PHARMACY STORE

Milind Jain CSE (2021165)
Abhijay Singh CSAM (2021226)

Project Scope:

With the growing popularity of online pharmaceutical stores, our aim is to develop a solution that is the easiest to access and fulfils all the different needs of our customers. Our main focus is to build an efficient functioning database management system for our pharmacy store. Organising and making a database of all the products like medicine, baby care, women's care, and even covid essentials with their price and quantity etc., attached to them in an easy-to-sort way is our goal. Constructing and maintaining a database for the customer records as well as the vendor's is what we hope to achieve by the end of the project. On the front end of things, we hope to build a simple, interactive and catchy interface that makes it clear to the customer of everything they can buy and have a hassle free shopping experience.

Technical Requirements:

Back end:

1. MySQL (Relational Database Management System)
2. Python (Programming Language)
3. Django (Python based back end framework)

Front end:

1. HTML
2. CSS(SASS)
3. JS(JSX)

Functional requirements:

Customer

Login/Signup
Create a new customer account

Browsing through the stock
Browse through a large selection of drugs and select the desired drugs and specify the quantity for each drug requred by you.

Managing Cart

Upon selection of the desired drugs of desire and respective quantities you can choose to add them to the cart after every selection is made.

Doctor Verification
Upload an image of your doctor's prescription and submit it for approval. Approval is done by our team of medical experts and takes about 2 hours. After approval is done, you can move to the next step.

Checkout
After approval is done, you can move to the checkout section. In the checkout make the payment after applying any coupon available using your desired payment method. Order is placed after payment verification.

Tracking and Updates
Track the order placed live and get updates on e-mail/phone.

Cancel their order
If you change your mind

Give feedback
Have any feedbacks or complaints you can get back to us.

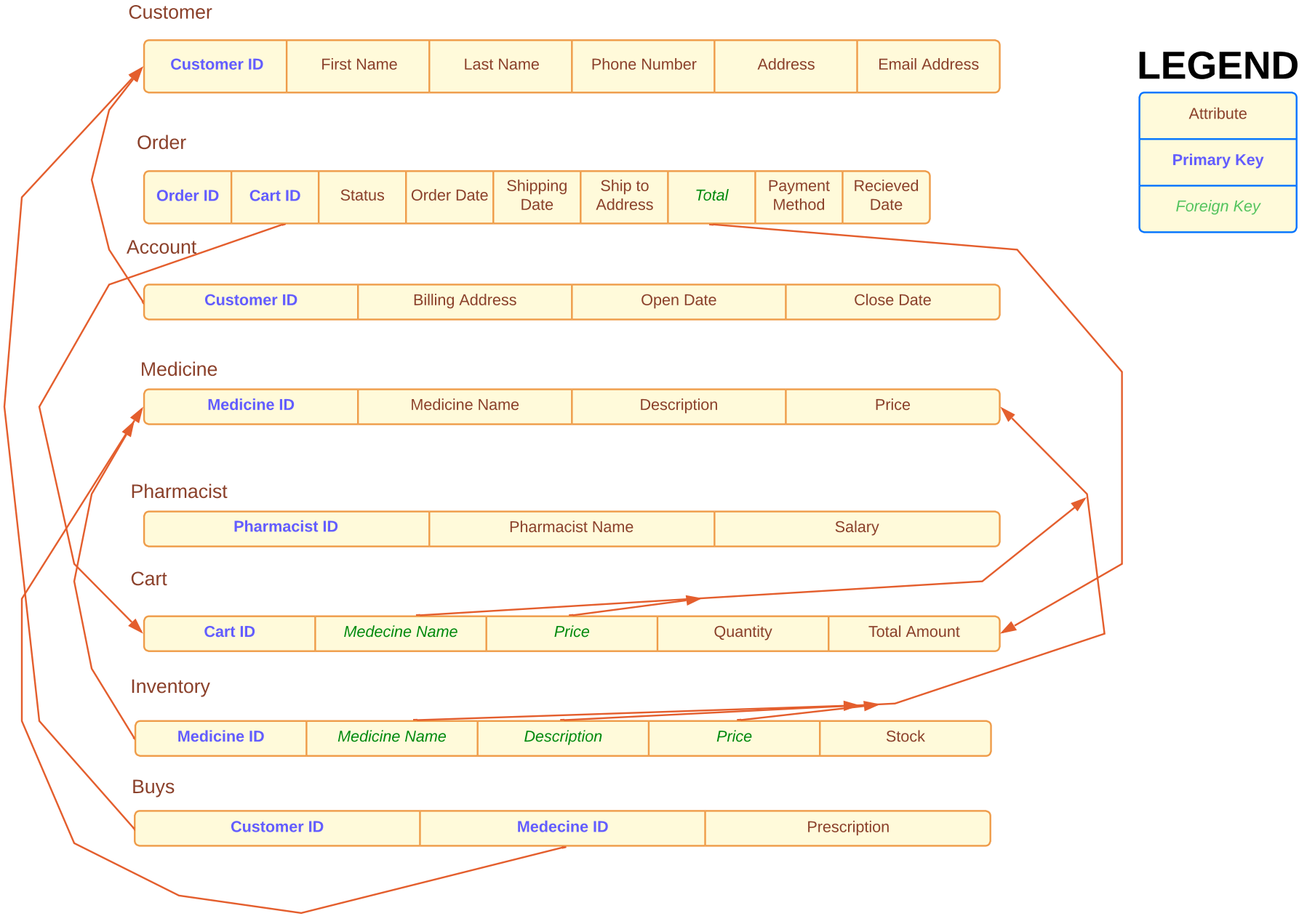
Admin

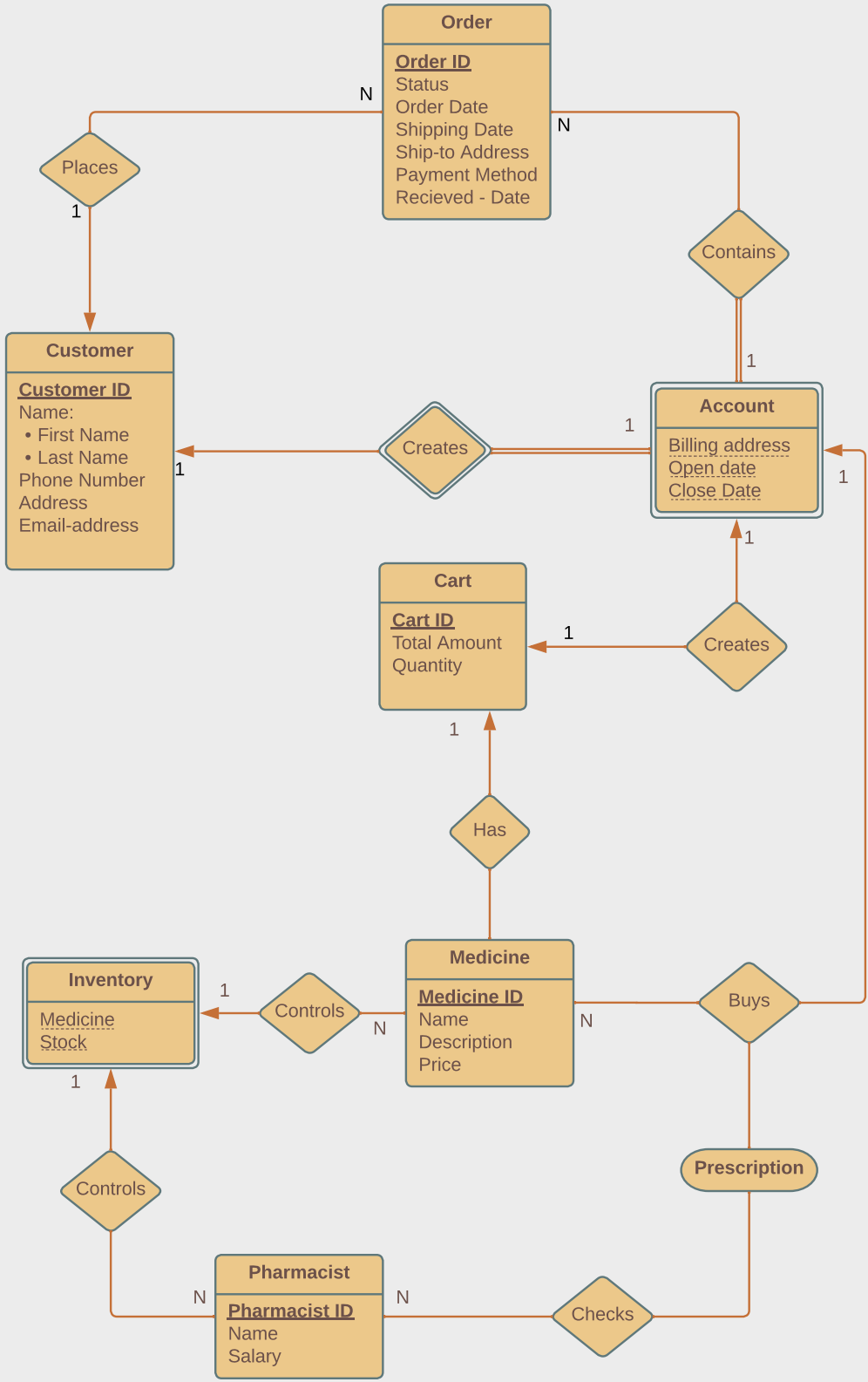
Stock Management
Add/Delete/Change any listed product.
Maintain quantity and price of any particular product/medicine.
View all the current, completed, and pending orders
View and approve a prescription uploaded by a customer

View Earnings
View total earnings and sales

Security and Privacy
Implement a system which ensures the safety and privacy of user data and maintaining a firewall.

View customer feedback
Add/Delete/Change customer account


Customer
Customer ID
First Name
Last Name
Phone Number
Address
Email Address
Order
Order ID
Cart ID
Status
Order Date
Shipping Date
Ship to Address
Total
Payment Method
Recieved Date
Account
Customer ID
Billing Address
Open Date
Close Date
Medicine
Medicine ID
Medicine Name
Description
Price
Pharmacist
Pharmacist ID
Pharmacist Name
Salary
Cart
Cart ID
Medecine Name
Price
Quantity
Total Amount
Inventory
Medicine ID
Medicine Name
Description
Price
Stock
Buys
Customer ID
Medecine ID
Prescription
LEGEND
Attribute
Primary Key
Foreign Key


Order
Order ID
Status
Order Date
Shipping Date
Ship-to Address
Payment Method
Recieved - Date
N
N
Places
1
Contains
1
Customer
Customer ID
Name:
• First Name
• Last Name
Phone Number
Address
Email-address
1
Creates
1
Account
Billing address
Open date
Close Date
1
1
Cart
Cart ID
Total Amount
Quantity
1
Creates
1
Has
1
Inventory
Medicine
Stock
1
Controls
N
Medicine
Medicine ID
Name
Description
Price
N
Buys
1
Controls
Prescription
N
Pharmacist
Pharmacist ID
Name
Salary
N
Checks

PROJECT DEADLINE-4

Fundamentals of DBMS

MILIND JAIN CSE 2021165
ABHIJAY SINGH CSAM 2021226

QUERY 1 (Listing customer details)

SELECT first_name, phone_number, address FROM pharmacy.customer where customer_id=1887

	first_name	phone_number	address
▶	Selene	7626761979	Suite 35

QUERY 2 (Listing some cart contents along with Order Amount using join)

SELECT DISTINCT cart.cart_id, orderr.total_amount, orderr.status

FROM cart,orderr

WHERE cart.cart_id=orderr.cart_id

	cart_id	total_amou...	status
▶	9955	223	Processing
	8724	37235	Processing
	9204	47631	Processing
	9586	58840	Delivered
	9302	61014	Shipped
	9668	62152	Delivered
	9577	68962	Delivered
	9068	82726	Processing
	8547	85380	Delivered
	9281	86057	Shipped
	9355	93804	Shipped
	8826	122302	Delivered
	9380	145174	Delivered

QUERY 3 (Adding a new customer)

INSERT INTO customer (`customer_ID`, `first_name`,`last_name`, `phone_number`, `address`, `email_address`) VALUES ('10000', 'Abhijay','Singh', '2021309000', 'Noida', 'abhijay21226@iiitd.ac.in');

SELECT * FROM customer;

9584	Kristofor	Halford	1522582064	Apt 965	khalford16@istockphoto.com
9670	Vito	Goneau	9521815654	Room 910	vgoneau2a@cmu.edu
10000	Abhijay	Singh	2021309000	Noida	abhijay21226@iiitd.ac.in
NULL	NULL	NULL	NULL	NULL	NULL

QUERY 4 (Changing address for a customer)

UPDATE customer SET `address` = 'Delhi' WHERE (`Customer_ID` = '10000');

SELECT * FROM customer;

9670	Vito	Goneau	9521815654	Room 910	vgoneau2a@cmu.edu
10000	Abhijay	Singh	2021309000	Delhi	abhijay21226@iiitd.ac.in
NULL	NULL	NULL	NULL	NULL	NULL

QUERY 5 (Creating a new empty cart)

INSERT INTO cart (`cart_content_id`,`cart_id`, `total_amount`) VALUES ('305','9956', '0');

SELECT * FROM cart;

301	9956	NULL	NULL	NULL	0
NULL	NULL	NULL	NULL	NULL	NULL

QUERY 6 (Adding items to the created cart)

INSERT INTO `pharmacy`.`cart` (`cart_content_id`, `cart_id`, `medicine_name`, `price`, `quantity`,`total_amount`) VALUES ('301', '9956', 'Doxycyclin', '250','2','500');

INSERT INTO `pharmacy`.`cart` (`cart_content_id`, `cart_id`, `medicine_name`, `price`, `quantity`,`total_amount`) VALUES ('302', '9956', 'Paracetamol', '2800','2','5600');

INSERT INTO `pharmacy`.`cart` (`cart_content_id`, `cart_id`, `medicine_name`, `price`, `quantity`,`total_amount`) VALUES ('303', '9956', 'Levocet', '3500','1','3500');

INSERT INTO `pharmacy`.`cart` (`cart_content_id`, `cart_id`, `medicine_name`, `price`, `quantity`,`total_amount`) VALUES ('304', '9956', 'MontairLC', '900','3','2700');

SELECT * FROM cart;

300	9955	Trihexyphenidyl Hydrochloride	2157	7	15099
301	9956	Doxycyclin	250	2	500
302	9956	Paracetamol	2800	2	5600
303	9956	Levocet	3500	1	3500
304	9956	MontairLC	900	3	2700
NULL	NULL	NULL	NULL	NULL	NULL

QUERY 7 (Updating stock after a purchase has been made)

UPDATE inventory SET stock = stock - 2

WHERE medicine_id = '9942';

SELECT * FROM inventory;

9942	Ciprofloxacin	Ciprofloxacin Hydrochloride	139	28

9942	Ciprofloxacin	Ciprofloxacin Hydrochloride	139	26

QUERY 8 (Selecting certain shipped orders)

SELECT * FROM orderr WHERE order_id>60000 and orderr.status = 'Shipped'

ORDER BY total_amount DESC;

order_id	cart_id	status	order_date	shipping_date	ship_to_addre...	total_amou...	payment_meth...	received_date
62464	3401	Shipped	2014-04-22 18:52:00	2003-04-22 17:45:00	PO Box 10238	165373	Card	2005-05-22 19:25:00
64861	5035	Shipped	2011-04-22 15:50:00	2022-04-22 10:18:00	PO Box 16209	152535	Card	2014-04-22 12:16:00
90225	5719	Shipped	2023-04-22 20:39:00	2021-04-22 11:32:00	Suite 23	150634	Net Banking	2011-04-22 17:03:00
76851	4483	Shipped	2020-04-22 06:49:00	2024-04-22 04:41:00	PO Box 80678	112009	Cash	2019-04-22 06:00:00
89383	9355	Shipped	2018-04-22 12:09:00	2014-04-22 01:51:00	Suite 47	93804	UPI	2024-04-22 05:17:00
63006	9281	Shipped	2008-04-22 16:36:00	2028-04-22 06:36:00	3rd Floor	86057	Card	2030-04-22 08:07:00
73038	9302	Shipped	2008-04-22 05:37:00	2016-04-22 03:16:00	Apt 902	61014	Card	2012-05-22 09:41:00
95273	8416	Shipped	2008-04-22 14:33:00	2012-04-22 20:35:00	PO Box 96419	60624	Cash	2006-04-22 01:47:00
96479	7858	Shipped	2021-04-22 19:02:00	2011-04-22 10:15:00	Apt 1267	40282	Net Banking	2002-04-22 04:07:00
62476	2556	Shipped	2026-04-22 12:28:00	2006-04-22 20:15:00	PO Box 45486	32602	Card	2030-05-22 23:05:00
73570	8397	Shipped	2006-04-22 05:30:00	2023-04-22 06:03:00	Suite 94	12050	Cash	2020-04-22 07:44:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

QUERY 9 (Deleting old orders that are delivered)

DELETE FROM orderr

WHERE status='Delivered' and order_date<'2005-04-22 00:00:00'

QUERY 10 (Selecting customers having valid Email-IDS)

SELECT * FROM customer

WHERE regexp_like(email_address,'google') or regexp_like(email_address,'yahoo');

custome...	first_name	last_name	phone_number	address	email_address
1063	Julianne	Cardno	4972439453	Suite 16	jcardnoj@google.fr
2582	Francisca	McAllester	9531435799	PO Box 1214	fmcallester18@google.com.au
2738	Toby	Stammirs	1943425197	20th Floor	tstammirs20@google.com
3556	Lorita	Bowker	3667197762	Apt 225	lbowker2k@google.com
4190	Remus	Goering	5754195796	16th Floor	rgoeringb@google.co.uk
6704	Candis	Hinkes	4579004688	1st Floor	chinkes11@google.com.br
7085	Gayelord	Valasek	7017862516	PO Box 11591	gvalasek1k@yahoo.com
NULL	NULL	NULL	NULL	NULL	NULL

PROJECT DEADLINE-5

Fundamentals of DBMS

MILIND JAIN CSE 2021165
ABHIJAY SINGH CSAM 2021226

EMBEDDED SQL QUERIES

1. **LOGIN (Checks the entered login info against our database to authenticate login)**
   SELECT customer_id FROM pharmacy.customer WHERE customer_id={customer_id}

```
#Login
if option==1:
    while (True):
        customer_id=int(input("Enter cutomer id: "))
        pswd=input("Enter password (Hint: password is same as your phone number): ")
        query=f"SELECT customer_id FROM pharmacy.customer WHERE customer_id={customer_id}"
        cursor.execute(query)
        row = cursor.fetchone()

        if (int(row[0])==customer_id):
            print(line)
            print("Login Successful")
            print(line)
            break
        else:
            print(line)
            print("Login failed, enter again")
            print(line)
```

2. **SIGNUP (Inserts a new customer row into the customer table)**
   INSERT INTO customer (`customer_ID`, `first_name`,`last_name`, `phone_number`, `address`,`email_address`) VALUES ('{customerid}', '{first_name}','{last_name}', '{phone}', '{addr}', '{email}')

```
#Signup
if option==2:
    first_name=input("Enter First Name: ")
    last_name=input("Enter Last Name: ")
    phone=input("Enter phone number: ")
    addr=input("Enter address: ")
    email=input("Enter email address: ")
    customerid=(random,random.randint(10001,20000))
    query = f"INSERT INTO customer (`customer_ID`, `first_name`,`last_na
    cursor.execute(query)
    print("")
    print("Signup successful!")

    print(f"Your customer id: {customerid}")
    print(f"Your password: {phone}")
```

3. **VIEW ADMIN INFO (Lists various tables of our database)**
   SELECT * FROM pharmacy.account
   SELECT * FROM pharmacy.customer
   SELECT * FROM pharmacy.inventory
   SELECT * FROM pharmacy.medicine
   SELECT * FROM pharmacy.orderr
   SELECT * FROM pharmacy.pharmacist

```
optionadmin=int(input("Choose the desired option: "))

if optionadmin==1:
    cursor.execute("SELECT * FROM pharmacy.account")
    print_table(cursor)

if optionadmin==2:
    cursor.execute("SELECT * FROM pharmacy.customer")
    print_table(cursor)

if optionadmin==3:
    cursor.execute("SELECT * FROM pharmacy.inventory")
    print_table(cursor)

if optionadmin==4:
    cursor.execute("SELECT * FROM pharmacy.medicine")
    print_table(cursor)

if optionadmin==5:
    cursor.execute("SELECT * FROM pharmacy.orderr")
    print_table(cursor)

if optionadmin==6:
    cursor.execute("SELECT * FROM pharmacy.pharmacist")
    print_table(cursor)
```

OLAP QUERIES

1. **Lists total sales of goods over different years through different modes of payment :**

```
SELECT YEAR(received_date), payment_method, SUM(total_amount)
FROM orderr
GROUP BY YEAR(received_date), payment_method WITH ROLLUP
```

2. **Lists the buying trends of customers since opening their accounts:**

```
SELECT quarter(open_date),SUM(total), quarter(shipping_date)
FROM orderr join account on orderr.id = account.customer_id where status = "delivered"
GROUP BY quarter(open_date), quarter(shipping_date) WITH ROLLUP
```

3. **Lists average number of medicines purchased by each customer over different orders:**

```
SELECT cart_id,AVG(quantity)
FROM orderr join cart on orderr.cart_id = cart.cart_id
GROUP BY cart_id WITH ROLLUP
```

4. **Lists the quarterly trends of number of goods sold and total sales done on average:**

```
SELECT quarter(shipping_date), payment_method, avg(total_amount),avg(quantity)
FROM orderr join cart on orderr.cart_id = cart.cart_id
GROUP BY quarter(shipping_date), payment_method, status WITH ROLLUP
```

TRIGGERS

1.
```
delimiter //
CREATE TRIGGER `update_status`
AFTER UPDATE ON `orderr`
FOR EACH ROW
BEGIN
   IF NEW.received_date !=OLD.received_date THEN

      UPDATE `orderr`
      SET `status` = 'delivered'
      WHERE `orderr`.cart_id = NEW.cart_id;
   END IF;
END;
delimiter;
```

2.
```
delimiter //
CREATE TRIGGER `insert_status`
AFTER INSERT ON `orderr`
FOR EACH ROW
BEGIN

      UPDATE `orderr`
      SET `status` = 'processing'
      WHERE `orderr`.cart_id = NEW.cart_id;

END;
delimiter;
```

PROJECT DEADLINE-6

Fundamentals of DBMS

MILIND JAIN CSE 2021165
ABHIJAY SINGH CSAM 2021226

NON CONFLICTING TRANSACTIONS

1. START TRANSACTION;
   INSERT INTO customer (customer_id, first_name, last_name, phone_number, address, email_address) VALUES (1, 'John', 'Doe', '555-1234', '123 Main St', 'johndoe@email.com');
   COMMIT;

2. START TRANSACTION;
   INSERT INTO account (customer_id, billing_address, open_date) VALUES (1, '456 Billing St', NOW());
   COMMIT;

3. START TRANSACTION;
   INSERT INTO orderr (order_id, cart_id, status, order_date, total_amount, payment_method) VALUES (1, 1, 'Pending', NOW(), 100, 'Credit Card');
   INSERT INTO cart (cart_content_id, cart_id, medicine_name, price, quantity, total_amount) VALUES (1, 1, 'Medicine A', 25, 4, 100);
   COMMIT;

4. START TRANSACTION;
   INSERT INTO product (product_id, name, price) VALUES (1, 'Medicine A', 25);
   COMMIT;

CONFLICTING TRANSACTIONS

1.
```
-- Transaction 1
START TRANSACTION;
INSERT INTO account (customer_id, billing_address, open_date)
VALUES (1, '456 Billing St', NOW());
COMMIT;
```

```
-- Transaction 2
START TRANSACTION;
INSERT INTO account (customer_id, billing_address, open_date)
VALUES (1, '789 Billing St', NOW());
COMMIT;
```

The two transactions are conflicting because they are trying to insert a new row into the same account table with the same customer_id value of 1. Since the customer_id column is the primary key of the account table, two rows cannot have the same value for customer_id.

If both transactions are executed one after the other, the first transaction will successfully insert a new row into the account table with billing_address set to '456 Billing St'. However, the second transaction will fail to insert a new row with the same customer_id value of 1 and will result in a primary key violation error.

To solve this conflict, one possible solution is to use the SELECT ... FOR UPDATE statement to lock the rows in the account table with the same customer_id value before inserting a new row. This ensures that only one transaction can modify the rows at a time and prevents conflicts.

2.

```
-- Transaction 1
START TRANSACTION;
INSERT INTO orderr (order_id, cart_id, status, order_date, total_amount, payment_method)
VALUES (1, 1, 'Pending', NOW(), 100, 'Credit Card');
INSERT INTO cart (cart_content_id, cart_id, medicine_name, price, quantity, total_amount)
VALUES (1, 1, 'Medicine A', 25, 4, 100);
COMMIT;

-- Transaction 2
START TRANSACTION;
INSERT INTO orderr (order_id, cart_id, status, order_date, total_amount, payment_method)
VALUES (2, 1, 'Pending', NOW(), 100, 'Credit Card');
INSERT INTO cart (cart_content_id, cart_id, medicine_name, price, quantity, total_amount)
VALUES (2, 1, 'Medicine A', 25, 4, 100);
COMMIT;
```

This is a conflicting transaction because both transactions are trying to insert data into the same cart table with the same cart_id value. The cart_id column in the cart table is referenced as a foreign key in the orderr table, so both transactions are also trying to insert data into the same orderr table with the same cart_id value. This will result in a constraint violation error, as the cart_id column in the orderr table references the cart_id column in the cart table as a foreign key.

The expected output is a constraint violation error due to the foreign key constraint between the cart and orderr tables.

To solve this conflict, we can use a different cart_id value in each transaction. For example, we can use cart_id = 1 in Transaction 1 and cart_id = 2 in Transaction 2. This will allow both transactions to insert data into the cart and orderr tables without violating the foreign key constraint.

USER GUIDE

- The code is a Python program for managing an online pharmacy.
- The program interacts with a MySQL database and prompts the user to enter the password and database name for the MySQL server.
- The program presents a menu to the user, offering options to login, sign up, enter as an admin, run OLAP queries, manage triggers, or exit the program.
- If the user chooses to login, they are prompted to enter their customer ID and password.
- The program checks the database for a matching customer ID and password combination.
- If the login is successful, the user is presented with a menu offering options to search for a medicine, view all products, add items to their cart, remove items from their cart, view their cart, place an order, track the order status, or logout.
- If the user chooses to search for a medicine, they are prompted to enter the name of the medicine.
- The program queries the database for medicines with a name containing the search string and displays the results in a table format.
- If the user chooses to view all products, the program queries the database for all medicines and displays the results in a table format.
- If the user chooses to add items to their cart, they are prompted to enter the medicine ID and quantity of the medicine they wish to purchase.
- The program queries the database to verify that the requested quantity is available in stock.
- If the requested quantity is available, the program calculates the total cost of the items and inserts a new row into the cart table.
- If the user chooses to remove items from their cart, they are prompted to enter the medicine ID of the item they wish to remove.
- The program then deletes the corresponding row from the cart table.
- If the user chooses to view their cart, the program queries the cart table for all items with a matching customer ID and displays the results in a table format.
- If the user chooses to place an order, the program first checks that the user has items in their cart.
- If so, the program inserts a new row into the orders table with the relevant information and updates the stock levels in the inventory table.
- If the user chooses to track the order status, the program prompts the user to enter their order ID and queries the orders table for the corresponding order status.
- The program uses a combination of SQL queries and Python constructs to provide a user-friendly interface.