DATABASE MANAGEMENT SYSTEM(UCS310)

Project Report on

Shopping Management System

Submitted By:

Harshleen Kaur (102303220)

Gursharen Kaur Suri (102303227)

Dishavpreet Kaur (102353006)

Karandeep Singh Bhasin (102353008)

Group**: 2C18**

Submitted to**: Ms. Namrata Dimari**

Computer Science and Engineering Department

Thapar Institute of Engineering & Technology, Patiala

March 2025

INDEX

INTRODUCTION

In the modern era of digital commerce, shopping has evolved beyond physical storefronts to virtual platforms that serve millions of users daily. E-commerce has become one of the fastest-growing industries worldwide, driven by convenience, speed, variety, and global accessibility. With the dramatic increase in online transactions, the importance of structured data storage, secure record management, and robust transaction handling has grown significantly. At the heart of all such systems lies a well-designed, normalized database capable of handling real-time operations and storing critical business information.

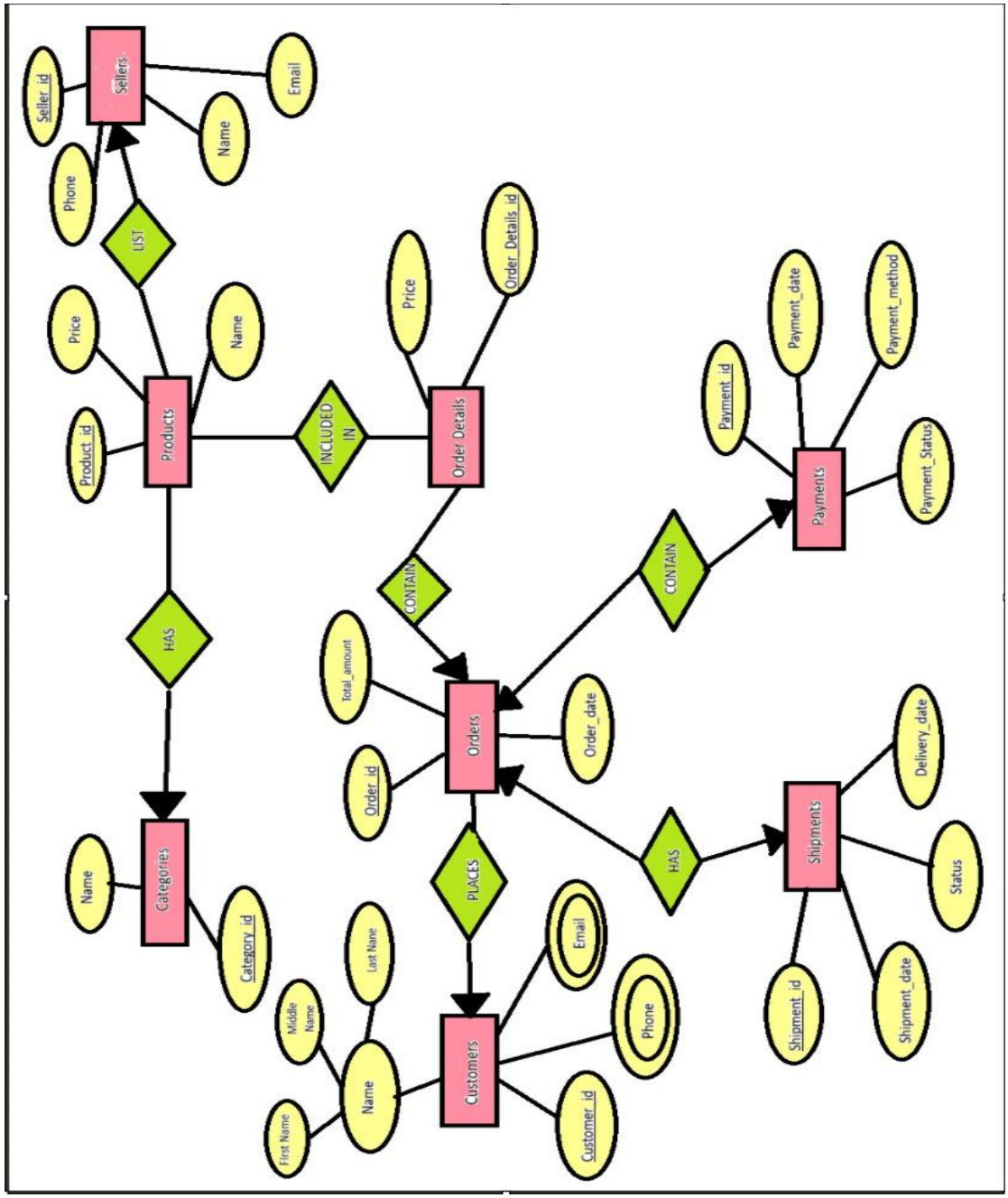
The Shopping Management System project is a database-focused solution that models the core functionalities of a retail shopping platform. This project aims to design and implement a fully normalized relational database system that manages and organizes data associated with customers, sellers, products, categories, orders, payments, and shipments. Unlike complete e-commerce software that includes frontend and backend interfaces, this project focuses solely on the backend database logic using Oracle SQL and PL/SQL. The design ensures accurate data representation, efficient query execution, and seamless integration with future application layers, if developed.

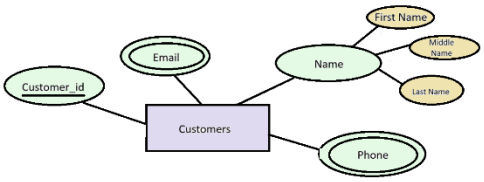
The database schema is carefully structured using Entity-Relationship (ER) modelling and subsequently transformed into relational tables systematically. All tables in the system are normalized up to Third Normal Form (3NF) to eliminate redundancy, prevent anomalies, and maintain consistency. The schema supports one-to-many and many-to-many relationships among entities like Customers and Orders, Products and Categories, or OrderDetails and Product Sets.

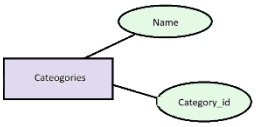
PROJECT RESOURCE REQUIREMENTS:

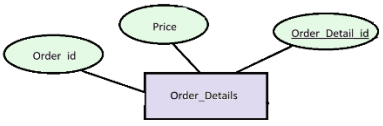
At the core of our system lies the Oracle Relational Database Management System (RDBMS), which serves as the foundation for storing, organizing, and managing all relevant data. Oracle is a robust and enterprise-grade database solution known for its scalability, strong data integrity, and high performance. It supports Structured Query Language (SQL) and operates on a client-server architecture, allowing seamless data access and management through defined queries and user interfaces. Though real-time sensors are not involved in this project, data such as user requests, employee tasks, transport details, and waste delivery records are manually inserted to simulate realistic scenarios.

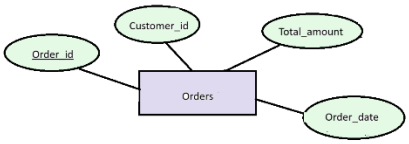
ER DIAGRAM

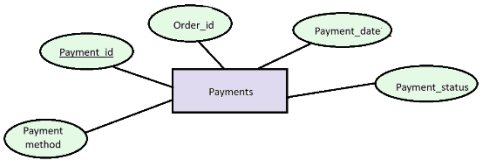

Sellers
Seller_id
Email
Name
Phone
LIST
Products
Price
Name
Product_id
INCLUDED IN
Order Details
Price
Order_Details_id
HAS
CONTAIN
CONTAIN
Payments
Payment_id
Payment_date
Payment_method
Payment_Status
Orders
total_amount
Order_date
Order_id
Categories
Name
Category_id
PLACES
HAS
Shipments
Delivery_date
Status
Shipment_id
Shipment_date
Customers
Email
Phone
Customer_id
Name
First Name
Middle Name
Last Name

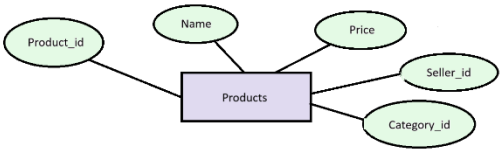


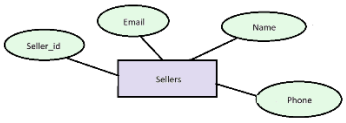


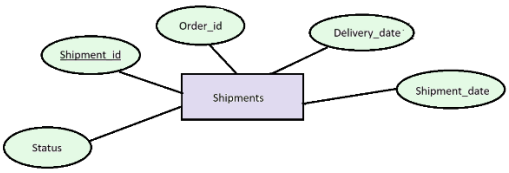












Total: 8 tables (Customers, Categories, Orders, Order_Details, Payments, Products, Sellers, Shipments)

ER DIAGRAM ANALYSIS

Identifying Entity Sets and Relationship Sets:

ENTITY SETS

1. Customers

- Customer_id (Primary Key)
- Name → (First_Name, Middle_Name, Last_Name)
- Phone
- Email

Represents the end-users of the shopping system who place orders.

2. Sellers

- Seller_id (Primary Key)
- Name
- Phone
- Email

Represents the vendors or merchants listing products for sale.

3. Products

- Product_id (Primary Key)
- Name
- Price

Represents the items available for purchase on the shopping platform.

4. Categories

- Category_id (Primary Key)
- Name

Classifies products into groups such as Electronics, Clothing, etc.

5. Orders

- Order_id (Primary Key)
- Order_date
- Total_amount

Captures the details of orders placed by customers.

6. Order Details

- Order_Details_id (Primary Key)
- Price

Represents the line items in each order, mapping specific products to their corresponding orders.

7. Shipments

- Shipment_id (Primary Key)
- Shipment_date
- Delivery_date
- Status

Captures shipping and delivery information of each order.

8. Payments

- Payment_id (Primary Key)
- Payment_date
- Payment_method
- Payment_status

Represents payment information for each order

RELATIONSHIP SETS

1. PLACES (Customer – Orders)

- A customer can place multiple orders.
- Each order is placed by one customer.

Cardinality: One-to-Many (Customer → Orders)

2. LIST (Sellers – Products)

- A seller can list multiple products.
- Each product is listed by one seller.

Cardinality: One-to-Many (Seller → Products)

3. HAS (Categories – Products)

- A category can include multiple products.
- Each product belongs to one category.

Cardinality: One-to-Many (Category → Products)

4. INCLUDED IN (Order Details – Products)

- An order detail includes exactly one product.
- A product can appear in multiple order details.

Cardinality: Many-to-One (Order Details → Products)

5. CONTAIN (Orders – Order Details)

- An order contains multiple order detail records.
- Each order detail is linked to one order.

Cardinality: One-to-Many (Order → Order Details)

6. HAS (Orders – Shipments)

- Each order has one corresponding shipment.
- Each shipment belongs to one order.

Cardinality: One-to-One

7. CONTAIN (Orders – Payments)

- Each order has a corresponding payment.
- Each payment is linked to one order.

Cardinality: One-to-One

ER TO TABLE MAPPING

1. **Customers Table**

Column Name	Data Type	Constraints
Customer_id	INT	PRIMARY KEY
First_Name	VARCHAR(50)	NOT NULL
Middle_Name	VARCHAR(50)	
Last_Name	VARCHAR(50)	NOT NULL
Phone	VARCHAR(15)	UNIQUE
Email	VARCHAR(100)	UNIQUE

2. **Orders Table**

Column Name	Data Type	Constraints
Order_id	INT	PRIMARY KEY
Customer_id	INT	FOREIGN KEY REFERENCES Customers(Customer_id)
Order_date	DATE	NOT NULL
Total_amount	DECIMAL(10,2)	NOT NULL

3. **Products Table**

Column Name	Data Type	Constraints
Product_id	INT	PRIMARY KEY
Name	VARCHAR(100)	NOT NULL
Category_id	INT	FOREIGN KEY REFERENCESCategories(Category_id)
Price	DECIMAL(10,2)	NOT NULL

4. **Sellers Table**

Column Name	Data Type	Constraints
Seller_id	INT	PRIMARY KEY
Name	VARCHAR(100)	NOT NULL
Phone	VARCHAR(15)	UNIQUE
Email	VARCHAR(100)	UNIQUE

5. Order_Details Table

Column Name	Data Type	Constraints
Order_Details_id	INT	PRIMARY KEY
Order_id	INT	FOREIGN KEY REFERENCES Orders(Order_id)
Product_id	INT	FOREIGN KEY REFERENCES Products(Product_id)
Price	DECIMAL(10,2)	NOT NULL

6. Payments Table

Column Name	Data Type	Constraints
Payment_id	INT	PRIMARY KEY
Order_id	INT	FOREIGN KEY REFERENCES Orders(Order_id)
Payment_date	DATE	NOT NULL
Payment_method	VARCHAR(50)	NOT NULL
Payment_status	VARCHAR(50)	NOT NULL

7. Shipments Table

Column Name	Data Type	Constraints
Shipment_id	INT	PRIMARY KEY
Order_id	INT	FOREIGN KEY REFERENCES Orders(Order_id)
Shipment_date	DATE	NOT NULL
Delivery_date	DATE	
Status	VARCHAR(50)	NOT NULL

8. Categories Table

Column Name	Data Type	Constraints
Category_id	INT	PRIMARY KEY
Name	VARCHAR(100)	NOT NULL

NORMALIZATION

1. Categories

Attribute	Type	Description
Category_id	Primary Key	Unique ID for each category
Name	Text	Name of the category

Functional Dependencies:

- Category_id → Name

Normalization Justification: (3NF)

- All attributes are fully functionally dependent on the primary key Category_id. No partial or transitive dependencies exist.

2. Products

Attribute	Type	Description
Product_id	Primary Key	Unique ID for each product
Category_id	Foreign Key (Categories)	Category of the product
Seller_id	Foreign Key (Sellers)	Seller offering the product
Name	Text	Name of the product
Price	Decimal	Price of the product

Functional Dependencies: (3NF)

- Product_id → Category_id, Seller_id, Name, Price

Normalization Justification:

- All non-key attributes are functionally dependent on Product_id.
- No transitive or partial dependencies exist.

3. Sellers

Attribute	Type	Description
Seller_id	Primary Key	Unique ID for each seller
Name	Text	Name of the seller
Phone	Text	Contact number of the seller
Email	Text	Email ID of the seller

Functional Dependencies: (3NF)

- Seller_id → Name, Phone, Email

Normalization Justification:

- All fields are fully dependent on the primary key, Seller_id.

4. IncludesIn

Attribute	Type	Description
SetProduct_id	Foreign Key (Products)	ID of the product in the set
OrderDetail_id	Foreign Key (Order_Details)	ID of the order detail

Composite Primary Key: (SetProduct_id, OrderDetail_id)

Functional Dependencies: (3NF)

- (SetProduct_id, OrderDetail_id) → [SetProduct_id, OrderDetail_id] (by definition of primary key)

Normalization Justification:

- Fully normalized with no partial or transitive dependencies.

5. Order_details

Attribute	Type	Description
OrderDetail_id	Primary Key	Unique ID for order detail
Order_id	Foreign Key (Orders)	Associated order
Price	Decimal	Price of the order detail

Functional Dependencies: (3NF)

- OrderDetail_id → Order_id, Price

Normalization Justification:

- No partial or transitive dependencies; all are fully dependent on OrderDetail_id.

6. Customers

Attribute	Type	Description
Customer_id	Primary Key	Unique ID for each customer
FirstName	Text	Customer's first name
MiddleName	Text	Customer's middle name (optional)
LastName	Text	Customer's last name

Functional Dependencies: (3NF)

- Customer_id → FirstName, MiddleName, LastName

Normalization Justification:

- All attributes are fully dependent on Customer_id.

7. Email

Attribute	Type	Description
Customer_id	Foreign Key (Customers)	Customer associated with the email
Email	Text	Email address of the customer

Composite Primary Key: (Customer_id, Email)

Functional Dependencies: (3NF)

- (Customer_id, Email) → [Customer_id, Email] (by definition)

Normalization Justification:

- Allows storing multiple emails per customer without redundancy.

8. Phone

Attribute	Type	Description
Customer_id	Foreign Key (Customers)	Customer associated with the phone
Phone	Text	Phone number of the customer

Composite Primary Key: (Customer_id, Phone)

Functional Dependencies: (3NF)

- (Customer_id, Phone) → [Customer_id, Phone]

Normalization Justification:

- Proper structure for customers with multiple phone numbers.

9. Shipments

Attribute	Type	Description
Shipment_id	Primary Key	Unique ID for each shipment
Order_id	Foreign Key (Orders)	Order being shipped
Shipment_date	Date	Date the order was shipped
Delivery_date	Date	Expected delivery date
Status	Text	Current shipment status

Functional Dependencies: (3NF)

- Shipment_id → Order_id, Shipment_date, Delivery_date, Status

Normalization Justification:

- All attributes are fully functionally dependent on Shipment_id.

10. Payments

Attribute	Type	Description
Payment_id	Primary Key	Unique ID for each payment
Order_id	Foreign Key (Orders)	Related order
Payment_date	Date	Date of payment
Payment_method	Text	Method used (e.g., Card, Cash)
Payment_status	Text	Payment success/failure status

Functional Dependencies: (3NF)

- Payment_id → Order_id, Payment_date, Payment_method, Payment_status

Normalization Justification:

- All attributes depend entirely on Payment_id.

11. Orders

Attribute	Type	Description
Order_id	Primary Key	Unique ID for each order
Customer_id	Foreign Key (Customers)	Customer who placed the order
Order_date	Date	Date when the order was placed
Total_amount	Decimal	Total amount of the order

Functional Dependencies: (3NF)

- Order_id → Customer_id, Order_date, Total_amount

Normalization Justification:

- All fields are fully dependent on the primary key Order_id.

SQL and PL/SQL

(Stored Procedures and Functions, Cursors, Triggers)

- **Create Table 'Categories'**

```
1    CREATE TABLE Categories (
2        Category_id NUMBER PRIMARY KEY,
3        Name VARCHAR2(100) NOT NULL
4    );
5
```

- **Create Table 'Sellers'**

```
CREATE TABLE Sellers (
    Seller_id NUMBER PRIMARY KEY,
    Name VARCHAR2(100) NOT NULL,
    Phone VARCHAR2(15),
    Email VARCHAR2(100)
);
```

- **Create Table 'Products'**

```
CREATE TABLE Products (
    Product_id NUMBER PRIMARY KEY,
    Category_id NUMBER REFERENCES Categories(Category_id),
    Seller_id NUMBER REFERENCES Sellers(Seller_id),
    Name VARCHAR2(100) NOT NULL,
    Price NUMBER(10,2) NOT NULL
);
```

- **Create table ‘Customers’**

```
CREATE TABLE Customers (
    Customer_id NUMBER PRIMARY KEY,
    FirstName VARCHAR2(50),
    MiddleName VARCHAR2(50),
    LastName VARCHAR2(50)
);
```

- **Create table ‘Customer_Emails’**

```
CREATE TABLE Customer_Emails (
    Customer_id NUMBER REFERENCES Customers(Customer_id),
    Email VARCHAR2(100),
    PRIMARY KEY (Customer_id, Email)
);
```

- **Create table ‘Customers’**

```
CREATE TABLE Customer_Phones (
    Customer_id NUMBER REFERENCES Customers(Customer_id),
    Phone VARCHAR2(15),
    PRIMARY KEY (Customer_id, Phone)
);
```

- **Create table ‘Orders’**

```
CREATE TABLE Orders (
    Order_id NUMBER PRIMARY KEY,
    Customer_id NUMBER REFERENCES Customers(Customer_id),
    Order_date DATE NOT NULL,
    Total_amount NUMBER(10,2)
);
```

- **Create table 'Orders_Details'**

```
CREATE TABLE Order_Details (
    OrderDetail_id NUMBER PRIMARY KEY,
    Order_id NUMBER REFERENCES Orders(Order_id),
    Price NUMBER(10,2)
);
```

- **Create table 'IncludesIn'**

```
CREATE TABLE IncludesIn (
    SetProduct_id NUMBER,
    OrderDetail_id NUMBER,
    PRIMARY KEY (SetProduct_id, OrderDetail_id),
    FOREIGN KEY (SetProduct_id) REFERENCES Products(Product_id),
    FOREIGN KEY (OrderDetail_id) REFERENCES Order_Details(OrderDetail_id)
);
```

- **Create table 'Shipments'**

```
CREATE TABLE Shipments (
    Shipment_id NUMBER PRIMARY KEY,
    Order_id NUMBER REFERENCES Orders(Order_id),
    Shipment_date DATE,
    Delivery_date DATE,
    Status VARCHAR2(50)
);
```

- **Create table 'Orders'**

```
CREATE TABLE Payments (
    Payment_id NUMBER PRIMARY KEY,
    Order_id NUMBER REFERENCES Orders(Order_id),
    Payment_date DATE,
    Payment_method VARCHAR2(50),
    Payment_status VARCHAR2(50)
);
```

- **Insertion of data into Tables**

```
INSERT INTO Payments VALUES (301, 1, TO_DATE('2024-05-01', 'YYYY-MM-DD'), 'Credit Card', 'Completed');
INSERT INTO Payments VALUES (302, 2, TO_DATE('2024-05-02', 'YYYY-MM-DD'), 'Net Banking', 'Completed');
INSERT INTO Payments VALUES (303, 3, TO_DATE('2024-05-03', 'YYYY-MM-DD'), 'UPI', 'Pending');
INSERT INTO Payments VALUES (304, 4, TO_DATE('2024-05-04', 'YYYY-MM-DD'), 'Debit Card', 'Completed');
INSERT INTO Payments VALUES (305, 5, TO_DATE('2024-05-05', 'YYYY-MM-DD'), 'Credit Card', 'Processing');
INSERT INTO Payments VALUES (306, 6, TO_DATE('2024-05-06', 'YYYY-MM-DD'), 'UPI', 'Completed');
INSERT INTO Payments VALUES (307, 7, TO_DATE('2024-05-07', 'YYYY-MM-DD'), 'Wallet', 'Completed');
INSERT INTO Payments VALUES (308, 8, TO_DATE('2024-05-08', 'YYYY-MM-DD'), 'Net Banking', 'Processing');
INSERT INTO Payments VALUES (309, 9, TO_DATE('2024-05-09', 'YYYY-MM-DD'), 'Credit Card', 'Completed');
INSERT INTO Payments VALUES (310, 10, TO_DATE('2024-05-10', 'YYYY-MM-DD'), 'UPI', 'Pending');
```

```
INSERT INTO Categories VALUES (1, 'Electronics');
INSERT INTO Categories VALUES (2, 'Books');
INSERT INTO Categories VALUES (3, 'Clothing');
INSERT INTO Categories VALUES (4, 'Home Appliances');
INSERT INTO Categories VALUES (5, 'Sports');
INSERT INTO Categories VALUES (6, 'Furniture');
INSERT INTO Categories VALUES (7, 'Toys');
INSERT INTO Categories VALUES (8, 'Beauty');
INSERT INTO Categories VALUES (9, 'Groceries');
INSERT INTO Categories VALUES (10, 'Automobile Accessories')
```

```
INSERT INTO Sellers VALUES (1, 'TechZone', '9999990001', 'techzone@example.com');
INSERT INTO Sellers VALUES (2, 'BookWorld', '9999990002', 'bookworld@example.com');
INSERT INTO Sellers VALUES (3, 'FashionFiesta', '9999990003', 'fashion@example.com');
INSERT INTO Sellers VALUES (4, 'ApplianceMart', '9999990004', 'appliance@example.com');
INSERT INTO Sellers VALUES (5, 'Sportify', '9999990005', 'sportify@example.com');
INSERT INTO Sellers VALUES (6, 'FurnitureHub', '9999990006', 'furniture@example.com');
INSERT INTO Sellers VALUES (7, 'Toyland', '9999990007', 'toyland@example.com');
INSERT INTO Sellers VALUES (8, 'GlamUp', '9999990008', 'glamup@example.com');
INSERT INTO Sellers VALUES (9, 'DailyMart', '9999990009', 'dailymart@example.com');
INSERT INTO Sellers VALUES (10, 'AutoParts', '9999990010', 'autoparts@example.com');
```

```
INSERT INTO Products VALUES (1, 1, 1, 'Smartphone', 699.99);
INSERT INTO Products VALUES (2, 2, 2, 'Novel - Mystery', 14.99);
INSERT INTO Products VALUES (3, 3, 3, 'T-Shirt', 9.99);
INSERT INTO Products VALUES (4, 4, 4, 'Microwave', 120.00);
INSERT INTO Products VALUES (5, 5, 5, 'Football', 20.00);
INSERT INTO Products VALUES (6, 6, 6, 'Office Chair', 89.99);
INSERT INTO Products VALUES (7, 7, 7, 'Lego Set', 45.50);
INSERT INTO Products VALUES (8, 8, 8, 'Lipstick', 11.00);
INSERT INTO Products VALUES (9, 9, 9, 'Pack of Rice', 25.00);
INSERT INTO Products VALUES (10, 10, 10, 'Car Wiper', 19.95);
```

```
INSERT INTO Customers VALUES (1, 'Aman', NULL, 'Verma');
INSERT INTO Customers VALUES (2, 'Neha', 'K.', 'Sharma');
INSERT INTO Customers VALUES (3, 'Raj', NULL, 'Kapoor');
INSERT INTO Customers VALUES (4, 'Sara', 'M.', 'Ali');
INSERT INTO Customers VALUES (5, 'Ravi', NULL, 'Kumar');
INSERT INTO Customers VALUES (6, 'Meena', NULL, 'Devi');
INSERT INTO Customers VALUES (7, 'Kunal', NULL, 'Joshi');
INSERT INTO Customers VALUES (8, 'Pooja', NULL, 'Batra');
INSERT INTO Customers VALUES (9, 'Arjun', NULL, 'Mehta');
INSERT INTO Customers VALUES (10, 'Simran', NULL, 'Singh');
```

```
INSERT INTO Customer_Emails VALUES (1, 'amanv@example.com');
INSERT INTO Customer_Emails VALUES (2, 'neha.sharma@example.com');
INSERT INTO Customer_Emails VALUES (3, 'rajk@example.com');
INSERT INTO Customer_Emails VALUES (4, 'saraali@example.com');
INSERT INTO Customer_Emails VALUES (5, 'ravik@example.com');
INSERT INTO Customer_Emails VALUES (6, 'meenad@example.com');
INSERT INTO Customer_Emails VALUES (7, 'kunalj@example.com');
INSERT INTO Customer_Emails VALUES (8, 'poojab@example.com');
INSERT INTO Customer_Emails VALUES (9, 'arjunm@example.com');
INSERT INTO Customer_Emails VALUES (10, 'simrans@example.com');
```

```
INSERT INTO Customer_Phones VALUES (1, '8888800001');
INSERT INTO Customer_Phones VALUES (2, '8888800002');
INSERT INTO Customer_Phones VALUES (3, '8888800003');
INSERT INTO Customer_Phones VALUES (4, '8888800004');
INSERT INTO Customer_Phones VALUES (5, '8888800005');
INSERT INTO Customer_Phones VALUES (6, '8888800006');
INSERT INTO Customer_Phones VALUES (7, '8888800007');
INSERT INTO Customer_Phones VALUES (8, '8888800008');
INSERT INTO Customer_Phones VALUES (9, '8888800009');
INSERT INTO Customer_Phones VALUES (10, '8888800010');
```

```
INSERT INTO Orders VALUES (1, 1, TO_DATE('2024-05-01', 'YYYY-MM-DD'), 699.99);
INSERT INTO Orders VALUES (2, 2, TO_DATE('2024-05-02', 'YYYY-MM-DD'), 34.98);
INSERT INTO Orders VALUES (3, 3, TO_DATE('2024-05-03', 'YYYY-MM-DD'), 89.99);
INSERT INTO Orders VALUES (4, 4, TO_DATE('2024-05-04', 'YYYY-MM-DD'), 45.50);
INSERT INTO Orders VALUES (5, 5, TO_DATE('2024-05-05', 'YYYY-MM-DD'), 120.00);
INSERT INTO Orders VALUES (6, 6, TO_DATE('2024-05-06', 'YYYY-MM-DD'), 11.00);
INSERT INTO Orders VALUES (7, 7, TO_DATE('2024-05-07', 'YYYY-MM-DD'), 25.00);
INSERT INTO Orders VALUES (8, 8, TO_DATE('2024-05-08', 'YYYY-MM-DD'), 20.00);
INSERT INTO Orders VALUES (9, 9, TO_DATE('2024-05-09', 'YYYY-MM-DD'), 19.95);
INSERT INTO Orders VALUES (10, 10, TO_DATE('2024-05-10', 'YYYY-MM-DD'), 14.99);
```

```
INSERT INTO Order_Details VALUES (101, 1, 699.99);
INSERT INTO Order_Details VALUES (102, 2, 14.99);
INSERT INTO Order_Details VALUES (103, 2, 19.99);
INSERT INTO Order_Details VALUES (104, 3, 89.99);
INSERT INTO Order_Details VALUES (105, 4, 45.50);
INSERT INTO Order_Details VALUES (106, 5, 120.00);
INSERT INTO Order_Details VALUES (107, 6, 11.00);
INSERT INTO Order_Details VALUES (108, 7, 25.00);
INSERT INTO Order_Details VALUES (109, 8, 20.00);
INSERT INTO Order_Details VALUES (110, 9, 19.95);
```

```
INSERT INTO IncludesIn VALUES (1, 101);
INSERT INTO IncludesIn VALUES (2, 102);
INSERT INTO IncludesIn VALUES (3, 103);
INSERT INTO IncludesIn VALUES (6, 104);
INSERT INTO IncludesIn VALUES (7, 105);
INSERT INTO IncludesIn VALUES (4, 106);
INSERT INTO IncludesIn VALUES (8, 107);
INSERT INTO IncludesIn VALUES (9, 108);
INSERT INTO IncludesIn VALUES (5, 109);
INSERT INTO IncludesIn VALUES (10, 110);
```

```
INSERT INTO Shipments VALUES (201, 1, TO_DATE('2024-05-01', 'YYYY-MM-DD')
, TO_DATE('2024-05-03', 'YYYY-MM-DD'), 'Delivered');
INSERT INTO Shipments VALUES (202, 2, TO_DATE('2024-05-02', 'YYYY-MM-DD'),
 TO_DATE('2024-05-05', 'YYYY-MM-DD'), 'Delivered');
INSERT INTO Shipments VALUES (203, 3, TO_DATE('2024-05-03', 'YYYY-MM-DD'),
 TO_DATE('2024-05-06', 'YYYY-MM-DD'), 'Shipped');
INSERT INTO Shipments VALUES (204, 4, TO_DATE('2024-05-04', 'YYYY-MM-DD'),
TO_DATE('2024-05-06', 'YYYY-MM-DD'), 'Delivered');
INSERT INTO Shipments VALUES (205, 5, TO_DATE('2024-05-05', 'YYYY-MM-DD'),
 NULL, 'Processing');
INSERT INTO Shipments VALUES (206, 6, TO_DATE('2024-05-06', 'YYYY-MM-DD'),
TO_DATE('2024-05-08', 'YYYY-MM-DD'), 'Delivered');
INSERT INTO Shipments VALUES (207, 7, TO_DATE('2024-05-07', 'YYYY-MM-DD'),
 TO_DATE('2024-05-09', 'YYYY-MM-DD'), 'Delivered');
INSERT INTO Shipments VALUES (208, 8, TO_DATE('2024-05-08', 'YYYY-MM-DD'),
 NULL, 'Shipped');
INSERT INTO Shipments VALUES (209, 9, TO_DATE('2024-05-09', 'YYYY-MM-DD'),
 TO_DATE('2024-05-11', 'YYYY-MM-DD'), 'Delivered');
INSERT INTO Shipments VALUES (210, 10, TO_DATE('2024-05-10', 'YYYY-MM-DD'),
 NULL, 'Processing');
```

- **Triggers/ Procedures**

```
CREATE OR REPLACE PROCEDURE AddCustomer(
    p_Customer_id IN NUMBER,
    p_FirstName IN VARCHAR2,
    p_MiddleName IN VARCHAR2,
    p_LastName IN VARCHAR2,
    p_Email IN VARCHAR2,
    p_Phone IN VARCHAR2
)
IS
BEGIN
    INSERT INTO Customers(Customer_id, FirstName, MiddleName, LastName)
    VALUES (p_Customer_id, p_FirstName, p_MiddleName, p_LastName);

    INSERT INTO Customer_Emails(Customer_id, Email)
    VALUES (p_Customer_id, p_Email);

    INSERT INTO Customer_Phones(Customer_id, Phone)
    VALUES (p_Customer_id, p_Phone);

    COMMIT;
END;
/
```

```
CREATE OR REPLACE TRIGGER Update_Shipment_Status
AFTER INSERT ON Payments
FOR EACH ROW
BEGIN
    UPDATE Shipments
    SET Status = 'Shipped'
    WHERE Order_id = :NEW.Order_id;
END;
/
```

```
CREATE OR REPLACE TRIGGER Prevent_Negative_Price
BEFORE INSERT OR UPDATE ON Products
FOR EACH ROW
BEGIN
    IF :NEW.Price < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Product price cannot be negative');
    END IF;
END;
/
```

```
/
CREATE OR REPLACE PROCEDURE PlaceOrder(
    p_Customer_id IN NUMBER,
    p_Product_id IN NUMBER,
    p_Quantity IN NUMBER
)
IS
    v_Order_id NUMBER;
    v_Total_amount NUMBER(10,2);
BEGIN
    -- Calculate total amount
    SELECT Price * p_Quantity INTO v_Total_amount
    FROM Products
    WHERE Product_id = p_Product_id;

    -- Insert into Orders
    SELECT NVL(MAX(Order_id), 500) + 1 INTO v_Order_id FROM Orders;

    INSERT INTO Orders(Order_id, Customer_id, Order_date, Total_amount)
    VALUES (v_Order_id, p_Customer_id, SYSDATE, v_Total_amount);

    -- Insert into Order Details
    INSERT INTO Order_Details(OrderDetail_id, Order_id, Price)
    VALUES (v_Order_id + 10000, v_Order_id, v_Total_amount);

    -- Insert into IncludesIn
    INSERT INTO IncludesIn(SetProduct_id, OrderDetail_id)
    VALUES (p_Product_id, v_Order_id + 10000);

    COMMIT;
END;
/
```

FINAL OUTPUTS

	CATEGORY_ID	NAME
1	1	Electronics
2	2	Books
3	3	Clothing
4	4	Home Appliances
5	5	Sports
6	6	Furniture
7	7	Toys
8	8	Beauty
9	9	Groceries
10	10	Automobile Accessor

	T_ID	CATEGORY_ID	SELLER_ID	NAME	PRICE
1	1	1	1	Smartphone	699.99
2	2	2	2	Novel - Mystery	14.99
3	3	3	3	T-Shirt	9.99
4	4	4	4	Microwave	120
5	5	5	5	Football	20
6	6	6	6	Office Chair	89.99
7	7	7	7	Lego Set	45.5
8	8	8	8	Lipstick	11
9	9	9	9	Pack of Rice	25
10	10	10	10	Car Wiper	19.95

	SELLER_ID	NAME	PHONE	EMAIL
1	1	TechZone	9999990001	techzone@example.(
2	2	BookWorld	9999990002	bookworld@example
3	3	FashionFiesta	9999990003	fashion@example.co
4	4	ApplianceMart	9999990004	appliance@example.
5	5	Sportify	9999990005	sportify@example.cc
6	6	FurnitureHub	9999990006	furniture@example.c
7	7	Toyland	9999990007	toyland@example.co
8	8	GlamUp	9999990008	glamup@example.cc
9	9	DailyMart	9999990009	dailymart@example.
10	10	AutoParts	9999990010	autoparts@example.

	SETPRODUCT_ID	ORDERDETAIL_ID
1	1	101
2	2	102
3	3	103
4	4	106
5	5	109
6	6	104
7	7	105
8	8	107
9	9	108
10	10	110

	CUSTOMER_ID	FIRSTNAME	MIDDLENAME	LASTNAME
1	1	Aman	*(null)*	Verma
2	2	Neha	K.	Sharma
3	3	Raj	*(null)*	Kapoor
4	4	Sara	M.	Ali
5	5	Ravi	*(null)*	Kumar
6	6	Meena	*(null)*	Devi
7	7	Kunal	*(null)*	Joshi
8	8	Pooja	*(null)*	Batra
9	9	Arjun	*(null)*	Mehta
10	10	Simran	*(null)*	Singh

	ORDERDETAIL_ID	ORDER_ID	PRICE
1	101	1	699.99
2	102	2	14.99
3	103	2	19.99
4	104	3	89.99
5	105	4	45.5
6	106	5	120
7	107	6	11
8	108	7	25
9	109	8	20
10	110	9	19.95

	CUSTOMER_ID	EMAIL
1	1	amanv@example.cor
2	2	neha.sharma@exam
3	3	rajk@example.com
4	4	saraali@example.cor
5	5	ravik@example.com
6	6	meenad@example.c
7	7	kunalj@example.con
8	8	poojab@example.cor
9	9	arjunm@example.co
10	10	simrans@example.cc

	CUSTOMER_ID	PHONE
1	1	8888800001
2	2	8888800002
3	3	8888800003
4	4	8888800004
5	5	8888800005
6	6	8888800006
7	7	8888800007
8	8	8888800008
9	9	8888800009
10	10	8888800010

	ORDER_ID	CUSTOMER_ID	ORDER_DATE	TOTAL_AMOUNT
1	1	1	5/1/2024, 12:00:00	699.99
2	2	2	5/2/2024, 12:00:00	34.98
3	3	3	5/3/2024, 12:00:00	89.99
4	4	4	5/4/2024, 12:00:00	45.5
5	5	5	5/5/2024, 12:00:00	120
6	6	6	5/6/2024, 12:00:00	11
7	7	7	5/7/2024, 12:00:00	25
8	8	8	5/8/2024, 12:00:00	20
9	9	9	5/9/2024, 12:00:00	19.95
10	10	10	5/10/2024, 12:00:0(	14.99

	ORDERDETAIL_ID	ORDER_ID	PRICE
1	101	1	699.99
2	102	2	14.99
3	103	2	19.99
4	104	3	89.99
5	105	4	45.5
6	106	5	120
7	107	6	11
8	108	7	25
9	109	8	20
10	110	9	19.95

	SETPRODUCT_ID	ORDERDETAIL_ID
1	1	101
2	2	102
3	3	103
4	4	106
5	5	109
6	6	104
7	7	105
8	8	107
9	9	108
10	10	110

	SHIPMENT_ID	ORDER_ID	SHIPMENT_DATE	DELIVERY_DATE	STATUS
1	201	1	5/1/2024, 12:00:00	5/3/2024, 12:00:00	Delivered
2	202	2	5/2/2024, 12:00:00	5/5/2024, 12:00:00	Delivered
3	203	3	5/3/2024, 12:00:00	5/6/2024, 12:00:00	Shipped
4	204	4	5/4/2024, 12:00:00	5/6/2024, 12:00:00	Delivered
5	205	5	5/5/2024, 12:00:00	*(null)*	Processing
6	206	6	5/6/2024, 12:00:00	5/8/2024, 12:00:00	Delivered
7	207	7	5/7/2024, 12:00:00	5/9/2024, 12:00:00	Delivered
8	208	8	5/8/2024, 12:00:00	*(null)*	Shipped
9	209	9	5/9/2024, 12:00:00	5/11/2024, 12:00:0(	Delivered
10	210	10	5/10/2024, 12:00:0(	*(null)*	Processing

	PAYMENT_ID	ORDER_ID	PAYMENT_DATE	PAYMENT_METHOD	PAYMENT
1	301	1	5/1/2024, 12:00:00	Credit Card	Completed
2	302	2	5/2/2024, 12:00:00	Net Banking	Completed
3	303	3	5/3/2024, 12:00:00	UPI	Pending
4	304	4	5/4/2024, 12:00:00	Debit Card	Completed
5	305	5	5/5/2024, 12:00:00	Credit Card	Processing
6	306	6	5/6/2024, 12:00:00	UPI	Completed
7	307	7	5/7/2024, 12:00:00	Wallet	Completed
8	308	8	5/8/2024, 12:00:00	Net Banking	Processing
9	309	9	5/9/2024, 12:00:00	Credit Card	Completed
10	310	10	5/10/2024, 12:00:0(	UPI	Pending

JOINS and SUBQUERIES

1) Display all products along with their category name and seller name:

```
Select P.Name AS Product_Name,S.Name AS Seller_Name,C.Name AS Customer_Name
 from Products P,Categories C,Sellers S
where P.Category_id=C.Category_id AND P.Seller_id=S.Seller_id;
```

Output:

	PRODUCT_NAME	SELLER_NAME	CUSTOMER_NAME
1	Smartphone	TechZone	Electronics
2	Novel - Mystery	BookWorld	Books
3	T-Shirt	FashionFiesta	Clothing
4	Microwave	ApplianceMart	Home Appliances
5	Football	Sportify	Sports
6	Office Chair	FurnitureHub	Furniture
7	Lego Set	Toyland	Toys
8	Lipstick	GlamUp	Beauty

2) Find the average price of products in each:

```
SELECT c.Name AS Category_Name, AVG(p.Price) AS Avg_Price
FROM Categories c
JOIN Products p ON c.Category_id = p.Category_id
GROUP BY c.Name;
```

Output:

	CATEGORY_NAME	AVG_PRICE
1	Electronics	699.99
2	Books	14.99
3	Clothing	9.99
4	Home Appliances	120
5	Sports	20
6	Furniture	89.99
7	Toys	45.5
8	Beauty	11

CONCLUSION

The Shopping Management System project successfully demonstrates the practical implementation of database design principles using PL/SQL and Oracle SQL. The developed ER model efficiently captures all critical components of an online shopping platform, including customers, sellers, products, orders, shipments, payments, and product categorization. Each entity and relationship has been carefully modeled to reflect real-world operations, ensuring both normalization and data integrity.

The use of PL/SQL provided the flexibility to handle complex business logic, such as automated order status updates, inventory tracking, and secure payment handling. Oracle SQL's robust querying capabilities enabled the efficient retrieval, manipulation, and management of large datasets, making the system scalable and responsive.

Through this project, we not only learned how to design and normalize relational databases but also gained hands-on experience in translating real-life commercial requirements into functional database structures. We ensured referential integrity through primary and foreign key constraints, and optimized the schema to minimize redundancy and improve data consistency.

In conclusion, the Shopping Management System serves as a comprehensive database solution for managing e-commerce operations. It lays a strong foundation for future enhancements, such as integrating user feedback, implementing role-based access control, and adding advanced analytics features. The project highlights the power of structured database design and PL/SQL programming in building reliable and efficient management systems.

REFERENCES

- W3Schools SQL and MySQL Tutorials
- GeeksforGeeks (DBMS and SQL Sections)
- Official MySQL Documentation
- Academic course notes from Database Management System lectures