EASYPHARM: PHARMACY AND SUBSCRIPTION MANAGEMENT SYSTEM

USE CASE FINAL REPORT

Group 2

Student 1: Devansh Doshi
Student 2: Muskan Jain

857-269-0018 (Devansh)

857-264-6541 (Muskan)

doshi.dev@northeastern.edu
jain.musk@northeastern.edu

Percentage of Effort Contributed by Student 1: 50 %
Percentage of Effort Contributed by Student 2: 50 %

Signature of Student 1: Devansh Doshi
Signature of Student 2: Muskan Jain

Submission Date:
12/09/23

I. INTRODUCTION

EasyPharm Corporation is a large pharmacy chain operating across multiple locations. The company manages a diverse range of pharmaceutical products, serves a wide customer base, and partners with various suppliers and manufacturers.

Problem Statement

EasyPharm Corporation is facing challenges in optimizing its operations, improving customer service, and enhancing inventory management. The company experiences difficulties in tracking product availability, managing customer subscriptions, and ensuring efficient supply chain management. Inventory management is one of the biggest challenges facing pharmacists today. Pharmacies must carry a wide range of medications, including both prescription and over-the-counter medications. They must also track the expiration dates of all their medications and ensure that they have enough stock on hand to meet customer demand. A Pharmacy Management System can help pharmacies to overcome the challenge of inventory management by providing real-time visibility into inventory levels, tracking expiration dates, and generating purchase orders automatically. This can help pharmacies to avoid stockouts, overstocking, and waste. Pharmacy management issues include expired subscriptions, billing and payment issues, license allocation and management, access restrictions, renewal reminders, and technical support.

Pharmacies can take several steps to fix subscription issues. One important step is to implement automated renewal reminders. This will help to ensure that pharmacies do not lapse on their subscriptions. Reminders can be sent via email, text message, or other channels. Another important step is to develop a robust system for tracking and managing subscription licenses.

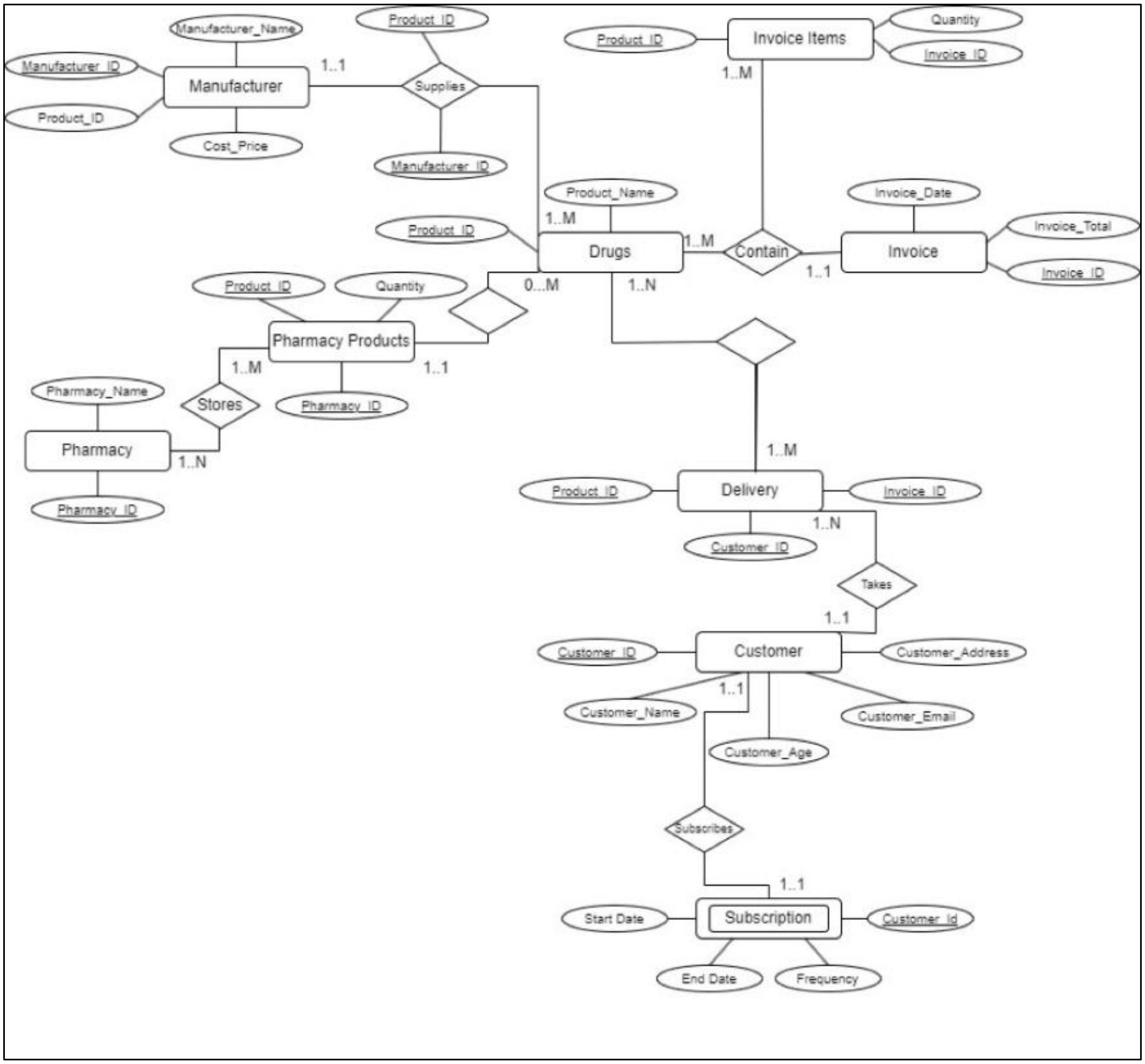
Goal of EasyPharm

EasyPharm is a comprehensive pharmacy and subscription management system designed to enhance efficiency. It streamlines various processes, such as prescription tracking, inventory management, and refill reminders. This allows pharmacies to optimize their workflow and serve patients more effectively and streamline prescription management, making it easier for patients to receive their medications on time and for healthcare providers to monitor and track medication adherence. By implementing EasyPharm, pharmacies can enhance patient care and improve medication management processes. By implementing such a system, pharmacies can experience a wide range of benefits. Firstly, it improves accuracy by reducing errors in prescription processing and dispensing. This not only ensures patient safety but also helps pharmacies maintain compliance with regulatory standards. EasyPharm also deals with affordable subscription prices. We understand the importance of making medications accessible and affordable for patients. EasyPharm offers a range of subscription plans at affordable prices. Patients can choose from monthly or quarterly subscription options, which provide them with a convenient and cost-effective way to receive their medications. By subscribing to EasyPharm, patients can enjoy the convenience of having their medications delivered right to their doorstep while also saving money on their prescriptions. Our goal at EasyPharm is to promote medication adherence and affordability, ensuring that patients have easy access to the medications they need without breaking the bank.

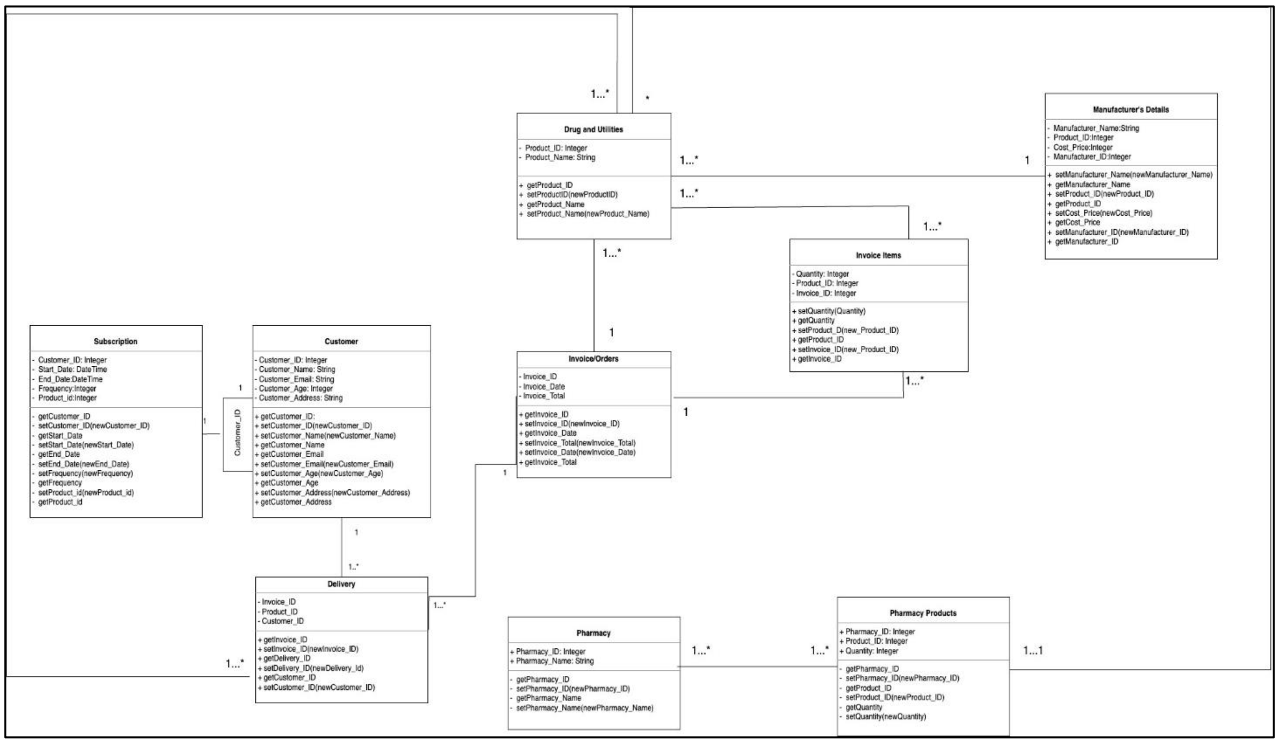
In conclusion, the integration of EasyPharm's subscription management system in pharmacies can improve prescription management processes, benefiting both patients and healthcare providers. Additionally, our affordable subscription prices offer patients a convenient and cost-effective way to receive their medications through EasyPharm.

II. CONCEPTUAL MODEL

EER MODEL



UML MODEL



III. RELATIONAL MODEL

1. Pharmacy (Pharmacy_ID, Pharmacy_Name, Pharmacy_Address)
• Primary Key: Pharmacy_ID

2. Customer(Customer_ID, Customer_Name, Customer_Age, Customer_Email,
Customer_Address)
• Primary Key: Customer_ID

3. Invoice/Orders(Invoice_Date, Invoice_Total, Invoice_ID)
• Primary Key: Invoice_ID

4. Invoice_Items(Invoice_ID, Product_ID, Quantity)
• Primary Key: Invoice_ID, Product_ID
• Foreign Key: Product_ID references Product_ID in Drugs/Ulcers
• Foreign Key: Invoice_ID references Invoice_ID in Invoice/Orders

5. Delivery(Customer_ID, Invoice_ID, Product_ID)
• Primary Key: Customer_ID, Invoice_ID, Product_ID
• Foreign Key: Invoice_ID references Invoice_ID in Invoice/Orders
• Foreign Key: Product_ID references Product_ID in Drugs/Ulcers
• Foreign Key: Customer_ID references Customer_ID in Customer

6. Subscription(Customer_ID, Frequency, Start_Date, End_Date)
• Primary Key: Customer_ID
• Foreign Key: Customer_ID references Customer_ID in Customer

7. Drugs_Ulcers (Product_ID, Product_Name)
• Primary Key: Product_ID

8. Ordered(Product_ID, Customer_ID, Invoice_ID)
• Primary Key: Product_ID, Customer_ID, Invoice_ID
• Foreign Key: Product_ID references Product_ID in Drugs/Ulcers
• Foreign Key: Invoice_ID references Invoice_ID in Invoice/Orders
• Foreign Key: Customer_ID references Customer_ID in Customer

9. Manufacturer(Manufacturer_Name, Product_ID, Cost_Price, Manufacturer_ID)
• Primary Key: Manufacturer_ID
• Foreign Key: Product_ID references Product_ID in Drugs/Ulcers

10. Supplies(Product_ID, Manufacturer_ID)
• Primary Key: Product_ID, Manufacturer_ID
• Foreign Key: Product_ID references Product_ID in Drugs/Ulcers
• Foreign Key: Manufacturer_ID references Manufacturer_ID in Manufacturer

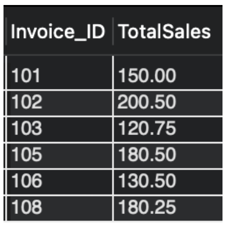
11. Has(Product_ID, Invoice_ID, Customer_ID)
• Primary Key: Product_ID, Invoice_ID, Customer_ID
• Foreign Key: Product_ID references Product_ID in Drugs/Ulcers
• Foreign Key: Invoice_ID references Invoice_ID in Invoice/Orders
• Foreign Key: Customer_ID references Customer_ID in Customer

12. Pharmacy_Products(Product_ID, Pharmacy_ID, Stock_Level)
• Primary Key: Product_ID, Pharmacy_ID
• Foreign Key: Product_ID references Product_ID in Drugs/Ulcers
• Foreign Key: Pharmacy_ID references Pharmacy_ID in Pharmacy

IV. IMPLEMENTATION OF RELATIONAL MODEL VIA MYSQL AND NOSQL

Query 1: Aggregate total sales amount for each invoice having total sales more than 100

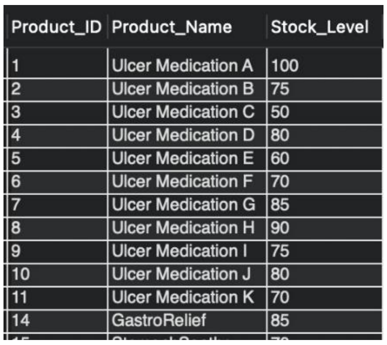
```
SELECT Invoice_ID,
SUM(Invoice_Total) AS TotalSales
FROM Invoice_Orders
```



Invoice_ID	TotalSales
101	150.00
102	200.50
103	120.75
105	180.50
106	130.50
108	180.25

Query 2: Write a JOIN query for Product and Stock Level

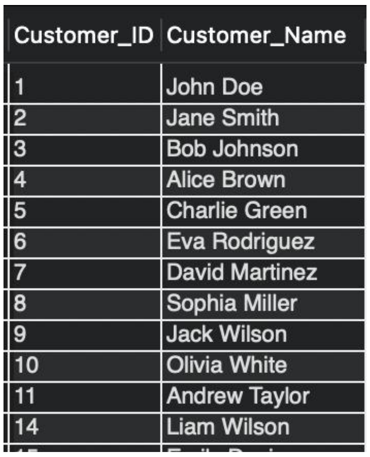
```
SELECT
   p.Product_ID,
   p.Product_Name,
   pp.Stock_Level
FROM
   Drugs_Ulcers p
JOIN
   Pharmacy_Products pp ON p.Product_ID = pp.Product_ID;
```



Product_ID	Product_Name	Stock_Level
1	Ulcer Medication A	100
2	Ulcer Medication B	75
3	Ulcer Medication C	50
4	Ulcer Medication D	80
5	Ulcer Medication E	60
6	Ulcer Medication F	70
7	Ulcer Medication G	85
8	Ulcer Medication H	90
9	Ulcer Medication I	75
10	Ulcer Medication J	80
11	Ulcer Medication K	70
14	GastroRelief	85

Query 3: Find customers who have subscriptions and made purchases in the last month.

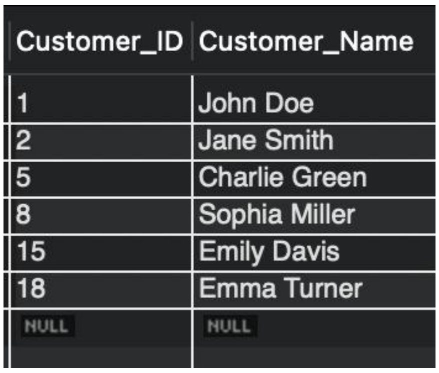
```
SELECT Customer_ID, Customer_Name
FROM Customer
WHERE Customer_ID IN (
   SELECT DISTINCT s.Customer_ID
   FROM Subscription s
   WHERE s.Start_Date <= DATE_ADD(NOW(), INTERVAL -1 MONTH)
   UNION
   SELECT DISTINCT o.Customer_ID
   FROM Ordered o
   INNER JOIN Invoice_Orders i ON o.Invoice_ID = i.Invoice_ID
   WHERE i.Invoice_Date >= DATE_ADD(NOW(), INTERVAL -1 MONTH)
);
```



Customer_ID	Customer_Name
1	John Doe
2	Jane Smith
3	Bob Johnson
4	Alice Brown
5	Charlie Green
6	Eva Rodriguez
7	David Martinez
8	Sophia Miller
9	Jack Wilson
10	Olivia White
11	Andrew Taylor
14	Liam Wilson

Query 4:Find the customers who have opted for delivery made orders with a total amount greater than the average total amount of all orders.

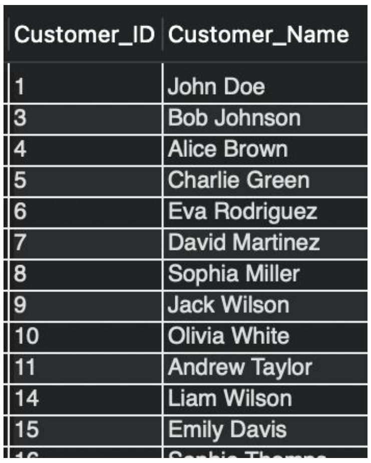
```
SELECT DISTINCT c.Customer_ID, c.Customer_Name
FROM Customer c
WHERE EXISTS (
   SELECT 1
   FROM Delivery d
   JOIN Invoice_Orders io ON d.Invoice_ID = io.Invoice_ID
   WHERE c.Customer_ID = d.Customer_ID
    AND io.Invoice_Total > (
       SELECT AVG(Invoice_Total)
       FROM Invoice_Orders io_avg
    )
);
```



Customer_ID	Customer_Name
1	John Doe
2	Jane Smith
5	Charlie Green
8	Sophia Miller
15	Emily Davis
18	Emma Turner
NULL	NULL

Query 5: Find customers who have made purchases greater than any order made by Customer_ID=2

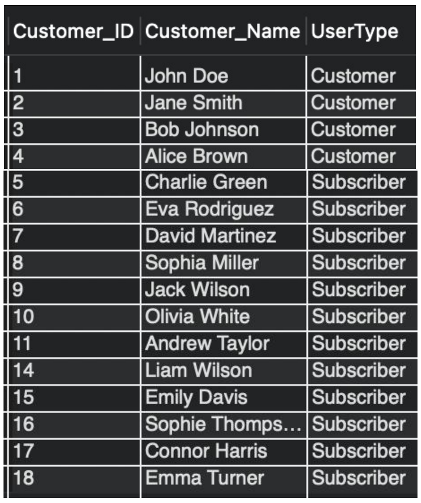
```
SELECT DISTINCT c.Customer_ID, c.Customer_Name
FROM Customer c
WHERE EXISTS (
   SELECT 1
   FROM Has h
   JOIN Invoice_Orders io ON h.Invoice_ID = io.Invoice_ID
   WHERE c.Customer_ID = h.Customer_ID
    AND io.Invoice_Total > ALL (
        SELECT Invoice_Total
        FROM Invoice_Orders
        WHERE Customer_ID = 2
     )
);
```



Customer_ID	Customer_Name
1	John Doe
3	Bob Johnson
4	Alice Brown
5	Charlie Green
6	Eva Rodriguez
7	David Martinez
8	Sophia Miller
9	Jack Wilson
10	Olivia White
11	Andrew Taylor
14	Liam Wilson
15	Emily Davis

Query 6: SQL query to retrieve a list of users, including their ID, name, and user type, from the pharmacy database. Users can be either regular customers or subscribers. For customers, the user type should be 'Customer', and for subscribers, the user type should be 'Subscriber'.

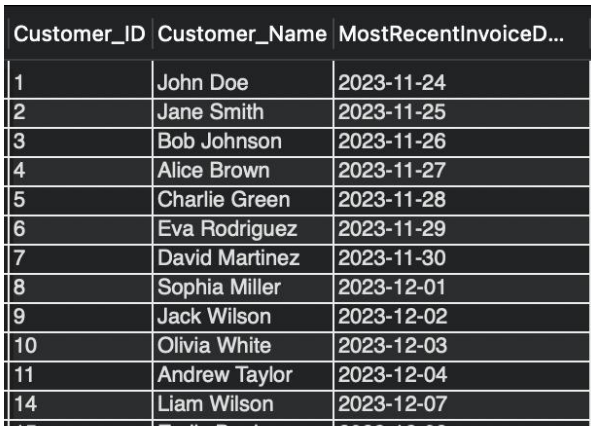
```
SELECT Customer_ID, Customer_Name, 'Customer' AS UserType
FROM Customer
UNION
SELECT s.Customer_ID, c.Customer_Name, 'Subscriber' AS UserType
FROM Subscription s
JOIN Customer c ON s.Customer_ID = c.Customer_ID;
```



Customer_ID	Customer_Name	UserType
1	John Doe	Customer
2	Jane Smith	Customer
3	Bob Johnson	Customer
4	Alice Brown	Customer
5	Charlie Green	Subscriber
6	Eva Rodriguez	Subscriber
7	David Martinez	Subscriber
8	Sophia Miller	Subscriber
9	Jack Wilson	Subscriber
10	Olivia White	Subscriber
11	Andrew Taylor	Subscriber
14	Liam Wilson	Subscriber
15	Emily Davis	Subscriber
16	Sophie Thomps…	Subscriber
17	Connor Harris	Subscriber
18	Emma Turner	Subscriber

Query 7: Subquery in SELECT and FROM clauses using the Has table.

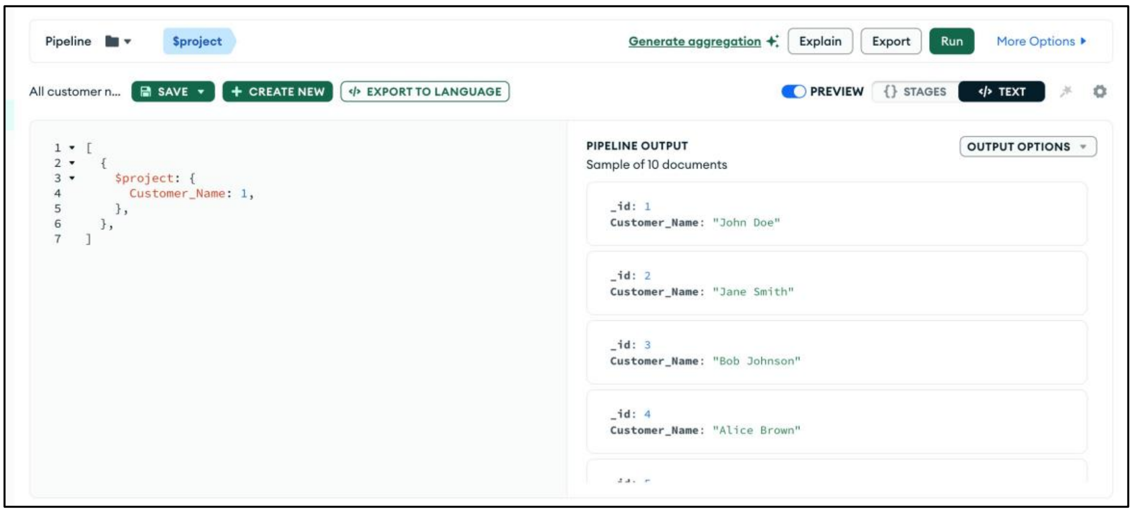
```
SELECT
   c.Customer_ID,
   c.Customer_Name,
   (
      SELECT MAX(io.Invoice_Date)
      FROM Invoice_Orders io
      JOIN Has h ON io.Invoice_ID = h.Invoice_ID
      WHERE h.Customer_ID = c.Customer_ID
   ) AS MostRecentInvoiceDate
FROM
   Customer c;
```



Customer_ID	Customer_Name	MostRecentInvoiceD...
1	John Doe	2023-11-24
2	Jane Smith	2023-11-25
3	Bob Johnson	2023-11-26
4	Alice Brown	2023-11-27
5	Charlie Green	2023-11-28
6	Eva Rodriguez	2023-11-29
7	David Martinez	2023-11-30
8	Sophia Miller	2023-12-01
9	Jack Wilson	2023-12-02
10	Olivia White	2023-12-03
11	Andrew Taylor	2023-12-04
14	Liam Wilson	2023-12-07

NOSQL

1. **To Find all the customer names**

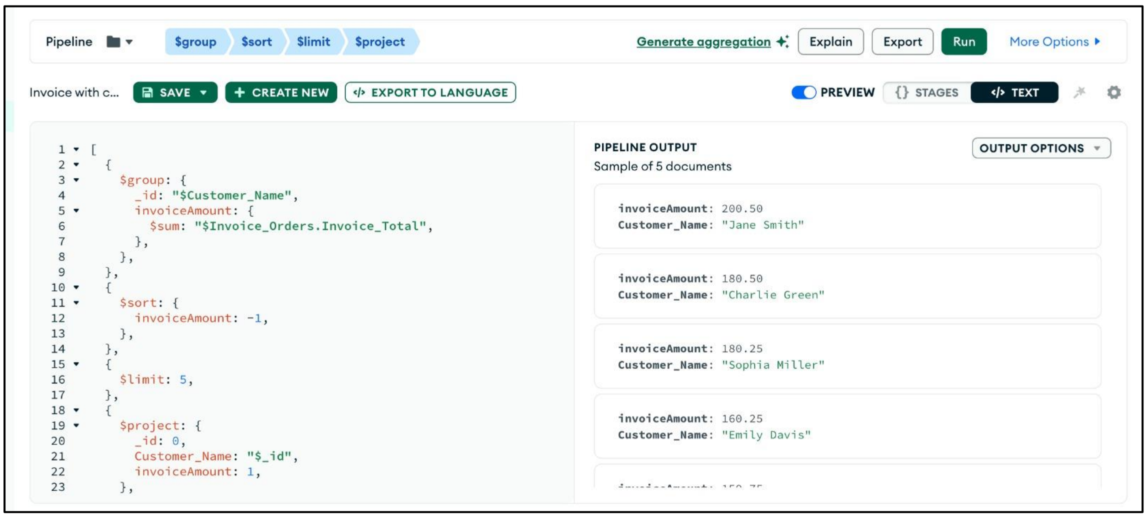


```
[
  {
    $project: {
      Customer_Name: 1,
    },
  },
]
```

Sample of 10 documents

```
_id: 1
Customer_Name: "John Doe"

_id: 2
Customer_Name: "Jane Smith"

_id: 3
Customer_Name: "Bob Johnson"

_id: 4
Customer_Name: "Alice Brown"
```

2. **To Find top 5 Customer with Invoice Amount**

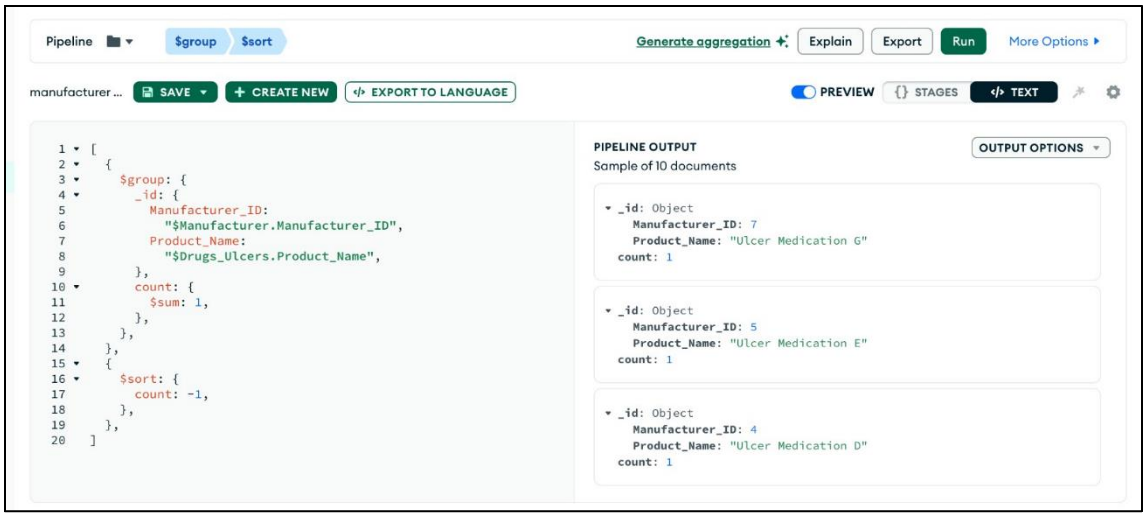


```
[
  {
    $group: {
      _id: "$Customer_Name",
      invoiceAmount: {
        $sum: "$Invoice_Orders.Invoice_Total",
      },
    },
  },
  {
    $sort: {
      invoiceAmount: -1,
    },
  },
  {
    $limit: 5,
  },
  {
    $project: {
      _id: 0,
      Customer_Name: "$_id",
      invoiceAmount: 1,
    },
```

Sample of 5 documents

```
invoiceAmount: 200.50
Customer_Name: "Jane Smith"

invoiceAmount: 180.50
Customer_Name: "Charlie Green"

invoiceAmount: 180.25
Customer_Name: "Sophia Miller"

invoiceAmount: 160.25
Customer_Name: "Emily Davis"
```

3. **To Find Manufacturer with Product they Supply**



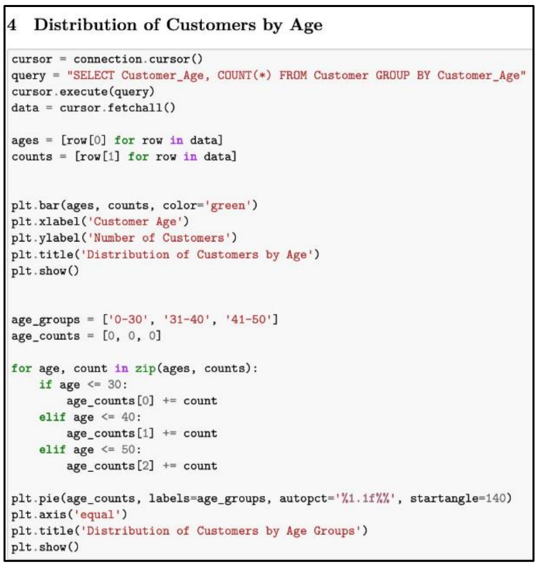
```
[
  {
    $group: {
      _id: {
        Manufacturer_ID:
          "$Manufacturer.Manufacturer_ID",
        Product_Name:
          "$Drugs_Ulcers.Product_Name",
      },
      count: {
        $sum: 1,
      },
    },
  },
  {
    $sort: {
      count: -1,
    },
  },
]
```

Sample of 10 documents

```
_id: Object
  Manufacturer_ID: 7
  Product_Name: "Ulcer Medication G"
count: 1

_id: Object
  Manufacturer_ID: 5
  Product_Name: "Ulcer Medication E"
count: 1

_id: Object
  Manufacturer_ID: 4
  Product_Name: "Ulcer Medication D"
count: 1
```

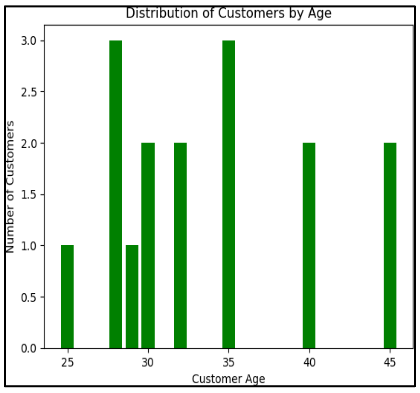
V. Database Access via R or Python

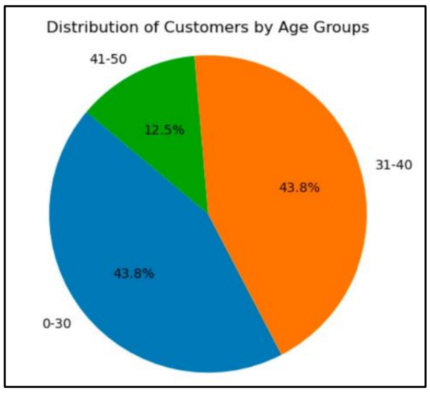
The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.excecute to run and fetchall from query.Used matplotlib library to plot the visualizations as well.And after the completion, closed the connection.

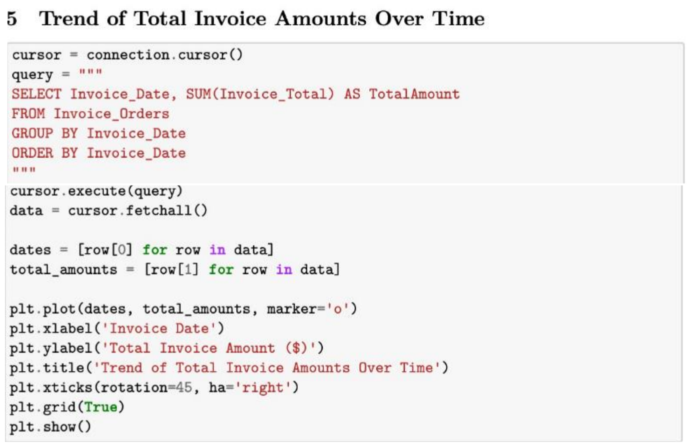


4 Distribution of Customers by Age

```
cursor = connection.cursor()
query = "SELECT Customer_Age, COUNT(*) FROM Customer GROUP BY Customer_Age"
cursor.execute(query)
data = cursor.fetchall()

ages = [row[0] for row in data]
counts = [row[1] for row in data]


plt.bar(ages, counts, color='green')
plt.xlabel('Customer Age')
plt.ylabel('Number of Customers')
plt.title('Distribution of Customers by Age')
plt.show()


age_groups = ['0-30', '31-40', '41-50']
age_counts = [0, 0, 0]

for age, count in zip(ages, counts):
    if age <= 30:
        age_counts[0] += count
    elif age <= 40:
        age_counts[1] += count
    elif age <= 50:
        age_counts[2] += count

plt.pie(age_counts, labels=age_groups, autopct='%1.1f%%', startangle=140)
plt.axis('equal')
plt.title('Distribution of Customers by Age Groups')
plt.show()
```

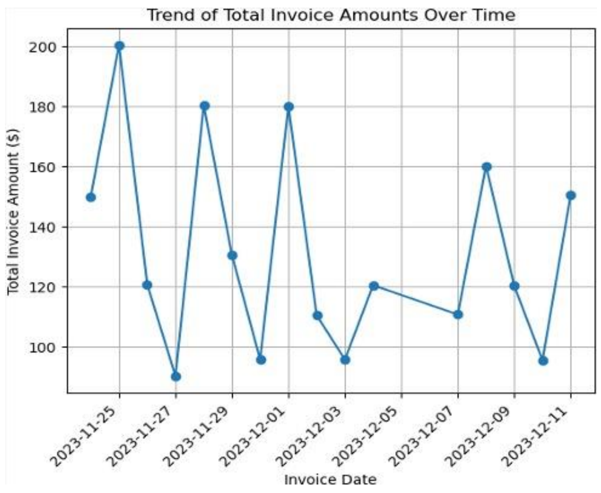


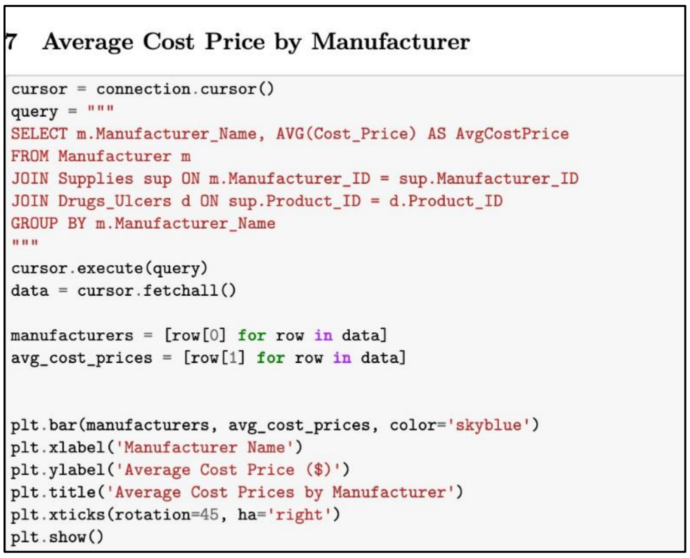




5 Trend of Total Invoice Amounts Over Time

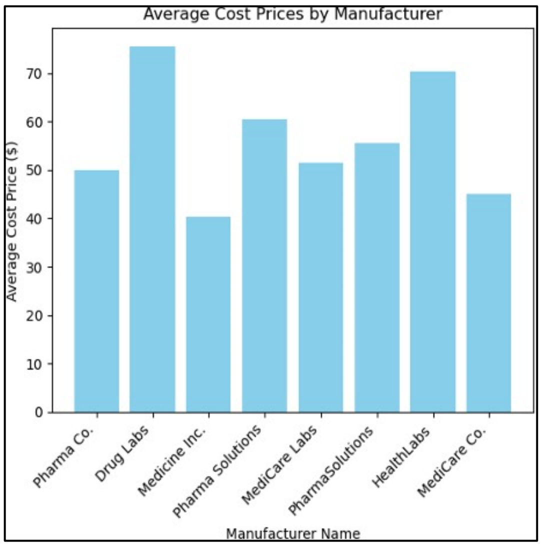
```
cursor = connection.cursor()
query = """
SELECT Invoice_Date, SUM(Invoice_Total) AS TotalAmount
FROM Invoice_Orders
GROUP BY Invoice_Date
ORDER BY Invoice_Date
"""
cursor.execute(query)
data = cursor.fetchall()

dates = [row[0] for row in data]
total_amounts = [row[1] for row in data]

plt.plot(dates, total_amounts, marker='o')
plt.xlabel('Invoice Date')
plt.ylabel('Total Invoice Amount ($)')
plt.title('Trend of Total Invoice Amounts Over Time')
plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.show()
```





7 Average Cost Price by Manufacturer

```
cursor = connection.cursor()
query = """
SELECT m.Manufacturer_Name, AVG(Cost_Price) AS AvgCostPrice
FROM Manufacturer m
JOIN Supplies sup ON m.Manufacturer_ID = sup.Manufacturer_ID
JOIN Drugs_Ulcers d ON sup.Product_ID = d.Product_ID
GROUP BY m.Manufacturer_Name
"""
cursor.execute(query)
data = cursor.fetchall()

manufacturers = [row[0] for row in data]
avg_cost_prices = [row[1] for row in data]


plt.bar(manufacturers, avg_cost_prices, color='skyblue')
plt.xlabel('Manufacturer Name')
plt.ylabel('Average Cost Price ($)')
plt.title('Average Cost Prices by Manufacturer')
plt.xticks(rotation=45, ha='right')
plt.show()
```



VI. Summary and Recommendation

Summary

EasyPharm, our user-friendly Pharmacy Management System (PMS), is at the heart of making pharmacy tasks smooth and accurate. It brings together essential features like managing stock, prescriptions, patient information, sales, and analytics.

EasyPharm ensures that pharmacies have just the right amount of medicines by keeping track of inventory and alerting when it's time to reorder. It simplifies handling prescriptions, making it easier for pharmacists to fill and keep track of them. Patient profiles are stored in a way that's simple to find and helps provide personalized service.

This system also makes transactions easy, creating clear invoices and receipts. It provides insights through reports, helping pharmacy owners make informed decisions. Importantly, it keeps everything secure and compliant with healthcare rules.

Recommendation

In our ongoing efforts to enhance EasyPharm, we are excited to share some significant developments aimed at simplifying processes and introducing innovative features to elevate the pharmacy experience.

Electronic Prescriptions:

One major stride involves our commitment to revolutionize the prescription process through electronic prescriptions. Instead of traditional paper prescriptions, doctors can now send them directly into the EasyPharm system using computers. This not only reduces the chances of errors often associated with handwritten prescriptions but also speeds up the process of getting the necessary medications.

Mobile App Development:

Embracing the widespread use of mobile technology, we are actively working on developing a dedicated EasyPharm app for both patients and pharmacists. This user-friendly app will offer a convenient interface, providing easy access to prescription information, medication reminders, and other essential features. Importantly, the app's functionality will be intricately linked to the central EasyPharm database, ensuring real-time synchronization of critical information.

Smart Inventory Management:

In our pursuit of efficiency, EasyPharm is incorporating smart technology to transform inventory management. By analyzing historical data stored in the database, EasyPharm will predict optimal inventory levels. This intelligent approach aims to reduce the likelihood of overstocking or stockouts, allowing pharmacies to operate more efficiently, minimize waste, and optimize resources.

In essence, these advancements underscore our commitment to making EasyPharm even more user-friendly and innovative. By simplifying processes and embracing emerging trends in healthcare technology, EasyPharm is evolving into a smarter and more effective platform. Our aim is to not only provide a seamless experience for patients but also to empower healthcare professionals in delivering top-notch services. EasyPharm is on a journey towards simplicity, efficiency, and innovation, ensuring it remains at the forefront of modern Pharmacy Management Systems.