MediDaily: Online Pharmacy Management

PROJECT REPORT

Group 30

Ansh Chhadva

Sakshi Gujarathi

857 376 7574 (Tel: Student 1)
857 222 1850 (Tel: Student 2)

chhadva.an@northeastern.edu
gujarathi.sa@northeastern.edu

Percentage of Effort Contributed by Student1: 50%

Percentage of Effort Contributed by Student2: 50%

Signature of Student 1:

Signature of Student 2:

Submission Date: 10/12/2022

Executive Summary:

The goal of this study was to design and implement a database for an online and offline pharmaceutical store, MediDaily. In order to tackle the increasing demand of healthcare goods and prescription drugs, MediDaily's subscription model feature aims to save users the hassle of reordering life-sustaining medications. The database will help with visualizing analytical solutions and efficient storage of data with close to zero or no duplication.

With connectivity to Python, analytics for all business needs and climates can be provided and calculated which would be of immense benefit to scale up the business. The database was modeled by taking into account the system requirements and sensitive medical data. The EER and UML diagrams were modeled and mapped into a relational model with appropriate primary keys, foreign keys and null values. After the normalization of tables to eliminate possible data redundancy, the same were created and implemented on MySQL followed by execution in MongoDB, a NoSQL environment.

The Medidaily database was created successfully to support external business data and internal in-store employee and store data. This would help keep an automated digital record of the data in place and easily access the same as and when required. MediDaily database is also capable of tracking trends and running business queries to generate useful insights such as the highest medicine sales in a particular location, a product with the highest rating etc. This would help them to take strategic business decisions such as pre-ordering more products from specific suppliers, giving discounts to customers who have not used their application often etc.

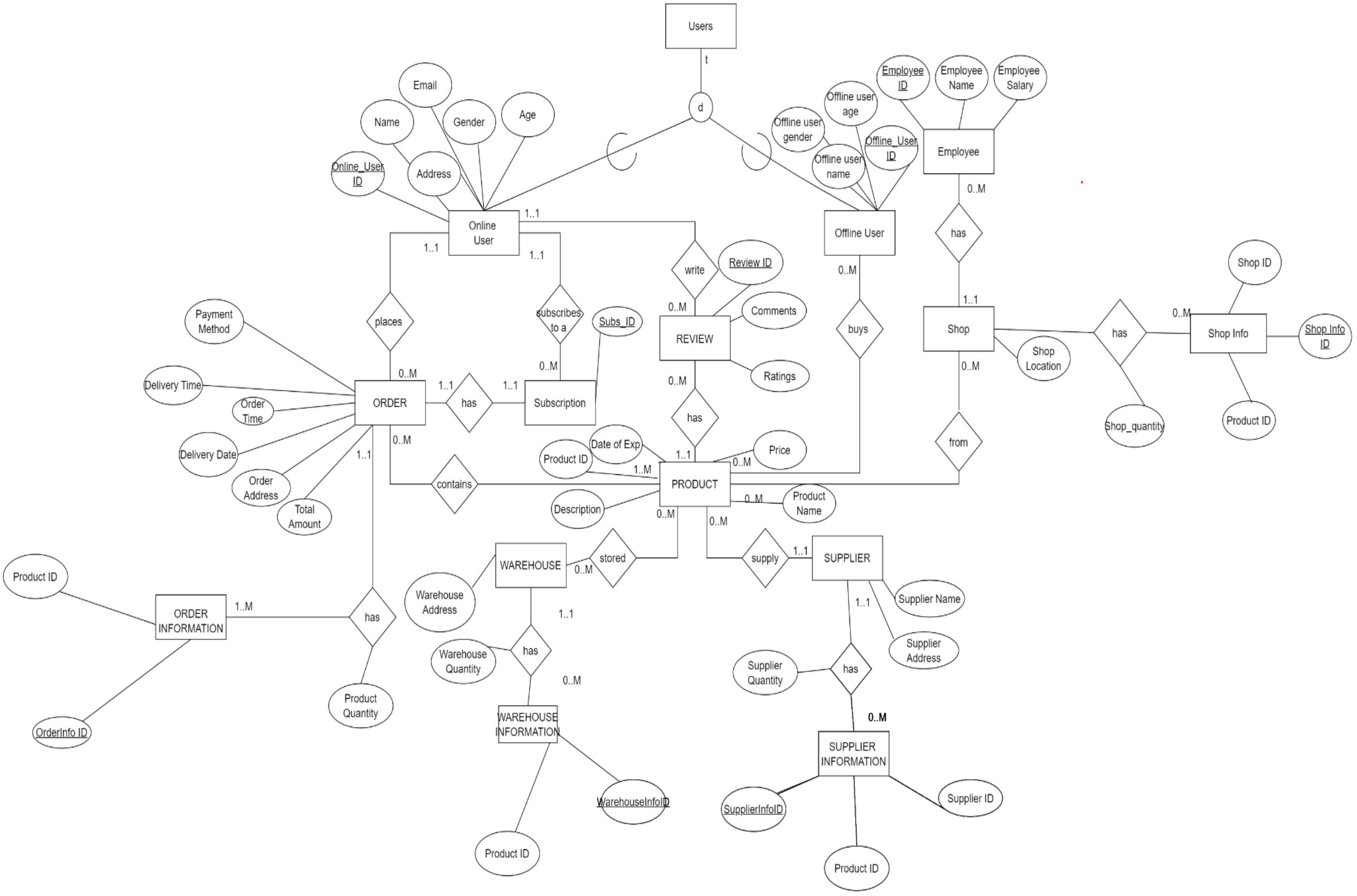
I. Business Problem Statement & Requirements:

Background: Due to the advent of the internet coupled with the COVID-19 pandemic, there has been a surge in demand for over-the-counter medicines and the pharmaceutical market in general. Apart from that, lack of physical activity, poor eating habits and stress have been significant contributors to the rise in chronic disorders like hypertension, high cholesterol and kidney diseases that need timely long-term medication thus increasing the demand for prescription drugs.

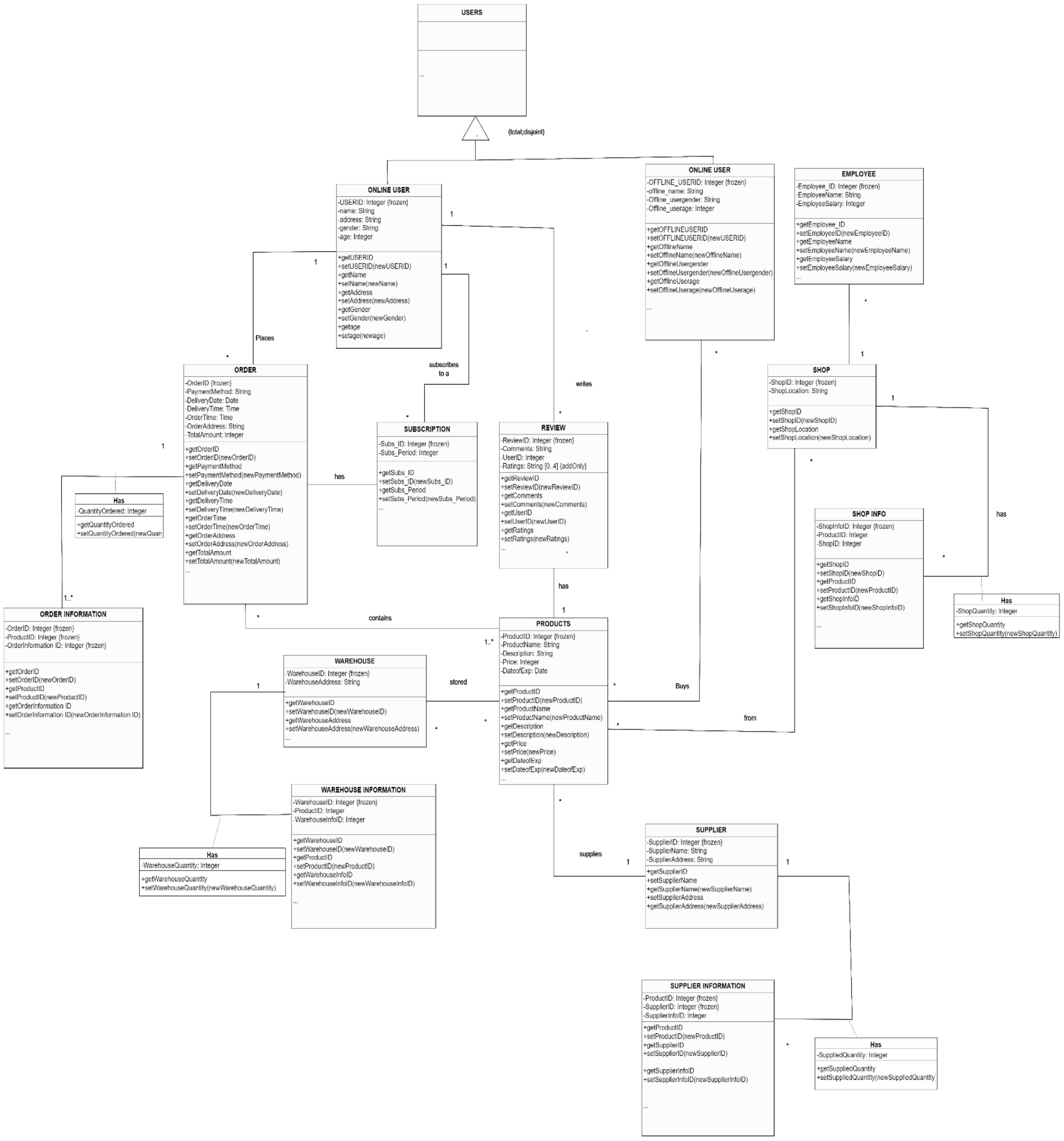
Problem Statement: Prevalent healthcare problems in the recent past have caused a tremendous rise in demand for healthcare products. To cater to these needs of consumers, MediDaily provides easy and convenient solutions for ordering medicines online or picking them up from the nearest outlets. This would save time and energy and give consumers a larger range of brands and products to choose from. With its monthly, quarterly, or yearly subscription models, it captures the market requirement of the elderly and health-cautious people of today.

In our project, we have given the users both options: they can order medicines online or buy them directly from the local shops. For each user, we would be storing the name, email id, gender, address, and age, and each product will have its name, description, and price. Whenever the user orders online, we would store the details of that particular order, like order date and time, all the names of products, total price, total number, delivery address, delivery date and time, etc. The users can also subscribe to a particular set of products which will be delivered automatically according to their needs. For this, we would record the number and name of products, the duration of the following orders, and much more. Moreover, the user can also give ratings and reviews for the products. We would be keeping track of the review description, the ratings of the product (out of 5), and the unique id of that particular user. MediDaily would also have shops and warehouses where we would record information like shop names, warehouse names, locations, and the ids of the products they have. Finally, we would also be storing the details of the employees, such as employee name, age, gender, and address, for safety purposes.

II. Conceptual Data Modeling - EER Diagram:



Conceptual Data Modeling - UML Diagram


USERS
...
{total;disjoint}
ONLINE USER
-USERID: Integer {frozen}
-name: String
-address: String
-gender: String
-age: Integer
+getUSERID
+setUSERID(newUSERID)
+getName
+setName(newName)
+getAddress
+setAddress(newAddress)
+getGender
+setGender(newGender)
+getage
+setage(newage)
ONLINE USER
-OFFLINE_USERID: Integer {frozen}
-offline_name: String
-Offline_usergender: String
-Offline_userage: Integer
+getOFFLINEUSERID
+setOFFLINEUSERID(newUSERID)
+getOfflineName
+setOfflineName(newOfflineName)
+getOfflineUsergender
+setOfflineUsergender(newOfflineUsergender)
+getOfflineUserage
+setOfflineUserage(newOfflineUserage)
...
EMPLOYEE
-Employee_ID: Integer {frozen}
-EmployeeName: String
-EmployeeSalary: Integer
+getEmployee_ID
+setEmployeeID(newEmployeeID)
+getEmployeeName
+setEmployeeName(newEmployeeName)
+getEmployeeSalary
+setEmployeeSalary(newEmployeeSalary)
...
Places
subscribes to a
writes
ORDER
-OrderID {frozen}
-PaymentMethod: String
-DeliveryDate: Date
-DeliveryTime: Time
-OrderTime: Time
-OrderAddress: String
-TotalAmount: Integer
+getOrderID
+setOrderID(newOrderID)
+getPaymentMethod
+setPaymentMethod(newPaymentMethod)
+getDeliveryDate
+setDeliveryDate(newDeliveryDate)
+getDeliveryTime
+setDeliveryTime(newDeliveryTime)
+getOrderTime
+setOrderTime(newOrderTime)
+getOrderAddress
+setOrderAddress(newOrderAddress)
+getTotalAmount
+setTotalAmount(newTotalAmount)
...
has
SUBSCRIPTION
-Subs_ID: Integer {frozen}
-Subs_Period: Integer
+getSubs_ID
+setSubs_ID(newSubs_ID)
+getSubs_Period
+setSubs_Period(newSubs_Period)
...
REVIEW
-ReviewID: Integer {frozen}
-Comments: String
-UserID: Integer
-Ratings: String [0..4] {addOnly}
+getReviewID
+setReviewID(newReviewID)
+getComments
+setComments(newComments)
+getUserID
+setUserID(newUserID)
+getRatings
+setRatings(newRatings)
...
SHOP
-ShopID: Integer {frozen}
-ShopLocation: String
+getShopID
+setShopID(newShopID)
+getShopLocation
+setShopLocation(newShopLocation)
has
SHOP INFO
-ShopInfoID: Integer {frozen}
-ProductID: Integer
-ShopID: Integer
+getShopID
+setShopID(newShopID)
+getProductID
+setProductID(newProductID)
+getShopInfoID
+setShopInfoID(newShopInfoID)
...
Has
-ShopQuantity: Integer
+getShopQuantity
+setShopQuantity(newShopQuantity)
Has
-QuantityOrdered: Integer
+getQuantityOrdered
+setQuantityOrdered(newQuan
ORDER INFORMATION
-OrderID: Integer {frozen}
-ProductID: Integer {frozen}
-OrderInformation ID: Integer {frozen}
+getOrderID
+setOrderID(newOrderID)
+getProductID
+setProductID(newProductID)
+getOrderInformation ID
+setOrderInformation ID(newOrderInformation ID)
...
contains
has
PRODUCTS
-ProductID: Integer {frozen}
-ProductName: String
-Description: String
-Price: Integer
-DateofExp: Date
+getProductID
+setProductID(newProductID)
+getProductName
+setProductName(newProductName)
+getDescription
+setDescription(newDescription)
+getPrice
+setPrice(newPrice)
+getDateofExp
+setDateofExp(newDateofExp)
...
Buys
from
stored
WAREHOUSE
-WarehouseID: Integer {frozen}
-WarehouseAddress: String
+getWarehouseID
+setWarehouseID(newWarehouseID)
+getWarehouseAddress
+setWarehouseAddress(newWarehouseAddress)
...
WAREHOUSE INFORMATION
-WarehouseID: Integer {frozen}
-ProductID: Integer
-WarehouseInfoID: Integer
+getWarehouseID
+setWarehouseID(newWarehouseID)
+getProductID
+setProductID(newProductID)
+getWarehouseInfoID
+setWarehouseInfoID(newWarehouseInfoID)
...
Has
-WarehouseQuantity: Integer
+getWarehouseQuantity
+setWarehouseQuantity(newWarehouseQuantity)
supplies
SUPPLIER
-SupplierID: Integer {frozen}
-SupplierName: String
-SupplierAddress: String
+getSupplierID
+setSupplierName
+getSupplierName(newSupplierName)
+setSupplierAddress
+getSupplierAddress(newSupplierAddress)
SUPPLIER INFORMATION
-ProductID: Integer {frozen}
-SupplierID: Integer {frozen}
-SupplierInfoID: Integer
+getProductID
+setProductID(newProductID)
+getSupplierID
+setSupplierID(newSupplierID)
+getSupplierInfoID
+setSupplierInfoID(newSupplierInfoID)
...
Has
-SuppliedQuantity: Integer
+getSuppliedQuantity
+setSuppliedQuantity(newSuppliedQuantity
1
*
1..*

III. Mapping Conceptual Model to Relational Model (Relational Mapping):

1. Order(<u>OrderID</u>,*OnlineUserID*,OrderTime,Delivery_Date,OrderAddress,TotalAmount,PaymentMethod)
   OnlineUserID(FK) : Not NULL

2. Subscription(Subs_ID,*OrderID,OnlineUserID*)
   OrderID(FK) : Not NULL OnlineUserID(FK) :Not NULL

3. Product(ProductID, Product name, Description, Price, DateofExpiry)

4. Product_Order(*ProductID,OrderID*,Quantity_Ordered)
   ProductID,OrderID(FK): Not NULL

5. Product_OfflineUser(*ProductID,OfflineUserID*,Offline_Quantity_ordered)
   ProductID,OfflineUserID(FK): Not NULL

6. Shop(ShopID,Shop_Location)

7. Product_Shop(*ProductID,ShopID*,shop_quantity)
   ProductID,ShopID(FK):Not NULL

8. Warehouse(WarehouseID,Warehouse_Address)
9. Product_Warehouse*(ProductID,WarehouseID*,Warehouse_Quantity
   ProductID,WarehouseID(FK):Not NULL

10. Supplier(SupplierID,SupplierName,Supplier_Address)

11. Product_Supplier(*ProductID,SupplierID*,Supplier_Quantity)
    ProductID,SupplierID(FK):Not NULL

12. Employee(EmployeeID, Employee Name, Employee Salary, *ShopID*)
    ShopID(FK) : Not NULL

13. Review(ReviewID, Comments, Ratings,*OnlineUserID*)
    OnlineUserID(FK) : Not NULL

14. Users(User_id)

15. OnlineUser(OnlineUser_ID,*User_id*.Name,Address,Gender)
    User_id(FK): Not NULL

16. OfflineUser(OfflineUser_ID,User_id,Offline_User_Name,Offline_User_Gender,Offline_User_Age)
    User_id(FK): Not NULL

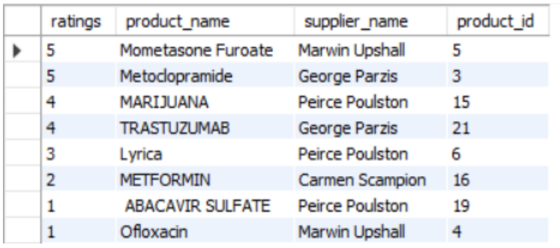
IV. Implementation of Relation Model via MySQL and NoSQL

A. MySQL Implementation:

After implementing the database in MySQL, the following queries were executed.

Query 1: Get those product names which have highest ratings and display their supplier names

```
select r.ratings,p.product_name,
s.supplier_name,ps.product_id from review r
JOIN product p on r.product_id = p.product_id
```

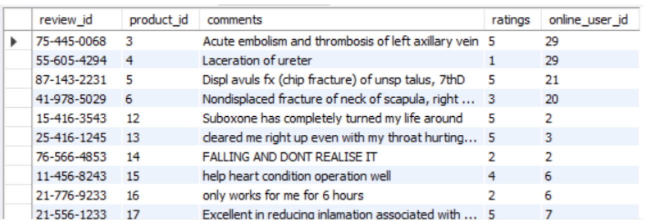


ratings	product_name	supplier_name	product_id
5	Mometasone Furoate	Marwin Upshall	5
5	Metoclopramide	George Parzis	3
4	MARIJUANA	Peirce Poulston	15
4	TRASTUZUMAB	George Parzis	21
3	Lyrica	Peirce Poulston	6
2	METFORMIN	Carmen Scampion	16
1	ABACAVIR SULFATE	Peirce Poulston	19
1	Ofloxacin	Marwin Upshall	4

```
Join product_supplier ps on ps.product_id=r.product_id
join supplier s on ps.supplier_id=s.supplier_id
order by ratings DESC;
```

Query 2: -- Retrieve only those product_ids who have some ratings and reviews

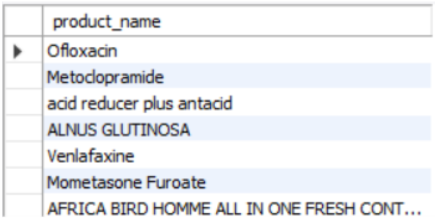
```
select * from review r where  exists
 (select * from product p where r.product_id =
p.product_id);
```



	review_id	product_id	comments	ratings	online_user_id
▶	75-445-0068	3	Acute embolism and thrombosis of left axillary vein	5	29
	55-605-4294	4	Laceration of ureter	1	29
	87-143-2231	5	Displ avuls fx (chip fracture) of unsp talus, 7thD	5	21
	41-978-5029	6	Nondisplaced fracture of neck of scapula, right ...	3	20
	15-416-3543	12	Suboxone has completely turned my life around	5	2
	25-416-1245	13	cleared me right up even with my throat hurting...	5	3
	76-566-4853	14	FALLING AND DONT REALISE IT	2	2
	11-456-8243	15	help heart condition operation well	4	6
	21-776-9233	16	only works for me for 6 hours	2	6
	21-556-1233	17	Excellent in reducing inlamation associated with ...	5	7

Query 3:-- select the product names whose quantity inside the shop is greater than 5000

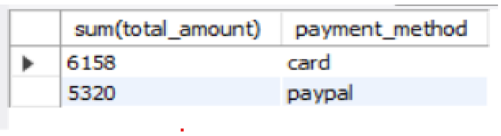
```
SELECT product_name FROM product WHERE product_id = ANY
 (SELECT product_id   FROM product_shop   WHERE shop_quantity <
5000) ;
```



	product_name
▶	Ofloxacin
	Metoclopramide
	acid reducer plus antacid
	ALNUS GLUTINOSA
	Venlafaxine
	Mometasone Furoate
	AFRICA BIRD HOMME ALL IN ONE FRESH CONT...

Query 4: -- Calculate the sum of total amount of orders done by each payment method

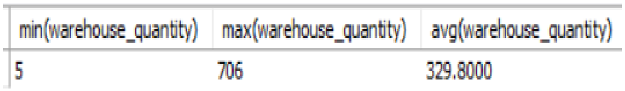
```
select sum(total_amount),payment_method  from order_table
group by payment_method;
```



	sum(total_amount)	payment_method
▶	6158	card
	5320	paypal

Query 5 -- Display the minimum ,maximum and average warehouse quantity

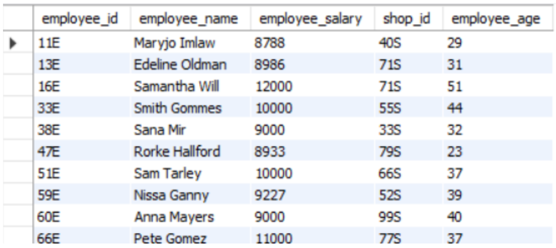
```
select min(warehouse_quantity),
max(warehouse_quantity),avg(warehouse_quantity)
from product_warehouse;
```



min(warehouse_quantity)	max(warehouse_quantity)	avg(warehouse_quantity)
5	706	329.8000

Query 6 -- select those employees whose salary is greater than the avg salary of all the employees

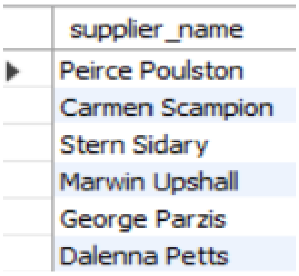
```
select * from employee where employee_salary >(select
avg(employee_salary) from employee);
```



	employee_id	employee_name	employee_salary	shop_id	employee_age
▶	11E	Maryjo Imlaw	8788	40S	29
	13E	Edeline Oldman	8986	71S	31
	16E	Samantha Will	12000	71S	51
	33E	Smith Gommes	10000	55S	44
	38E	Sana Mir	9000	33S	32
	47E	Rorke Hallford	8933	79S	23
	51E	Sam Tarley	10000	66S	37
	59E	Nissa Ganny	9227	52S	39
	60E	Anna Mayers	9000	99S	40
	66E	Pete Gomez	11000	77S	37

Query 7 retrive the supplier name who supplies those products whos price is greater than 50

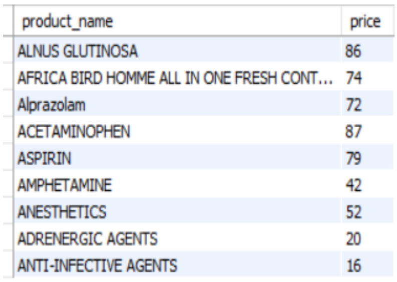
```
select supplier_name from supplier where supplier_id IN
(select supplier_id from product_supplier where product_id IN
(select product_id from product where price<50));
```



	supplier_name
▶	Peirce Poulston
	Carmen Scampion
	Stern Sidary
	Marwin Upshall
	George Parzis
	Dalenna Petts

Query 8 -- select those product name   as well as their price whose product name starts with A

```
select product_name,price from product
```



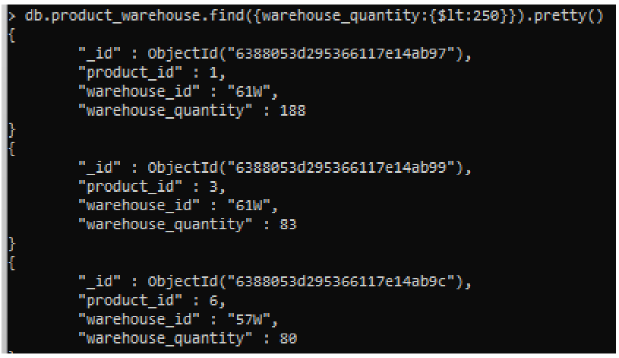
product_name	price
ALNUS GLUTINOSA	86
AFRICA BIRD HOMME ALL IN ONE FRESH CONT...	74
Alprazolam	72
ACETAMINOPHEN	87
ASPIRIN	79
AMPHETAMINE	42
ANESTHETICS	52
ADRENERGIC AGENTS	20
ANTI-INFECTIVE AGENTS	16

where product_name LIKE 'A%';

B. NoSQL Implementation:

Query 1: : Find warehouse products with a quantity of less than 250

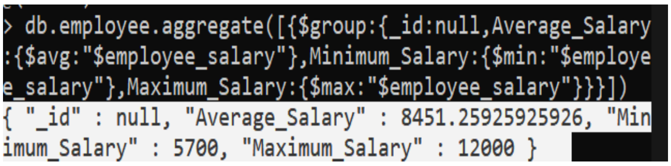
db.product_warehouse.find({ warehouse_quantity:{$lt:250}}).pretty()



```
> db.product_warehouse.find({warehouse_quantity:{$lt:250}}).pretty()
{
        "_id" : ObjectId("6388053d295366117e14ab97"),
        "product_id" : 1,
        "warehouse_id" : "61W",
        "warehouse_quantity" : 188
}
{
        "_id" : ObjectId("6388053d295366117e14ab99"),
        "product_id" : 3,
        "warehouse_id" : "61W",
        "warehouse_quantity" : 83
}
{
        "_id" : ObjectId("6388053d295366117e14ab9c"),
        "product_id" : 6,
        "warehouse_id" : "57W",
        "warehouse_quantity" : 80
```

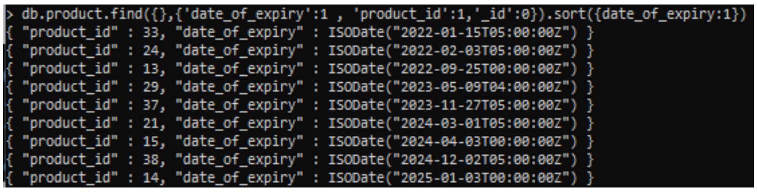
Query 2: Display the minimum, maximum and average salary of employees

db.employee.aggregate([{$group:{_id:null,Average_Salary:{$avg:"$employee_salary"}, Minimum_Salary:{$min:"$employee_salary"}, Maximum_Salary:{$max:"$employee_salary"}}}])



```
> db.employee.aggregate([{$group:{_id:null,Average_Salary
:{$avg:"$employee_salary"},Minimum_Salary:{$min:"$employe
e_salary"},Maximum_Salary:{$max:"$employee_salary"}}}])
{ "_id" : null, "Average_Salary" : 8451.25925925926, "Min
imum_Salary" : 5700, "Maximum_Salary" : 12000 }
```

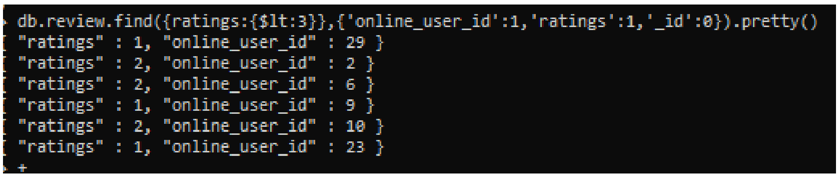
Query 3 :Products with the earliest date of expiration.
db.product.find({},{'date_of_expiry':1 , 'product_id':1,'_id':0}).
sort({date_of_expiry:1})



```
> db.product.find({},{'date_of_expiry':1 , 'product_id':1,'_id':0}).sort({date_of_expiry:1})
{ "product_id" : 33, "date_of_expiry" : ISODate("2022-01-15T05:00:00Z") }
{ "product_id" : 24, "date_of_expiry" : ISODate("2022-02-03T05:00:00Z") }
{ "product_id" : 13, "date_of_expiry" : ISODate("2022-09-25T00:00:00Z") }
{ "product_id" : 29, "date_of_expiry" : ISODate("2023-05-09T04:00:00Z") }
{ "product_id" : 37, "date_of_expiry" : ISODate("2023-11-27T05:00:00Z") }
{ "product_id" : 21, "date_of_expiry" : ISODate("2024-03-01T05:00:00Z") }
{ "product_id" : 15, "date_of_expiry" : ISODate("2024-04-03T00:00:00Z") }
{ "product_id" : 38, "date_of_expiry" : ISODate("2024-12-02T05:00:00Z") }
{ "product_id" : 14, "date_of_expiry" : ISODate("2025-01-03T00:00:00Z") }
```

Query 4: Display the online user IDs of user ratings below three.

db.review.find({ratings:{$lt:3}}, {'online_user_id':1,'ratings':1, '_id':0}).pretty()

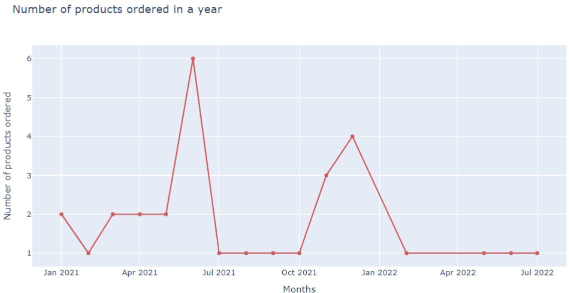


```
> db.review.find({ratings:{$lt:3}},{'online_user_id':1,'ratings':1,'_id':0}).pretty()
{ "ratings" : 1, "online_user_id" : 29 }
{ "ratings" : 2, "online_user_id" : 2 }
{ "ratings" : 2, "online_user_id" : 6 }
{ "ratings" : 1, "online_user_id" : 9 }
{ "ratings" : 2, "online_user_id" : 10 }
{ "ratings" : 1, "online_user_id" : 23 }
> +
```

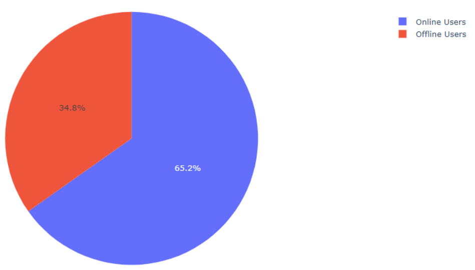
V. Database Access Python

The database is accessed using Python's library such as Pandas library and using Matplotlib to plot the graphs for the analytics.

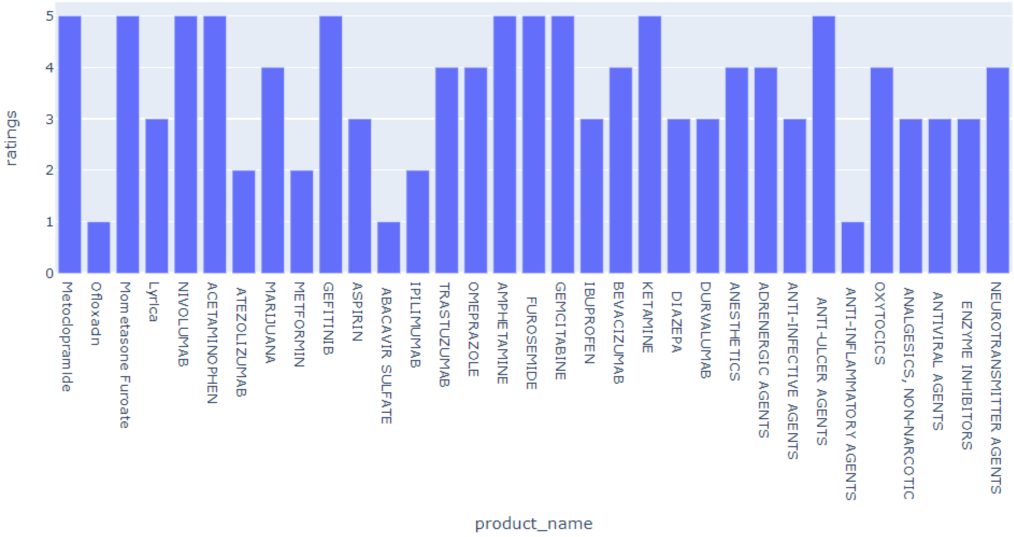
Graph 1: Plotted a line graph of no of products vs corresponding month



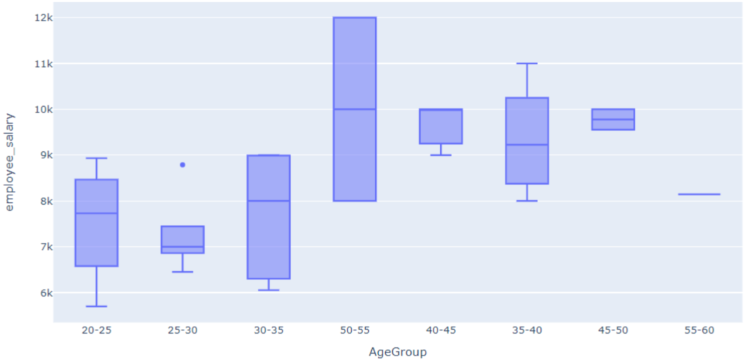
Graph2: Pie chartOffline vs Online



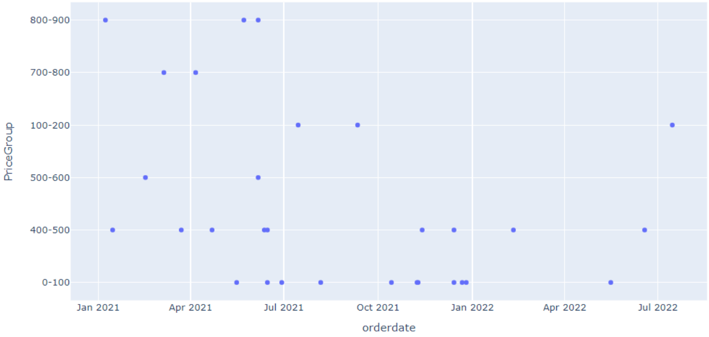
<u>Graph 3:</u>Bar graph with medicine name on the X-axis and rating on the Y-axis



<u>Graph 4:</u>Boxplot for Salary Distribution of the employees working at MediDaily stores:



<u>Graph 5:</u>created different price groups and scatter plot with



VII. Summary and recommendation

<u>A. Project Conclusion:</u>

-The MediDaily database is successfully implemented in mysql as well as mongo db database.This database has been designed and developed carefully taking care of limiting data redundancy as far as possible . It's a perfectly stable and ready to use version in the real world environment . Having such a database will be great as it will save a lot of time and energy thereby providing easy retrieval and storage process.Moreover,this particular database has great capabilities of providing well defined analytics. Overall,We have conveniently administered the problem statement and conceptual Modelling using EER & UML Diagram in SQL database as well as NoSQL database. Also, we have successfully implemented the database access via python & demonstrated practical visualisation analytics.

B. Shortcomings:

Instead of having all these databases inside a local server it would have been better if the database would be set up on cloud So that the whole organisation could have been easily accessed. Moreover, Using a tool such as Power Bi or Tableau , would have been better than using python since such tools are specifically focused on analytics only