IT-131 DATABASE

Project

Online retail Application store database

D. T.. A..

الاسم
ش
أ
م
ل
غ
ر
م
ن

Introduction

This project is part of the IT131 course, and the main goal was to apply what we learned in database systems in a practical way. We chose to work on a Online retail Application store database because it's something real and familiar — like Amazon or Noon apps that we use all the time.

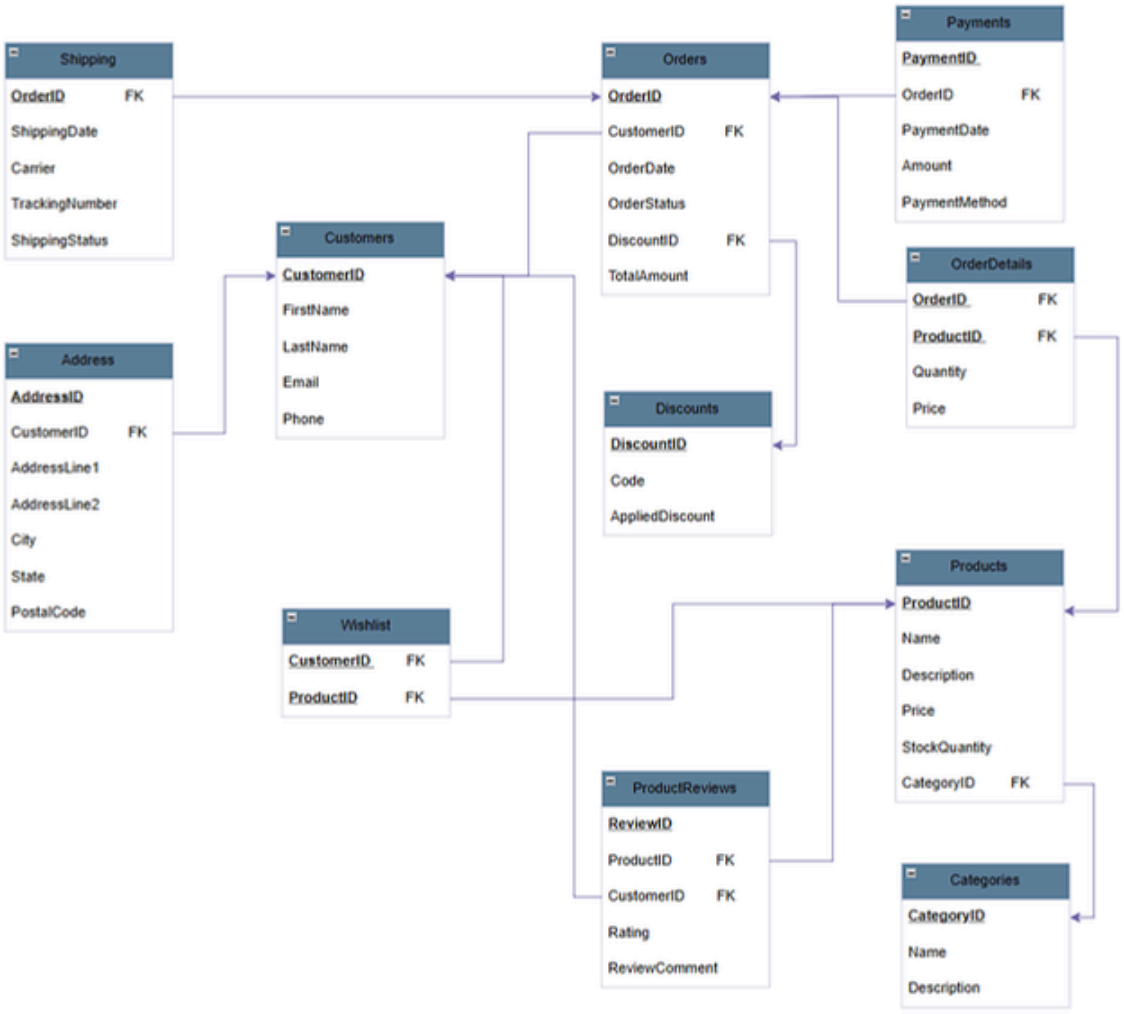
We started by identifying the main entities such as Customers, Products, Orders, Shipping, and Payments, and designed the database to include all the basic operations that a real online store needs. We used SQL to create the tables, insert the data, and run different types of queries, while making sure the tables are connected correctly using primary and foreign keys.

Throughout the project, we learned how to organize and link data, and how to use different SQL queries like joins and nested queries. Our main goal was to build a database that looks real and would actually work if used in a real online shop.

Goals & Objectives

1. **Design a realistic database** that includes essential tables and relationships used in e-commerce platforms.
2. **Apply SQL commands** (DDL and DML) to create tables, insert data, and execute various queries.
3. **Understand table relationships** by using primary and foreign keys to ensure referential integrity.
4. **Execute advanced queries** such as JOINs and nested subqueries.
5. **Create a professional ERD model.**
6. **Apply normalization techniques** (1NF, 2NF, 3NF, 4NF) to improve table design and reduce redundancy.
7. **Connect theory to practice** by simulating a real-life online store database system.

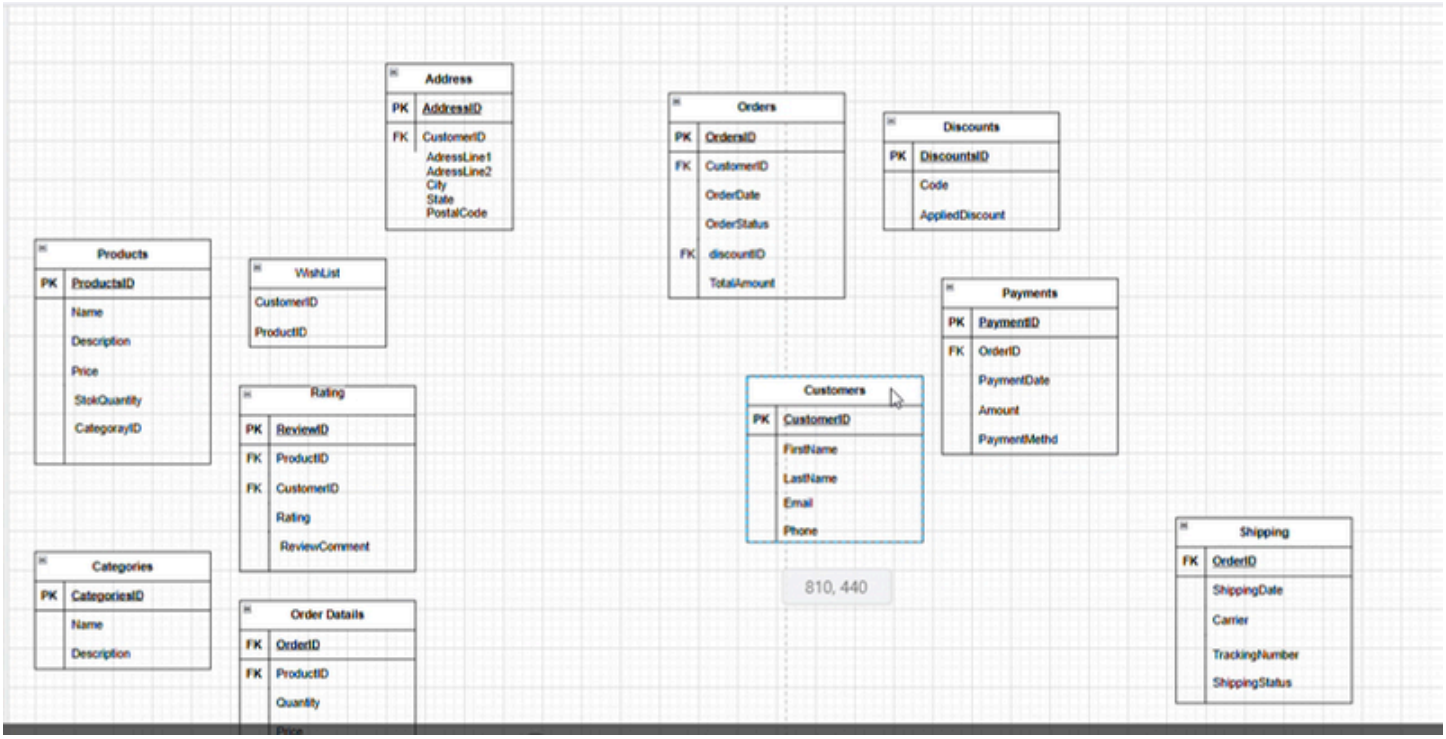
SCHEMA


Shipping
OrderID FK
ShippingDate
Carrier
TrackingNumber
ShippingStatus
Orders
OrderID
CustomerID FK
OrderDate
OrderStatus
DiscountID FK
TotalAmount
Payments
PaymentID
OrderID FK
PaymentDate
Amount
PaymentMethod
Customers
CustomerID
FirstName
LastName
Email
Phone
OrderDetails
OrderID FK
ProductID FK
Quantity
Price
Address
AddressID
CustomerID FK
AddressLine1
AddressLine2
City
State
PostalCode
Discounts
DiscountID
Code
AppliedDiscount
Products
ProductID
Name
Description
Price
StockQuantity
CategoryID FK
Wishlist
CustomerID FK
ProductID FK
ProductReviews
ReviewID
ProductID FK
CustomerID FK
Rating
ReviewComment
Categories
CategoryID
Name
Description

ER DIAGRAM

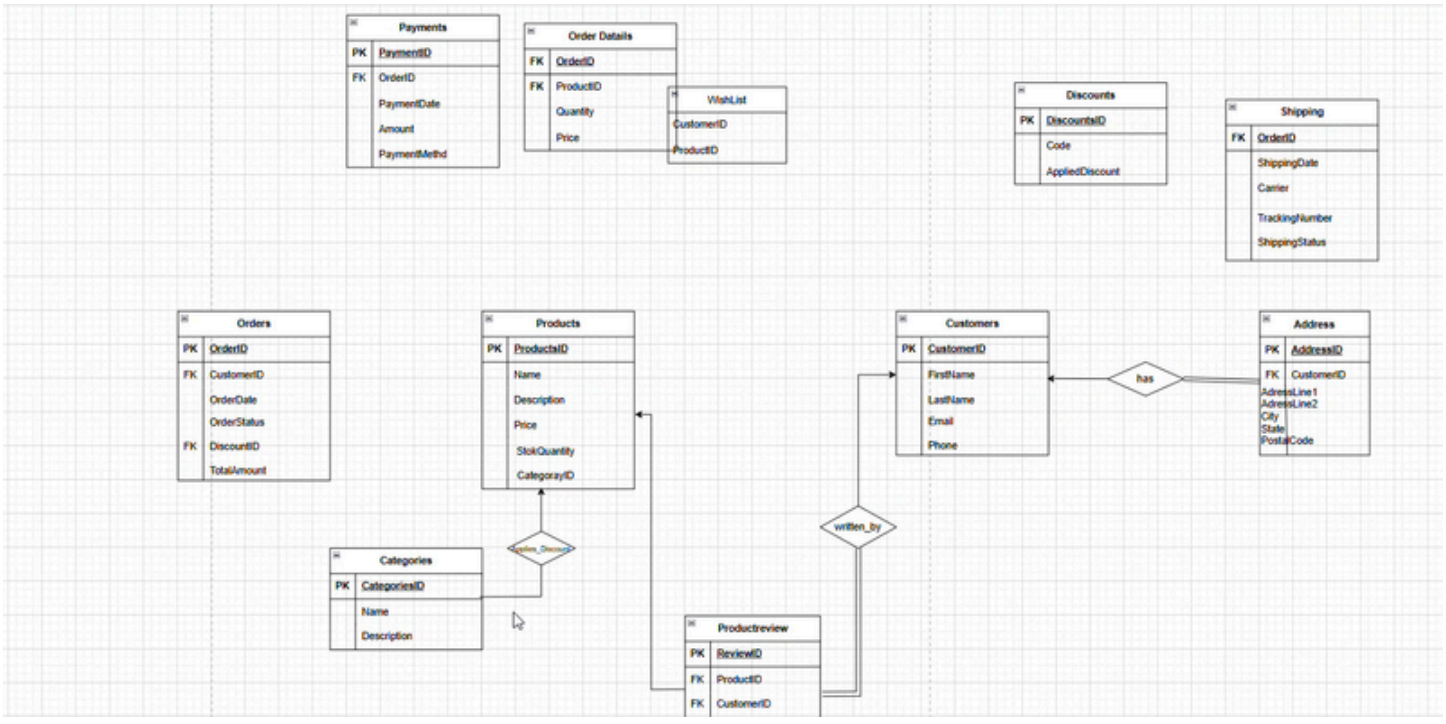
Step 1:

Defined all database tables as entities, and listed their corresponding attributes including primary keys.



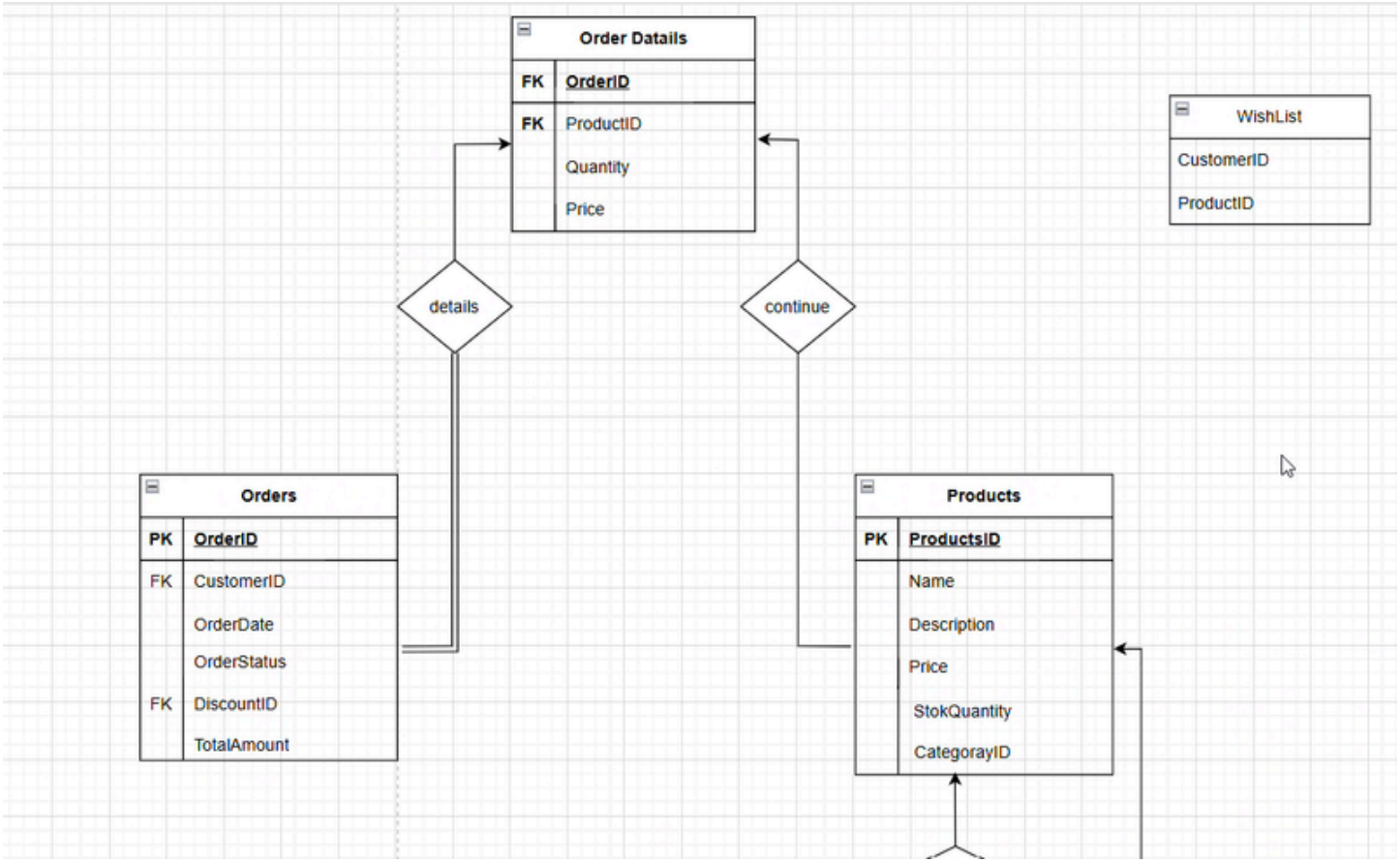
Step 2:

Added relationships and cardinalities between core entities like Customer, Address, Product, ProductReview, and Category. Proper symbols were used to connect them clearly.



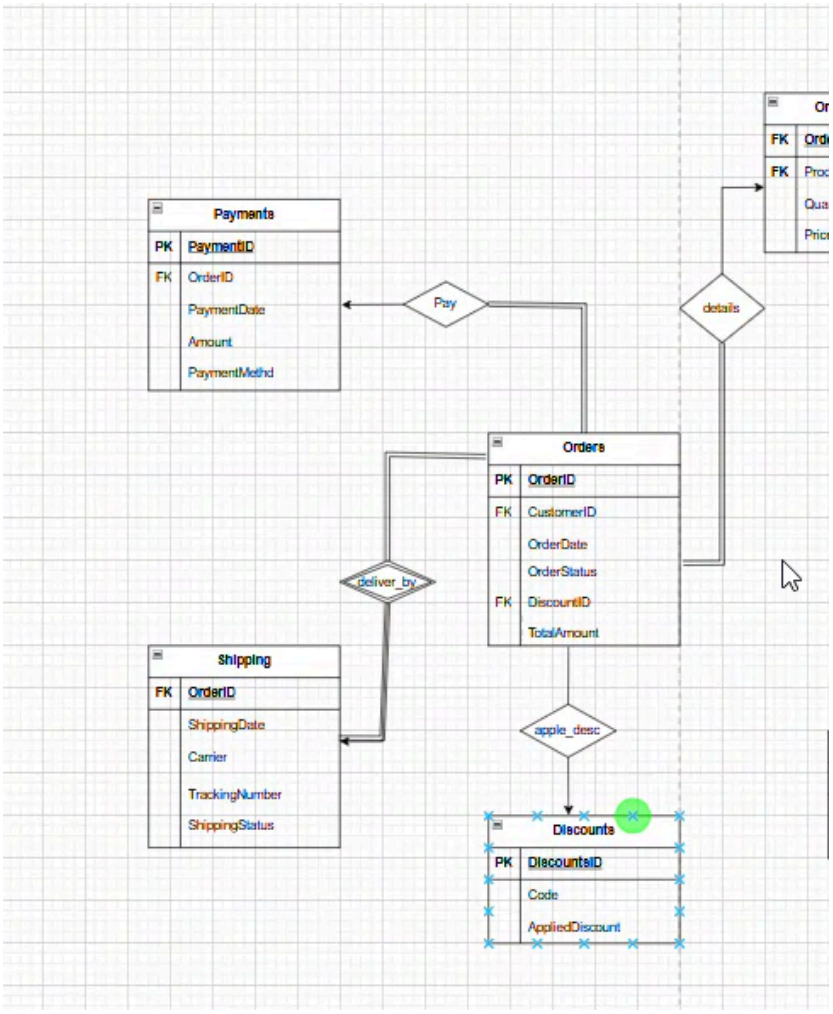
Step 3:

Focused on transaction-related entities such as Order, Product, and OrderDetails. Used a weak entity representation for OrderDetails with identifying relationships from Order and Product



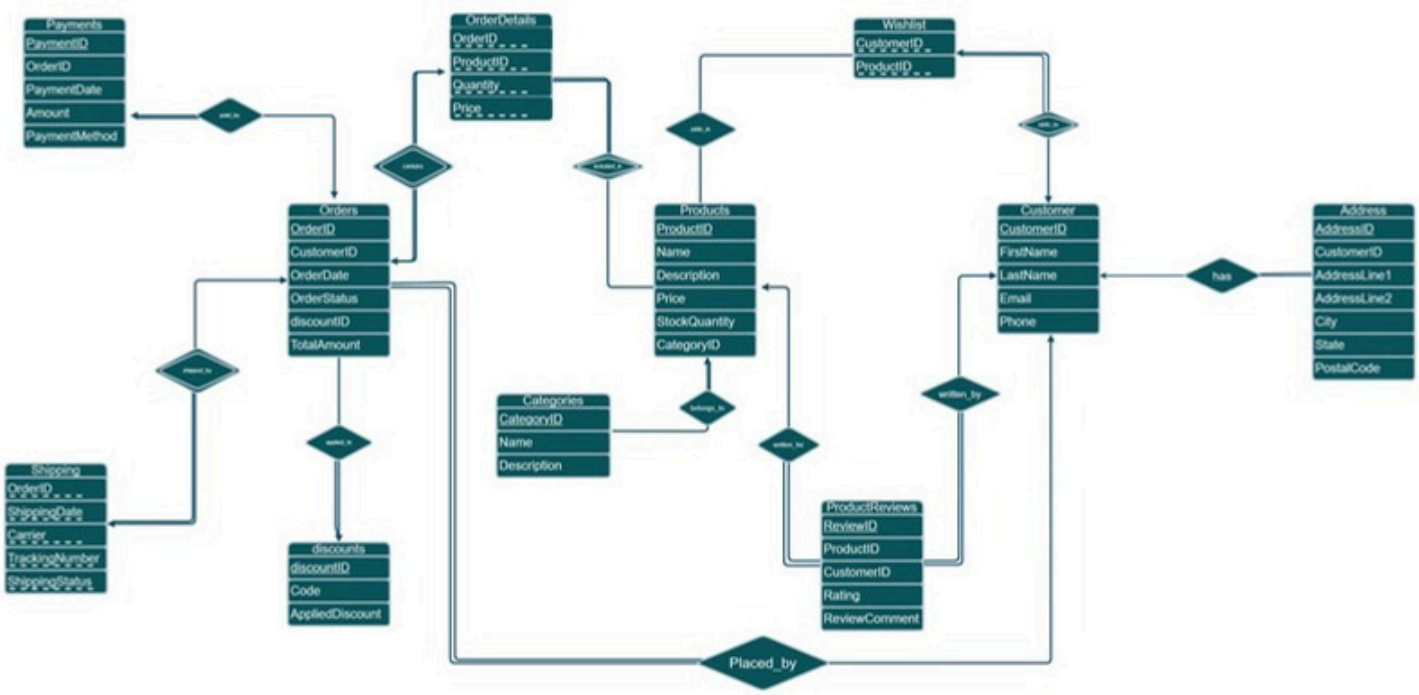
Step 4:

Incorporated additional supporting entities such as Shipping, Payment, and Discount. These were linked to the Order entity with accurate one-to-one or one-to-many relationships.


Payments
PK PaymentID
FK OrderID
PaymentDate
Amount
PaymentMethd
Pay
details
Orders
PK OrderID
FK CustomerID
OrderDate
OrderStatus
FK DiscountID
TotalAmount
deliver_by
Shipping
FK OrderID
ShippingDate
Carrier
TrackingNumber
ShippingStatus
apple_desc
Discounts
PK DiscountsID
Code
AppliedDiscount

Step 5:

Completed the final ER diagram.



video code:

DDL

Customer table

```
CREATE TABLE Customers (

CustomerID varchar(10),

FirstName VARCHAR(20) not null,

LastName VARCHAR(20) not null,

 Email VARCHAR(25) not null,

Phone VARCHAR(18),

CONSTRAINT pk_customers PRIMARY KEY (CustomerID)

);
```

Address table

```
 CREATE TABLE Address (

Addresid varchar(10),

Customerid varchar(10) unique not null,

AddressLine1 varchar(100) not null,

AddressLine2 varchar(100),

 City VARCHAR(20) not null,

State VARCHAR(10) not null,

PostalCode VARCHAR(20),

CONSTRAINT fk_CustomerWithaddress FOREIGN KEY (Customerid ) REFERENCES customers(Customerid)

CONSTRAINT pk_address PRIMARY KEY (addresid)

);
```

Categories table

```
 CREATE TE TABLE Categories (

CategoryID varchar(10) ,

Name VARCHAR(25) not null,

Description varchar(100) not null ,

CONSTRAINT pk_categories PRIMARY KEY (CategoryID)

);
```

Products table

```
CREATE TABLE Products (

ProductID varchar(10),

Name VARCHAR(25) not null,

Description varchar(100),

Price DECIMAL(10,2) not null,

StockQuantity INT,

CategoryID varchar(10),

CONSTRAINT pk_products PRIMARY KEY (ProductID),

CONSTRAINT fk_productsWithCategories FOREIGN KEY (CategoryID)
REFERENCES Categories(CategoryID),

 Constraint checkPriceForProduct Check (price >0 ),

Constraint checkQauntityforProduct chrck(StockQuantity >= 0)

);
```

Orders table

```
CREATE TABLE Orders (

OrderID varchar(10),

CustomerID varchar(10) ,

OrderDate DATE not null,

OrderStatus VARCHAR(20) not null,

discountID varchar(10),

TotalAmount DECIMAL(10,2) not null,

CONSTRAINT pk_orders PRIMARY KEY (OrderID)

CONSTRAINT fk_ordersWithCustomers FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

CONSTRAINT fk_ordersWithPromotion FOREIGN KEY (discountID) REFERENCES discounts (discountID),

Constraint checkTotalPriceForOrder check( totalAmount > 0),

Constraint checkStatusForOrder check (UPPER(OrderStatus) IN('PROCESSING','SHIPPED','DELIVERED','CANCELLED'))

);
```

Order details table

```
CREATE TABLE OrderDetails (

 OrderID varchar(10),

ProductID varchar(10),

 Quantity INT not null,

 Price DECIMAL(10,2) not null,

CONSTRAINT pk_orderdetails PRIMARY KEY (OrderID,ProductID),
```

CONSTRAINT fk_orderdetailsWithOrders FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

CONSTRAINT fk_orderdetailsWithProducts FOREIGN KEY (ProductID) REFERENCES Products(ProductID),

Constraint checkQauntityforOrder chrck(Quantity > 0)

);

Payments table

CREATE TABLE Payments (

PaymentID varchar(10),

OrderID varchar(10)

PaymentDate DATE not null,

Amount DECIMAL(10,2) not null ,

PaymentMethod VARCHAR(20) not null ,

Constraint checkPaymentMethod CHECK (UPPER(PaymentMethod) IN ('VISA','APPLE PAY','CASH ON DELIVERY','PAYPAL'))

CONSTRAINT pk_payments PRIMARY KEY (PaymentID) ,

CONSTRAINT fk_paymentsWithOrders FOREIGN KEY (OrderID) REFERENCES),
Orders(OrderID

Constraint checkAmountForPayment check(amount >= 0)

);

Shipping table

CREATE TABLE Shipping (

OrderID varchar(10) ,

ShippingDate DATE not null,

Carrier VARCHAR(50) not null,

TrackingNumber VARCHAR(50) not null,

ShippingStatus VARCHAR(20) not null,

CONSTRAINT pk_shipping PRIMARY KEY (OrderID) ,

CONSTRAINT fk_shippingWithOrders FOREIGN KEY (OrderID) REFERENCES , Orders(OrderID)

Constraint checkShippingStatus check(UPPER(Shipping Status) IN('IN , TRANSIT','DELIVERED'))

);

Wish list table

CREATE TABLE Wishlist (

CustomerID varchar(10) ,

ProductID varchar(10) ,

CONSTRAINT pk_wishlist PRIMARY KEY (CustomerID,productid),

CONSTRAINT fk_wishlistWithCustomers FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

CONSTRAINT fk_wishlistWithProducts FOREIGN KEY (ProductID) REFERENCES) Products(ProductID

);

Product reviews table

CREATE TABLE ProductReviews (

ReviewID varchar(10),

ProductID varchar(10) ,

CustomerID varchar(10) ,

Rating decimal(2,1) not null,

ReviewComment varchar(350) ,

CONSTRAINT pk_productreviews PRIMARY KEY (ReviewID),

CONSTRAINT fk_reviewsWithProducts FOREIGN KEY (ProductID) REFERENCES , Products(ProductID)

CONSTRAINT fk_reviewsWithCustomers FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

Constraint checkRating check(Rating Between 1 and 5)

);

Discounts table

```
CREATE TABLE Discounts (

DiscountID varchar(10),

Code varchar(20) unique not null,

AppliedDiscount varchar(5) not null,

CONSTRAINT pk_ discounts PRIMARY KEY (discountID),

Constraint checkAppliedDiscount CHECK (AppliedDiscount >= 0 AND
AppliedDiscount <= 100)

);
```

DML

Queries

1: The total number of orders from customers in" Buraidah".

This query counts how many total orders came from customers living in the city of Buraidah. It joins the Orders, Customers, and Address tables to link each order to the customer's city. The WHERE clause filters the results to only those where the city is "Buraidah".

SELECT COUNT(*) AS TotalOrdersFromBuraidah.

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Address a ON c.CustomerID = a.CustomerID

WHERE a.City = 'Buraidah';

```sql
SELECT COUNT(*) AS TotalOrdersFromBuraidah
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Address a ON c.CustomerID = a.CustomerID
WHERE a.City= 'Buraidah';
```

TOTALORDERSFROMBURAIDAH
2

2:Monthly order count for each month (from highest to lowest)

This query calculates how many orders were placed in each month. It uses the MONTH() function to extract the month from the order date, then groups the results by month and counts the number of orders. Finally, it sorts the months from highest to lowest by order count

---

SELECT

EXTRACT(MONTH FROM o.OrderDate) AS Month,

COUNT(*) AS OrdersCount

FROM Orders o

GROUP BY EXTRACT(MONTH FROM o.OrderDate)

ORDER BY OrdersCount DESC;

```sql
SELECT
  EXTRACT(MONTH FROM o.OrderDate) AS Month,
  COUNT(*) AS OrdersCount
FROM Orders o
GROUP BY EXTRACT(MONTH FROM o.OrderDate)
ORDER BY OrdersCount DESC;
```

MONTH	ORDERSCOUNT
9	5
12	4
4	4
10	3
1	3
5	2
8	2
7	2
6	2
2	2
3	1
11	1

3: This query finds the most ordered products in London🇬🇧

This query shows the most frequently ordered products by customers who live in London. It joins the orders, order details, products, customers, and address tables. The query filters to customers from “London”, groups by product, and counts how many times each product was ordered

---

SELECT Products.ProductID, Products.Name, COUNT(*) AS

TimesOrdered

FROM Orders

JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID

JOIN Products ON OrderDetails.ProductID=Products.ProductID

JOIN Customers ON Orders.CustomerID=Customers.CustomerID

JOIN Address ON Customers.CustomerID =Address.CustomersID

WHERE Address.City = 'London'

GROUP BY Products.ProductID, Products.Name

ORDER BY TimesOrdered DESC;

```
SELECT Products.ProductID, Products.Name, COUNT(*) AS TimesOrdered
FROM Orders
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
JOIN Address ON Customers.CustomerID = Address.CustomerID
WHERE Address.City = 'London'
GROUP BY Products.ProductID, Products.Name
ORDER BY TimesOrdered DESC;
```

PRODUCTID	NAME	TIMESORDERED
P007	Velvet Lipstick	1
P015	Micellar Water Cleanser	1

4: This query shows products that have never been ordered.

This query finds products that no one has ever ordered. It does a LEFT JOIN between Products and OrderDetails, then checks where the product ID in OrderDetails is NULL, meaning there is no match — hence, the product was never ordered

---

SELECT Products.ProductID, Products.Name

FROM Products

LEFT JOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID

WHERE OrderDetails.ProductID IS NULL;

```
SELECT Products.ProductID, Products.Name
FROM Products
LEFT JOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID
WHERE OrderDetails.ProductID IS NULL;
```

```
no data found
```

5: This query shows the top city with the most customer orders.

This query identifies the city that placed the highest number of orders. It joins the orders, customers, and address tables, groups the orders by city, counts how many orders each city has, and then selects the one with the most using FETCH FIRST 1 ROW

SELECT

Address.City, COUNT(Orders.OrderID) AS TotalOrders

FROM Orders

JOIN Customers ON Orders.CustomerID = Customers.CustomerID

JOIN Address ON Customers.CustomerID = Address.CustomerID

GROUP BY Address.City

ORDER BY TotalOrders DESC

FETCH FIRST 1 ROWS ONLY;

```
SELECT
  Address.City,
  COUNT(Orders.OrderID) AS TotalOrders
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
JOIN Address ON Customers.CustomerID = Address.CustomerID
GROUP BY Address.City
ORDER BY TotalOrders DESC
FETCH FIRST 1 ROWS ONLY;
```

CITY	TOTALORDERS
Dammam	4

6: This query retrieves products priced between 35 and 90 that have “Lip” in their name.

It selects products where the name contains the substring “Lip” and the price falls within the inclusive range of 35 to 90. This is achieved using the LIKE operator for

pattern matching and the BETWEEN operator for the price range

---

SELECT

p.ProductID,

p.Name AS ProductName,

p.Price,

c.Name AS CategoryName

FROM Products p JOIN Categories c

ON p.CategoryID = c.CategoryID

WHERE p.Price BETWEEN 35 AND 90 AND c.Name LIKE '%Lip%';

```
417  SELECT
418      p.ProductID,
419      p.Name AS ProductName,
420      p.Price,
421      c.Name AS CategoryName
422  FROM Products p
423  JOIN Categories c ON p.CategoryID = c.CategoryID
424  WHERE p.Price BETWEEN 35 AND 90
425      AND c.Name LIKE '%Lip%';
```

PRODUCTID	PRODUCTNAME	PRICE	CATEGORYNAME
P007	Velvet Lipstick	89	Lips
P008	Glossy Lip Shine	79	Lips
P009	Lip Liner Duo	39	Lips

7: This query finds products with ratings above 4 that have been ordered at most three times.

It joins the products, reviews, and order details tables to calculate the average rating and count the number of orders for each product. The HAVING clause filters the results to include only those products with an average rating greater than 4 and an order count of three or fewer

---

SELECT

P.ProductID, P.Name

FROM Products P JOIN ProductReviews R

ON P.ProductID = R.ProductID

JOIN OrderDetails O

ON P.ProductID = O.ProductID

GROUP BY P.ProductID, P.Name

HAVING AVG(R.Rating) > 4 AND SUM(O.Quantity) <= 3;

```
427
428 SELECT P.ProductID, P.Name
429 FROM Products P JOIN ProductReviews R
430     ON P.ProductID = R.ProductID
431 JOIN OrderDetails O
432     ON P.ProductID = O.ProductID
433 GROUP BY P.ProductID, P.Name
434 HAVING AVG(R.Rating) > 4 AND SUM(O.Quantity) <= 3;
435
```

PRODUCTID	NAME
P005	Bold Eyeshadow Palette

8: This query identifies the shipping company that has shipped the most orders.

It joins the orders and shipping companies tables, grouping the data by shipping company. The query counts the number of orders per company and orders the results in descending order to highlight the top-performing shipping company

---

SELECT

S.Carrier, COUNT(*) AS TotalShipments

FROM Shipping S

GROUP BY S.Carrier

ORDER BY TotalShipments DESC;

```
450 SELECT
451   S.Carrier, COUNT(*) AS TotalShipments
452 FROM Shipping S
453 GROUP BY S.Carrier
454 ORDER BY TotalShipments DESC;
455
```

UPS	5
DHL	1
Royal Mail	1

9: This query displays products with total order quantities greater than 10, along with their wishlist counts.

It aggregates order quantities from the order details table and counts wishlist additions from the wish list table. By joining these tables and grouping by product, the query filters for products ordered more than 10 times and provides both order and wishlist counts

---

SELECT

P.ProductID, P.Name, COUNT(W.CustomerID) AS WishlistCount, SUM(OD.Quantity) AS OrderCount

FROM Products P LEFT JOIN Wishlist W

ON P.ProductID = W.ProductID

LEFT JOIN OrderDetails OD

ON P.ProductID = OD.ProductID

GROUP BY P.ProductID, P.Name

HAVING SUM(OD.Quantity) > 10;

```
436  SELECT
437    P.ProductID,
438    P.Name,
439    COUNT(W.CustomerID) AS WishlistCount,
440    SUM(OD.Quantity) AS OrderCount
441  FROM Products P LEFT JOIN Wishlist W
442      ON P.ProductID = W.ProductID
443  LEFT JOIN OrderDetails OD
444      ON P.ProductID = OD.ProductID
445  GROUP BY P.ProductID, P.Name
446  HAVING SUM(OD.Quantity) > 10;
447
```

PRODUCTID	NAME	WISHLISTCOUNT	ORDERCOUNT
P007	Velvet Lipstick	9	12

10:This query retrieves customers who placed orders over 300 without using any discount.

It joins the customers and orders tables, filtering for orders with a total amount exceeding 300 and a discount value of zero. The result includes customers who made high-value purchases without applying any discounts

---

SELECT

C.FirstName, C.LastName AS CustomerName, O.OrderID, O.TotalAmount

FROM Customers C JOIN Orders O

ON C.CustomerID = O.CustomerID

WHERE O.DiscountID IS NULL AND O.TotalAmount > 300;

```
456 SELECT
457   C.FirstName, C.LastName AS CustomerName,
458   O.OrderID,
459   O.TotalAmount
460 FROM Customers C
461 JOIN Orders O ON C.CustomerID = O.CustomerID
462 WHERE O.DiscountID IS NULL AND O.TotalAmount > 300;
463
```

FIRSTNAME	CUSTOMERNAME	ORDERID	TOTALAMOUNT
Layla	Abdullah	1	337
Sara	Ibrahim	29	500
Samiah	Al-Malek	20	407
Hessah	Al-Njar	23	424

11. Most added products to the wishlist:

This query returns the names of products that were added most frequently to customers' wishlists. It joins the wishlist and products tables. Results are grouped

by product name and sorted in descending order by the number of times added.

SELECT p.Name, COUNT(w.ProductID) AS times_added

FROM wishlist w

JOIN products p ON w.ProductID = p.ProductID

GROUP BY p.Name

ORDER BY times_added DESC;

```sql
SELECT p.Name, COUNT(w.ProductID) AS times_added
FROM wishlist w
JOIN products p ON w.ProductID = p.ProductID
GROUP BY p.Name
ORDER BY times_added DESC;
```

NAME	TIMES_ADDED
Bold Eyeshadow Palette	4
Brow Sculpt Gel	3
Velvet Lipstick	3
Lip Liner Duo	3
Hydrating Primer	2
Conceal Pro Stick	2
Matte Foundation	2
Glossy Lip Shine	2
Volume Mascara	1

12. Products with 'matte' in the description:

This query fetches product details where the description contains the word "matte". It uses a wildcard search with LIKE '%matte%' to find matching entries. Results are ordered by price from highest to lowest.

SELECT ProductID, Name, Description, Price, StockQuantity

FROM Products

WHERE Description LIKE '%matte%'

ORDER BY Price DESC;

```
SELECT ProductID, Name, Description, Price, StockQuantity
FROM Products
WHERE Description LIKE '%matte%'
ORDER BY Price DESC;
```

PRODUCTID	NAME	DESCRIPTION	PRICE	STOCKQUANTITY
P001	Matte Foundation	Long-lasting matte finish foundation for oily skin.	159	50

13. Customers with orders paid using multiple payments:

This query shows customers who paid for a single order using more than one payment. It includes customer names, order IDs, and the count of payments. Only orders with more than one payment are displayed using a having clause.

SELECT

c.CustomerID, c.FirstName,c.LastName,o.OrderID,

COUNT(p.PaymentID) AS NumberOfPayments

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN Payments p ON o.OrderID = p.OrderID

GROUP BY c.CustomerID, c.FirstName, c.LastName, o.OrderID

HAVING COUNT(p.PaymentID) > 1;

```sql
SELECT
  c.CustomerID, c.FirstName,c.LastName,o.OrderID,
    COUNT(p.PaymentID) AS NumberOfPayments
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN Payments p ON o.OrderID = p.OrderID
GROUP BY c.CustomerID, c.FirstName, c.LastName, o.OrderID
HAVING COUNT(p.PaymentID) > 1;
```

CUSTOMERID	FIRSTNAME	LASTNAME	ORDERID	NUMBEROFPAYMENTS
15	Emily	Johnson	30	2
21	Faisal	AlHarbi	18	2
26	Serien	Al-Haj	22	2

14. Show order details that used a discount, with customer name and discount percentage:

This query displays order details where a discount was applied, along with the customer name and the discount percentage. It helps evaluate how discounts influence customer behavior and sales performance.

SELECT O.OrderID, C.FirstName || ' ' || C.LastName AS CustomerName, D.Code AS DiscountCode, D.AppliedDiscount, O.TotalAmount

FROM Orders O

JOIN Customers C ON O.CustomerID = C.CustomerID

JOIN Discounts D ON O.DiscountID = D.DiscountID;

```
SELECT O.OrderID, C.FirstName || ' ' || C.LastName AS CustomerName,
       D.Code AS DiscountCode, D.AppliedDiscount, O.TotalAmount
FROM Orders O
JOIN Customers C ON O.CustomerID = C.CustomerID
JOIN Discounts D ON O.DiscountID = D.DiscountID;
```

ORDERID	CUSTOMERNAME	DISCOUNTCODE	APPLIEDDISCOUNT	TOTALAMOUNT
5	Alaa Ahmad	15OFF	15	192.1
7	Alaa Ahmad	TENDEAL	10	267.3
3	Sara Ibrahim	FIVEOFF	5	327.75
4	Huda Sultan	TENDEAL	10	225
11	Reem Khalid	15OFF	15	267.75
9	Noor Farah	TENDEAL	10	373.5
13	Lina Ibrahim	SAVE20	20	300.8
17	Emily Johnson	SAVE20	20	311.2
30	Emily Johnson	FIVEOFF	5	245.1
14	Lina Haddad	DROP30	30	350
15	Noura AlQahtani	FIVEOFF	5	282.15
18	Faisal AlHarbi	DROP30	30	333.9
21	Hadeel Al-Mansori	TENDEAL	10	258.3
22	Serien Al-Haj	FIVEOFF	5	237.5
28	Hessah Al-Njar	FIVEOFF	5	253.65

15. Retrieve the best-selling product (based on total quantity ordered):

This query retrieves the best-selling product based on the total quantity ordered across all orders. It helps identify which product has the highest sales volume and popularity.

SELECT P.ProductID, P.Name, SUM(OD.Quantity) AS TotalSold

FROM OrderDetails OD

JOIN Products P ON OD.ProductID = P.ProductID

GROUP BY P.ProductID, P.Name

ORDER BY TotalSold DESC

FETCH FIRST 1 ROW ONLY;

```
SELECT P.ProductID, P.Name, SUM(OD.Quantity) AS TotalSold
FROM OrderDetails OD
JOIN Products P ON OD.ProductID = P.ProductID
GROUP BY P.ProductID, P.Name
ORDER BY TotalSold DESC
FETCH FIRST 1 ROW ONLY;
```

PRODUCTID	NAME	TOTALSOLD
P004	Volume Mascara	10

16.Display cities that have more than one customer (using Address table):

This query displays cities that have more than one customer, using data from the Address table. It helps understand the geographic distribution of customers and identify high-density areas.

SELECT City, COUNT(DISTINCT CustomerID) AS CustomerCount

FROM Address

GROUP BY City

HAVING COUNT(DISTINCT CustomerID) > 1;

```
SELECT City, COUNT(DISTINCT CustomerID) AS CustomerCount
FROM Address
GROUP BY City
HAVING COUNT(DISTINCT CustomerID) > 1;
```

CITY	CUSTOMERCOUNT
Abha	2
Dammam	2

17.List customers who placed more than one order, with order count:

This query lists customers who have placed more than one order, along with their order count. It is useful for identifying active customers and assessing their loyalty.

SELECT C.CustomerID, C.FirstName || ' ' || C.LastName AS FullName,

    COUNT(O.OrderID) AS OrderCount

FROM Customers C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.CustomerID, C.FirstName, C.LastName

HAVING COUNT(O.OrderID) > 1;

```
SELECT C.CustomerID, C.FirstName || ' ' || C.LastName AS FullName,
       COUNT(O.OrderID) AS OrderCount
FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.CustomerID, C.FirstName, C.LastName
HAVING COUNT(O.OrderID) > 1;
```

CUSTOMERID	FULLNAME	ORDERCOUNT
1	Alaa Ahmad	2
3	Emily Smith	2
15	Emily Johnson	2
25	Samiah Al-Malek	3
6	Sara Ibrahim	2
27	Maya Khoury	2
9	Dana Hassan	2
18	Lina Haddad	2
13	Zainab Adel	2
28	Hessah Al-Njar	2

18.Total payments made per payment method:

This query calculates the total payments made using each payment method. It helps analyze customer payment preferences and supports accurate financial and operational decision-making.

SELECT PaymentMethod, SUM(Amount) AS TotalPaid

FROM Payments

GROUP BY PaymentMethod;

```
SELECT PaymentMethod, SUM(Amount) AS TotalPaid
FROM Payments
GROUP BY PaymentMethod;
```

PAYMENTMETHOD	TOTALPAID
PAYPAL	1635.9
APPLE PAY	3639.95
VISA	3284.25

19. Top 3 Highest-Spending Customers:

This query finds the top 3 customers based on how many orders they placed.
If two or more customers placed the same number of orders, it compares total money spent.

SELECT TOP 3

c.CustomerID, c.FirstName + ' ' + c.LastName AS FullName,

COUNT(o.OrderID) AS NumberOfOrders,

SUM(o.TotalAmount) AS TotalSpent

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName

ORDER BY COUNT(o.OrderID) DESC, SUM(o.TotalAmount) DESC;

```
SELECT TOP 3
c.CustomerID, c.FirstName + ' ' + c.LastName AS FullName,
COUNT(o.OrderID) AS NumberOfOrders,
SUM(o.TotalAmount) AS TotalSpent
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.FirstName, c.LastName
ORDER BY COUNT(o.OrderID) DESC, SUM(o.TotalAmount) DESC;
```

	CustomerID	FullName	NumberOfOrders	TotalSpent
1	25	Samiah Al-Malek	3	806.10
2	6	Sara Ibrahim	2	827.75
3	28	Hessah Al-Njar	2	677.65

20. Annual Income and Orders Summary:

This query calculates how many orders were placed in each year.
It also sums up the total income (TotalAmount) for each year.
The result is sorted from the highest to the lowest income year.

SELECT

YEAR(OrderDate) AS OrderYear,

COUNT(OrderID) AS TotalOrders,

SUM(TotalAmount) AS TotalIncome

FROM Orders

GROUP BY YEAR(OrderDate)

ORDER BY TotalIncome DESC;

```sql
SELECT
    YEAR(OrderDate) AS OrderYear,
    COUNT(OrderID) AS TotalOrders,
    SUM(TotalAmount) AS TotalIncome
  FROM Orders
  GROUP BY YEAR(OrderDate)
  ORDER BY TotalIncome DESC;
```

	OrderYear	TotalOrders	TotalIncome
1	2023	9	2500.00
2	2025	7	1868.70
3	2024	7	1612.30
4	2021	5	1531.40
5	2022	3	1047.80

21. Most Ordered Products:

This query shows how many times each product was ordered across all orders.
It also sums the total quantity sold for each product.

SELECT

p.ProductID,

p.Name AS ProductName,

COUNT(DISTINCT od.OrderID) AS TimesOrdered,

SUM(od.Quantity) AS TotalQuantitySold

FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductID, p.Name

ORDER BY TotalQuantitySold DESC;

```sql
SELECT
    p.ProductID,
    p.Name AS ProductName,
    COUNT(DISTINCT od.OrderID) AS TimesOrdered,
    SUM(od.Quantity) AS TotalQuantitySold
FROM Products p
JOIN OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductID, p.Name
ORDER BY TotalQuantitySold DESC;
```

	ProductID	ProductName	TimesOrdered	TotalQuantitySold
1	P004	Volume Mascara	4	10
2	P001	Matte Foundation	7	9
3	P014	Precision Brush Set	6	8
4	P012	Sculpting Bronzer	3	7
5	P010	Rosy Blush Compact	4	7
6	P013	Blending Sponge	2	6
7	P009	Lip Liner Duo	2	5
8	P007	Velvet Lipstick	4	5
9	P002	Hydrating Primer	4	5
10	P003	Conceal Pro Stick	3	4
11	P008	Glossy Lip Shine	1	4
12	P011	Golden Highlighter	2	4
13	P015	Micellar Water Cleanser	3	4
14	P005	Bold Eyeshadow Pale...	3	3
15	P006	Brow Sculpt Gel	2	2

22. Customers Who Registered but Never Ordered:

This query finds customers who signed up but never placed any order.

SELECT

c.CustomerID,

c.FirstName + ' ' + c.LastName AS FullName

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE o.OrderID IS NULL;

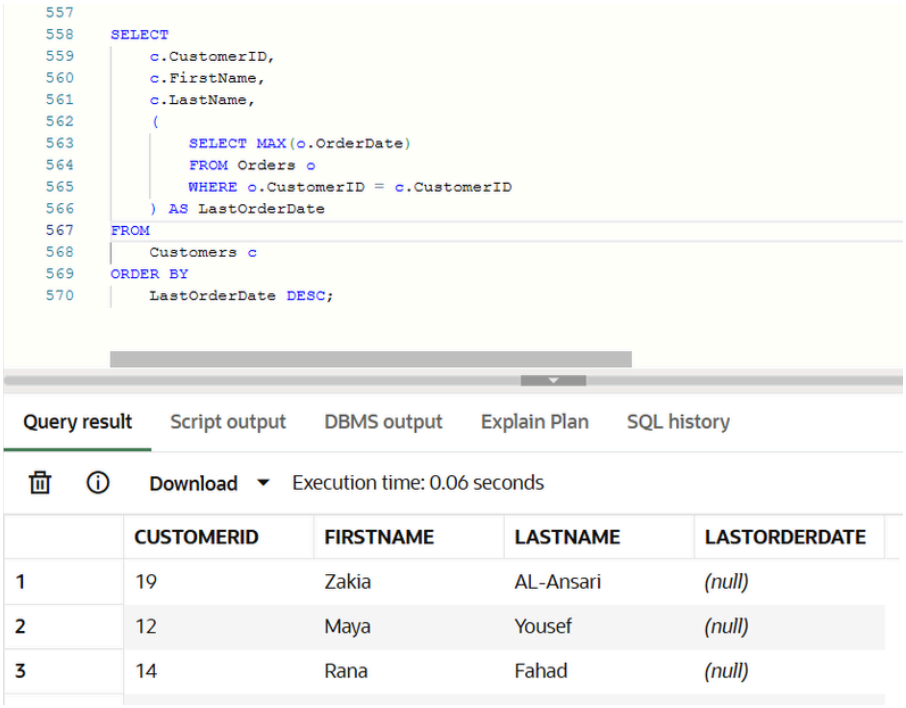
```sql
SELECT
    c.CustomerID,
    c.FirstName + ' ' + c.LastName AS FullName
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.OrderID IS NULL;
```

	CustomerID	FullName
1	12	Maya Yousef
2	14	Rana Fahad
3	17	Sophie Dubois
4	19	Zakia AL-Ansari
5	23	Najla Al-Zahrani
6	24	Mayar Al-Mubark
7	4	Noura Saleh
8	5	Aisha Ali

23.Last Order Date for Each Customer

```
SELECT
,c.CustomerID
,c.FirstName
,c.LastName
(
```

```
SELECT MAX(o .OrderDate)
FROM Orders o
WHERE o.CustomerID = c .CustomerID
AS LastOrderDate )
FROM
Customers c
ORDER BY
LastOrderDate DESC;
```



```
557
558 SELECT
559     c.CustomerID,
560     c.FirstName,
561     c.LastName,
562     (
563         SELECT MAX(o.OrderDate)
564         FROM Orders o
565         WHERE o.CustomerID = c.CustomerID
566     ) AS LastOrderDate
567 FROM
568     Customers c
569 ORDER BY
570     LastOrderDate DESC;
```

Query result | Script output | DBMS output | Explain Plan | SQL history

Download ▾ Execution time: 0.06 seconds

	CUSTOMERID	FIRSTNAME	LASTNAME	LASTORDERDATE
1	19	Zakia	AL-Ansari	*(null)*
2	12	Maya	Yousef	*(null)*
3	14	Rana	Fahad	*(null)*

Last Order Date for Each Customer: This query retrieves each customer's ID, first name, and last name, along with the date of their most recent order. It helps identify active and inactive customers based on their last purchase

24. Products Never Added to Any Wishlist

```
SELECT
p .ProductID,
p.Name
FROM
Products p
LEFT JOIN
Wishlist w ON p.ProductID = w.ProductID
WHERE
w.ProductID IS NULL;
```

```
572
573     SELECT
574     p.ProductID,
575     p.Name
576 FROM
577     Products p
578 LEFT JOIN
579     Wishlist w ON p.ProductID = w.ProductID
580 WHERE
581     w.ProductID IS NULL;
```

Query result | Script output | DBMS output | Explain

Download | Execution time: 0.009 second

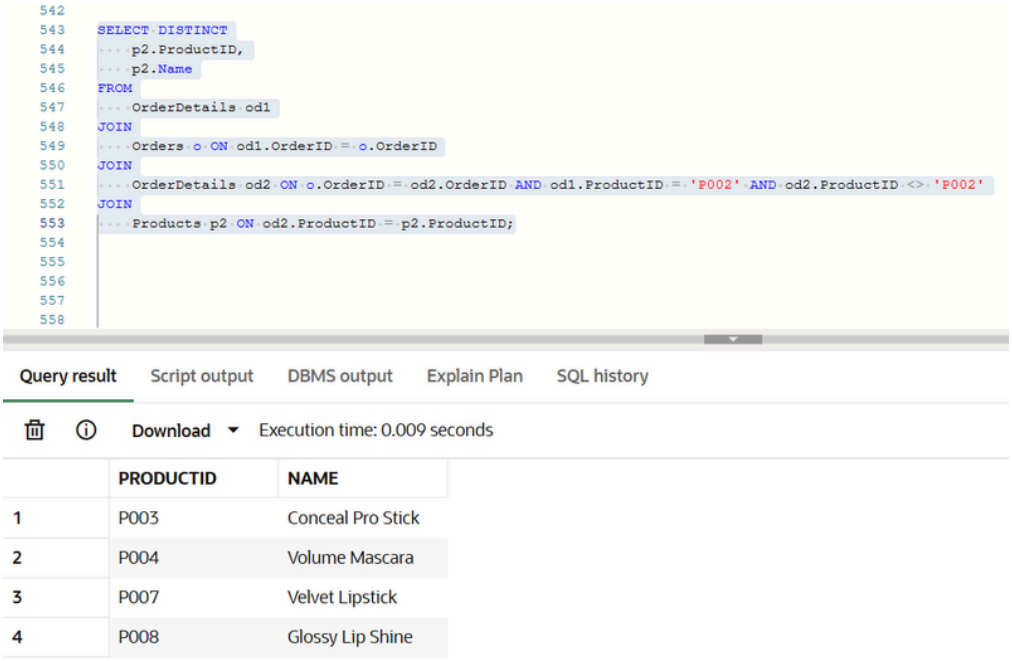
	PRODUCTID	NAME
1	P010	Rosy Blush Compact
2	P011	Golden Highlighter
3	P012	Sculpting Bronzer
4	P013	Blending Sponge
5	P014	Precision Brush Set
6	P015	Micellar Water Clean

Products Never Added to Any Wishlist: This query lists the product ID and name of all products that have never been added to any customer's wishlist. .This can highlight potentially unpopular or overlooked items in the catalog

25. Identifying Co-Purchased Products

```
SELECT DISTINCT
p2.ProductID,
p2.Name
FROM
```

FROM
OrderDetails od1
JOIN
Orders o ON od1.OrderID = o.OrderID
 JOIN
 OrderDetails od2 ON o.OrderID = od2.OrderID AND
'od1.ProductID = 'P002' AND od2.ProductID <> 'P002
JOIN
Products p2 ON od2.ProductID = p2.ProductID;



```
SELECT DISTINCT
    p2.ProductID,
    p2.Name
FROM
    OrderDetails od1
JOIN
    Orders o ON od1.OrderID = o.OrderID
JOIN
    OrderDetails od2 ON o.OrderID = od2.OrderID AND od1.ProductID = 'P002' AND od2.ProductID <> 'P002'
JOIN
    Products p2 ON od2.ProductID = p2.ProductID;
```

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.009 seconds

	PRODUCTID	NAME
1	P003	Conceal Pro Stick
2	P004	Volume Mascara
3	P007	Velvet Lipstick
4	P008	Glossy Lip Shine

Identifying Co-Purchased Products: This query analyzes order data to find products that are often bought together in the same customer orders. It helps understand which items are frequently associated with each other in purchasing .behavior, which can be useful for recommendations and marketing

Views

Unsimple views:

1- Product Reviews with Customer

To display product reviews with customer data and analyze opinions (positive/negative/neutral), which is useful for assessing customer satisfaction

---

```
CREATE VIEW ProductReviewsWithCustomer AS
SELECT
pr.ReviewID,
pr.ProductID,
 p.Name AS ProductName,
 pr.Rating,
 pr.ReviewComment,
 pr.CustomerID,
 c.FirstName,
 c.LastName,
 CASE
 WHEN pr.Rating >= 4 THEN 'Positive'
 WHEN pr.Rating <= 2 THEN 'Negative'
 ELSE 'Neutral'
 END AS Sentiment
FROM ProductReviews pr
JOIN Products p ON pr.ProductID = p.ProductID
```

JOIN Customers c ON pr.CustomerID = c.CustomerID;

```
  CREATE VIEW ProductReviewsWithCustomer AS
SELECT
    pr.ReviewID,
    pr.ProductID,
    p.Name AS ProductName,
    pr.Rating,
    pr.ReviewComment,
    pr.CustomerID,
    c.FirstName,
    c.LastName,
    CASE
        WHEN pr.Rating >= 4 THEN 'Positive'
        WHEN pr.Rating <= 2 THEN 'Negative'
        ELSE 'Neutral'
    END AS Sentiment
FROM
    ProductReviews pr
JOIN
    Products p ON pr.ProductID = p.ProductID
JOIN
    Customers c ON pr.CustomerID = c.CustomerID;
```

	ReviewID	ProductID	ProductName	Rating	ReviewComment	CustomerID	FirstName	LastName	Sentiment
1	R001	P013	Blending Sponge	1.5	absolutely useless.	6	Sara	Ibrahim	Negative
2	R002	P010	Rosy Blush Compact	5.0	Love love this product!! Long lasting no transfer!!	1	Alaa	Ahmad	Positive
3	R003	P006	Brow Sculpt Gel	2.0	Did no work for me at all.	11	Lina	Ibrahim	Negative
4	R004	P002	Hydrating Primer	3.6	love the primer itself but not the biggest fan of t...	18	Lina	Haddad	Neutral
5	R005	P001	Matte Foundation	4.7	My go-to foundation that I repurchase time and...	25	Samiah	Al-Malek	Positive
6	R006	P005	Bold Eyeshadow Palette	5.0	Perfect, loved this palette, been thinking of buyi...	25	Samiah	Al-Malek	Positive

Derived View:

This view is for ProductReviewsWithCustomer, and only displays reviews with a rating of 4 or higher, and can be used in storefront or marketing.

CREATE VIEW PositiveReviewsOnly AS

SELECT *

FROM ProductReviewsWithCustomer

WHERE Sentiment = 'Positive';

```
CREATE VIEW PositiveReviewsOnly AS
SELECT *
FROM ProductReviewsWithCustomer
WHERE Sentiment = 'Positive';
```

	ReviewID	ProductID	ProductName	Rating	ReviewComment	CustomerID	FirstName	LastName	Sentiment
1	R002	P010	Rosy Blush Compact	5.0	Love love this product!! Long lasting no transfer!!	1	Alaa	Ahmad	Positive
2	R005	P001	Matte Foundation	4.7	My go-to foundation that I repurchase time and time again	25	Samiah	Al-Malek	Positive
3	R006	P005	Bold Eyeshadow Palette	5.0	Perfect, loved this palette, been thinking of buying this for a long time.	25	Samiah	Al-Malek	Positive

2- Customer Order Details View

This view provides a comprehensive summary of customer orders, including personal details such as the customer's name, email, and phone number. It also includes order-specific information like order IDs, dates, product details, quantities, prices, and total amounts. The view is created by joining data from multiple tables: Customers, Address, Orders, OrderDetails, and Products.

```
CREATE VIEW CustomerOrderDetails AS
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    c.Email,
    c.Phone,
    a.City,
    o.OrderID,
    o.OrderDate,
    od.ProductID,
    p.Name,
    od.Quantity,
    od.Price,
    o.TotalAmount
FROM Customers c
JOIN Address a ON c.CustomerID = a.CustomerID
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN OrderDetails od ON o.OrderID = od.OrderID
```

```
JOIN Products p ON od.ProductID = p.ProductID;
```

```sql
 CREATE VIEW CustomerOrderDetails AS
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    c.Email,
    c.Phone,
    a.City,
    o.OrderID,
    o.OrderDate,
    od.ProductID,
    p.Name,
    od.Quantity,
    od.Price,
    o.TotalAmount
FROM Customers c
JOIN Address a ON c.CustomerID = a.CustomerID
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN OrderDetails od ON o.OrderID = od.OrderID
JOIN Products p ON od.ProductID = p.ProductID;
```

21	18-APR-25	258.3
21	18-APR-25	258.3
26	03-JAN-25	255
26	03-JAN-25	255
28	20-APR-25	253.65
25	09-FEB-24	250
8	01-SEP-24	250
24	22-AUG-23	246
24	22-AUG-23	246
24	22-AUG-23	246

12	08-OCT-23	297
16	17-OCT-23	287
16	17-OCT-23	287
15	16-MAR-25	282.15
19	27-MAY-23	277
19	27-MAY-23	277
11	01-MAY-21	267.75
11	01-MAY-21	267.75
7	25-JUL-22	267.3
7	25-JUL-22	267.3

ORDERID	ORDERDATE	TOTALAMOUNT
29	28-JUN-23	500
23	09-DEC-21	424
23	09-DEC-21	424
20	03-OCT-22	407
20	03-OCT-22	407
20	03-OCT-22	407
9	02-NOV-22	373.5
9	02-NOV-22	373.5
14	23-DEC-23	350
21	18-APR-25	258.3
21	18-APR-25	258.3
26	03-JAN-25	255
26	03-JAN-25	255
28	20-APR-25	253.65
25	09-FEB-24	250
8	01-SEP-24	250
24	22-AUG-23	246
24	22-AUG-23	246
24	22-AUG-23	246

30	20-JAN-25	245.1
30	20-JAN-25	245.1
22	10-APR-25	237.5
4	28-FEB-23	225
5	20-DEC-24	192.1
5	20-DEC-24	192.1
2	10-SEP-21	178
27	01-AUG-24	159.1
10	28-DEC-23	159
6	10-JAN-23	159

30	20-JAN-25	245.1
22	10-APR-25	237.5
4	28-FEB-23	225
5	20-DEC-24	192.1
5	20-DEC-24	192.1
2	10-SEP-21	178
27	01-AUG-24	159.1
10	28-DEC-23	159
6	10-JAN-23	159
31	19-SEP-24	149.1

This query retrieves all the data from the CustomerOrderDetails view for customers residing in the city of Riyadh. It filters the results to include only orders placed by customers whose address contains "Riyadh" as the city.

```
select *
    from CustomerOrderDetails
    where City = 'Riyadh';
```

CUSTOMERID	FIRSTNAME	LASTNAME	EMAIL	PHONE	CITY	ORDERID	ORDERDATE	PRODUCTID	NAME	QUANTITY	PRICE	TOTALAMOUNT
2	Layla	Abdullah	layla@example.com	0502345678	Riyadh	1	12-APR-25	P001	Matte Foundation	1	159	337
2	Layla	Abdullah	layla@example.com	0502345678	Riyadh	1	12-APR-25	P007	Velvet Lipstick	2	178	337

3-**Unpurchased Wishlist Products**

Shows wishlist items that customers haven't bought yet, including those who've never placed an order. This view helps target these customers with emails—like discount offers or low-stock alerts—to encourage them to complete a purchase

---

```
create view Customers_Wishlist as

select c.customerid, c.FirstName, c.email, p.productid, p.name as ProductName,
cat.name as CategoryName

from customers c join wishlist w on c.customerid = w.customerid

join products p on p.productid =  w.productid

join categories cat on cat.categoryid = p.categoryid

 where not exists (select * from orders o natural join orderdetails

where o.customerid = c.customerid

and orderdetails.productid = p.productid   );
```

```
Create view Customer_Wishlist as
select c.Customerid, c.FirstName, c.Email, p.Productid, p.Name as ProductName, cat.Name as CategoryName
from customers c join wishlist w on c.customerid = w.customerid
join products p on p.productid = w.productid
join categories cat on cat.categoryid = p.categoryid
where not exists (
    select * from orders o
    natural join orderdetails
    where o.customerid = c.customerid
    and orderdetails.productid = p.productid );
```

CUSTOMERID	FIRSTNAME	EMAIL	PRODUCTID	PRODUCTNAME	CATEGORYNAME
2	Layla	layla@example.com	P009	Lip Liner Duo	Lips
28	Hessah	imhessah@gmail.com	P002	Hydrating Primer	Face
18	Lina	linahaddad_5@example.com	P009	Lip Liner Duo	Lips
1	Alaa	fatima@example.com	P006	Brow Sculpt Gel	Eyes
14	Rana	rana_569r@example.com	P005	Bold Eyeshadow Palette	Eyes
7	Huda	huda@example.com	P006	Brow Sculpt Gel	Eyes
1	Alaa	fatima@example.com	P004	Volume Mascara	Eyes
20	Noura	noura.qahtani@gmail.com	P007	Velvet Lipstick	Lips
25	Samiah	sam4566@gmail.com	P003	Conceal Pro Stick	Face
6	Sara	sara@example.com	P002	Hydrating Primer	Face
7	Huda	huda@example.com	P008	Glossy Lip Shine	Lips
11	Lina	lina@example.com	P007	Velvet Lipstick	Lips
27	Maya	mmm_76683@gmail.com	P001	Matte Foundation	Face
23	Najla	njlaaa10@gmail.com	P009	Lip Liner Duo	Lips
20	Noura	noura.qahtani@gmail.com	P005	Bold Eyeshadow Palette	Eyes
6	Sara	sara@example.com	P008	Glossy Lip Shine	Lips
19	Zakia	zak.220h@example.com	P003	Conceal Pro Stick	Face
16	Aisha	aisha.khan@gmail.com	P006	Brow Sculpt Gel	Eyes
27	Maya	mmm_76683@gmail.com	P005	Bold Eyeshadow Palette	Eyes
18	Lina	linahaddad_5@example.com	P001	Matte Foundation	Face

This query retrieves wishlist products in the "Lips" category that customers haven't purchased yet :

---

select distinct productid, productname

from Customer_Wishlist

where CategoryName = 'Lips';

```
select  distinct productid, productname
from Customer_Wishlist
where CategoryName = 'Lips';
```

PRODUCTID	PRODUCTNAME
P007	Velvet Lipstick
P008	Glossy Lip Shine
P009	Lip Liner Duo

Simple view:

1- Low In Stock

For products with less than 30 items in stock, the warehouse department helps determine which products need to be reordered.

CREATE VIEW LowInStock AS

SELECT ProductID, Name, StockQuantity,Price

FROM Products WHERE StockQuantity < 30

WITH CHECK OPTION;

```
CREATE VIEW LowInStock
AS
SELECT ProductID, Name, StockQuantity, Price
FROM Products
WHERE StockQuantity < 30
WITH CHECK OPTION;
```

	ProductID	Name	StockQuantity	Price
1	P012	Sculpting Bronzer	28	99.00
2	P014	Precision Brush Set	22	250.00

- **Update simple view:**

In this update, product number P012 will be removed from View LowInStock and will have an in-stock value above 30.

UPDATE Products

SET StockQuantity = 50

WHERE ProductID = 'P012';

```
  UPDATE Products
SET StockQuantity = 50
WHERE ProductID = 'P012';
```

	ProductID	Name	StockQuantity	Price
1	P014	Precision Brush Set	22	250.00

- **Insert into simple view:**

When we tried to enter data that violated the condition, the transaction was rejected, which reflects the effectiveness of using WITH CHECK OPTION.

```
INSERT INTO LowInStock (ProductID, Name, StockQuantity, Price)
VALUES ('P016', 'Glow Setting Spray', 35, 95.00);
```

Msg 550, Level 16, State 1, Line 911 The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that specifies WITH CHECK OPTION and one or more rows resulting from the operation did not qualify under the CHECK OPTION constraint.

2- Processing Orders

To view orders that have a status of (in process) to facilitate tracking of orders that have not yet been executed.

CREATE VIEW ProcessingOrders AS

SELECT OrderID, CustomerID, OrderDate, TotalAmount

FROM Orders

WHERE UPPER(OrderStatus) = 'PROCESSING';

```sql
CREATE VIEW ProcessingOrders AS
SELECT OrderID, CustomerID, OrderDate, TotalAmount
FROM Orders
WHERE UPPER(OrderStatus) = 'PROCESSING';
```

	OrderID	CustomerID	OrderDate	TotalAmount
1	11	8	2021-05-01	267.75
2	14	18	2023-12-23	350.00
3	17	15	2024-09-01	311.20
4	2	3	2021-09-10	178.00
5	21	22	2025-04-18	258.30
6	24	27	2023-08-22	246.00
7	26	27	2025-01-03	255.00
8	28	28	2025-04-20	253.65
9	5	1	2024-12-20	192.10
10	8	9	2024-09-01	250.00

Normalization

1- Orders table

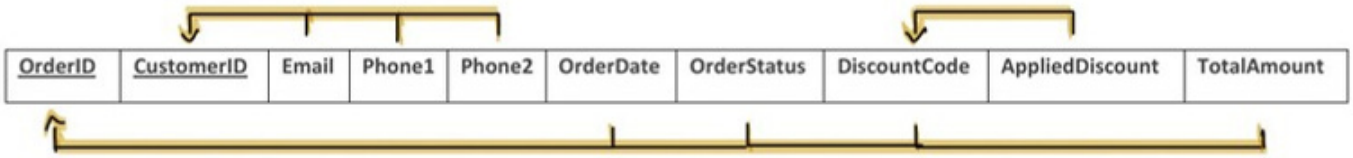
-Before normalization:

OrderID	CustomerID	Email	Phone	OrderDate	OrderStatus	DiscountCode	AppliedDiscount	TotalAmount
1	2	layla@example.com	0502345678, 0578559011	2025-04-12	SHIPPED	TENDEAL	10%	337.00
2	3	emily@example.com	0553456789	2021-09-10	PROCESSING	SAVE20	20%	178.00
3	6	sara@example.com	0556789012	2021-06-12	DELIVERED	FIVEOFF	5%	327.75
4	9	dana@example.com	0558901235, 0549812213	2024-09-01	PROCESSING	-	-	250.00
5	11	lina@example.com	0500123457	2024-09-05	DELIVERED	SAVE20	20%	300.80
6	18	linahaddad_5@example.com	3106993644, 3109663205	2023-12-23	PROCESSING	DROP30	30%	350.00
7	21	faisalharbi@example.com	054876543	2021-07-11	SHIPPED	DROP30	30%	333.90
8	18	linahaddad_5@example.com	3106993644	2023-05-27	DELIVERED	-	-	277.00
9	25	sam4566@gmail.com	0595411730	2022-10-03	DELIVERED	-	-	407.00
10	22	hadeell09@gmail.com	-	2025-04-18	PROCESSING	TENDEAL	10%	258.30
11	26	serien45_1@gmail.com	03924182, 03884109	2025-04-10	SHIPPED	TENDEAL	10%	237.50
12	28	imhessah@gmail.com	03512994	2021-12-09	CANCELLED	-	-	424.00
13	27	mmm_76683@gmail.com	03771952	2023-08-22	PROCESSING	FIVEOFF	5%	246.00
14	25	sam4566@gmail.com	0595411730	2024-02-09	CANCELLED	DROP30	30%	250.00
15	27	mmm_76683@gmail.com	03771952	2025-01-03	PROCESSING	-	-	255.00
16	13	28772@gmail.com	0502345679, 0505432215	2024-08-01	SHIPPED	SAVE20	20%	159.10
17	28	imhessah@gmail.com	03512994, 03668910	2025-04-20	PROCESSING	FIVEOFF	5%	253.65
18	6	sara@example.com	0556789012	2023-06-28	CANCELLED	-	-	500.00
19	15	emily.12@example.com	2125600388	2025-01-20	DELIVERED	SAVE20	20%	245.10

-After 1NF:

OrderID	CustomerID	Email	Phone1	Phone2	OrderDate	OrderStatus	DiscountCode	AppliedDiscount	TotalAmount
1	2	layla@example.com	0502345678	0578559011	2025-04-12	SHIPPED	TENDEAL	10%	337.00
2	3	emily@example.com	0553456789		2021-09-10	PROCESSING	SAVE20	20%	178.00
3	6	sara@example.com	0556789012		2021-06-12	DELIVERED	FIVEOFF	5%	327.75
4	9	dana@example.com	0558901235	0549812213	2024-09-01	PROCESSING	-	-	250.00
5	11	lina@example.com	0500123457		2024-09-05	DELIVERED	SAVE20	20%	300.80
6	18	linahaddad_5@example.com	3106993644	3109663205	2023-12-23	PROCESSING	DROP30	30%	350.00
7	21	faisalharbi@example.com	054876543		2021-07-11	SHIPPED	DROP30	30%	333.90
8	18	linahaddad_5@example.com	3106993644		2023-05-27	DELIVERED	-	-	277.00
9	25	sam4566@gmail.com	0595411730		2022-10-03	DELIVERED	-	-	407.00
10	22	hadeel09@gmail.com	-		2025-04-18	PROCESSING	TENDEAL	10%	258.30
11	26	serien45_1@gmail.com	03924182	03884109	2025-04-10	SHIPPED	TENDEAL	10%	237.50
12	28	imhessah@gmail.com	03512994		2021-12-09	CANCELLED	-	-	424.00
13	27	mmm_76683@gmail.com	03771952		2023-08-22	PROCESSING	FIVEOFF	5%	246.00
14	25	sam4566@gmail.com	0595411730		2024-02-09	CANCELLED	DROP30	30%	250.00
15	27	mmm_76683@gmail.com	03771952		2025-01-03	PROCESSING	-	-	255.00
16	13	28772@gmail.com	0502345679	0505432215	2024-08-01	SHIPPED	SAVE20	20%	159.10
17	28	imhessah@gmail.com	03512994	03668910	2025-04-20	PROCESSING	FIVEOFF	5%	253.65
18	6	sara@example.com	0556789012		2023-06-28	CANCELLED	-	-	500.00
19	15	emily.12@example.com	2125600388		2025-01-20	DELIVERED	SAVE20	20%	245.10

- [ ] **Multi values have been removed.**

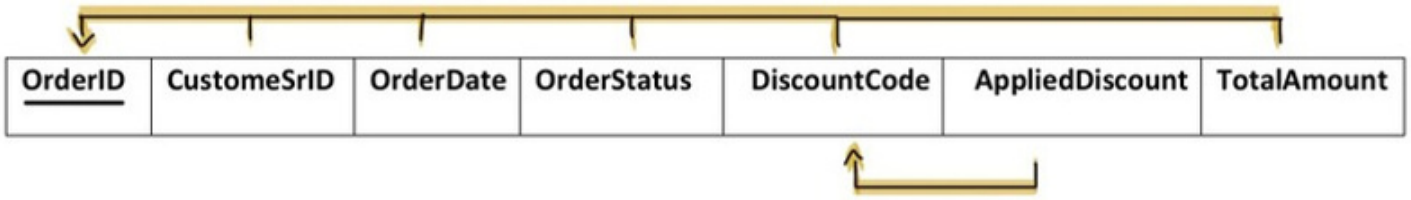


-After 2NF:

CustomerID	Email	Phone1	Phone2
2	layla@example.com	0502345678	0578559011
3	emily@example.com	0553456789	
6	sara@example.com	0556789012	
9	dana@example.com	0558901235	0549812213
11	lina@example.com	0500123457	
18	linahaddad_5@example.com	3106993644	3109663205
21	faisalharbi@example.com	054876543	
18	linahaddad_5@example.com	3106993644	
25	sam4566@gmail.com	0595411730	
22	hadeell09@gmail.com	-	
26	serien45_1@gmail.com	03924182	03884109
28	imhessah@gmail.com	03512994	
27	mmm_76683@gmail.com	03771952	
25	sam4566@gmail.com	0595411730	
27	mmm_76683@gmail.com	03771952	
13	28772@gmail.com	0502345679	0505432215
28	imhessah@gmail.com	03512994	03668910
6	sara@example.com	0556789012	
15	emily.12@example.com	2125600388	

OrderID	CustomerID	OrderDate	OrderStatus	DiscountCode	AppliedDiscount	TotalAmount
1	2	2025-04-12	SHIPPED	TENDEAL	10%	337.00
2	3	2021-09-10	PROCESSING	SAVE20	20%	178.00
3	6	2021-06-12	DELIVERED	FIVEOFF	5%	327.75
4	9	2024-09-01	PROCESSING	-	-	250.00
5	11	2024-09-05	DELIVERED	SAVE20	20%	300.80
6	18	2023-12-23	PROCESSING	DROP30	30%	350.00
7	21	2021-07-11	SHIPPED	DROP30	30%	333.90
8	18	2023-05-27	DELIVERED	-	-	277.00
9	25	2022-10-03	DELIVERED	-	-	407.00
10	22	2025-04-18	PROCESSING	TENDEAL	10%	258.30
11	26	2025-04-10	SHIPPED	TENDEAL	10%	237.50
12	28	2021-12-09	CANCELLED	-	-	424.00
13	27	2023-08-22	PROCESSING	FIVEOFF	5%	246.00
14	25	2024-02-09	CANCELLED	DROP30	30%	250.00
15	27	2025-01-03	PROCESSING	-	-	255.00
16	13	2024-08-01	SHIPPED	SAVE20	20%	159.10
17	28	2025-04-20	PROCESSING	FIVEOFF	5%	253.65
18	6	2023-06-28	CANCELLED	-	-	500.00
19	15	2025-01-20	DELIVERED	SAVE20	20%	245.10

- [ ] **Partial dependency has been removed.**



-After 3NF:

OrderID	CustomerID	OrderDate	OrderStatus	CodeID	TotalAmount
1	2	2025-04-12	SHIPPED	2	337.00
2	3	2021-09-10	PROCESSING	3	178.00
3	6	2021-06-12	DELIVERED	1	327.75
4	9	2024-09-01	PROCESSING	-	250.00
5	11	2024-09-05	DELIVERED	3	300.80
6	18	2023-12-23	PROCESSING	4	350.00
7	21	2021-07-11	SHIPPED	4	333.90
8	18	2023-05-27	DELIVERED	-	277.00
9	25	2022-10-03	DELIVERED	-	407.00
10	22	2025-04-18	PROCESSING	2	258.30
11	26	2025-04-10	SHIPPED	2	237.50
12	28	2021-12-09	CANCELLED	-	424.00
13	27	2023-08-22	PROCESSING	1	246.00
14	25	2024-02-09	CANCELLED	4	250.00
15	27	2025-01-03	PROCESSING	-	255.00
16	13	2024-08-01	SHIPPED	3	159.10
17	28	2025-04-20	PROCESSING	1	253.65
18	6	2023-06-28	CANCELLED	-	500.00
19	15	2025-01-20	DELIVERED	3	245.10

CodeID	DiscountCode	AppliedDiscount
1	FIVEOFF	5%
2	TENDEAL	10%
3	SAVE20	20%
4	DROP30	30%
5	15OFF	15%

- [ ] **Transitive dependency has been removed.**

2- Address table

-Before normalization:

AddressID	CustomerID	AddressLine1	City	State	PostalCode
A001	2	King Saud Street, Near Riyadh Park	Riyadh	Riyadh	12345
A002	3	Sunset Blvd	la	USA	-
A003	6	Imam Mohammed bin saud street	Abha	SA	62521
A004	7	Prince Mohammed Bin Fahd Street, Near Corniche	Dammam	SA	34567
A005	1	Al-Shati Street, Near Marina Mall	Khobar	SA	45678
A006	9	Al Olaya Street, Near Kingdom Tower	Tabuk	SA	56789
A007	10	King Khalid Road, Opposite Abha Mall	Abha	SA	67890
A008	13	Close to Al Nakheel Plaza	Buraidah	SA	78901
A009	8	Hail Central Street, Next to University	Hail	SA	-
A010	11	Najran Road, Near City Center	Najran	SA	90123
A011	18	123 Main Street	LA	USA	-
A012	20	Sheikh Zayed Road	Dubai	UAE	-
A013	16	45 curtain Road	London	UK	75001
A014	15	789 Broadway	NYC	USA	10003
A015	21	Hamdan street	Abu Dhabi	UAE	-
A016	22	Al-Nahda street	Sharjah	UAE	-
A017	27	Hamra street, 235 building 2nd floor	Beirut	Lebanon	1107
A018	28	Adliya, 3605 street, 13 building	Manamah	Bahrain	336
A019	26	Saida main street, building 45	Sidon	Lebanon	1600
A020	25	Al-Nakheel, 13A street	Dammam	SA	32244

-After 1NF:

AddressID	CustomerID	AddressLine1	AddressLine2	City	State	PostalCode
A001	2	King Saud Street	Near Riyadh Park	Riyadh	Riyadh	12345
A002	3	Sunset Blvd	-	la	USA	-
A003	6	Imam Mohammed bin saud street	-	Abha	SA	62521
A004	7	Prince Mohammed Bin Fahd Street	Near Corniche	Dammam	SA	34567
A005	1	Al-Shati Street	Near Marina Mall	Khobar	SA	45678
A006	9	Al Olaya Street	Near Kingdom Tower	Tabuk	SA	56789
A007	10	King Khalid Road	Opposite Abha Mall	Abha	SA	67890
A008	13	Close to Al Nakheel Plaza	-	Buraidah	SA	78901
A009	8	Hail Central Street	Next to University	Hail	SA	-
A010	11	Najran Road	Near City Center	Najran	SA	90123
A011	18	123 Main Street	-	LA	USA	-
A012	20	Sheikh Zayed Road	-	Dubai	UAE	-
A013	16	45 curtain Road	-	London	UK	75001
A014	15	789 Broadway	-	NYC	USA	10003
A015	21	Hamdan street	-	Abu Dhabi	UAE	-
A016	22	Al-Nahda street	-	Sharjah	UAE	-
A017	27	Hamra street	235 building 2nd floor	Beirut	Lebanon	1107
A018	28	Adliya, 3605 street	13 building	Manamah	Bahrain	336
A019	26	Saida main street	building 45	Sidon	Lebanon	1600
A020	25	Al-Nakheel, 13A street	-	Dammam	SA	32244

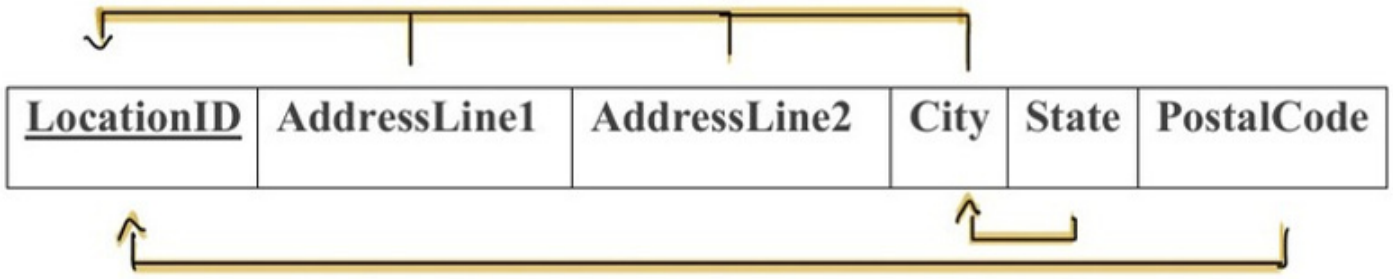
- [ ] **Multi values have been removed.**

-After 2NF:

AddressID	CustomerID	LocationID
A001	2	1
A002	3	2
A003	6	3
A004	7	4
A005	1	5
A006	9	6
A007	10	7
A008	13	8
A009	8	9
A010	11	10
A011	18	11
A012	20	12
A013	16	13
A014	15	14
A015	21	15
A016	22	16
A017	27	17
A018	28	18
A019	26	19
A020	25	20

LocationID	AddressLine1	AddressLine2	City	State	PostalCode
1	King Saud Street	Near Riyadh Park	Riyadh	Riyadh	12345
2	Sunset Blvd	-	la	USA	-
3	Imam Mohammed bin saud street	-	Abha	SA	62521
4	Prince Mohammed Bin Fahd Street	Near Corniche	Dammam	SA	34567
5	Al-Shati Street	Near Marina Mall	Khobar	SA	45678
6	Al Olaya Street	Near Kingdom Tower	Tabuk	SA	56789
7	King Khalid Road	Opposite Abha Mall	Abha	SA	67890
8	Close to Al Nakheel Plaza	-	Buraidah	SA	78901
9	Hail Central Street	Next to University	Hail	SA	-
10	Najran Road	Near City Center	Najran	SA	90123
11	123 Main Street	-	LA	USA	-
12	Sheikh Zayed Road	-	Dubai	UAE	-
13	45 curtain Road	-	London	UK	75001
14	789 Broadway	-	NYC	USA	10003
15	Hamdan street	-	Abu Dhabi	UAE	-
16	Al-Nahda street	-	Sharjah	UAE	-
17	Hamra street	235 building 2nd floor	Beirut	Lebanon	1107
18	Adliya, 3605 street	13 building	Manamah	Bahrain	336
19	Saida main street	building 45	Sidon	Lebanon	1600
20	Al-Nakheel, 13A street	-	Dammam	SA	32244

- [ ] **Partial dependency has been removed.**



-After 3NF:

City	State
Riyadh	SA
LA	USA
Abha	SA
Dammam	SA
Khobar	SA
Tabuk	SA
Buraidah	SA
Hail	SA
Najran	SA
Dubai	UAE
London	UK
NYC	USA
Abu Dhabi	UAE
Sharjah	UAE
Beirut	Lebanon
Manamah	Bahrain
Sidon	Lebanon

LocationID	AddressLine1	AddressLine2	City	PostalCode
1	King Saud Street	Near Riyadh Park	Riyadh	12345
2	Sunset Blvd	-	la	-
3	Imam Mohammed bin saud street	-	Abha	62521
4	Prince Mohammed Bin Fahd Street	Near Corniche	Dammam	34567
5	Al-Shati Street	Near Marina Mall	Khobar	45678
6	Al Olaya Street	Near Kingdom Tower	Tabuk	56789
7	King Khalid Road	Opposite Abha Mall	Abha	67890
8	Close to Al Nakheel Plaza	-	Buraidah	78901
9	Hail Central Street	Next to University	Hail	-
10	Najran Road	Near City Center	Najran	90123
11	123 Main Street	-	LA	-
12	Sheikh Zayed Road	-	Dubai	-
13	45 curtain Road	-	London	75001
14	789 Broadway	-	NYC	10003
15	Hamdan street	-	Abu Dhabi	-
16	Al-Nahda street	-	Sharjah	-
17	Hamra street	235 building 2nd floor	Beirut	1107
18	Adliya, 3605 street	13 building	Manamah	336
19	Saida main street	building 45	Sidon	1600
20	Al-Nakheel, 13A street	-	Dammam	32244

- [ ] Transitive dependency has been removed.

Authorization:

We planned the privilege distribution for virtual users in the database, based on their functional roles in the e-commerce store. The goal is to secure the data and prevent unauthorized access or modification.

❗ **Note:**

We were not able to execute the following commands due to the lack of CREATE USER privilege in our current environment. However, the commands are written in the correct SQL syntax and can be executed in any environment that supports user management.

User Creation Commands:

CREATE USER service_user IDENTIFIED BY "SrvUser_2025"
CREATE USER stock_user IDENTIFIED BY "StockUser#25"
CREATE USER manager_user IDENTIFIED BY "Mngr_User2025"
CREATE USER shipping_user IDENTIFIED BY "ShipUser@2025"

Granting Privileges:

//Customer service privileges

GRANT SELECT ON Customers TO service_user;

GRANT SELECT ON Address TO service_user;

// Inventory manager privileges

GRANT SELECT, INSERT, UPDATE ON Products TO stock_user;

//General manager privileges

GRANT ALL PRIVILEGES ON Customers TO manager_user;

GRANT ALL PRIVILEGES ON Orders TO manager_user;

GRANT ALL PRIVILEGES ON Payments TO manager_user;

GRANT ALL PRIVILEGES ON Products TO manager_user;

GRANT ALL PRIVILEGES ON Shipping TO manager_user;

//Shipping staff privileges

GRANT SELECT ON Orders TO shipping_user;

GRANT SELECT, UPDATE ON Shipping TO shipping_user;

Students' Contribution:

Schema	ر
E-R diagram	أ, م
Creation	ش, ر, أ
Insertion	م, ش, غ, ل, ن
Queries	غ, م ،ل, ن, ر, م
Views	ش, غ, أ, ر
Normalization	م, ش
Authorization	أ
Report layout	