AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Project Name: Pharmacy Management System

Course: Advance Database Management System

Section: A

Group members Name , ID & Contribution :

Name	ID	Contribution	Percentage
1. ATHOY KANTI RAY	20-43259-1	Class diagram, Normalization, Table creation, index, sequence, data insertion, query writing, PL/SQL>Trigger, Package, Cursor	23%
2. RAKIB AHAMED LIMON	20-42900-1	Scenario description, ER diagram, schema diagram, Use case,Pl/SQL>,function	19%
3. HASIN KHAN PROTICK	19-40252-1	User Interface	12%
4. MD SHAHRUKH HOSSAIN KHAN	20-43349-1	Activity Diagram, PL/SQL>Record	16%
5. FAIUS LABIB HASSAN KHAN	21-44755-1	Introduction, Cover page, Table of contentsProject proposal, conclusion	15%
6. MD TOUHIDUL ISLAM	20-43894-2	PL/SQL>Producer, Rel.alzebra	15%

Table of Contents

Introduction:

In the healthcare industry, the effective management of pharmacies plays an important role in ensuring the seamless delivery of medical services. Traditional pharmacy systems often faces challenges related to accuracy, inventory management, and customer service. In response to these challenges, the Pharmacy Management System (PMS) performs as a solution, taking the help of automation to enhance efficiency and streamline operations.

This report shows a comprehensive analysis of the Pharmacy Management System, aiming to provide a detailed understanding of its functionalities, benefits, and impact on the healthcare ecosystem. As we will drive inside this system, we will see how efficient it can make out life in terms of healthcare. This introduction sets the stage for an in-depth exploration of the PMS, shedding light on its significance in shaping the future of pharmacy management.

Background:

The pharmaceutical industry plays a critical role in healthcare, ensuring the availability and distribution of medications to patients. However, many pharmacies face challenges in managing their operations efficiently. Manual processes for inventory management, prescription processing, and sales tracking can lead to errors, inefficiencies, and increased operational costs. To address these challenges, the implementation of a robust Pharmacy Management System (PMS) is essential.

Problem Domain and Root Cause:

Problem Domain:

- **Inefficient Manual Processes:**
  - Many pharmacies rely on manual methods for inventory management, prescription processing, and sales tracking, leading to errors and delays.
- **Lack of Real-Time Information:**
  - Without a centralized system, pharmacies struggle to access real-time data on inventory levels, prescription history, and sales trends.
- **Compliance Challenges:**
  - Meeting regulatory requirements and maintaining data privacy becomes a complex task without a dedicated system.

Root Cause:

The root cause of these challenges is the absence of a comprehensive Pharmacy Management System. Manual processes are prone to human error, and the lack of real-time information hampers decision-making and customer service. The absence of a centralized system also contributes to compliance issues, posing risks to both patients and the business.

Objective:

The primary objective of this project is to develop and implement a Pharmacy Management System that addresses the challenges faced by pharmacies in their day-to-day operations. The system aims to automate processes, provide real-time information, enhance compliance, and ultimately improve the overall efficiency of pharmacy management.

Solution:

Key Features of the Pharmacy Management System:

1. **Inventory Management:**
    - Automated tracking of pharmaceutical stock with real-time updates.
    - Efficient reordering processes to prevent stockouts.
2. **Prescription Processing:**
    - Streamlined prescription recording and processing.
    - Generation of accurate labels with dosage instructions.
3. **Sales and Billing:**
    - Point-of-sale transactions and automated invoicing.
    - Sales tracking for reporting and analysis.
4. **Patient Management:**
    - Comprehensive patient database for quick retrieval of records.
    - Personalized care through access to prescription history.
5. **Supplier Management:**
    - Supplier information management and streamlined ordering processes.
    - Tracking of purchase history and supplier payments.
6. **Reporting and Analytics:**
    - Generation of reports on sales, inventory, and other key metrics.
    - Data-driven insights for informed decision-making.
7. **Security and Access Control:**
    - Implementation of robust security measures to ensure data confidentiality.

    - User roles and access controls to prevent unauthorized access.

Technology Stack:

The system will be developed using [technology stack], ensuring scalability, security, and usability.

Target User and Benefit:

Target User:

The Pharmacy Management System is designed for:

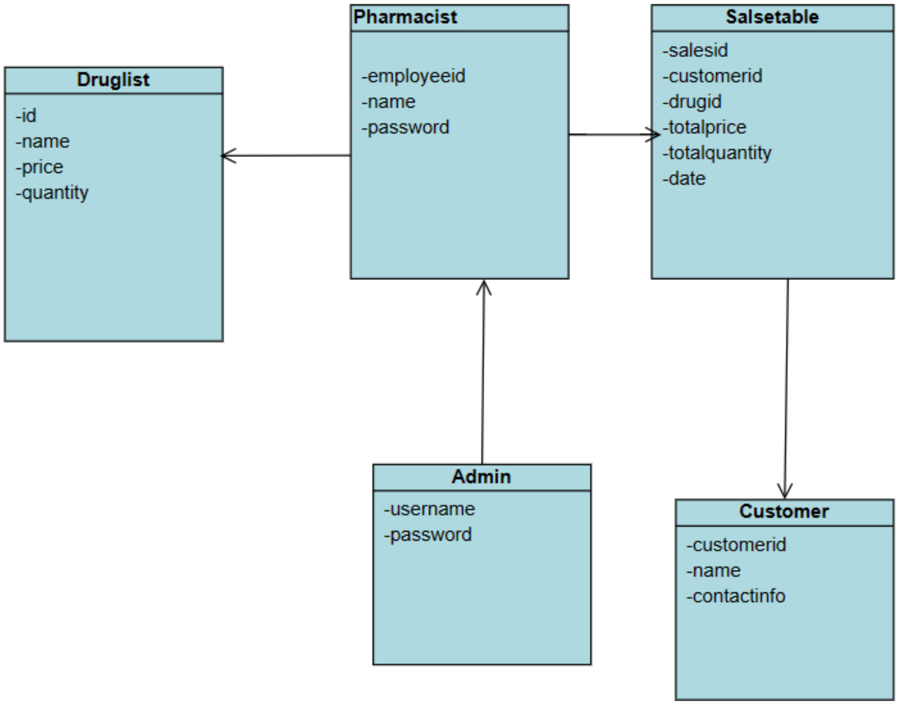
- **Pharmacy Owners and Managers:**
    - Streamline day-to-day operations.
    - Enhance decision-making through real-time insights.
- **Pharmacy Staff:**
    - Simplify tasks such as prescription processing and inventory management.
    - Improve customer service through quick access to patient records.

Benefits:

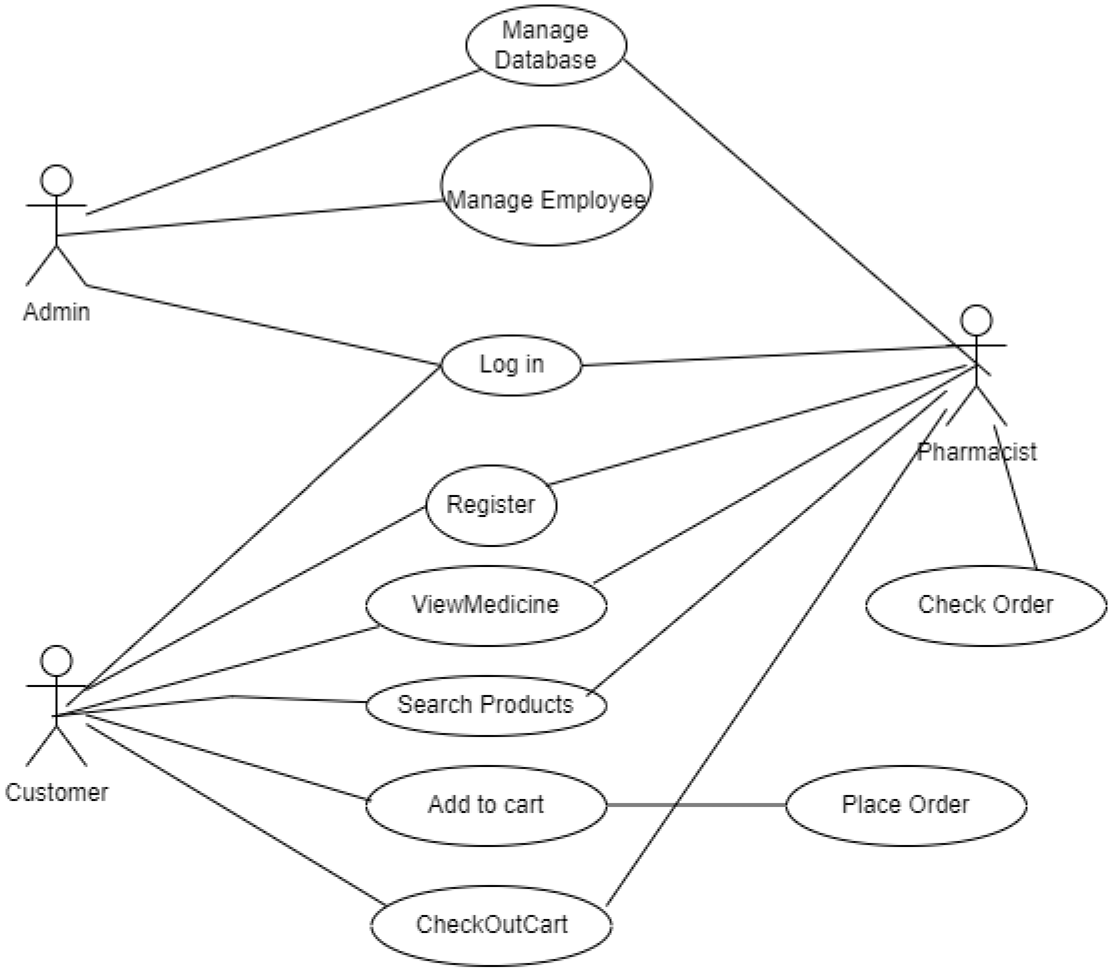
1. **Efficiency Improvement:**
    - Automation of manual processes reduces the time spent on administrative tasks.
    - Quick access to information improves overall workflow efficiency.
2. **Error Reduction:**
    - Minimizes the risk of errors in prescription processing and inventory management.
    - Improves accuracy in dispensing medications.
3. **Enhanced Customer Service:**
    - Provides quick access to patient records, improving customer service.
    - Facilitates personalized care by maintaining a comprehensive patient database.
4. **Regulatory Compliance:**
    - Helps pharmacies adhere to regulatory standards and reporting requirements.
    - Facilitates compliance with healthcare data privacy laws.
5. **Cost Control:**

- Prevents overstocking or understocking through effective inventory management.
- Optimizes purchasing processes to control costs.

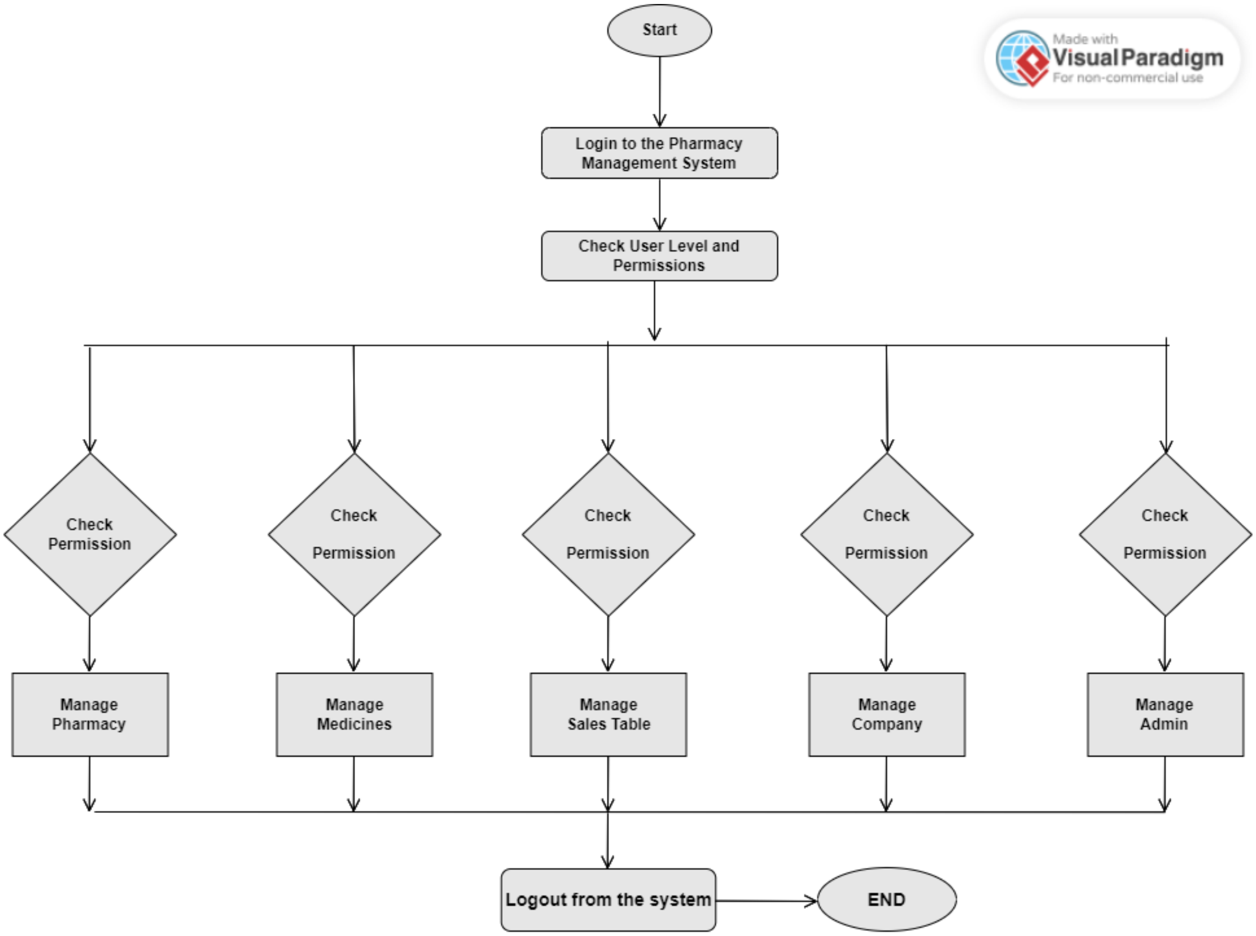
Class Diagram:



Use Case:

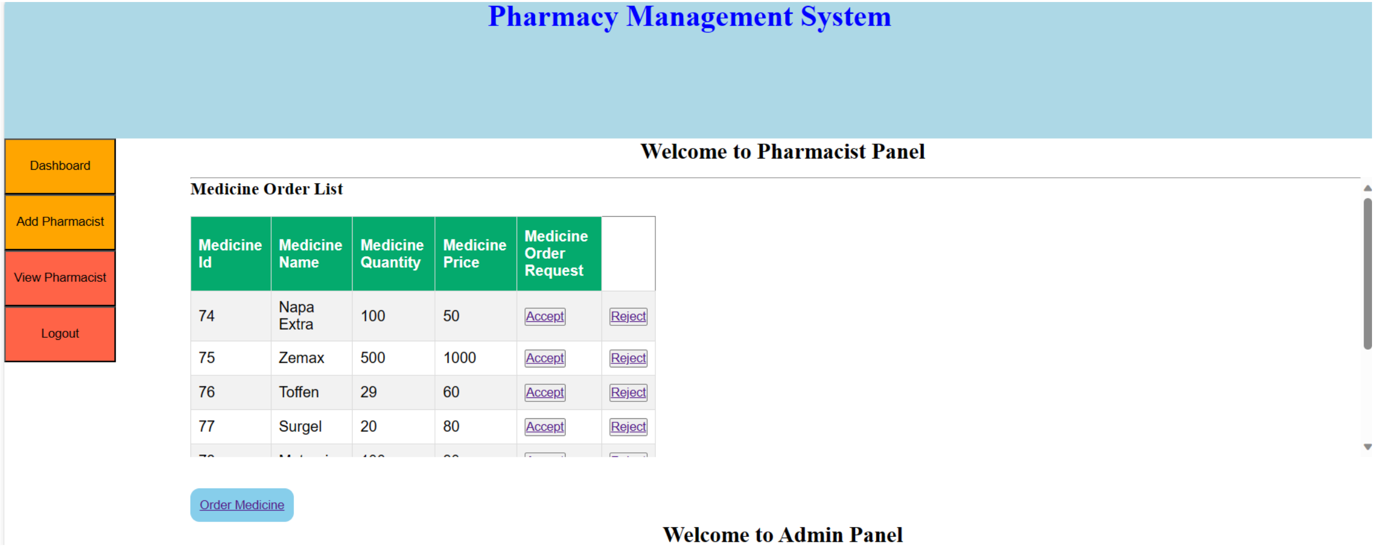

Manage Database
Manage Employee
Admin
Log in
Pharmacist
Register
ViewMedicine
Check Order
Search Products
Customer
Add to cart
Place Order
CheckOutCart

Activity Diagram:



Activity Diagram for Pharmacy Management System

User Interface:



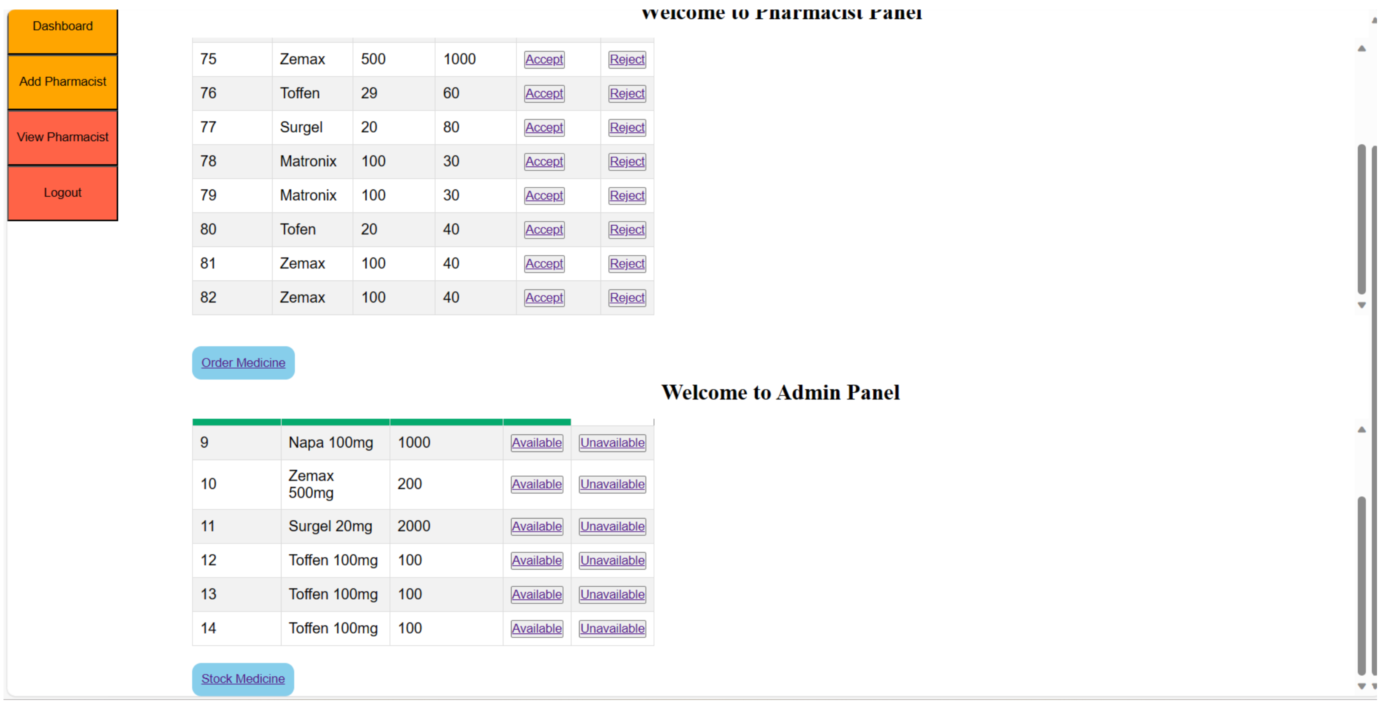
Pharmacy Management System

Welcome to Pharmacist Panel

Medicine Order List

Medicine Id	Medicine Name	Medicine Quantity	Medicine Price	Medicine Order Request	
74	Napa Extra	100	50	Accept	Reject
75	Zemax	500	1000	Accept	Reject
76	Toffen	29	60	Accept	Reject
77	Surgel	20	80	Accept	Reject

Welcome to Admin Panel

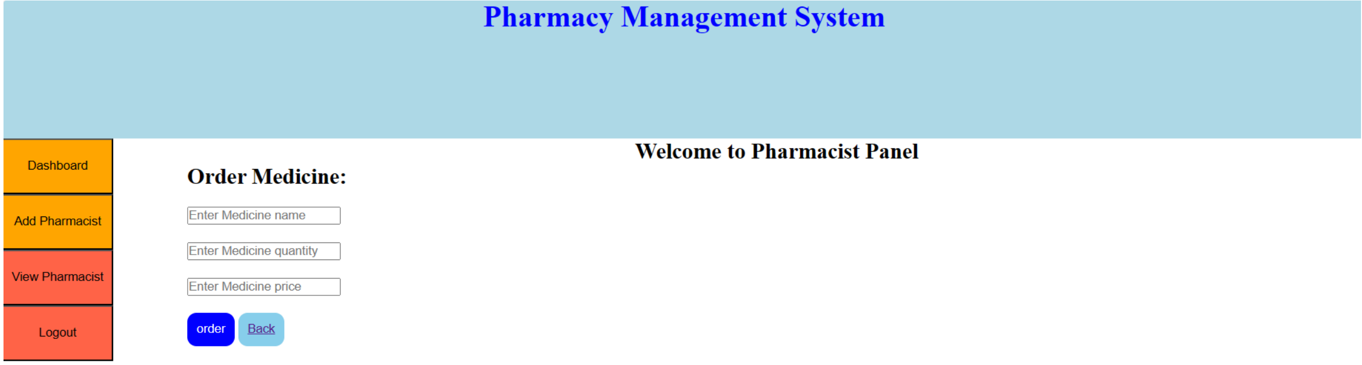


Welcome to Pharmacist Panel

75	Zemax	500	1000	Accept	Reject
76	Toffen	29	60	Accept	Reject
77	Surgel	20	80	Accept	Reject
78	Matronix	100	30	Accept	Reject
79	Matronix	100	30	Accept	Reject
80	Tofen	20	40	Accept	Reject
81	Zemax	100	40	Accept	Reject
82	Zemax	100	40	Accept	Reject

Welcome to Admin Panel

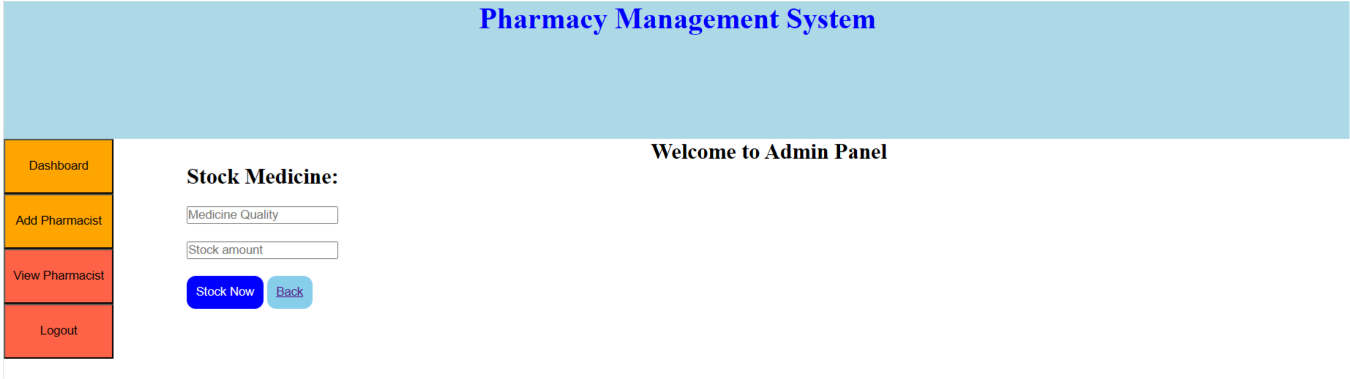
9	Napa 100mg	1000	Available	Unavailable
10	Zemax 500mg	200	Available	Unavailable
11	Surgel 20mg	2000	Available	Unavailable
12	Toffen 100mg	100	Available	Unavailable
13	Toffen 100mg	100	Available	Unavailable
14	Toffen 100mg	100	Available	Unavailable



Pharmacy Management System

Welcome to Pharmacist Panel

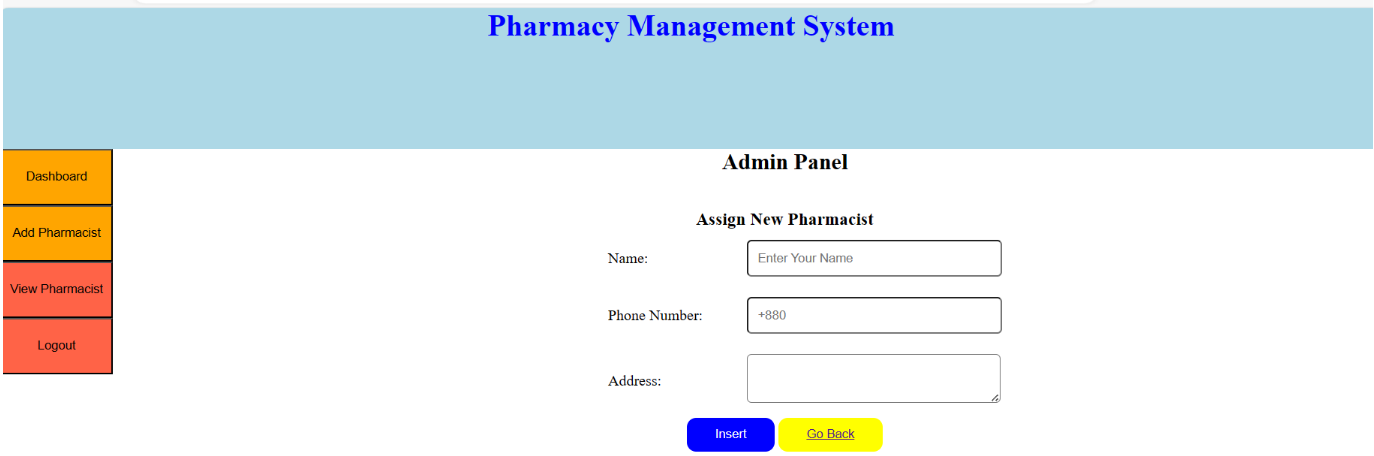
Order Medicine:



Pharmacy Management System

Welcome to Admin Panel

Stock Medicine:



Pharmacy Management System

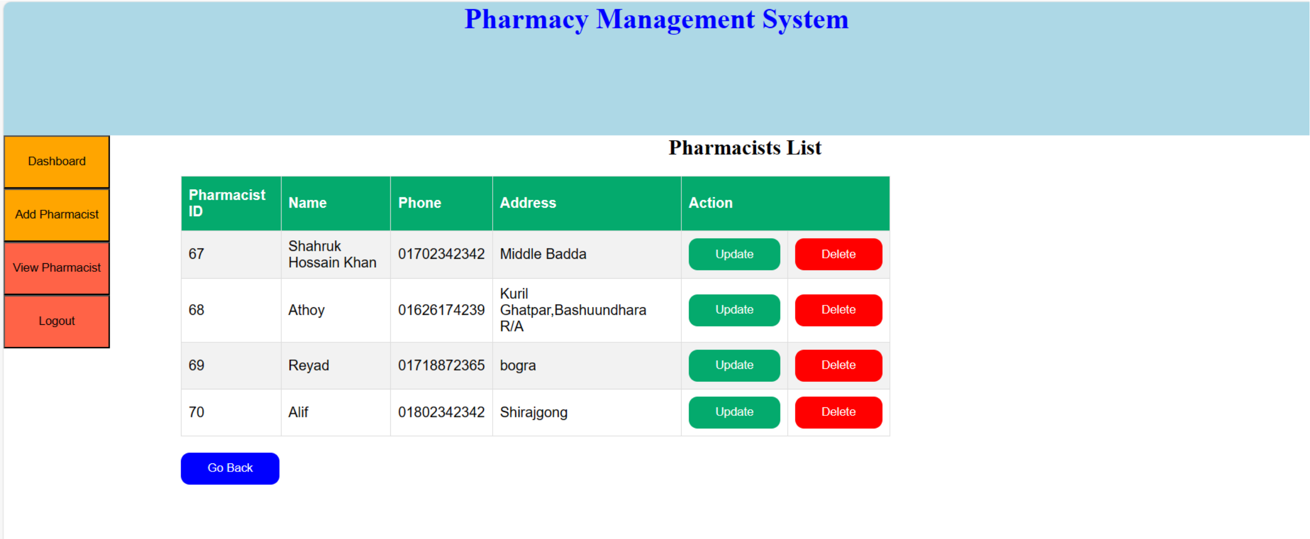
Admin Panel

Assign New Pharmacist

Name: Enter Your Name

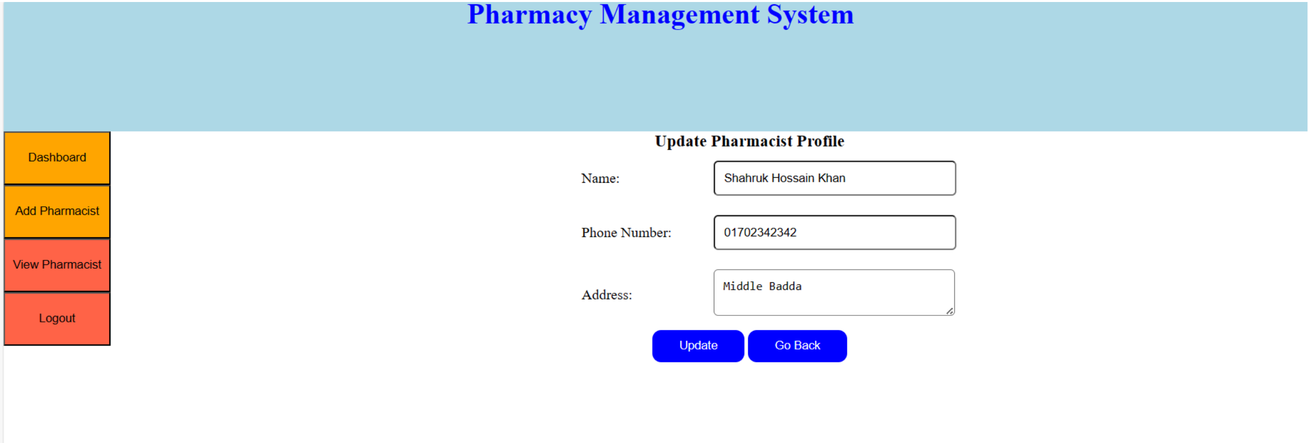
Phone Number: +880

Address:



Pharmacists List

Pharmacist ID	Name	Phone	Address	Action	
67	Shahruk Hossain Khan	01702342342	Middle Badda	Update	Delete
68	Athoy	01626174239	Kuril Ghatpar,Bashuundhara R/A	Update	Delete
69	Reyad	01718872365	bogra	Update	Delete
70	Alif	01802342342	Shirajgong	Update	Delete

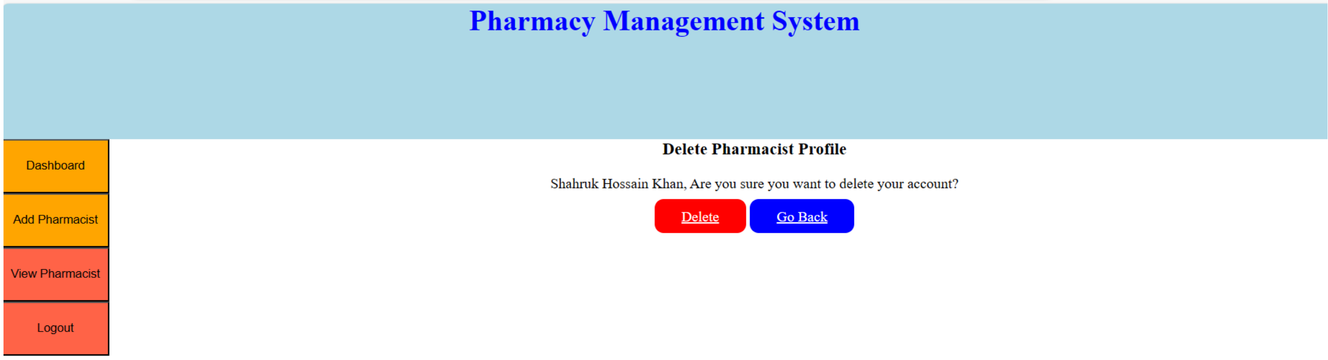


Update Pharmacist Profile

Name: Shahruk Hossain Khan

Phone Number: 01702342342

Address: Middle Badda



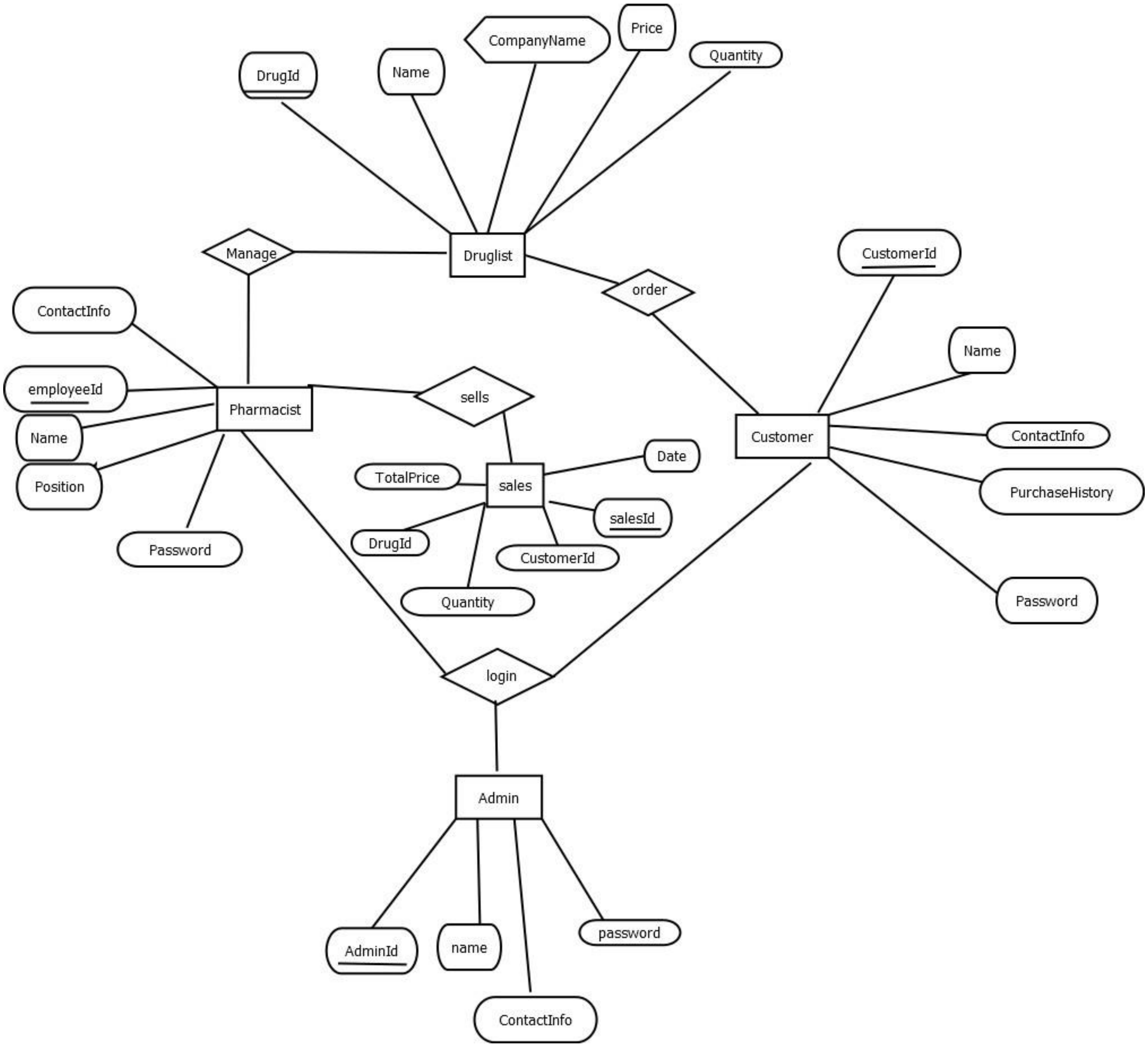
Delete Pharmacist Profile

Shahruk Hossain Khan, Are you sure you want to delete your account?

Scenario Description:

In a pharmacy management system, an admins are responsible for system configuration, employee management, and overseeing the addition of new medicines. Admins have unique login credentials and personal information. Admins manage employees, a one-to-many relationship. Here an admin can manage multiple employees. Each employee has a unique employee ID, personal details. Employees are managed by an admin, creating a many-to-one relationship. Employees can serve multiple customers, forming a one-to-many relationship. Customers creaing accounts by giving some personal information such as Customer ID (Primary Key), first Name, last name, gender, Date Of Birth, Address, phone, email. Customers can make multiple purchases, forming a one-to-many relationship with the Sales table. When an employee sells medicine to a customer, a data is added to the sales table. Each sale is associated with a specific employee which is many-to-one relationship with the Employee table. Each sale involves a specific medicine which is many- to-one relationship with the medicine list table. Each sale is linked to a customer which is many-to-one relationship with the Customer table. Medicine list table show the medicines available in the pharmacy. Each medicine has a unique ID, name, manufacturer details, expiry date, unit price, and quantity in stock.

ER Diagram:



Normalization:

Druglist (drugid, name, company_name, price, quantity)

1NF-> Company name. is a multivalued attribute.

2NF -> drugid, name, company_name, price

3NF-> drugid, name, company_name, price, quantity

Tables from ***Druglist***:

1) drugid, name.
2) drugid, name, company_name, price
3) drugid, name, company_name, price, quantit

Pharmacist (employeegid, name, position, contactinfo, password)

1NF-> Contact info. is a multivalued attribute.

2NF -> employeegid, name, position

3NF-> employeegid, name, position, contactinfo, password

Tables from ***Pharmacist***:

1) employeegid, name.
2) employeegid, name, position, contact info, password

Customer (customerid, name, contactinfo, password)

1NF-> Contact info. is a multivalued attribute.

2NF -> customerid, name, password

3NF-> customerid, name, contactinfo, password

Tables from ***Customer***:

4. customerid, name.
5. customerid, name, contact info, password

,

Admin (adminid, name, contactinfo, password)

1NF-> Contact info. is a multivalued attribute.

2NF -> adminid, name, position

3NF-> adminid, name, contactinfo, password

Tables from ***Admin***:

3) adminid, name.
4) adminid, name, contact info, password

SalesTable(sales id, customer id, medicineid, quantity sold, total price, sale date)

1NF-> Sale. is a multivalued attribute.

2NF-> sales id, quantity sold customer id

sale date

3NF-> sales id, total

price , medicine id

quantity sold

Tables from **SalesTable:**

1. sales id, quantity sold, customer is
2. customer is, total price ,price date sales id
3. medicine id , sale date, total price

Schema Diagram:

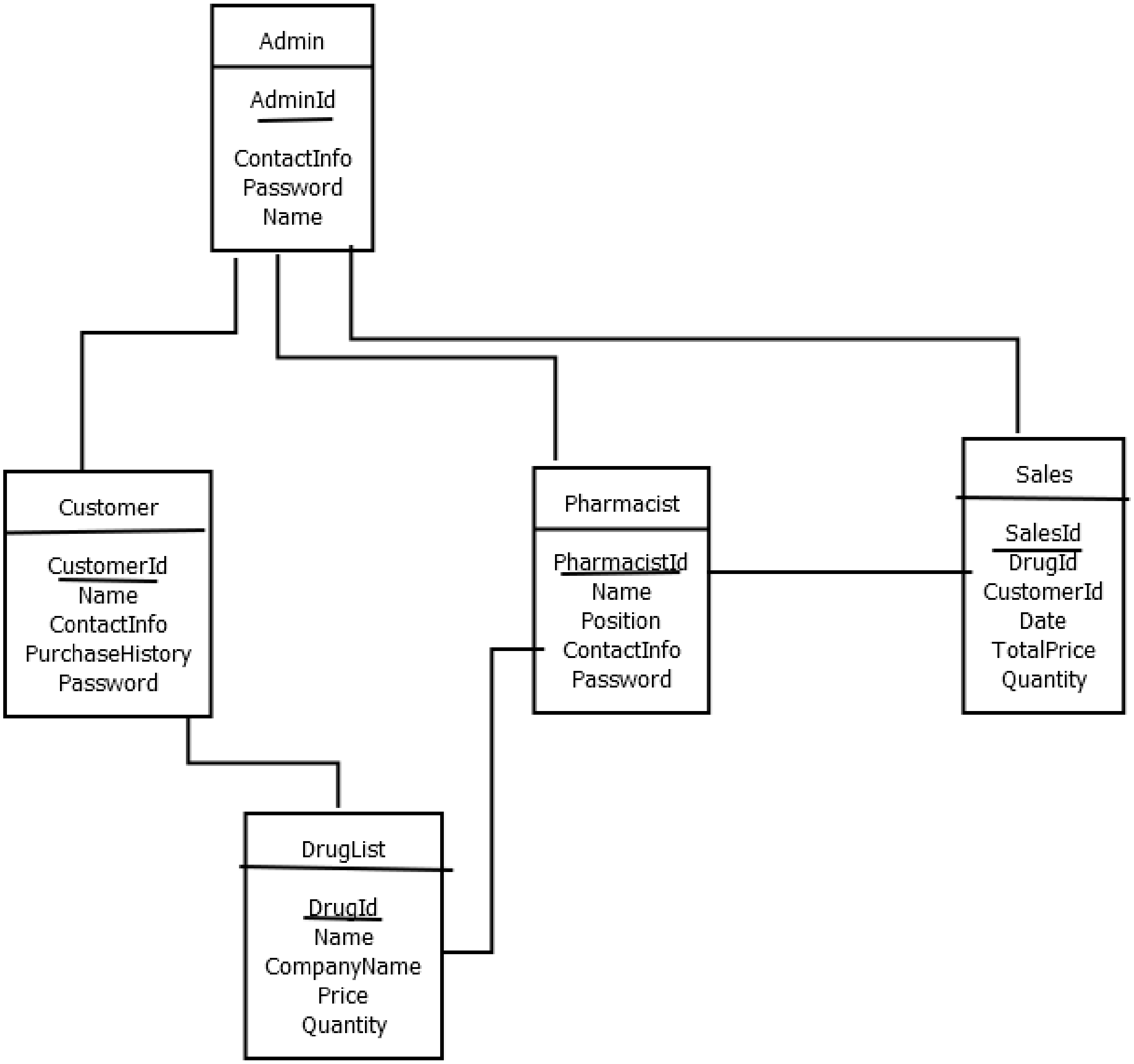
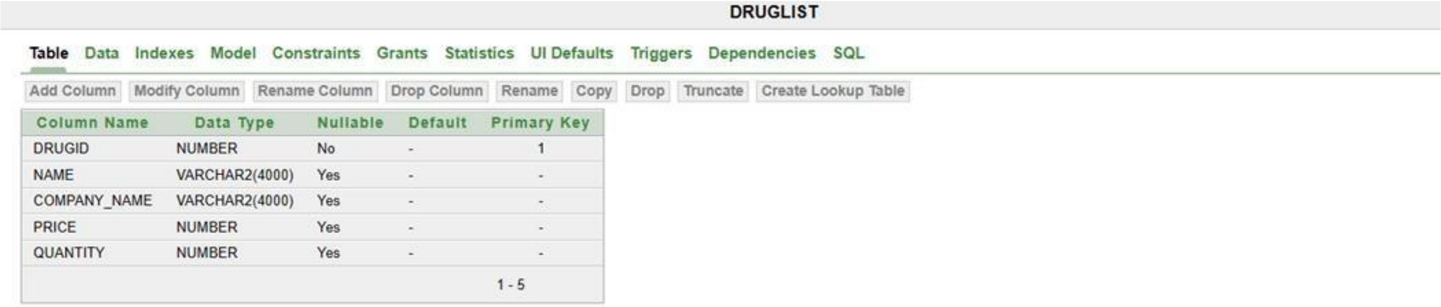


Table Creation:

Druglist :

```
create table Druglist(

Drugid Number(10) NOT NULL PRIMARY KEY,Nname VARCHAR2(4000),

Company_Name VARCHAR2(4000),

Price Number(10),

Quantity Number(10)

 desc Druglists;
```



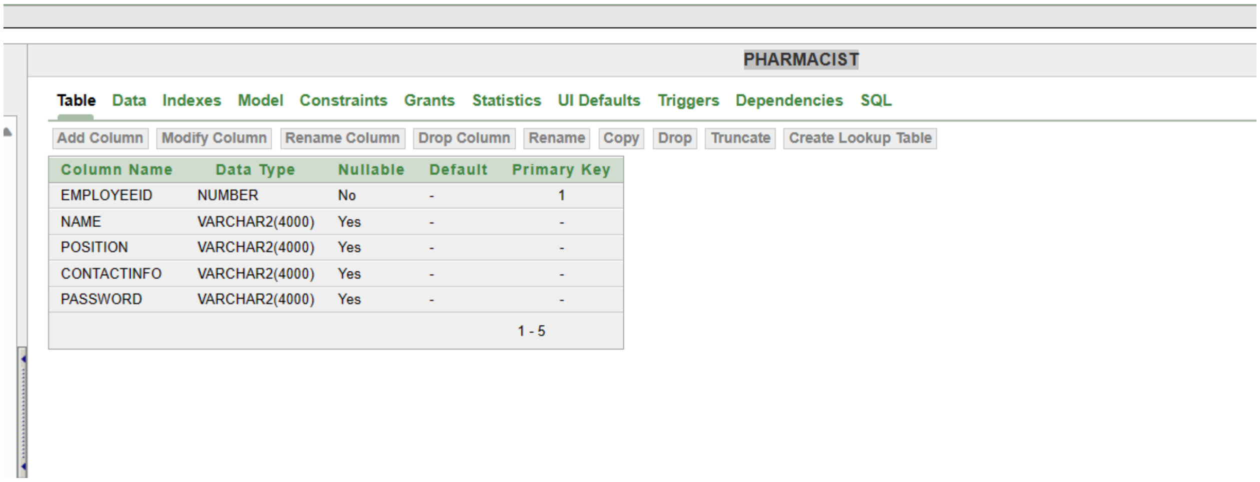
DRUGLIST

Column Name	Data Type	Nullable	Default	Primary Key
DRUGID	NUMBER	No	-	1
NAME	VARCHAR2(4000)	Yes	-	-
COMPANY_NAME	VARCHAR2(4000)	Yes	-	-
PRICE	NUMBER	Yes	-	-
QUANTITY	NUMBER	Yes	-	-

1 - 5

PHARMACIST:

```
Create table Pharmacist(


Employeeid Number NOT NULL PRIMARY KEY,

NameVarchar2(4000),

PositionVarcher2(40),


ContactinfoVarchar2(4000),


PasswordVarchar2(4000)  desc Pharmacist;
```

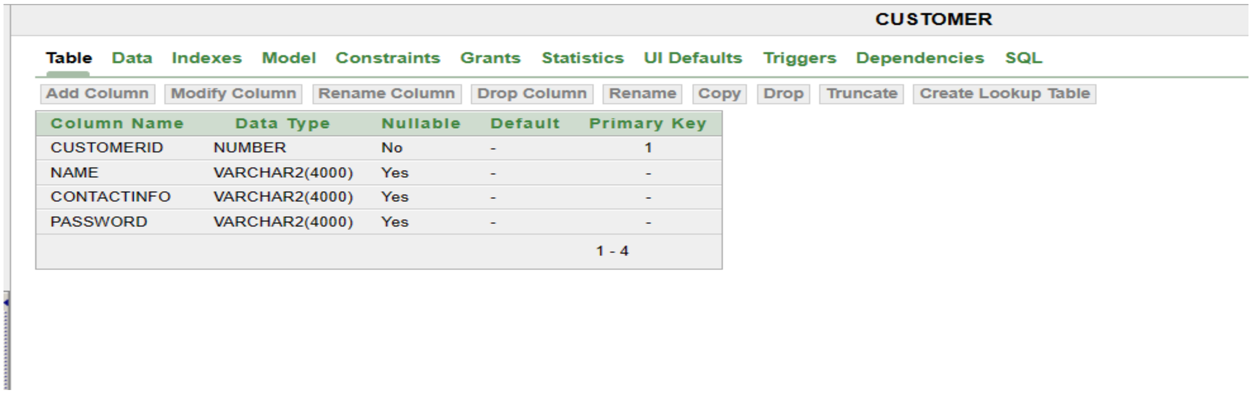


Column Name	Data Type	Nullable	Default	Primary Key
EMPLOYEEID	NUMBER	No	-	1
NAME	VARCHAR2(4000)	Yes	-	-
POSITION	VARCHAR2(4000)	Yes	-	-
CONTACTINFO	VARCHAR2(4000)	Yes	-	-
PASSWORD	VARCHAR2(4000)	Yes	-	-

1 - 5

Customer:

```
Create Table Customer(

 Customerid Number NOT NULL PRIMARY KEY,

Name Varchar(4000),

Contactinfo Varchar(4000),

Password Varchar(4000)

desc Customer
```

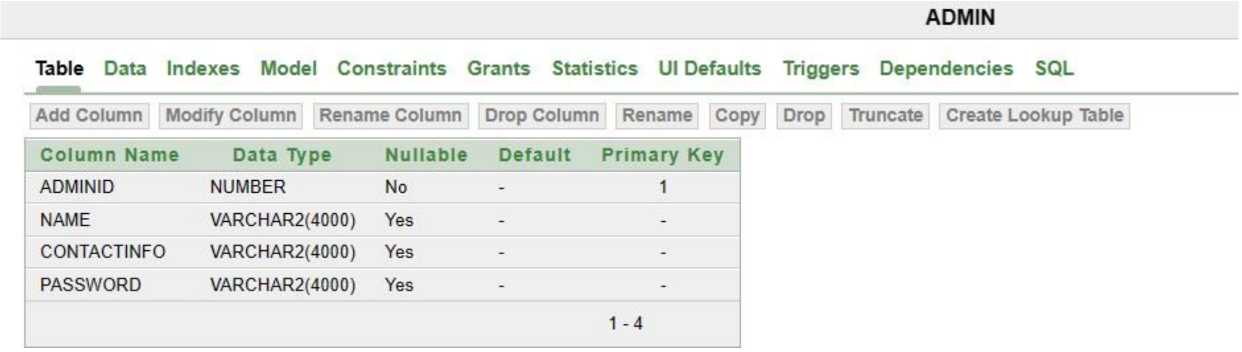


Column Name	Data Type	Nullable	Default	Primary Key
CUSTOMERID	NUMBER	No	-	1
NAME	VARCHAR2(4000)	Yes	-	-
CONTACTINFO	VARCHAR2(4000)	Yes	-	-
PASSWORD	VARCHAR2(4000)	Yes	-	-

1 - 4

Admin:

```
Create Table Admin( adminid Number Not NULL PRIMARY KEY,

Name Varchar(4000),


Contactinfo Varchar(4000),


Password Varchar(4000)


desc Admin
```



Column Name	Data Type	Nullable	Default	Primary Key
ADMINID	NUMBER	No	-	1
NAME	VARCHAR2(4000)	Yes	-	-
CONTACTINFO	VARCHAR2(4000)	Yes	-	-
PASSWORD	VARCHAR2(4000)	Yes	-	-

1 - 4

Sales Table:

```
            Create Table Sales(Salesid Number

            NOT NULL PRIMARY KEY,

            Foreign key(Customerid)referenced
            Customer(Customerid),
            Foreign key(Medicineid) reference Druglist(Drugid);

            Quantity_sold Number,
            Total_Price Number,

            Sales_Date Varchar;
```

Sequences:

Druglist

```
create sequence Druglist_Drugid increment by 1 start with 1 nocache nocycle;
```

Pharmacist

```
create sequence Pharmacist_Employeeid  increment by 1 start with 1 nocache nocycle;
```

Customer

```
create sequence Customer_Customerid increment by 1 start with 1 nocache nocycle;
```

Admin

```
create sequence  Admin_adminid increment by 1 start with 1 nocache nocycle;
```

SalesTable

```
create sequence SalesTable_salesid  increment by 1 start with 1 nocache nocycle;
```

Index:

Druglist:

```
CREATE INDEX Druglist

ON Druglist (Drugid,Name,Company_name,Price,Quantity);
```

Pharmacist:

```
CREATE INDEX Pharmacist ON Pharmacist(Employeeid,Name,
Position,contactinfo,Password);
```

Customer:

```
CREATE INDEX Customer

ON Customer(Customerid,Name,Contactinfo,Password);
```

Admin:

```
CREATE INDEX Admin

ON Admin(Adminid,Name,Contactinfo,Password);
```

SalesTable:

```
CREATE INDEX SalesTable

ON
SalesTable(Salesid,Customerid,Medicineid,Quantitysold,Total_price,Sales_Date);
```

Data Insertion:

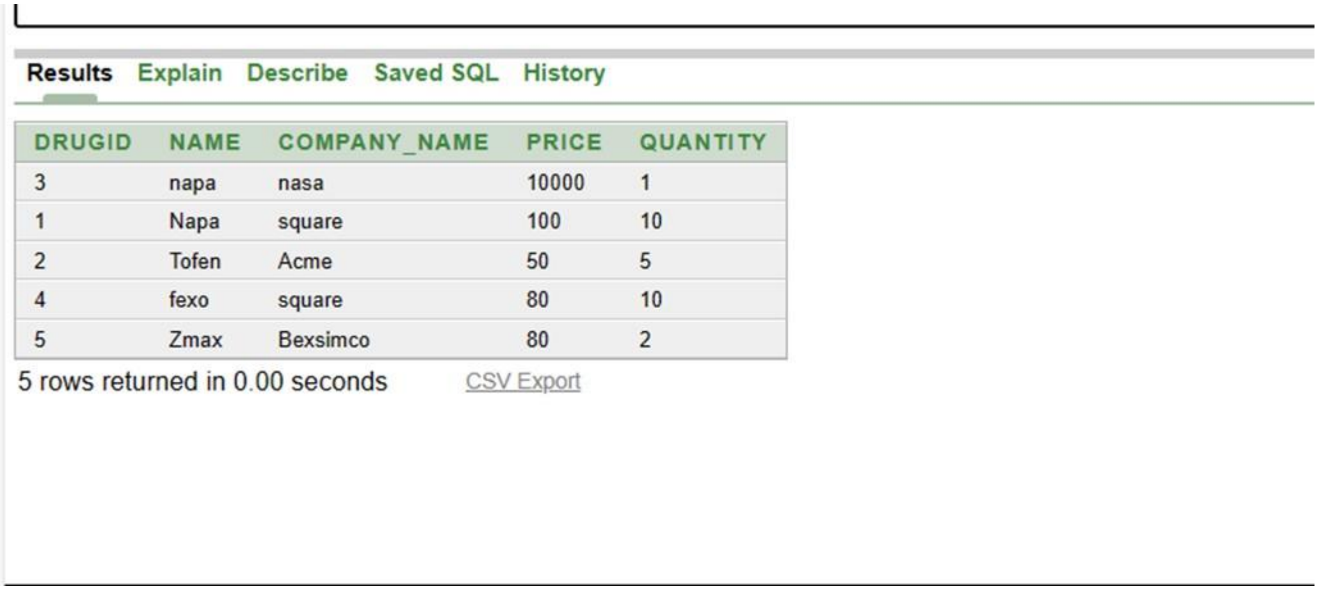
#Druglist

Insert INTO Druglist(Drugid,Name,Company_name,Price,quantity) values('1', 'Napa', 'square', '100', '10');

Insert INTO Druglist(Drugid,Name,Company_name,Price,quantity) values('2', 'Tofen', 'Acme', '50', '5');

Insert INTO Druglist(Drugid,Name,Company_name,Price,quantity) values('4', 'fexo', 'square', '80', '10');

Insert INTO Druglist(Drugid,Name,Company_name,Price,quantity) values('5', 'Zmax', 'Bexsimco', '80', '2');



DRUGID	NAME	COMPANY_NAME	PRICE	QUANTITY
3	napa	nasa	10000	1
1	Napa	square	100	10
2	Tofen	Acme	50	5
4	fexo	square	80	10
5	Zmax	Bexsimco	80	2

5 rows returned in 0.00 seconds

#Pharmacist

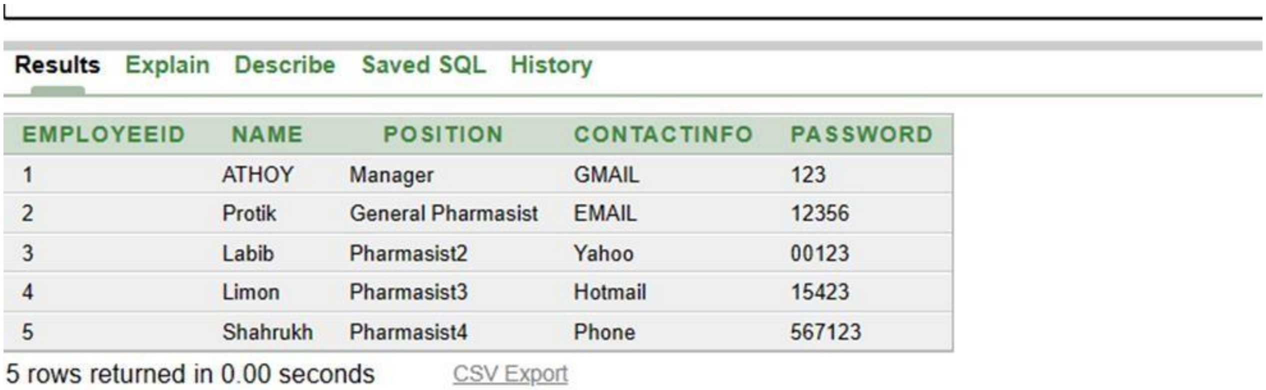
Insert INTO Pharmacist(Employeeid,Name,position,Contactinfo,Password) values('1', 'ATHOY', 'Manager', 'GMAIL', '123');

Insert INTO Pharmacist(Employeeid,Name,position,Contactinfo,Password) values('2', 'Protik', 'General Pharmasist', 'EMAIL', '12356');

Insert INTO Pharmacist(Employeeid,Name,position,Contactinfo,Password) values('3', 'Labib', 'Pharmasist2', 'Yahoo', '00123');

Insert INTO Pharmacist(Employeeid,Name,position,Contactinfo,Password) values('4', 'Limon', 'Pharmasist3', 'Hotmail', '15423');

Insert INTO Pharmacist(Employeeid,Name,position,Contactinfo,Password) values('5', 'Shahrukh', 'Pharmasist4', 'Phone', '567123');



EMPLOYEEID	NAME	POSITION	CONTACTINFO	PASSWORD
1	ATHOY	Manager	GMAIL	123
2	Protik	General Pharmasist	EMAIL	12356
3	Labib	Pharmasist2	Yahoo	00123
4	Limon	Pharmasist3	Hotmail	15423
5	Shahrukh	Pharmasist4	Phone	567123

5 rows returned in 0.00 seconds

# Customer

Insert INTO CUSTOMER(Customerid,Name,Contactinfo,Password) values('1', 'ATHOY', 'GMAIL', '123');

```
Insert INTO CUSTOMER(Customerid,Name,Contactinfo,Password) values('2', 'athoy', 'grg', 'hggihg');
```

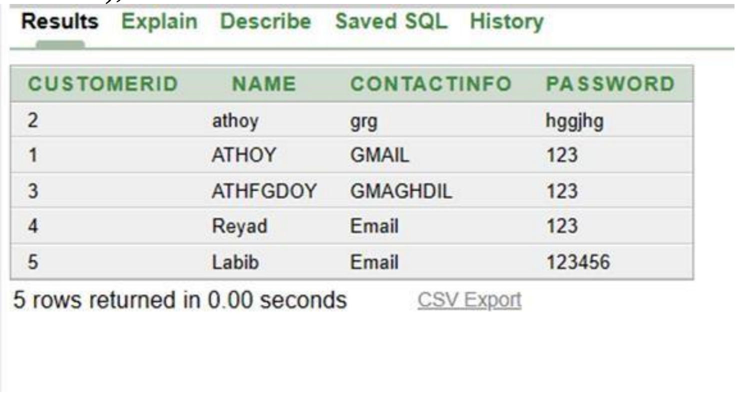
```
Insert INTO CUSTOMER(Customerid,Name,Contactinfo,Password) values('3', 'ATHFGDOY', 'GMAGHDIL', '123');
```

```
Insert INTO CUSTOMER(Customerid,Name,Contactinfo,Password) values('4', 'Reyad', 'Email', '123');
```

```
Insert INTO CUSTOMER(Customerid,Name,Contactinfo,Password) values('5', 'Labib', 'Email', '123456')
```

# Admin

```
 Insert INTO Admin(Adminid,Name,Contactinfo,Password) values('1', 'Shahruk', 'MsEmail', '13456');
 Insert INTO Admin(Adminid,Name,Contactinfo,Password) values('2', 'Labib', 'Email', '1345656');
 Insert INTO Admin(Adminid,Name,Contactinfo,Password) values('3', 'Limon', 'gmail', '3456');
Insert INTO Admin(Adminid,Name,Contactinfo,Password) values('4', 'Protik', 'Hotmail', '567890');
```



CUSTOMERID	NAME	CONTACTINFO	PASSWORD
2	athoy	grg	hggjhg
1	ATHOY	GMAIL	123
3	ATHFGDOY	GMAGHDIL	123
4	Reyad	Email	123
5	Labib	Email	123456

5 rows returned in 0.00 seconds

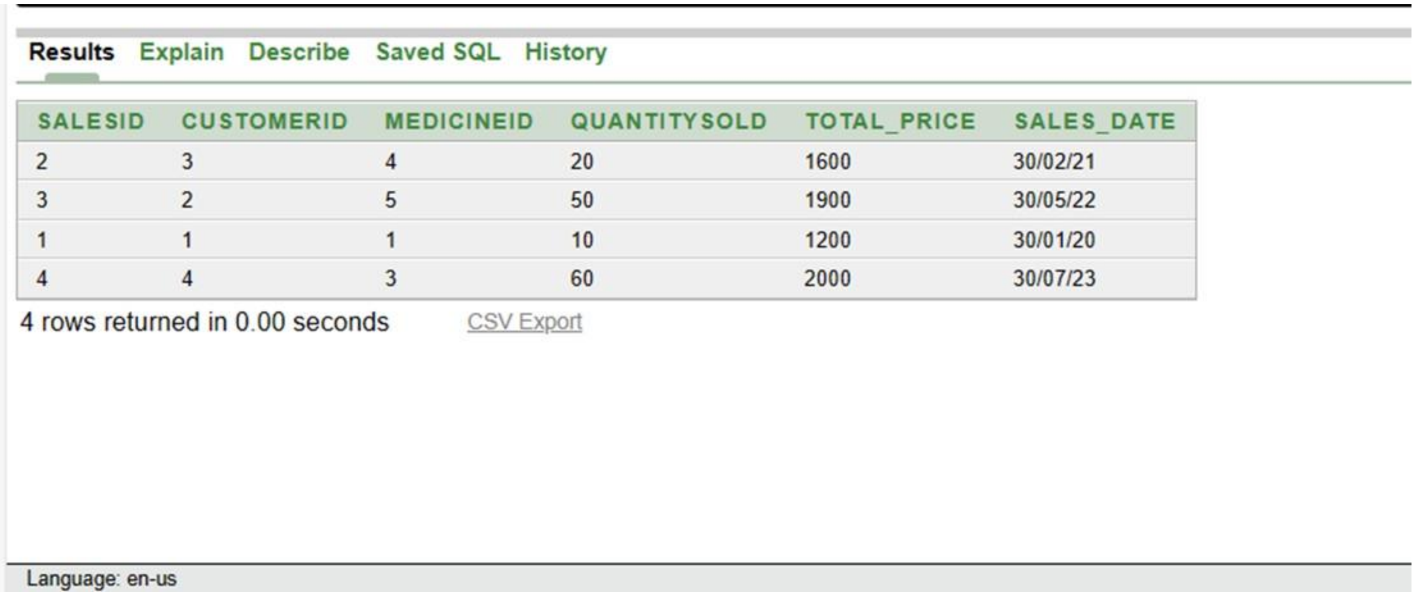
#SalesTable

Insert INTO Salestable(Salesid,Customerid,Medicineid,Quantitysold,Total_price,Sales_Date) values('1', '1', '1', '10','1200','30/01/20');

Insert INTO Salestable(Salesid,Customerid,Medicineid,Quantitysold,Total_price,Sales_Date) values('2', '3', '4', '20','1600','30/02/21');

Insert INTO Salestable(Salesid,Customerid,Medicineid,Quantitysold,Total_price,Sales_Date) values('3', '2', '5', '50','1900','30/05/22');

Insert INTO Salestable(Salesid,Customerid,Medicineid,Quantitysold,Total_price,Sales_Date) values('4', '4', '3', '60','2000','30/07/23');



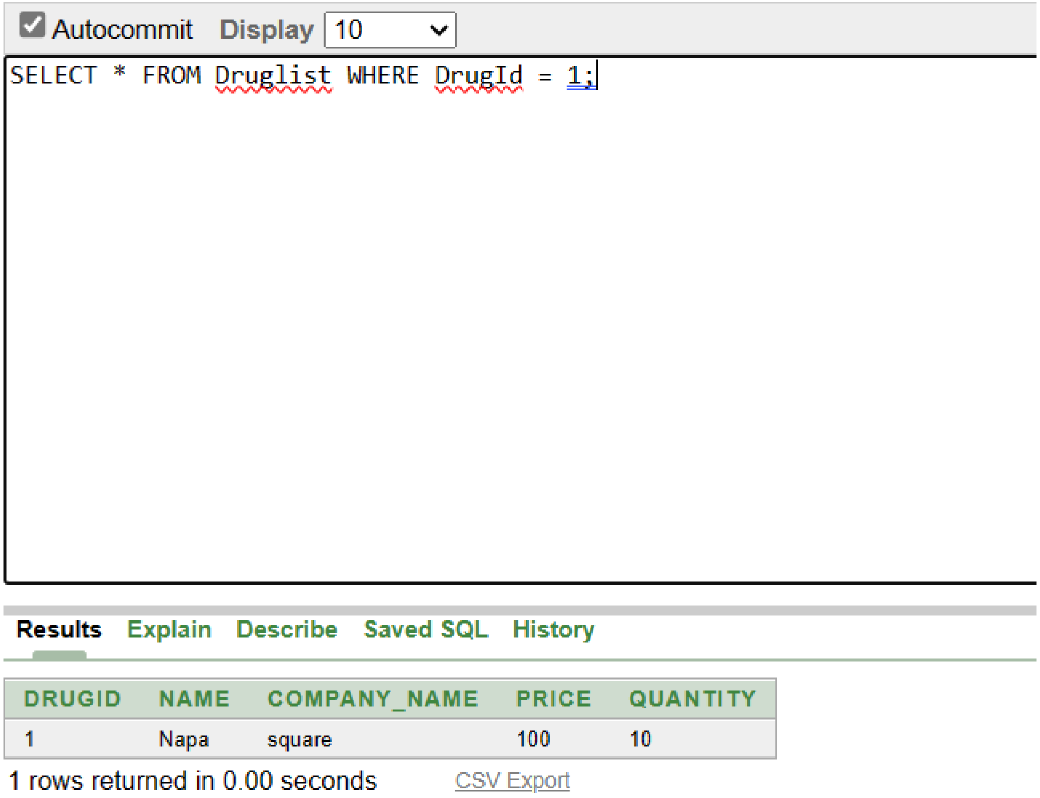
SALESID	CUSTOMERID	MEDICINEID	QUANTITYSOLD	TOTAL_PRICE	SALES_DATE
2	3	4	20	1600	30/02/21
3	2	5	50	1900	30/05/22
1	1	1	10	1200	30/01/20
4	4	3	60	2000	30/07/23

4 rows returned in 0.00 seconds

Query Writing:

Single Row:

1. Retrieve the details of the medicine with DrugId 1 from the Druglist table

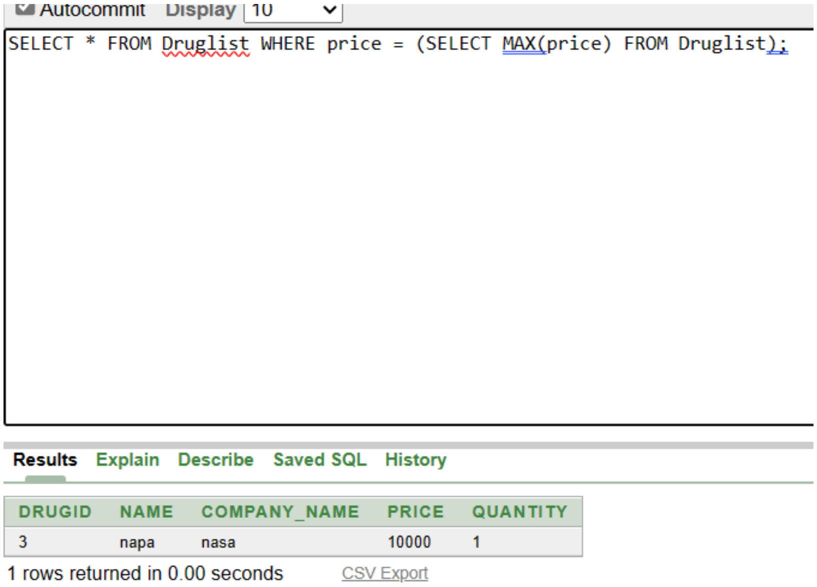


```
SELECT * FROM Druglist WHERE DrugId = 1;
```

DRUGID	NAME	COMPANY_NAME	PRICE	QUANTITY
1	Napa	square	100	10

1 rows returned in 0.00 seconds

2. Retrieve the details of the medicine with the highest price from the Druglist table

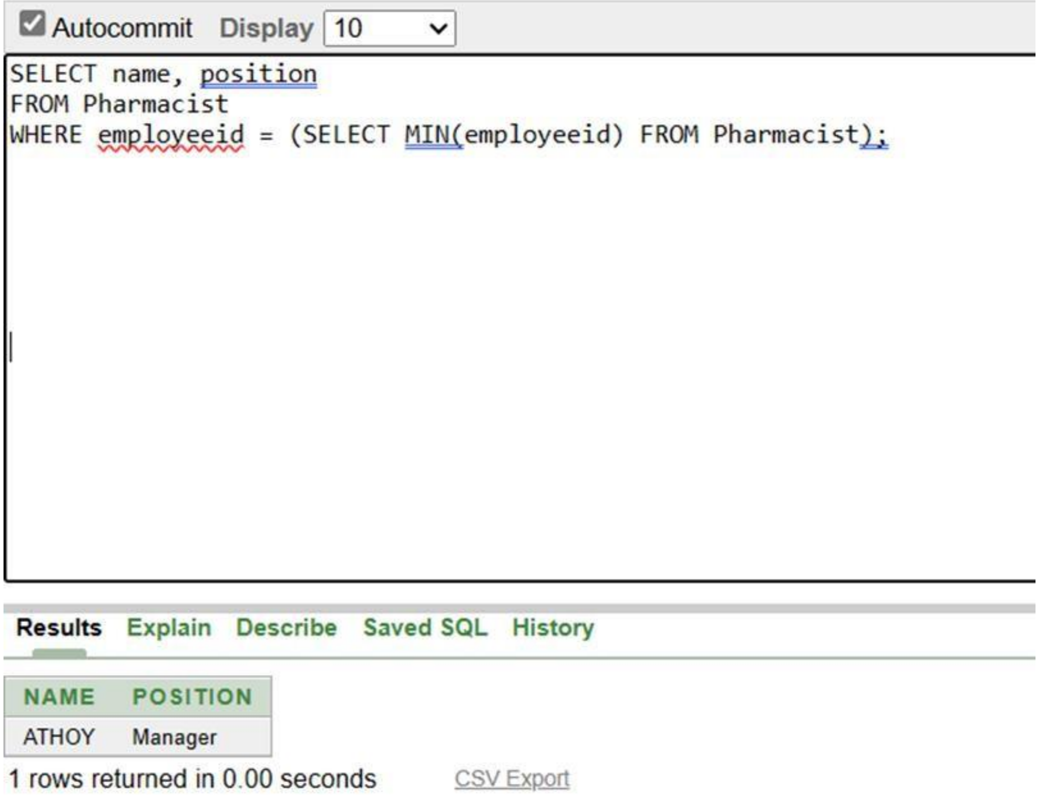


```
SELECT * FROM Druglist WHERE price = (SELECT MAX(price) FROM Druglist);
```

DRUGID	NAME	COMPANY_NAME	PRICE	QUANTITY
3	napa	nasa	10000	1

1 rows returned in 0.00 seconds

3. Retrieve the name and position of the pharmacist with the lowest employee ID



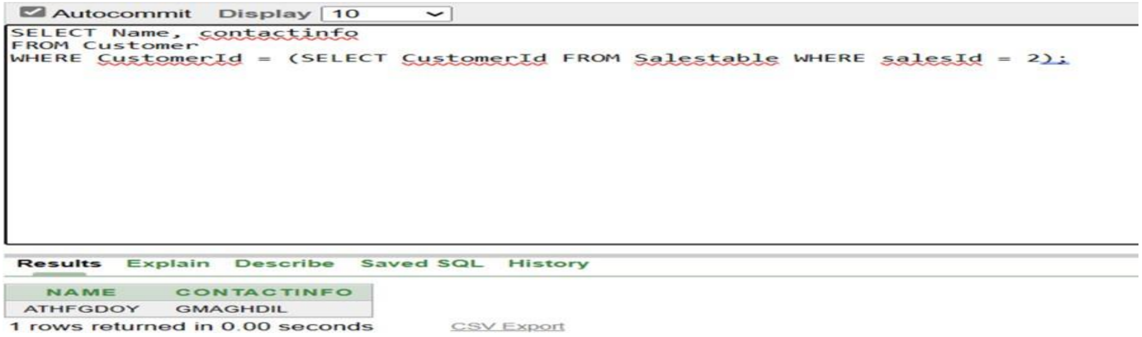
```
SELECT name, position
FROM Pharmacist
WHERE employeeid = (SELECT MIN(employeeid) FROM Pharmacist);
```

NAME	POSITION
ATHOY	Manager

1 rows returned in 0.00 seconds

Subquery:

1. Find the name and contact information of the customer who made the sale with salesId 2.

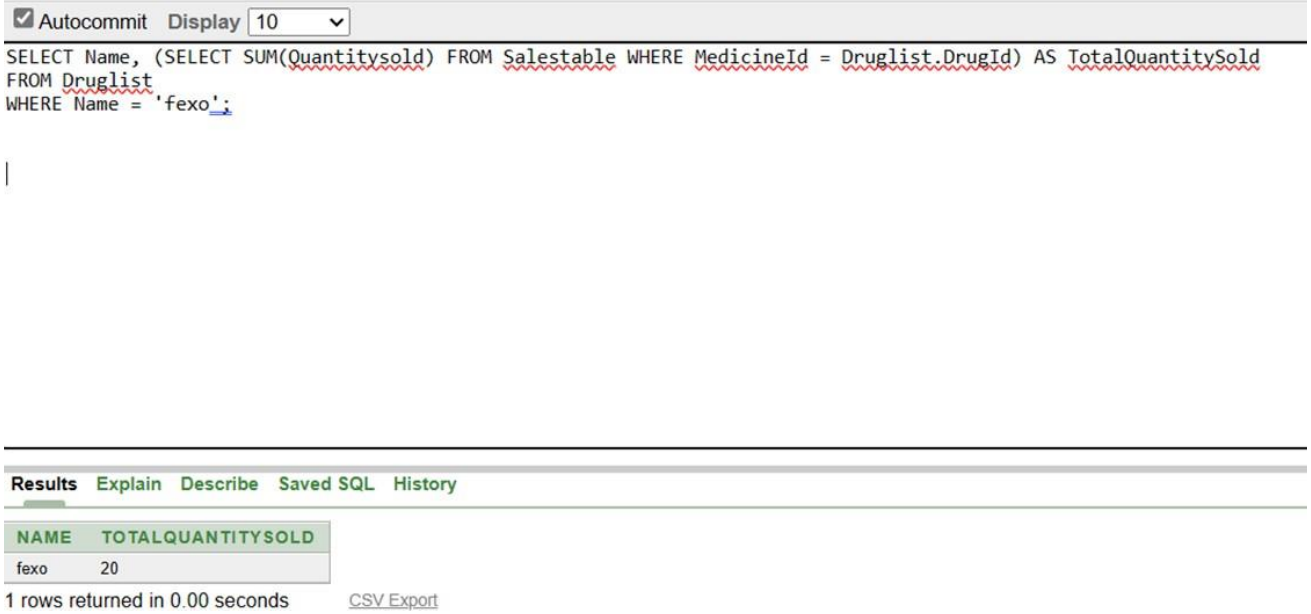


```
SELECT Name, contactinfo
FROM Customer
WHERE CustomerId = (SELECT CustomerId FROM Salestable WHERE salesId = 2);
```

NAME	CONTACTINFO
ATHFGDOY	GMAGHDIL

1 rows returned in 0.00 seconds

2. Find the total quantity sold for the medicine named "Fexo."

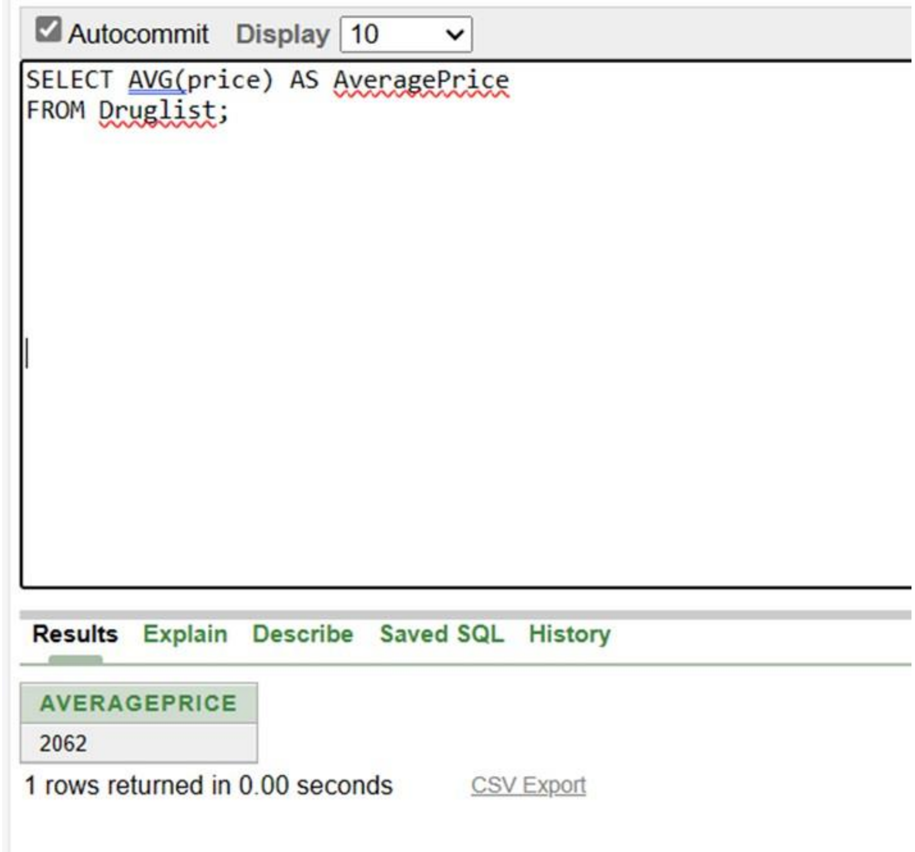


```
SELECT Name, (SELECT SUM(Quantitysold) FROM Salestable WHERE MedicineId = Druglist.DrugId) AS TotalQuantitySold
FROM Druglist
WHERE Name = 'fexo';
```

NAME	TOTALQUANTITYSOLD
fexo	20

```
1 rows returned in 0.00 seconds
```

3. Find the average price of all medicines in the Druglist table



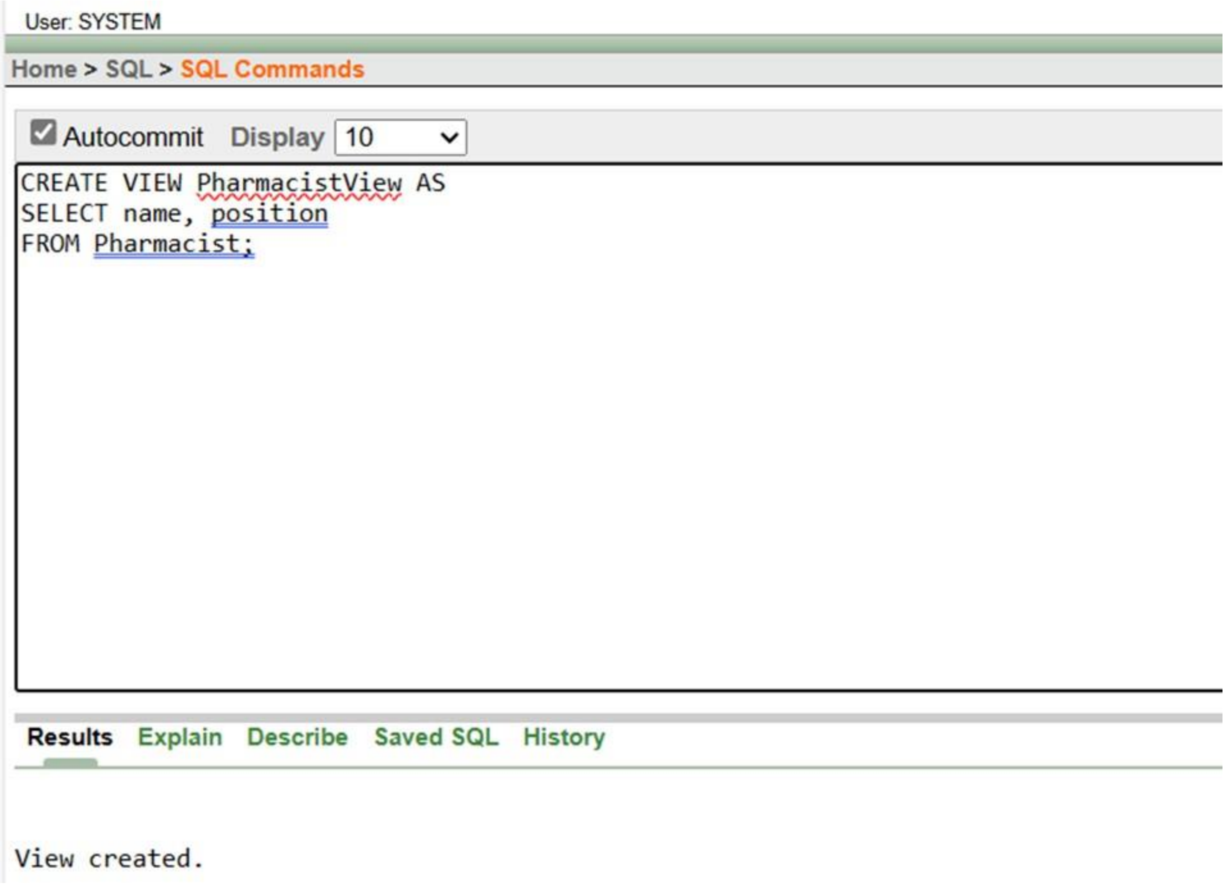
```
SELECT AVG(price) AS AveragePrice
FROM Druglist;
```

AVERAGEPRICE
2062

```
1 rows returned in 0.00 seconds
```

View:

1. Create a view that displays the names and positions of all pharmacists in the Pharmacist table.


User: SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Results Explain Describe Saved SQL History

```
CREATE VIEW PharmacistView AS
SELECT name, position
FROM Pharmacist;
```

```
View created.
```

Pl/SQL

Functions:

Function to Retrieve Medicine Information:

```
CREATE OR REPLACE FUNCTION get_medicine_info(med_id NUMBER)
RETURN VARCHAR2
AS
v_medicine_name VARCHAR2(100);
v_stock_quantity NUMBER;
BEGIN
-- Retrieve medicine information based on medicine_id
SELECT medicine_name, stock_quantity
INTO v_medicine_name, v_stock_quantity
FROM medicines
WHERE medicine_id = med_id;

-- Return formatted information
RETURN 'Medicine Name: ' || v_medicine_name || ', Stock Quantity: ' || v_stock_quantity;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN 'Medicine not found';
WHEN OTHERS THEN
RETURN 'Error retrieving medicine information';
END;
/
```

Function to Calculate Total Sales for a Given Period:

```
CREATE OR REPLACE FUNCTION calculate_total_sales(start_date DATE, end_date DATE)
RETURN NUMBER
AS
  v_total_sales NUMBER := 0;
BEGIN
  -- Calculate total sales for the given period
  SELECT SUM(total_price)
  INTO v_total_sales
  FROM sales
  WHERE sale_date BETWEEN start_date AND end_date;

  -- Return the total sales
  RETURN v_total_sales;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN 0; -- Return 0 if no sales found for the given period
  WHEN OTHERS THEN
    RETURN -1; -- Return -1 for other errors
END;
/
```

Function to Check Medicine Availability:

```
CREATE OR REPLACE FUNCTION is_medicine_available(med_id NUMBER, quantity NUMBER)
RETURN BOOLEAN
AS
  v_available_quantity NUMBER;
BEGIN
```

```
    -- Check if the required quantity is available

    SELECT stock_quantity

    INTO v_available_quantity

    FROM medicines

    WHERE medicine_id = med_id;



    -- Return TRUE if available, FALSE otherwise

    RETURN v_available_quantity >= quantity;

EXCEPTION

    WHEN NO_DATA_FOUND THEN

        RETURN FALSE; -- Medicine not found

    WHEN OTHERS THEN

        RETURN FALSE; -- Error in checking availability

END;

/
```

Producers:

Procedure to Add a New Medicine to the Inventory:

```
CREATE OR REPLACE PROCEDURE add_new_medicine(
    p_medicine_name VARCHAR2,
    p_stock_quantity NUMBER
)
AS
BEGIN
    -- Insert a new medicine into the medicines table
    INSERT INTO medicines(medicine_name, stock_quantity)
    VALUES (p_medicine_name, p_stock_quantity);

    COMMIT; -- Commit the transaction
EXCEPTION
    WHEN OTHERS THEN
        -- Handle errors (e.g., log the error or raise an exception)
        DBMS_OUTPUT.PUT_LINE('Error adding new medicine: ' || SQLERRM);
END;
/
```

Procedure to Record a Sale:

```
CREATE OR REPLACE PROCEDURE record_sale(
    p_medicine_id NUMBER,
    p_sale_quantity NUMBER,
    p_sale_price NUMBER
)
AS
BEGIN
    -- Insert a new sale record into the sales table
    INSERT INTO sales(medicine_id, sale_quantity, sale_price, sale_date)
    VALUES (p_medicine_id, p_sale_quantity, p_sale_price, SYSDATE);

    -- Update the stock_quantity in the medicines table
    UPDATE medicines
    SET stock_quantity = stock_quantity - p_sale_quantity
    WHERE medicine_id = p_medicine_id;

    COMMIT; -- Commit the transaction
EXCEPTION
    WHEN OTHERS THEN
        -- Handle errors (e.g., log the error or raise an exception)
        DBMS_OUTPUT.PUT_LINE('Error recording sale: ' || SQLERRM);
END;
/
```

Procedure to Update Medicine Information:

```
CREATE OR REPLACE PROCEDURE update_medicine_info(
    p_medicine_id NUMBER,
    p_new_name VARCHAR2,
    p_new_stock_quantity NUMBER
)
AS
BEGIN
    -- Update the medicine information in the medicines table
    UPDATE medicines
    SET medicine_name = p_new_name,
        stock_quantity = p_new_stock_quantity
    WHERE medicine_id = p_medicine_id;

    COMMIT; -- Commit the transaction
EXCEPTION
    WHEN OTHERS THEN
        -- Handle errors (e.g., log the error or raise an exception)
        DBMS_OUTPUT.PUT_LINE('Error updating medicine information: ' || SQLERRM);
END;
/
```

Record:

Inserting a Record into the Medicines Table:

```
DECLARE
   v_medicine_id NUMBER;
BEGIN
   -- Insert a new medicine record
   INSERT INTO medicines(medicine_name, stock_quantity)
   VALUES ('Aspirin', 100);

   -- Retrieve the generated medicine_id
   SELECT MAX(medicine_id) INTO v_medicine_id FROM medicines;

   DBMS_OUTPUT.PUT_LINE('Medicine record inserted with ID: ' || v_medicine_id);
END;
/
```

Inserting a Record into the Purchases Table:

```
DECLARE
   v_purchase_id NUMBER;
BEGIN
   -- Insert a new purchase record
   INSERT INTO purchases(medicine_id, purchase_quantity, purchase_date)
   VALUES (1, 50, SYSDATE); -- Assuming the medicine_id 1 corresponds to 'Aspirin'

   -- Retrieve the generated purchase_id
   SELECT MAX(purchase_id) INTO v_purchase_id FROM purchases;

   DBMS_OUTPUT.PUT_LINE('Purchase record inserted with ID: ' || v_purchase_id);
END;
/
```

Inserting a Record into the Sales Table:

```
DECLARE
   v_sale_id NUMBER;
BEGIN
   -- Insert a new sale record
   INSERT INTO sales(medicine_id, sale_quantity, sale_price, sale_date)
   VALUES (1, 30, 5.99, SYSDATE); -- Assuming the medicine_id 1 corresponds to 'Aspirin'

   -- Retrieve the generated sale_id
   SELECT MAX(sale_id) INTO v_sale_id FROM sales;

   DBMS_OUTPUT.PUT_LINE('Sale record inserted with ID: ' || v_sale_id);
END;
/
```

Cursor

Cursor to Retrieve Medicine Information:

```
DECLARE
   CURSOR medicine_cursor IS
      SELECT medicine_id, medicine_name, stock_quantity
      FROM medicines;

   v_medicine_id NUMBER;
   v_medicine_name VARCHAR2(100);
   v_stock_quantity NUMBER;
BEGIN
   OPEN medicine_cursor;
   LOOP
      FETCH medicine_cursor INTO v_medicine_id, v_medicine_name, v_stock_quantity;
      EXIT WHEN medicine_cursor%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('Medicine ID: ' || v_medicine_id || ', Name: ' || v_medicine_name || ', Stock: ' || v_stock_quantity);
   END LOOP;
   CLOSE medicine_cursor;
END;
/
```

Cursor to Calculate Total Sales for a Medicine:

```
DECLARE
   v_medicine_id NUMBER := 1; -- Assuming the medicine_id 1 corresponds to a specific medicine
   v_total_sales NUMBER := 0;

   CURSOR sales_cursor (p_medicine_id NUMBER) IS
      SELECT sale_quantity * sale_price AS total_sale
      FROM sales
      WHERE medicine_id = p_medicine_id;
BEGIN
   OPEN sales_cursor(v_medicine_id);
   LOOP
      FETCH sales_cursor INTO v_total_sales;
      EXIT WHEN sales_cursor%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('Total Sales for Medicine ID ' || v_medicine_id || ': ' || v_total_sales);
   END LOOP;
   CLOSE sales_cursor;
END;
/
```

Cursor to Display Purchase Information:

```
DECLARE

   CURSOR purchase_cursor IS

      SELECT purchase_id, medicine_id, purchase_quantity, purchase_date

      FROM purchases;
```

```
    v_purchase_id NUMBER;
    v_medicine_id NUMBER;
    v_purchase_quantity NUMBER;
    v_purchase_date DATE;
BEGIN
    OPEN purchase_cursor;
    LOOP
        FETCH purchase_cursor INTO v_purchase_id, v_medicine_id, v_purchase_quantity, v_purchase_date;
        EXIT WHEN purchase_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Purchase ID: ' || v_purchase_id || ', Medicine ID: ' || v_medicine_id ||
                             ', Quantity: ' || v_purchase_quantity || ', Date: ' || v_purchase_date);
    END LOOP;
    CLOSE purchase_cursor;
END;
/
```

Triggerql

Trigger to Update Stock Quantity After a Sale:

```
CREATE OR REPLACE TRIGGER update_stock_after_sale
AFTER INSERT ON sales
FOR EACH ROW
DECLARE
    v_medicine_id NUMBER;
    v_sale_quantity NUMBER;
BEGIN
```

```
    -- Retrieve medicine_id and sale_quantity from the new sale record
    v_medicine_id := :NEW.medicine_id;
    v_sale_quantity := :NEW.sale_quantity;

    -- Update the stock_quantity in the medicines table
    UPDATE medicines
    SET stock_quantity = stock_quantity - v_sale_quantity
    WHERE medicine_id = v_medicine_id;
END;
/
```

Trigger to Enforce Minimum Stock Quantity Threshold:

```
CREATE OR REPLACE TRIGGER check_minimum_stock
BEFORE UPDATE ON medicines
FOR EACH ROW
DECLARE
    v_minimum_stock NUMBER := 10; -- Set your desired minimum stock threshold
BEGIN
    -- Check if the new stock_quantity falls below the minimum threshold
    IF :NEW.stock_quantity < v_minimum_stock THEN
        -- Raise an exception or take appropriate action
        RAISE_APPLICATION_ERROR(-20001, 'Stock quantity cannot fall below the minimum threshold.');
    END IF;
END;
/
```

Trigger to Log High-Value Purchases:

```
CREATE OR REPLACE TRIGGER log_high_value_purchase
```

```
AFTER INSERT ON purchases
FOR EACH ROW
DECLARE
    v_purchase_value NUMBER;
BEGIN
    -- Calculate the total value of the purchase
    v_purchase_value := :NEW.purchase_quantity * :NEW.purchase_price;

    -- Check if the purchase value exceeds a certain threshold
    IF v_purchase_value > 1000 THEN
        -- Log the high-value purchase (you can modify this part based on your logging mechanism)
        INSERT INTO purchase_logs(purchase_id, purchase_value, log_date)
        VALUES (:NEW.purchase_id, v_purchase_value, SYSDATE);
    END IF;
END;
/
```

package

Package for Medicine Management:

```
CREATE OR REPLACE PACKAGE medicine_package AS
    PROCEDURE add_new_medicine(
        p_medicine_name VARCHAR2,
        p_stock_quantity NUMBER
    );

    FUNCTION get_medicine_info(
        p_medicine_id NUMBER
```

```
  ) RETURN VARCHAR2;

  PROCEDURE update_medicine_info(
    p_medicine_id NUMBER,
    p_new_name VARCHAR2,
    p_new_stock_quantity NUMBER
  );

  PROCEDURE delete_medicine(
    p_medicine_id NUMBER
  );
END medicine_package;
/

CREATE OR REPLACE PACKAGE BODY medicine_package AS
  PROCEDURE add_new_medicine(
    p_medicine_name VARCHAR2,
    p_stock_quantity NUMBER
  ) AS
  BEGIN
    -- Implement the logic to add a new medicine
    -- Insert into medicines table
  END;

  FUNCTION get_medicine_info(
    p_medicine_id NUMBER
  ) RETURN VARCHAR2 AS
```

```
    v_medicine_info VARCHAR2(200);
  BEGIN
    -- Implement the logic to retrieve medicine information
    -- Query the medicines table
    RETURN v_medicine_info;
  END;

  PROCEDURE update_medicine_info(
    p_medicine_id NUMBER,
    p_new_name VARCHAR2,
    p_new_stock_quantity NUMBER
  ) AS
  BEGIN
    -- Implement the logic to update medicine information
    -- Update the medicines table
  END;

  PROCEDURE delete_medicine(
    p_medicine_id NUMBER
  ) AS
  BEGIN
    -- Implement the logic to delete a medicine
    -- Delete from medicines table
  END;
END medicine_package;
/
```

Package for Sales Management:

```
CREATE OR REPLACE PACKAGE sales_package AS
    PROCEDURE record_sale(
        p_medicine_id NUMBER,
        p_sale_quantity NUMBER,
        p_sale_price NUMBER
    );

    FUNCTION calculate_total_sales(
        p_start_date DATE,
        p_end_date DATE
    ) RETURN NUMBER;

    PROCEDURE void_sale(
        p_sale_id NUMBER
    );
END sales_package;
/

CREATE OR REPLACE PACKAGE BODY sales_package AS
    PROCEDURE record_sale(
        p_medicine_id NUMBER,
        p_sale_quantity NUMBER,
        p_sale_price NUMBER
    ) AS
    BEGIN
        -- Implement the logic to record a sale
        -- Insert into sales table
        -- Update stock_quantity in medicines table
    END;

    FUNCTION calculate_total_sales(
        p_start_date DATE,
        p_end_date DATE
    ) RETURN NUMBER AS
        v_total_sales NUMBER;
    BEGIN
        -- Implement the logic to calculate total sales
        -- Query the sales table
        RETURN v_total_sales;
    END;

    PROCEDURE void_sale(
        p_sale_id NUMBER
    ) AS
    BEGIN
        -- Implement the logic to void a sale
        -- Delete from sales table
        -- Update stock_quantity in medicines table
    END;
END sales_package;
/
```

Package for Purchase Management:

```
CREATE OR REPLACE PACKAGE purchase_package AS
    PROCEDURE make_purchase(
    p_medicine_id NUMBER,
    p_purchase_quantity NUMBER,
    p_purchase_price NUMBER
  );

  PROCEDURE cancel_purchase(
    p_purchase_id NUMBER
  );
END purchase_package;
/

CREATE OR REPLACE PACKAGE BODY purchase_package AS
  PROCEDURE make_purchase(
    p_medicine_id NUMBER,
    p_purchase_quantity NUMBER,
    p_purchase_price NUMBER
  ) AS
  BEGIN
    -- Implement the logic to make a purchase
    -- Insert into purchases table
    -- Update stock_quantity in medicines table
  END;

  PROCEDURE cancel_purchase(
    p_purchase_id NUMBER
  ) AS
  BEGIN
```

```
        -- Implement the logic to cancel a purchase
        -- Delete from purchases table
        -- Update stock_quantity in medicines table
    END;
END purchase_package;
/
```

Relational Algebra:

1. List of Drug names, id, company name

Ans: πDrug_ID, Drug_Name, Drug_Date,

2. List of all Pharmacist who are

Ans: πMember_Name(

σMember_Role='Pharmacist'(Member))

3. List of all Customers

Ans:

πcustomer_ID, customer_Name, Event_(customername))

4. List SalseTable where selling drug

Ans:

πsales_ID, sales_Name, drug_ID (

σProduct_ID is not null(SalesTable))

5. List of Admin

Ans: πAdmin_ID,Admin_Name,drug_ID(

σdrug_ID is not null (Druglist**)**

Conclusion:

In conclusion, the development of a pharmacy management system is crucial for enhancing the efficiency of pharmaceutical operations. This project aims to address the complexities and challenges faced by traditional pharmacy systems by introducing a comprehensive and technologically advanced solution.In conclusion, the development of a pharmacy management system is crucial in streamlining and enhancing the efficiency of pharmaceutical operations. This project aims to address the complexities and challenges faced by traditional pharmacy systems by introducing a comprehensive and technologically advanced solution.Through the implementation of this system, we anticipate a significant improvement in various aspects of pharmacy management, including employee control, medicine processing, customer management, sales control and overall workflow optimization. The integration of features such as automated sales tracking, medicine management, and real-time reporting not only reduces the likelihood of errors but also enhances the overall accuracy and speed of operations.In future we will improve our system by integrating with telehealth platforms to facilitate virtual consultations and prescription deliveries.Stay updated on telehealth regulations and adapt the system accordingly**.**