Title: Deli-Meds

Use Case Study Report

Yaswanth Reddy Nalamalapu
Venkata Mani Sivasai Shanmukha Goparaju

617-777-5405 (Tel of Student 1)
857-397-5588 (Tel of Student 2)

nalamalapu.y@northeastern.edu
goparaju.v@northeastern.edu

Percentage of Effort Contributed by Student1: 50%

Percentage of Effort Contributed by Student2: 50%

Signature of Student 1: Yaswanth Reddy Nalamalapu

Signature of Student 2: Venkata Mani Sivasai Shanmukha Goparaju

Submission Date: 12/10/2023

USE CASE STUDY REPORT

Group No: Group 13

Students Name: Yaswanth Reddy Nalamalapu & Venkata Mani Sivasai Shanmukha Goparaju

Executive Summary:

The primary objective of this project is to design and implement a Relational Database that can be utilized to store the required data for the Deli-Meds project. Deli-Meds operates as an e-commerce platform focused on selling medicines, services, and products sourced from various pharmacies or through self-channels via an online website. The database plays a crucial role in storing data generated from orders on the website. It is utilized to recognize patterns, optimize sales, and generate recommendations through collected data, aiding in the development of various marketing strategies. Additionally, the database is instrumental in storing data generated from customer interactions with staff, addressing their concerns, and managing employee data to better track customer journeys.

To model the diverse data requirements inherent in the operation of a sales-oriented website, the database is conceptualized by creating an Entity-Relationship (EER) model and Unified Modeling Language (UML) diagrams. The EER model serves as the primary conceptual model, guiding the mapping to a relational model and subsequently generating a database schema. MySQL is employed as the software for implementing the relational model, while MongoDB is used as a test software to assess the feasibility of the relational model in a collection-based NoSQL model.

Following the generation and feasibility testing of the database, it is connected to Python using Jupyter Notebook. This connection facilitates drawing insights and generating visualizations for queries derived from the database, which stores the pertinent data for this project.

I. Introduction:

The pharmaceutical retail industry has experienced significant transformations over the past three years, driven by the evolving landscape due to the COVID-19 pandemic. To remain competitive and relevant in this dynamic market, several types of pharmacies are adapting and seeking new strategies. This adaptability includes exploring innovative approaches to cater to diverse consumer needs. While there is no one-size-fits-all solution, our proposal aims to revolutionize the pharmaceutical retail sector by uniting independent pharmacies and the online pharmacy model into a cohesive and nationally competitive service provider,

Market Segmentation:

In the US, the pharmaceutical retail market consists of four primary categories: national chains, regional pharmacies, independent pharmacies, and mail-order or online pharmacies. Each category employs unique strategies to maintain its market presence among these National chains and Regional Pharmacies have well-established market positions whereas independent pharmacies face challenges due to their smaller scale, while online pharmacies seek to thrive in an already saturated industry. Our primary business model centers on empowering independent pharmacies and enhancing the online pharmacy model to create a distinct, nationally

competitive entity. We envision collaborating with numerous independent pharmacies while preserving their individual brand identities.

Proposal:
As the Independent pharmacies are struggling due to their smaller scale in comparison to larger pharmacy chains. To remain viable, they must offer specialized services and focus on specific patient categories, providing personalized care. For example, a pharmacy could specialize in orthopedic care or incorporate telehealth services into their operations.

Our innovative concept involves enlisting independent pharmacies on our platform, allowing them to highlight their preferred service offerings. These services can range from none to multiple, tailored to their unique strengths and customer base.

In our project, we aim to aggregate numerous independent pharmacies within a locality, capturing essential business information such as company name, inventory, offered services, business credentials, medication and service pricing, date of enrollment, service hours, and more.

Business Requirements:

For a customer to add anything to the cart he/she should have a prescription.

A pharmacy can provide medicines, list their services, and sell their products too.

In future we can sell items ourselves too as our own brand.

A customer can pay his generated amount in one settlement or multiple settlements.

A Pharmacy can be associated with many carts from different customers, but a cart is associated with only one pharmacy.

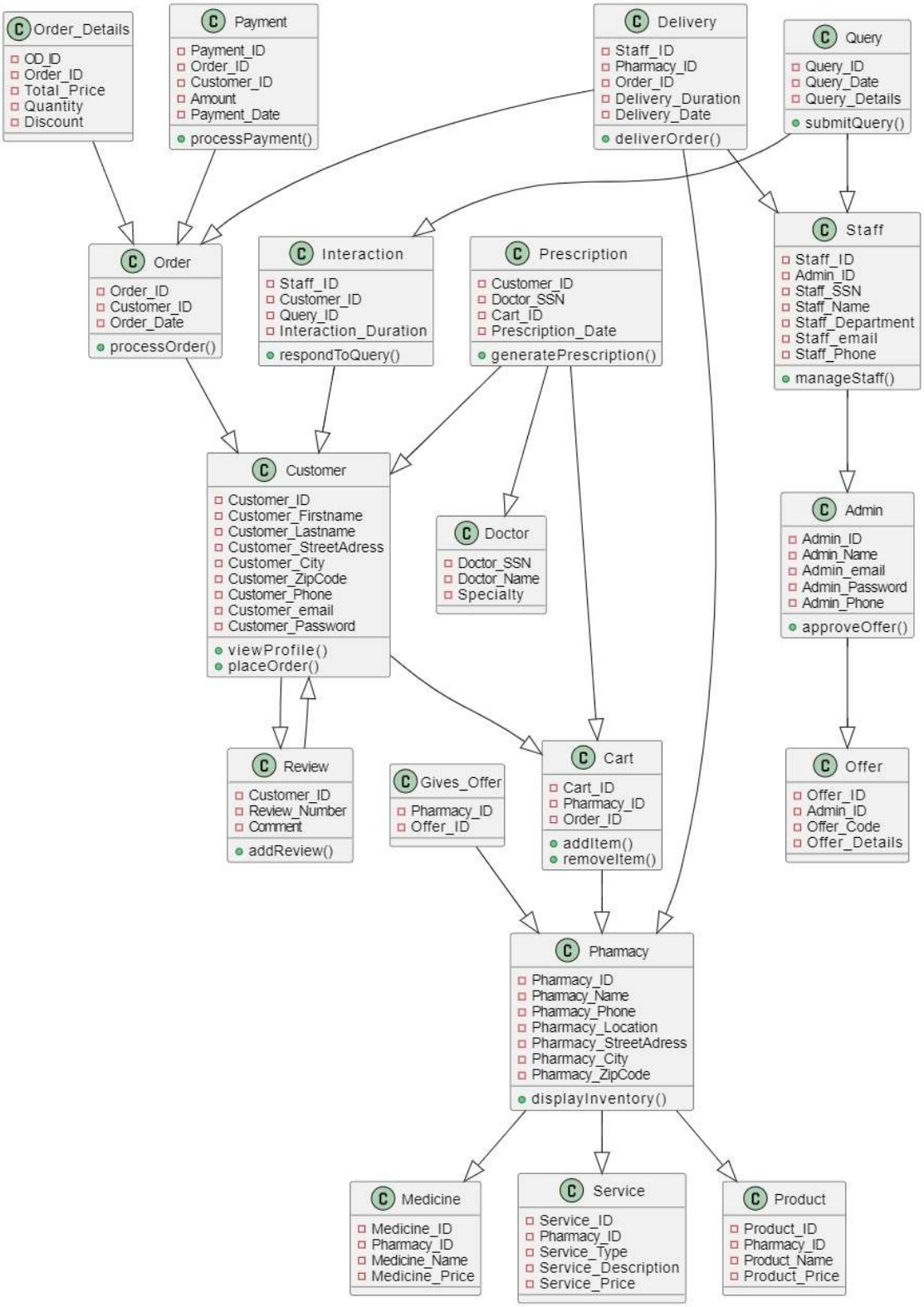
A medicine, service, and a product can associate themselves with the same pharmacy, if it provides all of them.

For a Interaction to occur there should exist a query, and for a particular query atleast 1 or many interactions can take place.

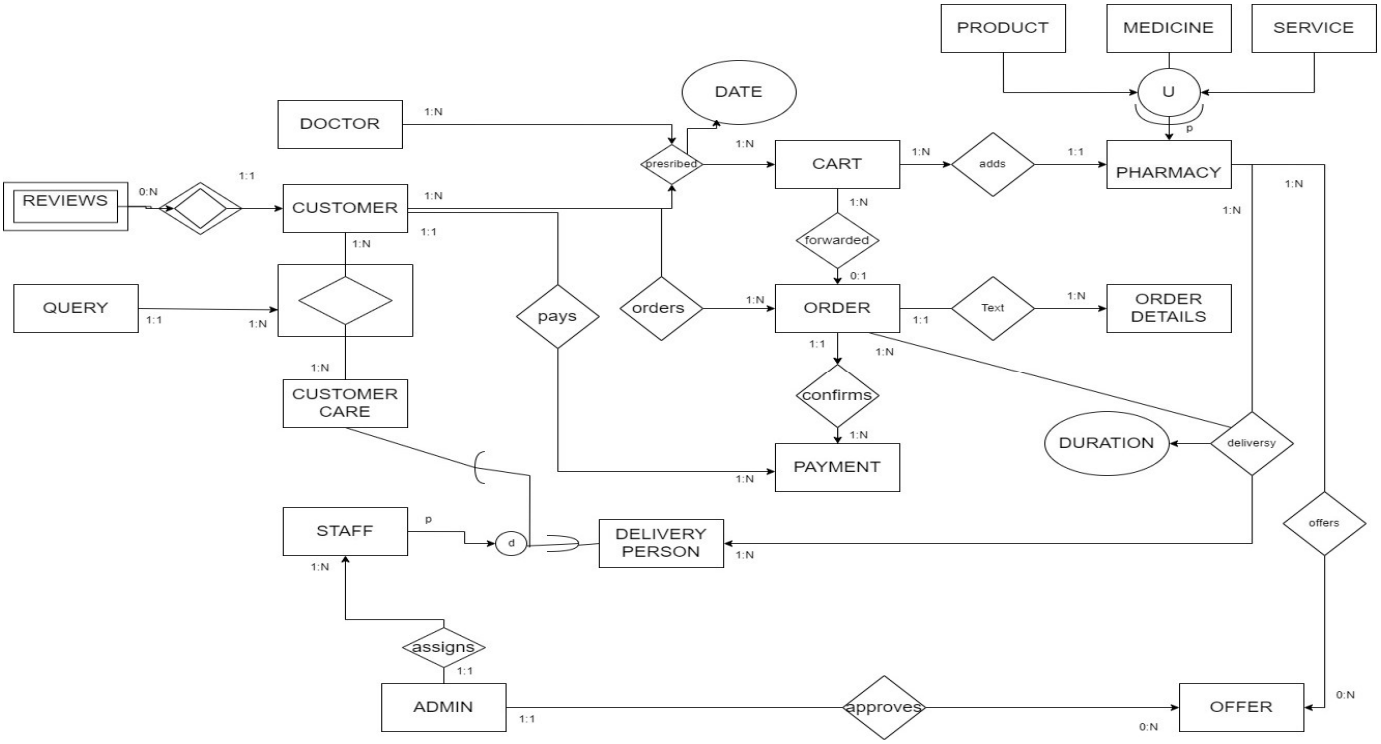
A customer can interaction with multiple staff members (Like if a solution is not found he can be forwarded to a higher-level staff) and a staff member can handle more than one customer at a time.

II. Conceptual Data Model:

1. UML Diagram:



2, EER Diagram:



III. Mapping Conceptual Model to Relational Model:

Primary Key – Underlined, **Foreign Key-** ***Italicized***

Customer (Customer_ID, Customer_Firstname, Customer_Lastname, Customer_StreetAdress, Customer_City, Customer_ZipCode, Customer_Phone, Customer_email, Customer_Password)

Primary Key: Customer_ID
Unique Keys: Customer_Phone, Customer_email, Customer_Password

Review (*Customer_ID*, Review_Number, Comment)
Primary Key: Customer_ID, Review_Number (Combination of Both)
Foreign Key: *Customer_ID*
Customer_ID is **NOT NULL (**References **Customer Table)**
Here Review is a weak entity of Customer

Doctor (Doctor_SSN, Doctor_Name, Specialty)
Primary Key: SSN

Cart (Cart_ID, *Pharmacy_ID*, *Order_ID*)
Primary Key: Cart_ID
Foreign Keys: *Pharmacy_ID*, *Order_ID*
Pharmacy_ID is **NOT NULL (**References **Pharmacy Table)**, Order_ID is **NULL Allowed (**References **Order Table**

Prescription (*Customer_ID*, *Doctor_SSN*, *Cart_ID*, Prescription_Date)
Primary Key: Customer_ID, Doctor_SSN, Cart_ID (Combination of all Three)
Foreign Keys: *Customer_ID*, *Doctor_SSN*, *Cart_ID*
Customer_ID **(**References **Customer Table)**, Doctor_SSN **(**References **Doctor Table)**, Cart_ID **(**References **Cart Table)** are all **NOT NULL**
Here Prescription is a Ternary Relationship between Cart, Doctor & Customer

Pharmacy (Pharmacy_ID, Pharmacy_Name, Pharmacy_Phone, Pharmacy_Location, Pharmacy_StreetAdress, Pharmacy_City, Pharmacy_ZipCode)
Primary Key: Pharmacy_ID (Can be called as a Surrogate Key)
Here Pharmacy is formed as a (Union & Partial) Categorization of Medicine, Service, & Product Entities

Pharmacy_Phone (Pharmacy_Phnr, *Pharmacy_ID*)
Primary Key: Pharmacy_Phnr

Medicine (Medicine_ID, *Pharmacy_ID*, Medicine_Name, Medicine_Price)
Primary Key: Medicine_ID
Foreign Key: *Pharmacy_ID*
Pharmacy_ID is **NOT NULL (**References **Pharmacy Table)**

Service (Service_ID, *Pharmacy_ID,* Service_Type, Service_Description, Service_Price)
Primary Key: Service_ID
Foreign Key: *Pharmacy_ID*
Pharmacy_ID is **NOT NULL (**References **Pharmacy Table)**

Product (Product_ID, *Pharmacy_ID*, Product_Name, Product_Price)
Primary Key: Product_ID
Foreign Key: *Pharmacy_ID*
Pharmacy_ID is **NOT NULL (**References **Pharmacy Table**

Order (Order_ID, *Customer_ID*, Order_Date)
Primary Key: Order_ID
Foreign key: *Customer_ID*
Customer_ID is **NOT NULL (**References **Customer Table)**

Order_Details (OD_ID, *Order_ID*, Total_Price, Quantity, Discount)
Primary Key: OD_ID
Foreign Key: *Order_ID*
Order_ID is **NOT NULL (**References **Order Table)**

Payment (Payment_ID, *Order_ID, Customer_ID*, Amount, Payment_Date)
Primary Key: Payment_ID
Foreign Keys: *Order_ID, Customer_ID*
Order_ID **(**References **Order Table)**, Customer_ID **(**References **Customer Table)** are **NOT NULL**

Admin (Admin_ID, Admin_Name, Admin_email, Admin_Password, Admin_Phone)
Primary Key: Admin_ID
Unique Keys: Admin_email, Admin_Password, Admin_Phone

Offer (Offer_ID, *Admin_ID*, Offer_Code, Offer_Details)

Primary Key: Offer_ID
Foreign Key: *Admin_ID*
Admin_ID is **NOT NULL (**References **Admin table)**

Gives_Offer (*Pharmacy_ID*, *Offer_ID*)
Primary Key: Pharmacy_ID, Offer_ID (Combination of Both)
Foreign Keys: *Pharmacy_ID*, *Offer_ID*
Pharmacy_ID **(**References **Pharmacy Table)**, Offer_ID **(**References **Offer Table)** are **NOT NULL**

Staff (Staff_ID, *Admin_ID*, Staff_SSN, Staff_Name, Staff_Department, Staff_email, Staff_Phone)
Primary Key: Staff_ID
Foreign Key: *Admin_ID*
Admin_ID is **NOT NULL (**References **Admin table)**
Unique Keys: Staff_SSN, Staff_email, Staff_Phone
Here Staff table has specialization (Partial & Disjoint) with Customer_Care & Delivery_Person as its subclass

Query (Query_ID, Query_Date, Query_Details)
Primary Key: Query_ID

Interaction (*Staff_ID*, *Customer_ID*, *Query_ID*, Interaction_Duration)
Primary Key: Staff_ID, Customer_ID (Combination of Both)
Foreign Keys: *Query_ID*, *Staff_ID*, *Customer_ID*
Query_ID **(**References **Query Table)**, Staff_ID **(**References **Staff Table)**, Customer_ID **(**References **Customer Table)** are all **NOT NULL**
Here Interaction is an Aggregation Between Customer & Staff which is in relation with Query

Delivery (*Staff_ID*, *Pharmacy_ID*, *Order_ID*, Delivery_Duration, Delivery_Date)
Primary Key: Staff_ID, Pharmacy_ID, Order_ID (Combination of all Three)
Foreign Keys: *Staff_ID*, *Pharmacy_ID*, *Order_ID*
Staff_ID **(**References **Staff Table)**, Pharmacy_ID **(**References **Pharmacy Table)**, Order_ID **(**References **Order Table)**, are all **NOT NULL**
Here Delivery is a ternary Relation between Order, Pharmacy, Staff

IV. Implementation of Relational Model via MySQL and NoSQL:

1. **MySQL Implenetation:**

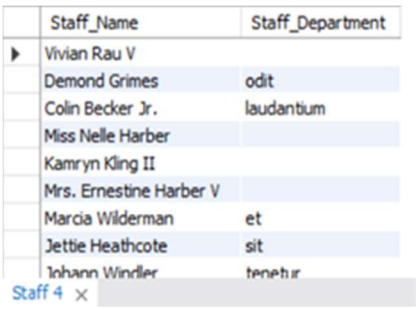
a. To find the Average payment paid by the customers:

```
SELECT c.Customer_ID, AVG(p.Amount) AS AverageAmountPaid
FROM Orders o, Customer c, Payment p
where c.Customer_ID = o.Customer_ID and
o.Order_ID = p.Order_ID
GROUP BY o.Customer_ID;
```

Customer_ID	AverageAmountPaid
8384774	183201.645000
8818411	17158680.120000
6430417	5984371.453000
9361179	8182088.278000
5363319	10565601.902000
6148846	10975285.245000
3842921	2468737.520000
8161728	10919795.685000
7642218	18889582.713000

b. To get Staff Names and their respective departments:

SELECT Staff_Name, Staff_Department FROM Staff;

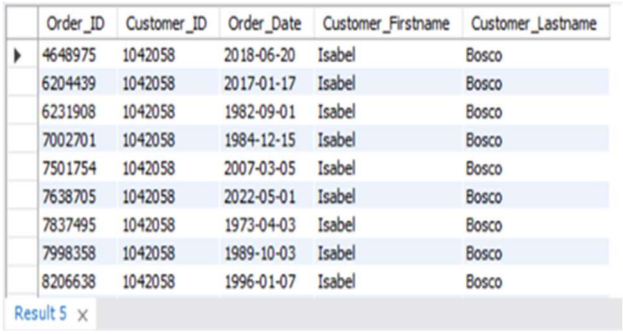


Staff_Name	Staff_Department
Vivian Rau V	
Demond Grimes	odit
Colin Becker Jr.	laudantium
Miss Nelle Harber	
Kamryn Kling II	
Mrs. Ernestine Harber V	
Marcia Wilderman	et
Jettie Heathcote	sit
Johann Windler	tenetur

Staff 4

c. To get The details of all the order with respect to customers first and last name:

SELECT Orders.*, Customer.Customer_Firstname, Customer.Customer_Lastname
FROM Orders
INNER JOIN Customer ON Orders.Customer_ID = Customer.Customer_ID;

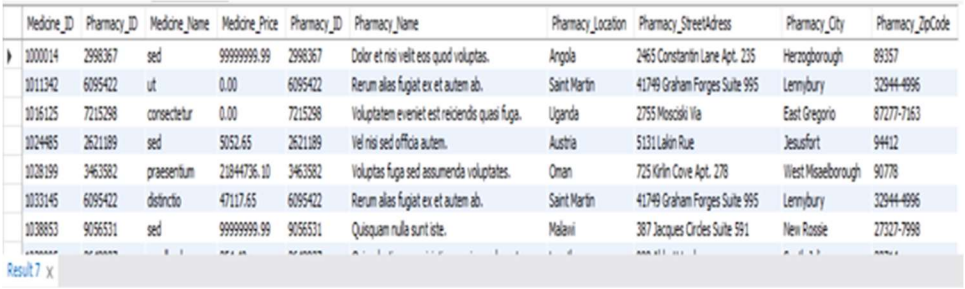


Order_ID	Customer_ID	Order_Date	Customer_Firstname	Customer_Lastname
4648975	1042058	2018-06-20	Isabel	Bosco
6204439	1042058	2017-01-17	Isabel	Bosco
6231908	1042058	1982-09-01	Isabel	Bosco
7002701	1042058	1984-12-15	Isabel	Bosco
7501754	1042058	2007-03-05	Isabel	Bosco
7638705	1042058	2022-05-01	Isabel	Bosco
7837495	1042058	1973-04-03	Isabel	Bosco
7998358	1042058	1989-10-03	Isabel	Bosco
8206638	1042058	1996-01-07	Isabel	Bosco

Result 5

d. Left outer Join of medicine and pharmacy table:

SELECT Medicine.*, Pharmacy.*
FROM Medicine
LEFT JOIN Pharmacy ON Medicine.Pharmacy_ID = Pharmacy.Pharmacy_ID;

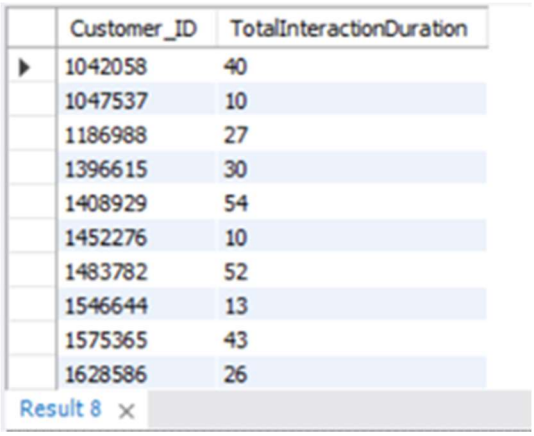


Medicine_ID	Pharmacy_ID	Medicine_Name	Medicine_Price	Pharmacy_ID	Pharmacy_Name	Pharmacy_Location	Pharmacy_StreetAdress	Pharmacy_City	Pharmacy_ZipCode
1000014	2998367	sed	99999999.99	2998367	Dolor et nisi velit eos quod voluptas.	Angola	2465 Constantin Lane Apt. 235	Herzogborough	89357
1011342	6095422	ut	0.00	6095422	Rerum alias fugiat ex et autem ab.	Saint Martin	41749 Graham Forges Suite 995	Lennybury	32944-4996
1016125	7215298	consectetur	0.00	7215298	Voluptatem eveniet est reiciendis quasi fuga.	Uganda	2755 Mosciski Via	East Gregorio	87277-7163
1024485	2621189	sed	5052.65	2621189	Vel nisi sed officia autem.	Austria	5131 Lakin Rue	Jesusfort	94412
1028199	3463582	praesentium	21844736.10	3463582	Voluptas fuga sed assumenda voluptates.	Oman	725 Kirlin Cove Apt. 278	West Misaeborough	90778
1033145	6095422	distinctio	47117.65	6095422	Rerum alias fugiat ex et autem ab.	Saint Martin	41749 Graham Forges Suite 995	Lennybury	32944-4996
1038853	9056531	sed	99999999.99	9056531	Quisquam nulla sunt iste.	Malawi	387 Jacques Circles Suite 591	New Rossie	27327-7998

Result 7

e. To get the total duration for the customers interacted with the customer care staff:

SELECT Customer_ID, SUM(Interaction_Duration) AS Total_Interaction_Duration
FROM Interaction
GROUP BY Customer_ID;



Customer_ID	TotalInteractionDuration
1042058	40
1047537	10
1186988	27
1396615	30
1408929	54
1452276	10
1483782	52
1546644	13
1575365	43
1628586	26

Result 8

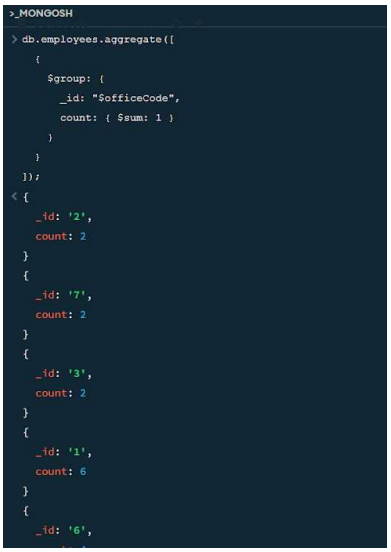
2. NoSQL Implementation:

Created a database named **deli_meds** in MongoDB through adding most of the tables as collections by importing the data in **JSON** format from **MySQL.**

The query shown in the above picture would be fetching the medicine_name with "**sed**" and with a medicine_price greater than **$10.**

The aggregated query shown in the above picture would be fetching the **number of employees** present with each individual **jobTitle** from the **employees** collection in the descending order of the number of employees per each jobTitle

As described above, the same goes with the number of office codes present in the employees collection and the respective count.

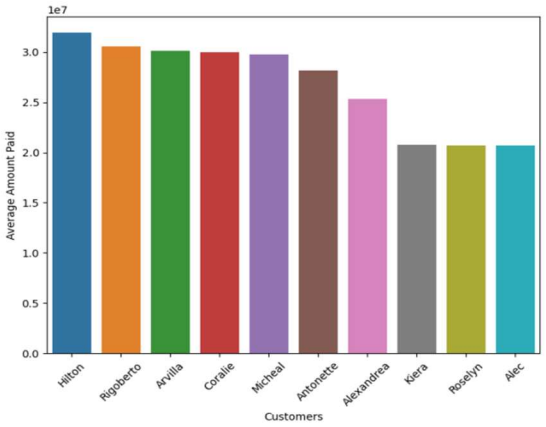


V. Database Access Via Python:

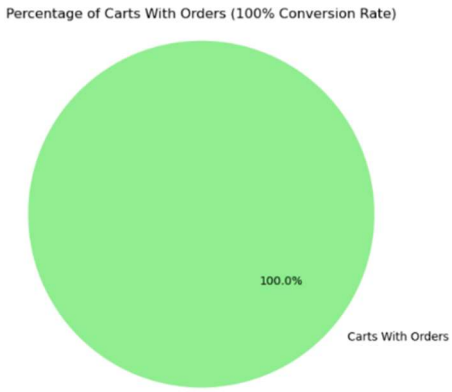
The Database is connected to python using Jupyter Notebook and Some queries are Visualized using python code.

This is achieved by usng mysql.connector library and using of cursor.excecute method from this library, then a function is defined to run the query which returns a list and then converting the result to a dataframe to visualize the results using matplotlib and pandas library.

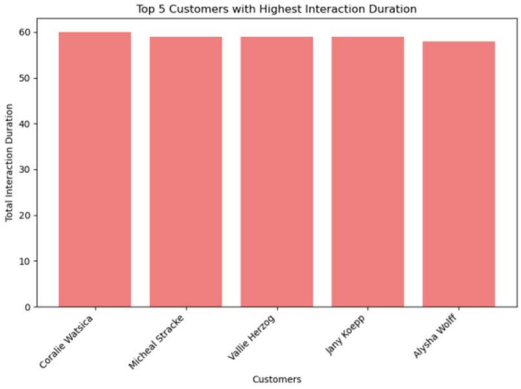
Plot 1: Top ten customers based on their average spending's:



Plot 2: To get the percentage of carts converted to orders:



Plot 3: Customers with Highest Interaction Duration:



VI. Summary and Recommendations:

This Database which was implemented in MySQL for the project not only fulfills the immediate need for a robust database for Deli-Meds but also explores alternative database models to ensure flexibility and scalability in the evolving landscape of e-commerce. The integration of relational and NoSQL databases, along with the connectivity to Python, positions the database as a valuable tool for data-driven decision-making in the context of an online sales platform.

In Future The database can be improved by adding views and triggers etc. to better govern the data and provide data integrity based on the requirements. There is also a scope for addition of the Prescriptions directly which can then be scanned for better ordering experience.

Also This database can be better improved by using NoSQL implementation, as here it was only tested to know whether the database is feasible or not and based on the observations it is noted that a document based NoSQL is more appropriate for this kind of data storage and retrieving tasks done for this model.