Amaan Mohammad

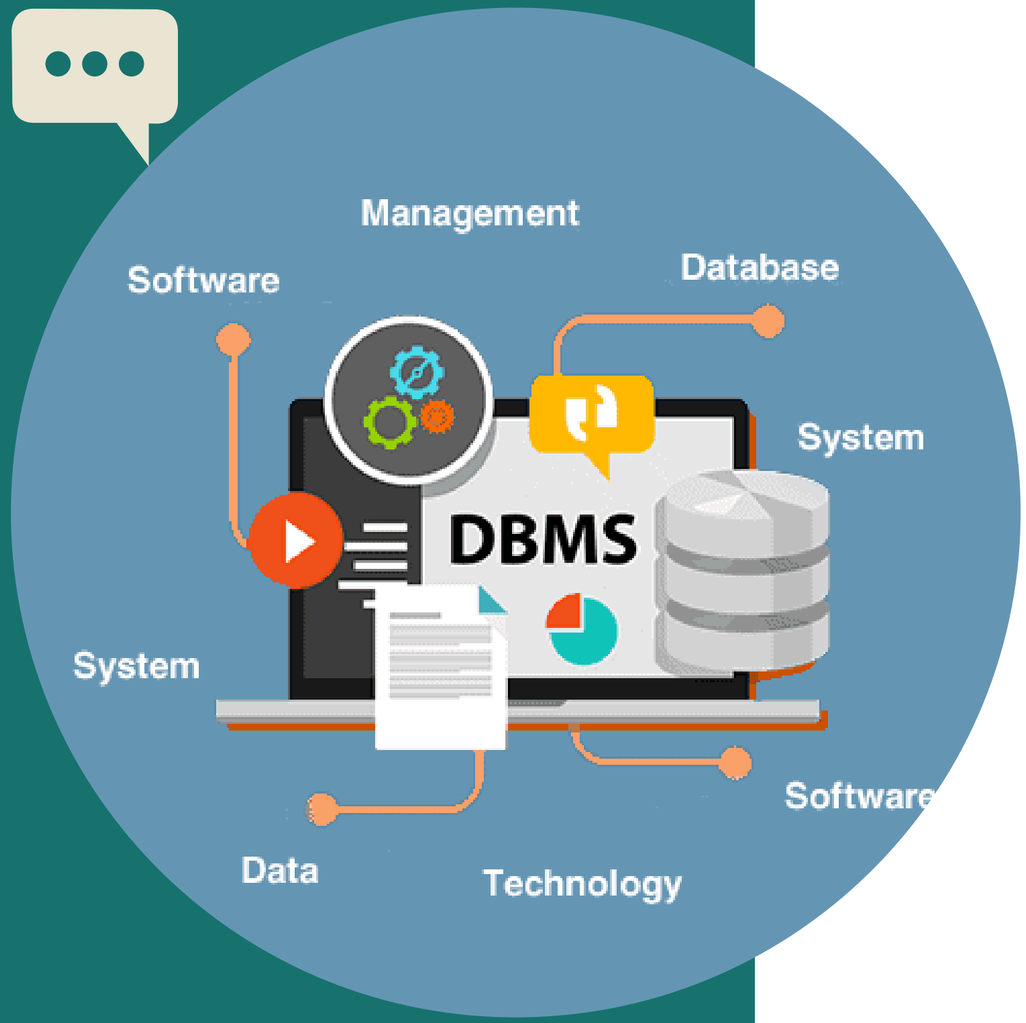
SUPPLY CHAIN MANAGEMENT

PROJECT PRESENTATION

Telephone
+91 7386254308

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

JULY 2024



INTRODUCTION

This project focuses on developing a comprehensive database management system designed to streamline the management of customer orders, product information and shipment details. The system ensures that every package is tracked accurately from the point of origin to its final destination. By implementing a structured and hierarchical data model, we enhance the reliability and accessibility of shipment data, allowing both shipment company and its customers to gather vital information seamlessly

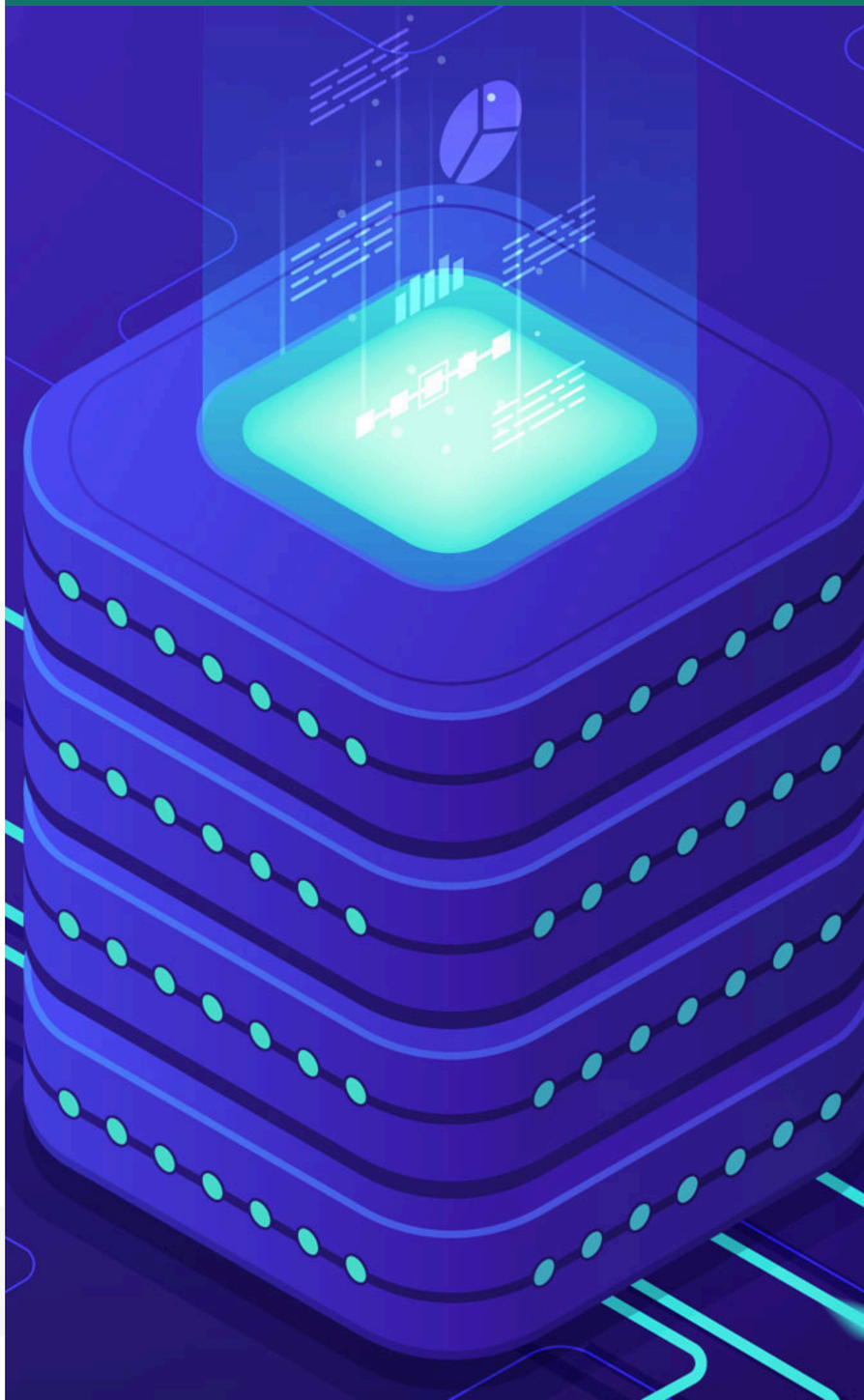
INTRODUCTION

The core of our database management system is built around several interconnected tables, each serving a specific function. Unique identifiers for each record ensure data integrity and streamline the tracking process. With features like package tracking and detailed financial metrics, the system not only improves the operational efficiency but also enhances the customer experience, making it a robust solution for modern shipment companies.

In this system, Customers are linked to their orders, which in turn detail the individual items purchased. Products are categorized into specific categories and departments, facilitating efficient inventory management and sales analysis.

TABLE OF CONTENT

PROJECT PRESENTATION



OBJECTIVE

The main goal of this database management system is to organize and manage all aspects of customer orders, including customer details, order details, product information, categories, and departments. The Objective is to create a robust and efficient system that ensures data storage, easy retrieval, and seamless updates to support business operations

KEY FEATURES:

- Customer Management
- Order Tracking
- Product Management
- Sales and Profit Analysis

TABLES

Stores all the details about the Customers

Customers

ATTRIBUTES	DATA TYPE	CONSTRAINTS
CustomerID	NUMBER	PRIMARY KEY
CustomerFname	VARCHAR(30)	NOT NULL
CustomerLname	VARCHAR(30)	NOT NULL
CustomerSegment	VARCHAR(30)	NOT NULL
CustomerCity	VARCHAR(30)	NOT NULL
CustomerState	VARCHAR(30)	NOT NULL
CustomerCountry	VARCHAR(30)	NOT NULL

TABLES

Stores all the details about the Orders

OrderDetails

ATTRIBUTES	DATA TYPE	CONSTRAINTS
OrderID	NUMBER	PRIMARY KEY
CustomerID	VARCHAR(30)	NOT NULL
OrderDate	VARCHAR(30)	NOT NULL
ShippingDate	VARCHAR(30)	NOT NULL
OrderStatus	VARCHAR(30)	NOT NULL
OrderCity	VARCHAR(30)	NOT NULL
OrderCountry	VARCHAR(30)	NOT NULL
DeliveryStatus	VARCHAR(30)	NOT NULL
Shippingmode	VARCHAR(30)	NOT NULL
Paymentmode	VARCHAR(30)	NOT NULL

TABLES

Stores all the details about the Products

Products

ATTRIBUTES	DATA TYPE	CONSTRAINTS
ProductID	NUMBER	PRIMARY KEY
ProductName	VARCHAR(30)	NOT NULL
Productprice	VARCHAR(30)	NOT NULL
CustomerID	VARCHAR(30)	NOT NULL

TABLES *Stores all the details about the OrderItems*

OrderLineItems

ATTRIBUTES	DATA TYPE	CONSTRAINTS
OrderItemID	**NUMBER**	**PRIMARY KEY**
OrderID	**VARCHAR(30)**	**NOT NULL**
ProductID	**VARCHAR(30)**	**NOT NULL**
OrderItemQuantity	**VARCHAR(30)**	**NOT NULL**
OrderItemPrice	**VARCHAR(30)**	**NOT NULL**
OrderItemDiscount	**VARCHAR(30)**	**NOT NULL**
OrderItemFinalPrice	**VARCHAR(30)**	**NOT NULL**
OrderItemProfit	**VARCHAR(30)**	**NOT NULL**

TABLES *Stores all the details about the different Categories* Category

ATTRIBUTES	DATA TYPE	CONSTRAINTS
CategoryID	NUMBER	PRIMARY KEY
CategoryName	VARCHAR(30)	NOT NULL
DepartmentID	VARCHAR(30)	NOT NULL

TABLES *Stores all the details about the different Departments* Departments

ATTRIBUTES	DATA TYPE	CONSTRAINTS
DepartmentID	NUMBER	PRIMARY KEY
DepartmentName	VARCHAR(30)	NOT NULL

ER Model Assumptions

01 A Customer can place multiple orders and is uniquely identified by their CustomerId.

02 Each Order is uniquely identified by its OrderId and is associated with only one customer

03 An Order can consist of multiple order items, but each Order item is associated with only one Order and is uniquely identified by its OrderItemId

04 Each Order item represents a specific product, which can be uniquely identified by it’s ProductID

05 Each product belongs to only one Category and can be identified by CategoryID

06 Each Category is associated with a particular department, and each department is uniquely identified by its DepartmentID

Functional Dependencies and Primary Key

01 Customers

CustomerID -> { CustomerFname, CustomerLname, CustomerSegment, CustomerCity, CustomerState, CustomerCity }

All the fields can be determined by CustomerID.

Hence, CustomerID is the Primary Key

02 OrderDetails

OrderID -> { CustomerID, OrderDate, OrderStatus, OrderState, OrderRegion, ShippingDate, ShoppingMode, PaymentMode, DeliveryStatus }

All the fields can be determined by OrderID.

Hence, OrderID is the Primary Key

Functional Dependencies and Primary Key

03 Products

ProductID -> { ProductName, ProductPrice, CategoryID }

All the fields can be determined by ProductID.

Hence, ProductID is the Primary Key

04 OrderLineItems

OrderItemID -> { OrderID, ProductID, OrderItemQuantity, OrderItemPrice, OrderItemDiscount, OrderItemFinalPrice, OrderItemProfit}

All the fields can be determined by OrderItemID.

Hence, OrderItemID is the Primary Key

Functional Dependencies and Primary Key

05 Category

CategoryID -> { CategoryName, DepartmentID }

All the fields can be determined by CategoryID.

Hence, CategoryID is the Primary Key

06 Department

DepartmentID -> { DepartmentName}

DepartmentName can be determined by DepartmentID

Hence, DepartmentID is the Primary Key

Normalization

01 Customers

Primary Key: CustomerID

Partial Dependency: All attributes depend fully on the CustomerID and not on part of any composite key, hence the table is in 2NF

Transitive Dependency: There are no transitive dependencies; all attributes depend directly on CustomerID, hence the table is in 3NF

BCNF: All determinants (CustomerID) are candidate keys, hence the table is in BCNF.

02 OrderDetails

Primary Key: OrderID

Partial Dependency: All attributes depend fully on the OrderID and not on part of any composite key, hence the table is in 2NF

Transitive Dependency: There are no transitive dependencies; all attributes depend directly on OrderID, hence the table is in 3NF

BCNF: All determinants (OrderID) are candidate keys, hence the table is in BCNF.

Normalization

03 Products

Primary Key: ProductID

Partial Dependency: All attributes depend fully on the ProductID and not on part of any composite key, hence the table is in 2NF

Transitive Dependency: There are no transitive dependencies; all attributes depend directly on ProductID, hence the table is in 3NF

BCNF: All determinants (ProductID) are candidate keys, hence the table is in BCNF.

04 OrderLineItems

Primary Key: OrderItemID

Partial Dependency: All attributes depend fully on the OrderItemID and not on part of any composite key, hence the table is in 2NF

Transitive Dependency: There are no transitive dependencies; all attributes depend directly on OrderItemID, hence the table is in 3NF

BCNF: All determinants (OrderItemID) are candidate keys, hence the table is in BCNF.

Normalization

05 Category

Primary Key: CategoryID

Partial Dependency: All attributes depend fully on the CategoryID and not on part of any composite key, hence the table is in 2NF

Transitive Dependency: There are no transitive dependencies; all attributes depend directly on CategoryID, hence the table is in 3NF

BCNF: All determinants (CategoryID) are candidate keys, hence the table is in BCNF.

06 Departments

Primary Key: DeaprtmentID

Partial Dependency: All attributes depend fully on the DepartmentID and not on part of any composite key, hence the table is in 2NF

Transitive Dependency: There are no transitive dependencies; all attributes depend directly on DeaartmentID, hence the table is in 3NF

BCNF: All determinants (DepartmentID) are candidate keys, hence the table is in BCNF.

Relationship Schema

01 Places

A one-to-many relationship exists between Customers and Orders. Each customer can place multiple orders.
The **CustomerID** is added as a foreign key in **Orders** table

02 Contains

A one-to-many relationship between Orders and OrderLineItems. Each order can contains multiple order line items.
The OrderID is added as a foreign key in Orderlineitems table

Relationship Schema

03 Is

A one-to-one relationship exists between Orderlineitems and products. Each Orderlineitem is a product.
The **ProductID** is added as a foreign key in **Orderlineitems** table

04 Belongs to

A many-to-one relationship exists between Products and Category. Each Products belongs to only one Category.
The **CategoryID** is added as a foreign key in **Products** table

Relationship Schema

05 Is part of

A many-to-one relationship exists between Category and Department. Each Category is part of only one Department.

The **DepartmentID** is added as a foreign key in **Category** table

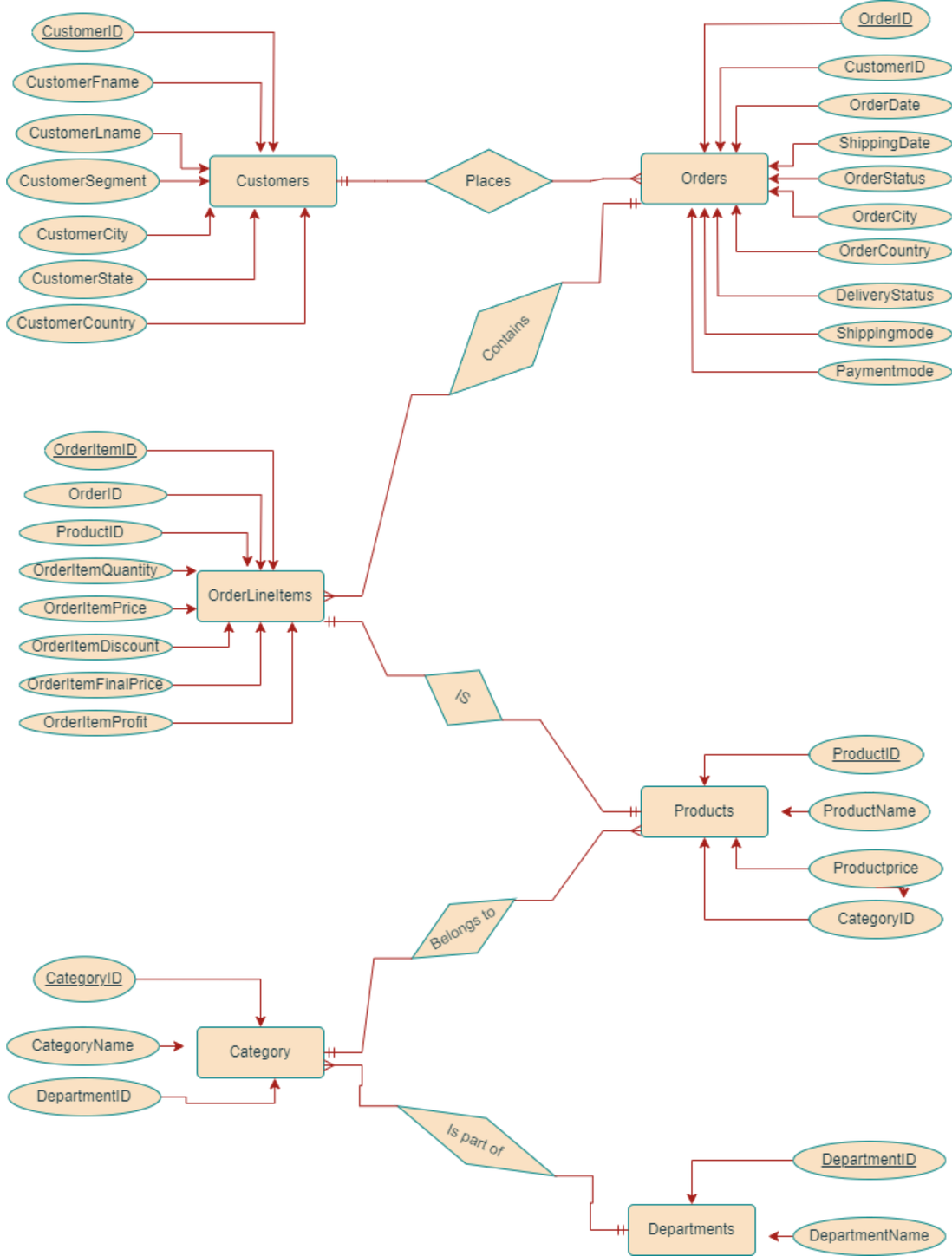
ER DIAGRAM

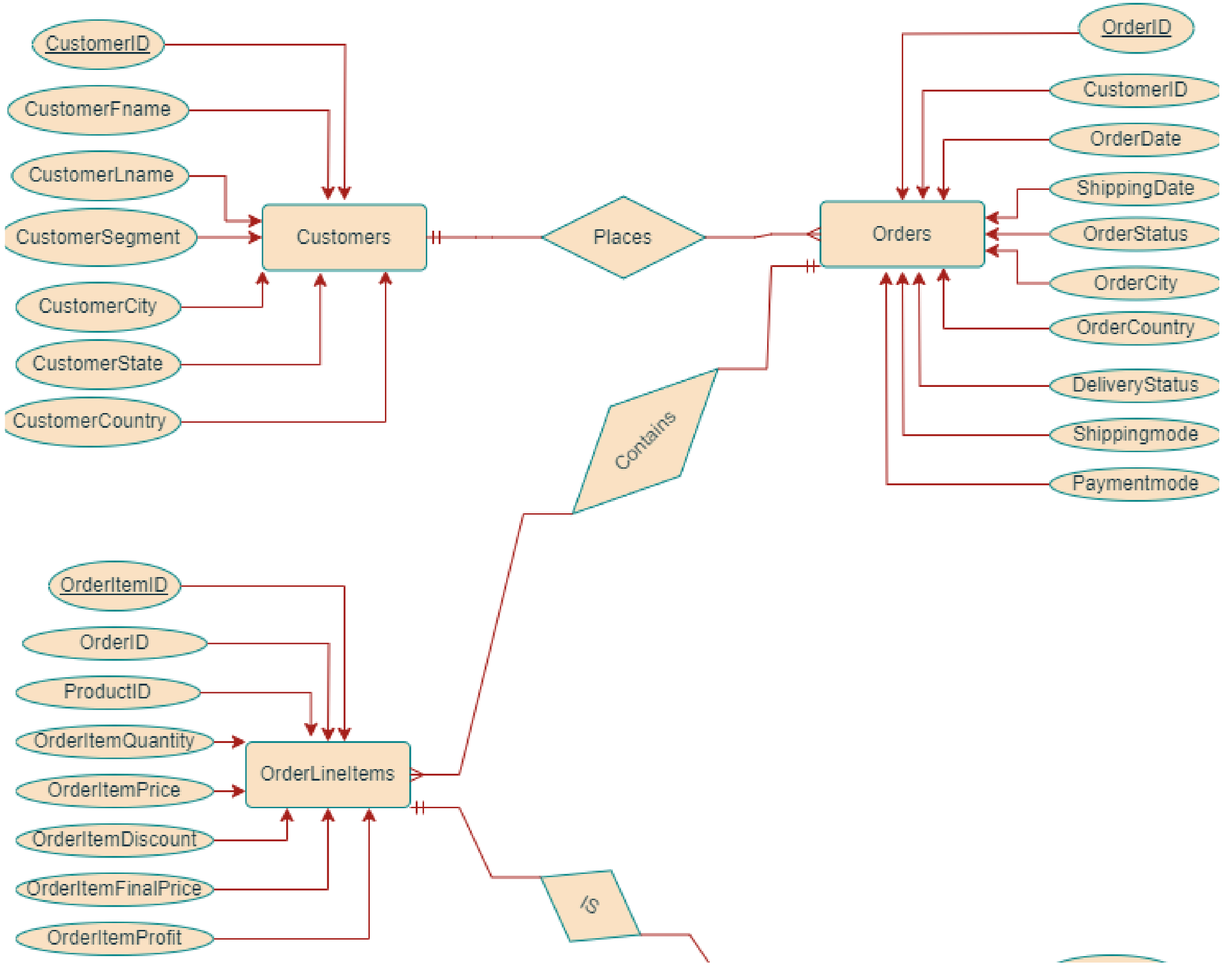
Entity

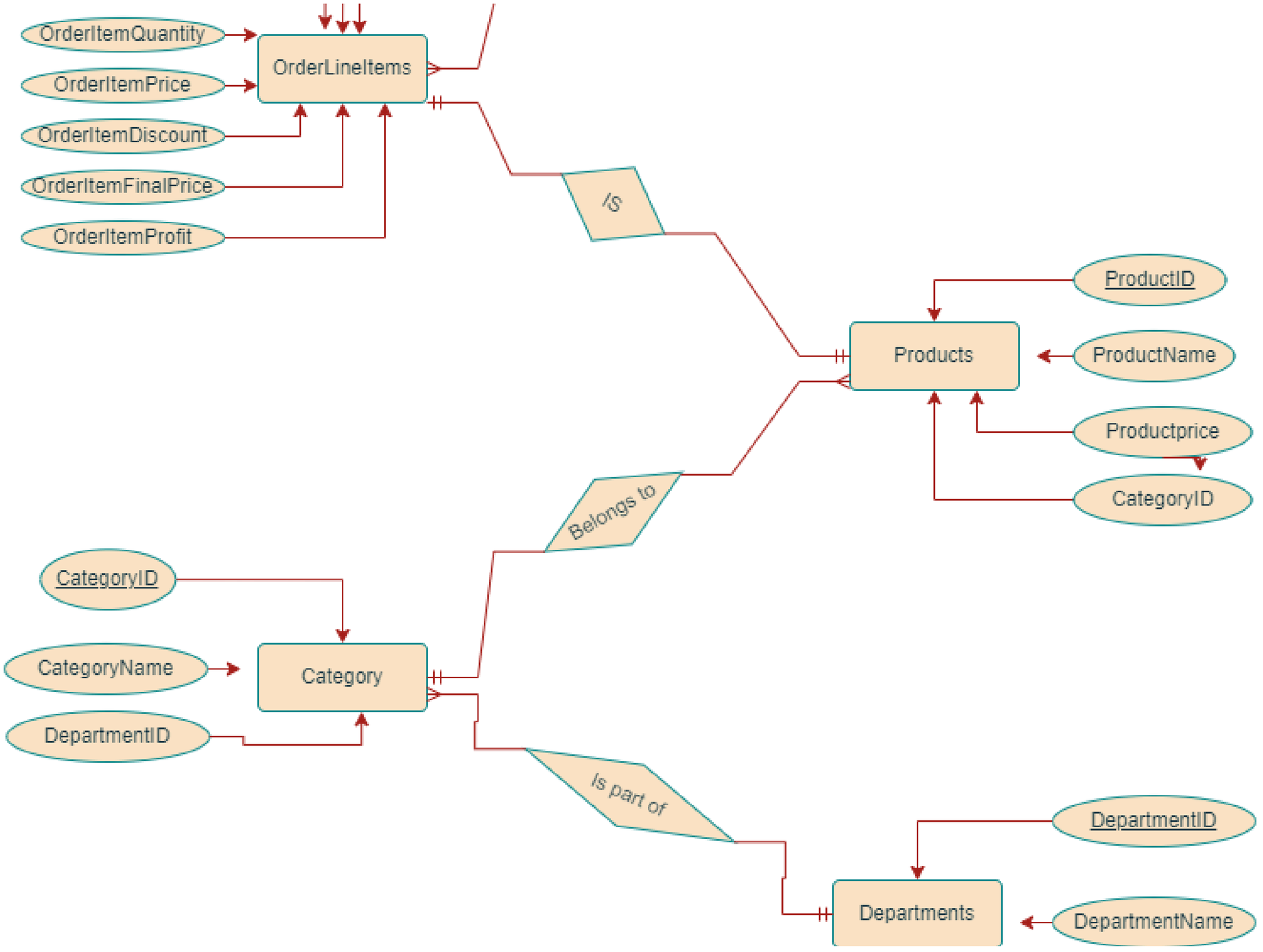
Relationship

Key Attribute

Attribute




CustomerID
CustomerFname
CustomerLname
CustomerSegment
CustomerCity
CustomerState
CustomerCountry
Customers
Places
Orders
OrderID
CustomerID
OrderDate
ShippingDate
OrderStatus
OrderCity
OrderCountry
DeliveryStatus
Shippingmode
Paymentmode
Contains
OrderItemID
OrderID
ProductID
OrderItemQuantity
OrderItemPrice
OrderItemDiscount
OrderItemFinalPrice
OrderItemProfit
OrderLineItems
Is


OrderItemQuantity
OrderItemPrice
OrderItemDiscount
OrderItemFinalPrice
OrderItemProfit
OrderLineItems
IS
Products
ProductID
ProductName
Productprice
CategoryID
Belongs to
CategoryID
CategoryName
DepartmentID
Category
Is part of
DepartmentID
Departments
DepartmentName

SQL IMPLEMENTATION

```sql
CREATE DATABASE Supply_chain_management;
USE Supply_chain_management;

CREATE TABLE Customers(
CustomerID INT PRIMARY KEY,
CustomerFname VARCHAR(30),
CusotmerLname VARCHAR(30),
CusotmerSegment VARCHAR(30),
CusotmerCity VARCHAR(30),
CusotmerState VARCHAR(30),
CusotmerCountry VARCHAR(30)
);

CREATE TABLE Orders(
OrderID INT PRIMARY KEY,
CustomerID VARCHAR(30),
OrderDate VARCHAR(30),
ShippingDate VARCHAR(30),
OrderStatus VARCHAR(30),
OrderCity VARCHAR(30),
OrderCountry VARCHAR(30),
DeliveryStatus VARCHAR(30),
ShippingMode VARCHAR(30),
PaymentMode VARCHAR(30)
);

CREATE TABLE Products(
ProductID INT PRIMARY KEY,
ProductName VARCHAR(30),
ProductPrice VARCHAR(30),
CategoryID VARCHAR(30)
);


CREATE TABLE OrderLineItems(
OrderItemID INT PRIMARY KEY,
OrderID INT,
ProdutID INT,
OrderItemQuantity INT,
OrderItemPrice INT,
OrderItemDiscount INT,
OrderItemFinalPrice INT,
OrderItemProfit INT
);

CREATE TABLE Category(
CategoryID INT PRIMARY KEY,
CategoryName VARCHAR(30),
DepartmentID VARCHAR(30)
);

CREATE TABLE Departments(
DepartmentID INT PRIMARY KEY,
DepartmentName VARCHAR(30)
);
```

Insertion of Values - Customers

```sql
INSERT INTO Customers (CustomerID, CustomerFname, CustomerLname, CustomerSegment, CustomerCity,
CustomerState, CustomerCountry) VALUES
(301, 'Michael', 'Johnson', 'Consumer', 'New York', 'NY', 'USA'),
(302, 'Sarah', 'Miller', 'Corporate', 'Los Angeles', 'CA', 'USA'),
(303, 'David', 'Smith', 'Home Office', 'Chicago', 'IL', 'USA'),
(304, 'Jessica', 'Brown', 'Consumer', 'Houston', 'TX', 'USA'),
(305, 'James', 'Davis', 'Corporate', 'Phoenix', 'AZ', 'USA'),
(306, 'Emily', 'Taylor', 'Home Office', 'Philadelphia', 'PA', 'USA'),
(307, 'Robert', 'Anderson', 'Consumer', 'San Antonio', 'TX', 'USA'),
(308, 'Linda', 'Moore', 'Corporate', 'San Diego', 'CA', 'USA'),
(309, 'William', 'Thomas', 'Home Office', 'Dallas', 'TX', 'USA'),
(310, 'Barbara', 'Jackson', 'Consumer', 'San Jose', 'CA', 'USA');
```

> Similarly Inserted into all other tables.

SQL QUERIES

01 Basic Joins and HAVING clause

Retrieve the list of full names of customers and their earliest order dates who reside in the USA, have used a credit card for payment, and placed the order between Aug 2, 2024 and Aug 10, 2024.

02 Advanced JOINS and Aggregate Functions

Retrieve the names of customers and the total profit generated from all their orders, but only include customers who have ordered from at least three different product categories.

03 Subqueries and Aggregate Functions

List the full names of customers along with their highest single order value. Only include customers who have at least one order with a value higher than the average order value across all customers.

04 Common Table Expressions (CTEs)

List all products along with their Catergory names, Total quantity sold, and total profit, only for orders which are shipping within 2 days. Sort the results by total profit in descending order.

05 Window Functions (DENSE_RANK, PARTITION BY)

For each product category, rank the customers based on the total profit generated from that category. Display the category name, customer full name, total profit, and rank. Ensure ties in profit receive the same rank.

06 Case Statements and Subqueries

Generate a pivot table that shows the Total sales of each product across different Order statuses. The output should have the product names as rows and the order statuses as columns, with the corresponding Total sales values filled in.

1) Retrieve the list of full names of customers and their earliest order dates who reside in the USA, have used a credit card for payment, and placed the order between Aug 2, 2024 and Aug 10, 2024.

```
178  /*Retrieve the list of full names of customers and their earliest order dates who reside in the USA, have used a credit card
179   for payment, and placed the order between August 2, 2024 and August 10, 2024. (Basic Joins and BETWEEN clause)*/
180
181  SELECT C.CustomerID,CONCAT(CustomerFname,' ',CustomerLname) as Name, MIN(OrderDate) as OrderDate
182  FROM Customers C JOIN Orders O ON C.customerID = O.customerID
183  WHERE CustomerCountry ='USA' and Paymentmode= 'Credit Card' and OrderDate BETWEEN '2024-07-02' AND '2024-07-09'
184  GROUP BY C.CustomerID,CONCAT(CustomerFname,' ',CustomerLname)
185  ORDER BY MIN(OrderDate);
186
187
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerID	Name	OrderDate
302	Sarah Miller	2024-07-03
303	David Smith	2024-07-07
304	Jessica Brown	2024-07-09

Result Grid | Form Editor

Result 1 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:29:58	SELECT C.CustomerID,CONCAT(CustomerFname,' ',CustomerLname) as Name, MIN(OrderDate) as OrderDate ...	3 row(s) returned	0.000 sec / 0.000 sec

2) Retrieve the names of customers and the total profit generated from all their orders, but only include customers who have ordered from at least three different product categories.

```
191
192  /*Retrieve the names of customers and the total profit generated from all their orders, but only include customers who have
193   ordered from at least three different product categories. */
194
195  SELECT C.customerID, CustomerFname, SUM(OrderItemProfit)
196  FROM Customers C JOIN Orders O ON C.customerID = O.customerID JOIN OrderLineItems OT ON O.OrderID = OT.OrderID
197  JOIN Products P ON OT.productID = P.productID JOIN Category CT ON CT.CategoryID = P.CategoryID
198  GROUP BY C.customerID
199  HAVING COUNT( DISTINCT CT.CategoryID) >2;
200
201
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customerID	CustomerFname	SUM(OrderItemProfit)
302	Sarah	57000
303	David	88000
304	Jessica	103000
306	Emily	57500
309	William	117000
310	Barbara	85000

Result Grid | Form Editor | Result 2 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:29:58	SELECT C.CustomerID,CONCAT(CustomerFname,' ',CustomerLname) as Name, MIN(OrderDate) as OrderDate ...	3 row(s) returned	0.000 sec / 0.000 sec
2	10:32:32	SELECT C.customerID, CustomerFname, SUM(OrderItemProfit) FROM Customers C JOIN Orders O ON C.cust...	6 row(s) returned	0.032 sec / 0.000 sec

3) List the full names of customers along with their highest single order value. Only include customers who have at least one order with a value higher than the average order value across all customers.

```sql
/* List the names of customers along with their highest single order value. Only include customers who have at least one
order with a value higher than the average order value across all customers. (Subqueries in SELECT and HAVING Clause)*/

SELECT C.CustomerID,C.CustomerFname AS name, MAX(OT.OrderItemQuantity * OT.OrderItemPrice) AS max_order_value
FROM Customers C
JOIN Orders O ON C.customerID = O.customerID
JOIN OrderLineItems OT ON O.orderID = OT.orderID
GROUP BY C.CustomerID
HAVING MAX(OT.OrderItemQuantity * OT.OrderItemPrice) > (
    SELECT AVG(order_avg)
    FROM (
        SELECT AVG(OT.OrderItemQuantity * OT.OrderItemPrice) AS order_avg
        FROM OrderLineItems OT
        GROUP BY OT.orderID) AS order_avg
        );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerID	name	max_order_value
301	Michael	199998
302	Sarah	104999
303	David	159998
304	Jessica	104999
305	James	199998
306	Emily	179998
307	Robert	209998
308	Linda	179998
309	William	199998
310	Barbara	104999

Result 3 | Read Only

4) List all products along with their Catergory names, Total quantity sold, and total profit, only for orders which are shipping within 2 days. Sort the results by total profit in descending order.

```
221
222  /*List all products along with their category names, total quantity sold, and total profit, only for orders which are shipping
223  within 2 days. Sort the results by total profit in descending order. */
224
225  WITH ProductSales AS ( SELECT P.ProductID, P.ProductName, C.CategoryName,
226      COALESCE(SUM(OT.OrderItemQuantity), 0) AS TotalQuantity, COALESCE(SUM(OT.OrderItemProfit), 0) AS TotalProfit
227      FROM Products P JOIN Category C ON P.CategoryID = C.CategoryID
228      JOIN OrderLineItems OT ON P.ProductID = OT.ProductID JOIN Orders O ON OT.OrderID = O.OrderID
229      WHERE DATEDIFF(O.OrderDate,O.ShippingDate) <=2
230      GROUP BY P.ProductID, P.ProductName, C.CategoryName)
231
232  SELECT * FROM ProductSales ORDER BY TotalProfit DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ProductID	ProductName	CategoryName	TotalQuantity	TotalProfit
401	Apple iPhone 14	MobilePhones	11	165000
406	Apple MacBook Pro	Laptops	8	160000
405	HP Ryzen 5	Laptops	10	150000
402	Samsung Galaxy S21	MobilePhones	9	108000
403	RRR Recliner Sofa	Furniture	13	104000
404	MP King size bed	Furniture	10	100000
408	Adidas Trainer	Shoes	3	4500
407	Nike Air Jordan	Shoes	3	2000

Result 4

Output

Action Output

#	Time	Action	Message	Duration

5) For each product category, rank the customers based on the total profit generated from that category. Display the category name, customer name, total profit, and rank. Ensure ties in profit receive the same rank.

```sql
/*For each product category, rank the customers based on the total profit generated from that category. Display the category name,
customer full name, total profit, and rank. Ensure ties in profit receive the same rank.*/

SELECT CT.CategoryName, CustomerFname, SUM(OrderItemProfit) as Total_profit,
DENSE_RANK () OVER (PARTITION BY CategoryName ORDER BY SUM(OrderItemProfit) DESC) as Ranks
FROM Customers C JOIN Orders O ON C.customerID = O.customerID JOIN OrderLineItems OT ON O.OrderID = OT.OrderID
JOIN Products P ON OT.productID = P.productID JOIN Category CT ON CT.CategoryID = P.CategoryID
GROUP BY  CategoryName,CustomerFname;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CategoryName	CustomerFname	Total_profit	Ranks
Furniture	Michael	48000	1
Furniture	David	34000	2
Furniture	William	28000	3
Furniture	Emily	26000	4
Furniture	Jessica	20000	5
Furniture	Robert	20000	5
Furniture	Barbara	18000	6
Furniture	Sarah	10000	7
Laptops	Jessica	70000	1
Laptops	Barbara	55000	2
Laptops	Linda	45000	3
Laptops	Robert	40000	4
Laptops	Sarah	35000	5
Laptops	Emily	30000	6
Laptops	William	20000	7
Laptops	David	15000	8
MobilePhones	Michael	102000	1
MobilePhones	William	66000	2
MobilePhones	David	39000	3
MobilePhones	James	30000	4
MobilePhones	Sarah	12000	5

Result 5 | Result Grid | Form Editor | Field Types | Query Stats | Execution Plan | Read Only

6) Generate a pivot table that shows the Total sales of each product across different Order statuses. The output should have the product names as rows and the order statuses as columns, with the corresponding Total sales values filled in.

```sql
/*Generate a pivot table that shows the Total sales of each product across different Order statuses.The output should have the
product names as rows and the order statuses as columns, with the corresponding Total sales values filled in.*/

SELECT ProductName, SUM( CASE WHEN OrderStatus = 'Shipped' THEN Sales END) as Shipped,
SUM( CASE WHEN OrderStatus = 'Pending' THEN Sales END) as Pending
FROM (SELECT ProductName,OrderStatus,  sum(OrderItemFinalPrice* OrderItemQuantity) as Sales
FROM Orders O JOIN OrderlineItems OT ON O.OrderID = OT.orderID JOIN Products P ON OT.productID = P.productID
GROUP BY P.PRODUCTname, O.orderID) A
GROUP BY ProductName
ORDER BY ProductName;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ProductName	Shipped	Pending
Adidas Trainer	25998	12999
Apple iPhone 14	872991	193998
Apple MacBook Pro	299997	499995
HP Ryzen 5	639992	159998
MP King size bed	257994	171996
Nike Air Jordan	32997	NULL
RRR Recliner Sofa	324990	97497
Samsung Galaxy S21	467994	233997

Result Grid | Form Editor

Result 14 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
35	12:10:59	SELECT ProductName, SUM(CASE WHEN OrderStatus = 'Shipped' THEN Sales END) as Shipped, SUM(C...	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggre...	0.000 sec
36	12:11:19	SELECT ProductName, SUM(CASE WHEN OrderStatus = 'Shipped' THEN Sales END) as Shipped, SUM(C...	8 row(s) returned	0.016 sec / 0.000 sec
37	12:14:51	SELECT ProductName, SUM(CASE WHEN OrderStatus = 'Shipped' THEN Sales END) as Shipped, SUM(C...	8 row(s) returned	0.016 sec / 0.000 sec

THE END