Hypermarket Database Management System

Milestone – Project Report

Group 17

Samarth Saxena

Neeraj Rangwani

857-376-1367

857-395-3475

saxena.sam@northeastern.edu
rangwani.n@northeastern.edu

Percentage of Effort Contributed by Samarth: 50
Percentage of Effort Contributed by Neeraj: 50

Signature of Samarth: SAMARTH SAXENA
Signature of Neeraj: NEERAJ RANGWANI

Submission Date: 12/10/2022

USE CASE STUDY REPORT

Group No.: Group 17

Student Names: Samarth Saxena and Neeraj Rangwani

Executive Summary:

The primary objective of this project was to design and implement a Database Management System (DBMS) for helping international students in Boston to accustomed to the city's purchasing practices. Our database can help students manage everything related to buying the best product according to their needs without disturbing their daily chores and studies.

The database was modelled considering the problems faced by international students on a day-to-day basis. For creating the EER and UML Model, we used an online tool called Lucidchart, which is a web-based application that allows users to visually collaborate on sharing charts and diagrams. This was followed by the mapping of the conceptual model to a relational model with the required primary and foreign keys. This relational model was then implemented using MySQL Workbench where we created 15 tables relevant to our problem statement and created sample queries to validate it. Three tables out of these were taken to implement the NoSQL part on MongoDB Compass.

The project also includes implementation on Python software. We connected our database to Python using MySQL Connector to work on some real-life analytics by generating some visualizations to understand the trend of our data. The future scope of this project would be to consider various other problems faced by international students so that we can expand the horizon of this project as well as submit our findings and implementations to the university so that they can help the students overcome these.

I. <u>Introduction</u>

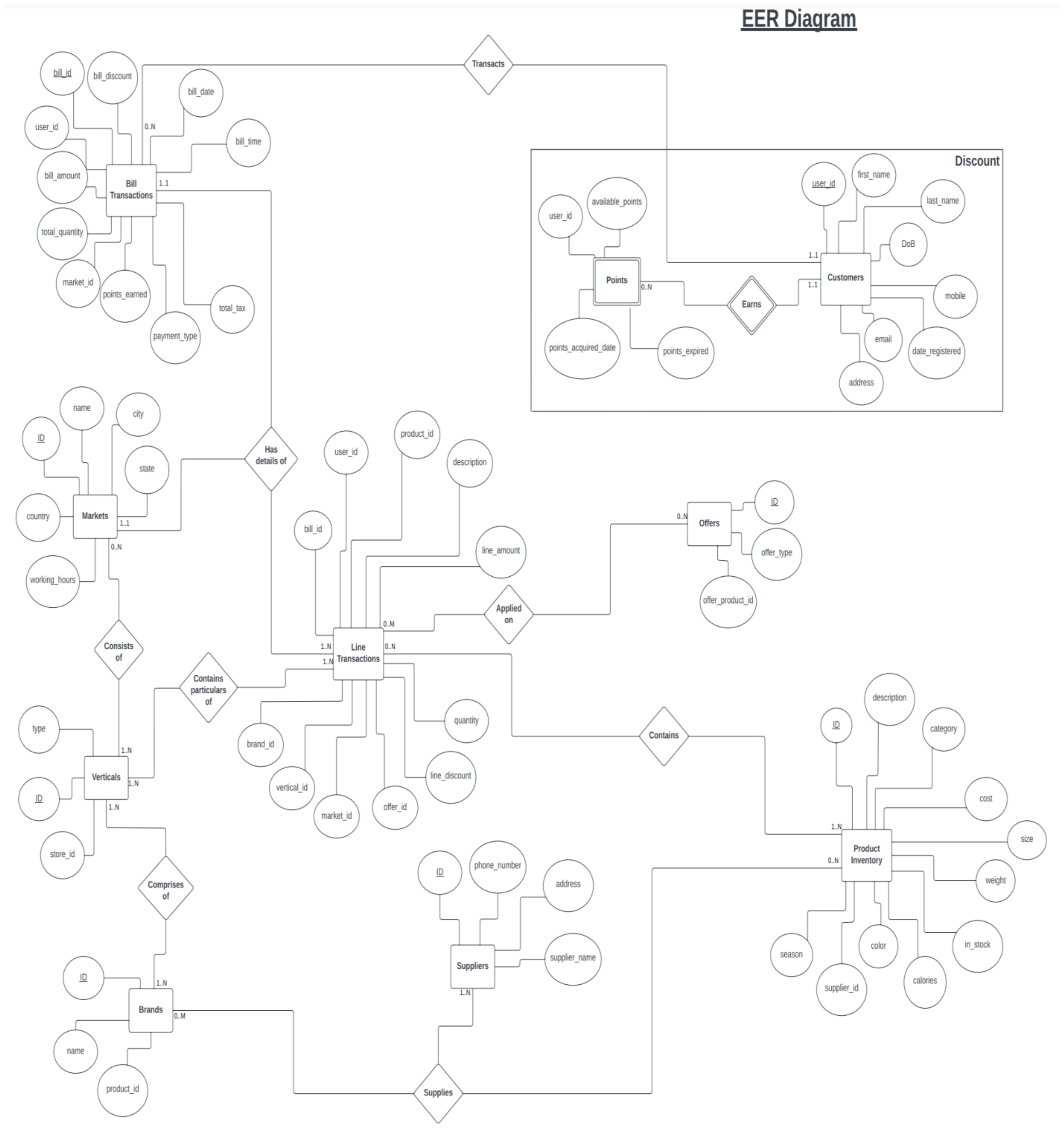
In the past few years, the number of international students traveling to the United States of America to pursue their higher education has increased sharply. The cultural diversity, vibrant lifestyle, and flexible education system makes the USA the topmost preferred destination for international students. According to a study abroad report, it is estimated that around 1,075,496 international students are enrolled in various universities spread across the country. Boston has traditionally been a desirable location for aspirational students because of its rich academic history. Students encounter a variety of challenges as they adjust to this new environment over the course of several months. International students may not be aware of many of those challenges, even if some may be clear. To be completely competitive in the United States, international students must overcome several significant obstacles, one of which is becoming accustomed to the country's purchasing practices.

A hypermarket is a big store combining a supermarket and a department store. As a result, a sizable retail space was created that houses a large variety of goods under one roof, including complete supermarket lines and general commodities. Customers can complete all their shopping needs in hypermarkets in a single trip. Some famous examples of hypermarkets in Boston are Walmart, Costco, Target, and IKEA. Hypermarkets have many advantages, but some of the significant problems international students (including us) have faced while buying products from hypermarkets are: American fast food is underhandedly ample, and international students find the quantity quite abundant. Hence it is essential to take calorie intake into account. There is a variation in the price of the same product in different stores in hypermarkets. Overspending on a product leads to very little savings at the end of the month; Students do not get timely information regarding sales, discounts, or arrival of a new product which leads to increased time and effort for traveling to the store, purchasing a product, and traveling back.

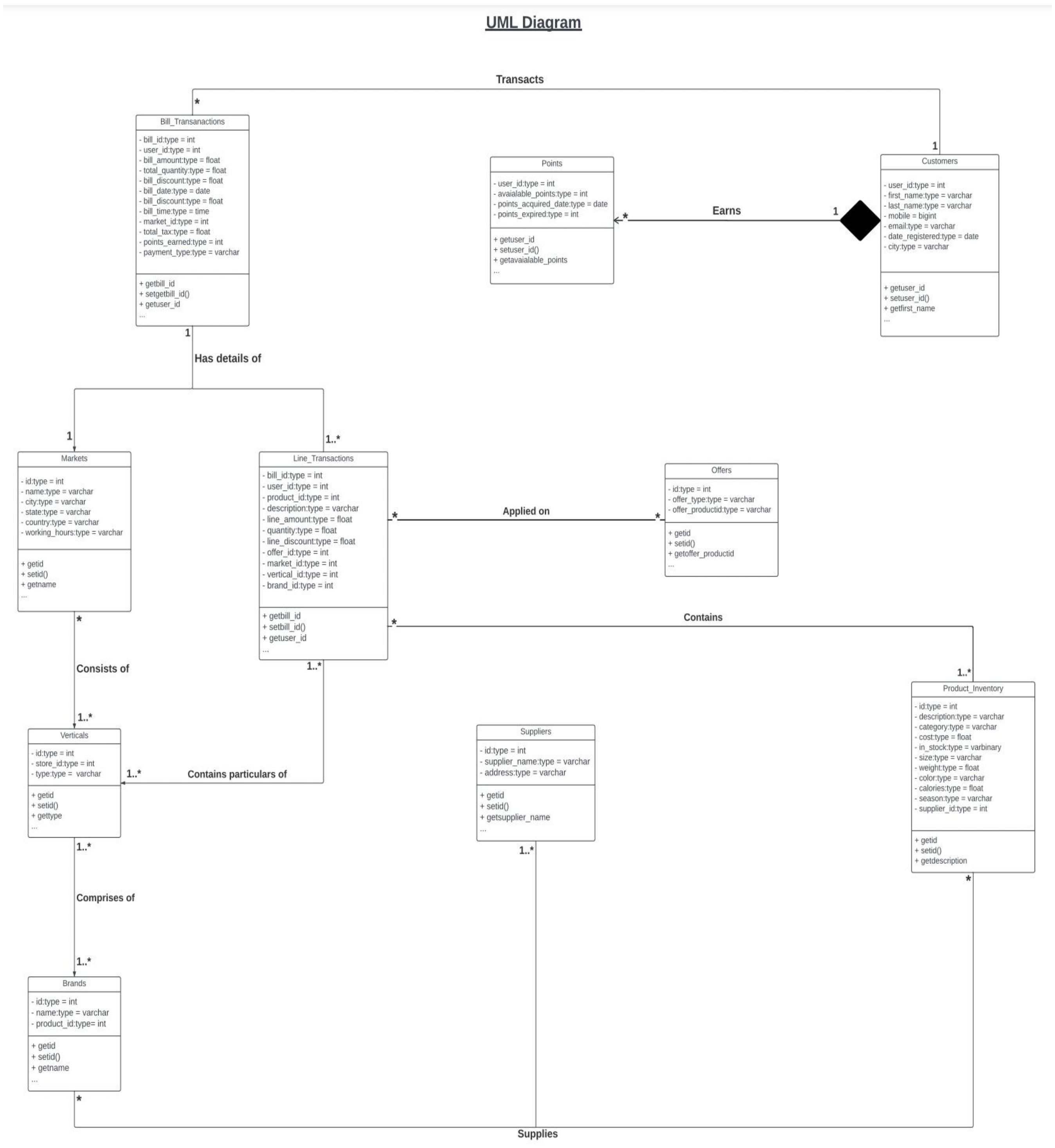
Hence there is a need to build a Database Management System (DBMS) to solve this real-life problem by helping students compare a particular product available in different hypermarkets that would be the best for them based on several factors. We have selected the most relevant domains that students buy from on a daily, weekly, or monthly basis, for example, groceries, clothing, toiletries, appliances, FnB, and electronics. Many students can benefit from this project by getting information about the best product at the most minimal cost and plan their shopping accordingly without wasting time researching the stores.

II. <u>Conceptual Data Modeling</u>

a. EER Diagram



b. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key – Underlined **Foreign Key – *Italicized***

- **Bill Transactions** (bill_id, *user_id*, bill_discount, bill_date, bill_time, bill_amount, total_quantity, market_id, total_tax, points_earned, payment_type)
  - User_ID foreign key refers to User_ID in Customers, NOT NULL

- **Line Transactions** (bill_id, *user_id*, product_id, description, line_amount, quantity, line_discount, offer_id, market_id, vertical_id, brand_id)
  - User_ID foreign key refers to User_ID in Customers, NOT NULL

- **Points** (*user_id*, available_points, points_acquired_date, points_expired)
  - User_ID foreign key refers to User_ID in Customers, NOT NULL

- **Customers** (user_id, first_name, last_name, mobile, email, date_registered, DoB, address)

- **Product Inventory** (ID, description, supplier_id, color, season size, in_stock, weight, category, calories, cost)

- **Offers** (ID, offer_type)

- **Markets** (ID, name, city, state, country, working_hours)

- **Verticals** (ID, type)

- **Brands** (ID, name, product_id, *V-ID*)

  - V-ID foreign key refers to ID in Verticals, NOT NULL

- **Suppliers** (ID**,** supplier_name, address, phone_number)

- **Consists of** (*M-ID*, *V-ID*)

  - M-ID foreign key refers to ID in Markets, NULL Allowed
  - V-ID foreign key refers to ID in Verticals, NOT NULL

- **Contains** (*bill_id*, *PI-ID*)
  - Bill_ID foreign key refers to Bill_ID in Line Transactions, NOT NULL
  - PI-ID foreign key refers to ID in Product Inventory, NOT NULL

- **Supplies** (*B-ID*, *S-ID*, *PI-ID*)
  - B-ID foreign key refers to ID in Brands, NOT NULL
  - S-ID foreign key refers to ID in Suppliers, NOT NULL
  - PI-ID foreign key refers to ID in Product Inventory, NOT NULL

- **Applied on** (*bill_id*, *O-ID*)
  - Bill_ID foreign key refers to Bill_ID in Line Transactions, NOT NULL
  - O-ID foreign key refers to ID in Offers, NULL Allowed

- **Discount** (*user_id*, discount_amount)
  - User_id foreign key refers to user_id in Customers, NOT NULL

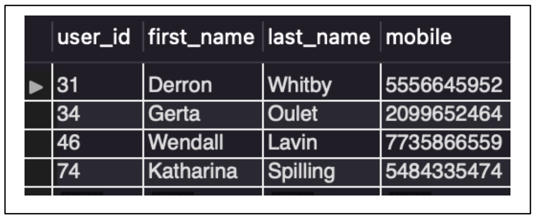
IV. Implementation of Relational Model via MySQL and NoSQL

MySQL Implementation:

Total 15 tables were created on MySQL Workbench, and the following queries were executed:

Query 1: Customers who registered between Jun 1, 2021 and Dec 31, 2021, and made a purchase in 2022

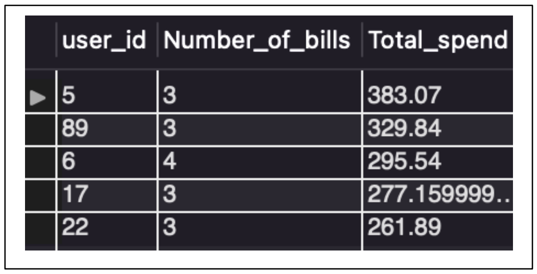
```
select user_id, first_name, last_name, mobile
from customers
where date_registered between '2021-06-01' and '2021-12-31'
and user_id in
(select user_id
from bill_transactions
where bill_date between '2022-01-01' and '2022-12-31');
```



user_id	first_name	last_name	mobile
31	Derron	Whitby	5556645952
34	Gerta	Oulet	2099652464
46	Wendall	Lavin	7735866559
74	Katharina	Spilling	5484335474

Query 2: Total number of bills and spend of top 5 customers (ranked by spend)

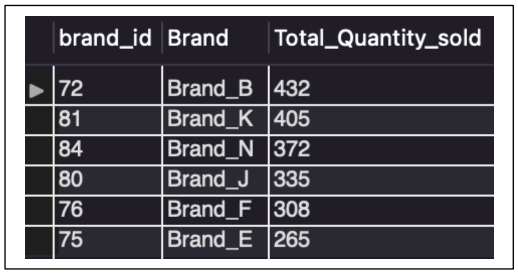
```
select user_id, count(distinct bill_id) Number_of_bills, sum(bill_amount) Total_spend
from bill_transactions
group by 1
order by 3 desc limit 5;
```



user_id	Number_of_bills	Total_spend
5	3	383.07
89	3	329.84
6	4	295.54
17	3	277.159999..
22	3	261.89

Query 3: Ranking brands according to total number of products sold, and considering only the brands which have sold more than 250 products

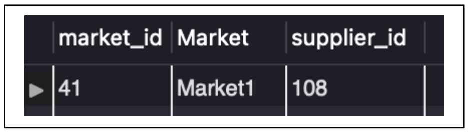
```
select brand_id, b.name, sum(quantity) Total_Quantity_sold
from brands b join consists_of c on b.v_id = c.v_id
join line_transactions l on l.brand_id = b.id
group by 1,2
having Total_Quantity_sold >= 250
order by 3 desc;
```



brand_id	Brand	Total_Quantity_sold
72	Brand_B	432
81	Brand_K	405
84	Brand_N	372
80	Brand_J	335
76	Brand_F	308
75	Brand_E	265

Query 4: Market and supplier which are providing the product number '1047' at the cheapest price

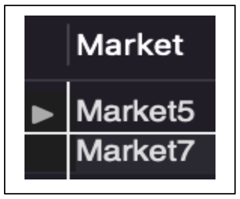
```
select market_id, Name Market, supplier_id
from line_transactions l join product_inventory p on l.product_id = p.id
join markets m on l.market_id = m.id
where product_id = '1047'
and line_amount <= ALL
 (select line_amount
from line_transactions
where product_id = '1047');
```



market_id	Market	supplier_id
41	Market1	108

Query 5: All the markets which can supply the product number '1026' at a price lower than its average priceselect distinct m.Name Market

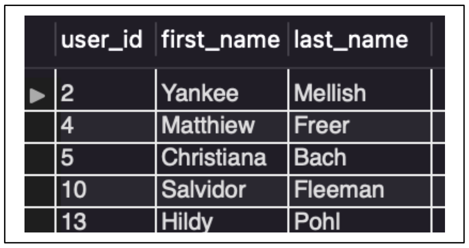
```
from markets m join line_transactions l on m.id = l.market_id
where product_id = '1026'
and line_amount <
(select avg(line_amount)
from line_transactions l1
where l.product_id = l1.product_id);
```



Market
Market5
Market7

Query 6: Customers who have purchased a product using a credit card

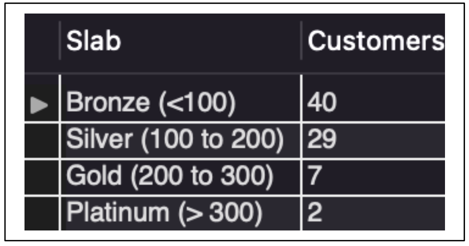
```
select c.user_id, c.first_name, c.last_name
from customers c
where exists
(select *
from bill_transactions b
where c.user_id = b.user_id
and payment_type = 'Credit Card')
order by 1 limit 5;
```



user_id	first_name	last_name
2	Yankee	Mellish
4	Matthiew	Freer
5	Christiana	Bach
10	Salvidor	Fleeman
13	Hildy	Pohl

Query 7: Categorize users into different slabs based on their total points earned, and show the number of customers in each slab.

```
with points_earned as
(select user_id, sum(points_earned) total_points from bill_transactions
group by 1),
slab as
(select user_id,
case when total_points < 100 then 'Bronze (<100)'
when total_points >= 100 and total_points < 200 then 'Silver (100 to 200)'
when total_points >= 200 and total_points < 300 then 'Gold (200 to 300)'
else 'Platinum (> 300)' end Slab
from points_earned)
select Slab, count(user_id) Customers from slab
group by 1 order by 2 desc;
```



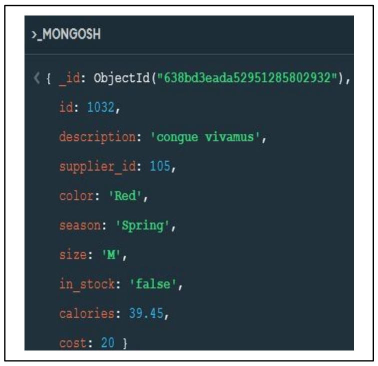
Slab	Customers
Bronze (<100)	40
Silver (100 to 200)	29
Gold (200 to 300)	7
Platinum (> 300)	2

NoSQL Implementation:

3 tables - Bill_transactions, Customers and Product_Inventory were created on MongoDB Compass, and the following queries in MongoDB were executed:

Query 1: Products which are red in color and not available

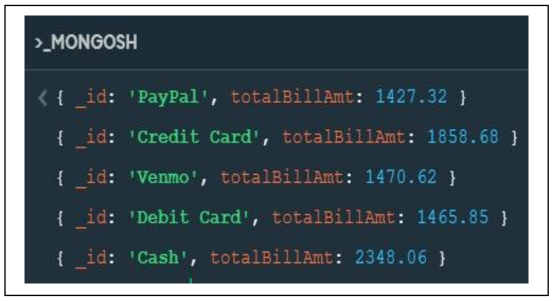
```
db.Product_Inventory.find( { color: 'Red', in_stock: 'false' } )
```



```
>_MONGOSH
< { _id: ObjectId("638bd3eada52951285802932"),
    id: 1032,
    description: 'congue vivamus',
    supplier_id: 105,
    color: 'Red',
    season: 'Spring',
    size: 'M',
    in_stock: 'false',
    calories: 39.45,
    cost: 20 }
```

Query 2: Total Sales by the payment methods

```
db.Bill_Transactions.aggregate([
{ $group : { _id : "$payment_type",
totalBillAmt: { $sum: "$bill_amount" } } }
])
```

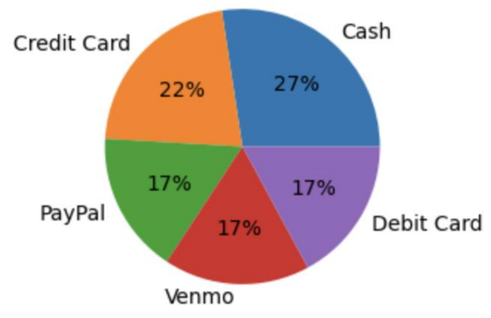


```
>_MONGOSH
< { _id: 'PayPal', totalBillAmt: 1427.32 }
  { _id: 'Credit Card', totalBillAmt: 1858.68 }
  { _id: 'Venmo', totalBillAmt: 1470.62 }
  { _id: 'Debit Card', totalBillAmt: 1465.85 }
  { _id: 'Cash', totalBillAmt: 2348.06 }
```

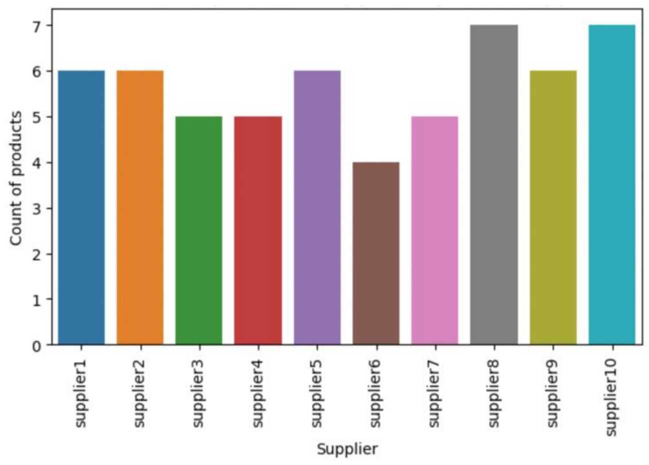
V. Database access via Python

MySQL workbench is accessed through Jupyter notebook and the connection is established using mysql.connector.

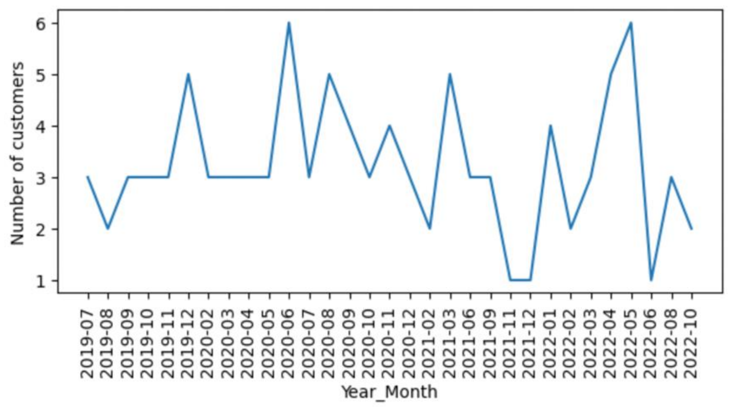
Graph 1: % of Sales Distribution for all the payment methods



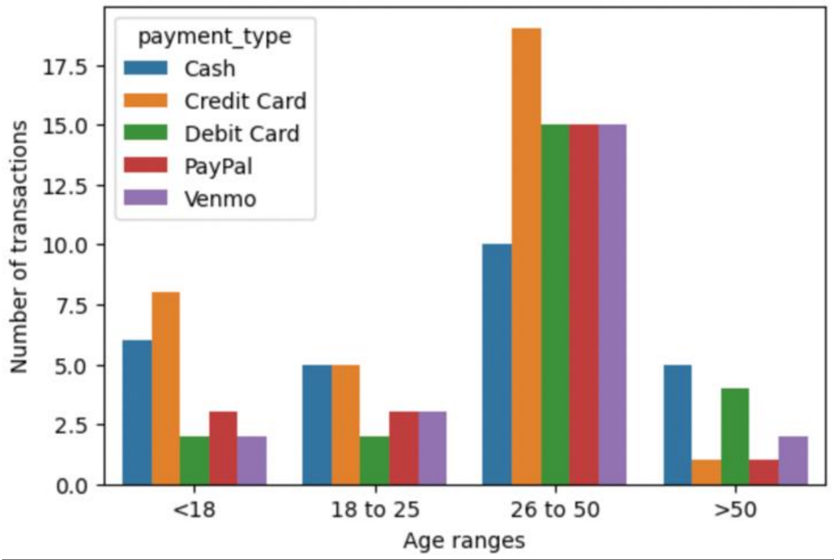
Graph 2: Total number of products supplied by each supplier



Graph 3: Trend of customers registered month wise between Jul 2019 and Oct 2022



Graph 4: Age ranges and their preferred payment methods



VI. Summary and Recommendation

A Database Management System was implemented successfully using MySQL, MongoDB, and Python. By learning about the greatest product at the lowest price and planning their shopping accordingly without wasting time looking up retailers, many people can benefit from this especially international student.

Implementation of entire database in MongoDB - ‘Contains’ table references the ‘bill_id’ in Line_transactions table and the ‘product_id’ from the Product_Inventory table which results in information duplication across the documents and can cause inconsistent data creation if the table is not updated in real-time.

There should be more research on how basic information duplicates across multiple documents, complicates the design, and leads to inconsistencies. Linking multiple documents using document identifiers can be a possible solution to this problem.