Data management for Analytics Project

Milestone-1

Real world business problem definition

A database model for an online shopping system must be able to hold data on the users who register on the website and place orders. The products that are for sale must also be listed, together with their prices and stock information. Before placing their order, customers must be allowed to add things they choose from the catalog to a shopping cart. Each consumer should then be able to keep track of their wish list, which is a collection of goods they are considering buying but haven't yet put in their shopping cart. The data model must be able to record the order information and payment when the customer confirms an order.

Conceptual Data Model for E-Commerce Website and it's Entities

The first step in creating a conceptual database model for an e-commerce website is to list all the system entities and their attributes. The following entities can be found in the majority of online purchasing data models:

- **Customer**: This entity stands for users of the online shopping platform who register and create accounts in order to place orders.
- **Product:** Describes the assortment of goods that may be purchased via the platform.
- **Category:** The groups of products that make up this category.
- **Order:** Purchase orders made by clients.
- **Items :** included in an order are referred to as order items.
- **Payment :** The sum the consumer pays after the order has been fulfilled.
- **Shipment** : Shipping details for an order, such as the delivery address and the tracking number, are referred to as the "shipment" details.
- **Cart**: The customer's electronic shopping bag or basket, which holds products until they are bought and added to an order.

- **Wishlist :** The customer's wish list contains the things they have selected for potential future purchases.

Extended Entity-Relationship

1. Customer and Order: Customer and Order have a one-to-many relationship, which means a single customer can place several orders, but only one customer is linked to each order.

2. Order and Order Item: The connection between an order and an order item is one to many. This implies that while a single order may include many items (products), each order item is associated with a particular order. You can keep track of the products inside each order thanks to this relationship.

3. Order Item and Product: Order Item and Product are related in a many-to-one manner. Although a product may be tied to more than one order item, each order item is associated with a single product. This connection makes it easier to connect the particular product in each order item.

4. Payment and Order: Order and Payment have a one-to-many relationship. One payment may be made for each order, however several orders may be connected to one payment. This connection makes it possible to keep track of payments made for each order.

5. Shipment and Order: There is also a one-to-many relationship between Shipment and Order. Each order can have one shipment, but a shipment can be associated with multiple orders. This relationship helps manage the shipping process for each order.

6. Product and Category: One product belongs to one category, but a category might have several products because there is a one-to-many relationship between the two. This association categorizes products for simpler management and browsing.

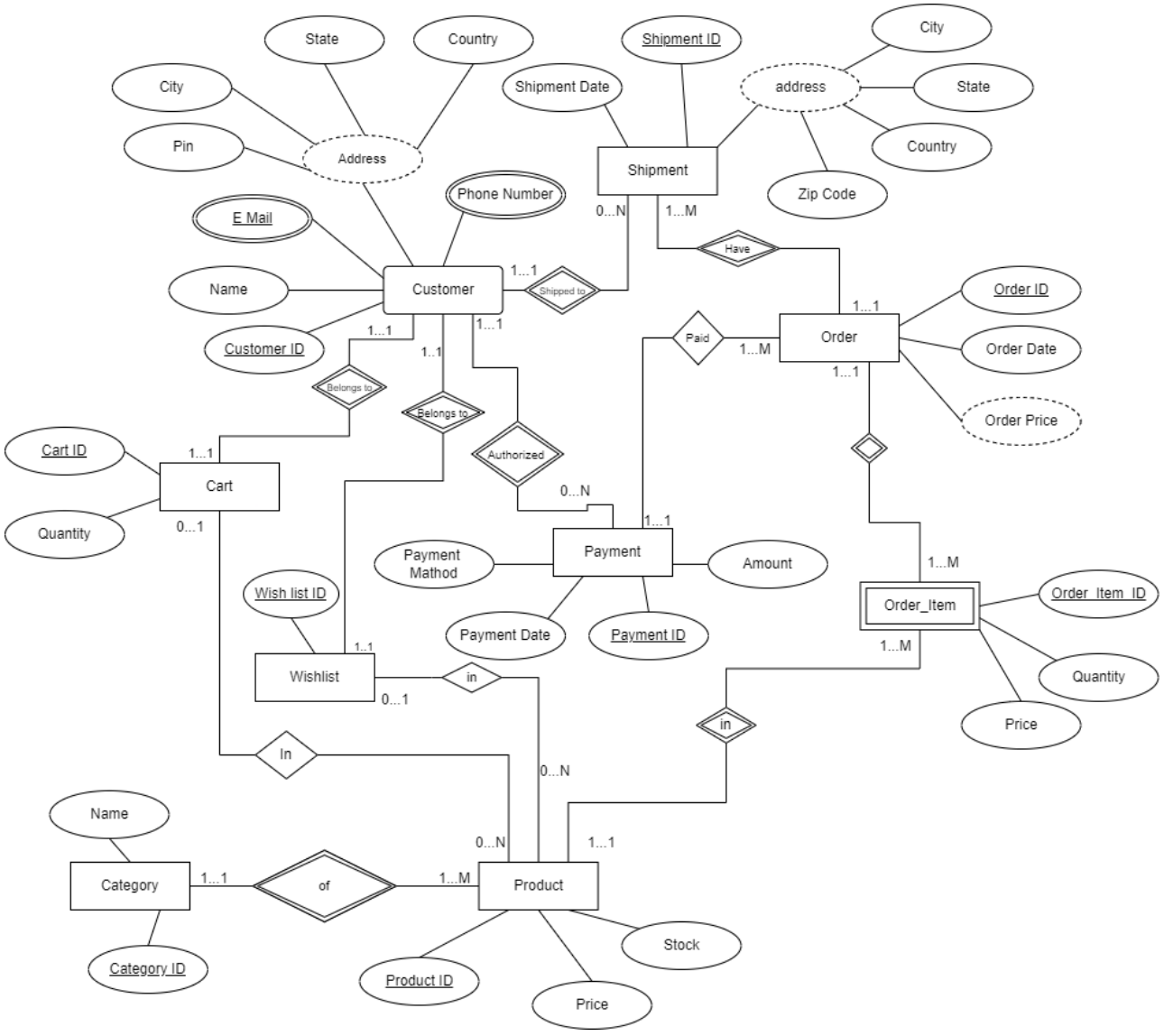
7. Customer, Cart, and Wishlist: Both Cart and Wishlist are dependent entities of Customer, which implies that they maintain a dependency relationship with the Customer entity. Each customer can have one shopping cart and one wish list.

8. Cart and Wishlist with Product: There are many-to-one links between the entities Cart and Wishlist and Product. This indicates that a single product may be linked to each instance of the

Cart and Wishlist. Customers can add multiple products to their cart and Wishlist, but each product is associated with a specific cart or Wishlist.

9. **Product and order Item :** A many-to-one relationship can be used to define the relationship between a product and an order item. Several order items can be related to the same product, which can be the case for multiple goods in various orders.

10. **Cart and Customer :** Each customer only have one shopping cart that they use to store products they intend to buy, hence the relationship between the cart and the customer can be regarded as a one-to-one relationship.



-Payment is a weak entity

ER Model of E commerce Website

Reference and Transactional data

Reference Data:

Information that is static or semi-static and doesn't change often is represented by reference data. Reference information on an e-commerce website could consist of the following:

1. Customer Information:
   - CustomerID (Primary Key)
   - Name
   - Email
   - Phone Number

2. Product Information:
   - ProductID (Primary Key)
   - Name
   - Price
   - Stock
   - CategoryID (Foreign Key to Category)

3. Category Information:
   - CategoryID (Primary Key)
   - Category Name

4. Payment Methods:
   - PaymentMethodID (Primary Key)
   - Payment Method Name

Transactional Data:

Transactional data is information that is updated often and keeps track of particular transactions or events. This contains information about orders, carts, wishlists, payments, shipments, and order items in an e-commerce context:

1. Order Information:
   - OrderID (Primary Key)
   - Order Date
   - Order Price
   - CustomerID (Foreign Key to Customer)

2. Order Items:
   - OrderItemID (Primary Key)
   - Quantity
   - Price
   - ProductID (Foreign Key to Product)
   - OrderID (Foreign Key to Order)

3. Payment Transactions:
   - PaymentID (Primary Key)
   - Payment Date
   - Amount
   - PaymentMethodID (Foreign Key to Payment Methods)
   - CustomerID (Foreign Key to Customer)
   - OrderID (Foreign Key to Order)

4. Shipment Information:
   - ShipmentID (Primary Key)
   - Shipment Date
   - AddressID (Foreign Key to Address)

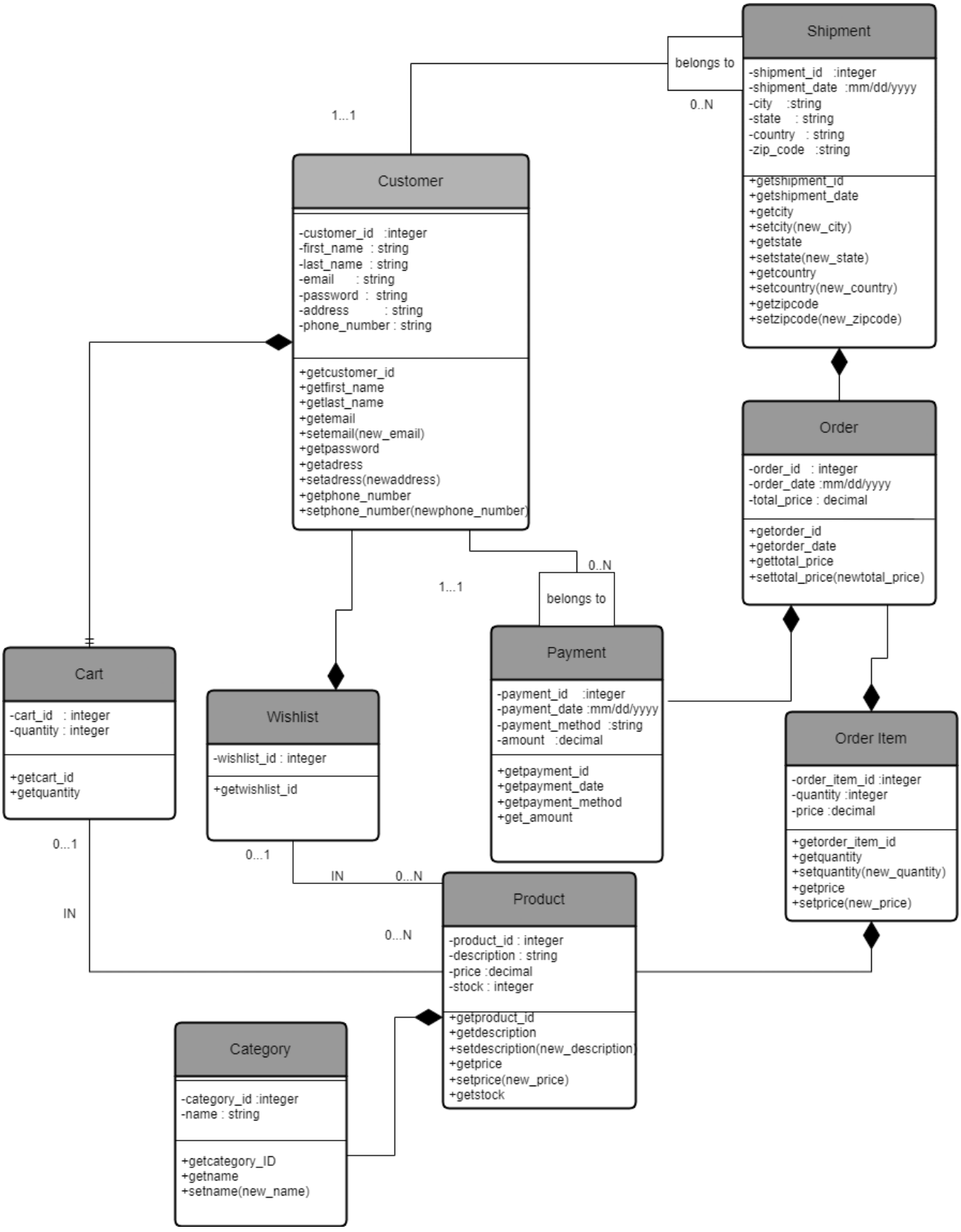
5. Cart Information:
   - CartID (Primary Key)
   - Quantity
   - ProductID (Foreign Key to Product)
   - CustomerID (Foreign Key to Customer)

6. Wishlist Information:
   - WishlistID (Primary Key)
   - ProductID (Foreign Key to Product)
   - CustomerID (Foreign Key to Customer)

7. Address Information:
   - AddressID (Primary Key)
   - Pin
   - City
   - State
   - Country
   - Zip code
   - CustomerID (Foreign Key to Customer)

As clients engage with the e-commerce website, particular transactional data can be recorded and kept in each of these tables. This structure enables you to efficiently support the operations and functionality of the online shopping system by storing and managing both reference data and transactional data.

Milestone-2



UML diagram of Proposed E commerce website model

A Relational model from Conceptual model:

Customer (CustomerID, Name, Email, Phone Number, AddressID)

Address (AddressID, pin code, city name, state)
Primary key: AddressID

Shipment (ShipmentID, CustomerID , OrderID ,Shipment Date, Address(City, State, Country, Zip code))

Forign key: CustomerID

Forign key: OrderID

Order (OrderID, Order Date, Order Price, Order_itemID, PaymentID)

Forign key: Order_itemID

Foreign key : PaymentID

Order_item (Order_itemID, Quantity, Price, ProductID)

Forign key: ProductID

Product (ProductID, Price, Stock, CategoryID)

Forign key: ProductID

Category (CategoryID, Name)

Payment (PaymentID, Payment method, Payment Date, Amount, CustomerID)

Primary key: CustomerID

Cart (CartID, Quantity, ProductID)

Forign key: ProductID

Wishlist (WishlistID, ProductID)

Forign key: ProductID

Up to 3.5 Normalized form of above relational model as given below

1. Customer (CustomerID, Name, Email, Billing phone,cityname)

2. Address (AddressID, Pin, City, State, Country, Zip code)

3. Shipment (ShipmentID, OrderID,AddressID)
   - Foreign key: CustomerID
   - Foreign key: OrderID
   - Foreign key: AdressID
4. Order (customerID,Product,ProductID)

5. Payment (PaymentID, Payment method,Payment value,)

6. Cart (CustomerID, ProductID)
   - *Foreign key: CustomerID*
   - *Foreign key: ProductID*

7. Wishlist (CustomerID, ProductID)
   - *Foreign key: CustomerID*
   - *Foreign key: ProductID*

8. Product (ProductID, Product, Brand, Category)

In this normalized structure:

- Customer data can be found in the "Customer" table.
- Addresses are kept in the "Address" table and are connected to clients via CustomerID.
- The AddressID foreign key is used to retrieve shipment details from the "Address" table.
- The transfer of order items to their own "OrderItem" database simplifies orders.
- Payment data is associated with specific clients.
- Wishlists and shopping carts are connected to clients.
- Product data is kept apart from the "Product" table, which has a foreign key called CategoryID to link products to categories.
- Distinct categories are kept up to date.

Milestone -3

1. Total sales by product category:

```
SELECT p.category, SUM(o.quantity * p.costprice) AS total_sales
FROM product p
JOIN orders o ON p.productID = o.productID
GROUP BY p.category limit 10;
```

category	total_sales
WATCHES	24552.37142888
Men Footwear	33047.68607389001
SUNGLASSES	8499.600120170002
Bags	8482.391596820004
Women Footwear	4116.311884520001
Women Apparel	11046.8009603
Sports Equipment	2465.21932767
Jewellery	2530.04621859
Home	1129.87238915
Men Apparel	871.1668668

Total Sales by Product Category: This query aims to identify the product categories that generate the most sales by calculating the total sales for each category. This information can be used to make informed decisions about inventory management, product development, and marketing strategies.

---

2. Average order value by customer:

```
SELECT c.cutomerID, AVG(o.quantity * p.costprice) AS average_order_value
FROM customer as c
JOIN orders as o ON c.cutomerID = o.cutomerID
JOIN product p ON o.productID = p.productID
GROUP BY c.cutomerID limit 10;
```

	cutomerID	average_order_value
▶	10049	115.65342138
	10068	100.55462184
	1023	121.84753902
	103045	59.24921968
	10376	81.23169268
	103765	130.6429772
	103865	59.24369748
	105015	70.0914766
	106115	110.50912365
	10674	45.96662665

Average Order Value by Customer: This query calculates the average order value for each customer, providing insights into customer purchasing behavior and spending patterns. This information can be used to identify high-value customers and tailor marketing strategies accordingly.

Top 10 customers by order count:

```
select o.cutomerID,p.product, o.quantity from orders as o
join product as p on p.productID = o.productID
order by quantity desc
LIMIT 10;
```

	cutomerID	product	quantity
▶	23247	E09525M1 White/White Analog Watch	4
	87045	Navy Blue Georgette Brocade Neck & Dupatta S...	4
	20422	E09525M1 White/White Analog Watch	4
	185635	White/ White Analog Watch	4
	277	E09525M1 White/White Analog Watch	4
	112875	Navy Blue Georgette Brocade Neck & Dupatta S...	4
	361612	E09525M1 White/White Analog Watch	4
	360317	E09525M1 White/White Analog Watch	4
	183565	The Overplay Vii White Basketball Shoes	4
	202575	25M1 White/White Analog Watch	4

202575

Top 10 Customers by Order Count: This query identifies the top 10 customers who have placed the most orders, revealing the most loyal and engaged customer base. This information can be used to focus customer retention efforts and develop targeted loyalty programs.

Average order value by payment method:

```
SELECT pm.payment_method, AVG(o.quantity * pm.paymentValue) AS average_order_value
FROM payment pm
JOIN orders o ON pm.paymentID = o.paymentID
JOIN product p ON o.productID = p.productID
GROUP BY pm.payment_method;
```

payment_method	average_order_value
Prepaid	185.67724337622778
COD	164.19305122130893

Average Order Value by Payment Method: This query analyzes the average order value associated with different payment methods, providing insights into customer preferences and the effectiveness of payment options. This information can be used to optimize payment gateways and enhance customer checkout experiences.

Top 10 most abandoned items in carts:

```
SELECT c.productID, p.product, COUNT(*) AS abandoned_count
FROM cart c
LEFT JOIN orders o ON c.productID = o.productID
JOIN product p ON c.productID = p.productID
WHERE o.productID IS NULL
GROUP BY c.productID, p.product
ORDER BY abandoned_count DESC
LIMIT 10;
```

productID	product	abandoned_count
P178	Erik Original Twill Jos Beige Casual Trousers	18

Top 10 Most Abandoned Items in Carts: This query identifies the products that are most frequently added to carts but not purchased, revealing potential conversion issues or product abandonment patterns. This information can be used to improve product recommendations, optimize product listings, and enhance cart abandonment recovery strategies.

Customer lifetime value (CLTV) for each customer:

```
SELECT c.cutomerID, SUM(o.quantity * p.costprice) AS total_spent
FROM customer c
JOIN orders o ON c.cutomerID = o.cutomerID
JOIN product p ON o.productID = p.productID
GROUP BY c.cutomerID order by total_spent desc limit 10;
```

cutomerID	total_spent
122515	306.67226892
142145	306.67226892
132605	282.15462184
202405	282.15462184
160995	264.7265306
215365	264.7265306
31978	264.7265306
361562	264.7265306
185635	264.7265306
22617	264.7265306

Customer Lifetime Value (CLTV) for Each Customer: This query identifies the top 10 customers with the highest lifetime value, providing insights into the most profitable and loyal customer segments. This information can be used to develop targeted customer retention strategies and optimize marketing campaigns.

Profit from each product

```
select p.productID,p.product, pm.paymentValue-p.costprice as Profit from payment as pm
inner join orders as o on o.paymentID = pm.paymentID
inner join product as p on o.productID = p.productID
limit 10;
```

productID	product	Profit
P101	E09525M1 White/White Analog Watch	31.228465039999996
P102	Bpb-0015-C Silver/White Analog Watch	20.5613697
P103	Street Tuneo Mid Black Sneakers	28.352378319999993
P104	Mercurial Veloce Fg Blue Football Shoes	23.959831360000003
P105	Grey Sunglasses	37.39446988
P106	Rb3025 004 Green Sunglasses	20.141157779999993
P107	Brown Handbag	25.32475145
P108	Ventilator Hls Grey Running Shoes	20.816901610000002
P109	Puma Sneakers Black	106.98690613
P110	Air Max Compete Tr White Running Shoes	30.861821850000005

Profit from Each Product: This query calculates the profit generated by each product by subtracting the cost price from the payment value. This information can be used to identify the most profitable products and prioritize inventory management and marketing efforts accordingly.

Product conversion rate from wishlist to purchase:

```
SELECT p.productID, p.product, COUNT( w.productID) AS wishlist_count, COUNT(DISTINCT
o.orderID) AS purchase_count
FROM product p
LEFT JOIN whishlist w ON p.productID = w.productID
LEFT JOIN orders o ON p.productID = o.productID
GROUP BY p.productID, p.product
order by wishlist_count desc limit 10;
```

productID	product	wishlist_count	purchase_count
P164	Embroidered Blue Saree - Mksp	572	13
P168	White/ Balck Chronograph	504	14
P167	Downing Street 04 Blue Handbag	495	15
P160	9166Sl04 Black Analog Watch	473	11
P172	Eureka Brussels Nest of Tables	468	12
P159	Silver/ Silver Analog Watch	465	15
P162	Embroidered Georgette Multi Saree	455	13
P171	Gold/White Analog Watches	440	11
P157	Th1790787/D Sport Black /White Chronograph-...	420	12
P155	Bpb-1004-C Silver/Black Analog Watch	420	12

Product Conversion Rate from Wishlist to Purchase: This query analyzes the conversion rate of products from wishlists to purchases, providing insights into the effectiveness of product recommendations and customer engagement strategies. This information can be used to optimize product listings, improve product discovery, and enhance cart abandonment recovery techniques.

Determine the top 5 brands with the highest total sales.

```
SELECT p.brand, sum(o.quantity * pm.paymentValue) as Total_Sales_Value FROM product as p
inner join orders o on o.productID = p.productID
inner join payment as pm on pm.paymentID = o.paymentID
GROUP BY p.Brand
ORDER BY Total_Sales_Value DESC
Limit 10;
```

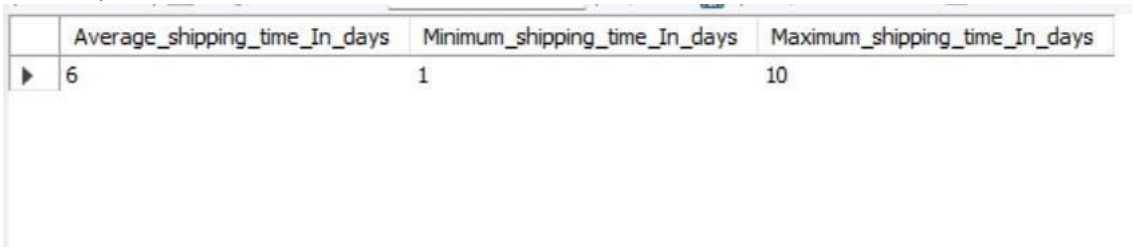
	brand	Total_Sales_Value
▶	NIKE	24616.011975479993
	RAY BAN	13504.838323139995
	HIDESIGN	12732.203592350003
	PUMA	11576.646706350004
	REEBOK	10014.011976120004
	MARC ECKO	9604.34730552
	ADIDAS	9168.095808199998
	CASIO	6607.401197710002
	PLAYBOY	5261.628742569999
	ETHNIC CLOSET	4694.586826330001

Top 5 Brands with the Highest Total Sales: This query identifies the top 5 brands that generate the most sales, revealing the most successful and popular brands. This information can be used to negotiate better deals with suppliers, develop targeted marketing campaigns, and enhance brand positioning strategies.

---

2. Determine the average shipping time for orders placed during a specific time period

```
SELECT CEILING(AVG(DATEDIFF(s.shipmentDate, o.orderDate))) AS
Average_shipping_time_In_days,
CEILING(min(DATEDIFF(s.shipmentDate, o.orderDate))) as Minimum_shipping_time_In_days,
CEILING(max(DATEDIFF(s.shipmentDate, o.orderDate))) as Maximum_shipping_time_In_days
FROM orders o
JOIN shipments s ON o.orderID = s.orderID;
```



Average_shipping_time_In_days	Minimum_shipping_time_In_days	Maximum_shipping_time_In_days
6	1	10

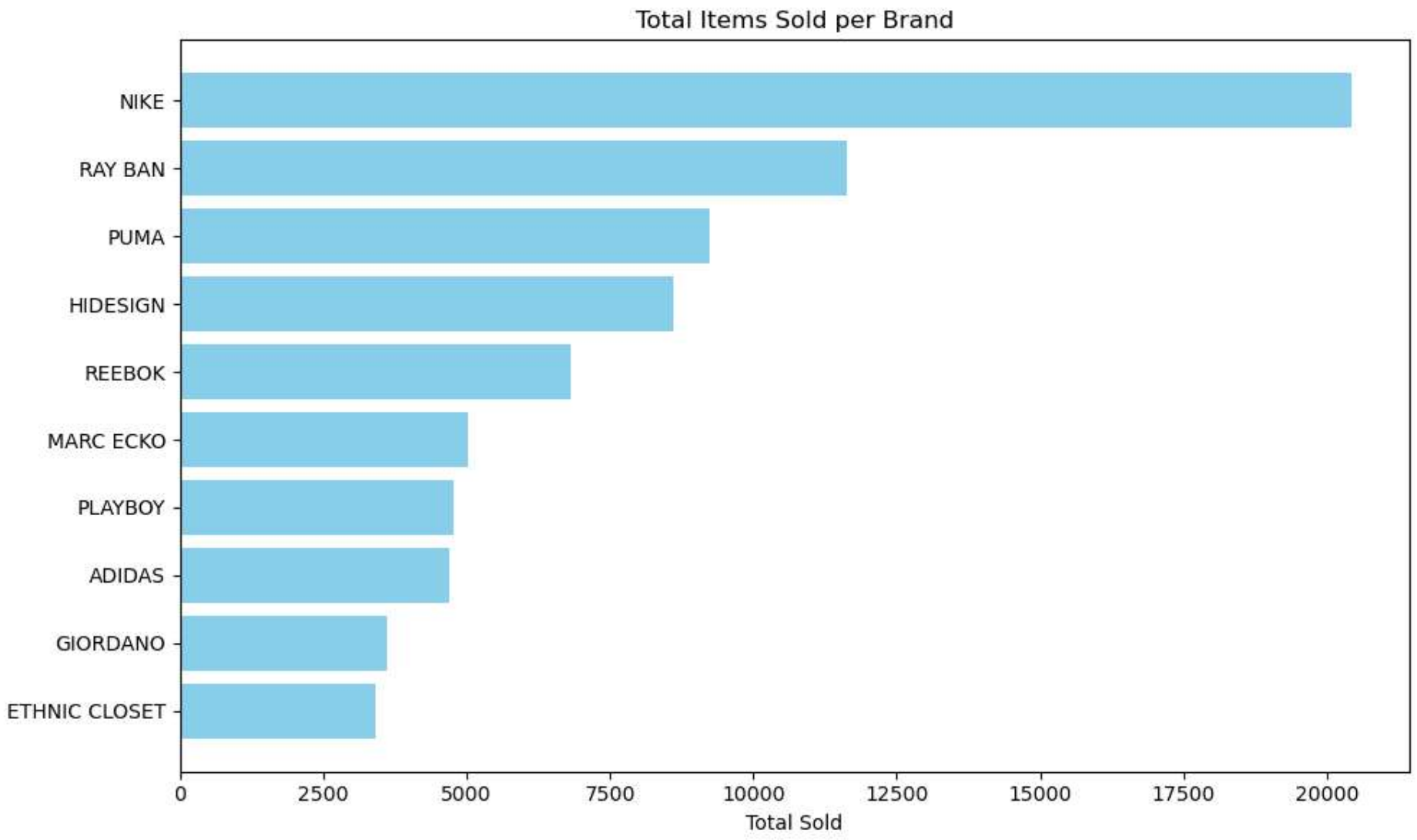
Average Shipping Time for Orders: This query calculates the average, minimum, and maximum shipping time for orders, providing insights into order fulfillment efficiency and potential delivery issues. This information can be used to optimize shipping processes, set realistic delivery expectations, and improve customer satisfaction.

Filtering Data

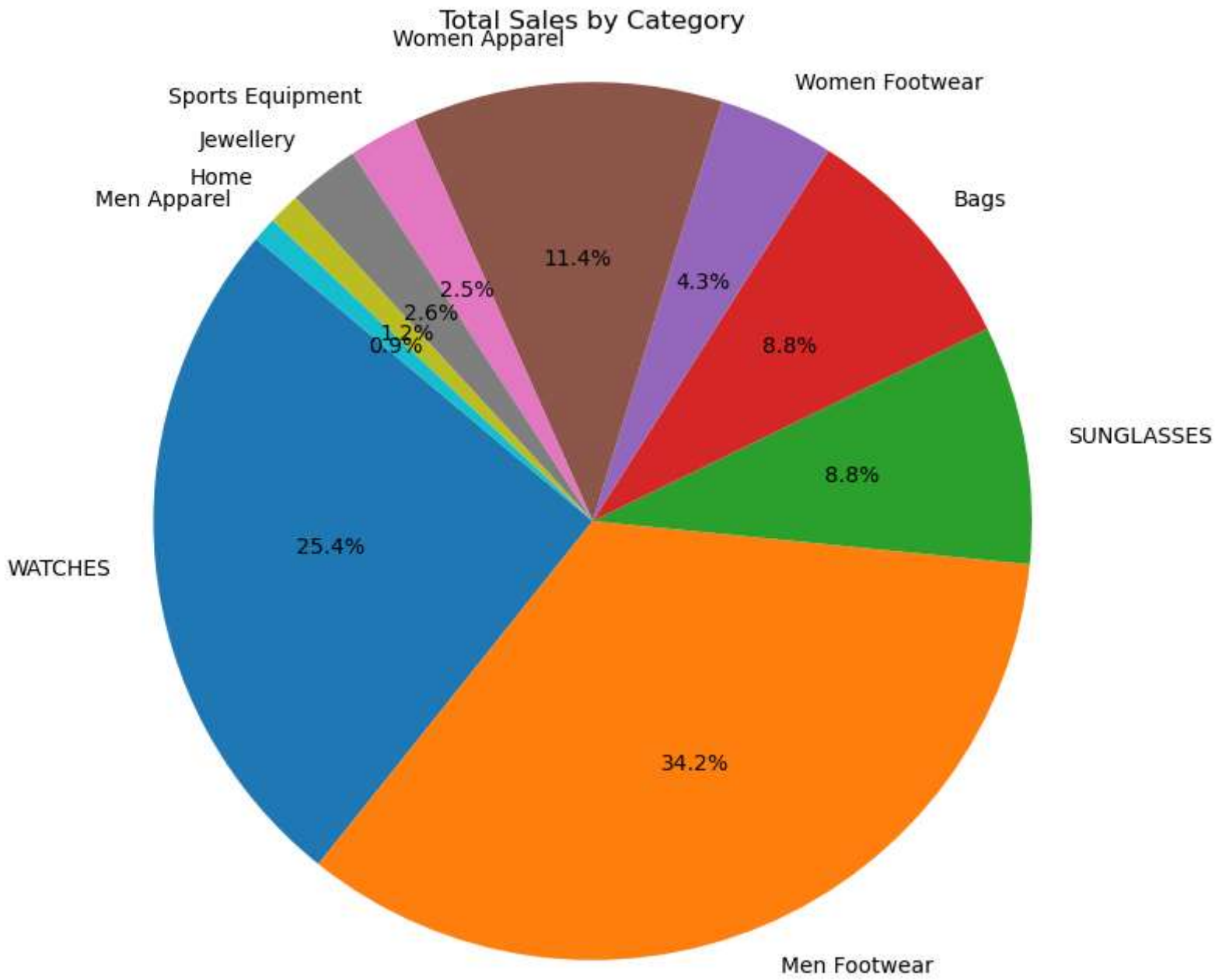
In [3]:
```
import pandas as pd

df = pd.read_csv("C:\\Users\\Raksh\\Downloads\\Online-Shoppers-374500.csv")

df_cleaned = df.dropna()

df_cleaned.to_csv("C:\\Users\\Raksh\\Downloads\\sitedata.csv", index=False)
```

Data Visualizatio

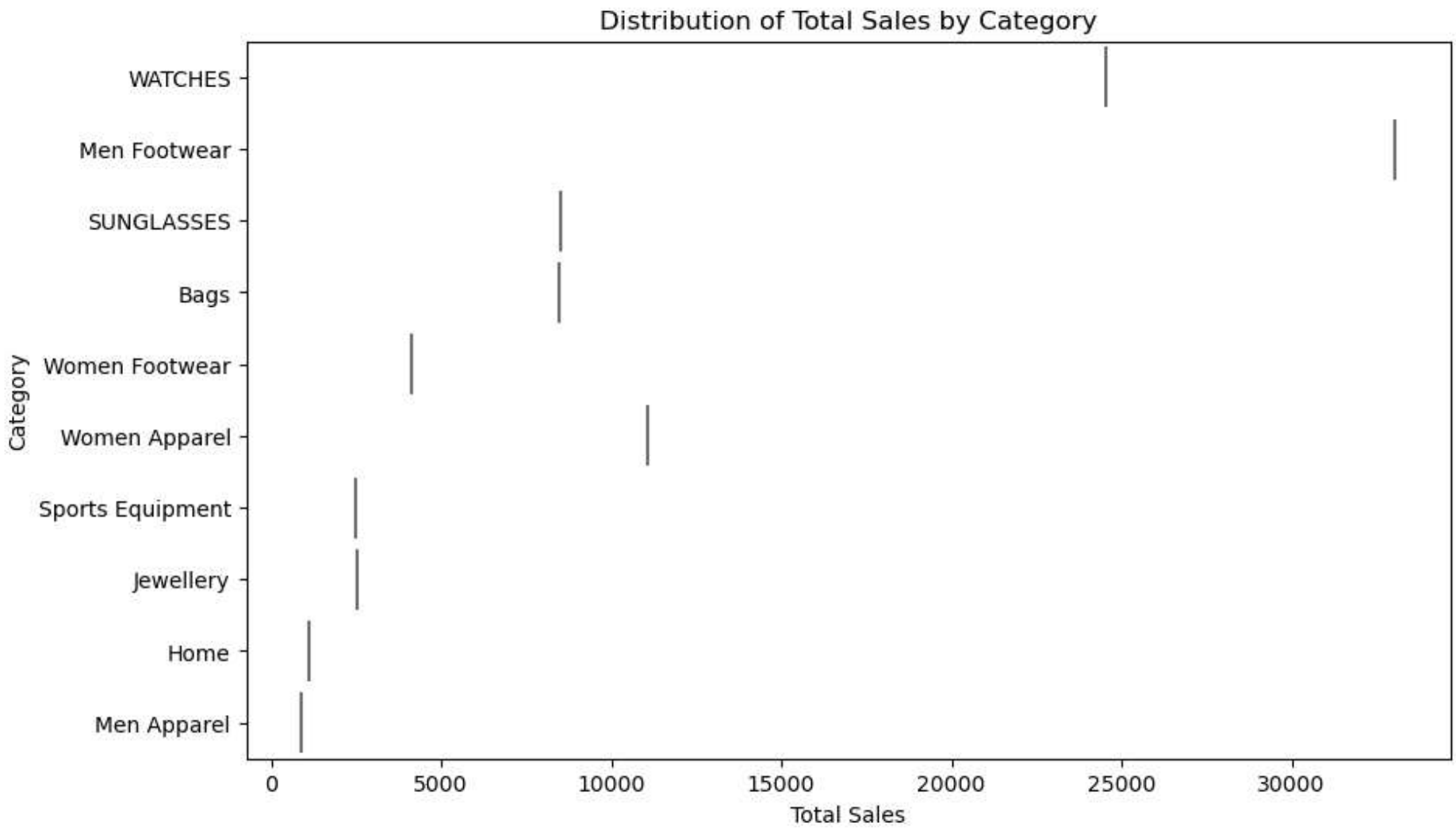
In [4]:
```
import matplotlib.pyplot as plt

data = pd.read_csv("F:\\data\\graph\\topsold.csv")
brands = [item["_id"] for item in data]
total_sold = [item["totalSold"] for item in data]

plt.figure(figsize=(10, 6))
plt.barh(brands, total_sold, color='skyblue')
plt.xlabel('Total Sold')
plt.title('Total Items Sold per Brand')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



In [5]:
```
import matplotlib.pyplot as plt

data = data = pd.read_csv("F:\\data\\graph\\topsoldcate.csv")


categories = list(data.keys())
sales = list(data.values())

plt.figure(figsize=(8, 8))
plt.pie(sales, labels=categories, autopct='%1.1f%%', startangle=140)
plt.axis('equal')
plt.title('Total Sales by Category')
plt.show()
```

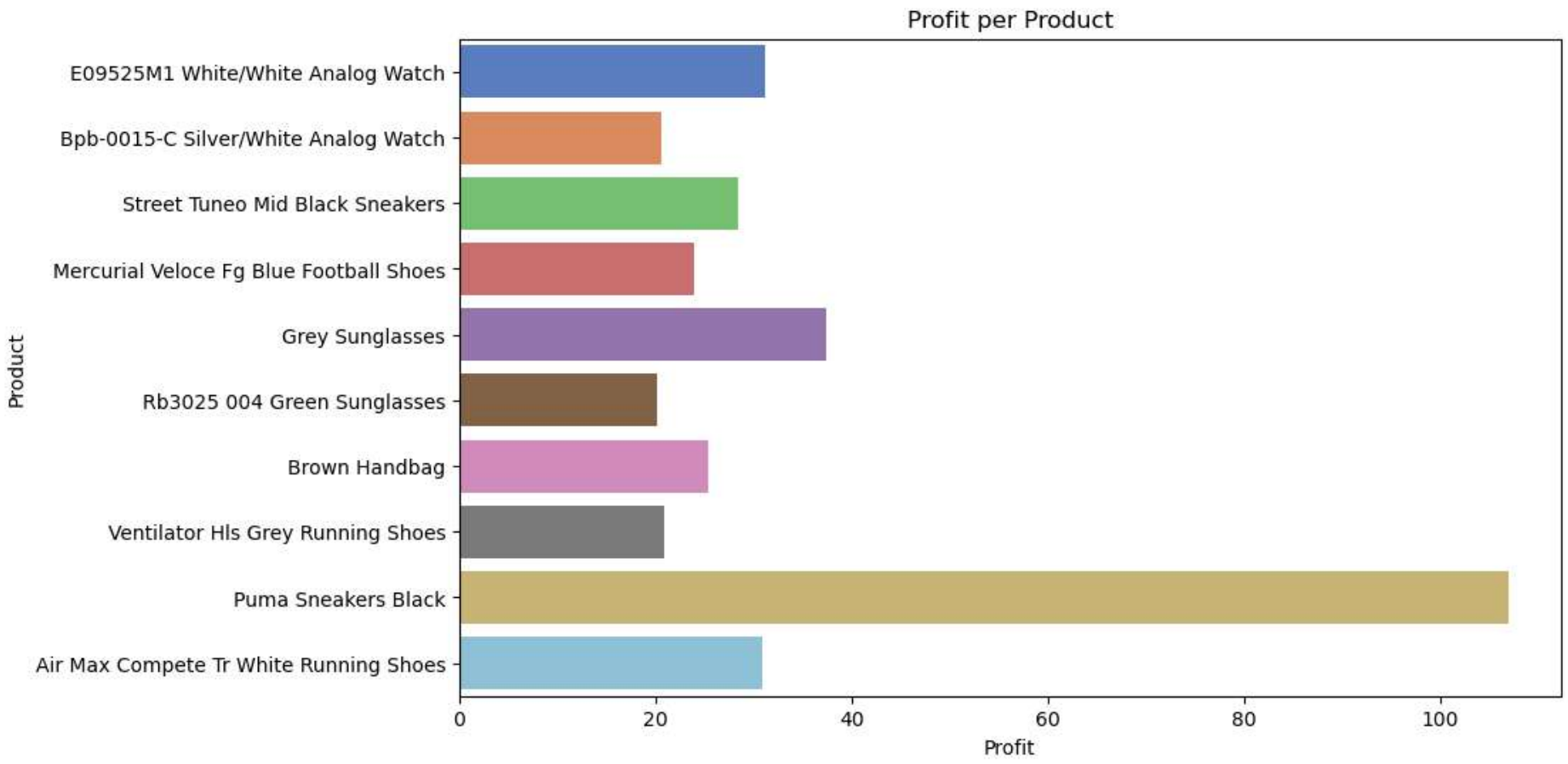


In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt

data = data = pd.read_csv("F:\\data\\graph\\cate_dist.csv")

df = pd.DataFrame(data)

plt.figure(figsize=(10, 6))
sns.violinplot(x='Total_Sales', y='Category', data=df, palette='pastel')
plt.title('Distribution of Total Sales by Category')
plt.xlabel('Total Sales')
plt.ylabel('Category')
plt.show()
```



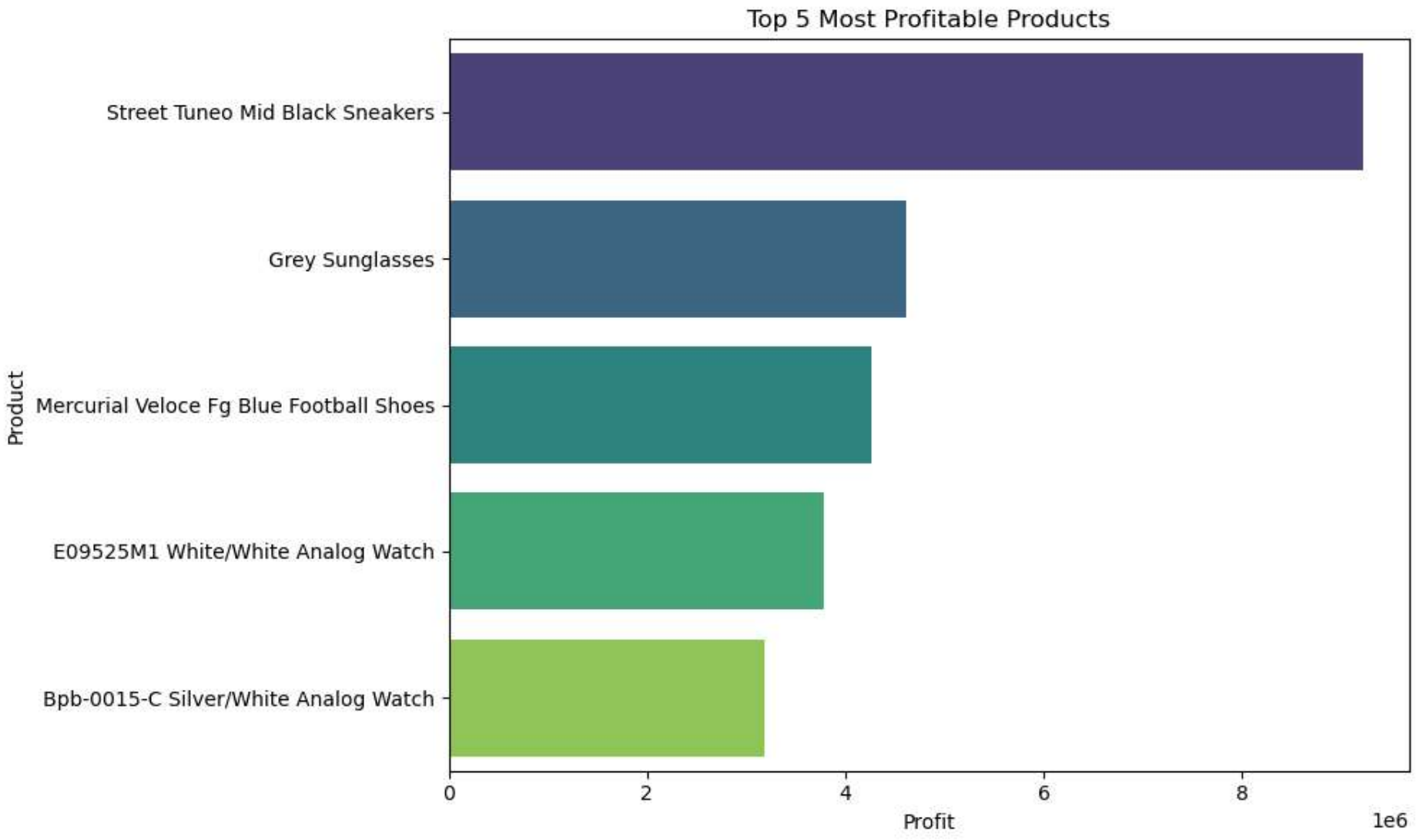
In [7]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data =data = pd.read_csv("F:\\data\\graph\\ppp.csv")

df = pd.DataFrame(data)

plt.figure(figsize=(10, 6))
sns.barplot(x='Profit', y='Product', data=df, palette='muted')
plt.title('Profit per Product')
plt.xlabel('Profit')
plt.ylabel('Product')
plt.show()
```

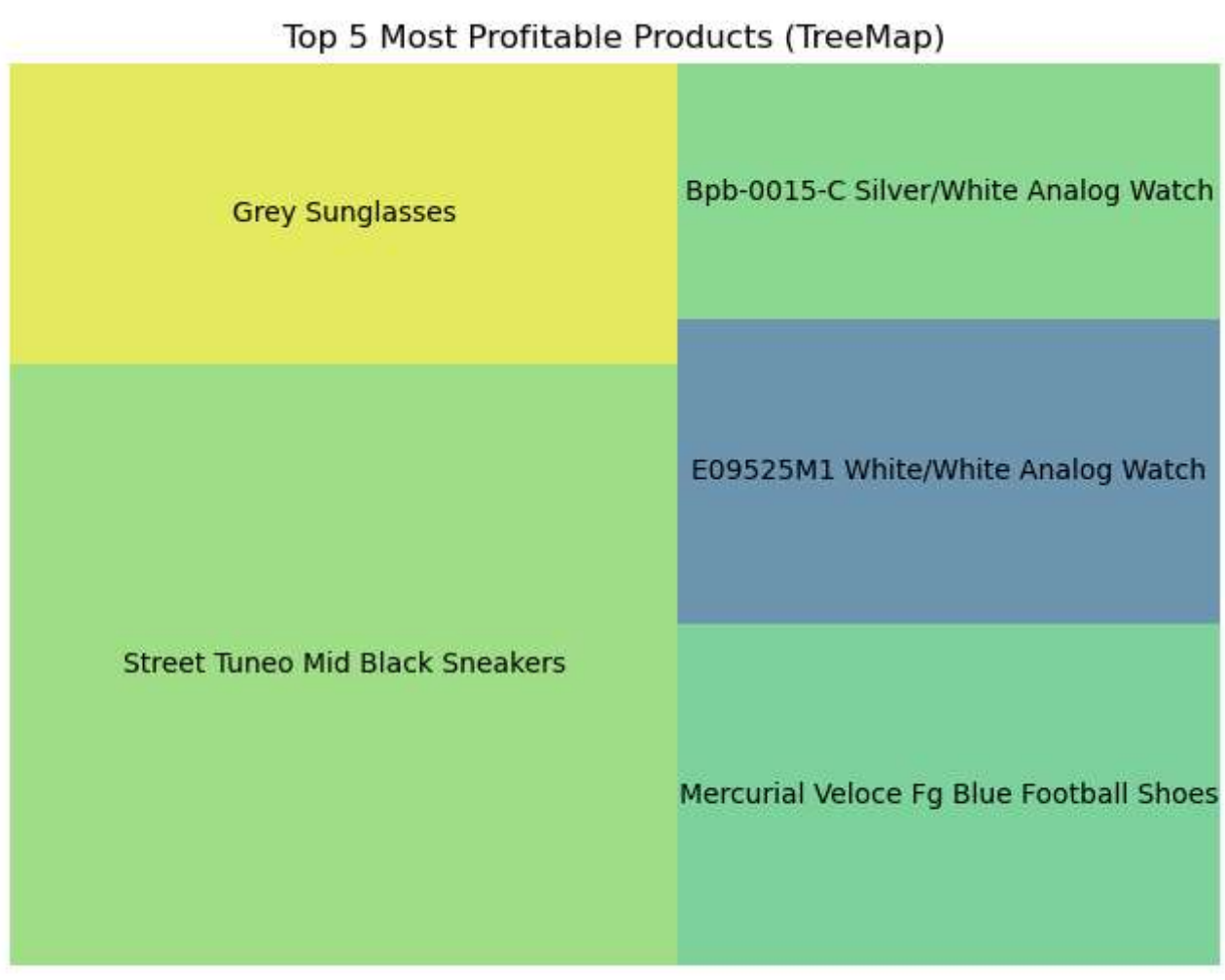


In [9]:

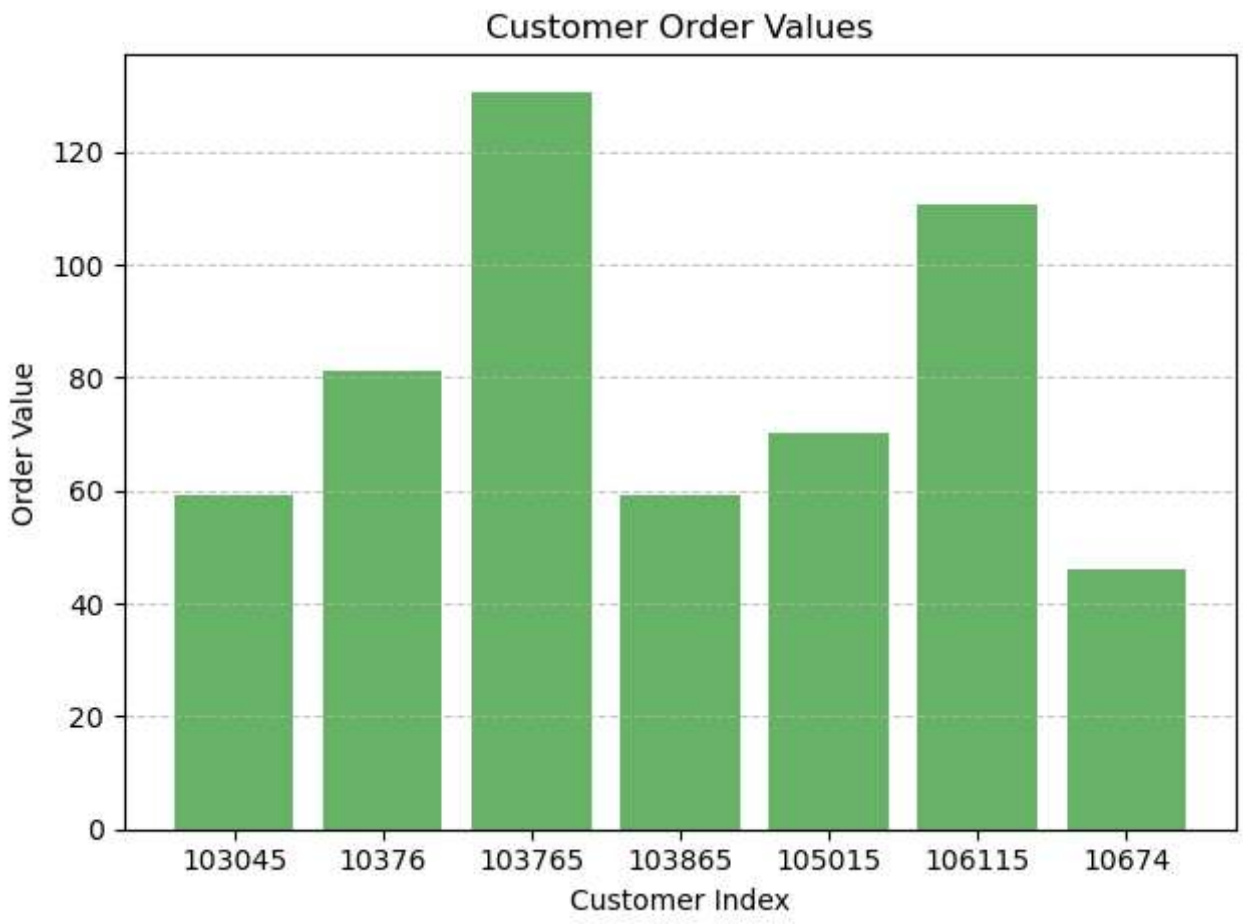
```
import seaborn as sns
import matplotlib.pyplot as plt

data =data = pd.read_csv("F:\\data\\graph\\5pp.csv")


# Sorting data based on profit in descending order and selecting top 5
sorted_data = sorted(data, key=lambda x: x['profit'], reverse=True)[:5]

# Extracting data into separate lists
products = [item["_id"] for item in sorted_data]
profits = [item["profit"] for item in sorted_data]

plt.figure(figsize=(10, 6))
sns.barplot(y=products, x=profits, palette='viridis')
plt.title('Top 5 Most Profitable Products')
plt.xlabel('Profit')
plt.ylabel('Product')
plt.tight_layout()
plt.show()
```



In [13]:
```python
import matplotlib.pyplot as plt
import squarify  # Library for creating TreeMap plots

data = [
    {"_id": "E09525M1 White/White Analog Watch", "profit": 3787099},
    {"_id": "Bpb-0015-C Silver/White Analog Watch", "profit": 3180799},
    {"_id": "Street Tuneo Mid Black Sneakers", "profit": 9227820},
    {"_id": "Mercurial Veloce Fg Blue Football Shoes", "profit": 4266243},
    {"_id": "Grey Sunglasses", "profit": 4607683},
    # ... Other data ...
]

# Sorting data based on profit in descending order and selecting top 5
sorted_data = sorted(data, key=lambda x: x['profit'], reverse=True)[:5]

# Extracting data into separate lists
products = [item["_id"] for item in sorted_data]
profits = [item["profit"] for item in sorted_data]

plt.figure(figsize=(8, 6))
squarify.plot(sizes=profits, label=products, alpha=0.7)
plt.axis('off')  # Turn off axis
plt.title('Top 5 Most Profitable Products (TreeMap)')
plt.show()
```



In [24]:
```
import matplotlib.pyplot as plt

customer_ids = [103045, 10376, 103765, 103865, 105015, 106115, 10674]
order_values = [59.24921968, 81.23169268, 130.6429772, 59.24369748, 70.0914766, 110.50912365, 45.96662665]

plt.bar(range(len(customer_ids)), order_values, color='green',alpha=0.6)
plt.xlabel('Customer Index')
plt.ylabel('Order Value')
plt.title('Customer Order Values')
plt.xticks(range(len(customer_ids)), customer_ids)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Data Management for Analytics

Data Model for E-Commerce Website

Group 5

- Rohit Reddy
- Rakshith Dharmappa

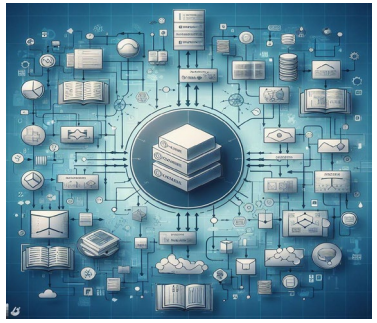
Agenda

01. Introduction

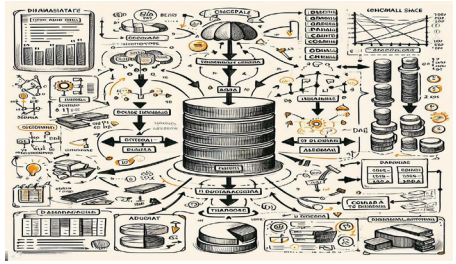
The database captures user details, orders, products, carts, wishlists, order specifics, and payment information for an online shopping platform.

02. Entities

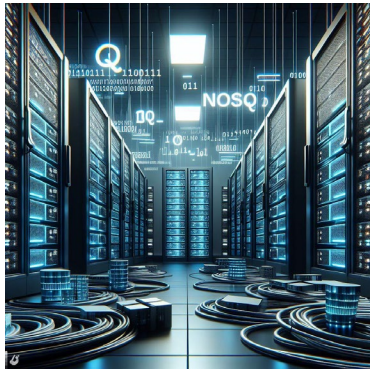
03. ER and UML



04. Conceptual DB and Its Normalized form



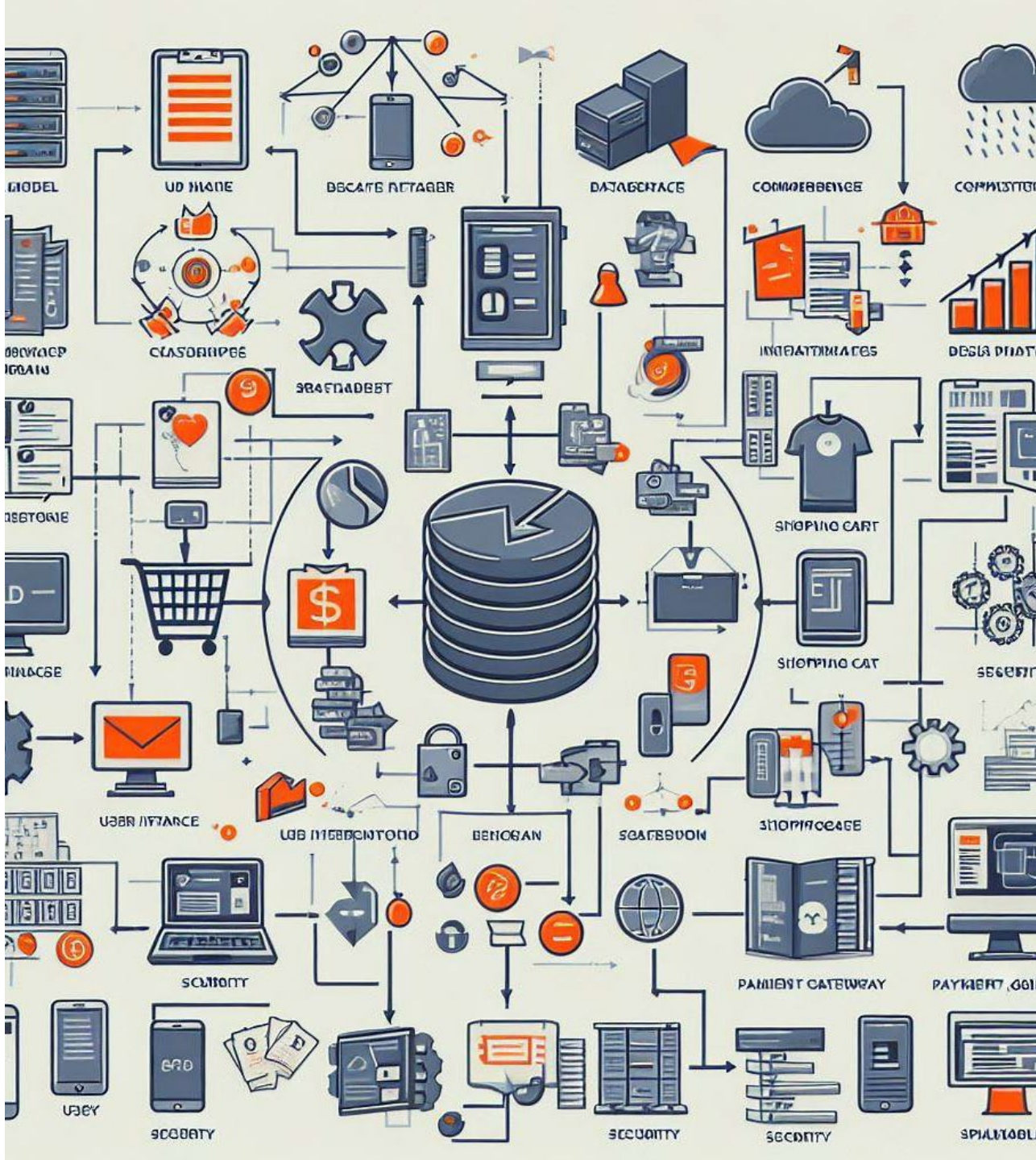
05. SQL & NoSQL



Problem Statement:-

Introduction-

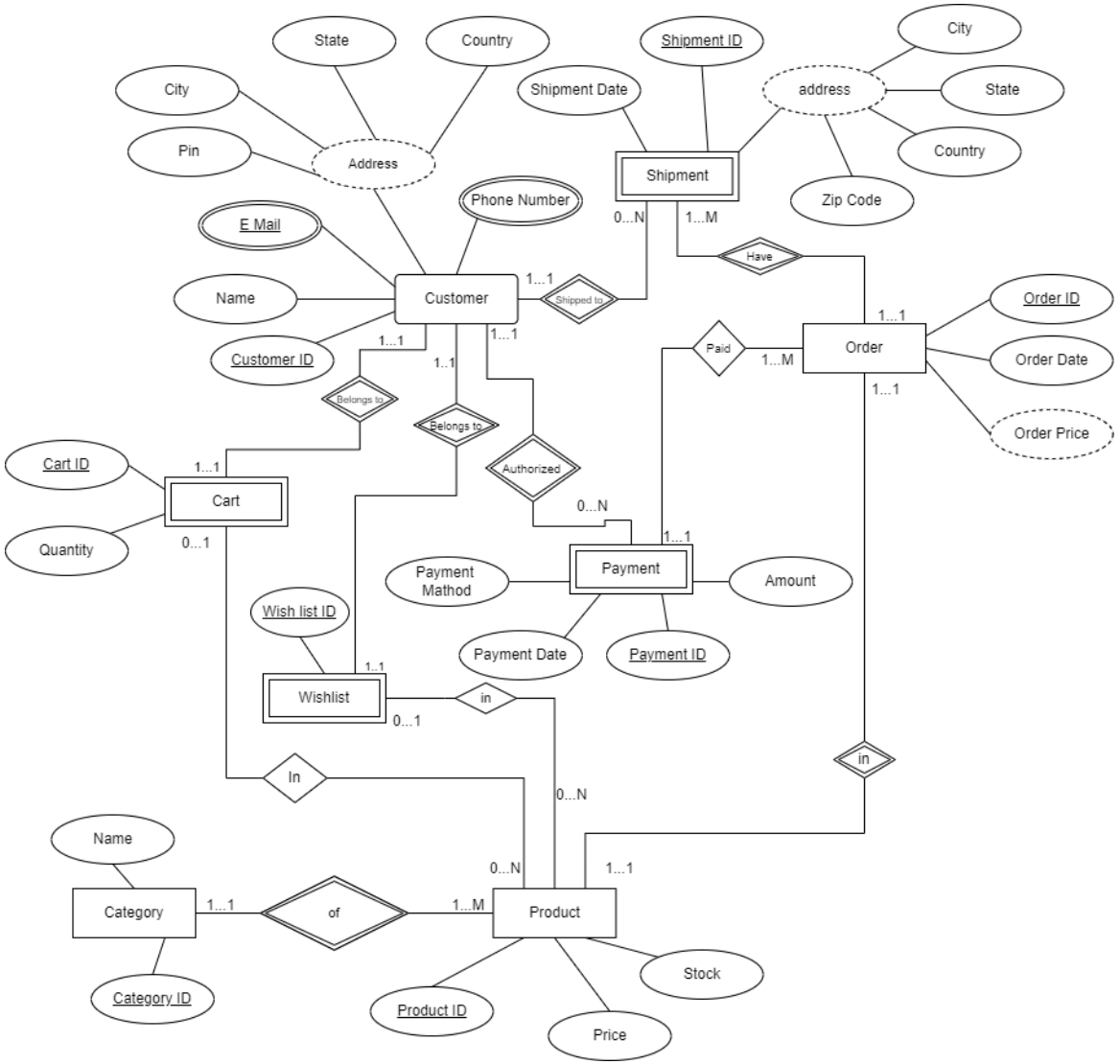
Welcome to our E-commerce Database Model Presentation. Our system is tailored to manage user details, orders, product listings with prices and stocks, carts, Wishlist, and payment information for seamless online shopping. The model showcases entities like Customers, Products, Orders, Payments, Shipments, and their relationships. It ensures efficient operations by allowing users to register, place orders, add products to carts or wishlists, and finalize payments. Join us as we explore how our model optimizes the online shopping experience.

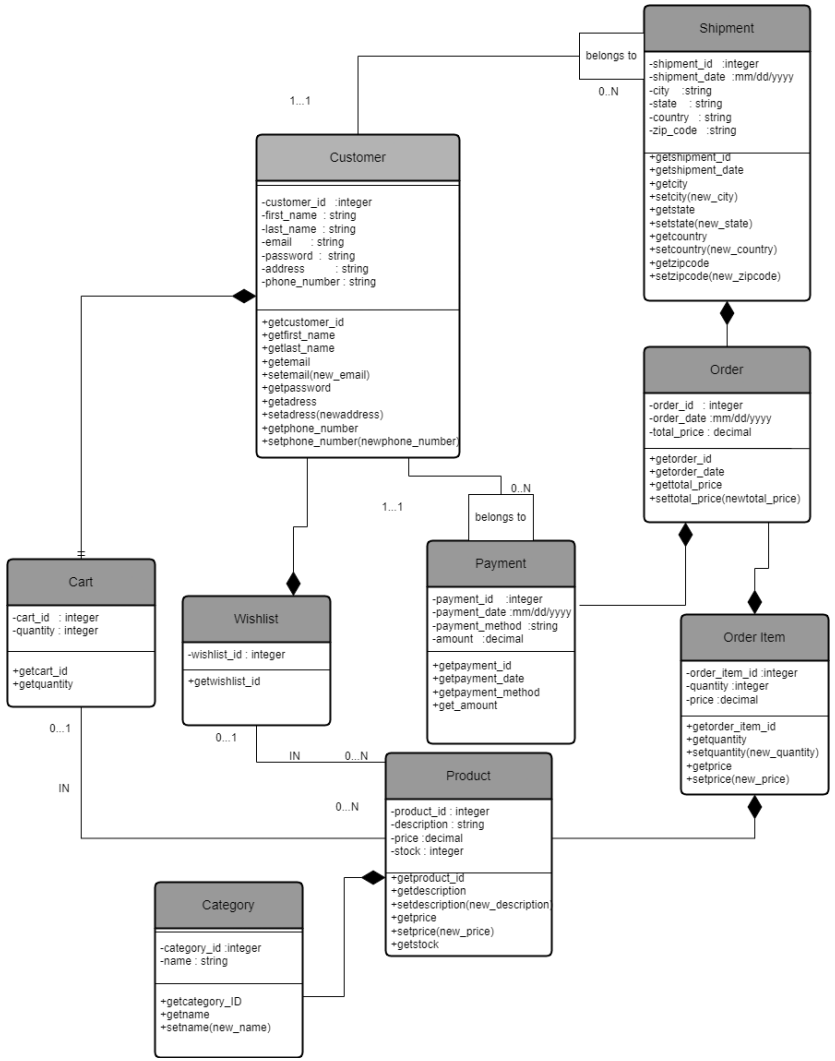


02. Entities

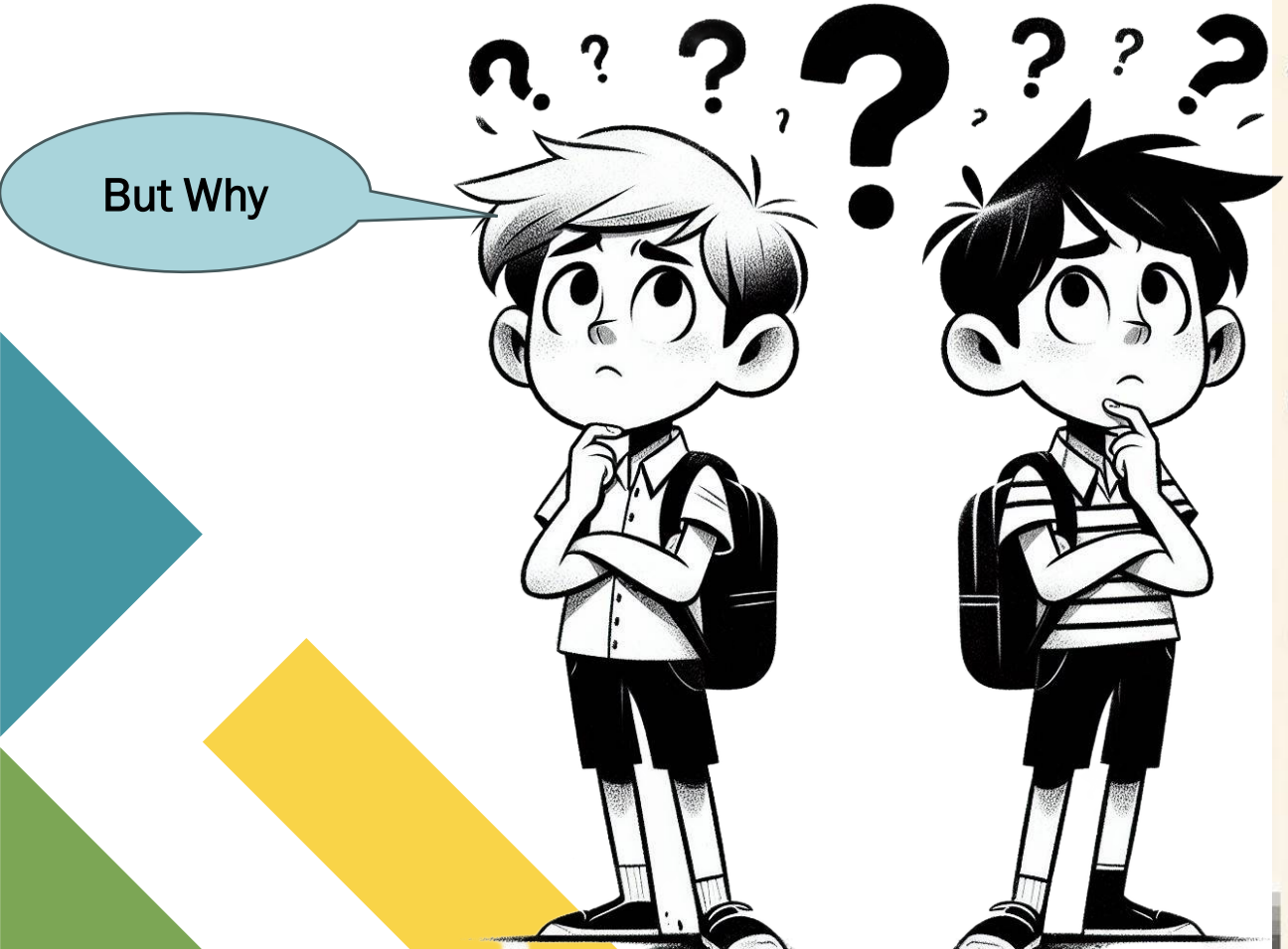
Customer	• This entity stands for users of the online shopping platform who register and create accounts in order to place orders.
Product	• Describes the assortment of goods that may be purchased via the platform.
Order	• Purchase orders made by clients.
Items	• Included in an order are referred to as order items.
Payment	• The sum the consumer pays after the order has been fulfilled.
Shipment	• Shipping details for an order, such as the delivery address and the tracking number, are referred to as the "shipment" details
Cart	• The customer's electronic shopping bag or basket, which holds products until they are bought and added to an order
Wishlist	• The customer's wish list contains the things they have selected for potential future purchases.

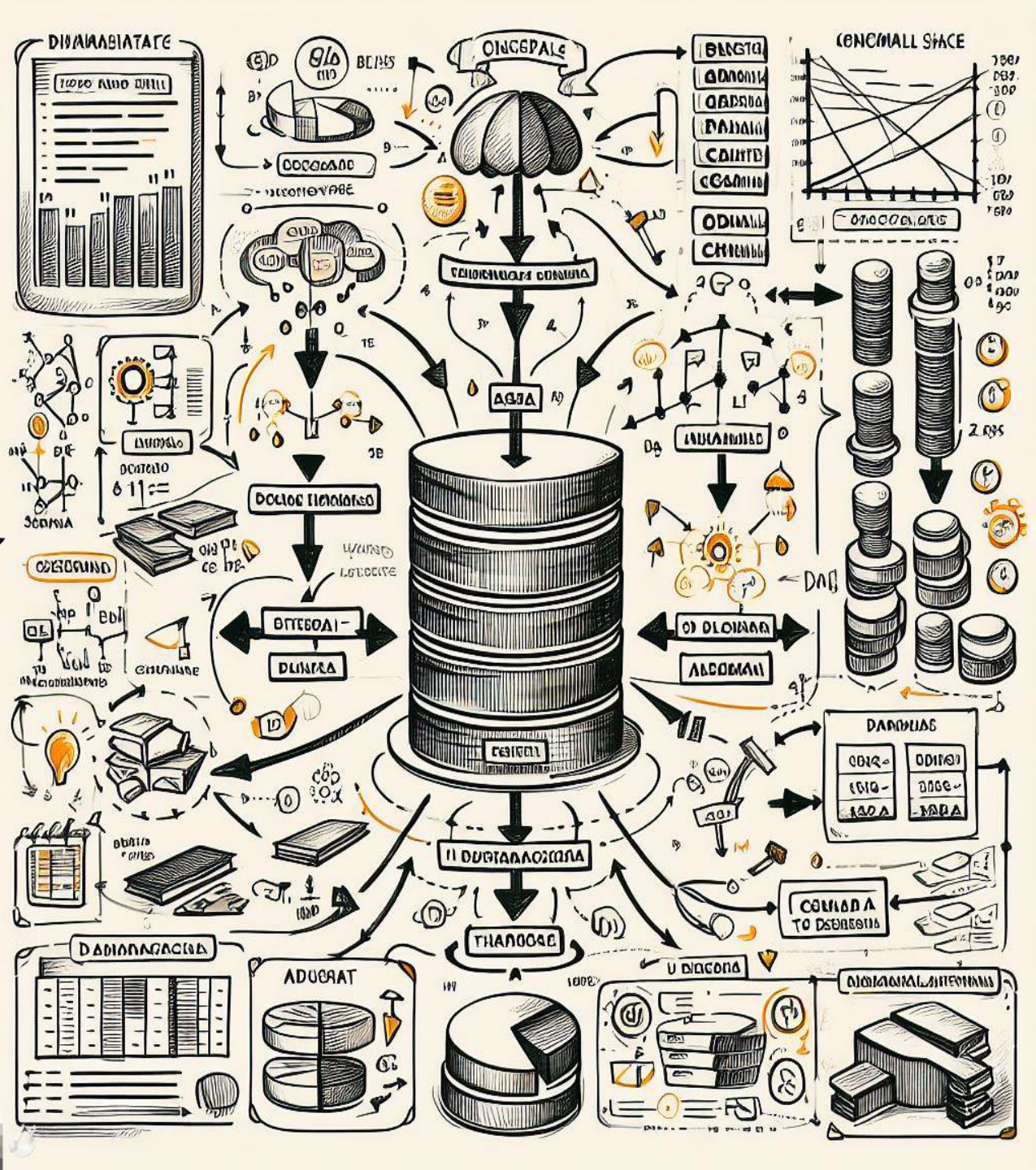
03. ER and UML


State
Country
City
Pin
Address
Shipment ID
Shipment Date
address
City
State
Country
Zip Code
Shipment
Phone Number
E Mail
Name
Customer
Customer ID
Shipped to
Have
Order ID
Order Date
Order Price
Order
Paid
Belongs to
Belongs to
Authorized
Cart ID
Cart
Quantity
Payment Mathod
Payment
Amount
Payment Date
Payment ID
Wish list ID
Wishlist
in
In
in
Name
Category
of
Product
Category ID
Product ID
Price
Stock
1...1
0...N
1...M
0...1
1..1


Shipment
-shipment_id :integer
-shipment_date :mm/dd/yyyy
-city :string
-state :string
-country :string
-zip_code :string
+getshipment_id
+getshipment_date
+getcity
+setcity(new_city)
+getstate
+setstate(new_state)
+getcountry
+setcountry(new_country)
+getzipcode
+setzipcode(new_zipcode)
belongs to
0..N
1...1
Customer
-customer_id :integer
-first_name : string
-last_name : string
-email : string
-password : string
-address : string
-phone_number : string
+getcustomer_id
+getfirst_name
+getlast_name
+getemail
+setemail(new_email)
+getpassword
+getadress
+setadress(newaddress)
+getphone_number
+setphone_number(newphone_number)
Order
-order_id : integer
-order_date :mm/dd/yyyy
-total_price : decimal
+getorder_id
+getorder_date
+gettotal_price
+settotal_price(newtotal_price)
Cart
-cart_id : integer
-quantity : integer
+getcart_id
+getquantity
Wishlist
-wishlist_id : integer
+getwishlist_id
Payment
-payment_id :integer
-payment_date :mm/dd/yyyy
-payment_method :string
-amount :decimal
+getpayment_id
+getpayment_date
+getpayment_method
+get_amount
Order Item
-order_item_id :integer
-quantity :integer
-price :decimal
+getorder_item_id
+getquantity
+setquantity(new_quantity)
+getprice
+setprice(new_price)
Product
-product_id : integer
-description : string
-price :decimal
-stock : integer
+getproduct_id
+getdescription
+setdescription(new_description)
+getprice
+setprice(new_price)
+getstock
Category
-category_id :integer
-name : string
+getcategory_ID
+getname
+setname(new_name)
1...1
0..N
0...1
IN
0...N

04. Converting our Conceptual DB to Relational Data Base and to Its Normalized form





Data Integrity: Relational databases ensure data integrity by enforcing constraints like primary keys, foreign keys, and relationships. This prevents anomalies and inconsistencies in data.

Reduction of Anomalies: Normalization minimizes data anomalies like insertion, update, and deletion anomalies, ensuring the database remains in a consistent state.

Easier Maintenance: With a well-normalized relational database, maintenance becomes more straightforward. It's easier to troubleshoot issues and maintain the overall health of the database.

Scalability: Normalized databases are generally more scalable. As data grows, the structure can accommodate changes and expansions without significant alterations, maintaining efficiency.

Efficient Data Storage : Normalization reduces data redundancy by organizing data into smaller, more manageable tables. This results in efficient use of storage space and reduces the chances of inconsistencies.

Industry Standard: Relational databases follow industry-standard practices, making it easier for developers and analysts to work with and understand the database structure.

Improved Query Performance: Relational databases with normalized tables often perform better in queries. Normalization allows for smaller tables that can be joined efficiently, leading to faster query execution.

Data Consistency: Relational databases promote consistent data across tables. Updates or changes made in one place reflect uniformly across related tables, ensuring consistency.

05(A) SQL

Total sales by product category:

```
SELECT p.category, SUM(o.quantity * p.costprice) AS
total_sales
FROM product p
JOIN orders o ON p.productID = o.productID
GROUP BY    category limit 10;
```

Average order value by customer

```
SELECT c.cutomerID,
AVG(o.quantity * p.costprice) AS average_order_value
FROM customer as c
JOIN orders as o ON c.cutomerID = o.cutomerID
JOIN product p ON o.productID = p.productID
GROUP BY c.cutomerID limit 10;
```

Top most abandoned item in carts

```
SELECT c.productID, p.product,
COUNT(*) AS abandoned_count FROM
cart c LEFT JOIN orders o ON
c.productID = o.productID JOIN
product p ON c.productID =
p.productID WHERE o.productID IS
NULL GROUP BY c.productID, p.product
ORDER BY abandoned_count DESC LIMIT
10;
```

Profit from each product

```
select p.productID,p.product,
pm.paymentValue-p.costprice as
Profit from payment as pm inner
join orders as o on o.paymentID =
pm.paymentID inner join product as
p on o.productID = p.productID
limit 10;
```

Segmenting Customers on order category

```
SELECT p.category, c.`cutomerID`,
c.`Name`, COUNT(o.`orderID`) as
total_orders
FROM product as p
JOIN orders as o ON p.`productID` =
o.`productID`
JOIN customer as c ON o.`cutomerID` =
c.`cutomerID`
GROUP BY p.category, c.`cutomerID`,
c.`Name`
ORDER BY p.category, total_orders DESC
```

05(B) NoSQL

The average selling price of each product

```
use("ecomsite")
db.customerdata.aggregate([
  { $group: { _id: "$productName",
avgSellingPrice: { $avg:
"$sellingPrice" } } }
 ])
```

The total quantity of each product sold

```
use('ecomsite’)

db.customerdata.aggregate([
  { $group: { _id: "$productName",
totalQuantity: { $sum: "$Quantity" } } }
])
```

The top-selling product by quantity

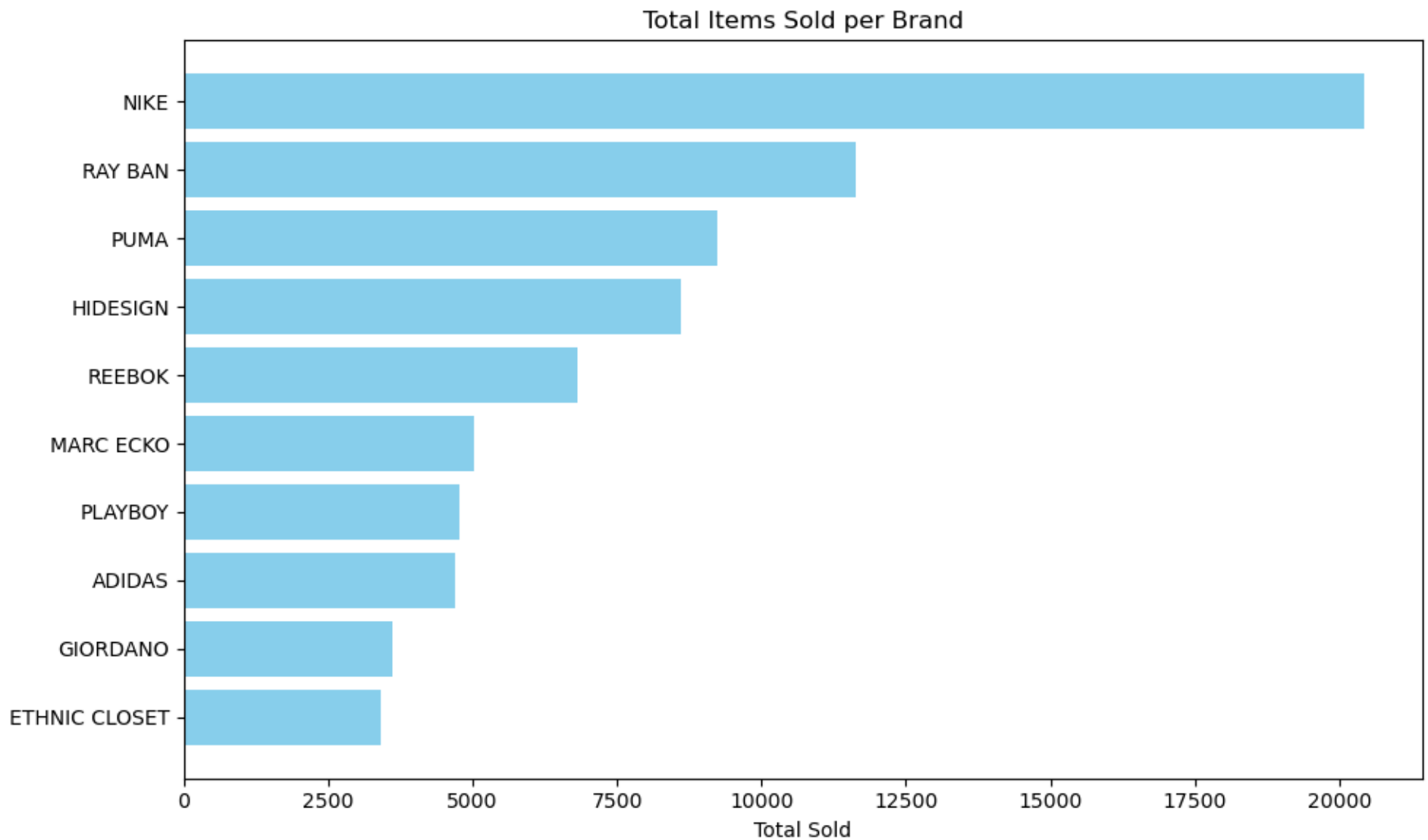
```
use('ecomsite')

db.customerdata.aggregate([{
$group: { _id: "$productName",
totalRevenue: { $sum: {
$multiply: [ "$Quantity",
"$sellingPrice" ] } } } }])
```

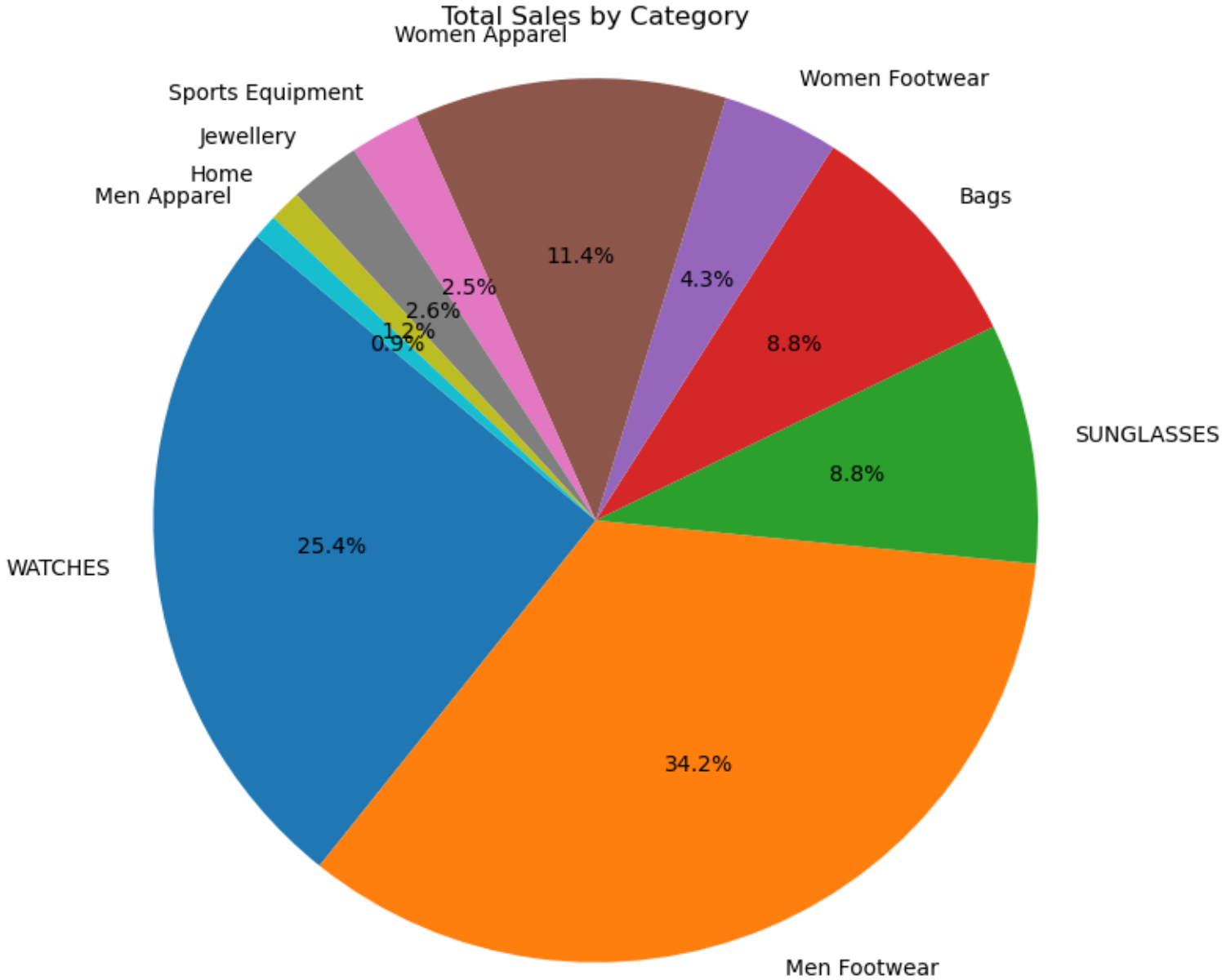
The profit from each product

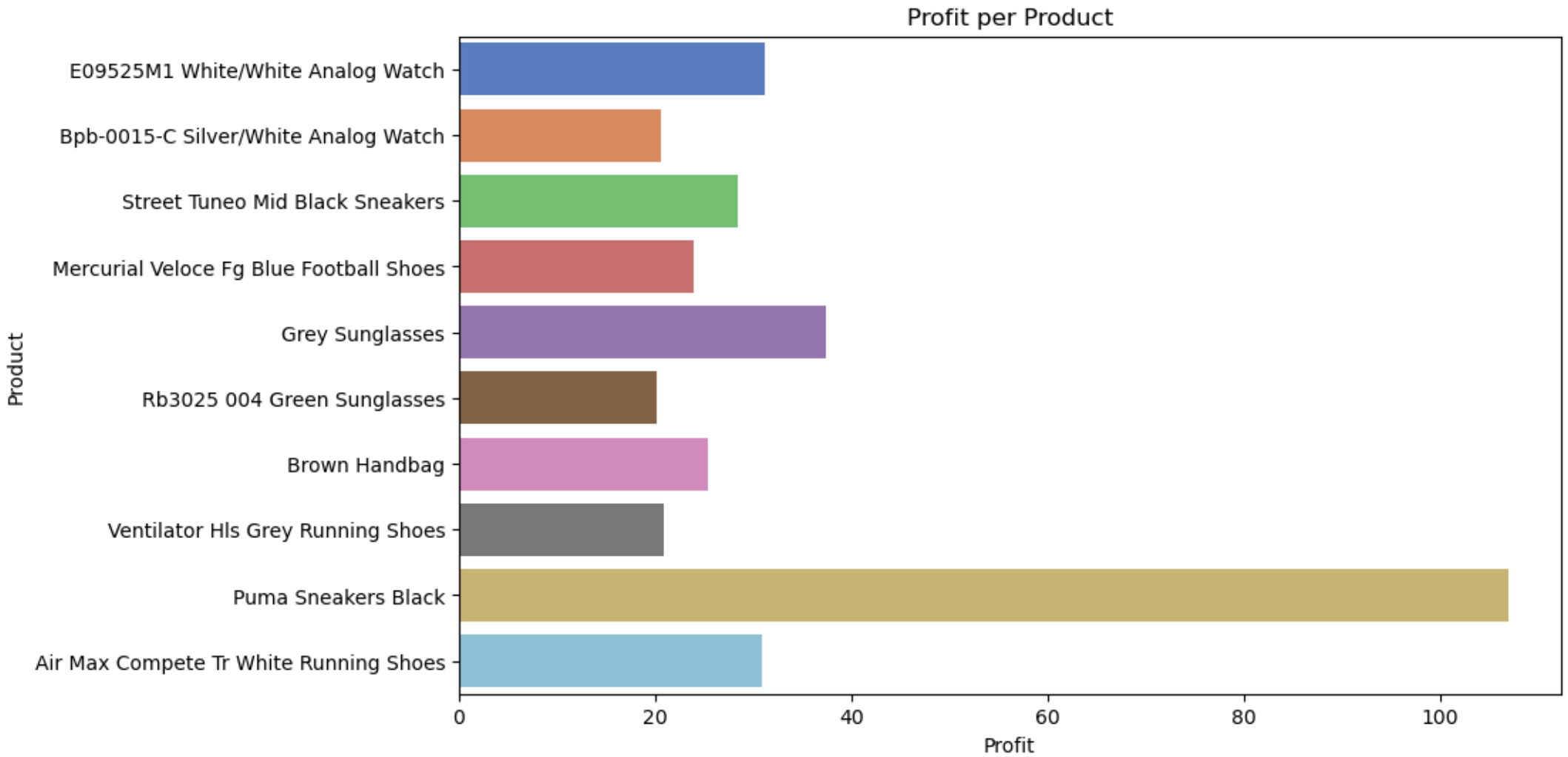
```
use('ecomsite')

db.customerdata.aggregate([{
$group: { _id: "$productName",
profit: { $sum: { $subtract: [
"$sellingPrice", "$Cost_Price" ]
} } } }])
```

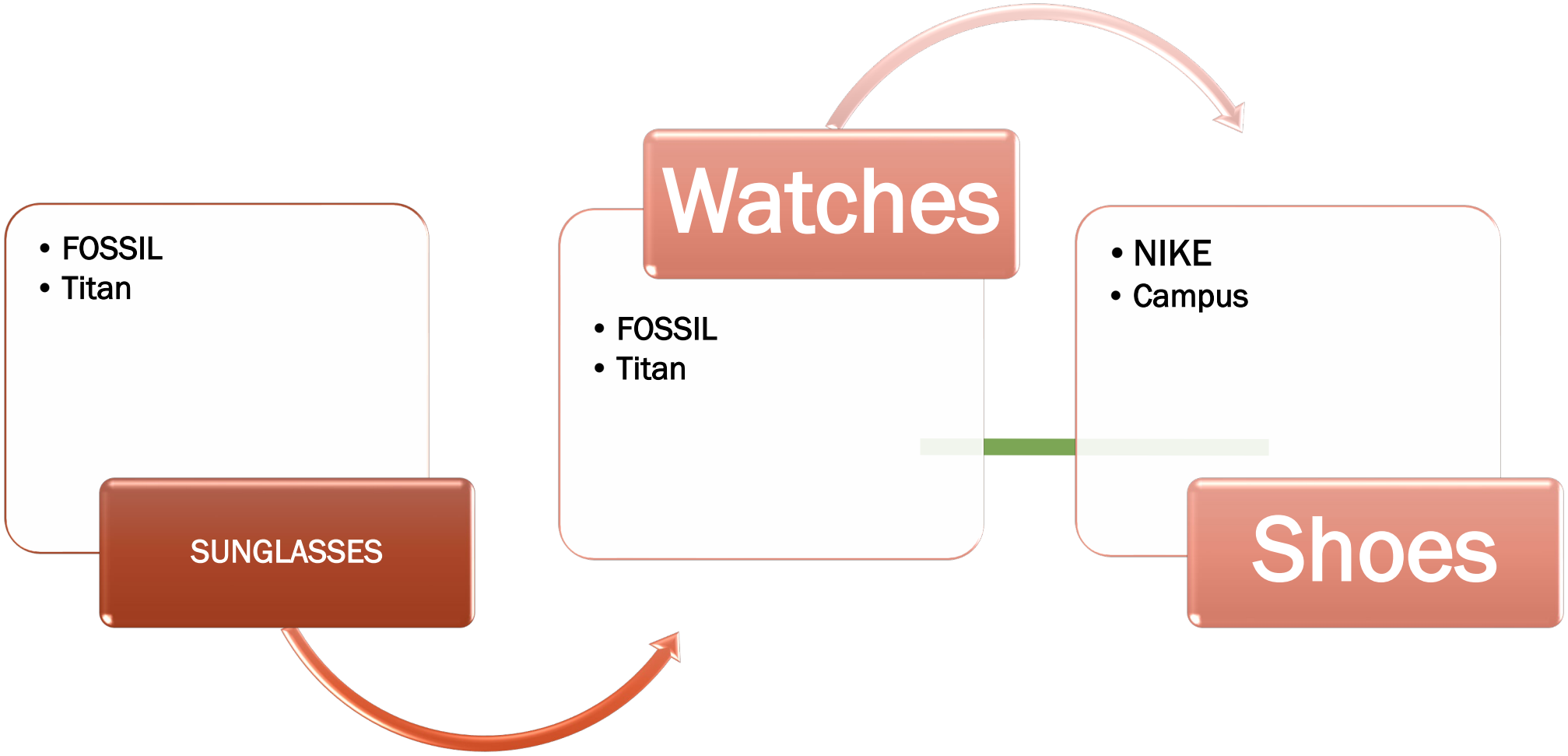
Find the top-selling brand

```
use('ecomsite’)

db.customerdata.aggregate([
  { "$group": { "_id":
"$Brand", "totalSold": {
"$sum": "$Quantity" } } },
  { "$sort": { "totalSold": -1
} },
  { "$limit": 10 }
])
```


Total Items Sold per Brand
NIKE
RAY BAN
PUMA
HIDESIGN
REEBOK
MARC ECKO
PLAYBOY
ADIDAS
GIORDANO
ETHNIC CLOSET
0
2500
5000
7500
10000
12500
15000
17500
20000
Total Sold


Total Sales by Category
Women Apparel
Women Footwear
Sports Equipment
Jewellery
Home
Men Apparel
Bags
11.4%
4.3%
2.5%
2.6%
1.2%
0.9%
8.8%
SUNGLASSES
8.8%
25.4%
WATCHES
34.2%
Men Footwear


Profit per Product
E09525M1 White/White Analog Watch
Bpb-0015-C Silver/White Analog Watch
Street Tuneo Mid Black Sneakers
Mercurial Veloce Fg Blue Football Shoes
Grey Sunglasses
Rb3025 004 Green Sunglasses
Brown Handbag
Ventilator Hls Grey Running Shoes
Puma Sneakers Black
Air Max Compete Tr White Running Shoes
Product
0
20
40
60
80
100
Profit

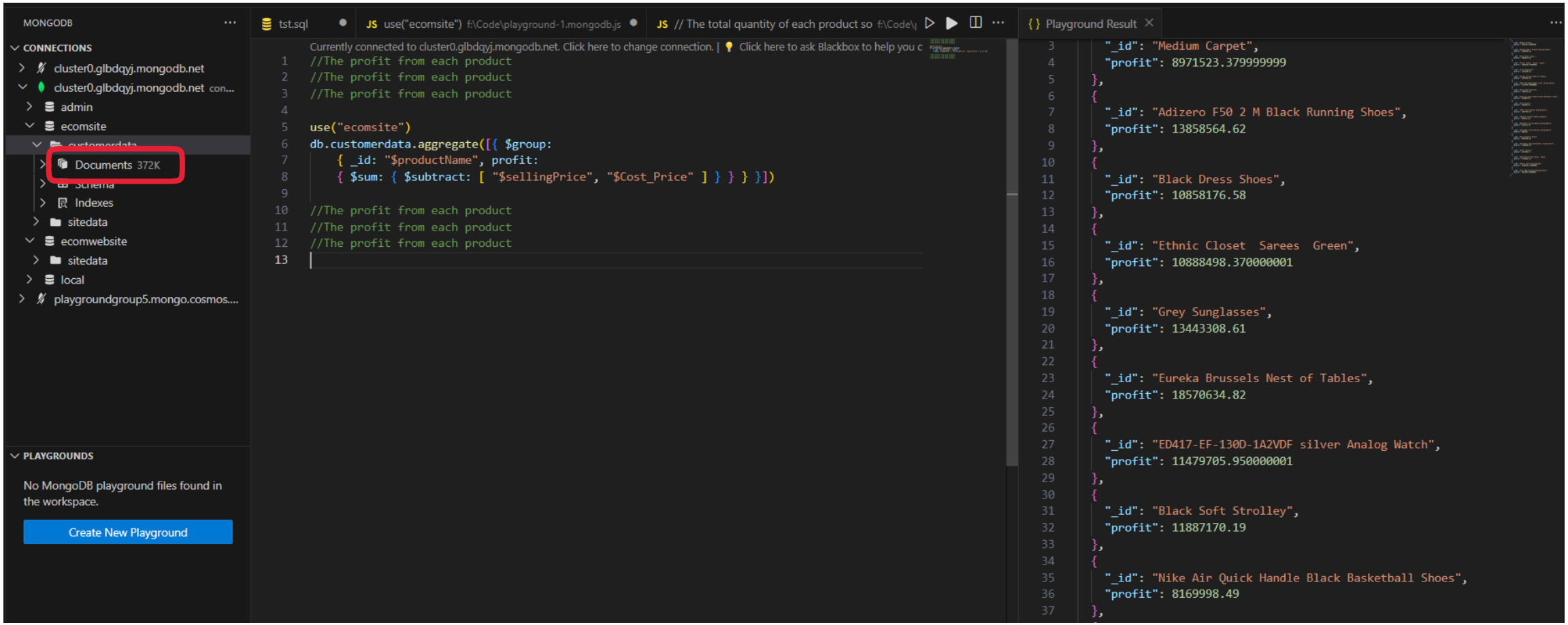

• FOSSIL
• Titan
SUNGLASSES
Watches
• FOSSIL
• Titan
• NIKE
• Campus
Shoes

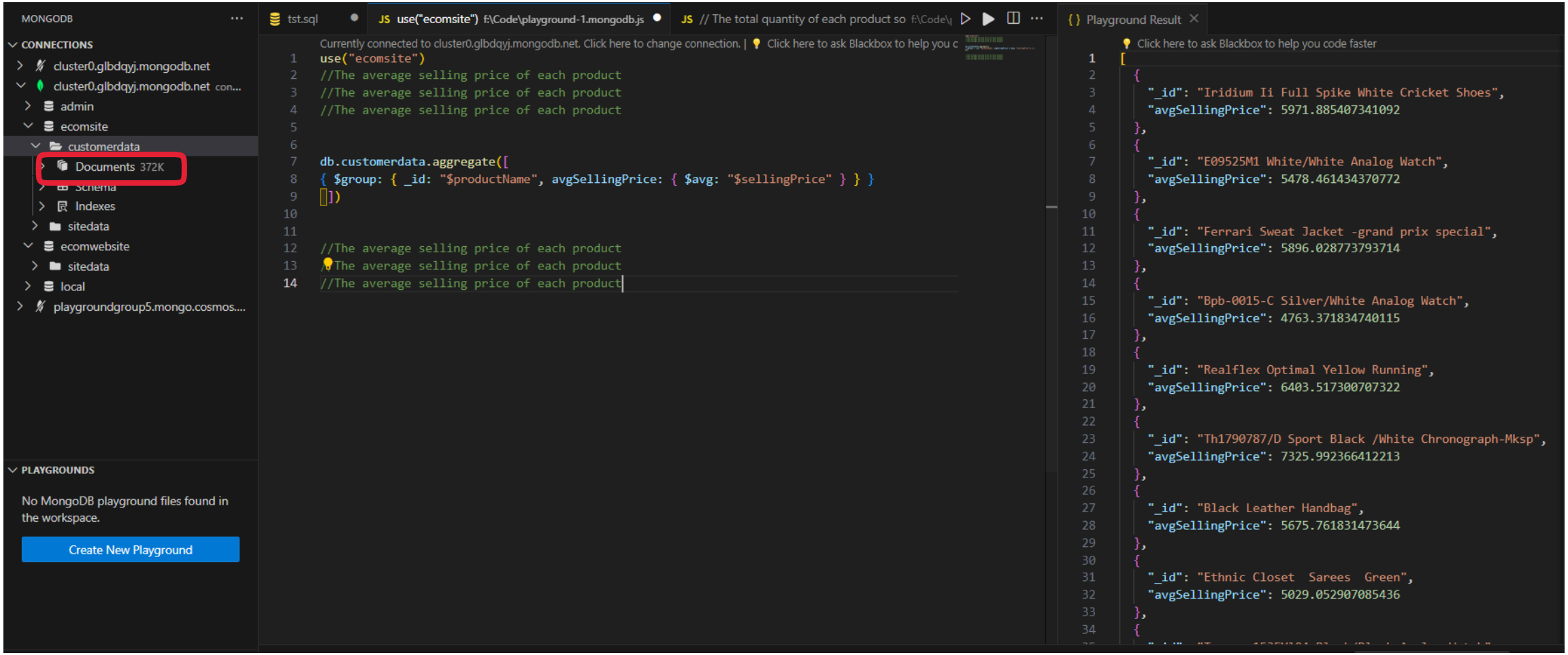
Summary

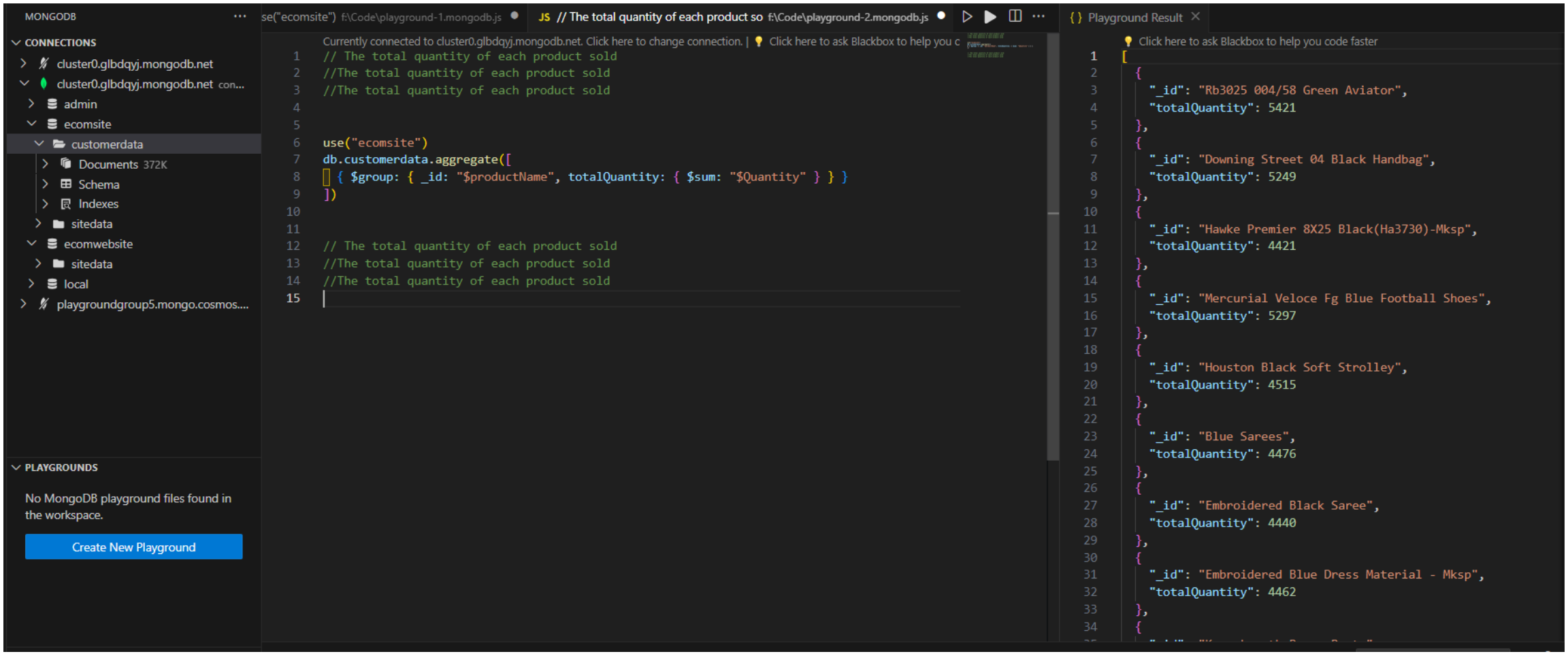
The conceptual database model for an e-commerce platform comprises entities like Customer, Product, Order, Payment, Shipment, Cart, Wishlist, and Category, each with specific attributes and relationships. Customers place orders connected through a one-to-many relationship, while order items link orders and products, facilitating tracking of items in each order. Payments and shipments maintain one-to-many ties with orders, managing payment details and shipping specifics. Categories categorize products, aiding in organization. Reference data includes static customer and product info, while transactional data captures dynamic details like orders, payments, and shipments, crucial for the system's real-time operation and effective e-commerce functionality.

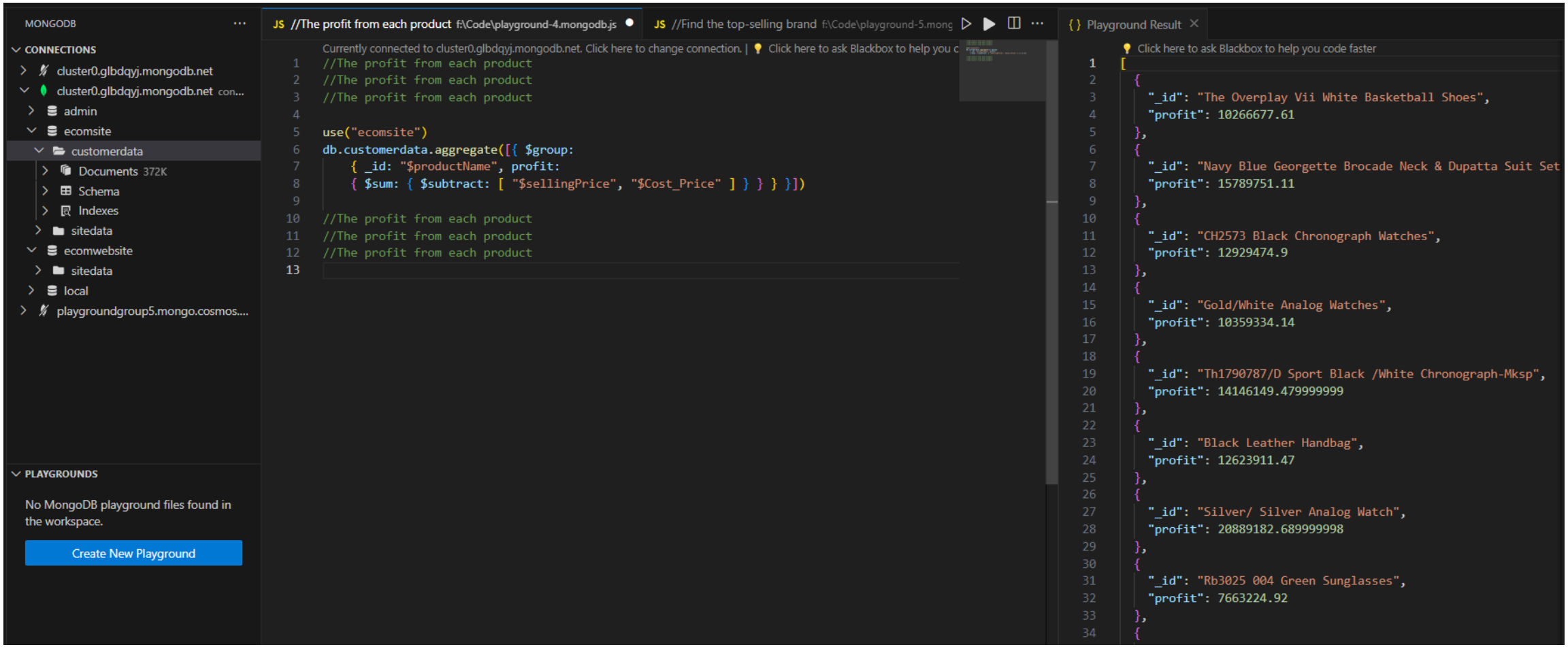
Thank you

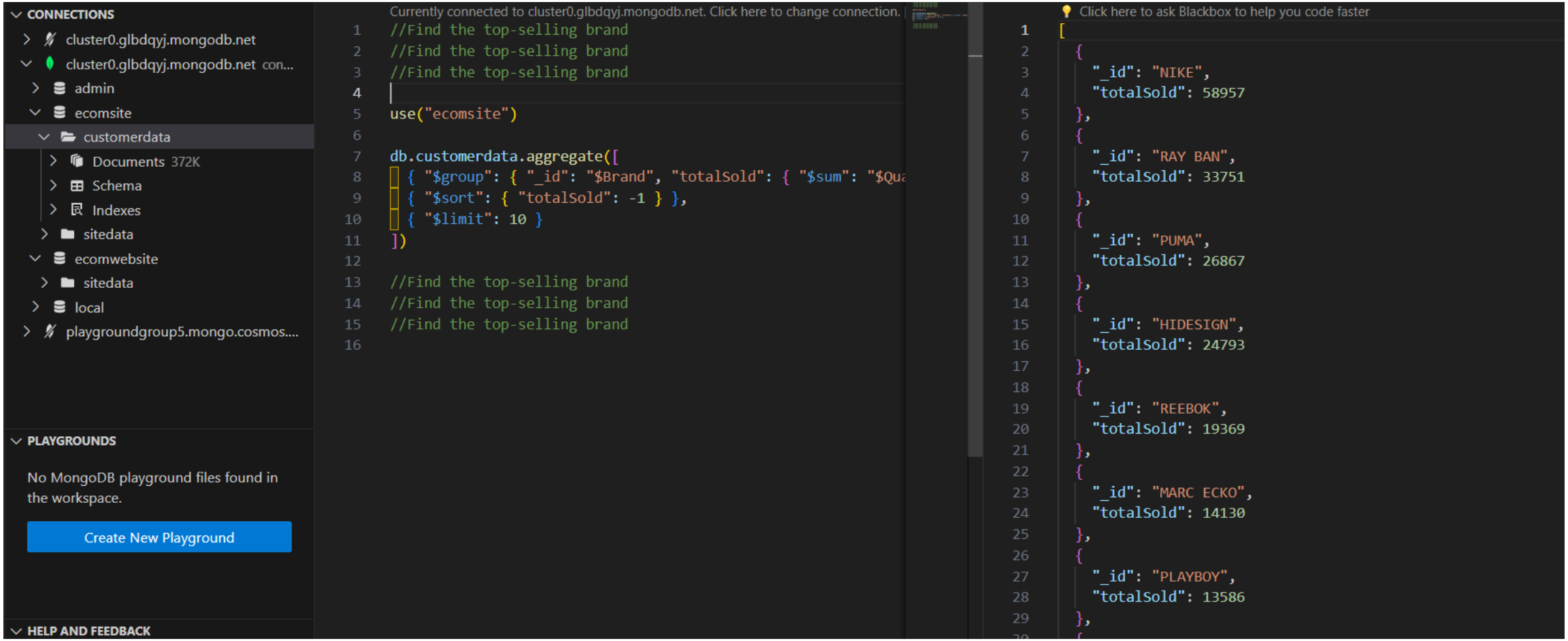
- Rakshith Dharmappa
- Rohith Reddy


MONGODB
CONNECTIONS
cluster0.glbdqyj.mongodb.net
cluster0.glbdqyj.mongodb.net con...
admin
ecomsite
Documents 372K
Indexes
sitedata
ecomwebsite
sitedata
local
playgroundgroup5.mongo.cosmos....
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
tst.sql
use("ecomsite") f:\Code\playground-1.mongodb.js
// The total quantity of each product so f:\Code\
Currently connected to cluster0.glbdqyj.mongodb.net. Click here to change connection. | Click here to ask Blackbox to help you c
//The profit from each product
//The profit from each product
//The profit from each product
use("ecomsite")
db.customerdata.aggregate([{ $group:
{ _id: "$productName", profit:
{ $sum: { $subtract: ["$sellingPrice", "$Cost_Price"] } } } }])
//The profit from each product
//The profit from each product
//The profit from each product
Playground Result
"_id": "Medium Carpet",
"profit": 8971523.379999999
"_id": "Adizero F50 2 M Black Running Shoes",
"profit": 13858564.62
"_id": "Black Dress Shoes",
"profit": 10858176.58
"_id": "Ethnic Closet Sarees Green",
"profit": 10888498.370000001
"_id": "Grey Sunglasses",
"profit": 13443308.61
"_id": "Eureka Brussels Nest of Tables",
"profit": 18570634.82
"_id": "ED417-EF-130D-1A2VDF silver Analog Watch",
"profit": 11479705.950000001
"_id": "Black Soft Strolley",
"profit": 11887170.19
"_id": "Nike Air Quick Handle Black Basketball Shoes",
"profit": 8169998.49


MONGODB
CONNECTIONS
cluster0.glbdqyj.mongodb.net
cluster0.glbdqyj.mongodb.net con...
admin
ecomsite
customerdata
Documents 372K
Schema
Indexes
sitedata
ecomwebsite
sitedata
local
playgroundgroup5.mongo.cosmos....
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
tst.sql
use("ecomsite") f:\Code\playground-1.mongodb.js
// The total quantity of each product so f:\Code\
Currently connected to cluster0.glbdqyj.mongodb.net. Click here to change connection. | Click here to ask Blackbox to help you c
use("ecomsite")
//The average selling price of each product
//The average selling price of each product
//The average selling price of each product
db.customerdata.aggregate([
{ $group: { _id: "$productName", avgSellingPrice: { $avg: "$sellingPrice" } } }
])
//The average selling price of each product
//The average selling price of each product
//The average selling price of each product
Playground Result
Click here to ask Blackbox to help you code faster
[
{
"_id": "Iridium Ii Full Spike White Cricket Shoes",
"avgSellingPrice": 5971.885407341092
},
{
"_id": "E09525M1 White/White Analog Watch",
"avgSellingPrice": 5478.461434370772
},
{
"_id": "Ferrari Sweat Jacket -grand prix special",
"avgSellingPrice": 5896.028773793714
},
{
"_id": "Bpb-0015-C Silver/White Analog Watch",
"avgSellingPrice": 4763.371834740115
},
{
"_id": "Realflex Optimal Yellow Running",
"avgSellingPrice": 6403.517300707322
},
{
"_id": "Th1790787/D Sport Black /White Chronograph-Mksp",
"avgSellingPrice": 7325.992366412213
},
{
"_id": "Black Leather Handbag",
"avgSellingPrice": 5675.761831473644
},
{
"_id": "Ethnic Closet Sarees Green",
"avgSellingPrice": 5029.052907085436
},
{


MONGODB
CONNECTIONS
cluster0.glbdqyj.mongodb.net
cluster0.glbdqyj.mongodb.net con...
admin
ecomsite
customerdata
Documents 372K
Schema
Indexes
sitedata
ecomwebsite
sitedata
local
playgroundgroup5.mongo.cosmos....
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
se("ecomsite") f:\Code\playground-1.mongodb.js
// The total quantity of each product so f:\Code\playground-2.mongodb.js
Currently connected to cluster0.glbdqyj.mongodb.net. Click here to change connection. | Click here to ask Blackbox to help you c
// The total quantity of each product sold
//The total quantity of each product sold
//The total quantity of each product sold
use("ecomsite")
db.customerdata.aggregate([
{ $group: { _id: "$productName", totalQuantity: { $sum: "$Quantity" } } }
])
// The total quantity of each product sold
//The total quantity of each product sold
//The total quantity of each product sold
Playground Result
Click here to ask Blackbox to help you code faster
[
{
"_id": "Rb3025 004/58 Green Aviator",
"totalQuantity": 5421
},
{
"_id": "Downing Street 04 Black Handbag",
"totalQuantity": 5249
},
{
"_id": "Hawke Premier 8X25 Black(Ha3730)-Mksp",
"totalQuantity": 4421
},
{
"_id": "Mercurial Veloce Fg Blue Football Shoes",
"totalQuantity": 5297
},
{
"_id": "Houston Black Soft Strolley",
"totalQuantity": 4515
},
{
"_id": "Blue Sarees",
"totalQuantity": 4476
},
{
"_id": "Embroidered Black Saree",
"totalQuantity": 4440
},
{
"_id": "Embroidered Blue Dress Material - Mksp",
"totalQuantity": 4462
},
{


MONGODB
CONNECTIONS
cluster0.glbdqyj.mongodb.net
cluster0.glbdqyj.mongodb.net con...
admin
ecomsite
customerdata
Documents 372K
Schema
Indexes
sitedata
ecomwebsite
sitedata
local
playgroundgroup5.mongo.cosmos....
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
//The profit from each product f:\Code\playground-4.mongodb.js
//Find the top-selling brand f:\Code\playground-5.mong
Currently connected to cluster0.glbdqyj.mongodb.net. Click here to change connection. | Click here to ask Blackbox to help you c
//The profit from each product
//The profit from each product
//The profit from each product
use("ecomsite")
db.customerdata.aggregate([{ $group:
{ _id: "$productName", profit:
{ $sum: { $subtract: ["$sellingPrice", "$Cost_Price"] } } } }])
//The profit from each product
//The profit from each product
//The profit from each product
Playground Result
Click here to ask Blackbox to help you code faster
[
{
"_id": "The Overplay Vii White Basketball Shoes",
"profit": 10266677.61
},
{
"_id": "Navy Blue Georgette Brocade Neck & Dupatta Suit Set
"profit": 15789751.11
},
{
"_id": "CH2573 Black Chronograph Watches",
"profit": 12929474.9
},
{
"_id": "Gold/White Analog Watches",
"profit": 10359334.14
},
{
"_id": "Th1790787/D Sport Black /White Chronograph-Mksp",
"profit": 14146149.479999999
},
{
"_id": "Black Leather Handbag",
"profit": 12623911.47
},
{
"_id": "Silver/ Silver Analog Watch",
"profit": 20889182.689999998
},
{
"_id": "Rb3025 004 Green Sunglasses",
"profit": 7663224.92
},
{


CONNECTIONS
cluster0.glbdqyj.mongodb.net
cluster0.glbdqyj.mongodb.net con...
admin
ecomsite
customerdata
Documents 372K
Schema
Indexes
sitedata
ecomwebsite
sitedata
local
playgroundgroup5.mongo.cosmos....
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
HELP AND FEEDBACK
Currently connected to cluster0.glbdqyj.mongodb.net. Click here to change connection.
//Find the top-selling brand
//Find the top-selling brand
//Find the top-selling brand
use("ecomsite")
db.customerdata.aggregate([
{ "$group": { "_id": "$Brand", "totalSold": { "$sum": "$Qua
{ "$sort": { "totalSold": -1 } },
{ "$limit": 10 }
])
//Find the top-selling brand
//Find the top-selling brand
//Find the top-selling brand
Click here to ask Blackbox to help you code faster
[
{
"_id": "NIKE",
"totalSold": 58957
},
{
"_id": "RAY BAN",
"totalSold": 33751
},
{
"_id": "PUMA",
"totalSold": 26867
},
{
"_id": "HIDESIGN",
"totalSold": 24793
},
{
"_id": "REEBOK",
"totalSold": 19369
},
{
"_id": "MARC ECKO",
"totalSold": 14130
},
{
"_id": "PLAYBOY",
"totalSold": 13586
},

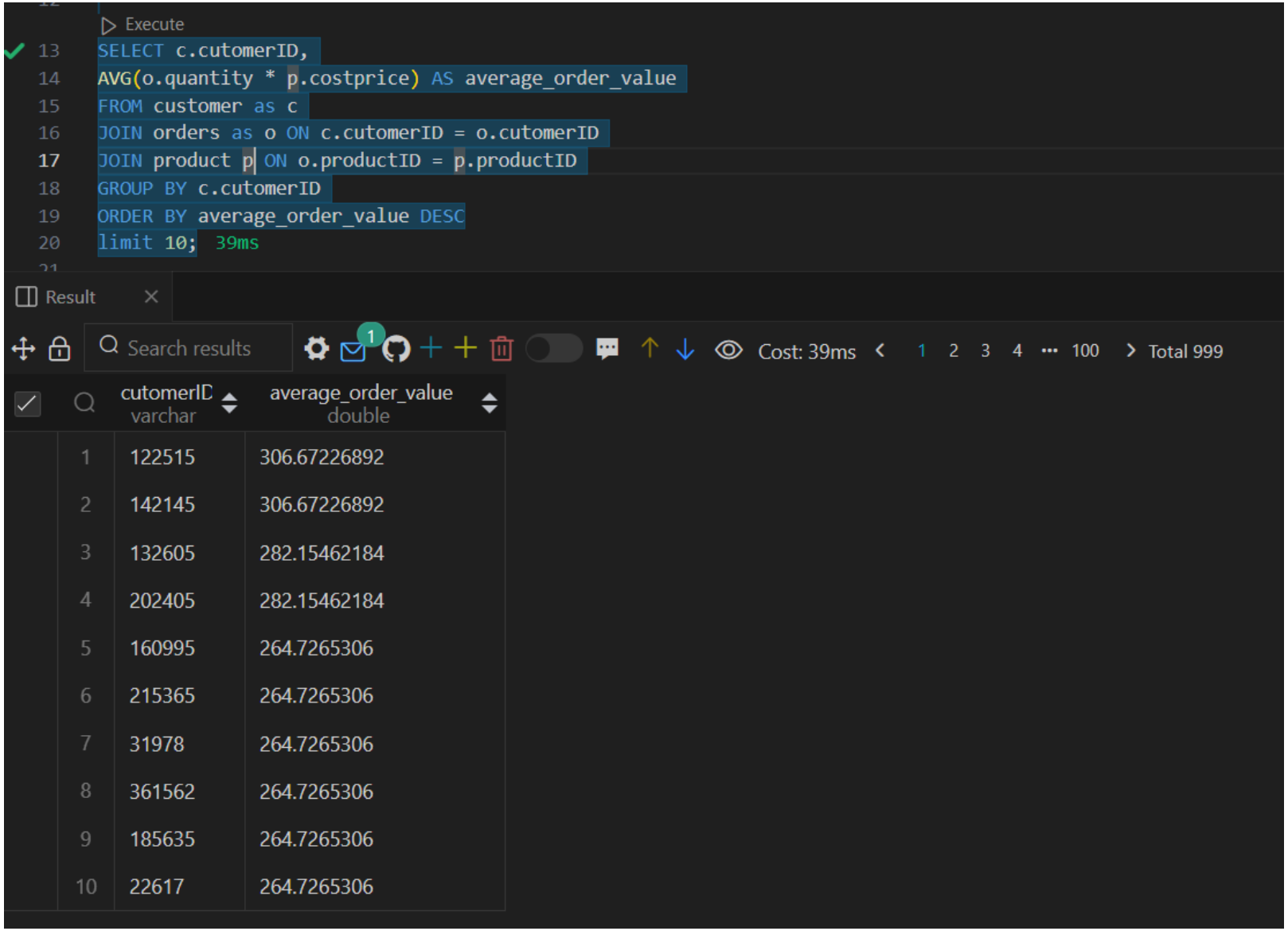
```sql
--Total sales by product category:

SELECT p.category, SUM(o.quantity * p.costprice) AS total_sales
FROM product p
JOIN orders o ON p.productID = o.productID
GROUP BY p.category limit 10;  155ms
```

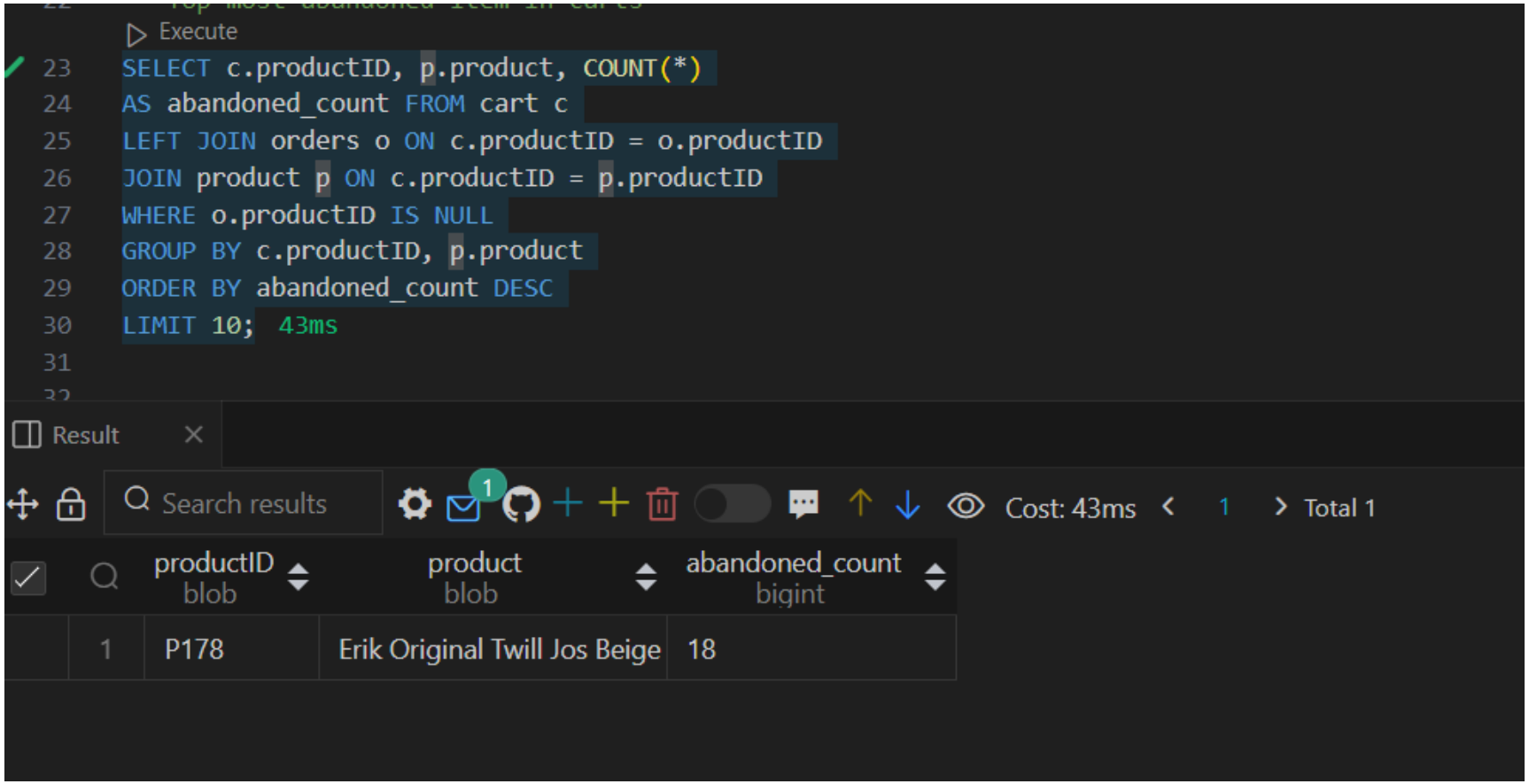
Result

Search results

Cost: 155ms 1 2 Total 11

	category blob	total_sales double
1	WATCHES	24552.37142888
2	Men Footwear	33047.68607389001
3	SUNGLASSES	8499.600120170002
4	Bags	8482.391596820004
5	Women Footwear	4116.311884520001
6	Women Apparel	11046.8009603
7	Sports Equipment	2465.21932767
8	Jewellery	2530.04621859
9	Home	1129.87238915
10	Men Apparel	871.1668668


Execute
13 SELECT c.cutomerID,
14 AVG(o.quantity * p.costprice) AS average_order_value
15 FROM customer as c
16 JOIN orders as o ON c.cutomerID = o.cutomerID
17 JOIN product p ON o.productID = p.productID
18 GROUP BY c.cutomerID
19 ORDER BY average_order_value DESC
20 limit 10; 39ms
Result
Search results
Cost: 39ms
1 2 3 4 100
Total 999
cutomerID varchar
average_order_value double
1 122515 306.67226892
2 142145 306.67226892
3 132605 282.15462184
4 202405 282.15462184
5 160995 264.7265306
6 215365 264.7265306
7 31978 264.7265306
8 361562 264.7265306
9 185635 264.7265306
10 22617 264.7265306


Execute
23 SELECT c.productID, p.product, COUNT(*)
24 AS abandoned_count FROM cart c
25 LEFT JOIN orders o ON c.productID = o.productID
26 JOIN product p ON c.productID = p.productID
27 WHERE o.productID IS NULL
28 GROUP BY c.productID, p.product
29 ORDER BY abandoned_count DESC
30 LIMIT 10; 43ms
31
Result
Search results
Cost: 43ms
1
Total 1
productID blob
product blob
abandoned_count bigint
1 P178 Erik Original Twill Jos Beige 18

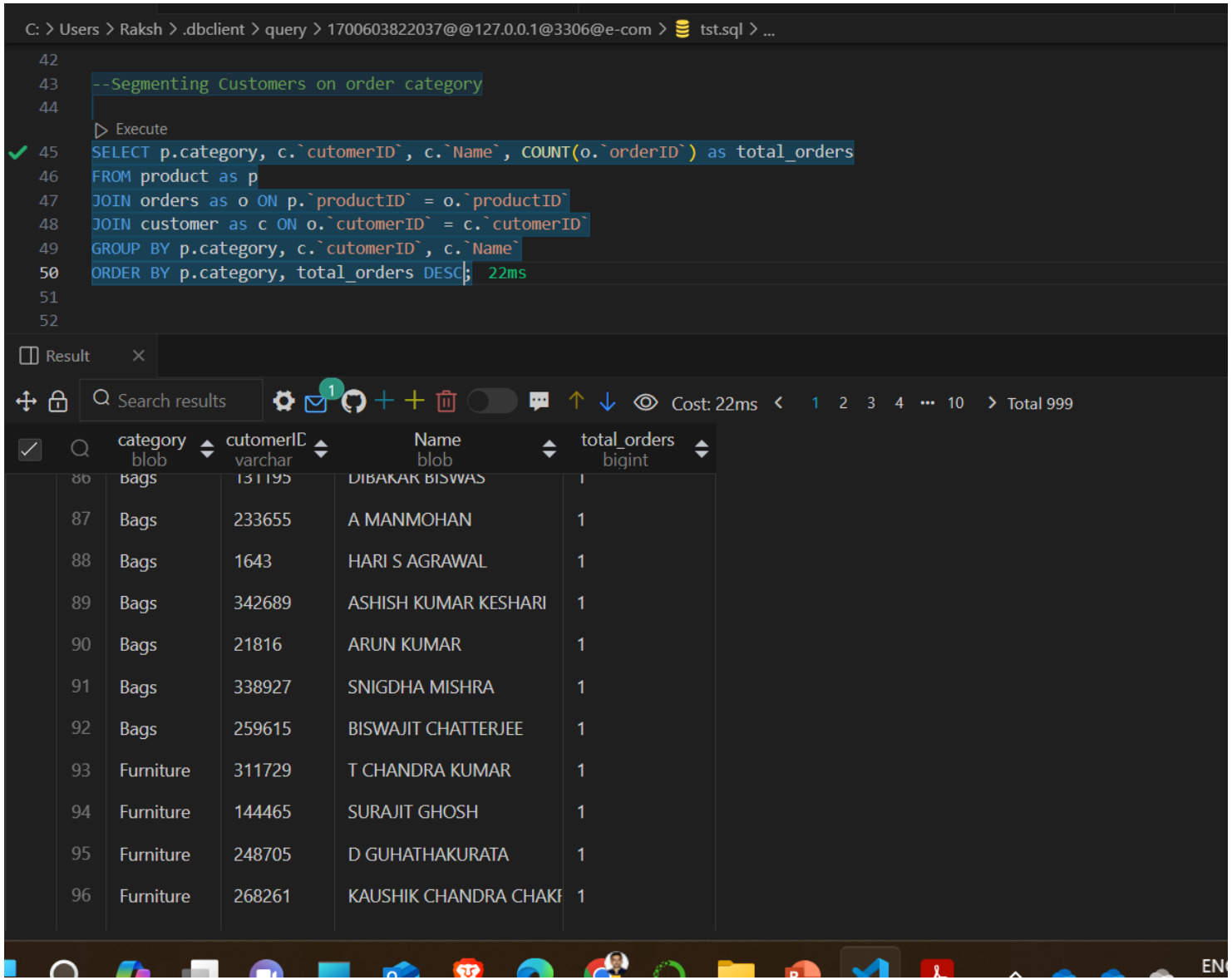
```sql
-- Profit from each product

▷ Execute
select p.productID,p.product, pm.paymentValue-p.costprice as Profit
from payment as pm
inner join orders as o on o.paymentID = pm.paymentID
inner join product as p on o.productID = p.productID
limit 10;  26ms
```

Result

Search results

Cost: 26ms 1 2 3 4 ··· 100 Total 999

	productID text	product text	Profit double
1	P101	E09525M1 White/White An	31.228465039999996
2	P102	Bpb-0015-C Silver/White A	20.5613697
3	P103	Street Tuneo Mid Black Sne	28.352378319999993
4	P104	Mercurial Veloce Fg Blue Fc	23.959831360000003
5	P105	Grey Sunglasses	37.39446988
6	P106	Rb3025 004 Green Sunglas	20.141157779999993
7	P107	Brown Handbag	25.32475145
8	P108	Ventilator Hls Grey Running	20.816901610000002
9	P109	Puma Sneakers Black	106.98690613
10	P110	Air Max Compete Tr White	30.861821850000005


C: > Users > Raksh > .dbclient > query > 1700603822037@@127.0.0.1@3306@e-com > tst.sql > ...
42
43 --Segmenting Customers on order category
44
Execute
45 SELECT p.category, c.`cutomerID`, c.`Name`, COUNT(o.`orderID`) as total_orders
46 FROM product as p
47 JOIN orders as o ON p.`productID` = o.`productID`
48 JOIN customer as c ON o.`cutomerID` = c.`cutomerID`
49 GROUP BY p.category, c.`cutomerID`, c.`Name`
50 ORDER BY p.category, total_orders DESC; 22ms
51
52
Result
Search results
Cost: 22ms
1 2 3 4 ... 10
Total 999
category blob
cutomerID varchar
Name blob
total_orders bigint
86 Bags 131195 DIBAKAR BISWAS 1
87 Bags 233655 A MANMOHAN 1
88 Bags 1643 HARI S AGRAWAL 1
89 Bags 342689 ASHISH KUMAR KESHARI 1
90 Bags 21816 ARUN KUMAR 1
91 Bags 338927 SNIGDHA MISHRA 1
92 Bags 259615 BISWAJIT CHATTERJEE 1
93 Furniture 311729 T CHANDRA KUMAR 1
94 Furniture 144465 SURAJIT GHOSH 1
95 Furniture 248705 D GUHATHAKURATA 1
96 Furniture 268261 KAUSHIK CHANDRA CHAKI 1