# IE 6700- DATA MANAGEMENT FOR ANALYTICS

## USE CASE STUDY REPORT

**Group No :** 11

**Student Names:** Prashanth Krishnamsetty and Akshay Nagireddy

## Executive Summary:

The objective of this project is to effectively design and implement a database for a huge apparel retailer for their online store. A company called Brivio is a pioneer in designer apparel and has been making a huge revenue through its brick-and-mortar stores. But, in the last 2 years due to the pandemic, they took a major hit on their revenue as most of their stores were closed, creating an unavailability of their apparel to their customers. The marketing head of the company approached us with an idea for launching an online store for their products. They are aiming for a very minimalistic, user-friendly and a comprehensive website and need our help with setting up their Database design and implementation for their website. The main stakeholders who will be using the website will be customers. The EER and UML diagrams were modelled, followed by the mapping of the conceptual model to a relational model with the required primary and foreign keys. This database was then implemented fully with MySQL and a prototype database was bult using MongoDB for performance and capability study. The created database is a great success, and the database was also connected to Python environment for further performing advanced analytics and visualizations, of which some of them were included in this report.

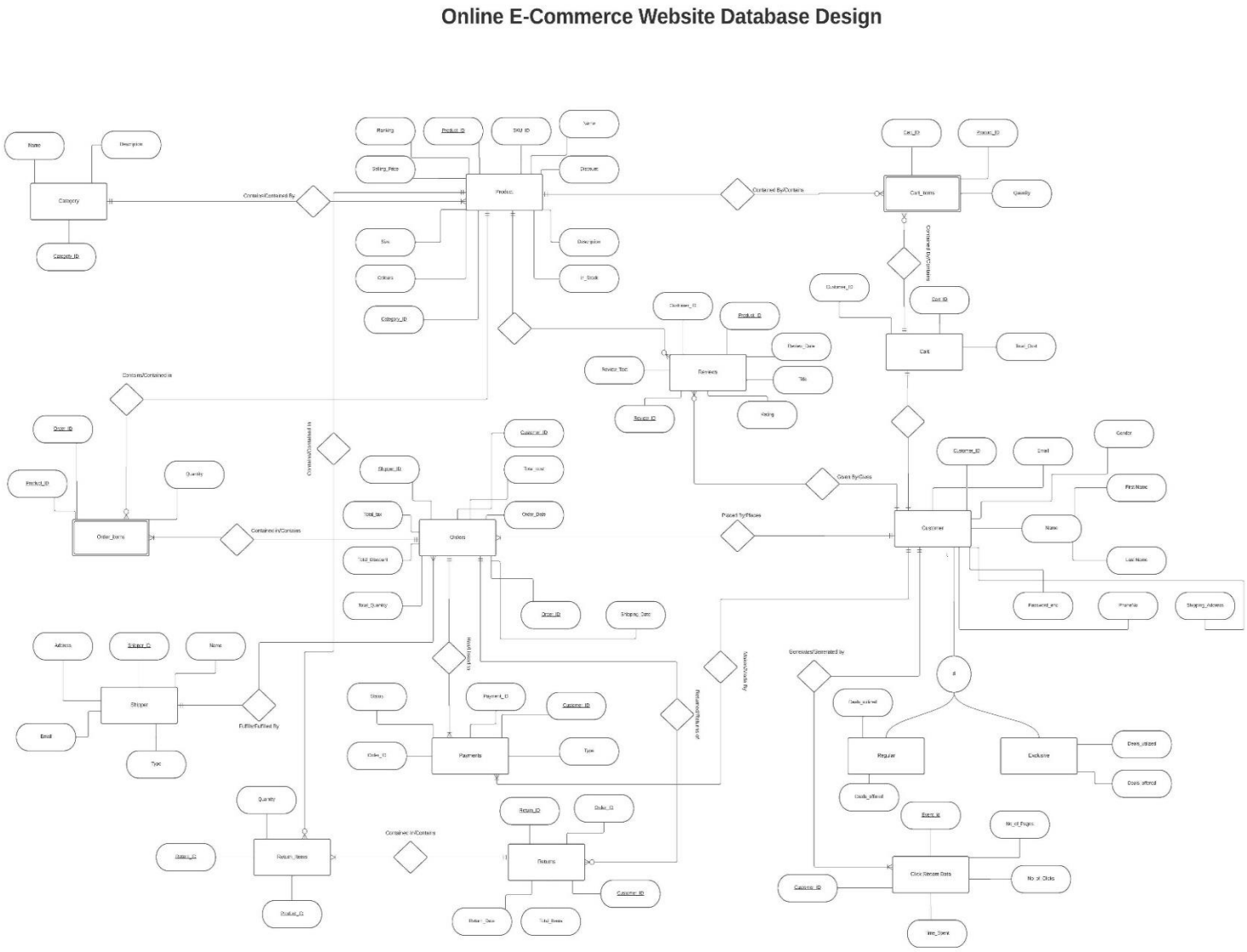
## Introduction

Online Shopping today is a lifestyle e-commerce web application, which retails various fashion and lifestyle products. This project allows customers to shop for diverse products available across store, register for a premium membership and various other capabilities which are listed below.

There are basically two types of customers, regular and exclusive. A customer is identified by unique id and their name (first and last name), email, password_encryption, phone number, gender needs to be captured. Both regular and exclusive customers receive deals personally tailored for them, through their emails and number of deals offered and utilized to be captured. A customer has only one cart and can add any number of items in their cart and save for it later. A cart is identified by a unique ID and the total cost need to be captured. The cart items need to contain the product, and quantity. The click stream data is collected for the analysis of the marketing team. An event is created every time a customer login. Each event is identified by a unique id and number of pages visited, number of clicks, time spent in total needs to be captured for that event. A customer needs to place at least one order and the order needs to have at least one item in it. An order is identified by a unique id and order date, total cost, total tax, total discount, total quantity, shipping date needs to be captured. The order items need to contain unique products and quantities respectively and the quantity cannot exceed 10 counts for each product. Each customer can write any number of reviews/suggestions for a particular product regardless of their purchase. These reviews will be utilized for further analysis. A customer needs to make at least one transaction for the order and each transaction is associated with unique customer. A transaction is identified by a unique id and status, type needs to be captured.

Each order is fulfilled by only one shipper. Every shipper is identified by a unique id and name, address, email, type needs to be captured. A shipper can take up to deliver any number of orders. A product is identified by a unique id and its name, description, sku id, ranking, selling price, discount, In/out of stock, size, colors, category, needs to be captured. A category needs to have at least one product, but a product can belong to only one category. A category is identified by a unique number and its name, description needs to be captured. A customer can also return any number of items that he/she has purchased in a specific order. The return table is identified by a return_id and it needs to capture return date, total number of items returned for a particular order. The return items need to contain unique products and quantities respectively.

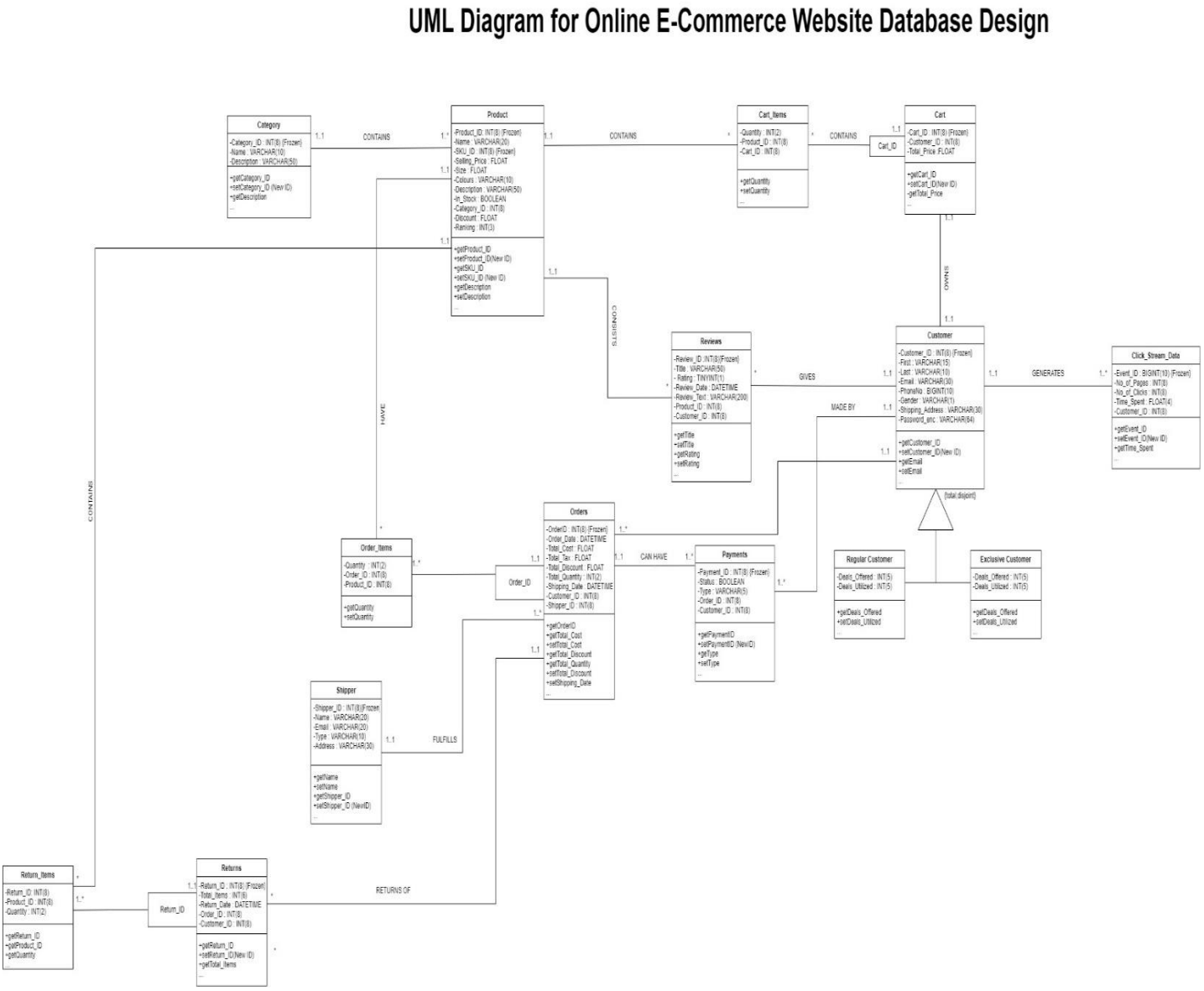
## Below is the EER diagram for the above database design




## The pdf version of the same EER diagram is available below

Adobe Acrobat Document

## Below is the UML diagram for the above database design




## The pdf version of the same UML diagram is available below

Adobe Acrobat Document

## The Relational Model for the above database design is as follows

**Primary Keys are underlined, and foreign keys are mentioned in *Italic* font**

Customer(Customer_ID, First_Name, Last_Name, Email, Gender, Password_enc, PhoneNo, Shipping_Address, Customer_Type, Deals_Utilized, Deals_Offered)

Click_Stream_Data(Event_ID, *Customer_ID*, No_of_pages, No_of_Clicks, Time_Spent)

Cart(Cart_ID, *Customer_ID*, Total_Cost)

Cart_Items(*Cart_ID*, *Product_ID*, Quantity)

Product(Product_ID, *Category_ID*, SKU_ID, Name, Ranking, Selling_Price, Discount, Description, In_Stock, Size, Colors)

Category(Category_ID, Name, Description)

Reviews(Review_ID, *Customer_ID*, Product_ID, Review_Text, Rating, Title, Review_Date)

Orders(Order_ID, *Shipper_ID*, *Customer_ID*, Total_Cost, Order_Date, Total_tax, Total_Discount, Total_Quantity,Shipping_Date)

Order_Items(*Order_ID*, *Product_ID*, Quantity)

Shipper(Shipper_ID, Address, Name, Email, Type)

Payments(Payment_ID, *Customer_ID*, *Order_ID*, Status, Type)

Returns(Return_ID, *Order_ID*, *Customer_ID*, Return_Date, Total_Items)

Return_Items(*Return_ID*, *Product_ID*, Quantity)

## Implementation of Relational Model via MySQL :

## Query 1:

**Find the average time a premium and regular customers spend on the website**

```
select c.customer_type, avg(cs.time_spent) from  customer as c inner join click_stream as cs on c.customer_id = cs.customer_id

group by c.customer_type;
```

| customer_type | avg(cs.time_spent) |
|---|---|
| Regular | 253.1477 |
| Premium | 241.0553 |

## Query 2:

**Get the top 5 ranked customers based on max orders from both premium and regular category**

```
SELECT * FROM (

WITH top as (

select c.Customer_id , c.customer_type,   count(order_id) as order_count

from customer  as c left join orders as o on c.customer_id = o.customer_id group by c.customer_id order by order_count desc

)

select *, DENSE_RANK() over(partition by customer_type order by order_count desc) as top_Category from top ) AS final

where top_Category <=3;
```

Result Grid | Filter Rows: | Export: | W

| Customer_id | customer_type | order_count | top_Category |
|---|---|---|---|
| 1163 | Premium | 5 | 1 |
| 1697 | Premium | 5 | 1 |
| 1739 | Premium | 4 | 2 |
| 1029 | Premium | 4 | 2 |
| 1419 | Premium | 4 | 2 |

## Query 3:

**Categorize the rating into 3 groups and identify the prodcts with high/neutral/low sentiment based on their average rating**

```
Select * , case

when Average_Rating >4 then 'High Sentiment'

WHEN Average_Rating >=3 and Average_Rating <4  then 'Neutral Sentimennt'

ELSE 'Negative Sentiment'

END AS Sentiment

FROM

(

select p.Product_id, p.P_name, p.selling_price, avg(r.rating) as Average_Rating from product as p, reviews as r

where p.product_id = r.product_id group by p.product_id ) as a order by Average_Rating desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Product_id | P_name | selling_price | Average_Rating | Sentiment |
|---|---|---|---|---|
| 237 | Shirt Dress | 2748 | 4.7143 | High Sentiment |
| 229 | Peplum Dress | 958 | 4.5714 | High Sentiment |
| 246 | Tent dress | 408 | 4.5385 | High Sentiment |
| 221 | Long Sleeve Dress | 2894 | 4.5294 | High Sentiment |
| 205 | Balloon Dress | 110 | 4.4500 | High Sentiment |

## Query 4:

**Find the average delivery time across all the orders form premium and regular customers across the shipping type**

select c.customer_type, s.s_type, avg(datediff(o.shipping_Date, o.order_date)) as Average_Delivery_Days from orders as o

left join customer as c on o.customer_id = c.customer_id

left join shipper as s on o.shipper_id = s.shipper_id

group by c.customer_type,s.s_type

| customer_type | s_type | Average_Delivery_Days |
|---|---|---|
| Regular | Non-Prime | 10.7581 |
| Premium | Prime | 10.8039 |
| Premium | Non-Prime | 9.9602 |
| Regular | Prime | 10.2197 |

## Query 5:

**Total revenue to the company for the fiscal year 2021-2022 with each individual contribution**

select *, round(Revenue/Total_revenue *100,2) as Contribution from (

select customer_id, sum(Total_Cost - Total_tax - Total_Discount ) as Revenue, sum(sum(Total_Cost - Total_tax - Total_Discount )) over() as Total_Revenue

from orders group by customer_id order by Revenue desc) as a;

| customer_id | Revenue | Total_Revenue | Contribution |
|---|---|---|---|
| 1007 | 17975 | 2321976 | 0.77 |
| 1886 | 12957 | 2321976 | 0.56 |
| 1697 | 12723 | 2321976 | 0.55 |
| 1634 | 12630 | 2321976 | 0.54 |
| 1672 | 12165 | 2321976 | 0.52 |

## Implementation of Relational Model via NoSQL using MongoDB:

## Query 1:

### Customers with maximum time spent on the website (top 10)

```
pipeline = [
  { "$group": {"_id": "$Customer_ID", "value": { "$avg": "$Time_Spent" } } },
  { "$sort" : {"value" : -1 } },
  { "$limit": 10 }
]
cursor = ds.aggregate(pipeline)
```

```
{'_id': 1673, 'value': 497.0}
{'_id': 1785, 'value': 497.0}
{'_id': 1471, 'value': 494.0}
{'_id': 1322, 'value': 494.0}
{'_id': 1407, 'value': 492.0}
{'_id': 1334, 'value': 492.0}
{'_id': 1611, 'value': 491.0}
{'_id': 1245, 'value': 490.0}
{'_id': 1795, 'value': 489.0}
{'_id': 1131, 'value': 489.0}
```

## Query 2:

### Find the customers who surfed more than 9 pages in a session and had more than 450 clicks

```
myquery = { "No_of_pages": { "$gt": 9 },
          "No_of_Clicks" : { "$gt": 450}
        }
```

```
{'_id': ObjectId('6260b119dfd857023bd30c84'), 'Event_ID': 7087, 'Customer_ID': 1235, 'No_of_pages': 10, 'No_of_Clicks': 459, 'T
ime_Spent': 430}
{'_id': ObjectId('6260b119dfd857023bd30cd2'), 'Event_ID': 7167, 'Customer_ID': 1531, 'No_of_pages': 10, 'No_of_Clicks': 477, 'T
ime_Spent': 103}
{'_id': ObjectId('6260b119dfd857023bd30d0a'), 'Event_ID': 7223, 'Customer_ID': 1517, 'No_of_pages': 10, 'No_of_Clicks': 468, 'T
ime_Spent': 5}
{'_id': ObjectId('6260b119dfd857023bd30d4c'), 'Event_ID': 7290, 'Customer_ID': 1342, 'No_of_pages': 10, 'No_of_Clicks': 463, 'T
ime_Spent': 346}
{'_id': ObjectId('6260b119dfd857023bd30d98'), 'Event_ID': 7367, 'Customer_ID': 1316, 'No_of_pages': 10, 'No_of_Clicks': 462, 'T
ime_Spent': 116}
{'_id': ObjectId('6260b119dfd857023bd30db5'), 'Event_ID': 7396, 'Customer_ID': 1887, 'No_of_pages': 10, 'No_of_Clicks': 468, 'T
ime_Spent': 409}
{'_id': ObjectId('6260b119dfd857023bd30dbe'), 'Event_ID': 7405, 'Customer_ID': 1461, 'No_of_pages': 10, 'No_of_Clicks': 487, 'T
ime_Spent': 21}
{'_id': ObjectId('6260b119dfd857023bd30e41'), 'Event_ID': 7537, 'Customer_ID': 1679, 'No_of_pages': 10, 'No_of_Clicks': 473, 'T
ime_Spent': 54}
{'_id': ObjectId('6260b119dfd857023bd30e85'), 'Event_ID': 7605, 'Customer_ID': 1952, 'No_of_pages': 10, 'No_of_Clicks': 470, 'T
ime_Spent': 47}
{'_id': ObjectId('6260b119dfd857023bd30ea4'), 'Event_ID': 7636, 'Customer_ID': 1173, 'No_of_pages': 10, 'No_of_Clicks': 481, 'T
ime_Spent': 468}
{'_id': ObjectId('6260b119dfd857023bd30ef2'), 'Event_ID': 7714, 'Customer_ID': 1477, 'No_of_pages': 10, 'No_of_Clicks': 489, 'T
ime_Spent': 371}
{'_id': ObjectId('6260b119dfd857023bd30f3d'), 'Event_ID': 7792, 'Customer_ID': 1609, 'No_of_pages': 10, 'No_of_Clicks': 468, 'T
ime_Spent': 232}
{'_id': ObjectId('6260b119dfd857023bd30fcb'), 'Event_ID': 7936, 'Customer_ID': 1810, 'No_of_pages': 10, 'No_of_Clicks': 495, 'T
ime_Spent': 116}
{'_id': ObjectId('6260b119dfd857023bd30fda'), 'Event_ID': 7951, 'Customer_ID': 1284, 'No_of_pages': 10, 'No_of_Clicks': 484, 'T
ime_Spent': 10}
{'_id': ObjectId('6260b119dfd857023bd30ffc'), 'Event_ID': 7985, 'Customer_ID': 1976, 'No_of_pages': 10, 'No_of_Clicks': 461, 'T
ime_Spent': 295}
```

## Query 3:

**Find the product with highest selling price across the store**

da.find_one(sort=[("Selling_Price", -1)])

```
{'_id': ObjectId('6260b119dfd857023bd31011'),
 'Product_ID': 207,
 'Category_ID': 456,
 'SKU_ID': 2117,
 'P_Name': 'Blouson Dress',
 'Ranking': 4,
 'Selling_Price': 2964.0,
 'Discount': 19.0,
 'P_Description': 'Loss control mv-pers NOS',
 'In_Stock': 'No',
 'Size': 'L',
 'Colors': 'Orange'}
```

## Query 4:

**Find the products which are not in stock and price greater than 500 dollars**

myquery = { "In_Stock": { "$ne": "Yes" },

"Selling_Price" : { "$gt": 500}

}

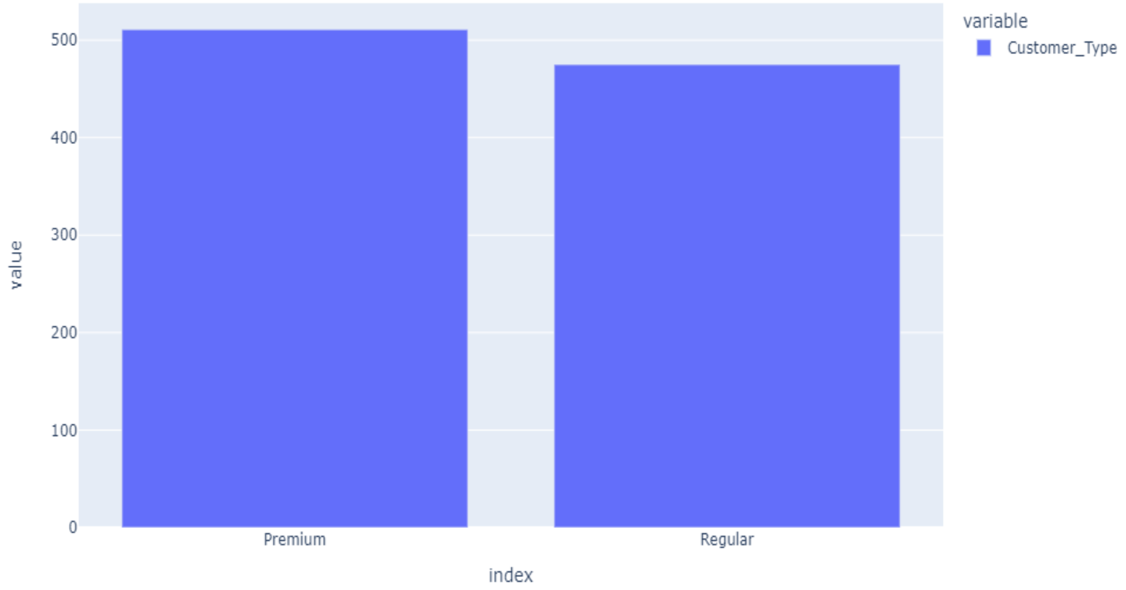
```
{'_id': ObjectId('6260b119dfd857023bd3100d'), 'Product_ID': 203, 'Category_ID': 101, 'SKU_ID': 4172, 'P_Name': 'Baby Doll Dres
s', 'Ranking': 37, 'Selling_Price': 1106.0, 'Discount': 32.0, 'P_Description': 'Disc dis NEC/NOS-lumbar', 'In_Stock': 'No', 'Si
ze': 'M', 'Colors': 'Aquamarine'}
{'_id': ObjectId('6260b119dfd857023bd31011'), 'Product_ID': 207, 'Category_ID': 456, 'SKU_ID': 2117, 'P_Name': 'Blouson Dress',
'Ranking': 4, 'Selling_Price': 2964.0, 'Discount': 19.0, 'P_Description': 'Loss control mv-pers NOS', 'In_Stock': 'No', 'Size':
'L', 'Colors': 'Orange'}
{'_id': ObjectId('6260b119dfd857023bd31013'), 'Product_ID': 209, 'Category_ID': 456, 'SKU_ID': 3221, 'P_Name': 'Camisole Dres
s', 'Ranking': 45, 'Selling_Price': 1615.0, 'Discount': 24.0, 'P_Description': 'Trans arthropathy-mult', 'In_Stock': 'No', 'Siz
e': 'S', 'Colors': 'Yellow'}
{'_id': ObjectId('6260b119dfd857023bd31016'), 'Product_ID': 212, 'Category_ID': 456, 'SKU_ID': 2489, 'P_Name': 'Denim Dress',
'Ranking': 4, 'Selling_Price': 2484.0, 'Discount': 11.0, 'P_Description': 'Behcet arthritis-mult', 'In_Stock': 'No', 'Size':
'S', 'Colors': 'Red'}
{'_id': ObjectId('6260b119dfd857023bd31017'), 'Product_ID': 213, 'Category_ID': 101, 'SKU_ID': 1125, 'P_Name': 'Empire Waisted
Dress', 'Ranking': 48, 'Selling_Price': 1990.0, 'Discount': 5.0, 'P_Description': 'Complic labor NOS-unsp', 'In_Stock': 'No',
'Size': 'S', 'Colors': 'Teal'}
{'_id': ObjectId('6260b119dfd857023bd31018'), 'Product_ID': 214, 'Category_ID': 101, 'SKU_ID': 2398, 'P_Name': 'Fit and Flare D
ress', 'Ranking': 25, 'Selling_Price': 1539.0, 'Discount': 62.0, 'P_Description': 'Ac vasc insuff intestine', 'In_Stock': 'No',
'Size': 'M', 'Colors': 'Indigo'}
{'_id': ObjectId('6260b119dfd857023bd31019'), 'Product_ID': 215, 'Category_ID': 123, 'SKU_ID': 3556, 'P_Name': 'Halter Neck Dre
ss', 'Ranking': 17, 'Selling_Price': 2010.0, 'Discount': 38.0, 'P_Description': 'Adv eff biologic NEC/NOS', 'In_Stock': 'No',
'Size': 'XL', 'Colors': 'Violet'}
{'_id': ObjectId('6260b119dfd857023bd3101a'), 'Product_ID': 216, 'Category_ID': 123, 'SKU_ID': 2163, 'P_Name': 'Handkerchief Dr
ess', 'Ranking': 31, 'Selling_Price': 1135.0, 'Discount': 83.0, 'P_Description': 'Episdic tension headache', 'In_Stock': 'No',
'Size': 'XL', 'Colors': 'Fuscia'}
{'_id': ObjectId('6260b119dfd857023bd3101b'), 'Product_ID': 217, 'Category_ID': 123, 'SKU_ID': 2390, 'P_Name': 'Kaftan Dress',
'Ranking': 46, 'Selling_Price': 1367.0, 'Discount': 83.0, 'P_Description': 'Fissure of nipple', 'In_Stock': 'No', 'Size': 'L',
'Colors': 'Teal'}
{'_id': ObjectId('6260b119dfd857023bd3101c'), 'Product_ID': 218, 'Category_ID': 123, 'SKU_ID': 4509, 'P_Name': 'Kimono Dress',
'Ranking': 32, 'Selling_Price': 928.0, 'Discount': 54.0, 'P_Description': 'Retinoph prematr,stage 0', 'In_Stock': 'No', 'Size':
'S', 'Colors': 'Aquamarine'}
```

## Database Access via Python

A connection from MySQL database to Python was established using MySQL connectors for performing advanced data analysis  and visualization. The data from the database was imported to the Python environment, summarized, and performed visualizations using Plotly. The samples are shown below as follows.

### 1.Word cloud for customer reviews which shows the uber level perception of the people towards the brand:

### 2.Customer distribution across premium and regular types



**3.Stacked bar plot for Deals offered and Deals utilized by premium and regular customers**


## Summary and Recommendation

Our team has successfully implemented the Online E-Commerce relational database in MySQL and NoSQL databases. This will help Brivio to increase its market cap and revenue. The establishment of Python to database connection will also help in enabling analytical capabilities for the company, which in turn can be utilized to generate actionable insights.

Improvement for the database design would be creating a greater number of relationships between tables and providing much more features to get more conversion rate from customers. Implementing the relational design on MongoDB database has some huge shortcomings such as non-availability of reliability functions, cross-platform support, and poor usability but provides consistency in performance and scalability. Since SQL databases provide great benefits for transactional data whose structure doesn't change frequently and provides ACID compliance, it helps in having greater control and feasibility over the real-world applications than the NoSQL databases.