sql

-- Create Movies table

CREATE TABLE Movies (

MovieID INT PRIMARY KEY,

Title VARCHAR(100),

Director VARCHAR(100),

ReleaseYear INT,

Rating DECIMAL(3,1)

);

-- Insert sample data

INSERT INTO Movies VALUES

(1, 'Inception', 'Christopher Nolan', 2010, 8.8),

(2, 'The Shawshank Redemption', 'Frank Darabont', 1994, 9.3),

(3, 'Avatar', 'James Cameron', 2009, 7.8),

(4, 'Titanic', 'James Cameron', 1997, 7.9);

-- Update movie ratings

UPDATE Movies

SET Rating = Rating + 0.2

WHERE Director = 'Christopher Nolan';

-- Delete movies based on release year

DELETE FROM Movies WHERE ReleaseYear < 2000;

-- Show results

SELECT \* FROM Movies;

```

### Q2: Inventory Table with CHECK Constraint

```sql

-- Create Inventory table with CHECK constraint

CREATE TABLE Inventory (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Quantity INT CHECK (Quantity >= 0),

Category VARCHAR(50)

);

-- Insert sample data

INSERT INTO Inventory VALUES

(1, 'Laptop', 5, 'Electronics'),

(2, 'Phone', 0, 'Electronics'),

(3, 'Desk', 10, 'Furniture');

-- This will fail due to CHECK constraint

INSERT INTO Inventory VALUES (4, 'Chair', -5, 'Furniture');

-- Display products with low stock

SELECT \* FROM Inventory WHERE Quantity < 5;

```

### Q3: Update Marks in Student Table

```sql

-- Create student table

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(100),

Marks INT

);

-- Insert sample data

INSERT INTO Students VALUES

(1, 'Alice', 35),

(2, 'Bob', 42),

(3, 'Charlie', 38),

(4, 'Diana', 55);

-- Update marks below 40

UPDATE Students

SET Marks = Marks + 5

WHERE Marks < 40;

-- Show updated table

SELECT \* FROM Students;

```