Experiment No:5

Objective:

1. To design a database schema for a simple system.
2. To implement the database using DDL statements.
3. To apply normalization step by step to ensure data integrity.

Prerequisites:

1. Understanding of database concepts.
2. Access to a relational database management system (RDBMS) like MySQL or PostgreSQL.

Equipment and Software:

1. RDBMS (e.g., MySQL, PostgreSQL).
2. Database client (e.g., MySQL Workbench, pgAdmin).

Database Design:

Entities:

1. **Customer:**
    - Attributes: CustomerID (Primary Key), FirstName, LastName, Email, Phone
2. **Order:**
    - Attributes: OrderID (Primary Key), OrderDate, TotalAmount
    - Foreign Key: CustomerID (references Customer)
3. **Product:**
    - Attributes: ProductID (Primary Key), ProductName, Price

DDL Statements:

1. Create Tables:

```
-- Customer Table
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100),
    Phone VARCHAR(20)
);

-- Order Table
CREATE TABLE Order (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    TotalAmount DECIMAL(10, 2),
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);

-- Product Table
CREATE TABLE Product (
```

```
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Price DECIMAL(8, 2)
);
```

2. Apply Normalization:

1NF (First Normal Form):

- All attributes must contain atomic values (no repeating groups or arrays).

2NF (Second Normal Form):

- Must be in 1NF.
- No partial dependencies on a concatenated key.

3NF (Third Normal Form):

- Must be in 2NF.
- No transitive dependencies.

The tables as created already satisfy 1NF.

Applying 2NF:

The **Order** table has a concatenated key (**OrderID**, **CustomerID**). To achieve 2NF, we need to create a separate table for order details.

```
-- OrderDetail Table
CREATE TABLE OrderDetail (
    OrderID INT,
```

```
    ProductID INT,
    Quantity INT,
    PRIMARY KEY (OrderID, ProductID),
    FOREIGN KEY (OrderID) REFERENCES Order(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);
```

```
-- Update Order Table
ALTER TABLE Order
DROP COLUMN TotalAmount;
-- Add TotalAmount to OrderDetail Table
ALTER TABLE OrderDetail
ADD COLUMN TotalAmount DECIMAL(10, 2);
```

Applying 3NF:

In the **OrderDetail** table, we have **Quantity** as a non-prime attribute dependent on **OrderID** and **ProductID**. We can separate it into a new table to achieve 3NF.

```
-- OrderQuantity Table
CREATE TABLE OrderQuantity (
    OrderID INT,
    ProductID INT,
    Quantity INT,
    PRIMARY KEY (OrderID, ProductID),
    FOREIGN KEY (OrderID) REFERENCES Order(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

-- Update OrderDetail Table
ALTER TABLE OrderDetail
DROP COLUMN Quantity;
```

Now, the tables are in 3NF.

Conclusion:

This set of DDL statements creates a simple database schema and applies normalization up to 3NF. Normalization helps in organizing data efficiently, reducing redundancy, and avoiding update anomalies. Feel free to adjust the schema based on specific requirements and constraints.