Database Foundations for Business Analytics

E-commerce Product Recommendation System

Group 14

Adharsha Velen(axg230033@utdallas.edu)
Arwin Kumar Ravi(axr220249@utdallas.edu)
Barath Kumar Dhanasekar(bxd220033@utdallas.edu)
Mahadevan Ramanan(mxr230040@utdallas.edu)
Prathamesh P Nagraj(pxn230011@utdallas.edu)
Vijay Refkin(vxp230008@utdallas.edu)
Jesse Jackson(jxy230003@utdallas.edu)

Abstract

This project report delineates the idea, design, modelling and implementation phases of the E-Commerce Product Recommendation System, with the aim of enhancing user engagement and boosting sales on an e-commerce platform. The document commences with an executive summary of the idea, providing a comprehensive overview of the project. Section 1 introduces the project, while Section 2 presents the Entity-Relationship (ER/EER) diagram and underlying assumptions. Section 3 expounds on the relational schema derived from the ER/EER diagram, outlining relationships, and accompanied by tabular data format specifications. Functional dependencies and normalization to the third normal form (3NF) are meticulously documented in Section 4. The report concludes with a concise summary, paving the way for the implementation phase and potential future refinements to address practical challenges and evolving requirements in the E-Commerce Product Recommendation System.

TABLE OF CONTENTS

1. ABOUT THE PROJECT:

1.1. Objective

This project seeks to create an E-Commerce Product Recommendation System designed to enrich the shopping experience for users on an e-commerce platform. Our primary objective is to address the challenge of enhancing user engagement, boosting sales, and elevating the shopping experience through personalized product recommendations. This approach aims to stimulate revenue growth, reinforce user loyalty, and establish a competitive edge for the e-commerce platform. We also aim to manage the inventory, by keeping track of the product availability.

1.2. Information Required:

To achieve our project goals, we need to gather and analyse various types of data, including:

- User profiles and preferences
- Product details and inventory information
- Brand information
- Product categories
- User interactions with products and categories
- User recommendations and recommendation scores
- Sales and order data

1.3. Role Distribution:

- Prathamesh Nagraj: Database Administrator, Documentation
- Mahadevan Ramanan: Database Designer, Documentation
- Adharsha Velen: Database Administrator, Database Designer
- Arwin Kumar Ravi: Database Administrator, Database Designer
- Barath Kumar Dhanasekar: Database Administrator, Database Designer
- Jesse Jackson: Data Analyst, Database Administrator
- Vijay Refkin: Data Analyst, Documentation

2. LOGIC AND CONCEPTUAL DESIGN:

2.1. ENTITY RELATIONSHIP DIAGRAM:

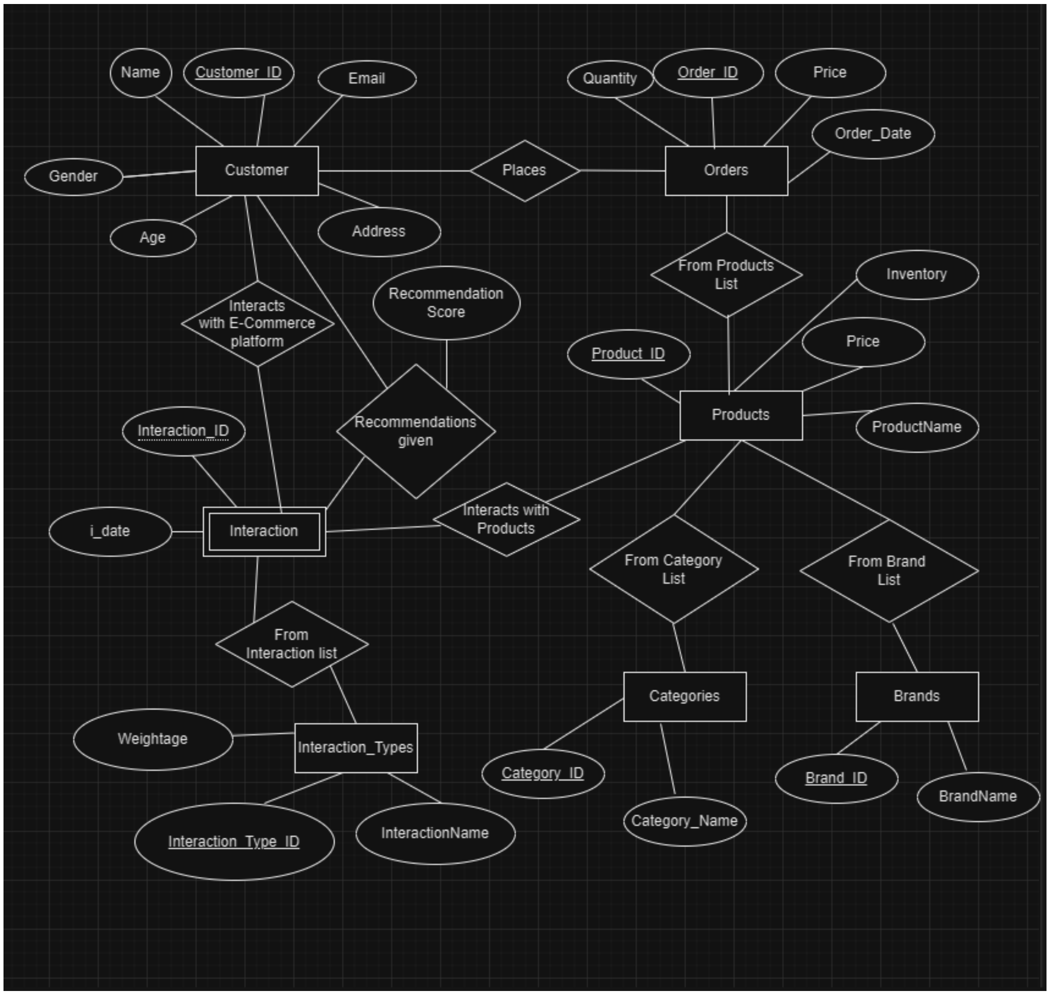


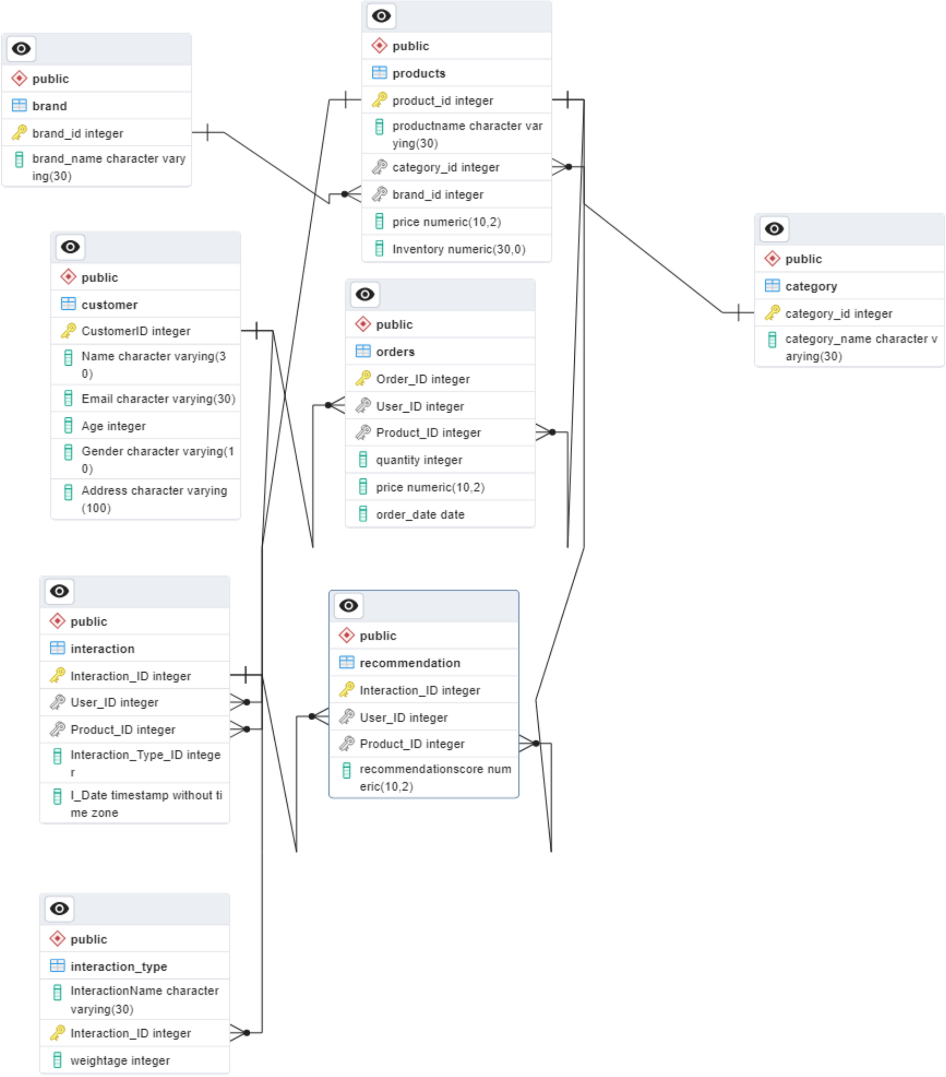
FIG 2.1 – ENTITY RELATIONSHIP DIAGRAM

2.2. CARDINALITY AND RELATIONSHIP EXPLANATION:

Expression	Discussion
Customer(0,N)------------->Orders(N,N)	One customer can place 0 to many orders(0,N). However, each order can be placed only by one customer(N,N)
Orders(N,N)------------------>Products(0,N)	Each order can contain one or more products(N,N), and each product may or may not be part of an order.(0,N)
Customer(1,N)--------------->Interactions(N,N)	Each user must have at least one interaction.(1,N) Each interaction must be associated with one user(N,N)

Products(1,1)----------------------->Brands(1,N)	Each product must belong to exactly one brand (1, 1). Each brand must be associated with at least one product (1, N).
Products(1,1)----------------------->Categories(1,N)	Each product must belong to exactly one brand (1, 1). Each category must be associated with at least one product (1, N).
Interactions(1,1)---------------->Interaction_Type	Each interaction must have exactly one interaction type (1, 1), and each interaction type must be associated with at least one interaction (1, N).

2.3. RELATIONAL SCHEMA DIAGRAM:



2.4 DATA FORMAT FOR EVERY RELATION:

Relation Name	Attributes	Datatype
Orders	Order_ID	Integer
	User_ID	Integer
	Product_ID	Integer
	Quantity	Integer
	Total_Price	Decimal
	Order_Date	MM/DD/YYYY, string = 10 chars

Relation Name	Attributes	Datatype
Products	Product Id	Integer
	ProductName	String <= 30
	Category_ID	Integer
	Brand_ID	Integer
	Price	Decimal

Relation Name	Attributes	Datatype
Brand	BrandName	String <= 30
	Brand_ID	Integer

Relation Name	Attributes	Datatype
Category	Category_ID	Integer
	Category_Name	String <= 30

Relation Name	Attributes	Datatype
Interaction_Type	InteractionName	String <= 30
	InteractionID	Integer
	Weightage	Decimal

Relation Name	Attributes	Datatype
Interaction	Interaction_ID	Integer
	User_ID	Integer
	Product_ID	integer
	Interaction_Type_ID	Integer
	I_time	datetime

Relation Names	Attributes	Datatype
Recommendation	Interaction_ID	Integer
	User_ID	Integer
	Product_ID	Integer
	Recommendation Score	Decimal

2.5 NORMALIZATION:

The defined schema and relational DB structure is already in normalized form. The below are the functional dependencies –

Customers:
CustomerID -> {Name, Email, Age, Gender, Address}

Orders:
Order_ID -> {CustomerID, Product_ID, Quantity, Total_Price, Order_Date}

Products:
Product_ID -> {ProductName, Category_ID, Brand_ID, Price}

Brands:
Brand_ID -> {BrandName}

Categories:
Category_ID -> {CategoryName}

Interaction_types:
Interaction_Type_ID -> {InteractionName}

Interactions:
Interaction_ID -> {CustomerID, Product_ID, Interaction_Type_ID, TimeStamp}

Recommendations:
Interaction_ID, CustomerID, Product_ID -> {RecommendationScore}

3. IMPLEMENTATION:

3.1 FUNCTIONS, TRIGGERS AND STORED PROCEDURES:

1. Function that takes different prompts and adds data to the orders table.

```
CREATE FUNCTION CreateOrder()
RETURNS INT
BEGIN
  DECLARE userid INT;
  DECLARE InproductID INT;
  DECLARE quantity INT;
  DECLARE totalcost DECIMAL(10,2);
  DECLARE continueOrder BOOLEAN DEFAULT TRUE;

  WHILE continueOrder DO
    SET continueOrder = FALSE;
    SET userid = EXISTS (SELECT * FROM Customers WHERE
customerid = INPUT('Enter your customer ID: '));
    IF NOT user THEN
      SELECT 'Invalid user ID!';
      CONTINUE;
    END IF;
    SET InproductID = INPUT('Enter product ID (or leave blank
to finish): ');
    IF InproductID IS NULL THEN
      LEAVE;
    END IF;
    SET InproductID = CONVERT_INT(InproductID);
    IF NOT EXISTS (SELECT * FROM Products WHERE ProductID =
InproductID) THEN
      SELECT 'Invalid product ID!';
      CONTINUE;
    END IF;
    SET price = (SELECT price FROM Products WHERE ProductID =
productID);

    -- Quantity and total cost
    SET quantity = CONVERT_INT(INPUT('Enter quantity: '));
    SET totalcost = price * quantity;
    SELECT 'Total cost for this item:', totalCost;

   -- Confirmation and order details
   IF UPPER(INPUT('Do you want to add this item (y/N)? ')) =
'Y' THEN
        INSERT INTO Orders (OrderID, CustomerId, ProductID,
Quantity, Cost, OrderDate)
        VALUES ((SELECT MAX(OrderID) + 1 FROM Orders),
Customerid, InproductID, quantity, totalCost, SYSDATE);

        -- Trigger stock update and order processing
        CALL trigger_placeorder();
        SET continueOrder = True;
   END IF;
  END WHILE;
   RETURN 0; -- Order successfully completed
END;

'
  -- Return latest order ID or 0
  RETURN IF lastOrderID IS NOT NULL THEN lastOrderID + 1 ELSE
0 END;
END;
```

2. **Trigger_placeorder():** This trigger is set to run after every insert into the orders table.

```
CREATE TRIGGER trigger_placeorder

AFTER INSERT ON Orders

FOR EACH ROW

BEGIN

  -- Update product stock

  UPDATE Products

  SET Stock = Stock - NEW.Quantity
  WHERE ProductID = NEW.ProductID;
  CALL ProcessOrders();
END;
```

3. **<u>ProcessOrders:</u>** This stored procedure processes the order, and displays the new order ID.

```
CREATE PROCEDURE ProcessOrders(newOrderID INT)
BEGIN
  SELECT 'Order placed successfully! Order ID:', newOrderID;
END;
```

4. **<u>trigger_AddProduct:</u>** This trigger is set to run every time a new product has been added to the products table.

```
   CREATE TRIGGER trigger_AddProduct
   AFTER INSERT ON Products
   FOR EACH ROW
   BEGIN
    DECLARE ProductName VARCHAR(255);
    DECLARE stock_quantity INT;
    DECLARE ProductID INT ;
      DECLARE Price DECIMAL(10,2);
      DECLARE CategoryID INT ;
      DECLARE CategoryName VARCHAR(255);
    SET ProductName = NEW.ProductName;
    SET stock_quantity = NEW.stock_quantity;
    SET ProductID = NEW.ProductID;
      SET Price - NEW.Price ;
      SET CategoryID = NEW.CategoryID;
      SET CategoryName = NEW.CategoryName;
CALL AddProduct(ProductName, stock_quantity, ProductID,Price,
CategoryID, CategoryName);
         END;
```

5. **<u>Add_Product:</u>** Set of commands that help in inserting data into the products table.

```
CREATE PROCEDURE AddProduct(
    IN p_ProductName VARCHAR(255),
    IN p_stock_quantity INT,
    IN p_ProductID INT,
    IN p_Price DECIMAL(10,2),
    IN p_CategoryID INT,
    IN p_CategoryName VARCHAR(255)
)
BEGIN
    -- Insert into Orders table
    INSERT INTO products (ProductID,
Quantity,p_stock_quantity, Cost)
    VALUES (p_ProductName,
p_ProductID,p_stock_quantity,p_Price);
    -- Update stock size in Products table
    UPDATE Products
    SET StockSize = StockSize + p_stock_quantity
    WHERE ProductID = p_ProductID;
END
```

6. **AfterInsertRecommendation:** This trigger is used to run the stored procedure for calculating the recommendations score for each user, for a particular product.

```
DELIMITER //

CREATE TRIGGER AfterInsertRecommendation
AFTER INSERT
ON interaction FOR EACH ROW
BEGIN
    -- Call the stored procedure to calculate and update
RecommendationScore
    CALL CalculateRecommendationScore(NEW.User_ID,
NEW.Product_ID);
END //
DELIMITER ;
```

7. **CalculateRecommendationScore:** This stored procedure is used to calculate the recommendation score for each user.

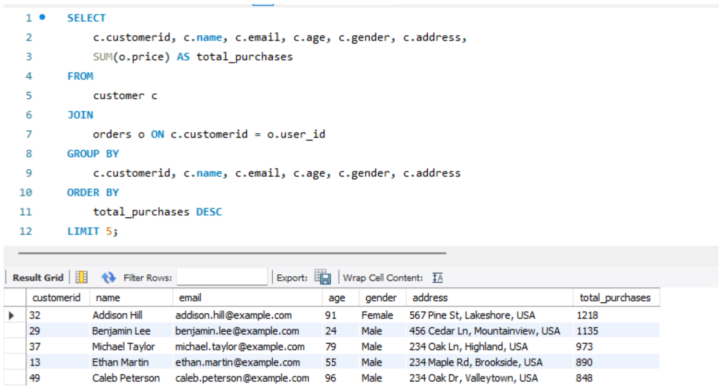
```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`CalculateRecommendationScore`(IN p_UserID INT, IN p_ProductID
INT)
BEGIN
    DECLARE mean_score DECIMAL(5, 2);
    DECLARE interaction_id_value INT;

    -- Calculate the mean of interaction weightages for the
specified User_ID and Product_ID
    SELECT AVG(IT.Weightage) INTO mean_score
    FROM Interaction I
    JOIN Interaction_Type IT ON I.Interaction_Type_ID =
IT.Interaction_ID
    WHERE I.User_ID = p_UserID AND I.Product_ID = p_ProductID;

    -- Get the Interaction_ID associated with the latest
interaction for the specified User_ID and Product_ID
    SELECT Interaction_ID INTO interaction_id_value
    FROM Interaction
    WHERE User_ID = p_UserID AND Product_ID = p_ProductID
    ORDER BY TimeStamp DESC
    LIMIT 1;
    -- Update the Recommendation Table with the calculated
RecommendationScore
    UPDATE recommendationtable
    SET RecommendationScore = mean_score, Interaction_ID =
interaction_id_value
    WHERE User_ID = p_UserID AND Product_ID = p_ProductID;

    -- If there is no existing record, insert a new record
    IF ROW_COUNT() = 0 THEN
        INSERT INTO recommendationtable (User_ID, Product_ID,
RecommendationScore, Interaction_ID)
        VALUES (p_UserID, p_ProductID, mean_score,
interaction_id_value);
    END IF;

END
```

3.2 SQL Queries for Data Analysis:

Analysing the dataset will help us in understanding trends and making key business decisions. We have created 10 queries, which are as follows:

1. **Details of top five customers with maximum expenditure:** This query will help identifying the top 5 customers, in terms of expenditure. The idea behind this query is to provide rewards for customers that purchase heavily in order to encourage them to purchase more.

```
SELECT c.customerid, c.name, c.email, c.age, c.gender,
c.address, SUM(o.price) AS total_purchases
FROM customer c
JOIN orders o ON c.customerid = o.user_id
GROUP BY c.customerid, c.name, c.email, c.age, c.gender,
c.address
ORDER BY total_purchases DESC
LIMIT 5;
```



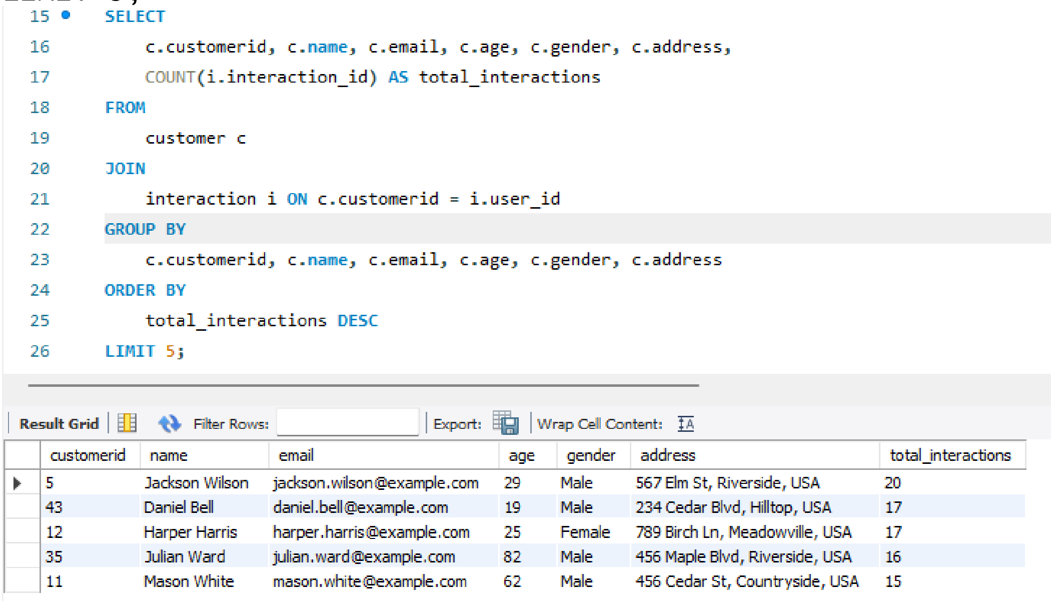
```
SELECT
    c.customerid, c.name, c.email, c.age, c.gender, c.address,
    SUM(o.price) AS total_purchases
FROM
    customer c
JOIN
    orders o ON c.customerid = o.user_id
GROUP BY
    c.customerid, c.name, c.email, c.age, c.gender, c.address
ORDER BY
    total_purchases DESC
LIMIT 5;
```

customerid	name	email	age	gender	address	total_purchases
32	Addison Hill	addison.hill@example.com	91	Female	567 Pine St, Lakeshore, USA	1218
29	Benjamin Lee	benjamin.lee@example.com	24	Male	456 Cedar Ln, Mountainview, USA	1135
37	Michael Taylor	michael.taylor@example.com	79	Male	234 Oak Ln, Highland, USA	973
13	Ethan Martin	ethan.martin@example.com	55	Male	234 Maple Rd, Brookside, USA	890
49	Caleb Peterson	caleb.peterson@example.com	96	Male	234 Oak Dr, Valleytown, USA	848

2. **Details of the top 5 customers with maximum interactions:** This query will help us identifying top 5 customers, in terms of maximum interactions. This query joins customer table with the interaction table and counts the maximum interactions for each customer.

```
SELECT c.customerid, c.name, c.email, c.age, c.gender,
c.address,
COUNT(i.interaction_id) AS total_interactions
```

```
FROM customer c
JOIN interaction i ON c.customerid = i.user_id
GROUP BY c.customerid, c.name, c.email, c.age, c.gender,
c.address
ORDER BY total_interactions DESC
LIMIT 5;
```

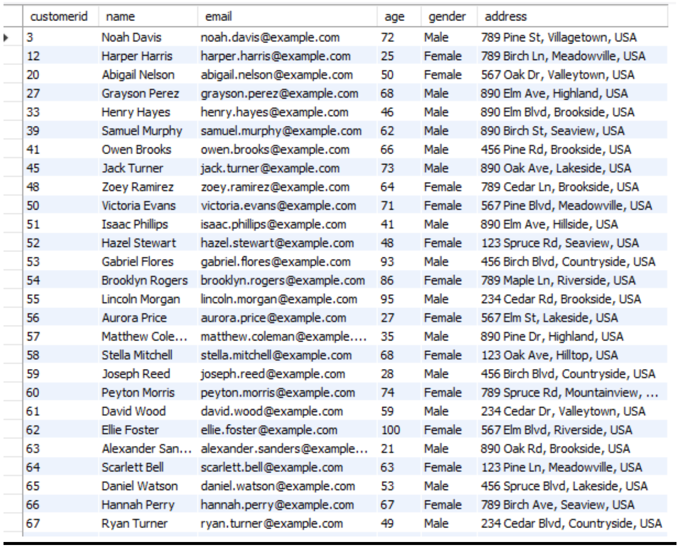


```
SELECT
    c.customerid, c.name, c.email, c.age, c.gender, c.address,
    COUNT(i.interaction_id) AS total_interactions
FROM
    customer c
JOIN
    interaction i ON c.customerid = i.user_id
GROUP BY
    c.customerid, c.name, c.email, c.age, c.gender, c.address
ORDER BY
    total_interactions DESC
LIMIT 5;
```

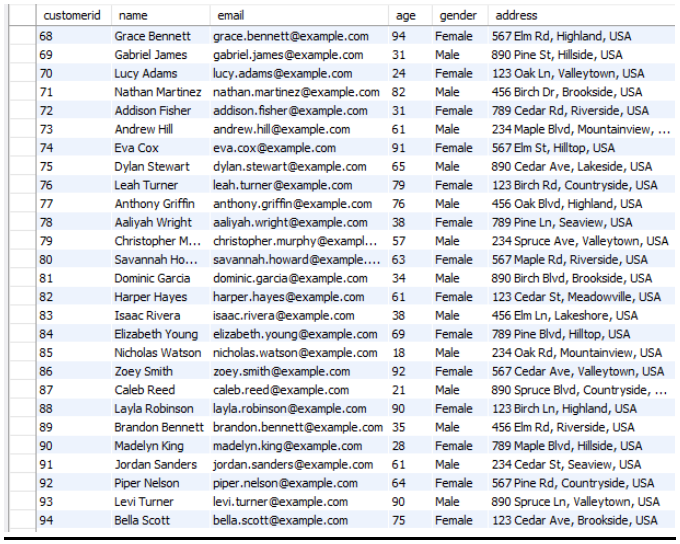
customerid	name	email	age	gender	address	total_interactions
5	Jackson Wilson	jackson.wilson@example.com	29	Male	567 Elm St, Riverside, USA	20
43	Daniel Bell	daniel.bell@example.com	19	Male	234 Cedar Blvd, Hilltop, USA	17
12	Harper Harris	harper.harris@example.com	25	Female	789 Birch Ln, Meadowville, USA	17
35	Julian Ward	julian.ward@example.com	82	Male	456 Maple Blvd, Riverside, USA	16
11	Mason White	mason.white@example.com	62	Male	456 Cedar St, Countryside, USA	15

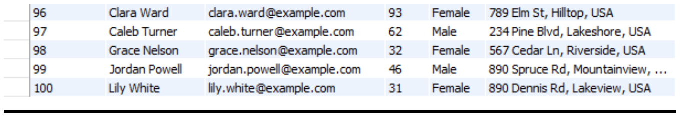
3. **Checking if any customer exists without any purchase:** This query will check for customer without any purchase. The main aim of this query is to check whether there is any customer without any purchase.

```
SELECT c.customerid, c.name, c.email, c.age, c.gender,
c.address
FROM customer c
LEFT JOIN orders o ON c.customerid = o.user_id
WHERE o.order_id IS NULL;
```



customerid	name	email	age	gender	address
3	Noah Davis	noah.davis@example.com	72	Male	789 Pine St, Villagetown, USA
12	Harper Harris	harper.harris@example.com	25	Female	789 Birch Ln, Meadowville, USA
20	Abigail Nelson	abigail.nelson@example.com	50	Female	567 Oak Dr, Valleytown, USA
27	Grayson Perez	grayson.perez@example.com	68	Male	890 Elm Ave, Highland, USA
33	Henry Hayes	henry.hayes@example.com	46	Male	890 Elm Blvd, Brookside, USA
39	Samuel Murphy	samuel.murphy@example.com	62	Male	890 Birch St, Seaview, USA
41	Owen Brooks	owen.brooks@example.com	66	Male	456 Pine Rd, Brookside, USA
45	Jack Turner	jack.turner@example.com	73	Male	890 Oak Ave, Lakeside, USA
48	Zoey Ramirez	zoey.ramirez@example.com	64	Female	789 Cedar Ln, Brookside, USA
50	Victoria Evans	victoria.evans@example.com	71	Female	567 Pine Blvd, Meadowville, USA
51	Isaac Phillips	isaac.phillips@example.com	41	Male	890 Elm Ave, Hillside, USA
52	Hazel Stewart	hazel.stewart@example.com	48	Female	123 Spruce Rd, Seaview, USA
53	Gabriel Flores	gabriel.flores@example.com	93	Male	456 Birch Blvd, Countryside, USA
54	Brooklyn Rogers	brooklyn.rogers@example.com	86	Female	789 Maple Ln, Riverside, USA
55	Lincoln Morgan	lincoln.morgan@example.com	95	Male	234 Cedar Rd, Brookside, USA
56	Aurora Price	aurora.price@example.com	27	Female	567 Elm St, Lakeside, USA
57	Matthew Cole...	matthew.coleman@example....	35	Male	890 Pine Dr, Highland, USA
58	Stella Mitchell	stella.mitchell@example.com	68	Female	123 Oak Ave, Hilltop, USA
59	Joseph Reed	joseph.reed@example.com	28	Male	456 Birch Blvd, Countryside, USA
60	Peyton Morris	peyton.morris@example.com	74	Female	789 Spruce Rd, Mountainview, ...
61	David Wood	david.wood@example.com	59	Male	234 Cedar Dr, Valleytown, USA
62	Ellie Foster	ellie.foster@example.com	100	Female	567 Elm Blvd, Riverside, USA
63	Alexander San...	alexander.sanders@example...	21	Male	890 Oak Rd, Brookside, USA
64	Scarlett Bell	scarlett.bell@example.com	63	Female	123 Pine Ln, Meadowville, USA
65	Daniel Watson	daniel.watson@example.com	53	Male	456 Spruce Blvd, Lakeside, USA
66	Hannah Perry	hannah.perry@example.com	67	Female	789 Birch Ave, Seaview, USA
67	Ryan Turner	ryan.turner@example.com	49	Male	234 Cedar Blvd, Countryside, USA

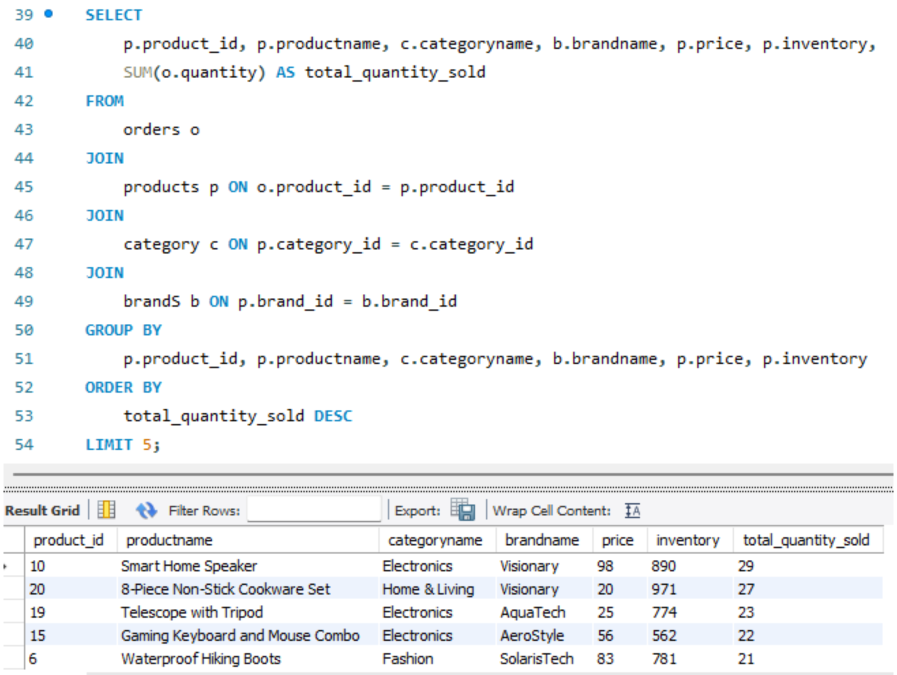




customerid	name	email	age	gender	address
68	Grace Bennett	grace.bennett@example.com	94	Female	567 Elm Rd, Highland, USA
69	Gabriel James	gabriel.james@example.com	31	Male	890 Pine St, Hillside, USA
70	Lucy Adams	lucy.adams@example.com	24	Female	123 Oak Ln, Valleytown, USA
71	Nathan Martinez	nathan.martinez@example.com	82	Male	456 Birch Dr, Brookside, USA
72	Addison Fisher	addison.fisher@example.com	31	Female	789 Cedar Rd, Riverside, USA
73	Andrew Hill	andrew.hill@example.com	61	Male	234 Maple Blvd, Mountainview, ...
74	Eva Cox	eva.cox@example.com	91	Female	567 Elm St, Hilltop, USA
75	Dylan Stewart	dylan.stewart@example.com	65	Male	890 Cedar Ave, Lakeside, USA
76	Leah Turner	leah.turner@example.com	79	Female	123 Birch Rd, Countryside, USA
77	Anthony Griffin	anthony.griffin@example.com	76	Male	456 Oak Blvd, Highland, USA
78	Aaliyah Wright	aaliyah.wright@example.com	38	Female	789 Pine Ln, Seaview, USA
79	Christopher M...	christopher.murphy@exampl...	57	Male	234 Spruce Ave, Valleytown, USA
80	Savannah Ho...	savannah.howard@example....	63	Female	567 Maple Rd, Riverside, USA
81	Dominic Garcia	dominic.garcia@example.com	34	Male	890 Birch Blvd, Brookside, USA
82	Harper Hayes	harper.hayes@example.com	61	Female	123 Cedar St, Meadowville, USA
83	Isaac Rivera	isaac.rivera@example.com	38	Male	456 Elm Ln, Lakeshore, USA
84	Elizabeth Young	elizabeth.young@example.com	69	Female	789 Pine Blvd, Hilltop, USA
85	Nicholas Watson	nicholas.watson@example.com	18	Male	234 Oak Rd, Mountainview, USA
86	Zoey Smith	zoey.smith@example.com	92	Female	567 Cedar Ave, Valleytown, USA
87	Caleb Reed	caleb.reed@example.com	21	Male	890 Spruce Blvd, Countryside, ...
88	Layla Robinson	layla.robinson@example.com	90	Female	123 Birch Ln, Highland, USA
89	Brandon Bennett	brandon.bennett@example.com	35	Male	456 Elm Rd, Riverside, USA
90	Madelyn King	madelyn.king@example.com	28	Female	789 Maple Blvd, Hillside, USA
91	Jordan Sanders	jordan.sanders@example.com	61	Male	234 Cedar St, Seaview, USA
92	Piper Nelson	piper.nelson@example.com	64	Female	567 Pine Rd, Countryside, USA
93	Levi Turner	levi.turner@example.com	90	Male	890 Spruce Ln, Valleytown, USA
94	Bella Scott	bella.scott@example.com	75	Female	123 Cedar Ave, Brookside, USA
96	Clara Ward	clara.ward@example.com	93	Female	789 Elm St, Hilltop, USA
97	Caleb Turner	caleb.turner@example.com	62	Male	234 Pine Blvd, Lakeshore, USA
98	Grace Nelson	grace.nelson@example.com	32	Female	567 Cedar Ln, Riverside, USA
99	Jordan Powell	jordan.powell@example.com	46	Male	890 Spruce Rd, Mountainview, ...
100	Lily White	lily.white@example.com	31	Female	890 Dennis Rd, Lakeview, USA

4. **Finding the most popular product:** This query will check for the most popular products by considering the product that has been purchased more.

```
SELECT p.product_id, p.productname, c.categoryname,
b.brandsname, p.price, p.inventory, SUM(o.quantity) AS
total_quantity_sold FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN category c ON p.category_id = c.category_id
JOIN brands b ON p.brands_id = b.brands_id
GROUP BY p.product_id, p.productname, c.categoryname,
b.brandsname, p.price, p.inventory
ORDER BY total_quantity_sold DESC
LIMIT 5;
```

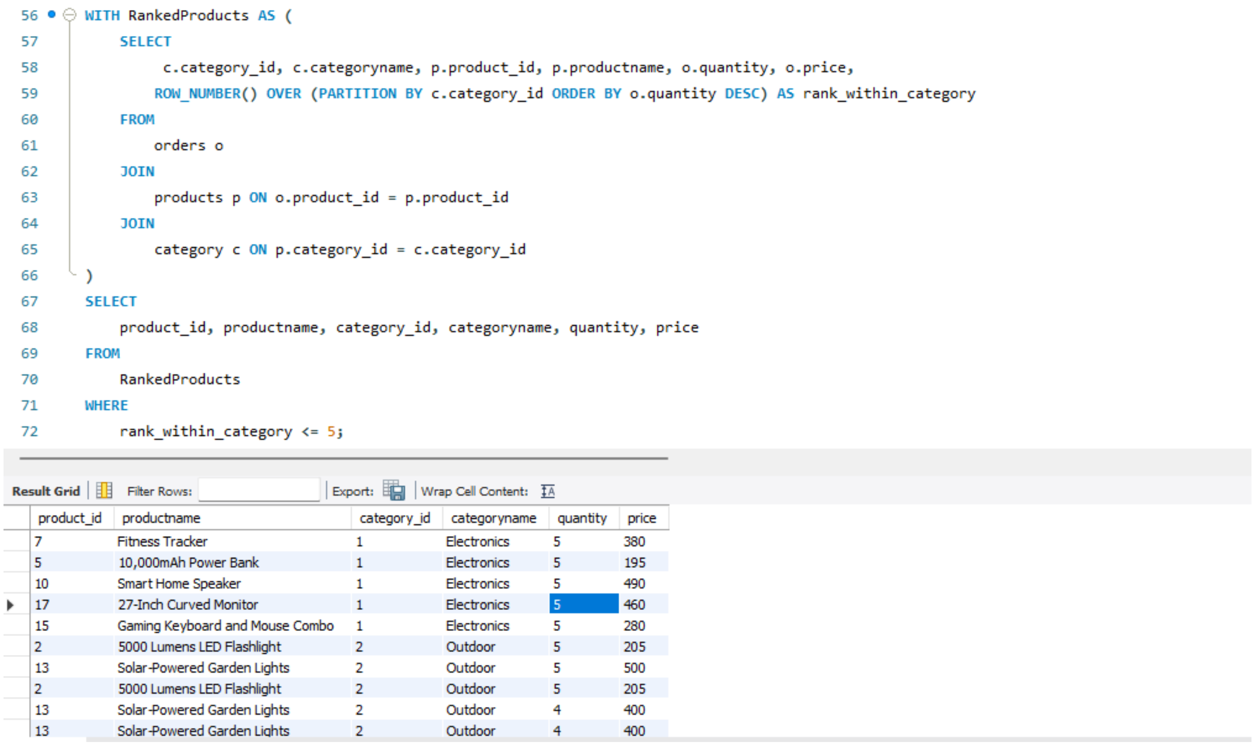


```
SELECT
    p.product_id, p.productname, c.categoryname, b.brandname, p.price, p.inventory,
    SUM(o.quantity) AS total_quantity_sold
FROM
    orders o
JOIN
    products p ON o.product_id = p.product_id
JOIN
    category c ON p.category_id = c.category_id
JOIN
    brandS b ON p.brand_id = b.brand_id
GROUP BY
    p.product_id, p.productname, c.categoryname, b.brandname, p.price, p.inventory
ORDER BY
    total_quantity_sold DESC
LIMIT 5;
```

product_id	productname	categoryname	brandname	price	inventory	total_quantity_sold
10	Smart Home Speaker	Electronics	Visionary	98	890	29
20	8-Piece Non-Stick Cookware Set	Home & Living	Visionary	20	971	27
19	Telescope with Tripod	Electronics	AquaTech	25	774	23
15	Gaming Keyboard and Mouse Combo	Electronics	AeroStyle	56	562	22
6	Waterproof Hiking Boots	Fashion	SolarisTech	83	781	21

5. **Based on each category top 5 sales:** In this query, our goal is to identify the products sold the most across each category.

```
WITH RankedProducts AS (
SELECT c.category_id, c.categoryname, p.product_id,
p.productname, o.quantity, o.price, ROW_NUMBER() OVER
(PARTITION BY c.category_id ORDER BY o.quantity DESC) AS
rank_within_category
FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN category c ON p.category_id = c.category_id
)
SELECT product_id, productname, category_id, categoryname,
quantity, price
FROM RankedProducts
WHERE rank_within_category <= 5;
```

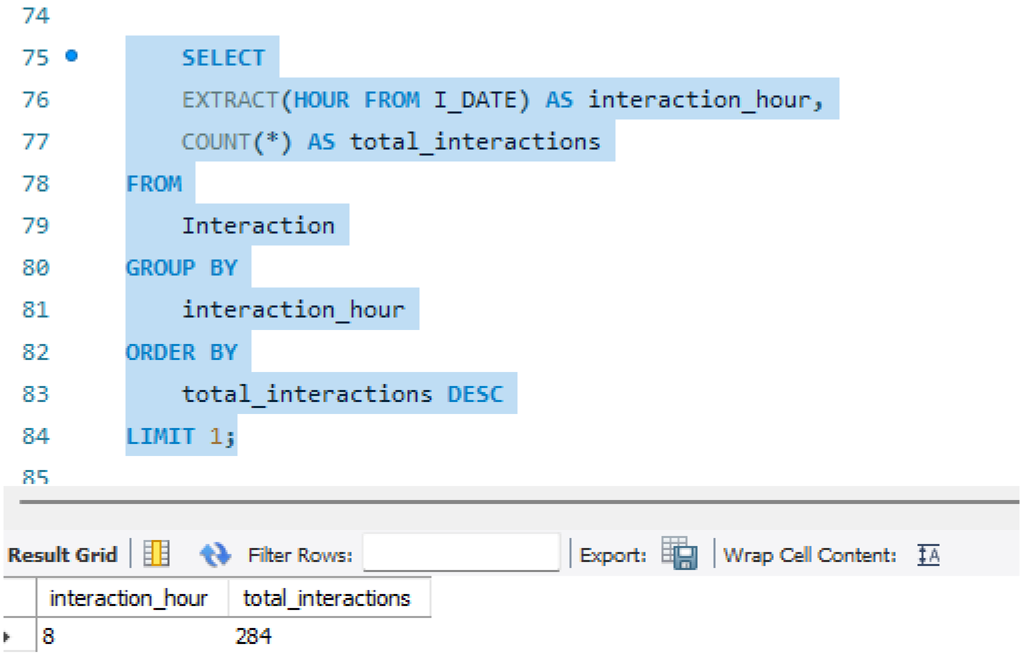


```
WITH RankedProducts AS (
    SELECT
         c.category_id, c.categoryname, p.product_id, p.productname, o.quantity, o.price,
        ROW_NUMBER() OVER (PARTITION BY c.category_id ORDER BY o.quantity DESC) AS rank_within_category
    FROM
        orders o
    JOIN
        products p ON o.product_id = p.product_id
    JOIN
        category c ON p.category_id = c.category_id
)
SELECT
    product_id, productname, category_id, categoryname, quantity, price
FROM
    RankedProducts
WHERE
    rank_within_category <= 5;
```

product_id	productname	category_id	categoryname	quantity	price
7	Fitness Tracker	1	Electronics	5	380
5	10,000mAh Power Bank	1	Electronics	5	195
10	Smart Home Speaker	1	Electronics	5	490
17	27-Inch Curved Monitor	1	Electronics	5	460
15	Gaming Keyboard and Mouse Combo	1	Electronics	5	280
2	5000 Lumens LED Flashlight	2	Outdoor	5	205
13	Solar-Powered Garden Lights	2	Outdoor	5	500
2	5000 Lumens LED Flashlight	2	Outdoor	5	205
13	Solar-Powered Garden Lights	2	Outdoor	4	400
13	Solar-Powered Garden Lights	2	Outdoor	4	400

6. **Identifying the hour at which maximum interactions were done:** The goal of this query is to identify the hour at which users interacted with the e-commerce platform the most. It will help in understanding the user traffic.

```
SELECT EXTRACT(HOUR FROM i_date) AS interaction_hour,COUNT(*)
AS total_interactions FROM Interaction
GROUP BY interaction_hour
ORDER BY total_interactions DESC
LIMIT 1;
```

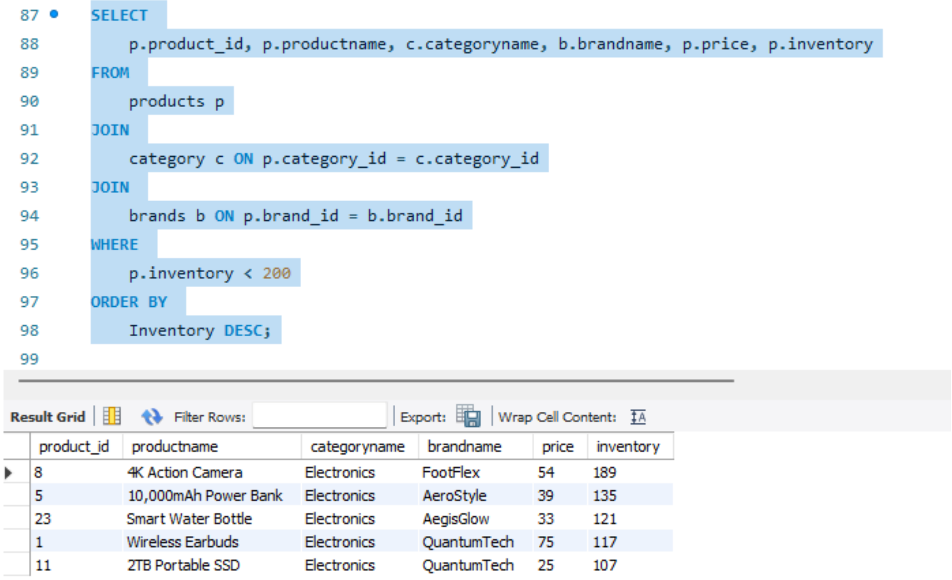


```
    SELECT
        EXTRACT(HOUR FROM I_DATE) AS interaction_hour,
        COUNT(*) AS total_interactions
    FROM
        Interaction
    GROUP BY
        interaction_hour
    ORDER BY
        total_interactions DESC
    LIMIT 1;
```

interaction_hour	total_interactions
8	284

7. **Identifying products with inventory size less than 200:** This query will identify products that have less than 200 units in the inventory. This output will help in planning the inventory management for products.

```
SELECT p.product_id, p.productname, c.categoryname,
b.brandsname, p.price, p.inventory
FROM products p
JOIN category c ON p.category_id = c.category_id
JOIN brands b ON p.brands_id = b.brands_id
WHERE p.inventory < 200
ORDER BY Inventory DESC;
```



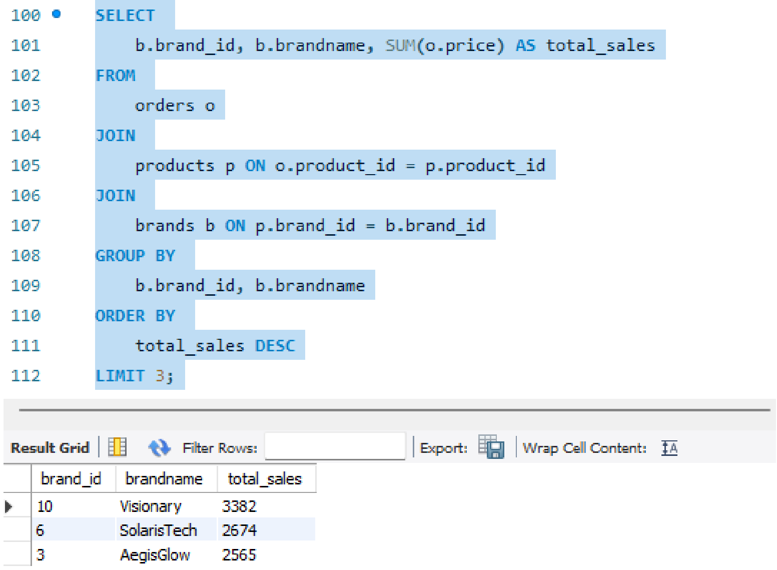
```
SELECT
    p.product_id, p.productname, c.categoryname, b.brandname, p.price, p.inventory
FROM
    products p
JOIN
    category c ON p.category_id = c.category_id
JOIN
    brands b ON p.brand_id = b.brand_id
WHERE
    p.inventory < 200
ORDER BY
    Inventory DESC;
```

product_id	productname	categoryname	brandname	price	inventory
8	4K Action Camera	Electronics	FootFlex	54	189
5	10,000mAh Power Bank	Electronics	AeroStyle	39	135
23	Smart Water Bottle	Electronics	AegisGlow	33	121
1	Wireless Earbuds	Electronics	QuantumTech	75	117
11	2TB Portable SSD	Electronics	QuantumTech	25	107

8. **Identifying top 3 brands based on sales:** This query will help identify the top 3 brands based on their sales. It will help in determining the top performing brands.

```
SELECT b.brands_id, b.brandsname, SUM(o.price) AS total_sales
FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN brands b ON p.brands_id = b.brands_id
GROUP BY b.brands_id, b.brandsname
ORDER BY total_sales DESC
```

```
LIMIT 3;
```



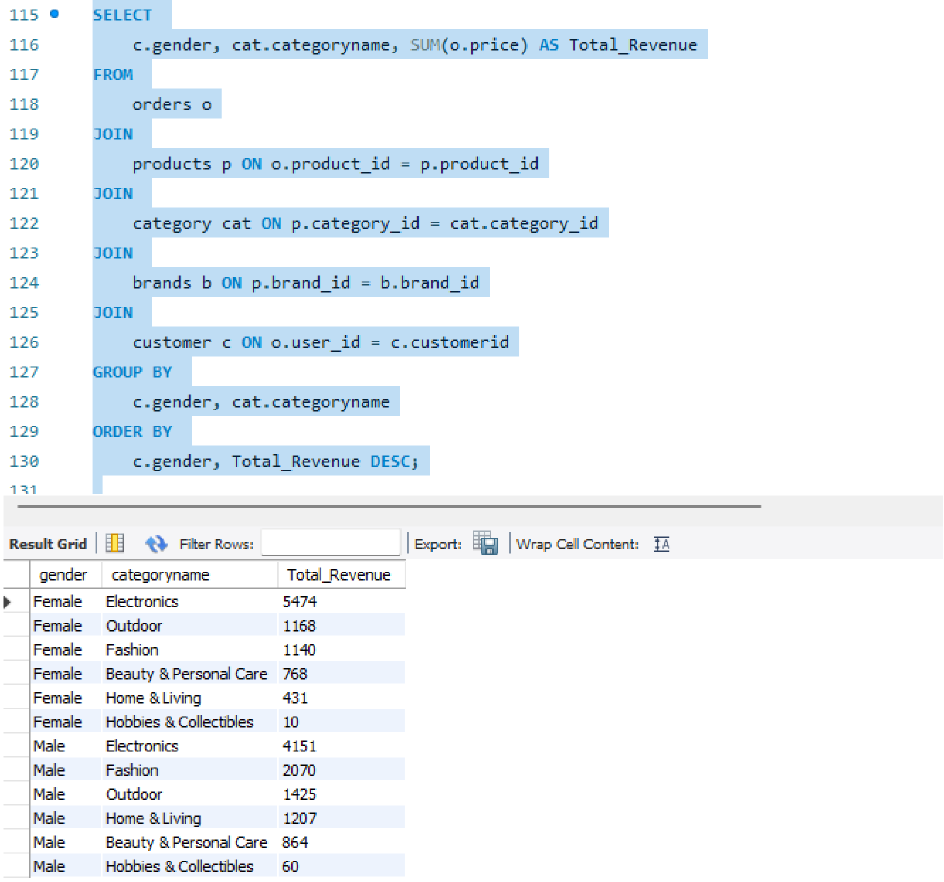
```
SELECT
    b.brand_id, b.brandname, SUM(o.price) AS total_sales
FROM
    orders o
JOIN
    products p ON o.product_id = p.product_id
JOIN
    brands b ON p.brand_id = b.brand_id
GROUP BY
    b.brand_id, b.brandname
ORDER BY
    total_sales DESC
LIMIT 3;
```

```
brand_id  brandname   total_sales
10        Visionary   3382
6         SolarisTech 2674
3         AegisGlow   2565
```

9. **Revenue per gender per category:** This shows the revenue that is coming from each gender, based on categories. This query will help in understanding user behaviour, based on their gender.

```
SELECT c.gender, cat.categoryname, SUM(o.price) AS
Total_Revenue
FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN category cat ON p.category_id = cat.category_id
JOIN brands b ON p.brands_id = b.brands_id
JOIN customer c ON o.user_id = c.customerid
GROUP BY c.gender, cat.categoryname
```

```
ORDER BY c.gender, Total_Revenue DESC;
```



```sql
SELECT
    c.gender, cat.categoryname, SUM(o.price) AS Total_Revenue
FROM
    orders o
JOIN
    products p ON o.product_id = p.product_id
JOIN
    category cat ON p.category_id = cat.category_id
JOIN
    brands b ON p.brand_id = b.brand_id
JOIN
    customer c ON o.user_id = c.customerid
GROUP BY
    c.gender, cat.categoryname
ORDER BY
    c.gender, Total_Revenue DESC;
```

gender	categoryname	Total_Revenue
Female	Electronics	5474
Female	Outdoor	1168
Female	Fashion	1140
Female	Beauty & Personal Care	768
Female	Home & Living	431
Female	Hobbies & Collectibles	10
Male	Electronics	4151
Male	Fashion	2070
Male	Outdoor	1425
Male	Home & Living	1207
Male	Beauty & Personal Care	864
Male	Hobbies & Collectibles	60

4. Conclusion

In this report, we have explained about how we have designed and developed a database system, how we have implemented the design using functions, stored procedures and triggers that will help in the functioning of the database in an E-Commerce Recommendation platform. We have also created some SQL queries, which can be used for data analysis which in turn can be used to make important business decisions.