MedMate

25.01.2022 - Present

Prashasti Gupta 2021346

Prajna Vohra 2021345

Github

INTRODUCTION

Inventory management is critical to any pharmaceutical business, particularly for non-moving and short-expiry drugs. It is essential to keep track of expiration dates and ensure that products are used or disposed of appropriately. Proper inventory management can help prevent losses due to expired or damaged products and ensure sufficient stock is available to meet customer demand.

Medmate is a platform designed to help pharmacists streamline these processes and improve the efficiency of their operations. By providing a centralized database of available drugs, customers, manufacturers, and delivery partners, Medmate can help pharmacies keep track of their inventory, manage medical records, and ensure that customers receive the medications they need promptly and efficiently.

Medmate can help pharmaceutical companies improve their inventory management, enhance customer service, and streamline their operations, ultimately leading to improved outcomes for both the company and its customers.

STAKEHOLDERS

CUSTOMERS

Medmate's customers are the primary users interacting with the platform's database. They have the freedom to browse the available medicines, add them to their cart, and place orders for delivery. Furthermore, customers can view medicines from specific drug manufacturers and access detailed product information before purchasing.

EMPLOYEES

Through Medmate, employees can easily access information about the medicines available within the company, including details about the drug manufacturers that supply them, information about delivery partners, and records of orders placed. Medmate provides a platform for employees to monitor their monthly sales and track their performance.

DRUG MANUFACTURERS & DELIVERY PARTNER

In addition, Medmate's delivery partners and drug manufacturers are also crucial stakeholders. These organizations rely on the platform to access information about orders and inventory, communicate with customers and pharmacies, and track their own performance and sales.

FUNCTIONALITIES AND WORKING

We have made two interfaces for the users, one for customers and one for employees

CUSTOMER INTERFACE

Customers first get an option to log-in to their existing accounts or create a new account by signing up. After they log-in to their accounts, they get a range of options to choose from:

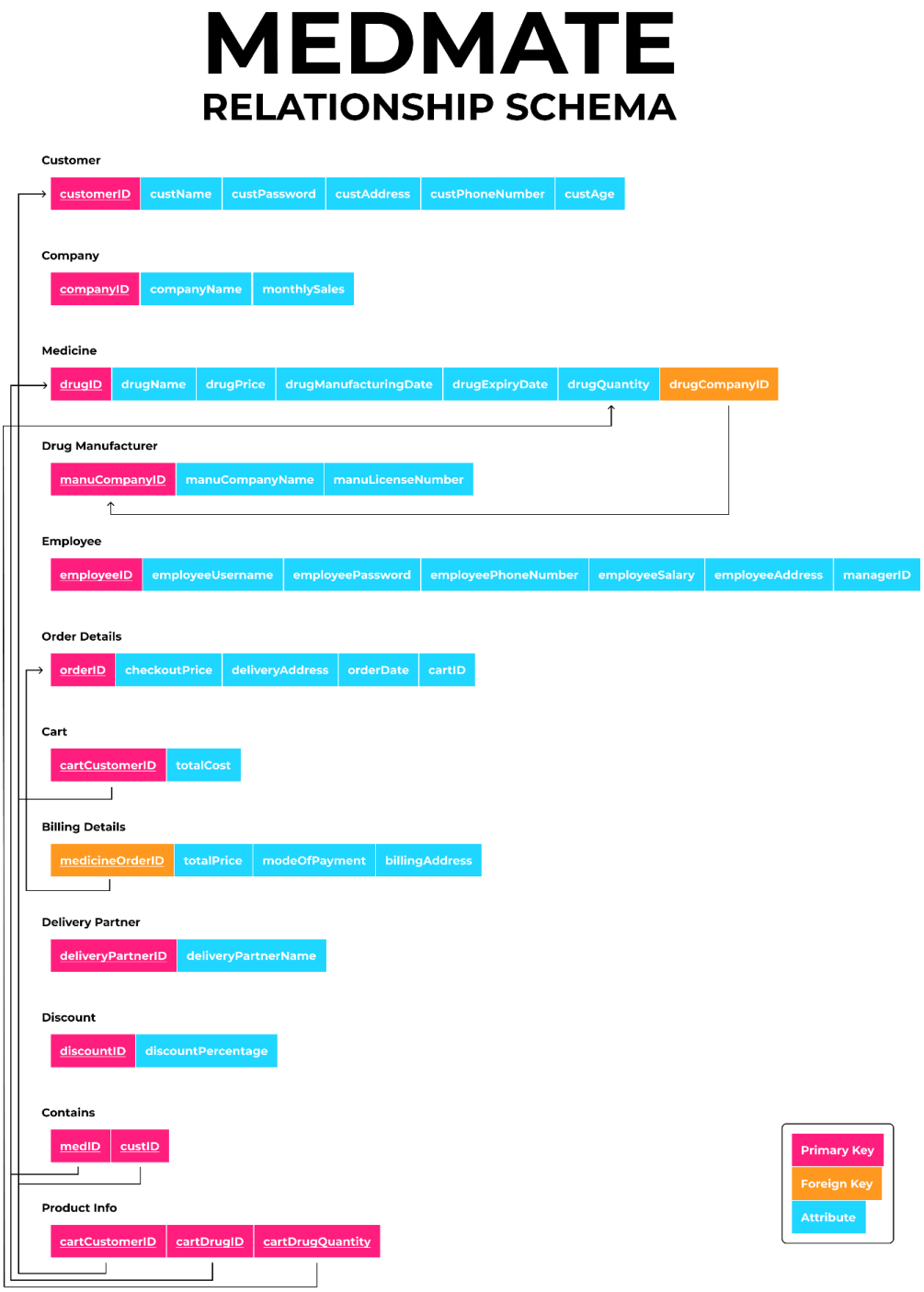
1. Buy Medicines
2. View all medicines available
3. View Medicines by a specific Drug Manufacturer
4. View Cart
5. Remove medicines from cart or Update quantities
6. Proceed to Checkout

EMPLOYEE INTERFACE

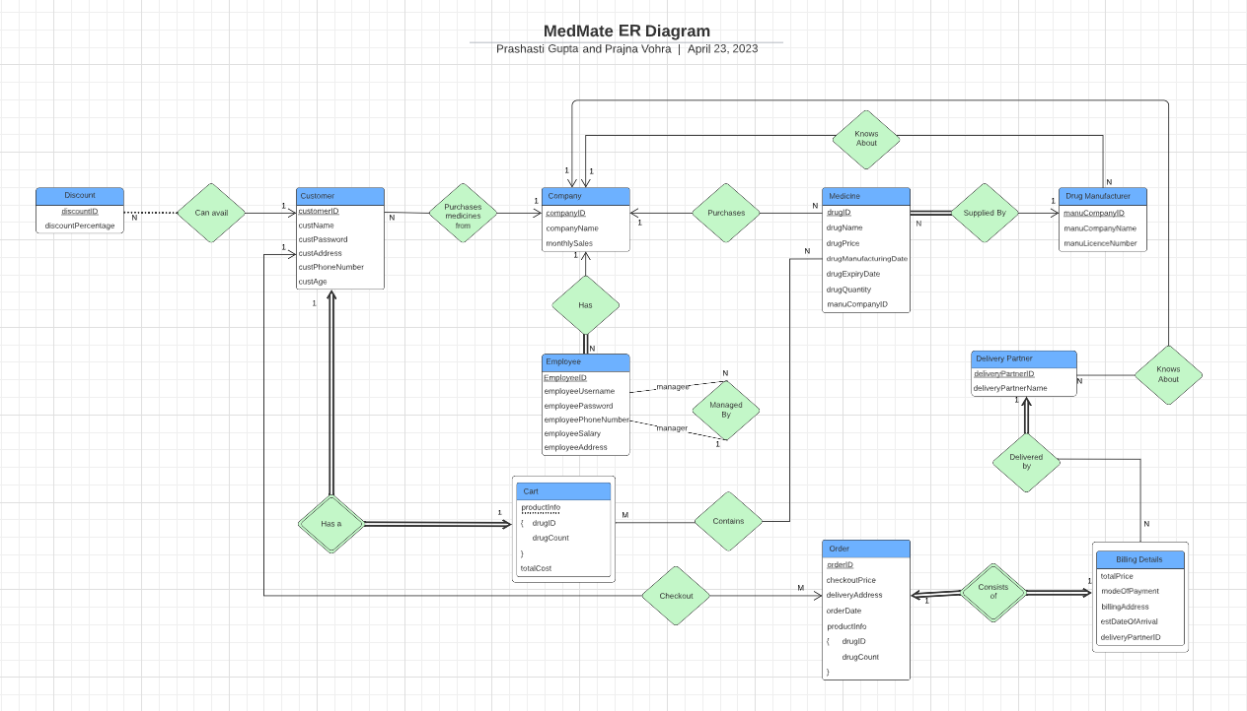
Employees have access to perform special tasks like adding a new medicine to the stock etc. They first log into their accounts and then can choose from the following set of options:

1. Add new medicines to stock
2. Delete a medicine from stock
3. Increase the quantity of medicine in stock
4. View all medicines and their details
5. View medicines sold by a particular drug manufacturer
6. Display the customer details of those who bought drugs from a particular company
7. Display records of all employees
8. View all orders
9. View orders between given dates
10. Get total sales of branch
11. Show details of delivery partners

RELATIONAL SCHEMA



ENTITY RELATIONSHIP SCHEMA



ENTITIES, ATTRIBUTES & SCHEMA

1) Company:

Field	Type	Null	Key	Default	Extra
companyID	int	NO	PRI	NULL	auto_increment
companyName	varchar(255)	NO	UNI	NULL	
monthlySales	Int	NO		NULL	

Constraints: `salesCheck` CHECK ((`monthlySales` >= 0))

2) Customer:

Field	Type	Null	Key	Default	Extra

customerID	int	NO	PRI	NULL	auto_increment
custName	varchar(255)	NO		NULL	
custPassword	varchar(255)	NO		NULL	
custAddress	varchar(255)	NO		NULL	
custPhoneNumber	bigint	NO	UNI	NULL	
custAge	int	YES		NULL	

Constraints: `ageCheck` CHECK ((`custAge` >= 13)),
`numberConstraint` CHECK ((`custPhoneNumber` > 0))

3) Customer:

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
manuCompanyID	int	NO	PRI	NULL	auto_increment
manuCompanyName	varchar(255)	NO		NULL	
manuLicenseNumber	bigint	NO		NULL	

Constraints: `ageCheck` CHECK ((`custAge` >= 13)),

4) Medicine:

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
drugID	int	NO	PRI	NULL	auto_increment
drugName	varchar(255)	NO	UNI	NULL	
drugPrice	int	NO		NULL	
drugManufacturingDate	date	NO		NULL	
drugExpiryDate	date	NO		NULL	
drugQuantity	int	YES		NULL	
drugCompanyID	Int	YES	MUL	NULL	

5) Delivery Partner:

Field	Type	Null	Key	Default	Extra
deliveryPartnerID	int	NO	PRI	NULL	auto_increment
deliveryPartnerName	varchar(255)	NO		NULL	

6) Drug Manufacturer:

Field	Type	Null	Key	Default	Extra
manuCompanyID	int	NO	PRI	NULL	auto_increment
manuCompanyName	varchar(255)	NO		NULL	
manuLicenceNumber	bigint	NO		NULL	

7) Order Details:

Field	Type	Null	Key	Default	Extra
orderID	int	NO	PRI	NULL	auto_increment
checkoutPrice	int	NO		NULL	
deliveryAddress	varchar(255)	NO		NULL	
orderDate	date	NO		NULL	
drugCount	int	NO		NULL	
orderDrugID	int	YES	MUL	NULL	

8) Product Information:

Field	Type	Null	Key	Default	Extra
cartCustomerID	int	NO	PRI	NULL	
cartDrugID	int	NO	PRI	NULL	
cartDrugQuantity	int	NO	PRI	NULL	

9) Cart:

Field	Type	Null	Key	Default	Extra
cartCustomerID	int	NO	PRI	NULL	
totalCost	int	YES		NULL	

Constraints: `ageCheck` CHECK ((`custAge` >= 13)),

10) Billing Details:

Field	Type	Null	Key	Default	Extra
medicineOrderID	int	NO	MUL	NULL	
totalPrice	int	NO		NULL	
modeOfPayment	varchar(255)	NO		NULL	
billingAddress	varchar(255)	NO		NULL	

Constraints: `ageCheck` CHECK ((`custAge` >= 13)),

11) Contains:

Field	Type	Null	Key	Default	Extra
medID	int	NO	PRI	NULL	
custID	int	NO	PRI	NULL	

12) Billing Details:

Field	Type	Null	Key	Default	Extra
discountID	int	NO	PRI	NULL	auto_increment
discountPercentage	int	NO		NULL	

Constraints: `ageCheck` CHECK ((`custAge` >= 13)),

SQL QUERIES

DIVERSE QUERIES

1. **Display customer names and their cart's total cost who are eligible for a discount (with total cart cost greater than 5000)**

```
SELECT customer.custName AS "Customers Eligible for Discount", cart.totalCost AS
"Cart Cost (Rs)"FROM customer
INNER JOIN cart ON cart.cartCustomerID=customer.customerID
WHERE cart.totalCost>5000
ORDER BY cart.totalCost DESC;
```

2. **Count number of orders with mode of payment as COD/UPI/Card**

```
SELECT modeOfPayment as "Mode of Payment", count(*) "Number Of Orders"
FROM billingdetails
GROUP BY modeOfPayment;
```

3. **Find all the drugs manufactured by a given Drug Manufacturer (example: for drug manufacturer name = "Oxygen")**

```
SELECT medicine.drugID, medicine.drugName,
drugManufacturer.manuCompanyName
FROM medicine
INNER JOIN drugManufacturer
ON drugManufacturer.manuCompanyID=medicine.drugCompanyID
WHERE drugManufacturer.manuCompanyName="Oxygen";
```

4. **Update the stock: Remove a particular batch of drugs if it has expired**

```
DELETE FROM medicine
WHERE drugExpiryDate<CURDATE();
```

5. **Rank the manufactured by each company on the basis of their Manufacturing Dates**

```
SELECT drugManufacturer.manuCompanyName, medicine.drugID,
medicine.drugName, dense_rank()
OVER
(PARTITION BY drugCompanyID ORDER BY drugManufacturingDate) AS "Rank of
Medicine"
FROM medicine INNER JOIN drugManufacturer
ON drugManufacturer.manuCompanyID=medicine.drugCompanyID order by
drugManufacturer.manuCompanyName;
```

6. **Display the average total cost of each cart**

```
SELECT AVG(totalCost)
FROM cart;
```

7. **Display the medicines which have less than 10 units in stock**

```
DROP VIEW if exists medicinesgoingoutofstock;
CREATE VIEW medicinesGoingOutOfStock
AS SELECT medicine.drugName
FROM medicine
WHERE drugQuantity<10;
```

8. **Increase the salaries of employees who are managers**

```
UPDATE employee
SET employeeSalary = employeeSalary * 1.2
WHERE managerID=1;
```

9. **Display names of customers who have a given drug in their cart**

```
SELECT customer.custName, productinfo.cartDrugID
FROM productinfo
INNER JOIN customer ON productinfo.cartCustomerID= customer.customerID
WHERE productinfo.cartDrugID=23
OR productinfo.cartDrugID=45;
```

10. **Display all customers who bought a medicine manufactured by a given drug manufacturer**

```
SELECT customer.customerID, customer.custName, medicine.drugID,
drugManufacturer.manuCompanyName
FROM (((customer
INNER JOIN productinfo ON customer.customerID=productInfo.cartCustomerID)
INNER JOIN medicine on productInfo.cartDrugID=medicine.drugID)
INNER join drugManufacturer ON
drugManufacturer.manuCompanyID=medicine.drugCompanyID)
WHERE drugManufacturer.manuCompanyName="Antiseptic";
```

11. **Display total sum of prices of all drugs sold by a drug manufacturer**

```
SELECT medicine.drugCompanyID, sum(medicine.drugPrice)
```

```
FROM (medicine INNER JOIN orderdetails ON
medicine.drugID=orderdetails.orderDrugID)
GROUP BY medicine.drugCompanyID;
```

12. **Display all orders with their drug names and drug manufacturer names**

```
SELECT DISTINCT orderdetails.orderDrugID, medicine.drugName,
drugManufacturer.manuCompanyName
FROM (drugmanufacturer INNER JOIN (orderdetails INNER JOIN medicine))
WHERE orderdetails.orderDrugID=medicine.drugID
AND medicine.drugCompanyID=drugManufacturer.manucompanyID;
```

RELATIONAL QUERIES

1. **UNION**

```
SELECT manuCompanyName from drugManufacturer
UNION ALL
SELECT drugCompanyID from medicine;
```

2. **PROJECTION**

```
SELECT drugID, drugName, drugManufacturingDate, drugExpiryDate
FROM medicine
WHERE drugName="Salicylic Acid";
```

3. **INTERSECTION**

```
SELECT drugID, drugName, drugPrice
FROM medicine left join orderdetails
ON medicine.drugID=orderdetails.orderDrugID
INTERSECT
SELECT drugID, drugName, drugPrice
FROM medicine right join orderdetails
ON medicine.drugID=orderdetails.orderDrugID;
```

4. **SET DIFFERENCE**

```
SELECT customerID
FROM customer cust
WHERE not exists
(SELECT  null
        FROM contains cont
        WHERE (cust.customerID=cont.custID OR cust.customerID is null and
cont.custID is null));
```

5. **CARTESIAN PRODUCT**

```
SELECT companyName, drugName
FROM company, medicine;
```

CONSTRAINTS

FOREIGN KEY CONSTRAINT

1. **CART**

```
SELECT  TABLE_NAME,COLUMN_NAME,CONSTRAINT_NAME,
REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME
FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
REFERENCED_TABLE_SCHEMA = 'medmate' AND
REFERENCED_TABLE_NAME = 'cart';
```

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
productinfo	cartCustomerID	productinfo_ibfk_1	cart	cartCustomerID

2. **MEDICINE**

```
SELECT  TABLE_NAME,COLUMN_NAME,CONSTRAINT_NAME,
REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME
FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
REFERENCED_TABLE_SCHEMA = 'medmate' AND
REFERENCED_TABLE_NAME = ‘medicine’;
```

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
contains	medID	contains_ibfk_1	medicine	drugID
orderdetails	orderDrugID	orderdetails_ibfk_1	medicine	drugID

3. **DRUG MANUFACTURER**

```
SELECT  TABLE_NAME,COLUMN_NAME,CONSTRAINT_NAME,
REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME
FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
REFERENCED_TABLE_SCHEMA = 'medmate' AND
REFERENCED_TABLE_NAME = ‘drugmanufacturer’;
```

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
medicine	drugCompanyID	medicine_ibfk_1	drugmanufacturer	manuCompanyID

4. **CUSTOMER**

```
SELECT  TABLE_NAME,COLUMN_NAME,CONSTRAINT_NAME,
REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME
FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
REFERENCED_TABLE_SCHEMA = 'medmate' AND
REFERENCED_TABLE_NAME = ‘customer’;
```

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
contains	custID	contains_ibfk_2	customer	customerID
cart	cartCustomerID	cart_ibfk_1	customer	customerID

5. **ORDER DETAILS**

```
SELECT  TABLE_NAME,COLUMN_NAME,CONSTRAINT_NAME,
REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME
FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
REFERENCED_TABLE_SCHEMA = 'medmate' AND
REFERENCED_TABLE_NAME =  ‘orderdetails’;
```

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
billingdetails	medicineOrderID	billingdetails_ibfk_1	orderdetails	orderID

OTHER CONSTRAINTS USING DESC COMMAND

1. **BILLING DETAILS**

Field	Type	Null	Key	Default	Extra
medicineOrderID	int	NO	MUL	NULL	
totalPrice	int	NO		NULL	
modeOfPayment	varchar(255)	NO		NULL	
billingAddress	varchar(255)	NO		NULL	

2. **CART**

Field	Type	Null	Key	Default	Extra
cartCustomerID	int	NO	PRI	NULL	
totalCost	int	YES		NULL	

3. **COMPANY**

Field	Type	Null	Key	Default	Extra
companyID	int	NO	PRI	NULL	auto_increment
companyName	varchar(255)	NO	UNI	NULL	
monthlySales	int	NO		NULL	

4. **CONTAINS**

Field	Type	Null	Key	Default	Extra
medID	int	NO	PRI	NULL	
custID	int	NO	PRI	NULL	

5. **CUSTOMER**

Field	Type	Null	Key	Default	Extra
customerID	int	NO	PRI	NULL	auto_increment
custName	varchar(255)	NO		NULL	
custPassword	varchar(255)	NO		NULL	
custAddress	varchar(255)	NO		NULL	
custPhoneNumber	bigint	NO	UNI	NULL	
custAge	int	YES		NULL	

6. **DELIVERY PARTNER**

Field	Type	Null	Key	Default	Extra
deliveryPartnerID	int	NO	PRI	NULL	auto_increment
deliveryPartnerName	varchar(255)	NO		NULL	

7. **DISCOUNT**

Field	Type	Null	Key	Default	Extra
discountID	int	NO	PRI	NULL	auto_increment
discountPercentage	int	NO		NULL	
isApplicable	int	NO		NULL	
discountExpiryDate	date	NO		NULL	

8. **DRUG MANUFACTURER**

Field	Type	Null	Key	Default	Extra
manuCompanyID	int	NO	PRI	NULL	auto_increment
manuCompanyName	varchar(255)	NO		NULL	
manuLicenseNumber	bigint	NO		NULL	

9. EMPLOYEE

Field	Type	Null	Key	Default	Extra
employeeID	int	NO	PRI	NULL	auto_increment
employeeUsername	varchar(255)	NO		NULL	
employeePassword	varchar(255)	NO		NULL	
employeePhoneNumber	bigint	NO	UNI	NULL	
employeeSalary	int	NO		NULL	
employeeAddress	varchar(255)	NO		NULL	
managerID	int	NO		NULL	

10. MEDICINE

Field	Type	Null	Key	Default	Extra
drugID	int	NO	PRI	NULL	auto_increment
drugName	varchar(750)	NO	UNI	NULL	
drugPrice	int	NO		NULL	
drugManufacturingDate	date	NO		NULL	
drugExpiryDate	date	NO		NULL	
drugQuantity	int	YES		NULL	
drugCompanyID	int	YES	MUL	NULL	

11. ORDER DETAILS

Field	Type	Null	Key	Default	Extra
orderID	int	NO	PRI	NULL	auto_increment
checkoutPrice	int	NO		NULL	
deliveryAddress	varchar(255)	NO		NULL	
orderDate	date	NO		NULL	
drugCount	int	NO		NULL	
orderDrugID	int	YES	MUL	NULL	

12. **PRODUCT INFO**

Field	Type	Null	Key	Default	Extra
cartCustomerID	int	NO	PRI	NULL	
cartDrugID	int	NO	PRI	NULL	
cartDrugQuantity	int	NO	PRI	NULL	

CONSTRAINTS FOR EACH TABLE

1. **medicine**
   SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
   FROM information_schema.table_constraints
   WHERE table_name='billingdetails';

2. **cart**
   SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
   FROM information_schema.table_constraints
   WHERE table_name='cart';

3. **company**
   SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
   FROM information_schema.table_constraints
   WHERE table_name='company';

4. **contains**
   SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
   FROM information_schema.table_constraints
   WHERE table_name='contains';

5. **customer**
   SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
   FROM information_schema.table_constraints

```
WHERE table_name='customer';
```

6. **deliverypartner**
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='deliverypartner';
```

7. **discount**
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='discount';
```

8. **drugmanufacturer**
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='drugmanufacturer';
```

9. **employee**
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='employee';
```

10. **medicine**
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='medicine';
```

11. **orderdetails**
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='orderdetails';
```

12. **productinfo**
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
FROM information_schema.table_constraints
WHERE table_name='productinfo';
```

OLAP QUERIES

1.
```
SELECT drugExpiryDate, drugName, drugID, max(drugQuantity)
FROM medicine
GROUP BY drugExpiryDate, drugName, drugID WITH ROLLUP
ORDER BY GROUPING(drugExpiryDate) DESC;
```
2.
```
SELECT drugName, drugCompanyID, drugID, max(drugQuantity)
FROM medicine
GROUP BY drugName, drugCompanyID, drugID WITH ROLLUP;
```

3. SELECT orderID, orderDrugID, drugCount, orderDate, max(checkoutPrice)
   FROM orderdetails
   GROUP BY orderID, orderDrugID, drugCount, orderDate WITH ROLLUP;
4. SELECT drugCompanyID, drugID, drugName, max(drugPrice)
   FROM medicine GROUP BY drugCompanyID, drugID, drugName WITH ROLLUP;

TRIGGER QUERIES

1) **This trigger shows a message to the pharmacy when the quantity of a drug goes below 10 so that the pharmacy can order more drugs and re-stock.**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `medicine_BEFORE_UPDATE` BEFORE
UPDATE ON `medicine` FOR EACH ROW BEGIN
        if NEW.drugQuantity<10 then
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Drug Quantity low';
        end if;
END
```

2) **This trigger adds data to the billingdetails table as soon as and order is placed. When an order is placed, a row is inserted with the required data in the orderdetails table and then automatically, using this trigger, the same row is entered into the billingdetails table with the mode of payment as an empty string ‘ ’.**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `orderdetails_AFTER_INSERT` AFTER
INSERT ON `orderdetails` FOR EACH ROW
BEGIN
        INSERT INTO billingdetails (medicineOrderID, totalPrice, modeOfPayment,
        billingAddress)
        VALUES (NEW.orderID, NEW.checkoutPrice, '', NEW.deliveryAddress);
END
```

3) **This trigger creates an empty cart for a user when the user makes a new account, ie, signs up. Hence, once a record is inserted in the 'customer' table, a new row would also be created in the 'cart' table with the same customer ID with the total cost of the cart as 0.**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `customer_AFTER_INSERT` AFTER
INSERT ON `customer` FOR EACH ROW BEGIN
        INSERT INTO cart (cartCustomerID, totalCost) VALUES (NEW.customerID,0);
END
```

EMBEDDED SQL QUERIES and OLAP QUERIES

```
import mysql.connector
mydb=mysql.connector.connect(host = "localhost", user = "root",
passwd = "12345")

mycur=mydb.cursor()
mycur.execute("USE medmate")

#Helper function to find maximum length of column needed for display
def find_max_col_length(recs, col_index, col_name):
    max_col_length = 0
    for record in recs:
        curr_length = len(str(record[col_index]))
        if curr_length > max_col_length:
            max_col_length = curr_length
    return max(max_col_length, len(col_name))
```

```
#Helper function to display the given table 'recs'
def display_table(recs, table_desciption):
    widths = []
    columns = []
    boundary = '|'
    separator = '+'
    index = 0

    for cd in table_desciption:
        widths.append(find_max_col_length(recs, index, cd[0]))
        columns.append(cd[0])
        index+=1

    for w in widths:
        boundary += " %-"+"%ss |" % (w,)
        separator += '-'*w + '--+'

    print(separator)
    print(boundary % tuple(columns))
    print(separator)
    for row in recs:
        print(boundary % row)
    print(separator)

def embedded_query1(companyID):
    st='''SELECT medicine.drugID, medicine.drugName,
drugManufacturer.manuCompanyName
```

```
    FROM medicine
    INNER JOIN drugManufacturer
    ON drugManufacturer.manuCompanyID=medicine.drugCompanyID
    WHERE drugManufacturer.manuCompanyID='''+companyID
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying all drugs sold by drug manufacturer with
ID="+companyID +": ")
        display_table(recs, mycur.description)


def embedded_query2(companyID):
    st='''SELECT customer.customerID, customer.custName,
medicine.drugID, drugManufacturer.manuCompanyName
    FROM (((customer
    INNER JOIN productinfo ON
customer.customerID=productInfo.cartCustomerID)
    INNER JOIN medicine on productInfo.cartDrugID=medicine.drugID)
    INNER join drugManufacturer ON
    drugManufacturer.manuCompanyID=medicine.drugCompanyID)
    WHERE drugManufacturer.manuCompanyID='''+companyID
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
```

```
    else:
        print("Displaying results of query 2: ")
        display_table(recs, mycur.description)

def show_medicines():
    st="Select * from medicine"
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying records of all medicines: ")
        display_table(recs, mycur.description)

def manager_query1():
    st="SELECT * from employee"
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying records of employees: ")
        display_table(recs, mycur.description)


def check_manager(id):
    st="SELECT managerID from employee where employeeID="+id
```

```
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
        return False
    for i in recs:
        if i[0]==1:
            return True
        else:
            print("You do not have access")
            return False
    return False

def olap1():
    st='''SELECT drugExpiryDate, drugName, drugID, max(drugQuantity)
    FROM medicine
    GROUP BY drugExpiryDate, drugName, drugID WITH ROLLUP
    ORDER BY GROUPING(drugExpiryDate) DESC'''
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying results of olap query 1: ")
        display_table(recs, mycur.description)

def olap2():
```

```
    st='''SELECT drugName, drugCompanyID, drugID, max(drugQuantity)
    FROM medicine
    GROUP BY drugName, drugCompanyID, drugID WITH ROLLUP'''
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying results of olap query 2: ")
        display_table(recs, mycur.description)


def olap3():
    st='''SELECT orderID, orderDrugID, drugCount, orderDate,
max(checkoutPrice)
    FROM orderdetails GROUP BY orderID, orderDrugID, drugCount,
orderDate WITH ROLLUP'''
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying results of olap query 3: ")
        display_table(recs, mycur.description)


def olap4():
    st='''SELECT drugCompanyID, drugID, drugName, max(drugPrice)
```

```
    FROM medicine GROUP BY drugCompanyID, drugID, drugName WITH
ROLLUP '''
    mycur.execute(st)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying results of olap query 4: ")
        display_table(recs, mycur.description)



def updation_trigger():
    did=input("Enter drug ID of drug you want to update quantity of:
")
    qty=input("Enter new quantity of drug: ")
    try:
        st="UPDATE medicine set drugQuantity= '"+qty+"' where drugID=
'"+did+"' "
        mycur.execute(st)
        print("Quantity has been updated")
    except:
        print("Drug Quantity low, can't update")
    mydb.commit()



def insertion_trigger():
    oid=input("Enter orderID: ")
```

```
    cprice=input("Enter checkout price: ")
    addr=input("Enter Delivery address: ")
    date=input("Enter order date: ")
    cnt=input("Enter drug count: ")
    did=input("Enter order drug ID: ")

    print()
    st="INSERT INTO orderdetails values('"+oid+"', '"+cprice+"',
'"+addr+"', '"+date+"', '"+cnt+"', '"+did+"')"
    mycur.execute(st)
    mydb.commit()

    st1="SELECT * FROM billingdetails"
    mycur.execute(st1)
    recs=mycur.fetchall()
    if len(recs)==0:
        print("No records found")
    else:
        print("Displaying records of billing details: ")
        display_table(recs, mycur.description)

ans1='y'
ans2='y'
choice1=int(input("Welcome to MedMate! Please choose:\n 1.Login as
customer\n 2.Login as manager\n"))
if choice1==1:
    while ans1=='y':
```

```
        #if choice1==1:
        choice2=int(input('''Please select the query you want to
run:\n
        1. Display all drugs sold by a particular drug manufacturer\n
        2. Display records of all medicines\n
        '''))
        if choice2==1:
            companyID=str(input("Enter the drug manufacturer ID: "))
            embedded_query1(companyID)
        elif choice2==2:
            show_medicines()
        else:
            print("Wrong Choice\n")
        ans1=str(input("Do you want to continue? y/n"))



elif choice1==2:
    id=str(input("Enter your employeeID: "))
    if check_manager(id):
        while ans2=='y':
            choice2=int(input('''Please select the query you want to
run:\n
            1. Display all drugs sold by a particular drug
manufacturer\n
            2. Display the customer details of those who bought drugs
from a particular company\n
            3.Display records of all emplpyees\n
            4. Run Olap query 1
```

```
            5. Run Olap query 2
            6. Run Olap query 3
            7. Run Olap query 4
            8. This trigger adds data to the billingdetails table as
soon as and order is placed.
            9. This trigger shows a message to the pharmacy when the
quantity of a drug goes below 10 so that the pharmacy can order more
drugs and re-stock\n
            '''))
            if choice2==1:
                companyID=str(input("Enter the drug manufacturer ID:
"))
                embedded_query1(companyID)
            elif choice2==2:
                companyID=str(input("Enter the drug manufacturer ID:
"))
                embedded_query2(companyID)
            elif choice2==3:
                manager_query1()
            elif choice2==4:
                olap1()
            elif choice2==5:
                olap2()
            elif choice2==6:
                olap3()
            elif choice2==7:
                olap4()
            elif choice2==8:
```

```
                insertion_trigger()
            elif choice2==9:
                updation_trigger()
            else:
                print("Wrong choice\n")
            ans2=str(input("Do you want to continue? y/n"))

print("Bye Bye")
```

- fin -