USE CASE STUDY REPORT

Group No.: Group 15

Student Names: Neha Patil & Shivie Saksenaa

E-Commerce Database Management System

Executive Summary:

Creating a database model an E-Commerce company called "Amazon" to store details of consumers and sellers, the items purchased, the payment method generated ands the reviews made about the items. The system follows a. Business to consumer (B2C) communication. B2C companies operate on the internet and sell products to customers online.

This case study entails the examination of an online retail Amazon dataset with the aim of delineating customer segments and their behavioral trends across various time spans. The examination will aid in forecasting customer attrition and tailoring services to individual. To leverage the data for addressing several inquiries about the customers, particularly concerning their visitation patterns, expenditure, and their preferred items. This will improve the delivery and provide more personalized experience.

The database construction primarily involved the utilization of MySQL, incorporating input from key properties and tables to ensure seamless system functionality. The process began with the modeling of Entity-Relationship (EER) and Unified Modeling Language (UML) diagrams, followed by the translation of the conceptual model into a relational model, including the definition of necessary primary, composite, and foreign keys. The complete implementation of this database took place in MySQL, resulting in a prototype featuring 15 tables and their respective relationships.

To enhance the system's versatility, additional collections were established in MongoDB Compass, utilizing a NoSQL database approach. This hybrid database solution proved to be highly successful. Integration with Python further amplified the analytical capabilities of the database. The data interaction was facilitated through Python, particularly in Jupyter Notebook, where we converted SQL tables into data frames and conducted visualizations for comprehensive analysis.

An Amazon User Account can register– A user can be a consumer or a seller

• A user can also subscribe - A user can be on a subscription plan **Subscription:** start_date, end_date, subscription_id **User**: Password, User_id, User_type,Name

• A seller can add items- this contains the details of a product.

• A seller can sell multiple or zero products.

• A seller may be a brand owner or a reseller.

Seller: Seller Type, Company_Name, Tax_ID

• A consumer can place order- each order contains multiple products.

Consumer: Address, Phone number, name, is_prime, password, consumer_id

- A consumer can have multiple address and contact details- users address and phone number saved in the account
- A consumer can give more than one review or no review at all about items.

  Review: Rating, Review_id, review_date,item_id

- A consumer can add products to a Wishlist- buyer can store the products which they like but not yet ready to buy

  Wishlist: date_added

- A consumer can add products to a shopping cart- consumers add to the shopping cart the products they want to buy.

  Shopping cart: date-added

- A consumer can have multiple order histories- consumers can add their item to the order from their order histories.

  Order History: order_id,user_id,date_of_order, order summary, payment method, grand total

- Items are can be added to the amazon order through the shopping cart, wishlist or and order history.

  Items: Item_name, Item_id, quantity, discount, description, item_price

- The warehouse holds multiple items.

  Warehouses: Name, warehouse_id, Location.

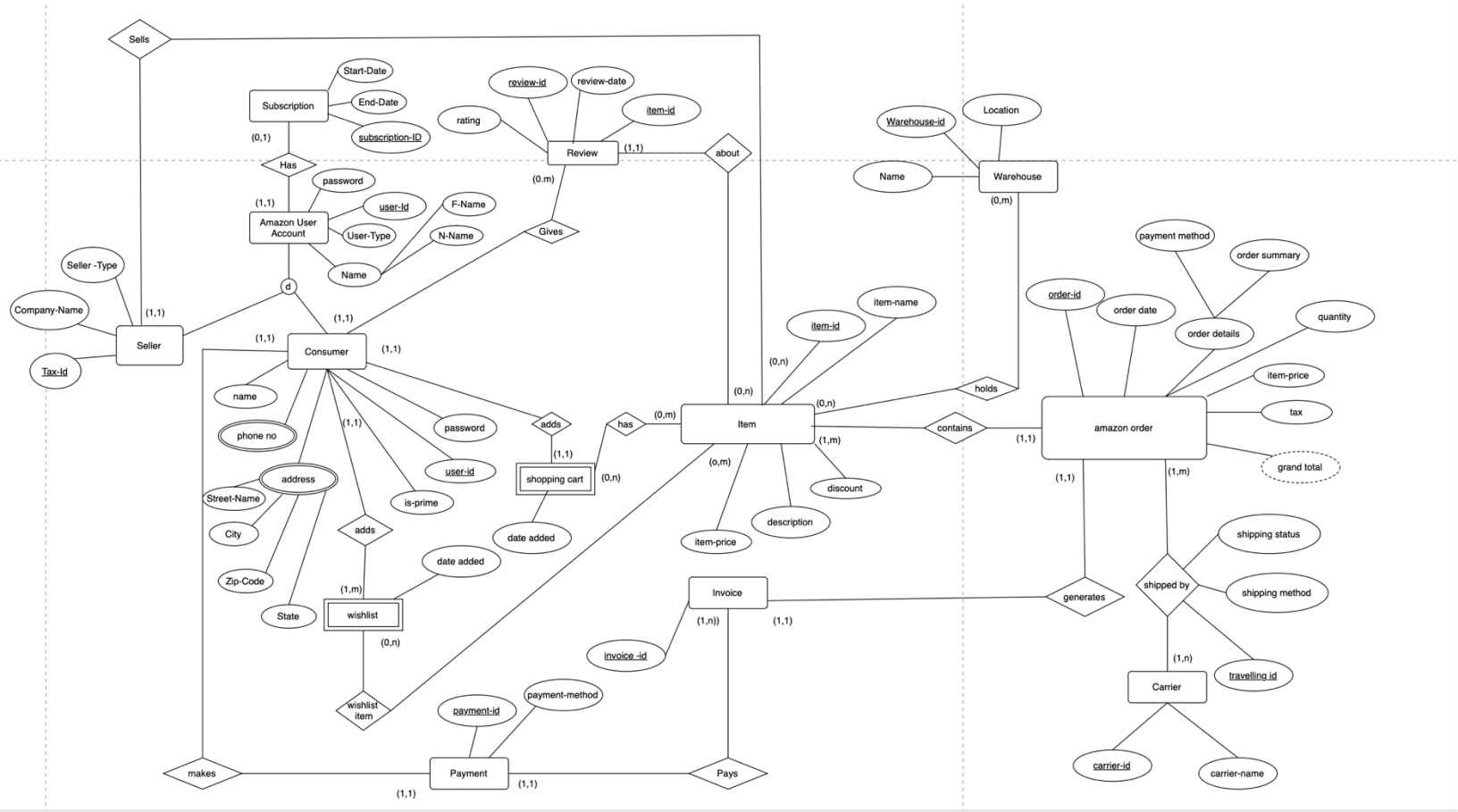
- An amazon order consists grand_total which has been derived from the combination of item_price and tax.
- More than one order can be shipped by multiple carriers.

  Carrier: Carrier_id, carrier_name

- The amazon order generates an invoice which leads to the payment that is to be done by the consumer and the order is shipped by the carrier. The order can contain one or many items.

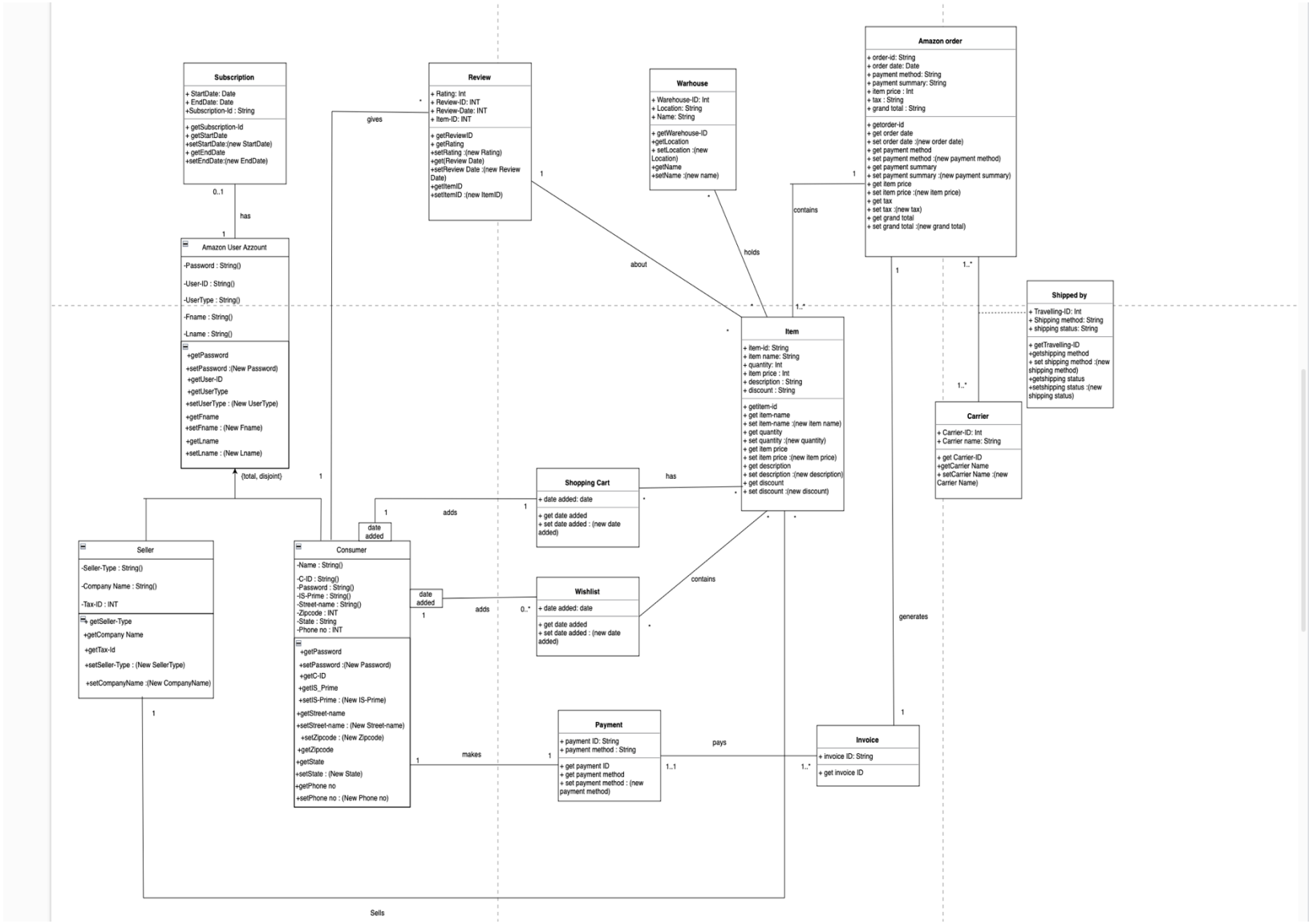
  Amazon_order:order_id, order_date, item_price, tax,
  Invoice: Invoice_id
  Payment: Payment_id, Payment_Method

EER :



- Transactional data:
  start_date, end_date, payment, item_price, tax
- Referential data:
  Invoice_id, travelling_id, order_id, item_id, review_id, user_id, subscription_id, tax_id

UML:



Relational Model :

Amazon User Account(<u>UserID</u>, User_Type, Password, Fname, Lname, *SubscriptionID*)

- UserID is a PRIMARY KEY
- UserID is NOT NULL
- Password is NOT NULL
- SubscriptionID is a FOREIGN KEY referencing SubscriptionID in Subscription Table
- SubscriptionID can be NULL

Subscription(SubscriptionID,EndDate,StartDate)

- SubscriptionID is a PRIMARY KEY
- SubscriptionID is NOT NULL

Seller(*UserID*, taxID, CompanyName,SellerType, password, user_type, Fname, Lname)

- UserID acts as a PRIMARY KEY to this table
- UserID is NOT NULL

Item(itemID, item_name, quantity, item price, description, discount, *orderID, reviewID, userID*)

- ItemID is a PRIMARY KEY (NOT NULL)
- userID is a FOREIGN KEY referencing userID in Seller Table
- userID is NOT NULL
- reviewID is a FOREIGN KEY referencing reviewID in Review Table
- reviewID is NOT NULL
- orderID is a FOREIGN KEY referencing orderID in amazon order Table
- orderID is NOT NULL

Warehouse(WarehouseID, name, location)

- WarehouseID is a PRIMARY KEY(NOT NULL)

Holds(*ItemID, warehouseID*)

- ItemID is a FOREIGN KEY referencing ItemID in Item Table
- WarehouseID is a FOREIGN KEY referencing WarehouseID in Warehouse Table
- ItemID and WarehouseID is a PRIMARY KEY of this table.
- ItemID and WarehouseID are NOT NULL

Amazon Order(Order_ID, order_date, order_details ,payment method, order summary, item_price, tax, grand_total, *InvoiceID*)

- Order_ID is a PRIMARY KEY(NOT NULL)
- InvoiceID is a FOREIGN KEY referencing invoiceID in Invoice table
- InvoiceID is NOT NULL

ShippedBy(TravelingID, *order_ID, carrierID*, shipping status, shipping method)

- TravelingID is a PRIMARY KEY (NOT NULL)
- order_ID is a FOREIGN KEY referencing order_ID in Amazon oder table

- orderID is NOT NULL
- carrierID is a FOREIGN KEY referencing carrierID in Carrier Table
- carrierID is NOT NULL

Invoice(invoiceID, *paymentID*)
- InvoiceID is a PRIMARY KEY (NOT NULL)
- PaymentID is a FOREIGN KEY referencing paymentID in Payment Table.
- PaymentID is NOT NULL

Payment(PaymentId, payment_method)
- PaymentId is a PRIMARY KEY of this table (NOT NULL)

Consumer(*userID*, password, id_prime, name, phone_no.,*paymentID*)
- UserID IS A PRIMARY KEY of this table (NOT NULL)
- PaymentID is a FOREIGN KEY referencing paymentID in Payment Table
- PaymentID is NOT NULL

Address(*UserID*, street name, city, zip code, state)
- UserID is a FOREIGN KEY referencing UserID in Consumer table
- UserID is also a PRIMARY KEY (NOT NULL)

Wishlist(*userID,* date_added)
- Wishlist is a weak entity
- UserID is a PRIMARY KEY of this table (NOT NULL)
- UserID is also a FOREIGN KEY referencing Consumer Table

Wishlist_Item(*userID,ItemID)*
- UserID is also a FOREIGN KEY referencing UserID in Consumer Table
- ItemId is also a FOREIGN KEY referencing ItemID in Item Table
- ItemId and UserID is a PRIMARY KEY of this table (NOT NULL)

Review(review_id, review date, rating ,*UserID)*
- review_id is a PRIMARY KEY(NOT NULL)
- UserId is a FOREIGN KEY referencing userID in Consumer table
- userID is NOT NULL

Carrier(carrierID, carrier_name)

- CarrierID is a PRIMARY KEY(NOT NULL)

Shopping cart(userID, date_added)

- Shopping cart is a weak entity
- UserID is a primary key of this table (NOT NULL)

ShoppingCart_Item(*UserId,ItemID*)

- UserID is a FOREIGN KEY referencing UserID in ConsumerID
- ItemID is a FOREIGN KEY referencing ItemID in Item table
- UserID and ItemID is a PRIMARY KEY of this table (NOT NULL)

Amazonorder_item(*Item_ID, Order_ID,* Item_Price)

- Item_ID is a FOREIGN KEY referencing UserID in ConsumerID
- Order_ID is a FOREIGN KEY referencing ItemID in Item table
  Item_ID and Order_ID is a PRIMARY KEY of this table (NOT NULL)

SQL Implementation :

1. CREATION OF TABLES

```
CREATE TABLE `Address` (
  `userID` int NOT NULL,
  `street_name` varchar(20) NOT NULL,
  `city` varchar(20) DEFAULT NULL,
  `zip_code` varchar(20) DEFAULT NULL,
  `state` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`userID`),
  CONSTRAINT `address_ibfk_1` FOREIGN KEY (`userID`) REFERENCES `Consumer` (`userID`)
)

CREATE TABLE `Amazon_Order` (
  `Order_ID` int NOT NULL,
  `order_date` date DEFAULT NULL,
  `order_details` varchar(40) DEFAULT NULL,
  `payment_method` varchar(20) DEFAULT NULL,
  `order_summary` varchar(40) DEFAULT NULL,
  `item_price` int NOT NULL,
  `tax` int NOT NULL,
  `grand_total` int DEFAULT NULL,
  `invoiceID` varchar(40) NOT NULL,
```

```
  `userID` int DEFAULT NULL,
  PRIMARY KEY (`Order_ID`),
  KEY `invoiceID` (`invoiceID`),
  KEY `userID` (`userID`),
  CONSTRAINT `amazon_order_ibfk_1` FOREIGN KEY (`invoiceID`) REFERENCES `Invoice` (`invoiceID`),
  CONSTRAINT `amazon_order_ibfk_2` FOREIGN KEY (`userID`) REFERENCES `Amazon_User_Account`
(`UserID`)
)

CREATE TABLE `Amazon_User_Account` (
  `UserID` int NOT NULL,
  `User_Type` varchar(12) DEFAULT NULL,
  `Password` varchar(20) DEFAULT NULL,
  `Name_User` char(30) DEFAULT NULL,
  `SubscriptionID` varchar(10) DEFAULT NULL,
  `Payment_ID` int DEFAULT NULL,
  PRIMARY KEY (`UserID`),
  KEY `SubscriptionID` (`SubscriptionID`),
  CONSTRAINT `amazon_user_account_ibfk_1` FOREIGN KEY (`SubscriptionID`) REFERENCES
`SUBSCRIPTION` (`SubscriptionID`)
)

CREATE TABLE `carrier` (
  `Carrier_ID` int NOT NULL,
  `carrier_Name` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`Carrier_ID`)
)

CREATE TABLE `consumer` (
  `userID` int NOT NULL,
  `Cons_password` varchar(20) NOT NULL,
  `is_prime` varchar(20) DEFAULT NULL,
  `cons_name` varchar(20) NOT NULL,
  `phone_no` bigint DEFAULT NULL,
  `Payment_ID` int DEFAULT NULL,
  PRIMARY KEY (`userID`),
  KEY `Payment_ID` (`Payment_ID`),
  CONSTRAINT `consumer_ibfk_1` FOREIGN KEY (`Payment_ID`) REFERENCES `Payment` (`Payment_ID`)
)

CREATE TABLE `invoice` (
  `invoiceID` varchar(40) NOT NULL,
  `Payment_ID` int NOT NULL,
  PRIMARY KEY (`invoiceID`),
  KEY `Payment_ID` (`Payment_ID`),
  CONSTRAINT `invoice_ibfk_1` FOREIGN KEY (`Payment_ID`) REFERENCES `Payment` (`Payment_ID`)
)

CREATE TABLE `item` (
  `ItemID` int NOT NULL,
  `Item_name` varchar(40) DEFAULT NULL,
```

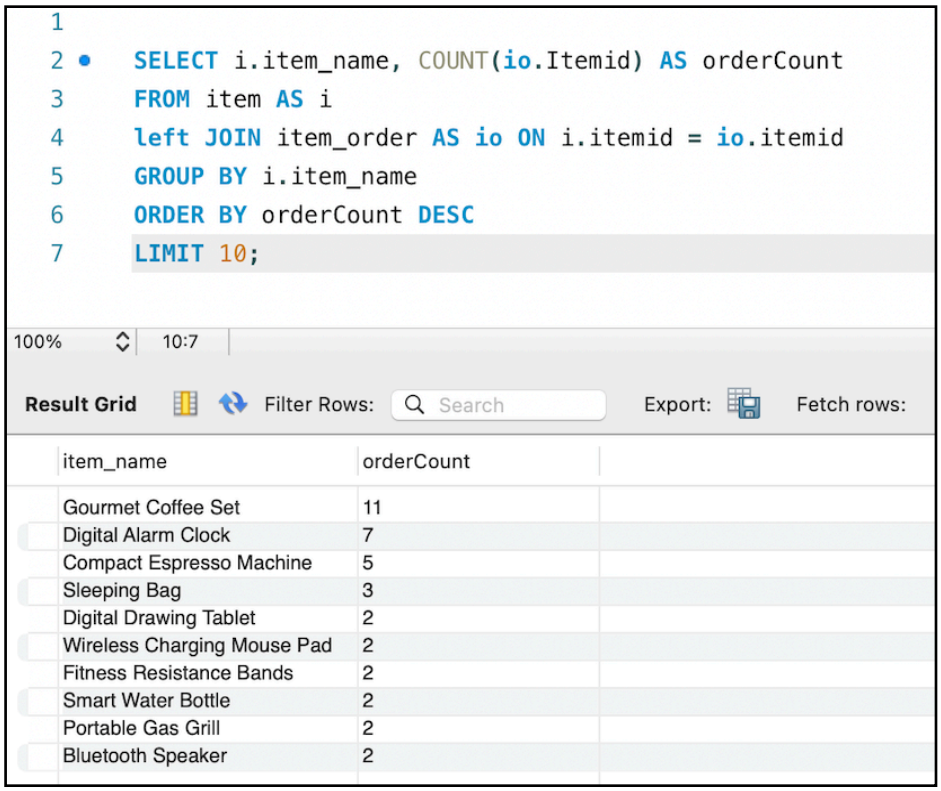
```
  `quantity` int DEFAULT NULL,
  `item_price` int DEFAULT NULL,
  `description` varchar(100) DEFAULT NULL,
  `discount` varchar(10) DEFAULT NULL,
  `Order_ID` int DEFAULT NULL,
  `Review_ID` int DEFAULT NULL,
  `UserID` int DEFAULT NULL,
  PRIMARY KEY (`ItemID`),
  KEY `UserID` (`UserID`),
  KEY `Review_ID` (`Review_ID`),
  KEY `Order_ID` (`Order_ID`),
  CONSTRAINT `item_ibfk_1` FOREIGN KEY (`UserID`) REFERENCES `Amazon_User_Account` (`UserID`),
  CONSTRAINT `item_ibfk_2` FOREIGN KEY (`Review_ID`) REFERENCES `Review` (`Review_ID`),
  CONSTRAINT `item_ibfk_3` FOREIGN KEY (`Order_ID`) REFERENCES `Amazon_Order` (`Order_ID`)
)

CREATE TABLE `item_order` (
  `order_ID` int NOT NULL,
  `ItemID` int NOT NULL,
  PRIMARY KEY (`order_ID`,`ItemID`),
  KEY `ItemID` (`ItemID`),
  CONSTRAINT `item_order_ibfk_1` FOREIGN KEY (`order_ID`) REFERENCES `Amazon_Order` (`Order_ID`),
  CONSTRAINT `item_order_ibfk_2` FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ItemID`)
)

CREATE TABLE `payment` (
  `Payment_ID` int NOT NULL,
  `Payment_method` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`Payment_ID`)
)

CREATE TABLE `review` (
  `Review_ID` int NOT NULL,
  `review_date` date DEFAULT NULL,
  `rating` decimal(5,2) DEFAULT NULL,
  `userID` int NOT NULL,
  PRIMARY KEY (`Review_ID`),
  KEY `userID` (`userID`),
  CONSTRAINT `review_ibfk_1` FOREIGN KEY (`userID`) REFERENCES `Consumer` (`userID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `seller` (
  `UserID` int NOT NULL,
  `CompanyName` varchar(20) DEFAULT NULL,
  `SellerType` varchar(20) DEFAULT NULL,
  `password` varchar(20) DEFAULT NULL,
  `user_type` varchar(10) DEFAULT NULL,
  `Name_Seller` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`UserID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
CREATE TABLE `shipped_by` (
  `TravelingID` int NOT NULL,
  `Order_ID` int DEFAULT NULL,
  `Carrier_ID` int DEFAULT NULL,
  `shipping_status` varchar(20) DEFAULT NULL,
  `shipping_method` varchar(40) DEFAULT NULL,
  `shipping_date` date DEFAULT NULL,
  PRIMARY KEY (`TravelingID`),
  KEY `Order_ID` (`Order_ID`),
  KEY `Carrier_ID` (`Carrier_ID`),
  CONSTRAINT `shipped_by_ibfk_1` FOREIGN KEY (`Order_ID`) REFERENCES `Amazon_Order` (`Order_ID`),
  CONSTRAINT `shipped_by_ibfk_2` FOREIGN KEY (`Carrier_ID`) REFERENCES `Carrier` (`Carrier_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `shopping_cart` (
  `userID` int NOT NULL,
  `date_added` date DEFAULT NULL,
  PRIMARY KEY (`userID`),
  CONSTRAINT `shopping_cart_ibfk_1` FOREIGN KEY (`userID`) REFERENCES `Consumer` (`userID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `subscription` (
  `SubscriptionID` varchar(10) NOT NULL,
  `Start_Date` date DEFAULT NULL,
  `End_Date` date DEFAULT NULL,
  PRIMARY KEY (`SubscriptionID`)
)

CREATE TABLE `warehouse` (
  `WarehouseID` int NOT NULL,
  `warehouse_name` varchar(40) DEFAULT NULL,
  `Location` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`WarehouseID`)
)
CREATE TABLE `wishlist` (
  `userID` int NOT NULL,
  `date_added` date DEFAULT NULL,
  PRIMARY KEY (`userID`),
  CONSTRAINT `wishlist_ibfk_1` FOREIGN KEY (`userID`) REFERENCES `Consumer` (`userID`)
)
```

2. TABLES ARE CREATED AND DATA IS ALSO GENERATED

3. ANALYSIS

1. On the thriving E-commerce platform, the top 10 most frequently ordered items by customers have become clear champions in popularity. These coveted products, driven by consumer demand and preferences, have consistently secured their positions as the go-to choices for shoppers.. From essential everyday item **like coffee sets, alarm clock** these top 10 products have established themselves as the heartbeat of the platform, embodying the pulse of consumer trends and preferences. As we celebrate the success of these frequently ordered items, it reflects not only their intrinsic value but also the dynamic nature of our platform, where customer choices shape the landscape of our bustling digital marketplace.



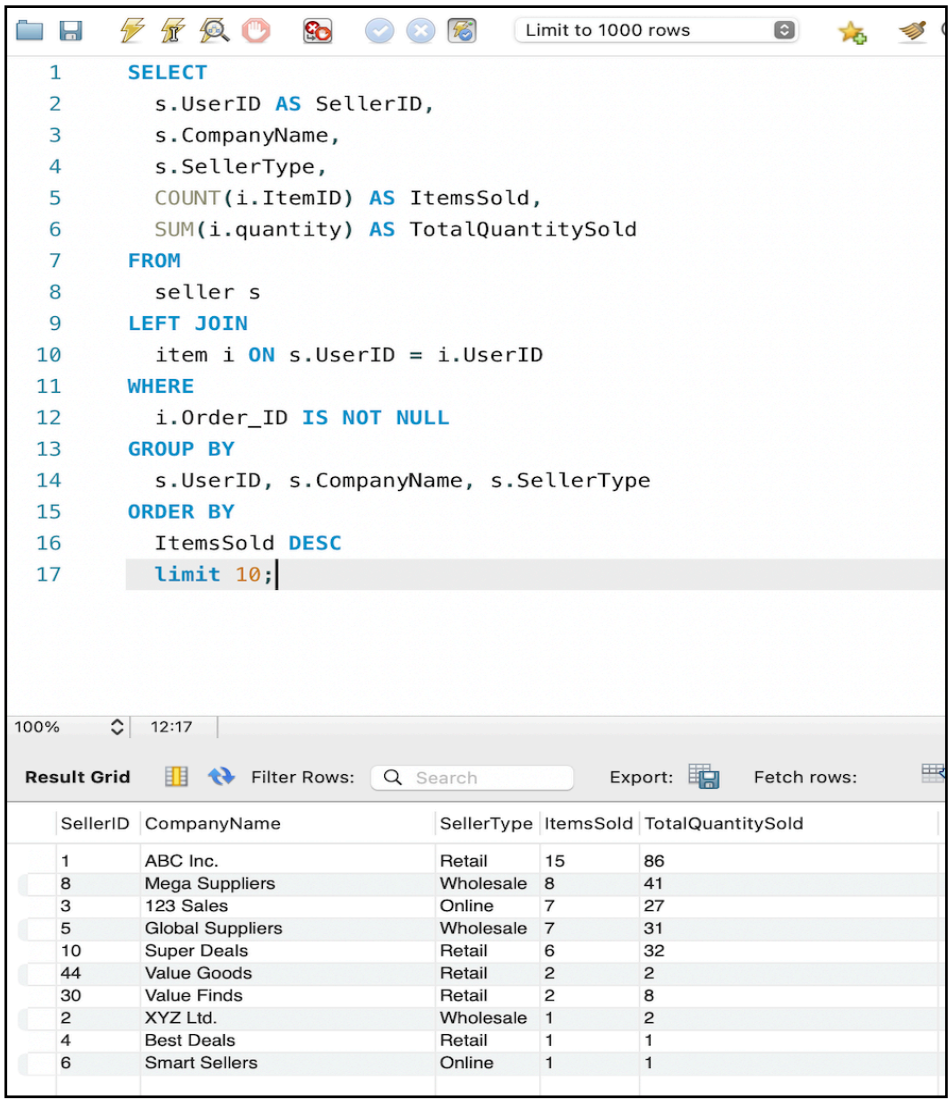
```
1
2 •  SELECT i.item_name, COUNT(io.Itemid) AS orderCount
3    FROM item AS i
4    left JOIN item_order AS io ON i.itemid = io.itemid
5    GROUP BY i.item_name
6    ORDER BY orderCount DESC
7    LIMIT 10;
```

100% 10:7

Result Grid Filter Rows: Search Export: Fetch rows:

item_name	orderCount
Gourmet Coffee Set	11
Digital Alarm Clock	7
Compact Espresso Machine	5
Sleeping Bag	3
Digital Drawing Tablet	2
Wireless Charging Mouse Pad	2
Fitness Resistance Bands	2
Smart Water Bottle	2
Portable Gas Grill	2
Bluetooth Speaker	2

2. Executing a SQL query on our e-commerce database reveals a comprehensive overview of **seller data**. The total items sold are meticulously tracked, showcasing the platform's dynamic marketplace. The differentiation between the sales of wholesale, retail and online sellers is discernible, providing insights into the diverse nature of our seller ecosystem. This data-driven approach not only facilitates strategic decision-making but also underscores the platform's commitment to transparency and performance evaluation. As we delve into the SQL results, we gain a nuanced understanding of the intricate relationships between seller types and their respective contributions to the overall transactional landscape. We see that the **Retail type of sellers sell the most items, both as in the variety of items and the quantity.**



SellerID	CompanyName	SellerType	ItemsSold	TotalQuantitySold
1	ABC Inc.	Retail	15	86
8	Mega Suppliers	Wholesale	8	41
3	123 Sales	Online	7	27
5	Global Suppliers	Wholesale	7	31
10	Super Deals	Retail	6	32
44	Value Goods	Retail	2	2
30	Value Finds	Retail	2	8
2	XYZ Ltd.	Wholesale	1	2
4	Best Deals	Retail	1	1
6	Smart Sellers	Online	1	1

3. The below results give us an insight about **users who have subscribed**, including their subscription details and whether they are Prime members. The LEFT JOIN ensures that even users without subscriptions are included in the result. The WHERE clause filters out users without subscriptions, and the result is ordered by UserID.

```sql
SELECT
  u.UserID,
  u.Name_User AS UserName,
  u.User_Type AS UserType,
  c.is_prime AS IsPrimeMember,
  s.SubscriptionID,
  s.Start_Date AS SubscriptionStartDate,
  s.End_Date AS SubscriptionEndDate
FROM
  Amazon_User_Account u
LEFT JOIN
  Subscription s ON u.SubscriptionID = s.SubscriptionID
LEFT JOIN
  Consumer c ON u.UserID = c.userID
WHERE
  u.SubscriptionID IS NOT NULL
ORDER BY
  u.UserID
  limit 10;
```

100% 3:20

Result Grid Filter Rows: Search Export: Fetch rows:

UserID	UserName	UserType	IsPrimeMember	SubscriptionID	SubscriptionStartDate	SubscriptionEndDate
1	John Doe	Customer	Yes	A12345	2023-01-01	2023-02-01
2	Alice Smith	Seller	No	B67890	2023-02-15	2023-03-15
3	Bob Johnson	Customer	Yes	C54321	2023-03-05	2023-04-05
4	Eva Davis	Seller	No	D98765	2023-04-10	2023-05-10
5	Charlie Brown	Customer	Yes	E23456	2023-05-20	2023-06-20
6	Sophie White	Seller	No	F78901	2023-06-25	2023-07-25
7	David Lee	Customer	Yes	G65432	2023-07-08	2023-08-08
8	Emma Taylor	Seller	No	H10987	2023-08-15	2023-09-15
9	Frank Thomas	Customer	Yes	I87654	2023-09-18	2023-10-18
10	Grace Martin	Seller	No	J32109	2023-10-22	2023-11-22

4. The below query provides a snapshot of consumer order trends, including the total number of orders, the first and latest order dates, and the duration between the first and latest orders. Adjustments can be made based on specific requirements or additional information you want to capture. We come across the top 10 loyal customers and the intervals/days in which they place their latest orders.

```sql
SELECT
  c.userID AS ConsumerID,
  c.cons_name AS ConsumerName,
  COUNT(o.Order_ID) AS TotalOrders,
  MIN(o.order_date) AS FirstOrderDate,
  MAX(o.order_date) AS LatestOrderDate,
  DATEDIFF(MAX(o.order_date), MIN(o.order_date)) AS DaysBetweenFirstAndLatestOrder
FROM
  consumer c
JOIN
  Amazon_Order o ON c.userID = o.userID
GROUP BY
  c.userID, c.cons_name
ORDER BY
  TotalOrders DESC
  limit 10;
```

100% 12:16

Result Grid Filter Rows: Search Export: Fetch rows:

ConsumerID	ConsumerName	TotalOrders	FirstOrderDate	LatestOrderDa...	DaysBetweenFirstAndLatestOr...
1	John Doe	9	2023-01-01	2023-04-11	100
3	Bob Johnson	6	2023-01-03	2023-07-02	180
8	Fiona Ford	6	2023-01-08	2023-09-27	262
10	Holly Hayes	5	2023-01-10	2023-10-05	268
4	Alice Williams	4	2023-01-04	2023-05-07	123
2	Jane Smith	2	2023-01-02	2023-02-08	37
5	Charlie Brown	2	2023-01-05	2023-05-07	122
6	Diana Davis	2	2023-01-06	2023-06-19	164
7	Evan Evans	2	2023-01-07	2023-07-02	176
9	George Gardner	2	2023-01-09	2023-09-27	261

5. Within our dynamic E-commerce realm, the top 10 customers have distinguished themselves not just by their frequent engagement but by the **recency of their orders,** embodying the ever-evolving preferences of our diverse clientele. These distinguished patrons have, with striking regularity, placed orders that reflect the cutting edge of consumer trends, underlining their keen eye for the latest and most coveted products on our platform. As we delve into the data, it becomes evident that these top 10 customers stand as the vanguards of our digital marketplace, consistently seeking the newest offerings and contributing to the vibrant tapestry of our ever-expanding product catalog. Their commitment to staying at the forefront of our inventory mirrors the dynamic nature of our platform, where the latest order date becomes a testament to their unwavering loyalty and the embodiment of our commitment to offering the best and most up-to-date products.

```
28    JOIN
29      (SELECT
30        o.Order_ID,
31        o.order_date,
32        o.order_details,
33        o.payment_method,
34        o.order_summary,
35        o.item_price,
36        o.tax,
37        o.grand_total,
38        o.userID
39      FROM
40        Amazon_Order o) AS o ON a.userID = o.userID
41    JOIN
42      (SELECT
43        u.UserID,
44        u.User_Type,
45        u.Name_User,
46        u.SubscriptionID
47      FROM
```

100% 13:49

Result Grid Filter Rows: Search Export: Fetch rows:

UserID	street_name	city	zip_code	state	Order_ID	order_date	order_details	payment_meth...	order_summary	item_price	tax	grand_total	User_Type	Name_User	Subscription...
1	123 Main St	New York City	10001	NY	1	2023-01-01	Details1	Credit Card	Summary1	100	10	110	Customer	John Doe	A12345
2	456 Oak St	Los Angeles	90001	CA	2	2023-01-02	Details2	PayPal	Summary2	150	15	165	Seller	Alice Smith	B67890
3	789 Pine St	Chicago	60601	IL	3	2023-01-03	Details3	Debit Card	Summary3	120	12	132	Customer	Bob Johnson	C54321
4	101 Maple Ave	Houston	77002	TX	4	2023-01-04	Details4	Credit Card	Summary4	80	8	88	Seller	Eva Davis	D98765
5	202 Cedar Blvd	Phoenix	85001	AZ	5	2023-01-05	Details5	PayPal	Summary5	200	20	220	Customer	Charlie Brown	E23456
6	303 Elm Ln	Philadelphia	19101	PA	6	2023-01-06	Details6	Debit Card	Summary6	90	9	99	Seller	Sophie White	F78901
7	404 Birch Dr	San Antonio	78201	TX	7	2023-01-07	Details7	Credit Card	Summary7	110	11	121	Customer	David Lee	G65432
8	505 Oak Rd	San Diego	92101	CA	8	2023-01-08	Details8	PayPal	Summary8	130	13	143	Seller	Emma Taylor	H10987
9	606 Pine Ct	Dallas	75201	TX	9	2023-01-09	Details9	Debit Card	Summary9	180	18	198	Customer	Frank Thomas	I87654
10	707 Maple Ave	San Jose	95101	CA	10	2023-01-10	Details10	Credit Card	Summary10	95	10	105	Seller	Grace Martin	J32109

6.In the vibrant landscape of our online platform, the top 10 customers have not only showcased their penchant for the latest trends but have also contributed significantly to our thriving digital marketplace. Now, delving into a more nuanced aspect of their engagement, **we explore the average grand total spent by each user—a metric that encapsulates the breadth and depth of their transactions.** Through the lens of correlated queries, we uncover the distinctive spending habits of our users. The query drills down into the average grand total for every individual, offering a nuanced understanding of their economic footprint on our platform. These figures not only serve as a testament to the diverse preferences of our users but also highlight the unique contribution of each customer to the overall financial tapestry of our digital ecosystem. As we unravel the intricacies of user spending patterns, it becomes evident that our top customers are not only trendsetters in product choices but also integral players in shaping the financial dynamics of our ever-evolving online marketplace.

```sql
SELECT
  ao.Order_ID,
  ao.order_date,
  ao.order_details,
  ao.payment_method,
  ao.order_summary,
  ao.item_price,
  ao.tax,
  ao.grand_total,
  ao.invoiceID,
  ao.userID,
  (
    SELECT AVG(ao2.grand_total)
    FROM amazon_order ao2
    WHERE ao2.userID = ao.userID
  ) AS avg_grand_total
FROM
  amazon_order ao
  limit 10;
```

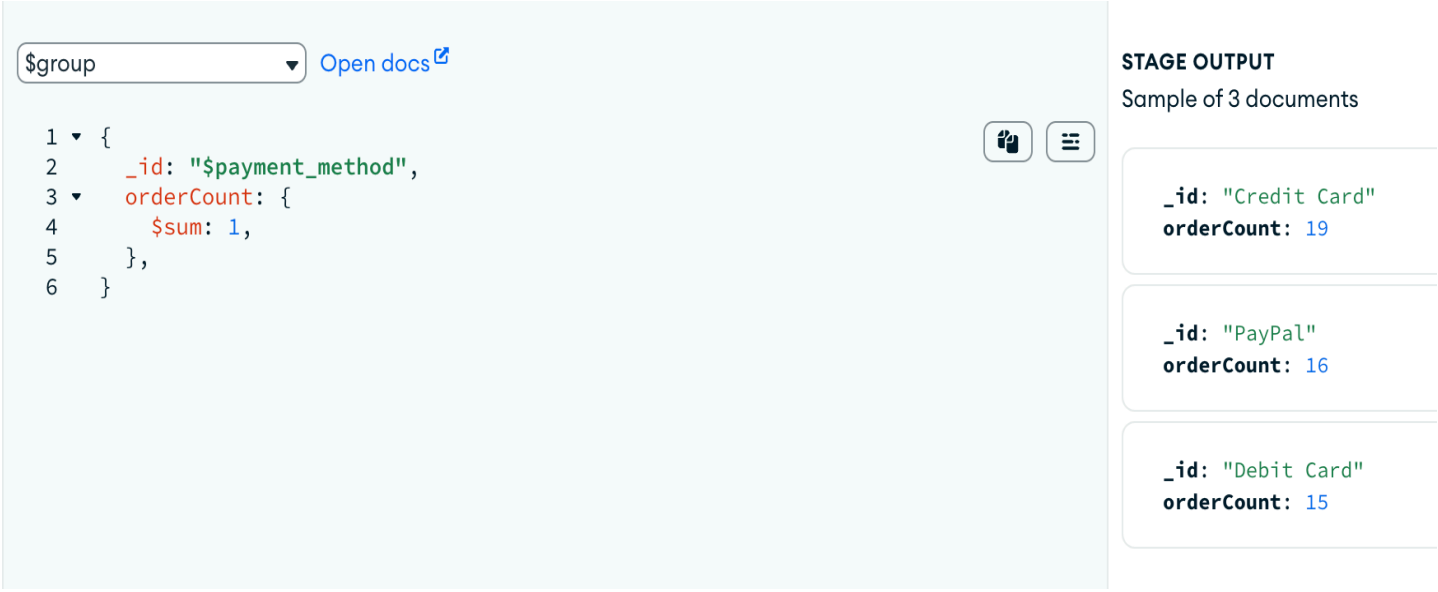
100% 11:19

Result Grid Filter Rows: Search Export: Fetch rows:

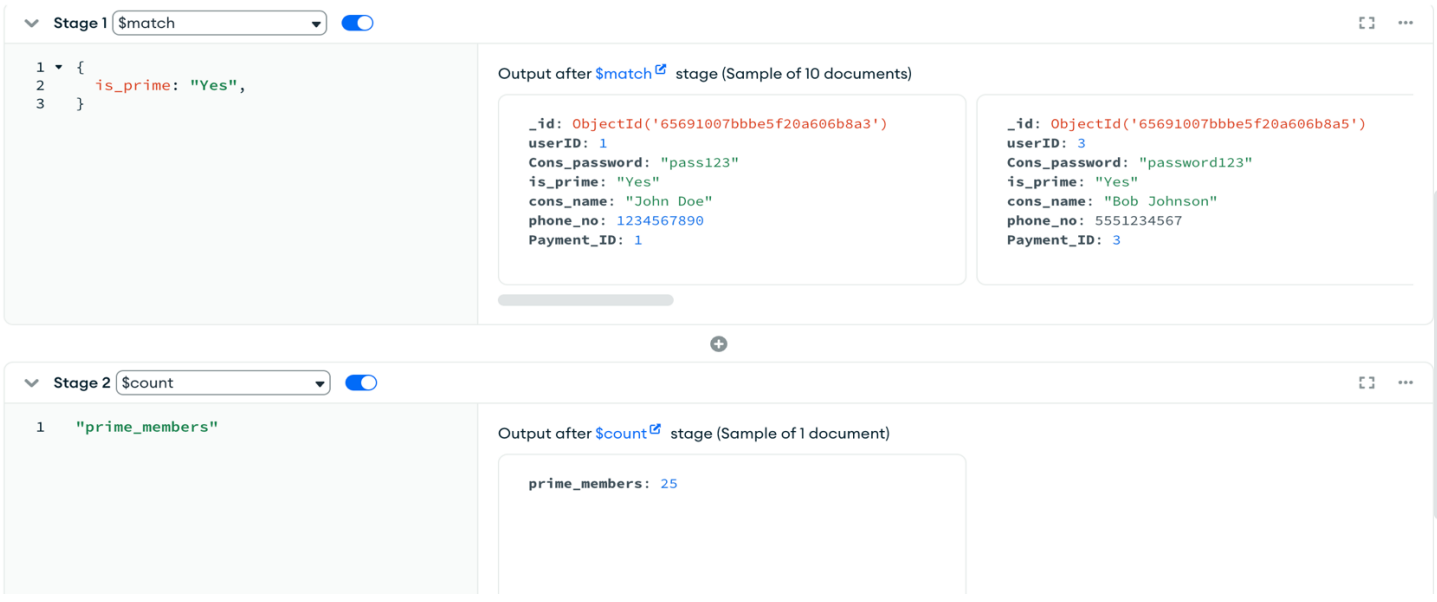
Order_ID	order_date	order_details	payment_meth...	order_summary	item_price	tax	grand_total	invoiceID	userID	avg_grand_total
1	2023-01-01	Details1	Credit Card	Summary1	100	10	110	INV2023001	1	96.4444
2	2023-01-02	Details2	PayPal	Summary2	150	15	165	INV2023002	2	123.5000
3	2023-01-03	Details3	Debit Card	Summary3	120	12	132	INV2023003	3	91.6667
4	2023-01-04	Details4	Credit Card	Summary4	80	8	88	INV2023004	4	104.5000
5	2023-01-05	Details5	PayPal	Summary5	200	20	220	INV2023005	5	159.5000
6	2023-01-06	Details6	Debit Card	Summary6	90	9	99	INV2023006	6	82.5000
7	2023-01-07	Details7	Credit Card	Summary7	110	11	121	INV2023007	7	104.5000
8	2023-01-08	Details8	PayPal	Summary8	130	13	143	INV2023008	8	119.3333
9	2023-01-09	Details9	Debit Card	Summary9	180	18	198	INV2023009	9	151.5000
10	2023-01-10	Details10	Credit Card	Summary10	95	10	105	INV2023010	10	78.2000

NOSQL Implementation:

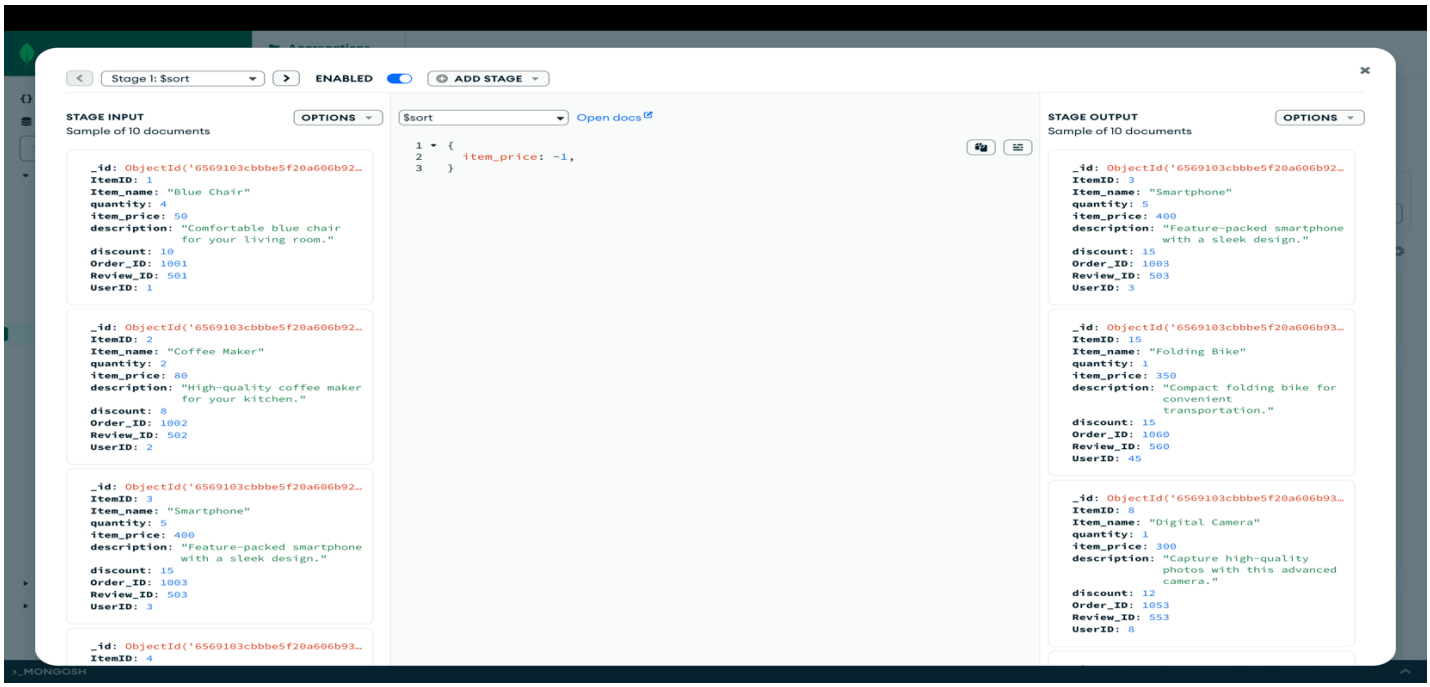
1. The query is aggregating data from the "amazon_order" collection based on the payment method. It then calculates the count of orders for each unique payment method. The result will be a summary of the total number of orders for each distinct payment method in the collection. This kind of analysis can provide insights into the distribution of orders across different payment methods, helping in business decision-making and strategy planning.

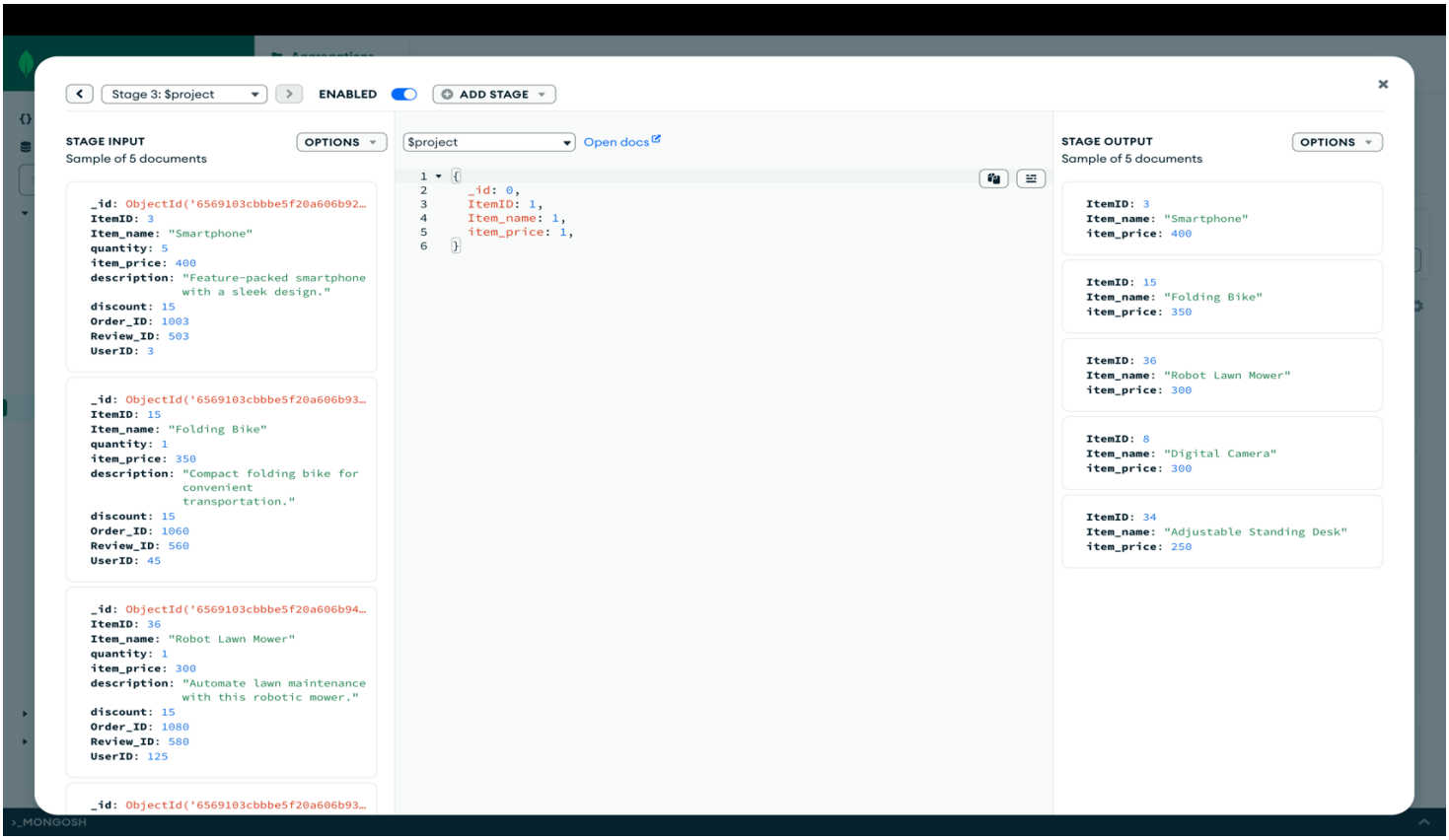


2. To determine the count of documents in the 'prime_members' collection where the field 'is_prime' is marked as "Yes." This could be valuable for analyzing and understanding the distribution or prevalence of prime members within the dataset. Prime members might represent a special group of users or customers with specific privileges or characteristics, and counting them could be essential for business insights, marketing strategies, or personalized services tailored to this group.



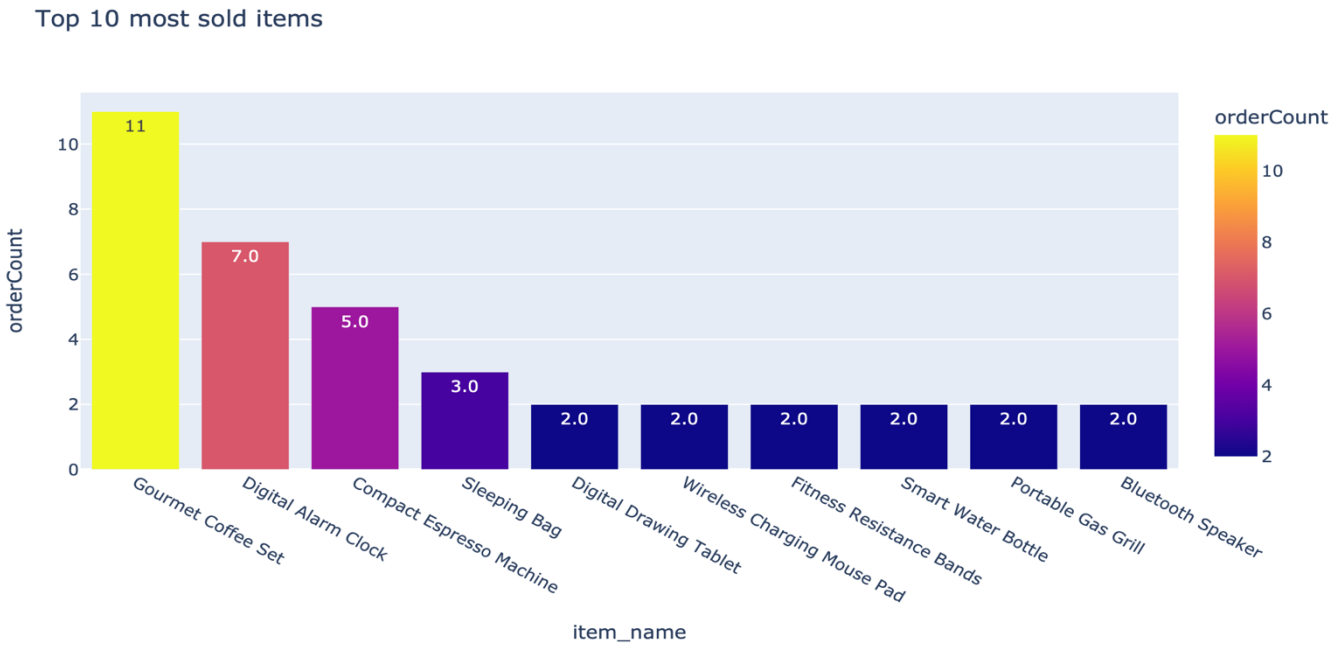
3. To identify, display, and potentially analyze the top 5 items based on their prices. By sorting in descending order of item prices, limiting the result to only the top 5 items, and projecting specific fields, the query facilitates a focused examination of the most expensive items in the dataset. This could be valuable for pricing analysis, inventory management, or other business strategies that involve understanding and highlighting high-value items.



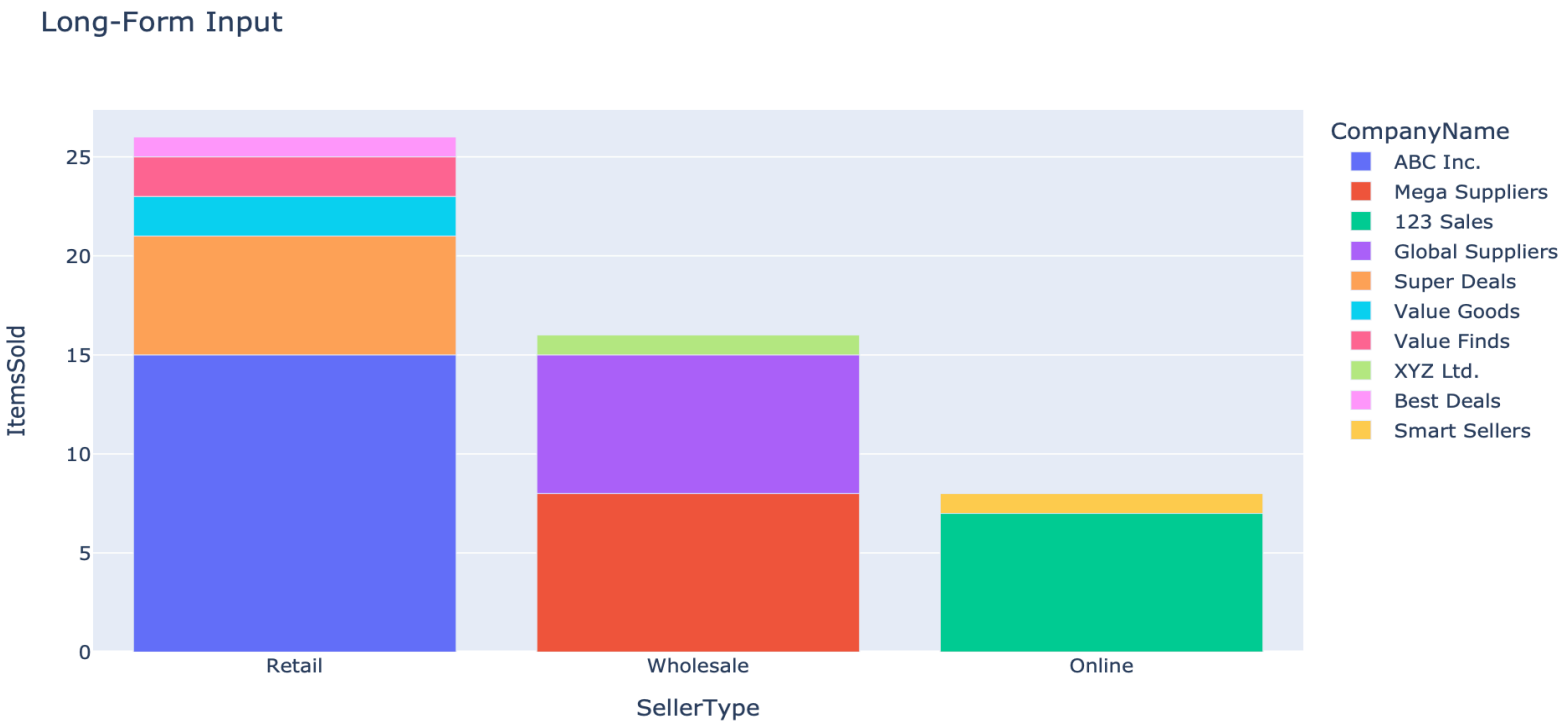


Python Analysis:

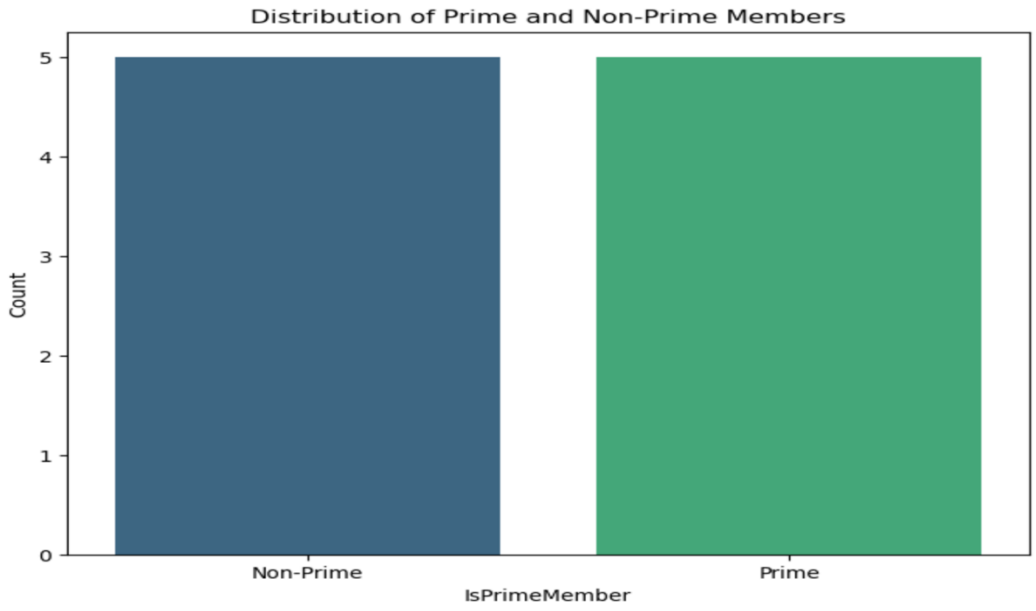
1. Top 10 most sold items



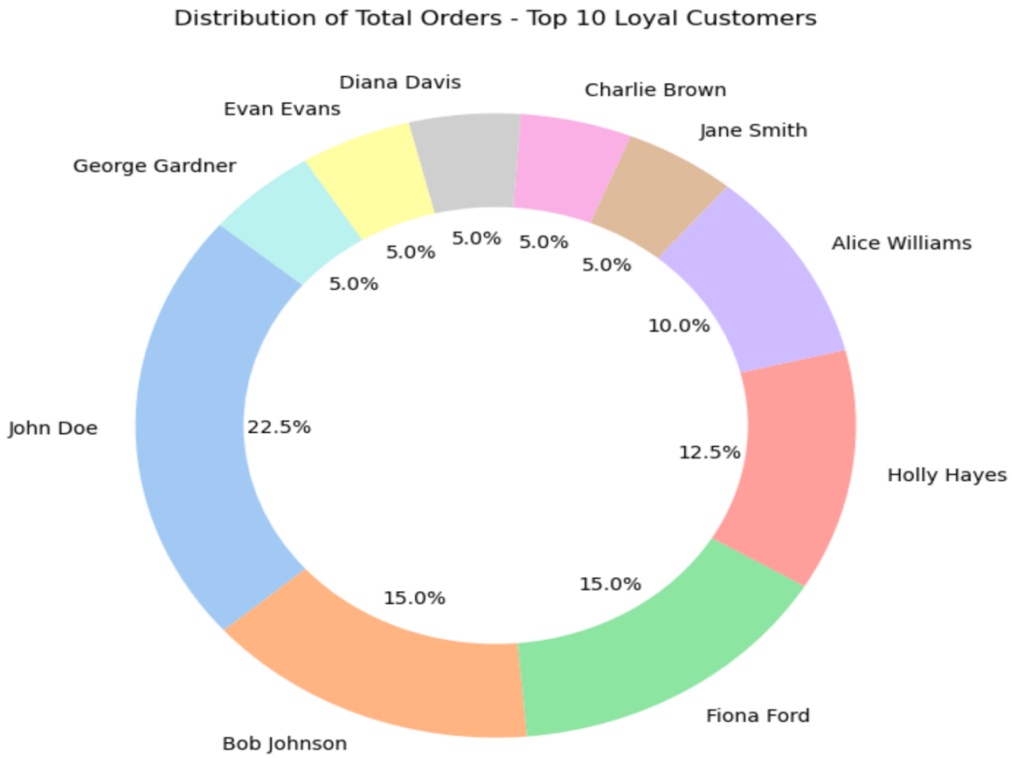
2. Items sold by types of seller



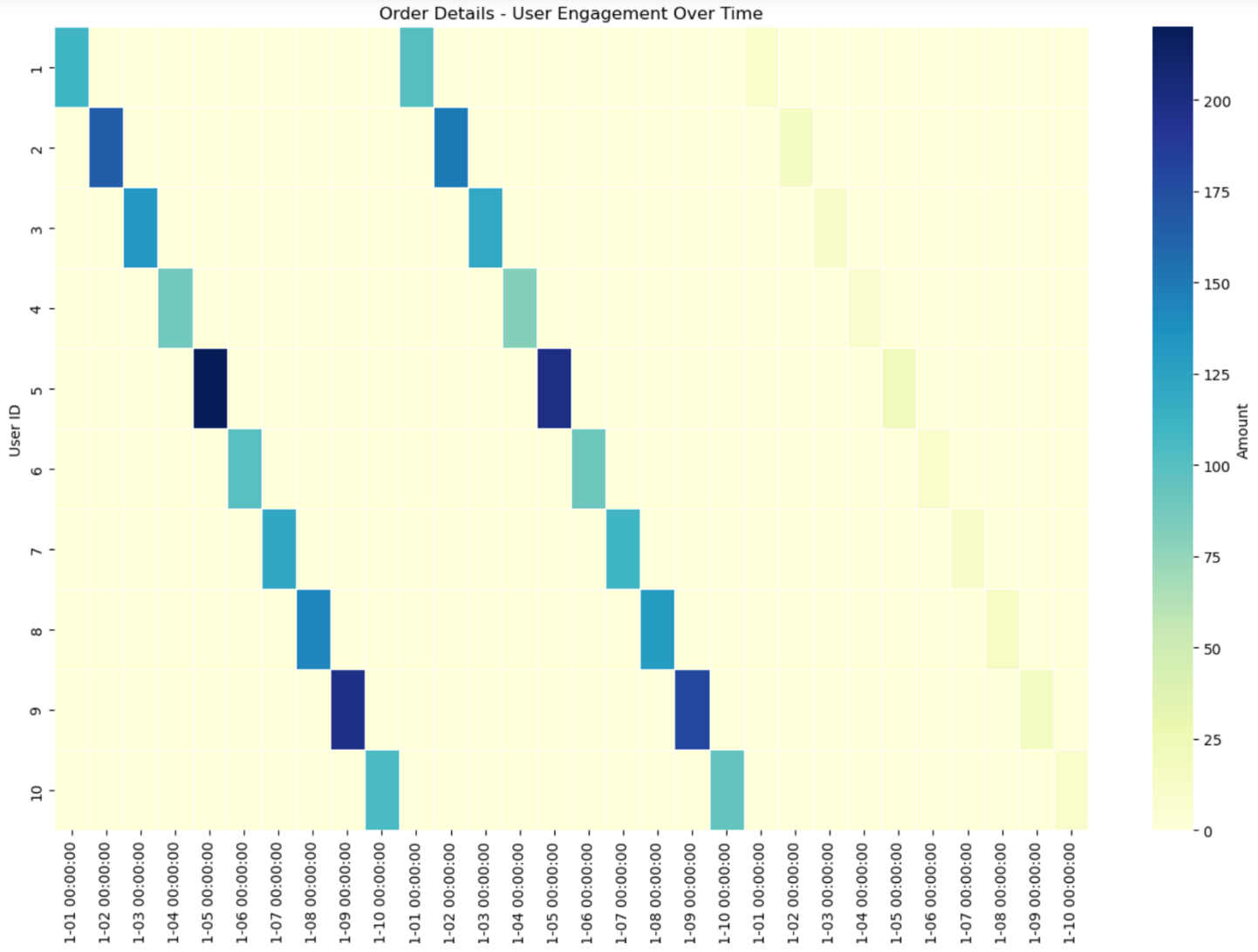
3.Distribution of Prime and Non-Prime Members



4.Top 10 loyal customers



5.User Engagement Over Time



Summary and Recommendation:

The E-commerce Database Management System project aims to create a robust foundation for a dynamic and scalable online retail platform. By addressing key components such as product management, order processing, customer nformation, and security measures, the system seeks to optimize both operational efficiency and customer satisfaction in the competitive e-commerce landscape.

- Pricing Strategy: The information about the top-priced items can be crucial for refining the pricing strategy. Consider reviewing the pricing structure of these high-value items and assess whether adjustments are needed to optimize revenue.
- Inventory Management: Identify the inventory levels and availability of the top-priced items. Ensure that there is sufficient stock to meet potential demand, especially for items that contribute significantly to revenue.
- Marketing Focus: Highlight the top-priced items in marketing campaigns to leverage their premium status. This can attract customers looking for high-quality or exclusive products.
- Competitor Analysis: Compare the prices of the identified top items with those of competitors. This analysis can help in positioning products competitively in the market.
- Customer Segmentation: Consider analyzing the customer demographics or behaviors associated with purchases of these high-priced items. This can guide targeted marketing efforts and personalized services for specific customer segments.
- Regular Updates: Since pricing and product popularity can change over time, it's advisable to periodically run similar analyses to stay informed about evolving trends and adjust strategies accordingly.

By incorporating these recommendations, the business can capitalize on the insights gained from the analysis, leading to informed decision-making and potentially improving overall performance in terms of revenue and customer satisfaction.