Introduction to SQL and Advanced Functions – Final Assignment (Q1–Q10)

Question 1:

DDL defines structure (CREATE, ALTER, DROP), DML manipulates data (INSERT, UPDATE, DELETE), and DQL retrieves data (SELECT). Examples: DDL: CREATE TABLE Students (...); DML: INSERT INTO Students VALUES (...); DQL: SELECT * FROM Students;

Question 2:

SQL constraints maintain data integrity. PRIMARY KEY uniquely identifies records. FOREIGN KEY maintains relationships between tables. UNIQUE ensures all values in a column are different.

Question 3:

LIMIT restricts number of rows returned and OFFSET skips rows. Example (3rd page, 10 records per page): SELECT * FROM table_name LIMIT 10 OFFSET 20;

Question 4:

A Common Table Expression (CTE) is a temporary named result set created using WITH. It improves readability and simplifies complex queries. Example: WITH Temp AS (SELECT * FROM Employees) SELECT * FROM Temp;

Question 5:

Normalization reduces redundancy. 1NF: 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 Records
INSERT INTO Categories VALUES
(1,'Electronics'),(2,'Books'),(3,'Home Goods'),(4,'Apparel');
```

```
INSERT INTO Products VALUES
(101,'Laptop Pro',1,1200.00,50),
(102,'SQL Handbook',2,45.50,200),
(103,'Smart Speaker',1,99.99,150),
(104,'Coffee Maker',3,75.00,80),
(105,'Novel: The Great SQL',2,25.00,120),
(106,'Wireless Earbuds',1,150.00,100),
(107,'Blender X',3,120.00,60),
(108,'T-Shirt Casual',4,20.00,300);

INSERT INTO Customers VALUES
(1,'Alice Wonderland','alice@example.com','2023-01-10'),
(2,'Bob the Builder','bob@example.com','2022-11-25'),
(3,'Charlie Chaplin','charlie@example.com','2023-03-01'),
(4,'Diana Prince','diana@example.com','2021-04-26');

INSERT INTO Orders VALUES
(1001,1,'2023-04-26',1245.50),
(1002,2,'2023-10-12',99.99),
(1003,1,'2023-07-01',145.00),
(1004,3,'2023-01-14',150.00),
(1005,2,'2023-09-24',120.00),
(1006,1,'2023-06-19',20.00);
```

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

Question 10:

```
-- 1. Top 5 customers by spending
SELECT c.first_name, c.last_name, c.email, SUM(p.amount) AS total_spent
FROM customer c
JOIN payment p ON c.customer_id = p.customer_id
GROUP BY c.customer_id
ORDER BY total_spent DESC
LIMIT 5;

-- 2. Top 3 movie categories by rentals
SELECT cat.name AS category_name, COUNT(r.rental_id) AS rental_count
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 rental_count DESC
LIMIT 3;

-- 3. Films available per store & never rented
SELECT s.store_id,
COUNT(i.inventory_id) AS total_films,
SUM(CASE WHEN r.rental_id IS NULL THEN 1 ELSE 0 END) AS never_rented
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 for 2023
SELECT MONTH(payment_date) AS month, SUM(amount) AS revenue
FROM payment
WHERE YEAR(payment_date) = 2023
GROUP BY MONTH(payment_date);

-- 5. Customers with more than 10 rentals in last 6 months
SELECT c.customer_id, c.first_name, c.last_name, COUNT(r.rental_id) AS rental_count
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;
```