IE 7374 Data Management for Analytics

Use Case Study Report

Group No.: Group 11

Student Names : Sai Varun Kumar Namburi and FNU Meenal

Executive Summary:

Efficient inventory management enhances gross profits and net profits by reducing the cost of procured pharmaceutical products and associated operational expenses. We are focusing on selecting appropriate fulfillment strategies that match the business's size and the volume and type of orders it receives. Based on these strategies we can help the organization maintain an inventory that can ship products faster, minimize waste, and improve customer satisfaction which will help maintain the most effective workflow.

The model will include the data required for the day-to-day operations to help maintain information about the inventory. At a high level, this includes data about transactional data like receiving and shipping inventory, fulfilling orders, and managing inventory in warehouse space.

This model ensures that the product will be utilized before it expires and should help your organization to achieve greater efficiency and fulfill orders more accurately so you can do more at a lower cost.

I. Introduction

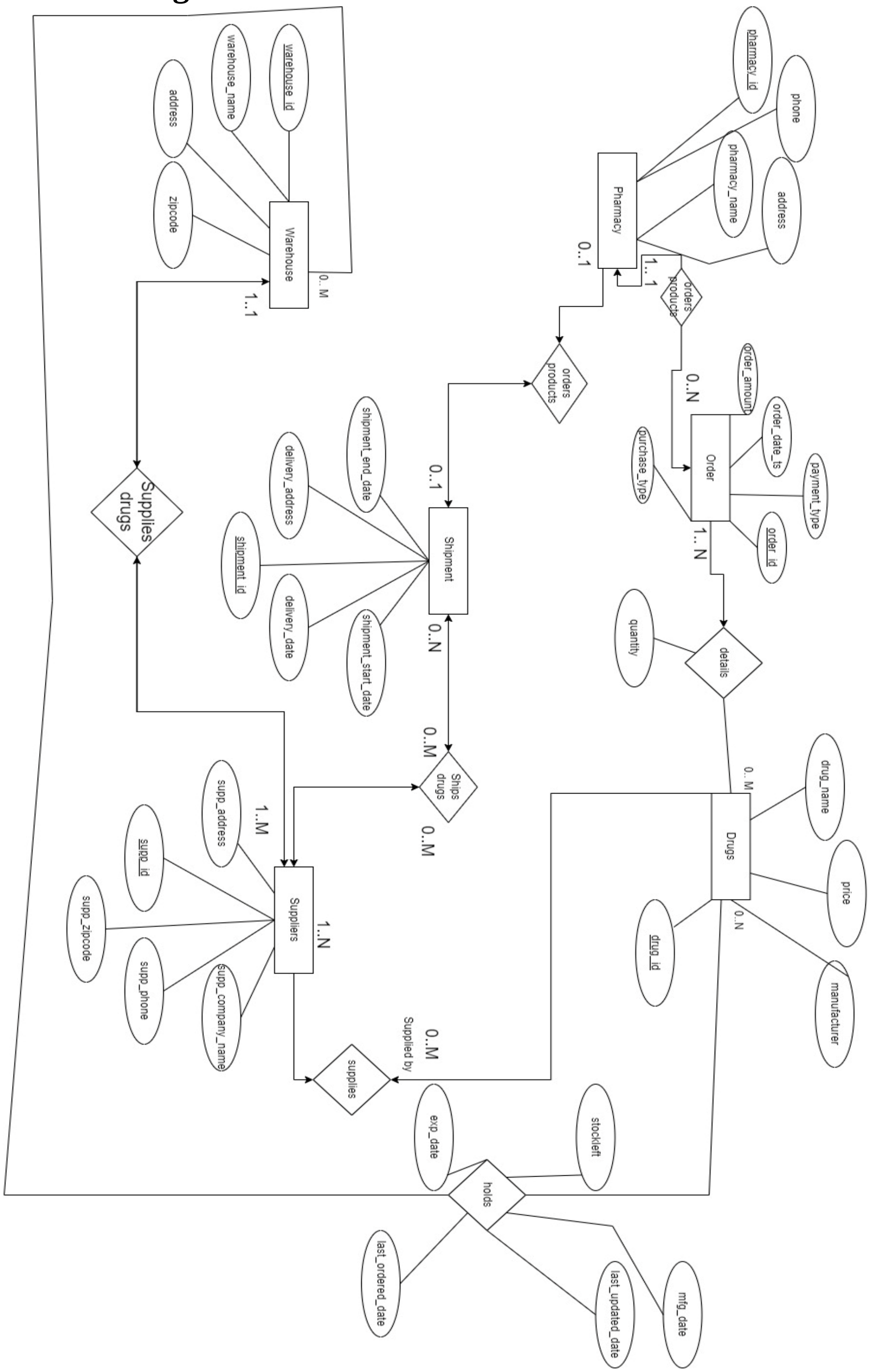
Inventory management encompasses the principles and processes involved in running the day-to-day operations of a retailer. At a high level, this includes receiving and organizing inventory space, scheduling labor, managing inventory, and fulfilling orders. Zoom in closer and you'll see that effective business management involves optimizing and integrating each of those processes to ensure all aspects of a warehouse operation work together to increase productivity and keep costs low.

In our project, we are going to track the movements of the drugs on a basis of day to day from warehouse to seller and to customer. We are going to record their Drug ID, Drug Name, Warehouse ID, expiry date, date of purchase, and how many items are left in the store. From the warehouse supplier, we are going to record the Warehouse ID, Warehouse Name, Drug ID, Count of the packs, expiry date, and date of arrival.

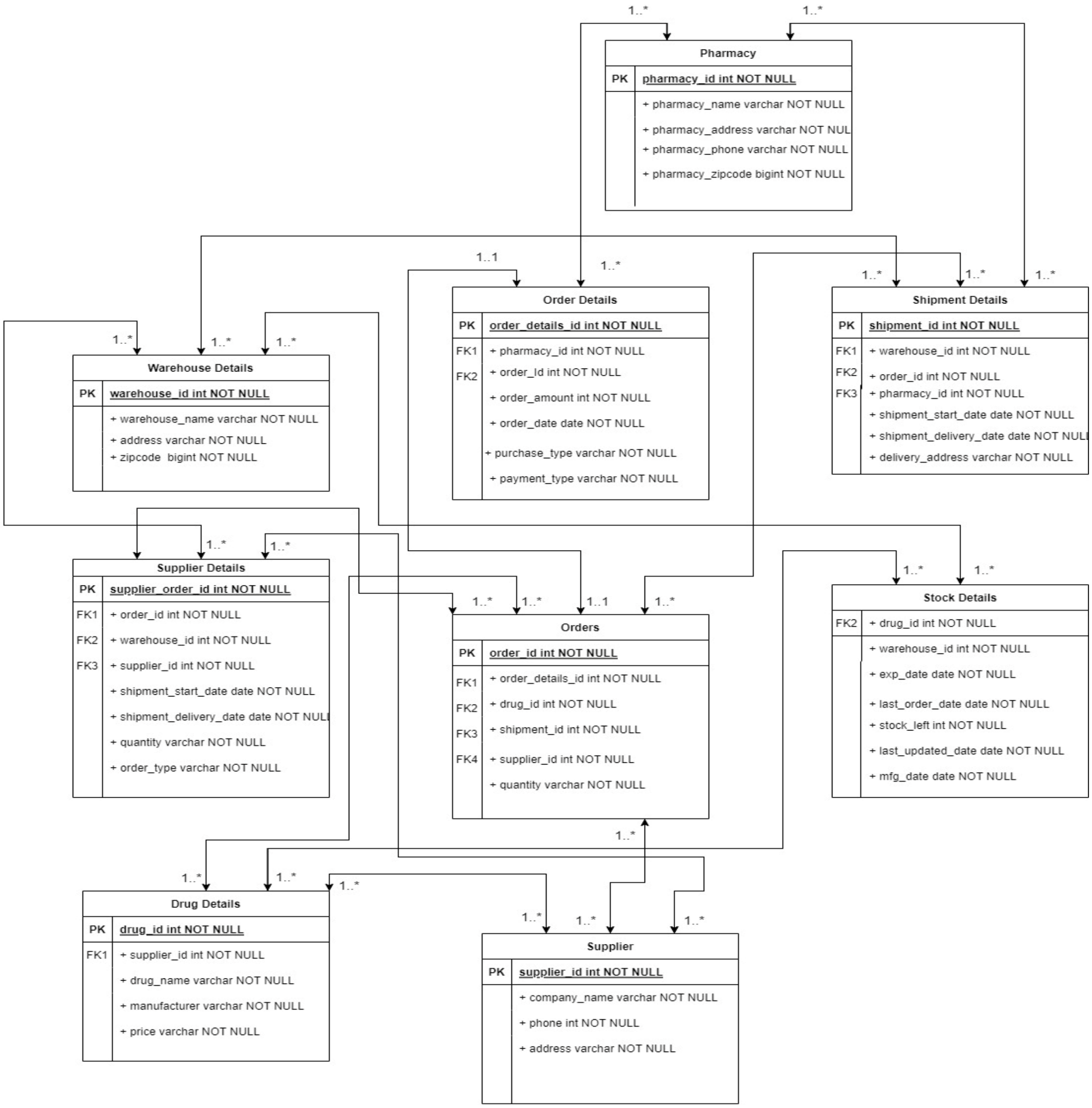
The process of the supply chain is recorded, the retail store needs to record the supplier's ID, name, address, and city. For products, the company needs to know their ID, name, type, and available quantity. At the same time, the purchase price and delivery period of the products also should be recorded. Each time when a customer purchases a product from the store, they will generate an order number and corresponding product type, order time, quantity, and product price. In the background, it needs to verify if the stock of that product is less than the threshold, and it should be notified to the respective supplier and place order for the same.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram


1..*
1..*
Pharmacy
PK
pharmacy_id int NOT NULL
+ pharmacy_name varchar NOT NULL
+ pharmacy_address varchar NOT NUL
+ pharmacy_phone varchar NOT NULL
+ pharmacy_zipcode bigint NOT NULL
1..1
1..*
1..*
1..*
1..*
Order Details
PK
order_details_id int NOT NULL
FK1
FK2
+ pharmacy_id int NOT NULL
+ order_Id int NOT NULL
+ order_amount int NOT NULL
+ order_date date NOT NULL
+ purchase_type varchar NOT NULL
+ payment_type varchar NOT NULL
Shipment Details
PK
shipment_id int NOT NULL
FK1
FK2
FK3
+ warehouse_id int NOT NULL
+ order_id int NOT NULL
+ pharmacy_id int NOT NULL
+ shipment_start_date date NOT NULL
+ shipment_delivery_date date NOT NULL
+ delivery_address varchar NOT NULL
1..*
1..*
1..*
Warehouse Details
PK
warehouse_id int NOT NULL
+ warehouse_name varchar NOT NULL
+ address varchar NOT NULL
+ zipcode bigint NOT NULL
1..*
1..*
Supplier Details
PK
supplier_order_id int NOT NULL
FK1
FK2
FK3
+ order_id int NOT NULL
+ warehouse_id int NOT NULL
+ supplier_id int NOT NULL
+ shipment_start_date date NOT NULL
+ shipment_delivery_date date NOT NULL
+ quantity varchar NOT NULL
+ order_type varchar NOT NULL
1..*
1..*
1..1
1..*
Orders
PK
order_id int NOT NULL
FK1
FK2
FK3
FK4
+ order_details_id int NOT NULL
+ drug_id int NOT NULL
+ shipment_id int NOT NULL
+ supplier_id int NOT NULL
+ quantity varchar NOT NULL
1..*
1..*
Stock Details
FK2
+ drug_id int NOT NULL
+ warehouse_id int NOT NULL
+ exp_date date NOT NULL
+ last_order_date date NOT NULL
+ stock_left int NOT NULL
+ last_updated_date date NOT NULL
+ mfg_date date NOT NULL
1..*
1..*
1..*
1..*
Drug Details
PK
drug_id int NOT NULL
FK1
+ supplier_id int NOT NULL
+ drug_name varchar NOT NULL
+ manufacturer varchar NOT NULL
+ price varchar NOT NULL
1..*
1..*
1..*
1..*
Supplier
PK
supplier_id int NOT NULL
+ company_name varchar NOT NULL
+ phone int NOT NULL
+ address varchar NOT NULL

III. Mapping Conceptual Model to Relational Model

Primary Key: Underlined with blue color
Foreign Key: Dotted lines with green color

Entity -1:

orders → (order_id, pharmacy_id, shipment_id, order_date, payment_type, total_amount)

Description:
order_id → Primary Key
pharmacy_id, shipment_id → Foreign Key
Orders entity contains whole information of a single order which is linked with pharmacy.
pharmacy_id → It is a foreign key in orders table, and it is a primary key in the drugs table, we join these tables with the drug_id as a reference.
shipment_id → It is a foreign key in orders table, and it is a primary key in the shipment table, we join these tables with the shipment_id as a reference.

Entity-2:

order_details → (order_id, drug_id, quantiy)

Description:
order_id + drug_id → Primary Key
order_id, drug_id → Foreign Key
Order details entity contains the information of which drugs has ordered by pharmacy_id, and the total amount and payment type data in this table.
order_id → It is a foreign key in the orders details table and a primary key in the orders table which gives us a relation between both these tables.
drug_id → It is a foreign key in the order details table and a primary key in the pharmacy table

Entity-3:

stock_details → (stock_id, drug_id, stock_left, last_ordered_date, last_updated_date, warehouse_id, mfg_date, exp_date)

Description:
stock_id → Primary Key
drug_id, warehouse_id → Foreign Key
Stocks entity contains all the stock information of all the pharmacies with respect to each drug, and maintains the minimum stocks in each pharmacy store.
drug_id → It is a foreign key in stock_details table, and it is a primary key in the drugs table, we join these tables with the drug_id as a reference

warehouse_id → It is a foreign key in stock_details table, and it is a primary key in the warehouse table, we join these tables with the warehouse_id as a reference

Entity-4:

shipment_details → (shipment_id, warehouse_id, shipment_start_date, order_id, shipment_end_date)

Description:
shipment_id → Primary Key
warehouse_id, order_id → Foreign Key
Shipment entity contains all the shipment information like start and end date.
order_id → It is a foreign key in the shipment table and a primary key in the orders table which gives us a relation between both these tables.
warehouse_id → It is a foreign key in the shipment table and a primary key in the warehouse table which gives us a relation between both these tables.

Entity-5:

suppliers → (supplier_id, company_name, phone_no, address)

Description:
supplier_id → Primary Key
Here suppliers are a master tables which holds the information about all the suppliers

Entity-6:

drugs → (drug_id, drug_name, manufacturer, mrp_price, supplier_id)

Description:
drug_id → Primary Key
supplier_id → Foreign Key
Drugs entity contains all the information of the different drugs and stores the main info of it. It also has supplier_id in this table, by this we can know which supplier has supplied that drug.
supplier_id → It is a foreign key in the drugs table and a primary key in the supplier table which gives us a relation between both these tables. It refers to the which supplier is supplying which drugs

Entity-7:

warehouse_details → (warehouse_id, warehouse_name, address, zipcode)

Description:
warehouse_id → Primary Key
Warehouse entity contains all the details of the warehouse information like address, location and zipcode.

Entity-8:

pharmacy → (pharmacy_id, pharmacy_name, address, phone_no)

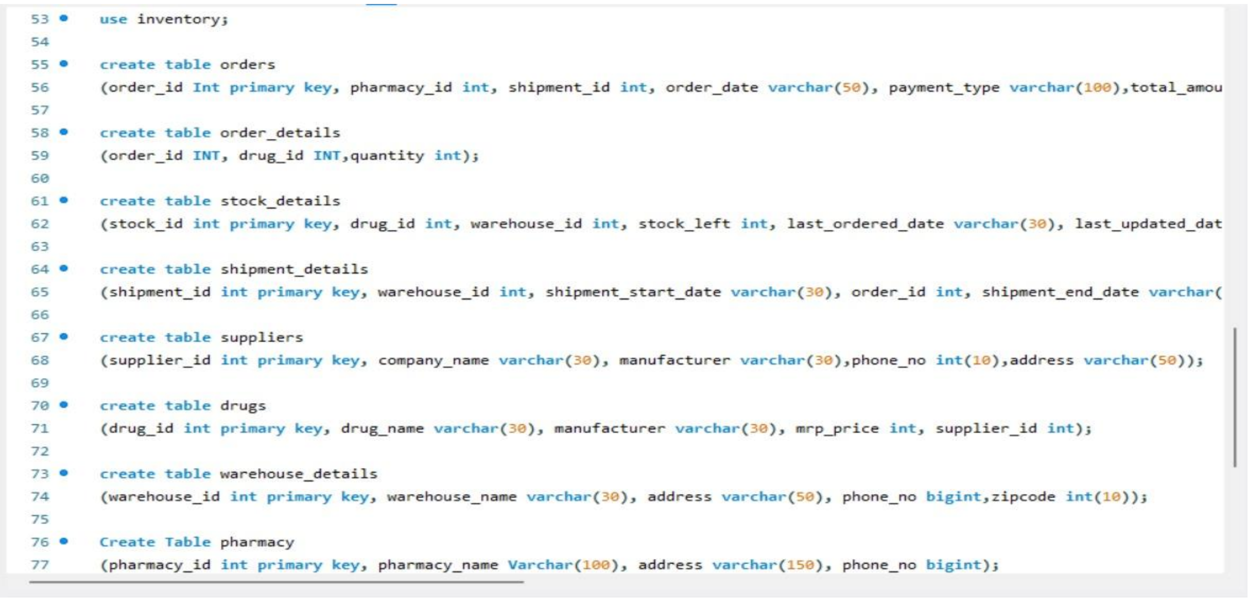
Description:
pharmacy_id → Primary Key
Pharmacy entity contains all the details of the pharmacies with their respective address and zipcode.

IV. Implementation of Relation Model via MySQL and NoSQL

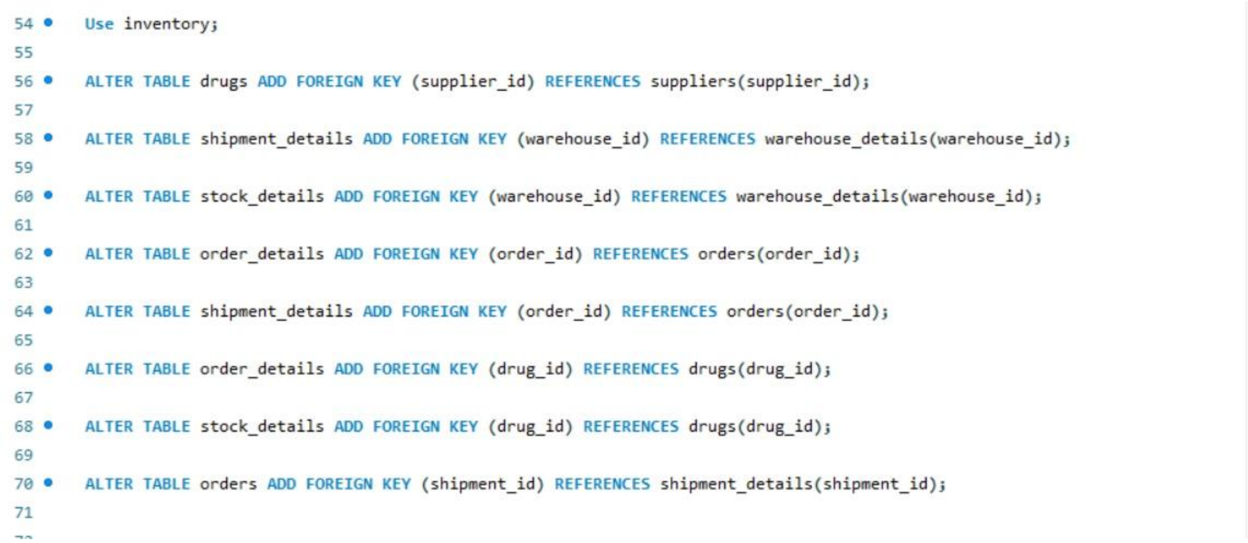
MySQL Implementation:

The database Inventory was created in MySQL with the following tables:

> Orders, order_details, stock_details, shipment_details, suppliers, drugs, warehouse_details, pharmacy.



```
use inventory;

create table orders
(order_id Int primary key, pharmacy_id int, shipment_id int, order_date varchar(50), payment_type varchar(100),total_amou

create table order_details
(order_id INT, drug_id INT,quantity int);

create table stock_details
(stock_id int primary key, drug_id int, warehouse_id int, stock_left int, last_ordered_date varchar(30), last_updated_dat

create table shipment_details
(shipment_id int primary key, warehouse_id int, shipment_start_date varchar(30), order_id int, shipment_end_date varchar(

create table suppliers
(supplier_id int primary key, company_name varchar(30), manufacturer varchar(30),phone_no int(10),address varchar(50));

create table drugs
(drug_id int primary key, drug_name varchar(30), manufacturer varchar(30), mrp_price int, supplier_id int);

create table warehouse_details
(warehouse_id int primary key, warehouse_name varchar(30), address varchar(50), phone_no bigint,zipcode int(10));

Create Table pharmacy
(pharmacy_id int primary key, pharmacy_name Varchar(100), address varchar(150), phone_no bigint);
```

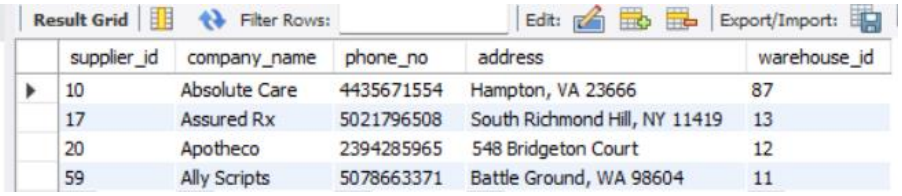


```
Use inventory;

ALTER TABLE drugs ADD FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id);

ALTER TABLE shipment_details ADD FOREIGN KEY (warehouse_id) REFERENCES warehouse_details(warehouse_id);

ALTER TABLE stock_details ADD FOREIGN KEY (warehouse_id) REFERENCES warehouse_details(warehouse_id);

ALTER TABLE order_details ADD FOREIGN KEY (order_id) REFERENCES orders(order_id);

ALTER TABLE shipment_details ADD FOREIGN KEY (order_id) REFERENCES orders(order_id);

ALTER TABLE order_details ADD FOREIGN KEY (drug_id) REFERENCES drugs(drug_id);

ALTER TABLE stock_details ADD FOREIGN KEY (drug_id) REFERENCES drugs(drug_id);

ALTER TABLE orders ADD FOREIGN KEY (shipment_id) REFERENCES shipment_details(shipment_id);
```

Later the following queries were performed on the databases:

Q1 Finding all the suppliers whose company name starts with 'A'

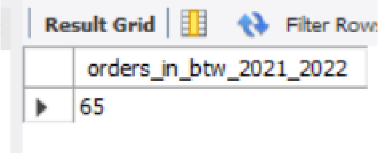
```
SELECT * FROM suppliers
WHERE company_name
LIKE 'A%';
```



supplier_id	company_name	phone_no	address	warehouse_id
10	Absolute Care	4435671554	Hampton, VA 23666	87
17	Assured Rx	5021796508	South Richmond Hill, NY 11419	13
20	Apotheco	2394285965	548 Bridgeton Court	12
59	Ally Scripts	5078663371	Battle Ground, WA 98604	11

Q2 Querying the orders table to find the count of orders placed in the year 2021 and 2022

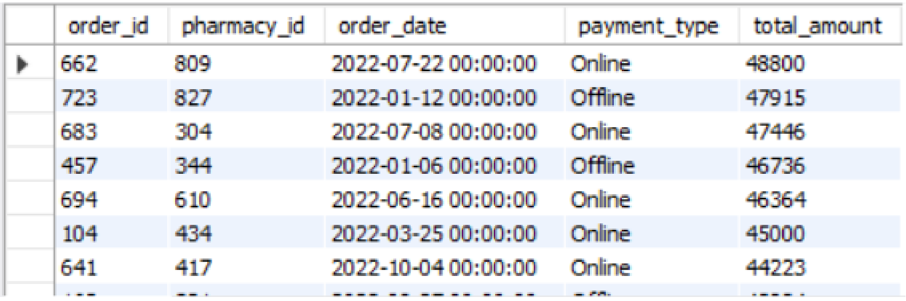
```
SELECT count(*) as orders_in_btw_2021_2022 FROM orders
WHERE order_date between '2021-01-01' AND '2022-12-31';
```



orders_in_btw_2021_2022
65

Q3 Querying the orders table to get the order details which are more than 30000 and orders placed in 2022

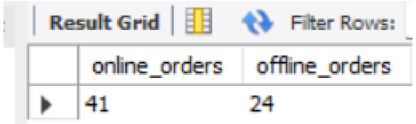
```
SELECT * FROM orders
WHERE total_amount>30000
AND order_date>'2022-01-01'
ORDER BY total_amount
desc;
```



order_id	pharmacy_id	order_date	payment_type	total_amount
662	809	2022-07-22 00:00:00	Online	48800
723	827	2022-01-12 00:00:00	Offline	47915
683	304	2022-07-08 00:00:00	Online	47446
457	344	2022-01-06 00:00:00	Offline	46736
694	610	2022-06-16 00:00:00	Online	46364
104	434	2022-03-25 00:00:00	Online	45000
641	417	2022-10-04 00:00:00	Online	44223

Q5 Querying orders table to find the count of online and offline orders

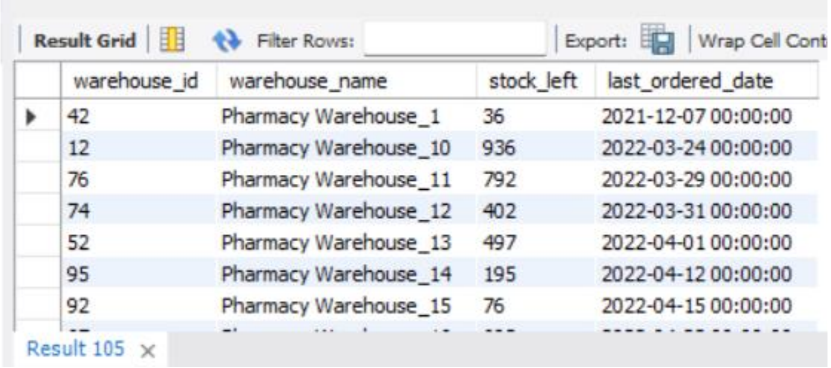
```
SELECT count(case when payment_type ='Online' then 1 end) as
online_orders, count(case when payment_type='Offline' then 1
end) as offline_orders FROM orders;
```



online_orders	offline_orders
41	24

Q6 Querying the warehouse and stocks table to get the warehousename and stocks left in warehouse by join operations

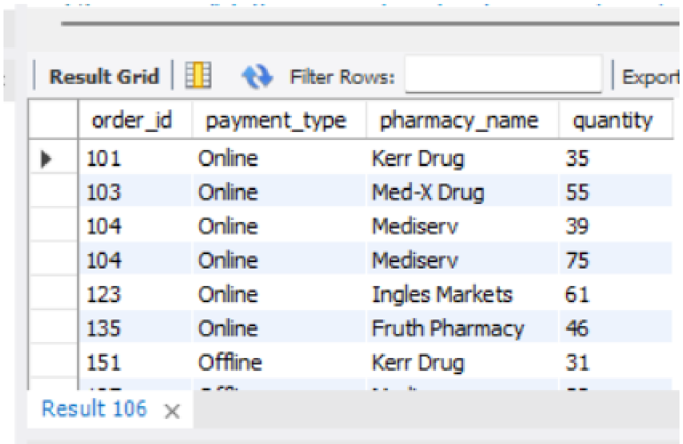
```
SELECT wd.warehouse_id,
wd.warehouse_name, sd.stock_left,
sd.last_ordered_date
FROM warehouse_details wd join
stock_details sd
ON wd.warehouse_id=sd.warehouse_id
ORDER BY wd.warehouse_name;
```



warehouse_id	warehouse_name	stock_left	last_ordered_date
42	Pharmacy Warehouse_1	36	2021-12-07 00:00:00
12	Pharmacy Warehouse_10	936	2022-03-24 00:00:00
76	Pharmacy Warehouse_11	792	2022-03-29 00:00:00
74	Pharmacy Warehouse_12	402	2022-03-31 00:00:00
52	Pharmacy Warehouse_13	497	2022-04-01 00:00:00
95	Pharmacy Warehouse_14	195	2022-04-12 00:00:00
92	Pharmacy Warehouse_15	76	2022-04-15 00:00:00

Q7 Querying for finding order,pharmacy, payment type for each order that exceeds order quantity of 30

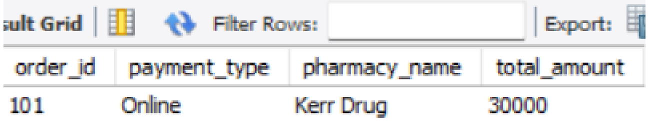
SELECT o.order_id, o.payment_type, p.pharmacy_name, od.quantity
FROM orders o, pharmacy p, order_details od
WHERE o.order_id=od.order_id AND o.pharmacy_id = p.pharmacy_id AND od.quantity >=30;



order_id	payment_type	pharmacy_name	quantity
101	Online	Kerr Drug	35
103	Online	Med-X Drug	55
104	Online	Mediserv	39
104	Online	Mediserv	75
123	Online	Ingles Markets	61
135	Online	Fruth Pharmacy	46
151	Offline	Kerr Drug	31

Q8 Querying for finding order, pharmacy, payment type which has the highest order amount in the month of sept 2022

SELECT o.order_id, o.payment_type, p.pharmacy_name, o.total_amount
FROM orders o, pharmacy p WHERE o.pharmacy_id = p.pharmacy_id
AND o.order_id IN (SELECT order_id FROM orders
WHERE order_date between '2022-09-01 00:00:00' AND '2022-09-30 00:00:00')
ORDER BY o.total_amount DESC LIMIT 1;



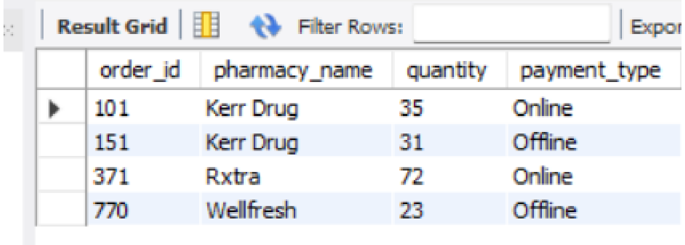
order_id	payment_type	pharmacy_name	total_amount
101	Online	Kerr Drug	30000

Q9 Query to retrieve the names of the orders, pharmacy name who have the highest quantity ordered by pharmacy within 2022 (Using Any)

SELECT o.order_id,p.pharmacy_name ,od.quantity,o.payment_type

FROM orders o join order_details od on o.order_id=od.order_id join pharmacy p on o.pharmacy_id=p.pharmacy_id

WHERE p.pharmacy_id IN (SELECT pharmacy_id FROM pharmacy WHERE pharmacy_name in ('Rxtra','Wellfresh','Pharmanic','Kerr Drug')

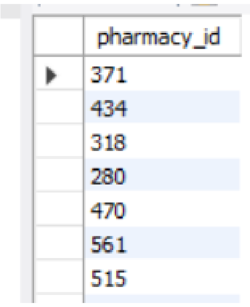


order_id	pharmacy_name	quantity	payment_type
101	Kerr Drug	35	Online
151	Kerr Drug	31	Offline
371	Rxtra	72	Online
770	Wellfresh	23	Offline

AND od.quantity> Any (SELECT quantity FROM order_details group by order_id having quantity = max(quantity)));

Q10 Correlated Query to retrieve the order_id of all orders with at least one supplied drugs to pharmacies who placed an Offline Orders

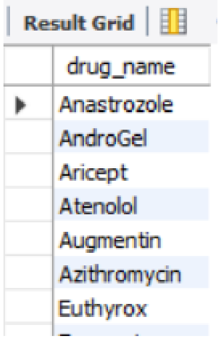
SELECT o.pharmacy_id FROM orders o WHERE 1 <= (SELECT COUNT(*) FROM pharmacy p WHERE p.pharmacy_id = o.pharmacy_id AND o.payment_type='Offline');



pharmacy_id
371
434
318
280
470
561
515

Q11 Query for retrieving the names of thhe drugnames which have stock more than 800 (Using EXISTS)

SELECT drug_name FROM drugs WHERE EXISTS (SELECT drug_id FROM stock_details WHERE stock_details.drug_id=drugs.drug_id AND stock_details.stock_left>800) ORDER BY drug_name;



drug_name
Anastrozole
AndroGel
Aricept
Atenolol
Augmentin
Azithromycin
Euthyrox

Q12 Query to update last updated date for that stock details to today when new order arrives.

CREATE trigger `IsStockAvailable` BEFORE insert on order_details
FOR EACH ROW UPDATE stock_details
SET last_updated_date=curdate() WHERE drug_id= new.drug_id;

Inserting data into order_details to check triggers

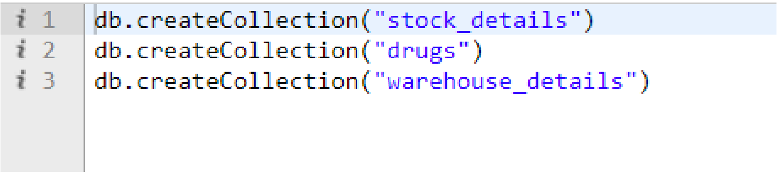
INSERT INTO order_details values (112, 30,20);
SELECT * FROM stock_details WHERE drug_id=30;

Q13 Query for Viewing the expired stock and to get records where stock_left is greater than 50 from the View Expired Stock

Create view `ExpiredStock` as SELECT * FROM Stock_details WHERE exp_date<curdate();
SELECT * FROM ExpiredStock WHERE stock_left>50;

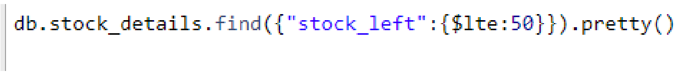
NoSQL Implementation:

Create tables (stock_details, drugs and warehouse_details) and insert values into them:

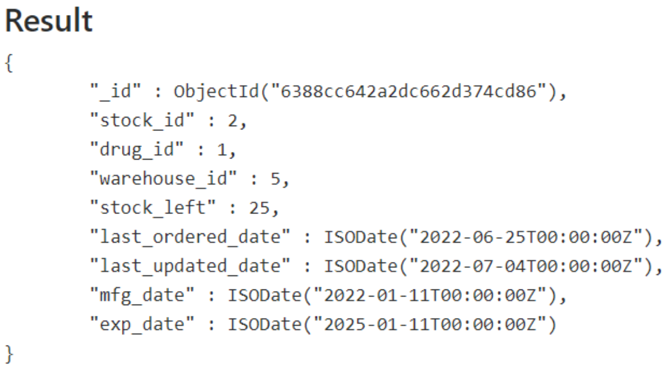


```
db.createCollection("stock_details")
db.createCollection("drugs")
db.createCollection("warehouse_details")
```

Q1 Find products that are less than 50 in quantity



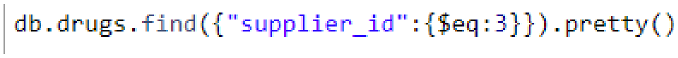
```
db.stock_details.find({"stock_left":{$lte:50}}).pretty()
```



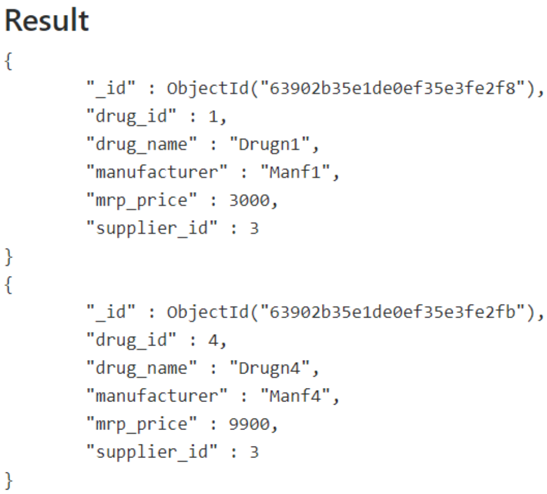
Result

```
{
        "_id" : ObjectId("6388cc642a2dc662d374cd86"),
        "stock_id" : 2,
        "drug_id" : 1,
        "warehouse_id" : 5,
        "stock_left" : 25,
        "last_ordered_date" : ISODate("2022-06-25T00:00:00Z"),
        "last_updated_date" : ISODate("2022-07-04T00:00:00Z"),
        "mfg_date" : ISODate("2022-01-11T00:00:00Z"),
        "exp_date" : ISODate("2025-01-11T00:00:00Z")
}
```

Q2 Drugs manufactured by Supplier no 3



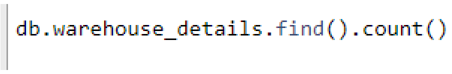
```
db.drugs.find({"supplier_id":{$eq:3}}).pretty()
```



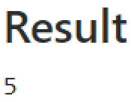
Result

```
{
        "_id" : ObjectId("63902b35e1de0ef35e3fe2f8"),
        "drug_id" : 1,
        "drug_name" : "Drugn1",
        "manufacturer" : "Manf1",
        "mrp_price" : 3000,
        "supplier_id" : 3
}
{
        "_id" : ObjectId("63902b35e1de0ef35e3fe2fb"),
        "drug_id" : 4,
        "drug_name" : "Drugn4",
        "manufacturer" : "Manf4",
        "mrp_price" : 9900,
        "supplier_id" : 3
}
```

Q3 Find number of warehouses owned



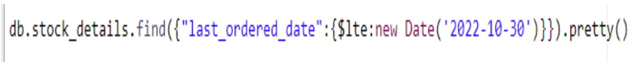
```
db.warehouse_details.find().count()
```



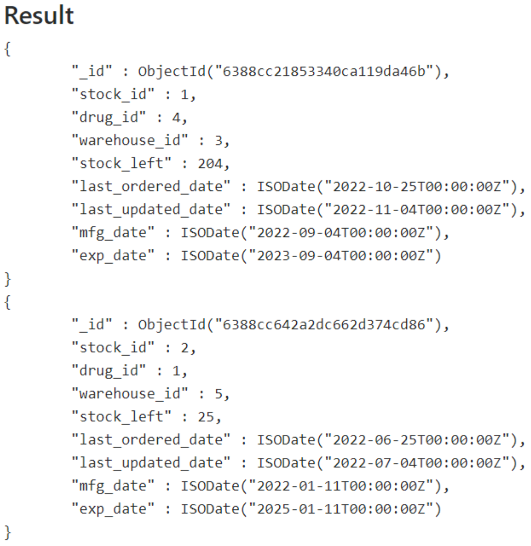
Result

```
5
```

Q4 Products ordered last before 30th October 2022



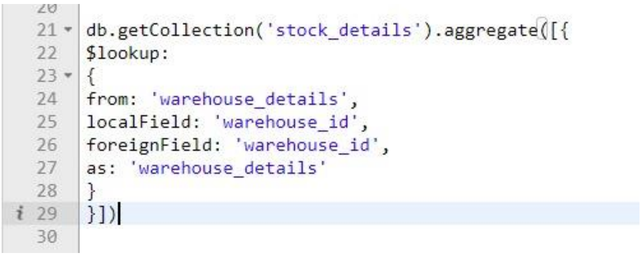
```
db.stock_details.find({"last_ordered_date":{$lte:new Date('2022-10-30')}}).pretty()
```



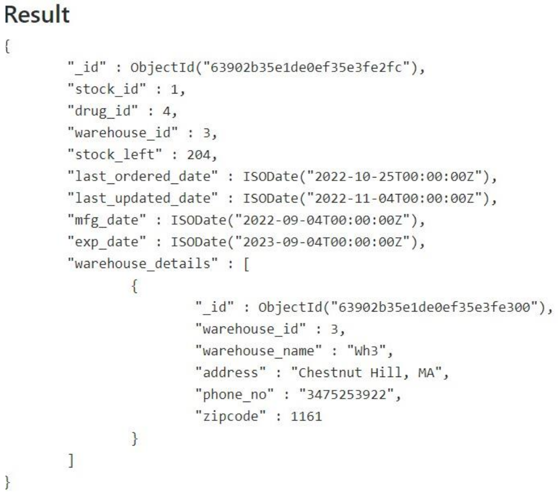
Result

```
{
        "_id" : ObjectId("6388cc21853340ca119da46b"),
        "stock_id" : 1,
        "drug_id" : 4,
        "warehouse_id" : 3,
        "stock_left" : 204,
        "last_ordered_date" : ISODate("2022-10-25T00:00:00Z"),
        "last_updated_date" : ISODate("2022-11-04T00:00:00Z"),
        "mfg_date" : ISODate("2022-09-04T00:00:00Z"),
        "exp_date" : ISODate("2023-09-04T00:00:00Z")
}
{
        "_id" : ObjectId("6388cc642a2dc662d374cd86"),
        "stock_id" : 2,
        "drug_id" : 1,
        "warehouse_id" : 5,
        "stock_left" : 25,
        "last_ordered_date" : ISODate("2022-06-25T00:00:00Z"),
        "last_updated_date" : ISODate("2022-07-04T00:00:00Z"),
        "mfg_date" : ISODate("2022-01-11T00:00:00Z"),
        "exp_date" : ISODate("2025-01-11T00:00:00Z")
}
```

Q5 Availability of drugs in warehouses

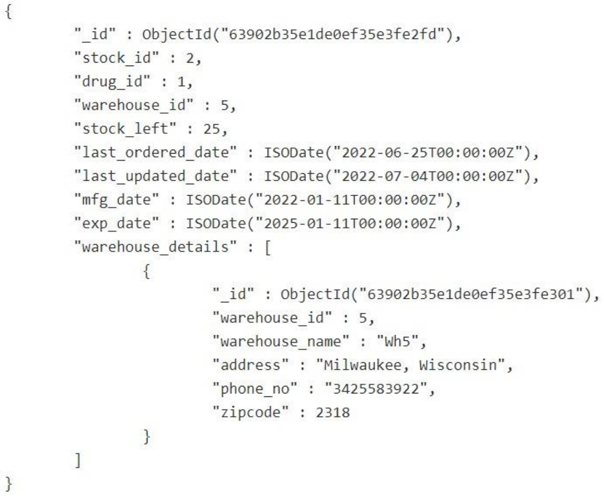


```
db.getCollection('stock_details').aggregate([{
$lookup:
{
from: 'warehouse_details',
localField: 'warehouse_id',
foreignField: 'warehouse_id',
as: 'warehouse_details'
}
}])
```



Result

```
{
        "_id" : ObjectId("63902b35e1de0ef35e3fe2fc"),
        "stock_id" : 1,
        "drug_id" : 4,
        "warehouse_id" : 3,
        "stock_left" : 204,
        "last_ordered_date" : ISODate("2022-10-25T00:00:00Z"),
        "last_updated_date" : ISODate("2022-11-04T00:00:00Z"),
        "mfg_date" : ISODate("2022-09-04T00:00:00Z"),
        "exp_date" : ISODate("2023-09-04T00:00:00Z"),
        "warehouse_details" : [
                {
                        "_id" : ObjectId("63902b35e1de0ef35e3fe300"),
                        "warehouse_id" : 3,
                        "warehouse_name" : "Wh3",
                        "address" : "Chestnut Hill, MA",
                        "phone_no" : "3475253922",
                        "zipcode" : 1161
                }
        ]
}
```

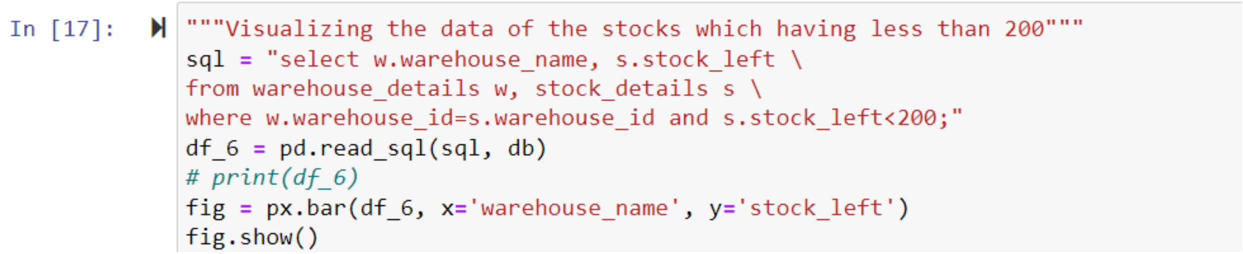


```
{
        "_id" : ObjectId("63902b35e1de0ef35e3fe2fd"),
        "stock_id" : 2,
        "drug_id" : 1,
        "warehouse_id" : 5,
        "stock_left" : 25,
        "last_ordered_date" : ISODate("2022-06-25T00:00:00Z"),
        "last_updated_date" : ISODate("2022-07-04T00:00:00Z"),
        "mfg_date" : ISODate("2022-01-11T00:00:00Z"),
        "exp_date" : ISODate("2025-01-11T00:00:00Z"),
        "warehouse_details" : [
                {
                        "_id" : ObjectId("63902b35e1de0ef35e3fe301"),
                        "warehouse_id" : 5,
                        "warehouse_name" : "Wh5",
                        "address" : "Milwaukee, Wisconsin",
                        "phone_no" : "3425583922",
                        "zipcode" : 2318
                }
        ]
}
```

V. Database Access via Python

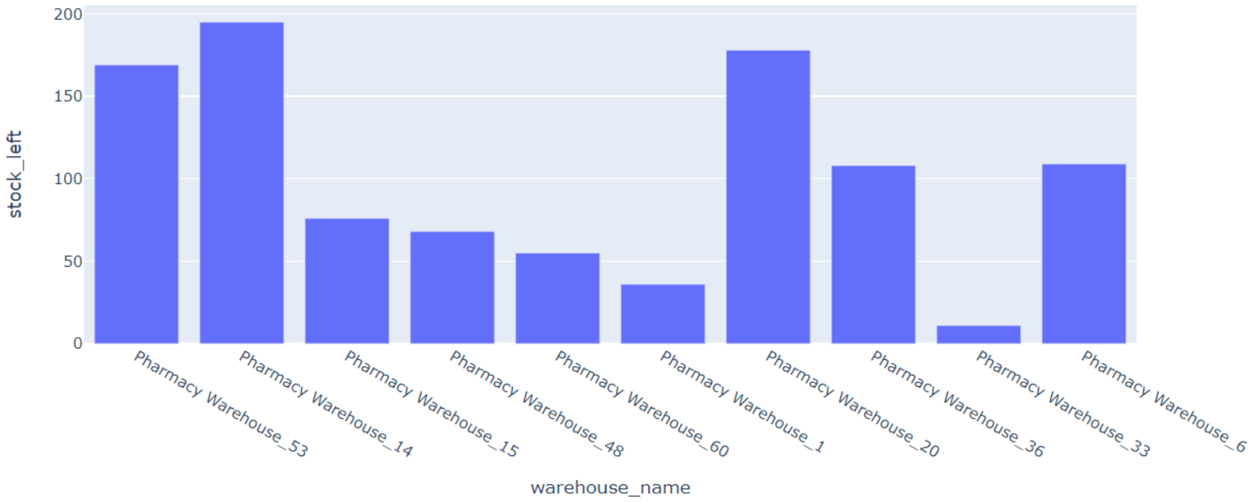
The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using MySQL. Connector, followed by cursor.excecute to run and fetchall FROM query, followed by converting the list into a dataframe using pandas library and using matplotlib to plot the graphs for the analytics.

Q1 Visualizing the data of the stocks which having less than 200

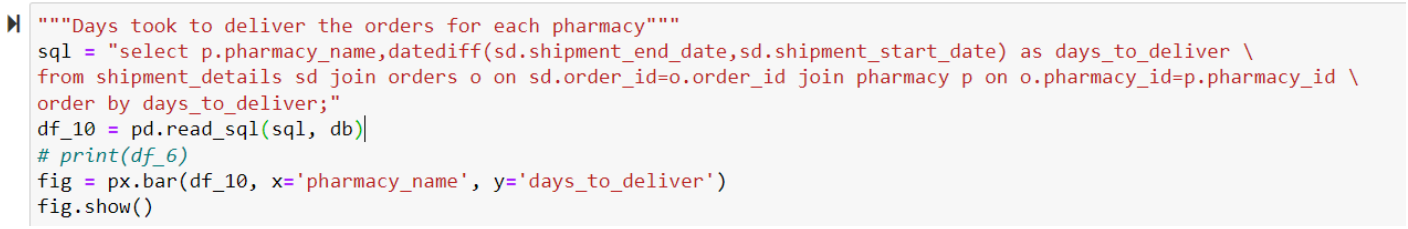


In [17]:

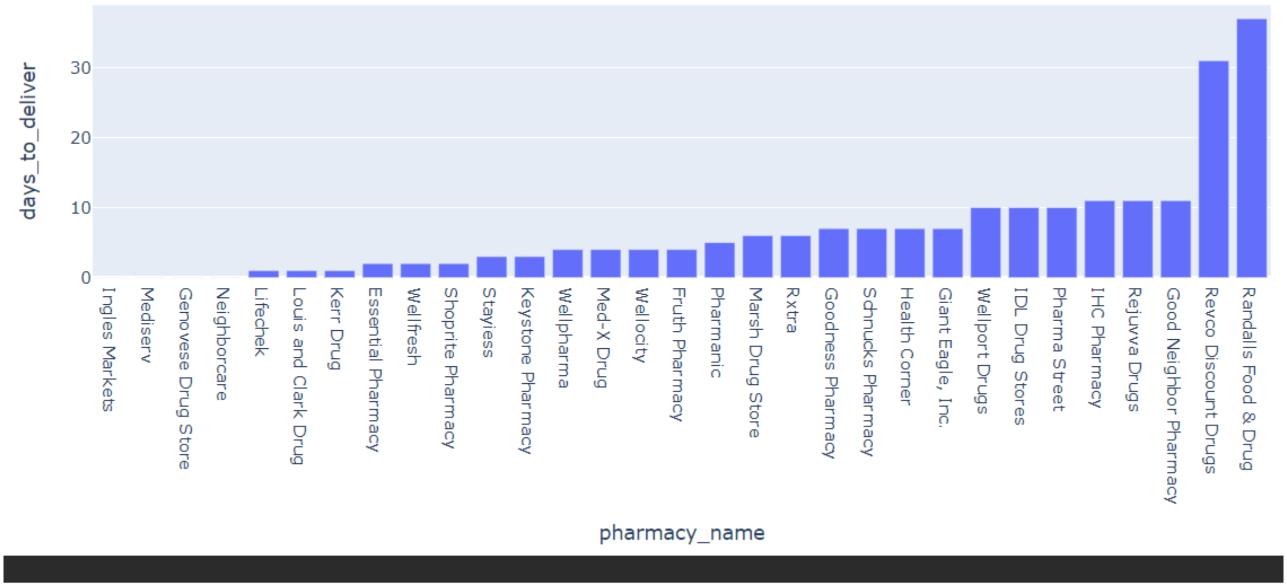
```
"""Visualizing the data of the stocks which having less than 200"""
sql = "select w.warehouse_name, s.stock_left \
from warehouse_details w, stock_details s \
where w.warehouse_id=s.warehouse_id and s.stock_left<200;"
df_6 = pd.read_sql(sql, db)
# print(df_6)
fig = px.bar(df_6, x='warehouse_name', y='stock_left')
fig.show()
```



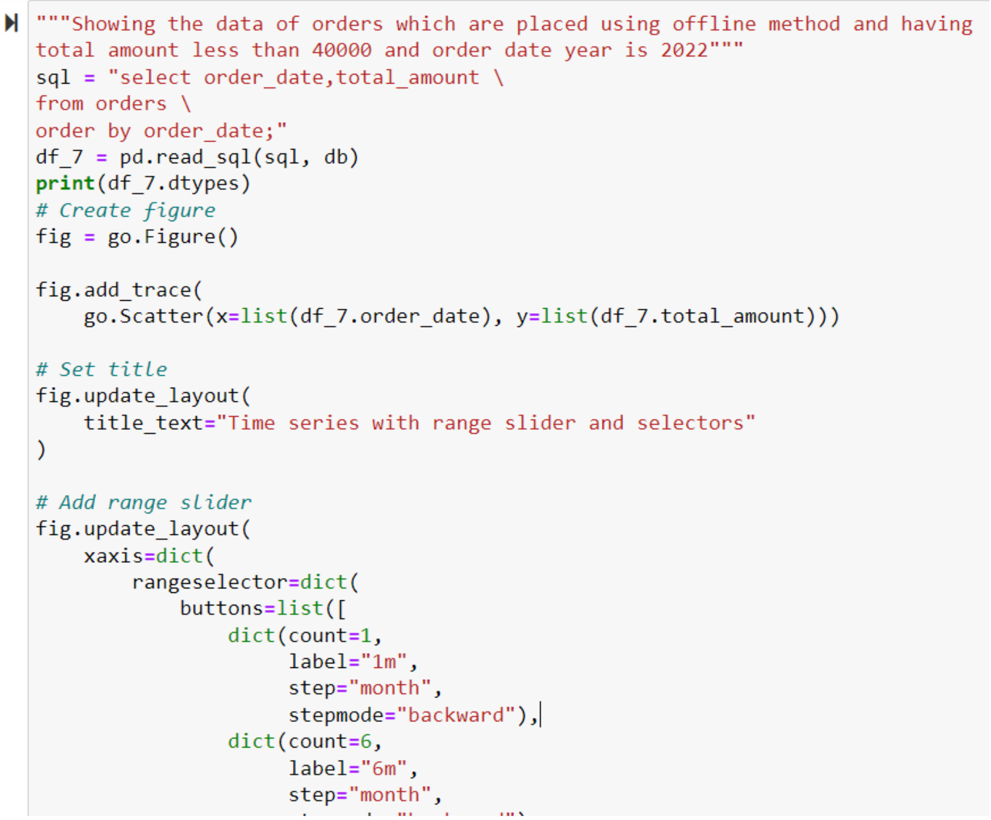
Q2 Days took to deliver the orders for each pharmacy



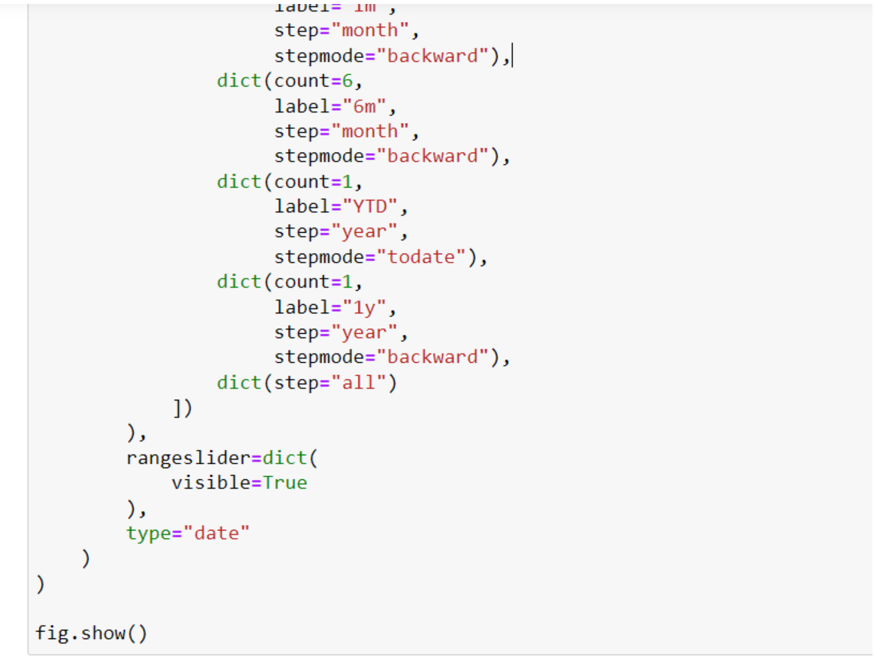
```
"""Days took to deliver the orders for each pharmacy"""
sql = "select p.pharmacy_name,datediff(sd.shipment_end_date,sd.shipment_start_date) as days_to_deliver \
from shipment_details sd join orders o on sd.order_id=o.order_id join pharmacy p on o.pharmacy_id=p.pharmacy_id \
order by days_to_deliver;"
df_10 = pd.read_sql(sql, db)
# print(df_6)
fig = px.bar(df_10, x='pharmacy_name', y='days_to_deliver')
fig.show()
```



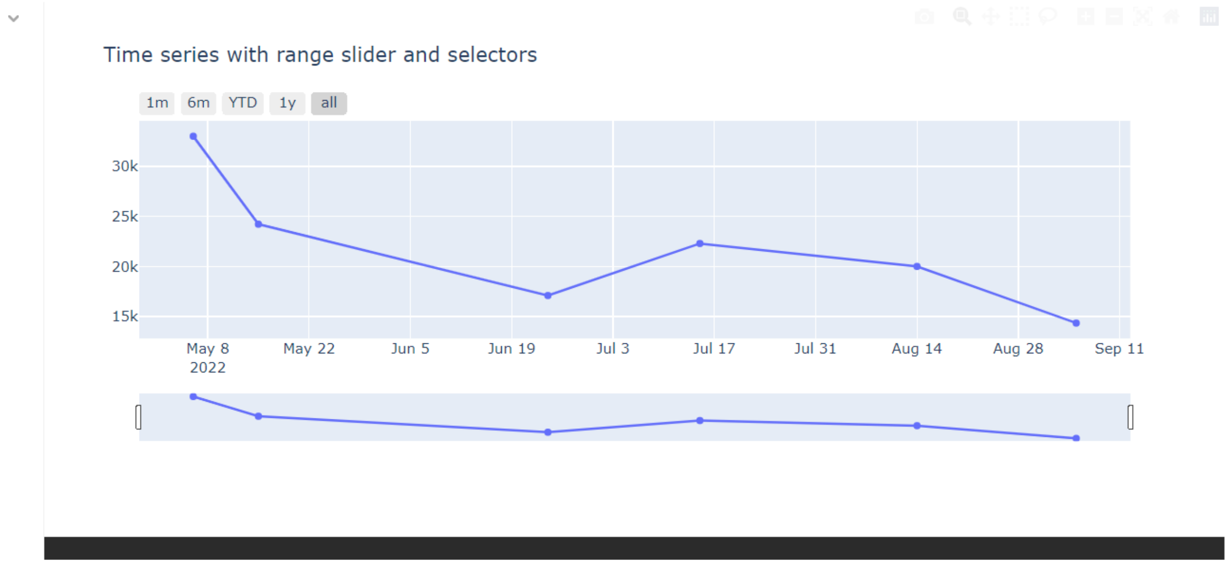
Q3 Showing the data of orders which are placed using offline method and having total amount less than 40000 and order date year is 2022



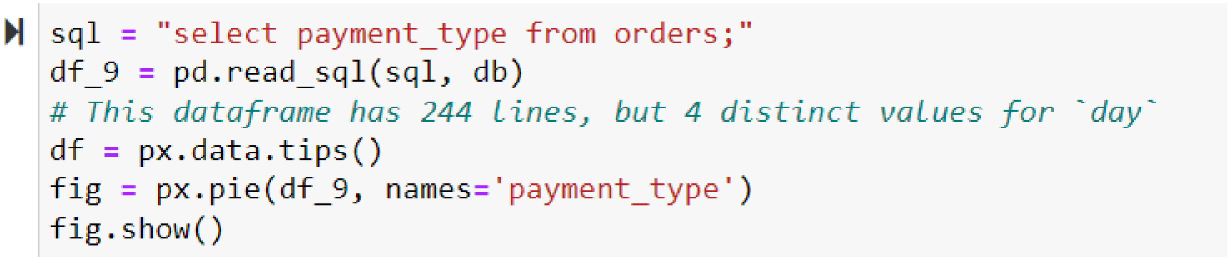
```
"""Showing the data of orders which are placed using offline method and having
total amount less than 40000 and order date year is 2022"""
sql = "select order_date,total_amount \
from orders \
order by order_date;"
df_7 = pd.read_sql(sql, db)
print(df_7.dtypes)
# Create figure
fig = go.Figure()

fig.add_trace(
    go.Scatter(x=list(df_7.order_date), y=list(df_7.total_amount)))

# Set title
fig.update_layout(
    title_text="Time series with range slider and selectors"
)

# Add range slider
fig.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1,
                     label="1m",
                     step="month",
                     stepmode="backward"),
                dict(count=6,
                     label="6m",
                     step="month",
```



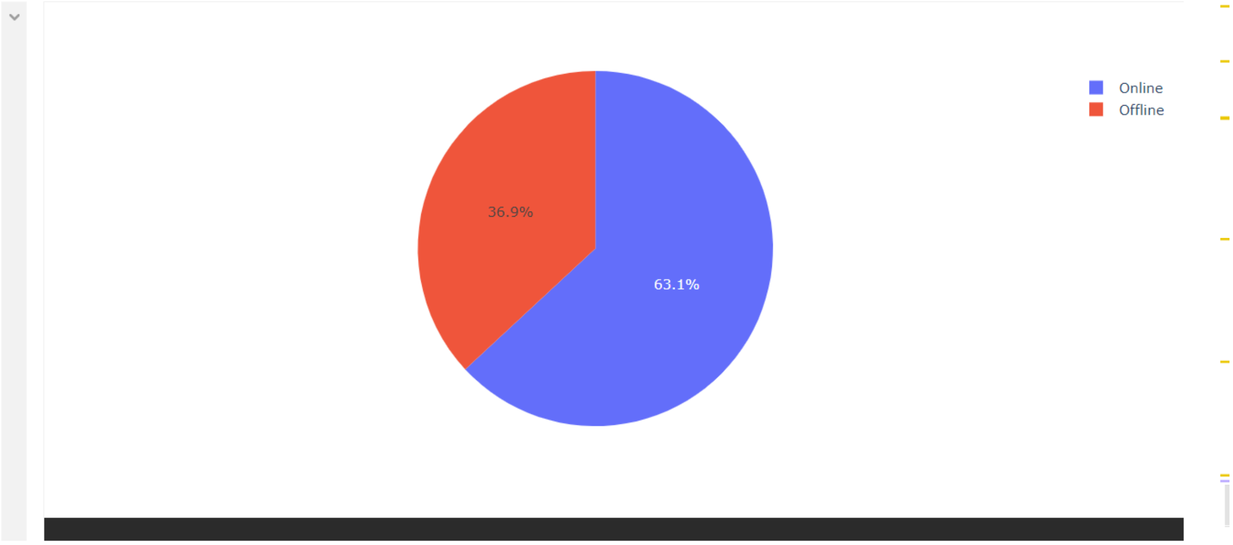
```
                     label="1m",
                     step="month",
                     stepmode="backward"),
                dict(count=6,
                     label="6m",
                     step="month",
                     stepmode="backward"),
                dict(count=1,
                     label="YTD",
                     step="year",
                     stepmode="todate"),
                dict(count=1,
                     label="1y",
                     step="year",
                     stepmode="backward"),
                dict(step="all")
            ])
        ),
        rangeslider=dict(
            visible=True
        ),
        type="date"
    )
)

fig.show()
```



Q4 Ratio of offline to online orders in total



```
sql = "select payment_type from orders;"
df_9 = pd.read_sql(sql, db)
# This dataframe has 244 lines, but 4 distinct values for `day`
df = px.data.tips()
fig = px.pie(df_9, names='payment_type')
fig.show()
```



VI. Summary and Recommendation

We have basically developed a Warehouse Management System offers visibility into a business' entire inventory and manages supply chain fulfillment operations from the distribution center to the store shelf. Also enables efficient routes for minimum cost to fulfill orders and have efficient management systems like to automatically orders more stock from supplier when it is about to finish.

Additional requirements we have tried to fulfill involve:

1. A warehouse can hold one to infinite orders to fulfill the inventory requirements. A supplier can supply to any number of stores in the city.
2. An inventory can sell one to infinite orders to the customers, the store can sell different drugs at a time.
3. We are going to have a master table that contains all the drug names and drug ids, these ids can be mapped to all the different tables.
4. Warehouse can contain products in bulk, which can be stored as Product Id, Product Names, quantity, and expiry date, this is used to handle the inventory orders.
5. Strengthening the relationships with the warehouse as well as customers
6. Ensuring resources are used optimally and without loss
7. Provide insight into inventory location and quantity using predefined rules for receiving, picking, and packing orders
8. Checking in and logging incoming items. Verify that you’re receiving the right quantity, in the right condition, at the right time.
9. Move items from the receiving dock to their correct storage locations.
10. Safely store and logically arrange inventory to enable fast and accurate picking collects the items needed to fulfill sales orders.
11. Prepare the picked items for shipment. They must be safely packed into the correct packaging with an accurate packing slip.
12. Send out the finalized sales orders, ensuring that they are on the right vehicle, at the right time, with the correct documentation, so customers receive their orders on time.