CS3120: DBMS Lab

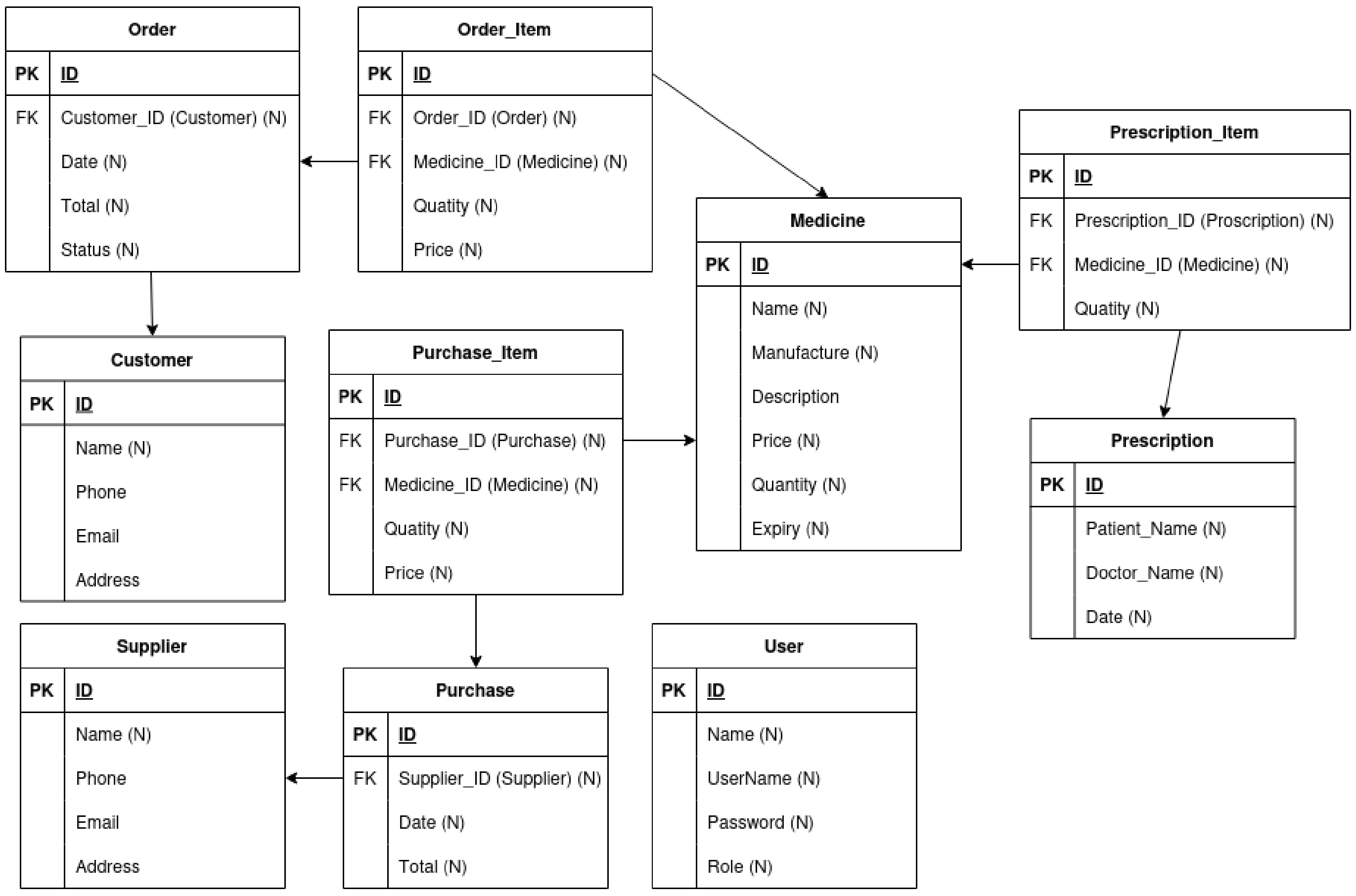
MidTerm Project Submission

PHARMACY MANAGEMENT SYSTEM

- Submission by:
  - 112001015 - Harsh Kanani
  - 112001016 - Kaustubh Sapkale
  - 112001017 - Kaustubh Chavan

The pharmacy management system database is designed to track and manage a pharmacy's inventory, sales, prescriptions, purchases and customers. The database contains multiple tables that are related to each other through foreign key constraints, allowing the system to maintain data consistency and integrity. The system also includes different roles and privileges for users to access and modify the data according to their job responsibilities.

Diagram



As a pharmacy, we buy medicines from different suppliers and sell it to different customers. To manage this records we have different tables which are interconnected with each other. On each purchase from suppliers/sell to customers it may contain multiple medicines per order. To get the information of all the medicines puchased/sold in an order we have a relation Order_Item/Purchase_Item. This contains foreign key medicine_id which can give us information about medicines from Medicine table. Other than this we have User table so which will consist of all the user using the database with their roles. We also store prescription information given by patients to see the trends of doctors vs medicines.

Queries

1. All customers who have made at least one purchase in the last year, along with their total spending for that period.

```sql
SELECT Customer.Name, SUM(Order.Total) AS Total_Spending
FROM Customer
JOIN Order ON Customer.ID = Order.Customer_ID
WHERE Order.Date BETWEEN DATEADD(year, -1, GETDATE()) AND GETDATE()
GROUP BY Customer.Name;
```

2. All prescriptions that have been issued for a particular patient, along with the medicine name and quantity prescribed.

```sql
SELECT Prescription.ID, Medicine.Name, Prescription_Item.Quantity
FROM Prescription_Item
JOIN Prescription ON Prescription_Item.Prescription_ID = Prescription.ID
JOIN Medicine ON Prescription_Item.Medicine_ID = Medicine.ID
JOIN Patient ON Prescription.Patient_ID = Patient.ID
WHERE Patient.Name = 'John Smith';
```

3. Names of all medicines that have been sold to a particular customer, along with the quantity and price for each sale.

```sql
SELECT Medicine.Name, Order_Item.Quantity, Order_Item.Price
FROM Medicine
JOIN Order_Item ON Medicine.ID = Order_Item.Medicine_ID
JOIN Order ON Order_Item.Order_ID = Order.ID
JOIN Customer ON Order.Customer_ID = Customer.ID
WHERE Customer.Name = 'John Doe';
```

4. Total revenue generated from all sales made in the last month, broken down by medicine.

```sql
SELECT Medicine.Name, SUM(Order_Item.Quantity) AS Total_Quantity,
SUM(Order_Item.Price * Order_Item.Quantity) AS Total_Revenue
FROM Medicine
JOIN Order_Item ON Medicine.ID = Order_Item.Medicine_ID
JOIN Order ON Order_Item.Order_ID = Order.ID
WHERE Order.Date BETWEEN DATEADD(month, -1, GETDATE()) AND GETDATE()
GROUP BY Medicine.Name;
```

5. All prescriptions that have been issued for a particular medicine, along with the patient name and the quantity of medicine prescribed.

```
SELECT Prescription.ID, Patient.Name, Prescription_Item.Quantity
FROM Prescription_Item
JOIN Prescription ON Prescription_Item.Prescription_ID = Prescription.ID
JOIN Medicine ON Prescription_Item.Medicine_ID = Medicine.ID
JOIN Patient ON Prescription.Patient_ID = Patient.ID
WHERE Medicine.Name = 'Aspirin';
```

Views

1. View that displays the current stock level of each medicine.

```
CREATE VIEW Medicine_Stock AS
SELECT Medicine.Name, Medicine.Quantity
FROM Medicine;
```

2. View that displays the total revenue generated from each customer, broken down by month.

```
CREATE VIEW Customer_Revenue AS
SELECT Customer.Name, MONTH(Order.Date) AS Month, SUM(Order.Total) AS
Total_Revenue
FROM Customer
JOIN Order ON Customer.ID = Order.Customer_ID
GROUP BY Customer.Name, MONTH(Order.Date);
```

Roles and Privileges

1. Created a role called "Salesperson" that allows users assigned to it to view and create sales, but not modify or delete them.

```
CREATE ROLE Salesperson;
GRANT SELECT, INSERT ON Order TO Salesperson;
```

2. Created a role called "Manager" that allows users assigned to it to view and modify all tables in the database except user.

```
CREATE ROLE Manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON
```

```
Medicine, Supplier, Customer, Order, Order_Item, Purchase,
Purchase_Item, Prescription, Prescription_Item TO Manager;
```

3. Created a user called "Jane" and assign her to the "Salesperson" role.

```
CREATE USER Jane WITH PASSWORD 'password';
ALTER ROLE Salesperson ADD MEMBER Jane;
```

4. One role will be admin that will have all permissions.

Function

Function that calculates the total revenue generated by a particular medicine.

```
CREATE FUNCTION Get_Medicine_Revenue (@Medicine_Name VARCHAR(255))
RETURNS DECIMAL(18,2)
AS
BEGIN
  DECLARE @Revenue DECIMAL(18,2);

  SELECT @Revenue = SUM(Order_Item.Price * Order_Item.Quantity)
  FROM Order_Item
  JOIN Medicine ON Order_Item.Medicine_ID = Medicine.ID
  WHERE Medicine.Name = @Medicine_Name;

  RETURN @Revenue;
END;
```

Procedure

Procedure that adds a new medicine to the database.

```
CREATE PROCEDURE Add_Medicine (@Name VARCHAR(255), @Manufacturer
VARCHAR(255), @Price DECIMAL(18,2), @Quantity INT, @Description TEXT,
@Expiry TEXT)
AS
BEGIN
  INSERT INTO Medicine (Name, Manufacturer, Price, Quantity, Description,
Expiry)
  VALUES (@Name, @Manufacturer, @Price, @Quantity, @Description, @Expiry);
END;
```