WALMART GROCERY DELIVERY SYSTEM

Names: Geethika Meka

Executive Summary:

In this project, we are designing a relational database for Walmart that users can utilize to move/deliver groceries from one place to another within the city, state, or country. A customer will create an account through an app, and the order and payment details will be recorded. Grocery items can either be regular or frozen. The destination and delivery information can be recorded, and users can choose between an express deliver (1-2 hours) and a standard delivery (depending on factors). After the customer places their order, their driver and vehicle details are recorded, along with the type of vehicle and destination details. The system will allow users to upload photos and leave reviews of damaged food items, and we will use NoSQL to implement that in the later stages.

- Each user is allowed to create only one account
- User can order multiple products at a time
- Multiple orders can be placed under a single account
- You can only place an order from one account at a time
- User is allowed to pay through one type of payment method
- Each destination can accept several orders
- It is possible for the drivers to deliver multiple orders to different locations
- Multiple vehicles can be driven by a single driver based on vehicle availability

The database created has been successful, and by connecting it to Python, the analytical and visualization capabilities are boundless, some of the basic ones have been shown in the study. These queries can be very helpful in tracking users and their payments, orders and deliveries and gathering insightful information about consumer behavior.

I. Introduction

On-demand grocery mobile apps prove to be beneficial for both the grocery stores as well as their customers. While the customers save time to purchase groceries, the grocery stores gain more customers through such apps. It's a win-win situation for both sides.

Grocery Delivery System is a company that helps in solving the transportation of goods no matter the size and weight, all are accepted.

In this project, we are designing a relational database for a grocery delivery company that users can utilize to move/deliver groceries from one place to another within the cities.

A customer will create an account through an app, and the order and payment details will be recorded

BUSINESS MODEL

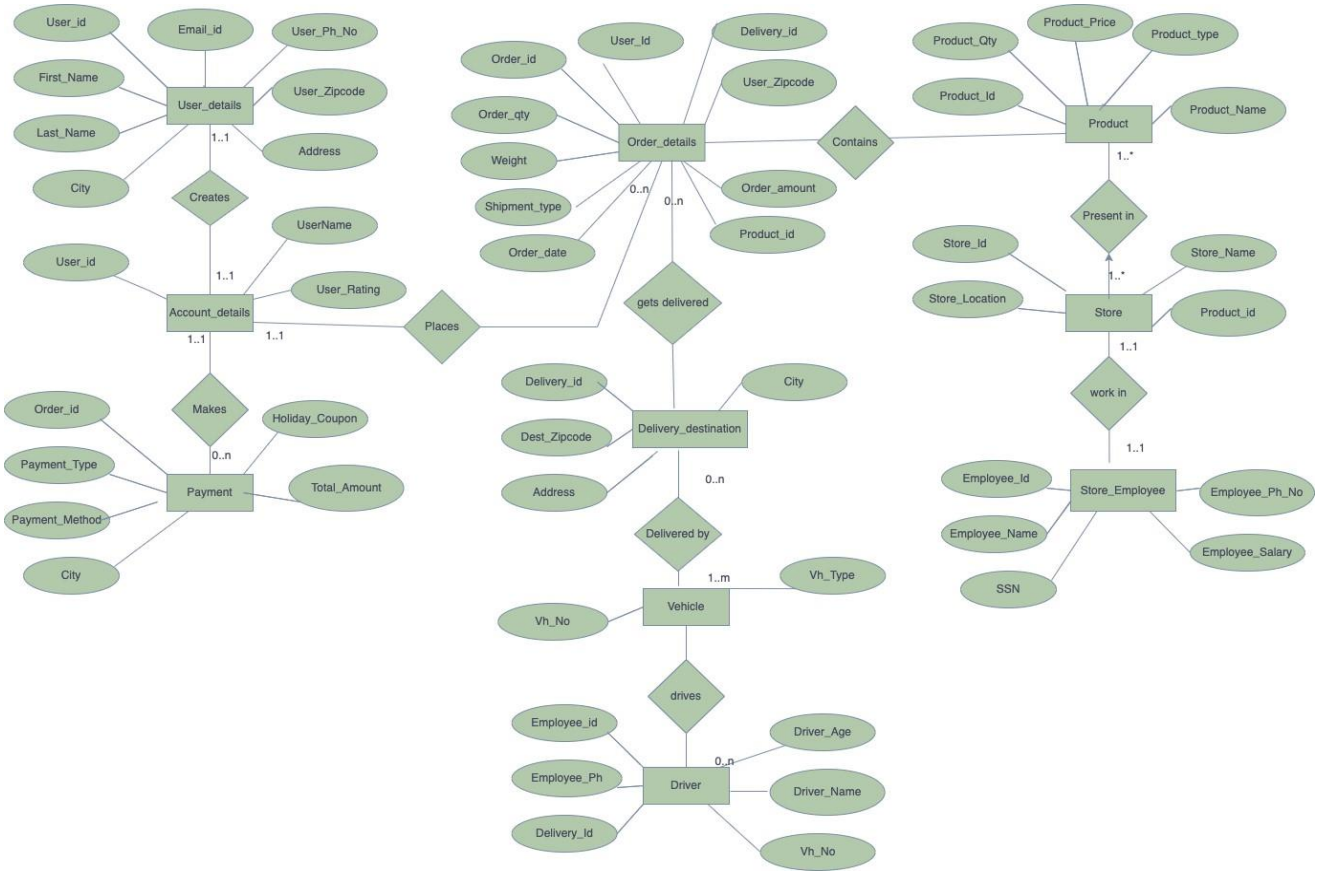
The app will be used to create an account and the order and payment details will be recorded. Various shipment types can be recorded in the orders.

The destination and delivery details can be recorded, and users can either request an express (1-2 days) or a standard delivery (depends on factors)

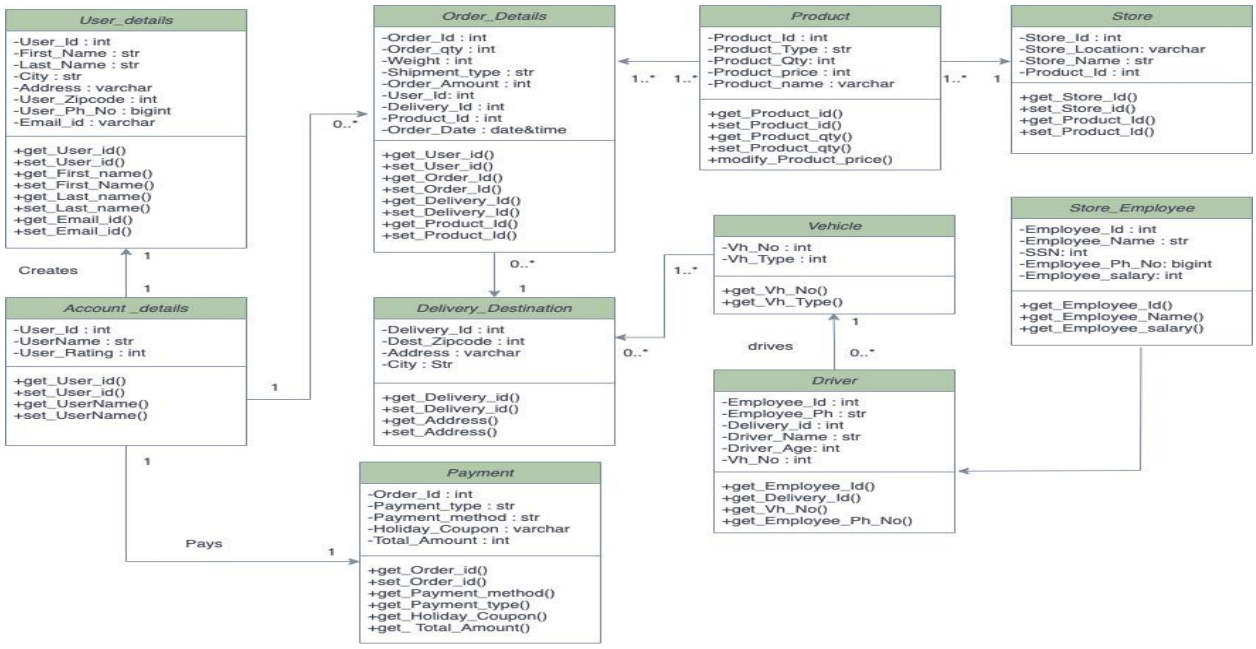
The driver and the vehicle details will be given once the order is placed, and the type of vehicle and the destination details are recorded.

Stores will be rated based on their product quality and quantity for business purposes. Payment method can be made by Cash when the payment type is opted for Cash On Delivery When the payment type is Online Payment, then will have to pay by credit/debit cards.

II. Conceptual Data Modeling



UML Diagram:



III. Mapping Conceptual Model to Relational Model

Primary Key- Underlined, Foreign Key- Bold

- User_DETAILS (User_ID, First_Name, Last_Name, City, Address, User_Zipcode, User_Ph_No, Email_Id)
- Account _Details (User_ID, Username, User_Rating)
- Delivery_Destination (Delivery_Id, Address, City)
- Product (Product_ID, Product_Type, Product_Qty, Product_Price, Product_Name)
- Order_Details (Order_Id, Order_Qty, Weight, Shipment_Type, Order_Amount, Order_Date, **User_ID**, **Delivery_Id**, **Product_ID**)
- Store (Store_ID, Store_Location, Store_Name, **Product_ID,** Employee_Salary)
- Payment (**Order_Id**, Payment_Type, Payment_Method, Holiday_Coupon, Total_Amount)
- Vehicle (Vh_No, Vh_Type)
- Store_Employee (Employee_ID, Employee_Name, SSN, Employee_Ph_No)
- Driver (**Employee_ID**, **Delivery_Id**, Driver_Name, Employee_Ph_No, Driver_Age, **Vh_No**)

IV. Implementation of Relation Model via MySQL and NoSQL

The database was created in MySQL and the following queries were performed:

MySQL Implementation:

Query1: Display the cities with 5-star rating given by the user; select OD.Delivery_ID, DD.CITY,AD.User_Rating from

Delivery_Destination DD join Order_Details OD on DD.Delivery_ID = OD.Delivery_ID join User_Details UD on UD.User_ID=OD.User_ID join Account_Details AD on AD.User_ID=OD.User_ID where AD.User_Rating = 5;

```
select OD.Delivery_ID, DD.CITY,AD.User_Rating from
Delivery_Destination DD join Order_Details OD on DD.Delivery_ID = OD.Delivery_ID
join User_Details UD on UD.User_ID=OD.User_ID join Account_Details AD on AD.User_ID=OD.User_ID
where AD.User_Rating = 5;
```

100% 26:7

Result Grid Filter Rows: Search Export:

Delivery_ID	CITY	User_Rating
80828	Lake Roel	5
88038	North Cullen	5
93682	Nyasiaville	5
155498	Klington	5
168987	Boyerberg	5
253368	Balistreriborough	5
265137	Connellymouth	5
320992	Reillyborough	5
373301	South Layla	5
411434	New Elizaton	5
451091	Janieview	5
478943	North Mathew	5
500885	West Fredy	5
550193	North Sonnyview	5
666896	Marksborough	5
727141	Ludieburgh	5
795822	North Dallin	5
795860	Port Earleneview	5
853538	Normachester	5

Query2:Display the top 10 payments made by users

select concat(UD.FirstName,UD.LastName) as UserName,Pt.Total_Amount from Payment PT join Order_Details OD on PT.Order_ID = OD.Order_ID join User_Details UD on UD.User_ID = OD.User_ID order by PT.Total_Amount desc limit 10;

```
select concat(UD.FirstName,UD.LastName) as UserName,Pt.Total_Amount from Payment PT
join Order_Details OD on PT.Order_ID = OD.Order_ID
join User_Details UD on UD.User_ID = OD.User_ID order by PT.Total_Amount desc limit 10;
```

100% 1:13

Result Grid Filter Rows: Search Export: Fetch rows:

UserName	Total_Amount
AngeloStokes	974
AdolfBartoletti	971
SharonTremblay	967
WatsonNienow	943
MagnoliaBlock	941
DiegoLeffler	939
JocelynRenner	931
AbigaylePaucek	930
RetaBode	914
TheresiaKonopelski	902

Query3: Display average total price of all user payments

select Round(avg(Pt.Total_Amount),2) as Average_Amount from Payment PT join Order_Details OD on PT.Order_ID = OD.Order_ID join User_Details UD on UD.User_ID = OD.User_ID;

```
16 •  select Round(avg(Pt.Total_Amount),2) as Average_Amount from Payment PT
17    join Order_Details OD on PT.Order_ID = OD.Order_ID
18    join User_Details UD on UD.User_ID = OD.User_ID;
19
```

100% 49:18

Result Grid Filter Rows: Search Export:

Average_Amount
506.52

Query4: Display the user who spent huge amount on orders with less order quantity with the shipment type.

select OD.SHIPMENT_TYPE, concat(UD.FirstName,UD.LastName) as UserName,OD.ORDER_AMOUNT,OD.Order_Qty from Order_Details OD join User_Details UD on OD.User_ID = UD.User_ID where OD.ORDER_AMOUNT >900 and Order_Qty<10 order by OD.Order_Qty desc;

```
21 •  select OD.SHIPMENT_TYPE, concat(UD.FirstName,UD.LastName) as UserName,OD.ORDER_AMOUNT,OD.Order_Qty
22    from Order_Details OD join User_Details UD
23    on OD.User_ID = UD.User_ID where OD.ORDER_AMOUNT >900 and Order_Qty<10 order by OD.Order_Qty desc;
24
```

100% 1:24

Result Grid Filter Rows: Search Export:

SHIPMENT_TYPE	UserName	ORDER_AMOUNT	Order_Qty
STANDARD	VivianneNader	971	7
EXPRESS	ShannaNader	981	5
EXPRESS	AyanaHuels	971	1

Query5: Which payment type used a greater number of coupons.

select count(PT.Payment_Type) as No_of_Coupons_used ,PT.Payment_Type from Payment PT join Order_Details OD on PT.Order_ID = OD.Order_ID join User_Details UD on UD.User_ID = OD.User_ID where PT.Holiday_Coupon not in ('') group by PT.Payment_Type order by No_of_Coupons_used desc;

```
26
27 •  select count(PT.Payment_Type) as No_of_Coupons_used ,PT.Payment_Type from Payment PT
28    join Order_Details OD on PT.Order_ID = OD.Order_ID
29    join User_Details UD on UD.User_ID = OD.User_ID
30    where PT.Holiday_Coupon not in ('') group by PT.Payment_Type
31    order by No_of_Coupons_used desc;
32
```

100% 34:31

Result Grid Filter Rows: Search Export:

No_of_Coupons_used	Payment_Type
56	ONLINE PAYMENT
17	COD

```
select sum(P.Product_Qty) as Product_Quantity,P.Product_Name, sum(OD.Order_Amount) as Total_amount
from Product P join Order_Details OD on P.Product_ID=OD.Product_ID
group by P.Product_Name order by Total_amount desc;
```

Product_Quantity	Product_Name	Total_amount
1030	CAPSICUM	20056
1258	Chicken	18309
220	IceCream	5050
170	Fanta	3345
267	Cocacola	2848

Query7: which store is delivering more number of orders

select count(OD.Order_ID) as orders, S.Store_Name from Store S join
Product P on S.Product_ID=P.Product_ID
join Order_Details OD on P.Product_ID=OD.Product_ID group by S.Store_Name order by orders desc;

```
select count(OD.Order_ID) as orders, S.Store_Name from Store S
join Product P on S.Product_ID=P.Product_ID
join Order_Details OD on P.Product_ID=OD.Product_ID group by S.Store_Name order by orders desc;
```

orders	Store_Name
27	New Patsyland
21	Lake Hermann
20	East Altheastad,
17	Jonesfort
7	Hansenland
6	Jamarland
2	South Maida

from Product
order by no_of_products desc;

```
select distinct count(Product_Type) over (partition by Product_Name
order by  Product_Type desc) as no_of_products,Product_Name, Product_Type
from Product
order by no_of_products desc;
```

no_of_produ...	Product_Name	Product_Type
43	Chicken	MEAT
37	CAPSICUM	VEGETABLES
33	Avocado	VEGETABLES
29	BOUNTY	PAPER
26	Tomato	VEGETABLES
25	Mayonnaise	CONDIMENTS
21	Milk	DAIRY
16	Noodles	CANNED
15	Apple	FRUITS
11	sugar	BAKING
9	IceCream	BEVERAGES
9	MUFFINS	BAKERY
7	Bread	BAKERY
7	Cocacola	BEVERAGES
7	IceCream	FROZEN
7	POPCORN	SNACKS
6	Fanta	BEVERAGES
5	Carrot	VEGETABLES
2	Chilly Powder	SPICES
1	sugar	Grocery

Query9: Which year has the highest number of orders

select count(Order_ID) as No_of_Orders,year(Order_Date) as Order_Year from Order_details group by Order_Year order by Order_Year desc;

```
54 •  select count(Order_ID) as No_of_Orders,year(Order_Date) as Order_Year
55    from Order_details
56    group by Order_Year order by Order_Year desc;
57
```

00% 46:56

Result Grid | Filter Rows: Search | Export:

No_of_Orders	Order_Year
56	2022
44	2021

Query10: Which employee is getting the highest salary

select Rank() over(Order by Employee_salary desc) as salary_rank, Employee_Salary,Employee_Name from store_employee limit 10;

```
59
60 •  select Rank() over(Order by Employee_salary desc) as salary_rank,
61    Employee_Salary,Employee_Name from store_employee limit 10;
62
```

00% 60:61

Result Grid | Filter Rows: Search | Export:

salary_rank	Employee_Salary	Employee_Name
1	80378	Mrs. Freeda Bogisich III
2	70000	Augustine Bechtelar
3	60670	Frida Zboncak
4	60608	Cloyd Schowalter
5	50678	Prof. Quinton Braun DVM
6	50608	Magdalen Buckridge
7	50278	Jacques Quigley
8	40870	Marcelo Jacobs
9	40600	Remington Nader
9	40600	Mr. Sanford Rolfson

NoSQL Implementation:

The database was created in MySQL and the following queries were performed:

Query1: How many orders are placed by users. db.Order_Details.aggregate([{ $group: { _id: "$User_ID", total_orders: { $sum: 1 } } }, { $sort: { total_orders: -1 } }])

```
MongoDB Enterprise > db.Order_Details.aggregate(  [ { $group: { _id: "$User_ID", total_orders: { $sum: 1 } } }, { $sort: { total_orders: -1 } }  ])
{ "_id" : "992926", "total_orders" : 17 }
{ "_id" : "992236", "total_orders" : 13 }
{ "_id" : "992936", "total_orders" : 11 }
{ "_id" : "902926", "total_orders" : 9 }
{ "_id" : "993936", "total_orders" : 2 }
MongoDB Enterprise >
```

Query2: What is the order quantity based on the shipping method

db.Order_Details.aggregate([{$group : {_id : "$Shipment_Type", Order_Qty : {$sum : 1}}}])

```
MongoDB Enterprise > db.Order_Details.aggregate([{$group : {_id : "$Shipment_Type", Order_Qty : {$sum : 1}}}])
{ "_id" : "EXPRESS", "Order_Qty" : 17 }
{ "_id" : "STANDARD", "Order_Qty" : 35 }
MongoDB Enterprise >
```

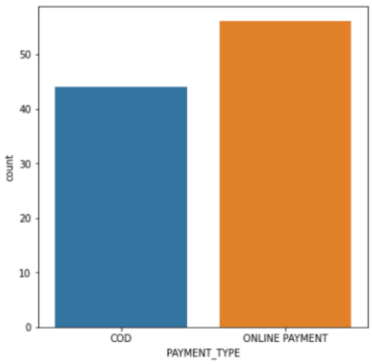
Query3: Which orders have the order quantity higher than 50?

```
MongoDB Enterprise > db.Order_details.find({Order_Qty: {$gte: 50}}).pretty()
{
        "_id" : ObjectId("6268035139dd6b0b73ae83cd"),
        "Order_Id" : "72378",
        "Order_Qty" : 56,
        "Shipment_Type" : "STANDARD",
        "Order_Amount" : "259",
        "Dept_ID" : "721780",
        "User_ID" : "992926",
        "Delivery_ID" : "961397",
        "Product_ID" : "922444",
        "Order_Date" : "2021-03-14"
}
{
        "_id" : ObjectId("62680d5339dd6b0b73ae83dd"),
        "Order_Id" : 72378,
        "Order_Qty" : 56,
        "Shipment_Type" : "STANDARD",
        "Order_Amount" : 259,
        "Dept_ID" : "721780",
        "User_ID" : 992926,
        "Delivery_ID" : "961397",
        "Product_ID" : "922444",
        "Order_Date" : 20210314
}
MongoDB Enterprise >
```

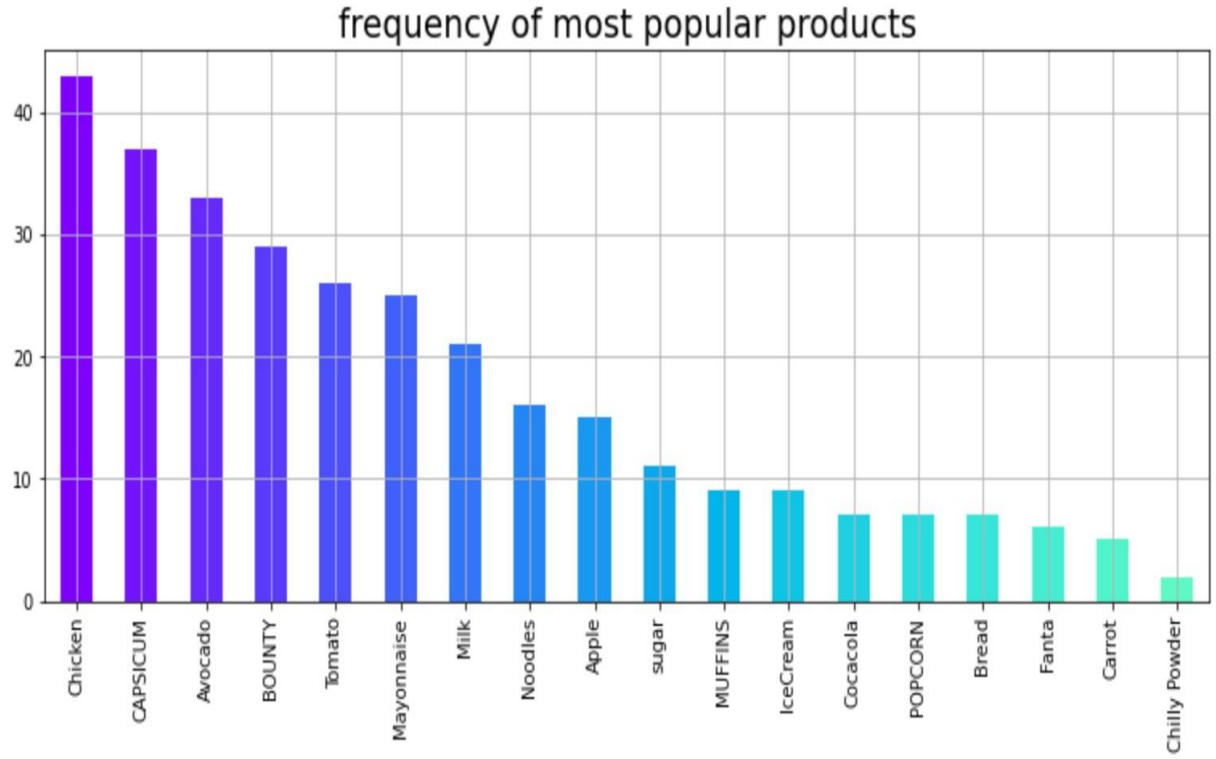
V. Database Access via Python

The database is accessed using Python and the visualization of analyzed data is shown below. The connection of MySQL to Python is done using pymysql, followed by (!pip install pymysql) to run and fetchall from query, followed by converting the list into a dataframe to plot the graphs for the analytics using the libraries pandas, pd.read_sql_query. We have taken use of the library dplyr, seaborn, numpy, matplotlib, network, and seaborn for basic analytical needs.

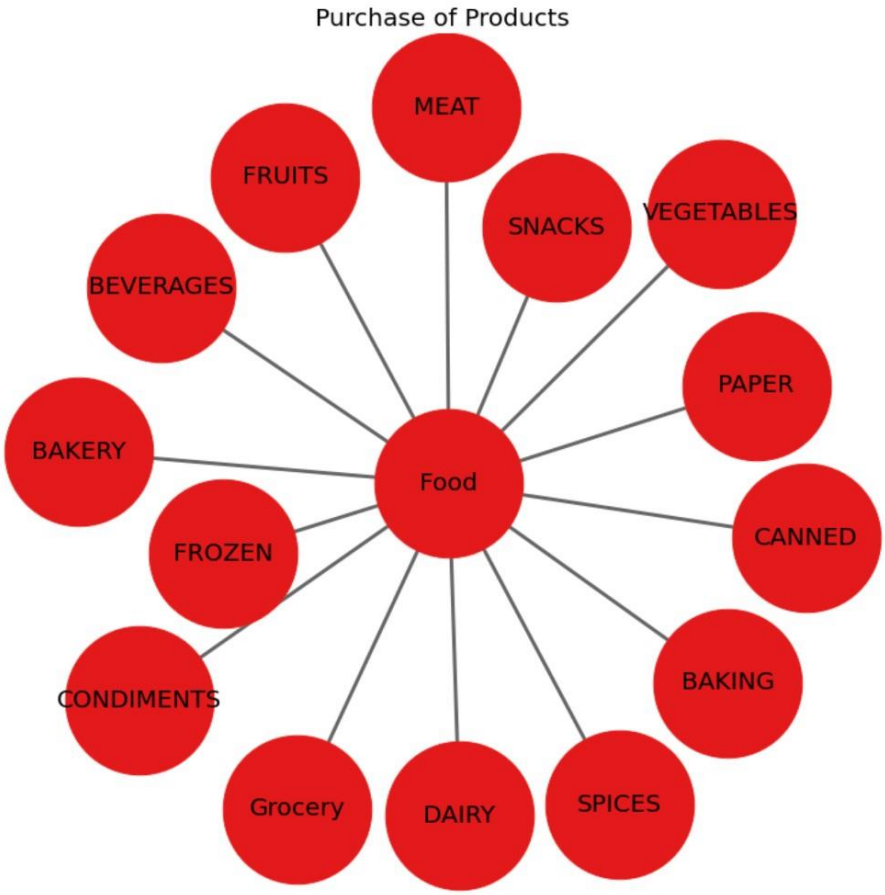
Graph 1: People who spent the orders using COD and Online Payment.



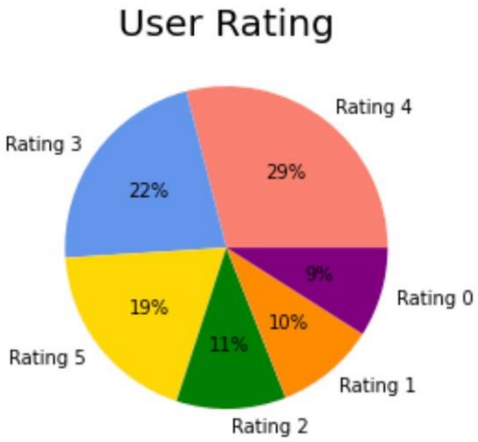
Graph 2: Frequency of most popular products purchased by users



Graph 3: Display the Products purchased by users(network graph showing the highest and lowest)



Graph 4: Display the Rating percentage given by users



VI. Summary and Recommendation

As shown, we have implemented our conceptual model in a SQL database and converted that to a NoSQL database successfully. The database for our company's Grocery Delivery Management system is up and running and is ready to go live and receive our first customer! We hope that the holiday offers and coupons we included, and the promise of delivering on-time lead us to our first customers.

The advantages would be that the delivery and logistics market has been hit badly by the pandemic, and we plan to make use of this opportunity with our ready-to-go database. We also provide different vehicles and services catering to the needs of all kinds of customers, with no minimum weight category or minimum distance to a destination.

The shortcomings now would be that we must shift between completely different environments for SQL, and NoSQL and compute our analytical insights using Python. As the company grows and we generate more revenue, we would like to improve on the database by shifting it to an AWS environment, with all the services it has to offer. It would be a one-stop-shop for our Relational, Non-Relational, Dashboarding, and machine learning needs with a seamless flow of data.