E-commerce Platform system

Business Requirements Specifications (BRS)for E-commerce Platform System

Introduction

This section outlines the business requirements for the development and implementation of an e-commerce platform. These requirements are essential for ensuring the platform meets the business objectives and delivers value to all stakeholders.

System Scope

- **Product Management:** Adding, editing, deleting, and categorizing products; managing product inventory and stock levels; setting product prices and descriptions.
- **Customer Management:** User registration, login, profile management, order history, and customer support.
- **Order Processing:** Adding items to cart, checkout, order confirmation, payment processing, and shipping.
- **Inventory Management:** Tracking stock levels, generating reorder alerts, and managing inventory across warehouses.
- **Delivery Tracking:** Order tracking, delivery status updates, and notifications to customers.
- **Search and Navigation:** Product search and filtering, category browsing, and easy navigation within the website.
- **Payment Gateway Integration:** Integration with various payment gateways (e.g., credit cards, PayPal, etc.).
- **Shipping Integration:** Integration with shipping providers for order fulfillment and delivery tracking.
- **User Interface (UI) and User Experience (UX):** Designing and developing a user-friendly and intuitive interface for customers and administrators.

System Objectives

- Provide a seamless and user-friendly online shopping experience for customers.
- Enable efficient order processing and fulfillment for the business.
- Increase online sales revenue and expand market reach.
- Improve customer satisfaction and loyalty.
- Gain valuable insights into customer behavior and market trends.
- Ensure the security and privacy of customer data and financial transactions.
- Maintain a high level of system availability and performance.
- Provide a scalable platform that can accommodate future growth and business expansion.

System Requirements Specifications (SRS)for E-commerce Platform system

Introduction

A System Requirements Specification outlines the complete description of a software system's functional and non-functional requirements. It serves as a guide for system developers and stakeholders to understand the features and constraints of the E-commerce system.

Functional Requirements:

1- **Product Management**

- Add, edit, and delete products.
- Categorize products and manage its' inventory and stock levels for easy browsing.

2- **Customer Management**

- User registration with personal details and shipping information.
- Provide functionality for customers to log in, update profiles, and view order history.

3- **Order Processing**

- Enable customers to browse products and add items to their cart and place orders.
- Calculate shipping costs based on customer location and delivery speed.
- Support multiple payment methods (credit card, PayPal, etc.).

4- **inventory Management**

- Automatically update stock levels when orders are placed.
- Generate reports for product stock levels and reorder alerts.

5- **Delivery Tracking**

- Assign orders to shipping providers and track delivery status.
- Notify customers of delivery progress via email or SMS.

Non-Functional Requirements:

1-Performance

- Ensure Fast page load times.
- The system should support concurrent user interactions efficiently.

2- **Security**

- Use Encryption for all sensitive data, including customer details and payment information.
- Implement strong authentication and authentication.

3- **Usability**

- Provide user-friendly interface
- Support mobile and desktop access with responsive design principles.
- User feedback mechanisms should be in place for continuous improvement.

4- **Scalability**

- Ability to handle increased traffic and data volume.

5- **Backup and Recovery**

- Perform daily backups of all transactional and customer data.
- Implement a recovery plan to restore data any failure.

6- **Availability**

- Ensure high system availability with minimal downtime through a disaster recovery plan and load balancing to handle server failures seamlessly.

Stakeholder Requirements Specifications (SRS)for E-commerce Platform system

Introduction

This section outlines the specific requirements of each key stakeholder involved in the development and utilization of the e-commerce platform.

1- **Customers:**

- Easy product browsing and search.
- Ability to add items to the cart and checkout securely.
- Multiple payment options (credit card, PayPal, etc.) with high level of security for personal and payment information.
- Real-time order tracking and delivery updates.
- Ability to view order history and manage account details.
- User-friendly and intuitive interface.

2- **System Administrators:**

- Manage product catalog (add, edit, delete).
- Define product categories and subcategories.
- Monitor inventory levels and generate reorder alerts.
- Process orders and manage customer accounts.
- Configure shipping options and integrate with shipping providers.
- Access control and role-based permissions.

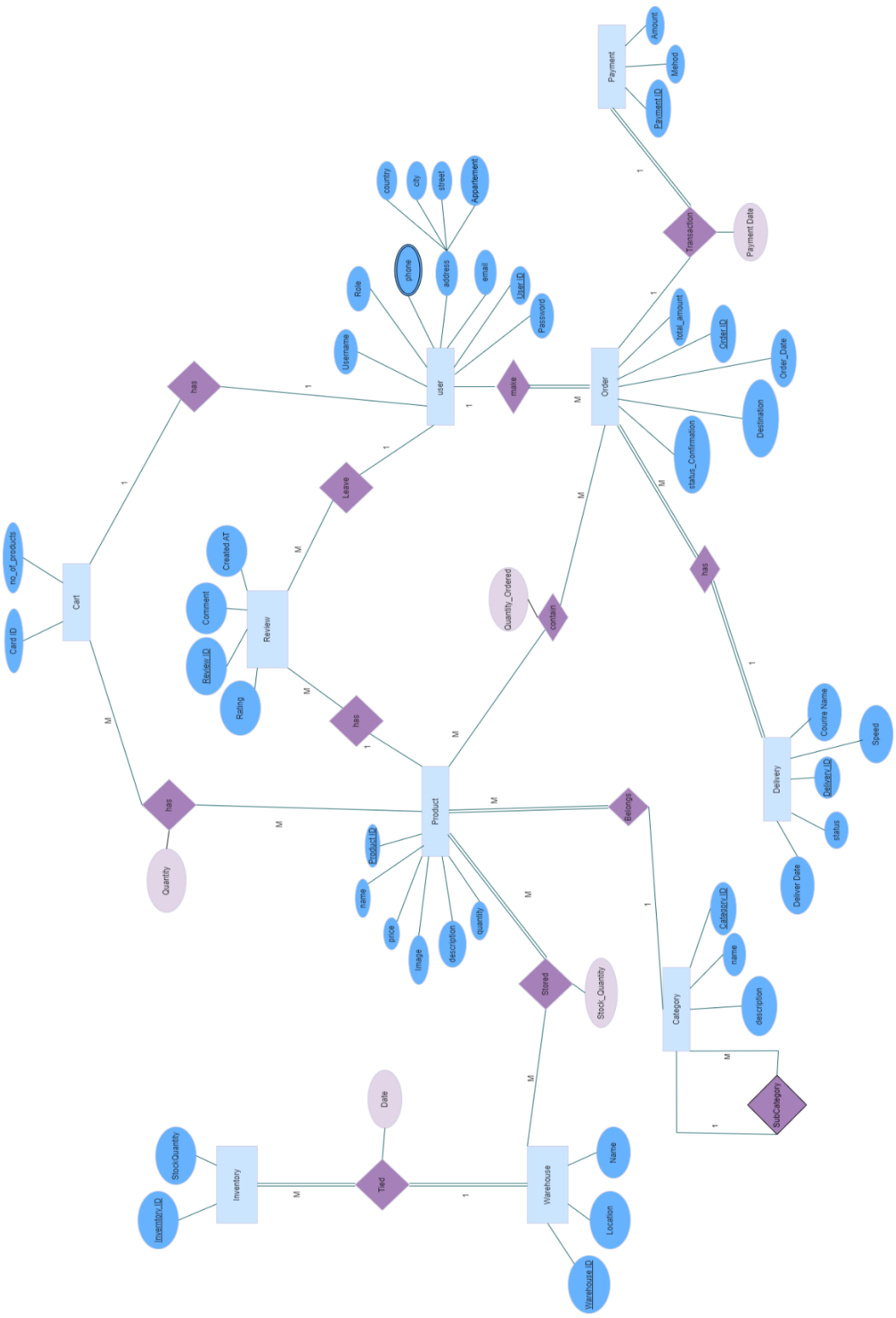
3- **Business Owners:**

- Increased revenue through online sales.
- Enhanced market reach with a structured product catalog and promotional capabilities.
- Operational efficiency via automated processes for inventory and order management.
- Actionable insights through sales and inventory reports.

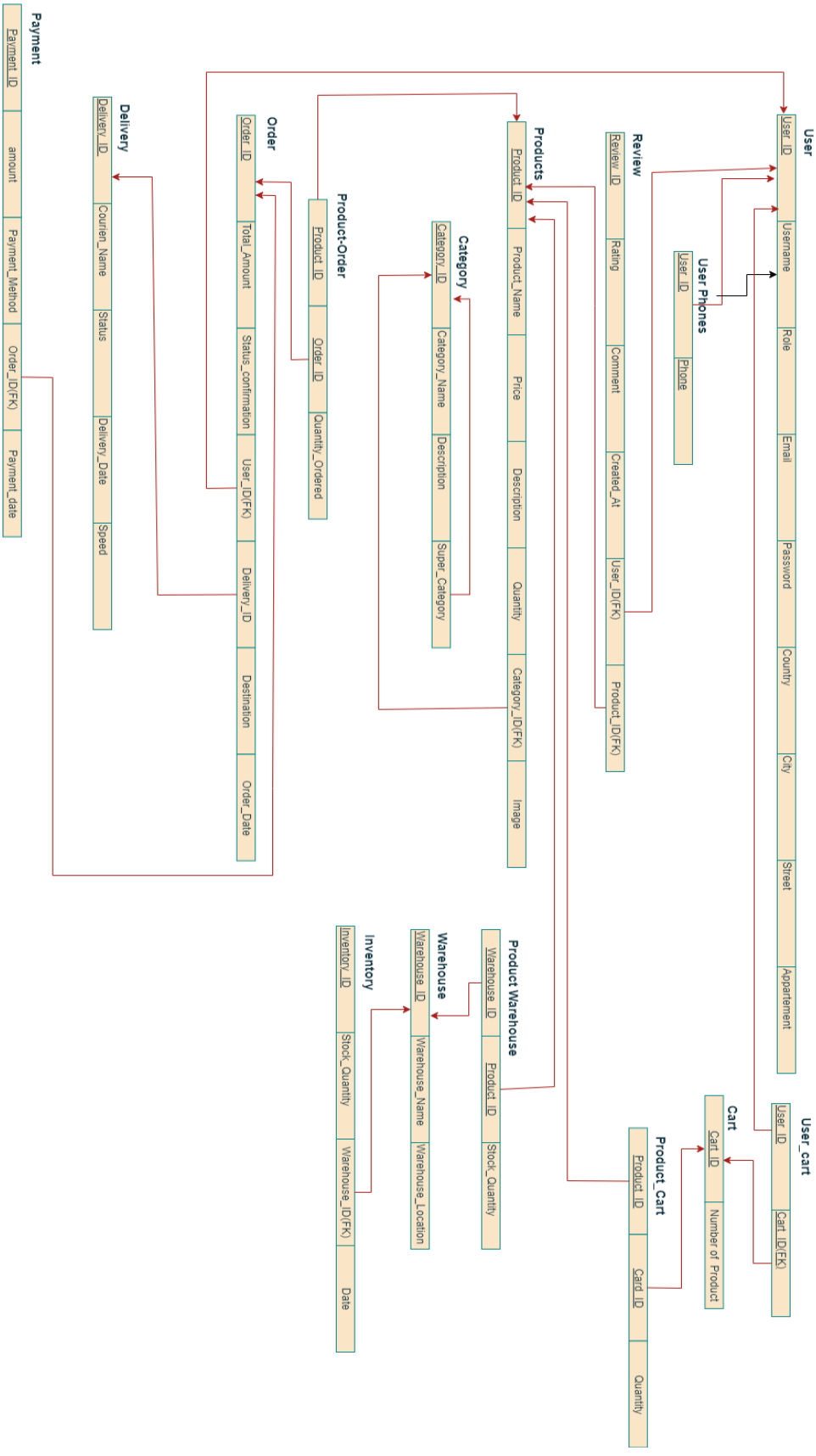
4- **Delivery Providers:**

- Efficient assignment of orders for delivery.
- Real-time updates on delivery status.
- Integration for seamless communication with the system.

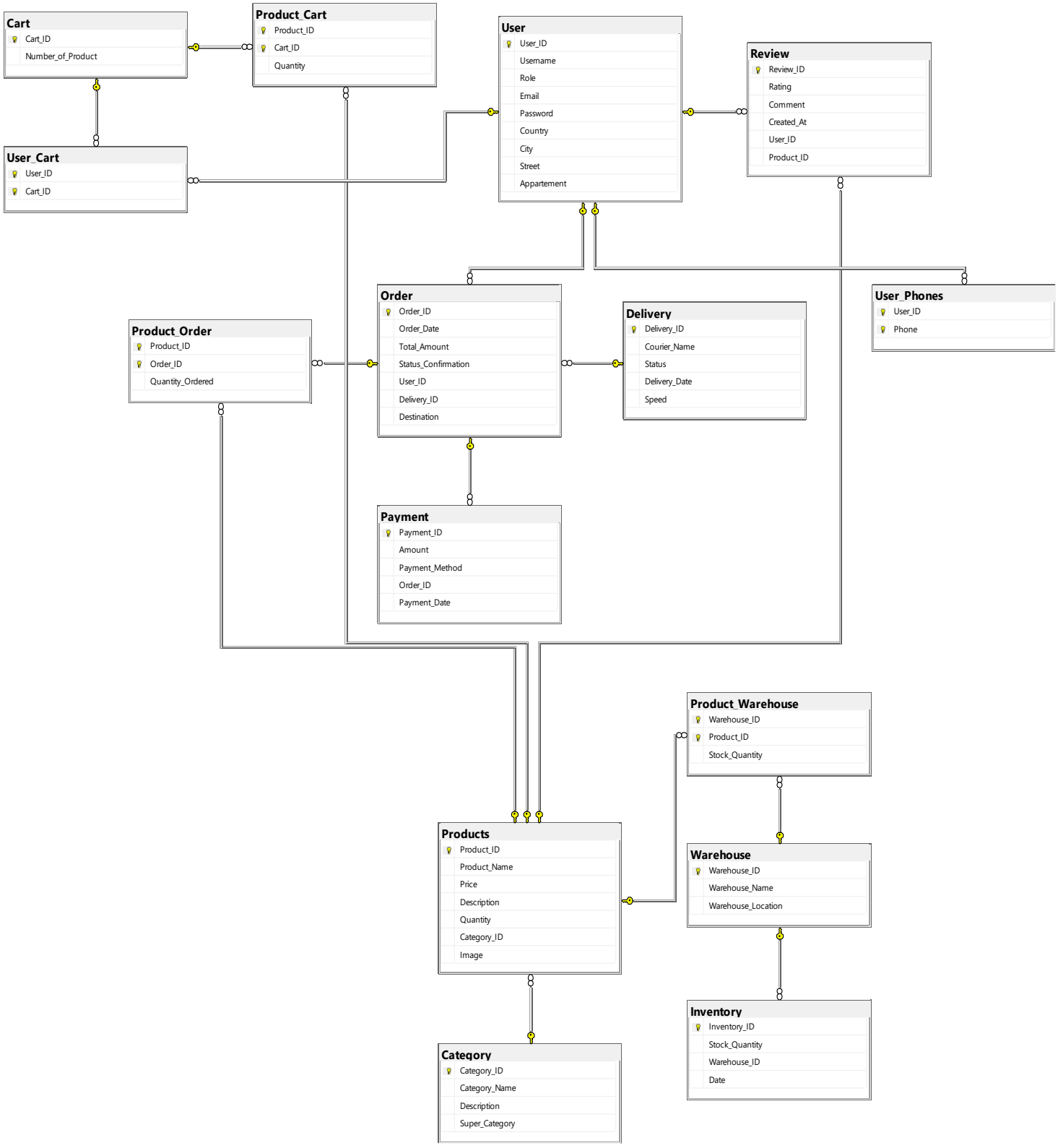
ER Diagram for E-commerce platform system



Mapping for ER Diagram for E-commerce platform system



Schema for E-commerce platform system



SQL for E-commerce platform system

Gamila.ECommerce...- Diagram_Schema | SQLQueryFinal.sql...(GAMILA\mself (52))*

```sql
CREATE DATABASE ECommerce;

USE ECommerce;

-- User Table
CREATE TABLE [User] (
    User_ID INT PRIMARY KEY IDENTITY(1,1),
    Username VARCHAR(50) NOT NULL,
    Role NVARCHAR(20) DEFAULT('Customer'),
    Email VARCHAR(100) UNIQUE NOT NULL,
    Password VARCHAR(16) NOT NULL,
    Country VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    Street VARCHAR(100) NOT NULL,
    Appartement VARCHAR(50) NOT NULL
);

-- User_Phones Table
CREATE TABLE User_Phones (
    User_ID INT NOT NULL,
    Phone VARCHAR(15) NOT NULL,
    PRIMARY KEY (User_ID, Phone),
    FOREIGN KEY (User_ID) REFERENCES [User](User_ID)
);

-- Cart Table
CREATE TABLE Cart (
    Cart_ID INT PRIMARY KEY IDENTITY(1,1),
    Number_of_Product INT NOT NULL DEFAULT 0
);

-- User_Cart Table
CREATE TABLE User_Cart (
    User_ID INT NOT NULL,
    Cart_ID INT NOT NULL,
    PRIMARY KEY (User_ID, Cart_ID),
    FOREIGN KEY (User_ID) REFERENCES [User](User_ID),
    FOREIGN KEY (Cart_ID) REFERENCES Cart(Cart_ID)
);

-- Category Table
CREATE TABLE Category (
    Category_ID INT PRIMARY KEY IDENTITY(1,1),
    Category_Name VARCHAR(50) NOT NULL UNIQUE,
    Description VARCHAR(255),
    Super_Category INT
);
```

100 %

Results | Messages

Gamila.ECommerce...- Diagram_Schema | SQLQueryFinal.sql...(GAMILA\mself (52))*

```sql
-- Products Table
CREATE TABLE Products (
    Product_ID INT PRIMARY KEY IDENTITY(1,1),
    Product_Name VARCHAR(50) NOT NULL,
    Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0),
    Description VARCHAR(255),
    Quantity INT NOT NULL CHECK (Quantity >= 0),
    Category_ID INT,
    Image VARCHAR(255) NOT NULL,
    FOREIGN KEY (Category_ID) REFERENCES Category(Category_ID)
);

-- Product_Cart Table
CREATE TABLE Product_Cart (
    Product_ID INT NOT NULL,
    Cart_ID INT NOT NULL,
    Quantity INT NOT NULL CHECK (Quantity >= 1) DEFAULT 1,
    PRIMARY KEY (Product_ID, Cart_ID),
    FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID),
    FOREIGN KEY (Cart_ID) REFERENCES Cart(Cart_ID)
);


-- Delivery Table
CREATE TABLE Delivery (
    Delivery_ID INT PRIMARY KEY IDENTITY(1,1),
    Courier_Name VARCHAR(50) NOT NULL,
    Status VARCHAR(20) DEFAULT 'In Transit',
    Delivery_Date DATETIME,
    Speed VARCHAR(20) DEFAULT 'Normal'
);


-- Order Table
CREATE TABLE [Order] (
    Order_ID INT PRIMARY KEY IDENTITY(1,1),
    Order_Date DATETIME NOT NULL DEFAULT GETDATE(),
    Total_Amount DECIMAL(12, 2) NOT NULL CHECK (Total_Amount >= 0),
    Status_Confirmation VARCHAR(20) NOT NULL DEFAULT 'Pending',
    User_ID INT NOT NULL,
    Delivery_ID INT,
    Destination VARCHAR(255),
    FOREIGN KEY (User_ID) REFERENCES [User](User_ID),
    FOREIGN KEY (Delivery_ID) REFERENCES Delivery(Delivery_ID)
);

-- Product Order Table
```

Gamila.ECommerce...- Diagram_Schema | SQLQueryFinal.sql...(GAMILA\mself (52))*

```sql
-- Product_Order Table
CREATE TABLE Product_Order (
    Product_ID INT NOT NULL,
    Order_ID INT NOT NULL,
    Quantity_Ordered INT NOT NULL CHECK (Quantity_Ordered > 0),
    PRIMARY KEY (Product_ID, Order_ID),
    FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID),
    FOREIGN KEY (Order_ID) REFERENCES [Order](Order_ID)
);

-- Review Table
CREATE TABLE Review (
    Review_ID INT PRIMARY KEY IDENTITY(1,1),
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    Comment VARCHAR(255),
    Created_At DATETIME DEFAULT GETDATE(),
    User_ID INT NOT NULL,
    Product_ID INT NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES [User](User_ID),
    FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID)
);



-- Payment Table
CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY IDENTITY(1,1),
    Amount DECIMAL(12, 2) NOT NULL CHECK (Amount > 0),
    Payment_Method VARCHAR(50) NOT NULL,
    Order_ID INT NOT NULL,
    Payment_Date DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (Order_ID) REFERENCES [Order](Order_ID)
);

-- Warehouse Table
CREATE TABLE Warehouse (
    Warehouse_ID INT PRIMARY KEY IDENTITY(1,1),
    Warehouse_Name VARCHAR(50) NOT NULL,
    Warehouse_Location VARCHAR(100) NOT NULL
);

-- Product_Warehouse Table
CREATE TABLE Product_Warehouse (
    Warehouse_ID INT NOT NULL,
    Product_ID INT NOT NULL,
    Stock Quantity INT NOT NULL CHECK (Stock Quantity >= 0),
```

100 %

Results | Messages

Product_Name	Total_Quantity

```sql
    Warehouse_Location VARCHAR(100) NOT NULL
);

-- Product_Warehouse Table
CREATE TABLE Product_Warehouse (
    Warehouse_ID INT NOT NULL,
    Product_ID INT NOT NULL,
    Stock_Quantity INT NOT NULL CHECK (Stock_Quantity >= 0),
    PRIMARY KEY (Warehouse_ID, Product_ID),
    FOREIGN KEY (Warehouse_ID) REFERENCES Warehouse(Warehouse_ID),
    FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID)
);

-- Inventory Table
CREATE TABLE Inventory (
    Inventory_ID INT PRIMARY KEY IDENTITY(1,1),
    Stock_Quantity INT NOT NULL CHECK (Stock_Quantity >= 0),
    Warehouse_ID INT NOT NULL,
    Date DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (Warehouse_ID) REFERENCES Warehouse(Warehouse_ID)
);


--insert data in the user table
SET IDENTITY_INSERT [User] ON;

INSERT INTO [User] (User_ID, Username, Role, Email, Password, Country, City, Street, Appartement)
VALUES
(1, 'John Doe', 'Customer', 'john.doe@gmail.com', 'Password123', 'USA', 'New York', '5th Avenue', 'Apt 101'),
(2, 'Jane Smith', 'Customer', 'jane.smith@gmail.com', 'Passw0rd456', 'Canada', 'Toronto', 'Bay Street', 'Suite 202'),
(3, 'Ahmed Ali', 'Customer', 'ahmed.ali@gmail.com', 'Secret789', 'Egypt', 'Cairo', 'Fisal', '205'),
(4, 'Habiba Mohamed', 'Customer', 'habiba.mohamed@gmail.com', 'Habiba123', 'Egypt', 'Alex', 'Somoha', '328'),
(5, 'Mohamed Omar', 'Customer', 'mohamed61485@gmail.com', 'Mohamed456', 'Syria', 'Hama', 'Alnaoora', '186'),
(6, 'Alice Johnson', 'Customer', 'alice.johnson@gmail.com', 'Alice789', 'USA', 'Chicago', 'Lake Shore Drive', 'Apt 502'),
(7, 'Robert Brown', 'Customer', 'robert.brown@gmail.com', 'Robert123', 'UK', 'London', 'Baker Street', 'Flat 21'),
(8, 'Chen Wei', 'Customer', 'chen.wei@gmail.com', 'Chen456', 'China', 'Beijing', 'Chang an Avenue', 'Bldg 3, Apt 14'),
(9, 'Maria Gonzalez', 'Customer', 'maria.gonzalez@gmail.com', 'Maria789', 'Mexico', 'Mexico City', 'Reforma', 'Apt 808'),
(10, 'Ivan Petrov', 'Customer', 'ivan.petrov@gmail.com', 'Ivan123', 'Russia', 'Moscow', 'Tverskaya', 'Flat 12'),
(11, 'Emma Thompson', 'Customer', 'emma.thompson@gmail.com', 'Emma456', 'Australia', 'Sydney', 'George Street', 'Suite 19'),
(12, 'Carlos Martinez', 'Customer', 'carlos.martinez@gmail.com', 'Carlos789', 'Spain', 'Madrid', 'Gran Via', 'Apt 5B'),
(13, 'Sara Ahmed', 'Customer', 'sara.ahmed@gmail.com', 'Sara123', 'Egypt', 'Giza', 'Pyramids Street', 'Flat 76'),
(14, 'Liam Wilson', 'Customer', 'liam.wilson@gmail.com', 'Liam456', 'USA', 'Seattle', 'Pine Street', 'Apt 1201'),
(15, 'Sophia Taylor', 'Customer', 'sophia.taylor@gmail.com', 'Sophia789', 'Canada', 'Vancouver', 'Robson Street', 'Suite 304'),
(16, 'Daniel Wilson', 'Customer', 'daniel.wilson@example.com', 'Daniel123', 'Canada', 'Vancouver', 'Granville St', 'Apt 505'),
(17, 'Sophia Anderson', 'Customer', 'sophia.anderson@example.com', 'Sophia456', 'UK', 'London', 'Baker St', 'Flat 1A'),
(18, 'Liam Thomas', 'Customer', 'liam.thomas@example.com', 'Liam789', 'UK', 'Manchester', 'Deansgate', 'Flat 2B'),
(19, 'Olivia Moore', 'Customer', 'olivia.moore@example.com', 'Olivia123', 'Australia', 'Sydney', 'George St', 'Unit 3C'),
```

Gamila.ECommerce...- Diagram_Schema | SQLQueryFinal.sql...(GAMILA\mself (52))*

```
SELECT  *
from [User];
SELECT  *
from User_Phones;
SELECT  *
from Cart;
SELECT  *
from User_Cart;
SELECT  *
from Category;
```

100 %

Results | Messages

	User_ID	Username	Role	Email	Password	Country	City	Street	Appartement
1	1	John Doe	Customer	john.doe@gmail.com	Password123	USA	New York	5th Avenue	Apt 101
2	2	Jane Smith	Customer	jane.smith@gmail.com	Passw0rd456	Canada	Toronto	Bay Street	Suite 202
3	3	Ahmed Ali	Customer	ahmed.ali@gmail.com	Secret789	Egypt	Cairo	Fisal	205
4	4	Habiba Mohamed	Customer	habiba.mohamed@gmail.com	Habiba123	Egypt	Alex	Somoha	328
5	5	Mohamed Omar	Customer	mohamed61485@gmail.com	Mohamed456	Syria	Hama	Alnaoora	186
6	6	Alice Johnson	Customer	alice.johnson@gmail.com	Alice789	USA	Chicago	Lake Shore Drive	Apt 502
7	7	Robert Brown	Customer	robert.brown@gmail.com	Robert123	UK	London	Baker Street	Flat 21
8	8	Chen Wei	Customer	chen.wei@gmail.com	Chen456	China	Beijing	Chang an Avenue	Bldg 3, Apt 14

	User_ID	Phone
1	1	123-456-7890
2	2	234-567-8901
3	3	345-678-9012
4	4	456-789-0123
5	5	567-890-1234
6	6	678-901-2345
7	7	789-012-3456
8	8	890-123-4567

	Cart_ID	Number_of_Product
1	1	3
2	2	5
3	3	2
4	4	8

	User_ID	Cart_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

	Category_ID	Category_Name	Description	Super_Category
1	1	Electronics	Electronic Devices	NULL
2	2	Clothing	Apparel and Accessories	NULL
3	3	Books	Books and Stationery	NULL
4	4	Furniture	Home and Office Furnit...	NULL
5	5	Toys	Toys and Games	NULL
6	6	Beauty	Beauty and Personal C...	NULL
7	7	Sports	Sports and Fitness	NULL
8	8	Automotive	Automotive Accessories	NULL

Query executed successfully.

GAMIL

Gamila.ECommerce...- Diagram_Schema | SQLQueryFinal.sql...(GAMILA\mself (52))*

```
SELECT *
from Category;
SELECT *
from Products;
SELECT *
from Product_Cart;
SELECT *
from Delivery;
SELECT *
from [Order];
SELECT *
from Product_Order;
```

100 %

Results | Messages

	Product_ID	Product_Name	Price	Description	Quantity	Category_ID	Image
1	1	Laptop	999.99	High-performance laptop	100	1	https://example.com/images/laptop.jpg
2	2	Smartphone	799.99	Latest model smartphone	200	1	https://example.com/images/smartphone.jpg
3	3	T-shirt	19.99	Cotton t-shirt	500	2	https://example.com/images/tshirt.jpg
4	4	Jeans	49.99	Denim jeans	300	2	https://example.com/images/jeans.jpg
5	5	Fiction Book	14.99	Best-selling novel	150	3	https://example.com/images/fiction_book.jpg

	Product_ID	Cart_ID	Quantity
1	1	1	2
2	2	1	1
3	3	2	3
4	4	2	1

	Delivery_ID	Courier_Name	Status	Delivery_Date	Speed
1	1	DHL Express	In Transit	2024-01-01 10:30:00.000	Fast
2	2	FedEx	Delivered	2024-01-02 12:00:00.000	Normal
3	3	UPS	Pending	NULL	Normal
4	4	Amazon Logistics	In Transit	2024-01-03 14:15:00.000	Fast
5	5	Blue Dart	Delivered	2024-01-04 16:45:00.000	Normal
6	6	USPS	Pending	NULL	Normal
7	7	DTDC	In Transit	2024-01-05 11:20:00.000	Normal
8	8	Aramex	Delivered	2024-01-06 13:10:00.000	Fast

	Order_ID	Order_Date	Total_Amount	Status_Confirmation	User_ID	Delivery_ID	Destination
1	1	2024-01-01 00:00:00.000	150.00	Pending	1	NULL	123 Elm St, Springfield
2	2	2024-01-02 00:00:00.000	500.00	Completed	2	2	456 Oak St, Springfield
3	3	2024-01-03 00:00:00.000	60.00	Pending	3	NULL	789 Pine St, Springfi...
4	4	2024-01-04 00:00:00.000	120.00	Completed	4	5	321 Maple St, Spring...
5	5	2024-01-05 00:00:00.000	30.00	Cancelled	5	NULL	654 Cedar St, Springf...
6	6	2024-01-06 00:00:00.000	300.00	Pending	6	NULL	987 Birch St, Springfi...
7	7	2024-01-07 00:00:00.000	80.00	Completed	7	10	147 Walnut St, Sprin...
8	8	2024-01-08 00:00:00.000	200.00	Pending	8	NULL	258 Cherry St, Spring...

	Product_ID	Order_ID	Quantity_Ordered
1	11	11	1
2	12	12	2
3	13	13	4
4	14	14	1
5	15	15	3
6	16	16	1

Query executed successfully.

Gamila.ECommerce...- Diagram_Schema | SQLQueryFinal.sql...(GAMILA\mself (52))*

```
SELECT *
from Product_Order;
SELECT *
from Review;
SELECT *
from Payment;
SELECT *
from Warehouse;
SELECT *
from Product_Warehouse;
SELECT *
from Inventory;
```

100 %

Results | Messages

	Review_ID	Rating	Comment	Created_At	User_ID	Product_ID
1	1	5	Excellent product!	2024-12-24 19:37:23.670	1	1
2	2	4	Very good quality.	2024-12-24 19:37:23.670	2	2
3	3	3	Average experience.	2024-12-24 19:37:23.670	3	3
4	4	2	Not what I expected.	2024-12-24 19:37:23.670	4	4

	Payment_ID	Amount	Payment_Method	Order_ID	Payment_Date
1	1	150.00	Credit Card	1	2024-01-01 10:30:00.000
2	2	500.00	PayPal	2	2024-01-02 11:00:00.000
3	3	60.00	Credit Card	3	2024-01-03 09:45:00.000
4	4	120.00	Debit Card	4	2024-01-04 14:15:00.000
5	5	30.00	Cash	5	2024-01-05 13:20:00.000
6	6	300.00	Credit Card	6	2024-01-06 15:30:00.000
7	7	80.00	PayPal	7	2024-01-07 16:45:00.000
8	8	200.00	Debit Card	8	2024-01-08 10:50:00.000

	Warehouse_ID	Warehouse_Name	Warehouse_Location
1	1	Central Warehouse	New York, NY
2	2	East Coast Storage	Boston, MA
3	3	West Coast Hub	Los Angeles, CA
4	4	Midwest Distribution	Chicago, IL
5	5	South Warehouse	Houston, TX
6	6	North Storage	Minneapolis, MN
7	7	Mountain Region Hub	Denver, CO
8	8	Pacific Northwest De...	Seattle, WA

	Warehouse_ID	Product_ID	Stock_Quantity
1	1	1	500
2	1	2	300
3	1	51	30
4	2	3	450

	Inventory_ID	Stock_Quantity	Warehouse_ID	Date
1	1	500	1	2024-01-01 00:00:00.000
2	2	300	2	2024-01-02 00:00:00.000
3	3	450	3	2024-01-03 00:00:00.000
4	4	250	4	2024-01-04 00:00:00.000
5	5	600	5	2024-01-05 00:00:00.000
6	6	400	6	2024-01-06 00:00:00.000
7	7	700	7	2024-01-07 00:00:00.000

Query executed successfully.

Queries and it's Corresponding output and relational algebra

1. List all products in the "Electronics" category.

SQL:

SELECT Product_Name

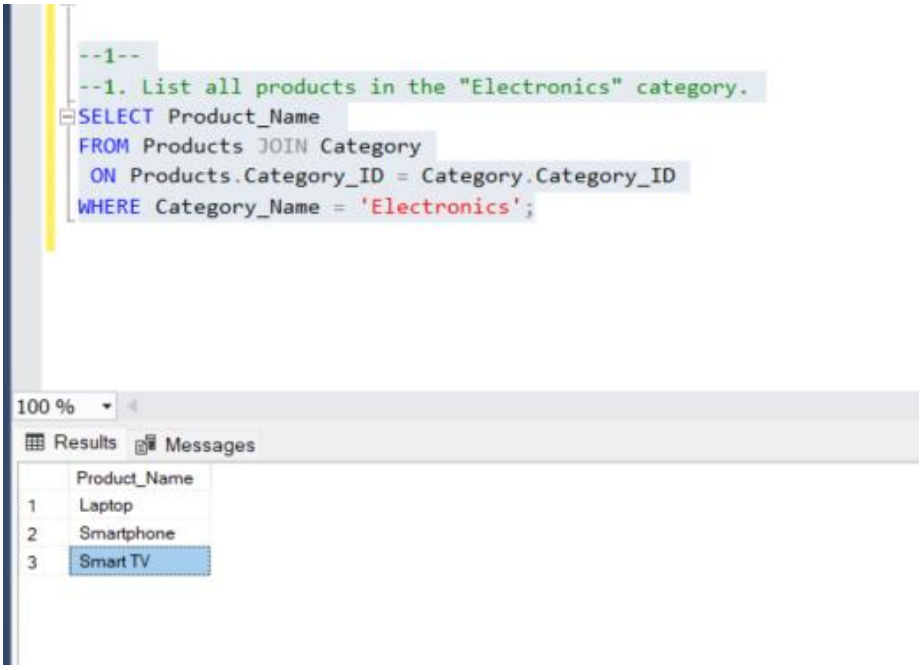
FROM Products JOIN Category

ON Products.Category_ID = Category.Category_ID

WHERE Category_Name = 'Electronics';

Relational Algebra:

π Product_Name (σ Category_Name = 'Electronics' (Products ⋈ Category))



2. Find the total number of orders placed by each customer(user).

SQL:

SELECT User_ID, COUNT(Order_ID) AS Total_Orders

FROM [Order]

GROUP BY User_ID;

Relational Algebra: **Total_Orders** ← γ User_ID, COUNT(Order_ID)

R ← π User_ID, **Total_Orders** (Orders)

```
--2--
--2. Find the total number of orders placed by each customer(user).
SELECT User_ID, COUNT(Order_ID) AS Total_Orders
FROM [Order]
GROUP BY User_ID;
```

100 %

Results | Messages

	User_ID	Total_Orders
1	1	2
2	2	2
3	3	2
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1
11	11	1
12	12	1
13	13	1
14	14	1
15	15	1
16	16	1
17	17	1
18	18	1
19	19	1
20	20	1
21	21	1
22	22	1
23	23	1

3. Identify Customers(users) who have placed more than 1 orders.

SQL:

SELECT U.User_ID, U.Username

FROM [User] U JOIN [Order] o

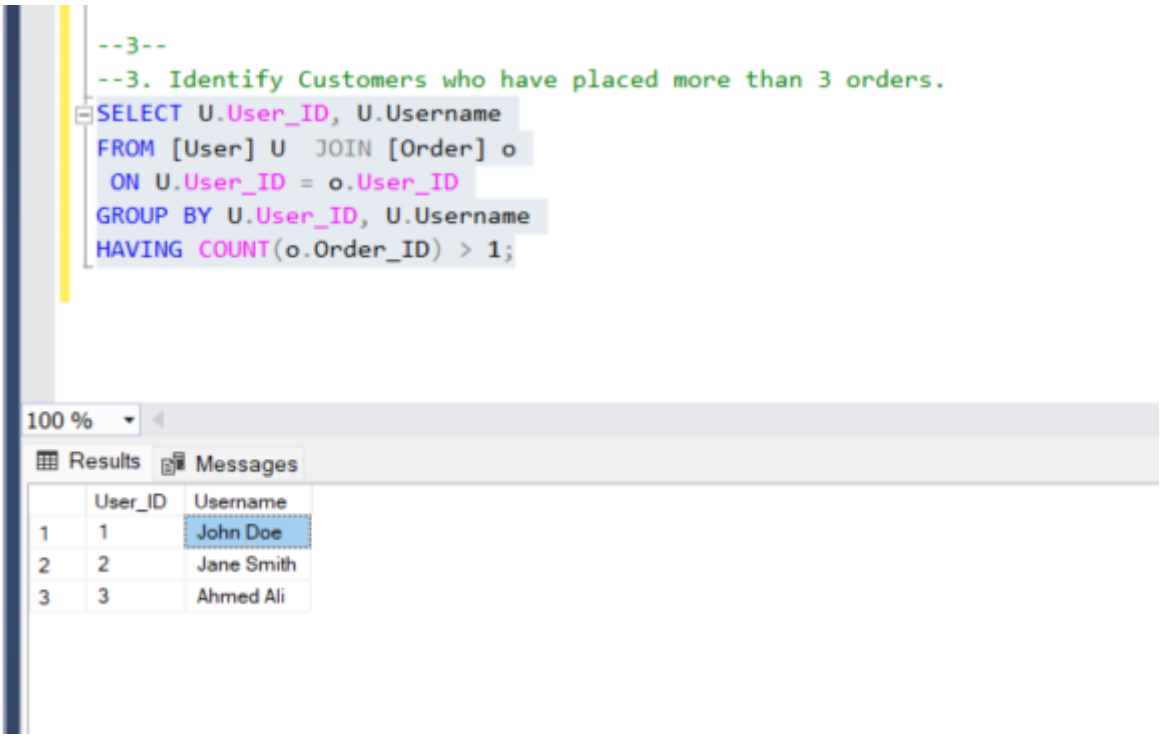
ON U.User_ID = o.User_ID

GROUP BY U.User_ID, U.Username

HAVING COUNT(o.Order_ID) > 1;

Relational Algebra:

π User_ID, Username (σ COUNT(Order_ID) > 1 (γ User_ID, Username, COUNT(Order_ID) (User ⋈ Orders)))



4. Top 10 Best Selling Products:

SQL:

SELECT TOP 10 p.Product_Name, SUM(o.Quantity_Ordered) AS Total_Quantity_Sold

FROM Products p JOIN Product_Order o

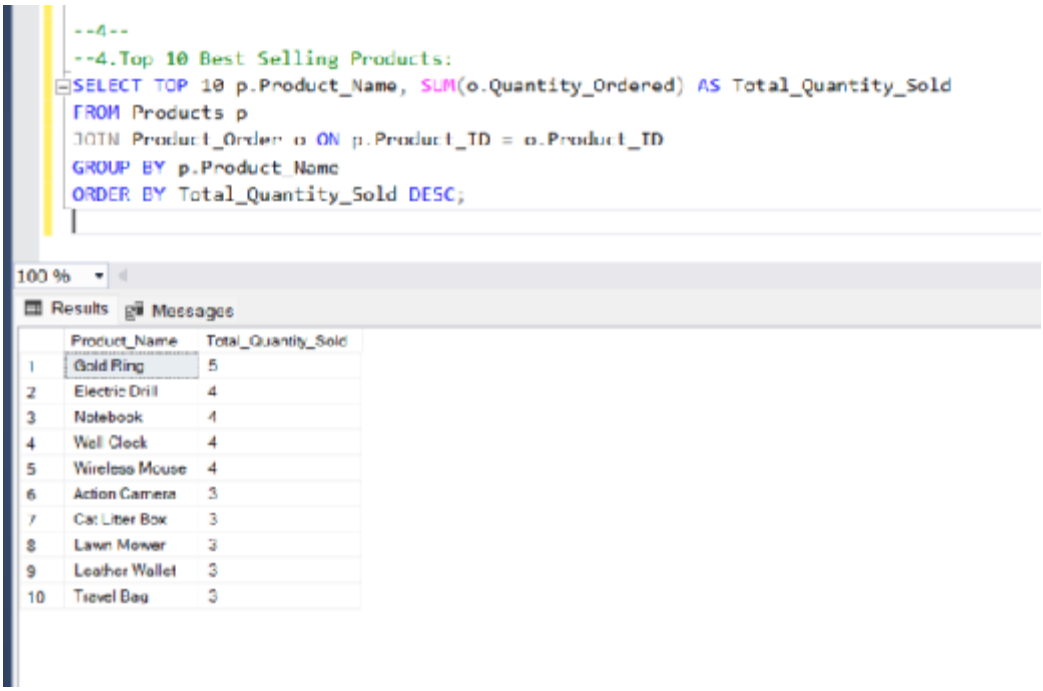
ON p.Product_ID = o.Product_ID

GROUP BY p.Product_Name

ORDER BY Total_Quantity_Sold DESC;

Relational Algebra:

π Product_Name, SUM(Quantity_Ordered) (γ Product_Name, SUM(Quantity_Ordered) (Products ⋈ Product_Order))



```
--4--
--4.Top 10 Best Selling Products:
SELECT TOP 10 p.Product_Name, SUM(o.Quantity_Ordered) AS Total_Quantity_Sold
FROM Products p
JOIN Product_Order o ON p.Product_ID = o.Product_ID
GROUP BY p.Product_Name
ORDER BY Total_Quantity_Sold DESC;
```

100 %

Results | Messages

	Product_Name	Total_Quantity_Sold
1	Gold Ring	5
2	Electric Drill	4
3	Notebook	4
4	Wall Clock	4
5	Wireless Mouse	4
6	Action Camera	3
7	Cat Litter Box	3
8	Lawn Mower	3
9	Leather Wallet	3
10	Travel Bag	3

5. Find the products that have never been ordered.

SQL:

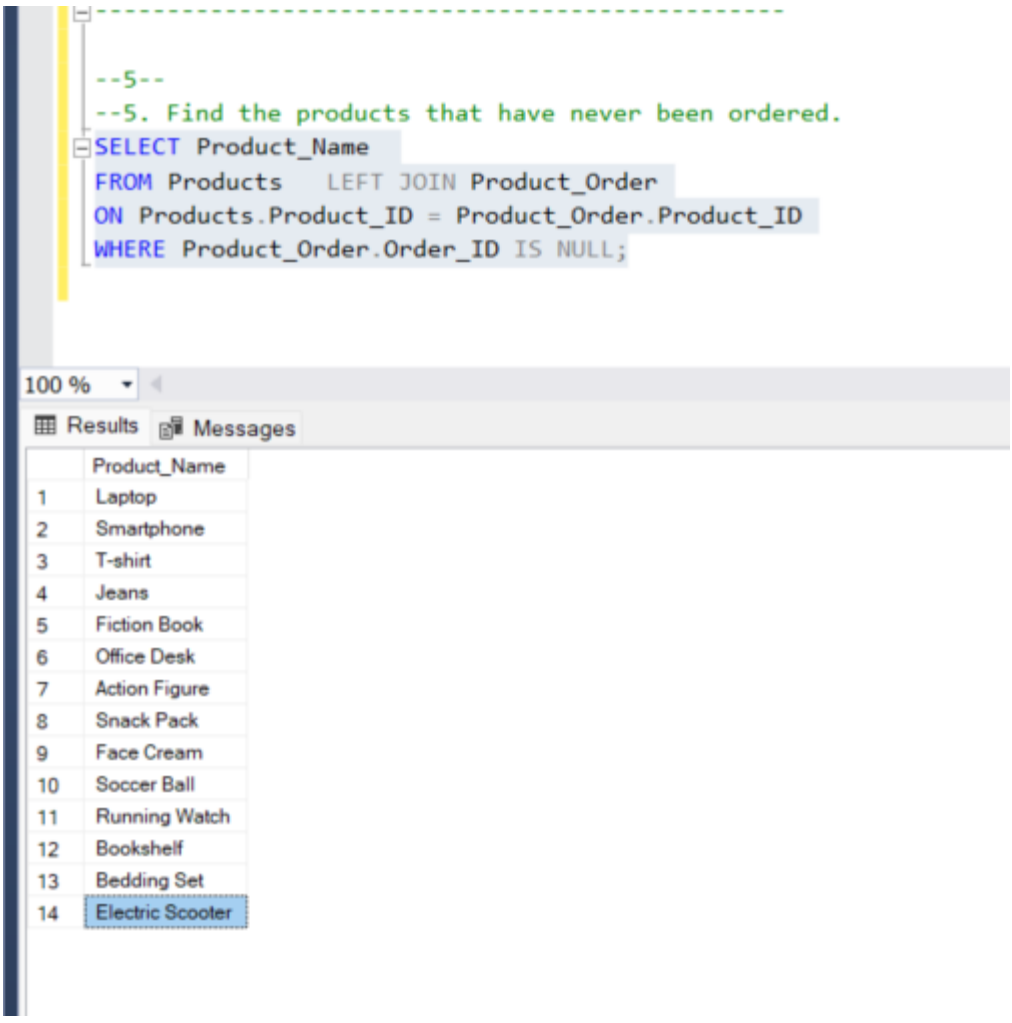
SELECT Product_Name

FROM Products LEFT JOIN Product_Order

ON Products.Product_ID = Product_Order.Product_ID

WHERE Product_Order.Order_ID IS NULL;

Relational Algebra: π Product_Name (σ Order_ID IS NULL (Products ⟕ Product_Order))



6. Retrieves the names and stock quantities of all products available in a specific warehouse

SQL:

SELECT

p.Product_Name, w.Warehouse_Name, h.Stock_Quantity

FROM

Products p JOIN Product_Warehouse h

ON p.Product_ID = h.Product_ID JOIN Warehouse w

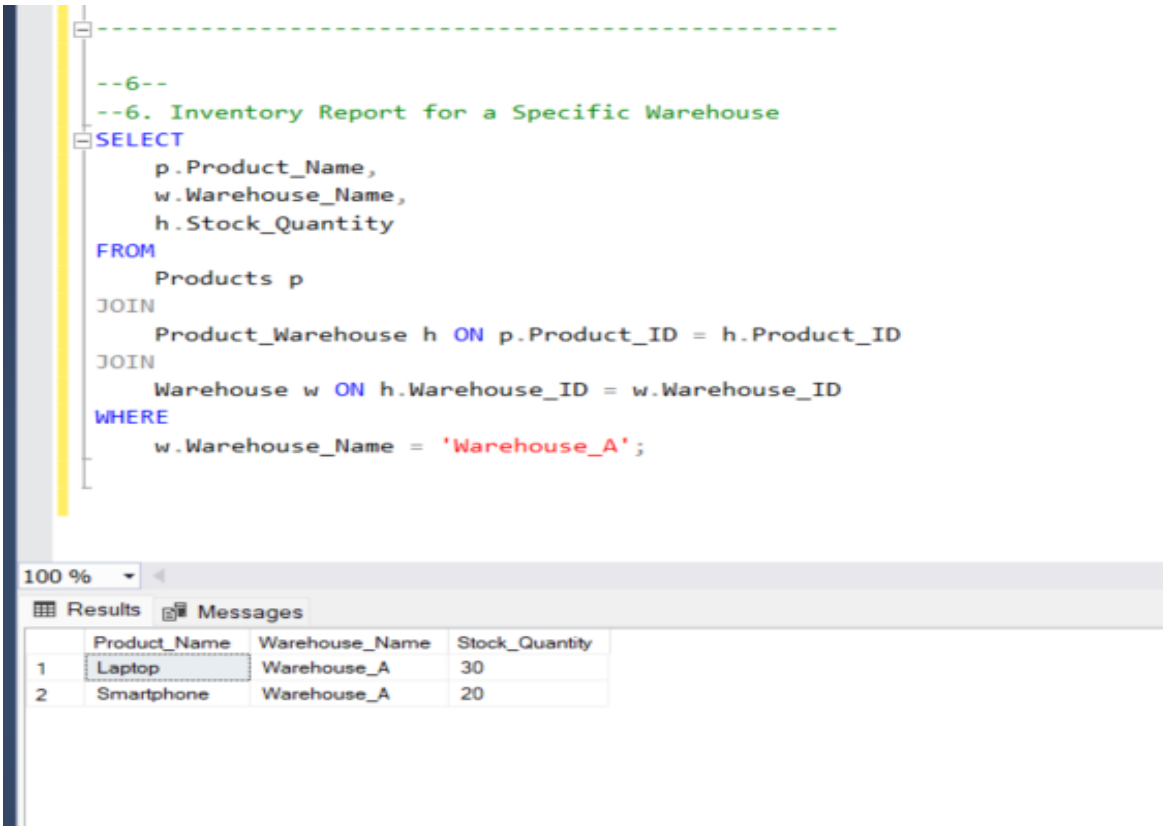
ON h.Warehouse_ID = w.Warehouse_ID

WHERE

w.Warehouse_Name = 'Warehouse_A';

Relational Algebra:

π Product_Name, Warehouse_Name, Stock_Quantity (σ Warehouse_Name = 'Warehouse_A' (Products ⋈ Product Warehouse ⋈ Warehouse))



```
----------------------------------------------------

--6--
--6. Inventory Report for a Specific Warehouse
SELECT
    p.Product_Name,
    w.Warehouse_Name,
    h.Stock_Quantity
FROM
    Products p
JOIN
    Product_Warehouse h ON p.Product_ID = h.Product_ID
JOIN
    Warehouse w ON h.Warehouse_ID = w.Warehouse_ID
WHERE
    w.Warehouse_Name = 'Warehouse_A';
```

100 %

Results | Messages

	Product_Name	Warehouse_Name	Stock_Quantity
1	Laptop	Warehouse_A	30
2	Smartphone	Warehouse_A	20

7. Products with Low Stock Across All Warehouses

SQL:

```
SELECT
   p.Product_Name,  SUM(h.Stock_Quantity) AS Total_Quantity
FROM Products p JOIN Product_Warehouse h
   ON p.Product_ID = h.Product_ID
GROUP BY
   p.Product_ID, p.Product_Name
HAVING
   SUM(h.Stock_Quantity) < 230;
```

Relational Algebra:

π Product_Name, Total_Quantity (σ Total_Quantity < 230 (γ Product_ID, Product_Name, SUM(Stock_Quantity) (Products ⋈ Product_Warehouse)))

```
--7--
--7. Products with Low Stock Across All Warehouses
SELECT
    p.Product_Name,
    SUM(h.Stock_Quantity) AS Total_Quantity
FROM
    Products p JOIN Product_Warehouse h
    ON p.Product_ID = h.Product_ID
GROUP BY
    p.Product_ID, p.Product_Name
HAVING
    SUM(h.Stock_Quantity) < 230;

------------------------------------------------------------

--8--
```

100 %

Results | Messages

	Product_Name	Total_Quantity
1	Face Cream	200
2	Guitar	120
3	Wrist Watch	210
4	Electric Drill	200
5	Product_A	60
6	Product_B	40
7	Product_C	100

8. Product Reviews with Highest Ratings

SQL:

```
SELECT
   p.Product_Name, AVG(r.Rating) AS Average_Rating
FROM   Products p JOIN   Review r
   ON p.Product_ID = r.Product_ID
GROUP BY
   p.Product_ID, p.Product_Name
ORDER BY
   Average_Rating DESC;
```

Relational Algebra: π Product_Name, AVG(Rating) (γ Product_ID, Product_Name, AVG(Rating) (Products ⋈ Review))

```
--8--
--8. Product Reviews with Highest Ratings
SELECT
    p.Product_Name,
    AVG(r.Rating) AS Average_Rating
FROM
    Products p JOIN   Review r
    ON p.Product_ID = r.Product_ID
GROUP BY
    p.Product_ID, p.Product_Name
ORDER BY
    Average_Rating DESC;
```

100 %

Results | Messages

	Product_Name	Average_Rating
1	Laptop	5
2	Office Desk	5
3	Car Cover	5
4	Pet Collar	5
5	Painting Set	5
6	Bluetooth Speaker	5
7	Backpack	5
8	Hair Dryer	5
9	Kids Tablet	5
10	Bedding Set	5
11	Rice Cooker	4
12	Running Watch	4
13	Car Vacuum	4
14	Water Bottle	4
15	Leather Wallet	4
16	Office Chair	4
17	Gold Ring	4
18	Guitar	4
19	Action Figure	4
20	Smartphone	4
21	T-shirt	3
22	Snack Pack	3
23	Notebook	3

9. Count the number of users by country

SQL :

```
SELECT Country, COUNT(*) AS UserCount
FROM [User]
GROUP BY Country
ORDER BY UserCount DESC;
```

Relational Algebra: **UserCount** ← γ **Country, COUNT(*)**

R ← π User_ID, **Total_Orders** (**User**)

```
SELECT Country, COUNT(*) AS UserCount
FROM [User]
GROUP BY Country
ORDER BY UserCount DESC;
```

106 %

Results | Messages

	Country	UserCount
1	USA	14
2	UK	7
3	Australia	7
4	Canada	7
5	New Zealand	5
6	Egypt	3
7	Ireland	2
8	Mexico	1
9	China	1
10	Russia	1
11	Spain	1
12	Syria	1

10. the cart with the maximum number of products

SQL :

SELECT Cart_ID, Number_of_Product

FROM Cart

WHERE Number_of_Product = (SELECT MAX(Number_of_Product) FROM Cart);

Relational Algebra:

πCart_ID,Number_of_Product(σNumber_of_Product=MAX(Number_of_Product)(Cart))

```
SELECT Cart_ID, Number_of_Product
FROM Cart
WHERE Number_of_Product = (SELECT MAX(Number_of_Product) FROM Cart);
```

106 %

Results | Messages

	Cart_ID	Number_of_Product
1	10	10
2	20	10
3	30	10
4	40	10
5	50	10

11.the average rating for each product

SQL :

SELECT Product_ID, AVG(Rating) AS Avg_Rating

FROM Review

GROUP BY Product_ID;

Relational Algebra:

γProduct_ID,AVG(Rating)→Avg_Rating(πProduct_ID,Rating(Product_Reviews))

```
SELECT Product_ID, AVG(Rating) AS Avg_Rating
FROM Review
GROUP BY Product_ID;
```

%

Results | Messages

Product_ID	Avg_Rating
1	5
2	4
3	3
4	2
5	1
6	5
7	4
8	3
9	2
10	1
11	5
12	4
13	3
14	2
15	1
16	5
17	4
18	3
19	2
20	1
21	5
22	4
23	3
24	2
25	1
26	5
27	4
28	3
29	2
30	1
31	5

12.Number of completed orders

SQL :

SELECT COUNT(Order_ID) AS Completed_Orders

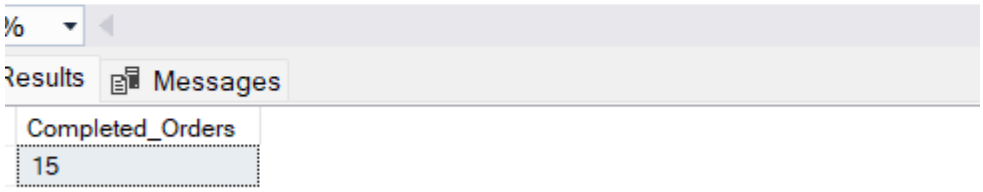
FROM [Order]

WHERE Order_Status = 'Completed';

Relational Algebra:

γCOUNT(Order_ID)→Completed_Orders(σOrder_Status='Completed'(Order))

```
SELECT COUNT(Order_ID) AS Completed_Orders
FROM [Order]
WHERE Status_Confirmation = 'Completed';
```



Gui for E-commerce platform problem