Introduction to SQL and Advanced Functions Assignment Answers

Question 1: Explain the fundamental differences between DDL, DML, and DQL commands in SQL. Provide one example for each.

Answer: SQL commands are categorized based on the type of operation they perform.

DDL defines database structure.

```
CREATE TABLE Students (
  StudentID INT PRIMARY KEY,
  StudentName VARCHAR(50)
);
```

DML manipulates data inside tables.

```
INSERT INTO Students VALUES (1, 'Rahul');
```

DQL retrieves data from tables.

```
SELECT * FROM Students;
```

Question 2: What is the purpose of SQL constraints? Explain any three constraints with examples.

Answer: Constraints enforce rules to maintain accuracy and integrity of data.

Primary Key uniquely identifies records.

```
CustomerID INT PRIMARY KEY
```

Foreign Key maintains relationships.

```
CategoryID INT REFERENCES Categories(CategoryID)
```

Unique constraint prevents duplicate values.

```
Email VARCHAR(100) UNIQUE
```

Question 3: Explain LIMIT and OFFSET clauses with example.

Answer: LIMIT specifies number of rows, OFFSET specifies number of rows to skip.

```
SELECT * FROM Products
LIMIT 10 OFFSET 20;
```

Question 4: What is a Common Table Expression (CTE)? Explain with example.

Answer: A CTE is a temporary result set defined using WITH clause.

```
WITH ExpensiveProducts AS (
  SELECT ProductName, Price
  FROM Products
  WHERE Price > 100
)
SELECT * FROM ExpensiveProducts;
```

Question 5: Explain SQL Normalization and first three normal forms.

Answer: Normalization reduces redundancy and improves data integrity.

1NF: Atomic values. 2NF: No partial dependency. 3NF: No transitive dependency.

Question 6: Create ECommerceDB database, tables, and insert records.

```
CREATE DATABASE ECommerceDB;
USE ECommerceDB;

CREATE TABLE Categories (
  CategoryID INT PRIMARY KEY,
  CategoryName VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE Products (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(100) NOT NULL UNIQUE,
  CategoryID INT,
  Price DECIMAL(10,2) NOT NULL,
  StockQuantity INT,
  FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

CREATE TABLE Customers (
  CustomerID INT PRIMARY KEY,
  CustomerName VARCHAR(100) NOT NULL,
  Email VARCHAR(100) UNIQUE,
  JoinDate DATE
);

CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  CustomerID INT,
  OrderDate DATE NOT NULL,
  TotalAmount DECIMAL(10,2),
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Question 7: Generate report showing customer name, email, and total number of orders.

```
SELECT
  c.CustomerName,
  c.Email,
  COUNT(o.OrderID) AS TotalNumberOfOrders
FROM Customers c
LEFT JOIN Orders o
ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerName, c.Email
ORDER BY c.CustomerName;
```

Question 8: Retrieve product information with category.

```
SELECT
  p.ProductName,
  p.Price,
  p.StockQuantity,
  c.CategoryName
FROM Products p
JOIN Categories c
ON p.CategoryID = c.CategoryID
ORDER BY c.CategoryName, p.ProductName;
```

Question 9: Display top 2 most expensive products in each category.

```
WITH RankedProducts AS (
  SELECT
    c.CategoryName,
    p.ProductName,
    p.Price,
    ROW_NUMBER() OVER (
      PARTITION BY c.CategoryName
      ORDER BY p.Price DESC
    ) AS rn
  FROM Products p
  JOIN Categories c
  ON p.CategoryID = c.CategoryID
)
SELECT CategoryName, ProductName, Price
FROM RankedProducts
WHERE rn <= 2;
```

Question 10: Sakila Database – Top 5 customers by spending.

```
SELECT CONCAT(c.first_name,' ',c.last_name) AS CustomerName,
       c.email,
       SUM(p.amount) AS TotalSpent
FROM customer c
JOIN payment p ON c.customer_id = p.customer_id
GROUP BY c.customer_id
ORDER BY TotalSpent DESC
LIMIT 5;
```