Introduction to SQL and Advanced Functions - Assignment Solutions

```
Question 1:
DDL (Data Definition Language) defines or modifies database structure (CREATE, ALTER, DROP).
DML (Data Manipulation Language) manipulates data (INSERT, UPDATE, DELETE).
DQL (Data Query Language) retrieves data (SELECT).

Example:
DDL: CREATE TABLE Students (ID INT PRIMARY KEY, Name VARCHAR(50));
DML: INSERT INTO Students VALUES (1, 'John');
DQL: SELECT * FROM Students;

------------------------------------------------------------

Question 2:
SQL Constraints ensure data integrity and accuracy.

1. PRIMARY KEY – Uniquely identifies each record.
2. FOREIGN KEY – Maintains referential integrity.
3. UNIQUE – Ensures all values in a column are different.

------------------------------------------------------------

Question 3:
LIMIT restricts number of rows returned.
OFFSET skips specified number of rows.

Third page (10 records per page):
SELECT * FROM table_name
LIMIT 10 OFFSET 20;

------------------------------------------------------------

Question 4:
CTE (Common Table Expression) is a temporary result set defined using WITH clause.

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

------------------------------------------------------------

Question 5:
Normalization organizes data to reduce redundancy and improve integrity.

1NF: No repeating groups, atomic values.
2NF: No partial dependency.
3NF: No transitive dependency.

------------------------------------------------------------

Question 6:

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)
);

-- Insert statements included as per assignment.

------------------------------------------------------------

Question 7:

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:

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:

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:

1. Top 5 Customers:
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;

2. Top 3 Categories by Rental Count:
SELECT cat.name, COUNT(r.rental_id) AS RentalCount
FROM category cat
JOIN film_category fc ON cat.category_id = fc.category_id
JOIN inventory i ON fc.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY cat.name
ORDER BY RentalCount DESC
LIMIT 3;
```

3. Films per store and never rented:

```
SELECT s.store_id,
       COUNT(i.inventory_id) AS TotalFilms,
       SUM(CASE WHEN r.rental_id IS NULL THEN 1 ELSE 0 END) AS NeverRented
FROM store s
JOIN inventory i ON s.store_id = i.store_id
LEFT JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY s.store_id;
```

4. Monthly Revenue 2023:

```
SELECT MONTH(payment_date) AS Month,
       SUM(amount) AS TotalRevenue
FROM payment
WHERE YEAR(payment_date) = 2023
GROUP BY MONTH(payment_date)
ORDER BY Month;
```

5. Customers rented more than 10 times in last 6 months:

```
SELECT c.customer_id,
       CONCAT(c.first_name,' ',c.last_name) AS CustomerName,
       COUNT(r.rental_id) AS RentalCount
FROM customer c
JOIN rental r ON c.customer_id = r.customer_id
WHERE r.rental_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
GROUP BY c.customer_id
HAVING COUNT(r.rental_id) > 10;
```