LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

Autumn 2024

Student Name: Krish Bhattarai

London Met ID: 22067570

Assignment Submission Date: 14/01/2024

Word Count: 1880

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Gadget Emporium is an online store founded by an entrepreneur and an electronic enthusiast Mr. John. The focus of the online store is to sell electronic devices and accessories. It provides a diverse range of electronic products to its customers ensuring that the customers have access to the latest electronics.

To support the “Gadget Emporium” online marketplace, the database system manages electronic products, customer details, orders, vendor, and inventory. It categorizes customers into Regular, Staff, VIP each with specific discount rates. Order details includes products, quantities, prices, and payment information. The Vendor is associated with product, and real-time inventory management. Additionally, it incorporates a number of payment methods, such as e-wallets, credit/debit cards, and cash on delivery. Following order confirmation, an invoice is created including the client, order, and payment information along with any applicable discounts.

1.1 Business Rule

- A product can be a part of more than one order, and an order may include more than one product.
- A customer can order multiple products.
- Consumers need to be divided into three categories: VIP, Staff, and Regular, each with a different discount rate.
- To distribute products efficiently, customer addresses are stored.
- Same product can be included in multiple orders.
- The Unit price attribute in Product entity captures the price of each product.
- When an order is placed, an invoice is generated which then confirms the order and sets the order.
- To maintain accuracy and consistency in tracking consumer purchases, the Order entity acts as a single repository for all order transactions.
- Order tracking can be facilitated by using the OrderDate, which offers an exact date for the order's placement.

2. Initial ERD

Before the normalization process, an initial ERD is created. It is a visual representation of the key entities, relationships and attributes in a database system that helps analyze the flow of data.

2.1 List of objects created.

- Product – ProductID(PK), ProductName, Description, UnitPrice, Stock, ProductCategory
- Customer – CustomerID(PK), CustomerName, Address, Discount, CustomerCategory
- Order – OrderID(PK), CustomerID(FK), OrderDate, InvoiceID, PaymentOption, TotalAmount, InvoiceTotal

2.2 Identification and Representation of Primary and Foreign Keys

In the Product table, the ProductID is the primary key, in Customer table, the CustomerID is the primary key and in the Order table, the OrderID is the primary key with CustomerID being the foreign key from the Customer table.

2.3 Data Dictionary

Product

Attributes	Data Type	Constraints
ProductID (PK)	Number	Primary Key
ProductName	VARCHAR	Not Null
ProductDescription	VARCHAR	Not Null
UnitPrice	Number	Not Null
Stock	Number	Not Null
ProductCategory	VARCHAR	Not Null

Table 1: Data Dictionary for Product.

Customer

Attributes	Data Type	Constraint
CustomerID(PK)	Number	Primary Key
CustomerName	VARCHAR	Not Null
Address	VARCHAR	Not Null
Discount	NUMBER	Not Null
CustomerCategory	VARCHAR	Not Null

Table 2: Data Dictionary for Customer.

Order

Attributes	Data Type	Constraint
OrderID(PK)	NUMBER	Primary Key
CustomerID (FK)	NUMBER	Foreign Key
OrderDate	DATE	Not Null
InvoiceNo	NUMBER	Not Null
PaymentOption	VARCHAR	Not Null
TotalAmount	NUMBER	Not Null
InvoiceTotal	NUMBER	Not Null

Table 3: Data Dictionary for Order.

2.4 Entity Relationship Diagram

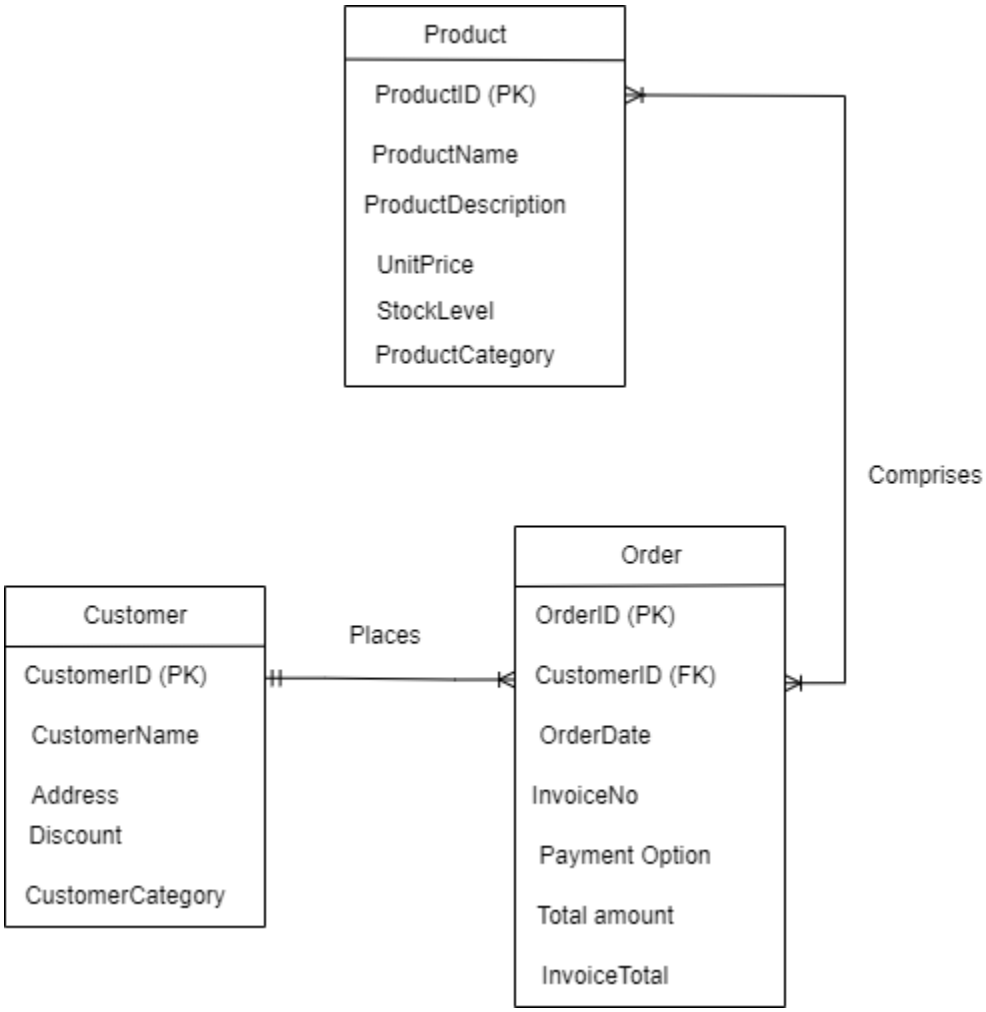


Figure 1: Initial ERD

3. Normalization

The process of organizing data in a database so that all related data are stored in one place and there is no redundancy of data is normalization. Normalization is important because it takes less disk space which results in performance increase. There are three main types of normalization (Rouse, 2023). They are:

3.1 UNF

It is the first step of normalization. All the attributes are gathered and put in a list. The relation is named and identification if keys is done. In this scenario, the attributes are:

Order (OrderID, OrderDate, InvoiceNo, InvoiceTotal, PaymentOption, TotalAmount, Discount, CustomerID, CustomerCategory, CustomerName, Address {ProductID, ProductName, ProductCategory, Description, StockLevel, OrderQuantity, UnitPrice, TotalOrder VendorID, VendorName})

3.2 1NF

It is the initial level of database normalization. Basic criteria of seperation of repeating groups is done in this level. The following tables are formed in 1NF:

Order-1 (OrderID, OrderDate, InvoiceNo, InvoiceTotal, PaymentOption, TotalAmount, Discount, CustomerID, CustomerCategory, CustomerName, Address)

Product-Order-1 (ProductID, OrderID*, ProductName, ProductCategory, Description, StockLevel, OrderQuantity, UnitPrice, TotalOrder VendorID, VendorName)

3.3 2NF

In 2NF, partial dependency is removed. All the functional dependencies are identified in 1NF, and a new primary key is made in new Relation. The following tables are generated in 2NF:

OrderID → X

OrderID, ProductID → OrderQuantity

ProductID → ProductName, ProductCategory, Description, StockLevel, UnitPrice, TotalOrder VendorID, VendorName

Table:

Order-2 (OrderID, OrderDate, InvoiceNo, Invoice Total, PaymentOption, TotalAmount, Discount, CustomerID, CustomerCategory, CustomerName, Address)

Product-Order-2 (OrderID*, ProductID*, OrderQuantity)

Product-2 (ProductID, ProductName, ProductCategory, Description, Stock, UnitPrice, TotalOrder VendorID, VendorName)

3.4 3NF

3NF is the last step of normalization. In this step, transitive dependency is eliminated. It specifies that there should not be any transitive dependencies and that all non-prime attributes in a table must be functionally reliant on the primary key. The following tables are generated in 3NF:

From Order,

OrderID →CustomerID

CustomerID → CustomerName, CustomerCategory, Discount, Address

CustomerCategory → Discount

InvoiceNo → PaymentOption, TotalAmount, Invoice Total

From Product,

ProductID → VendorID

VendorID → VendorName

Table:

Order-3 (OrderID, CustomerID*, OrderDate, TotalOrder, InvoiceID*)

Customer-3 (CustomerID, CustomerName, CustomerAddress, CustomerCategory*)

Product-order-3 (OrderID*, ProductID*, OrderQuantity)

Product-3 (ProductID, ProductName, Description, ProductCategory, StockLevel, VendorID*, ProductPrice)

Discount-3 (CustomerCategory, Discount)

Vendor-3 (VendorID, VendorName)

Invoice-3 (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)

4. Final ERD

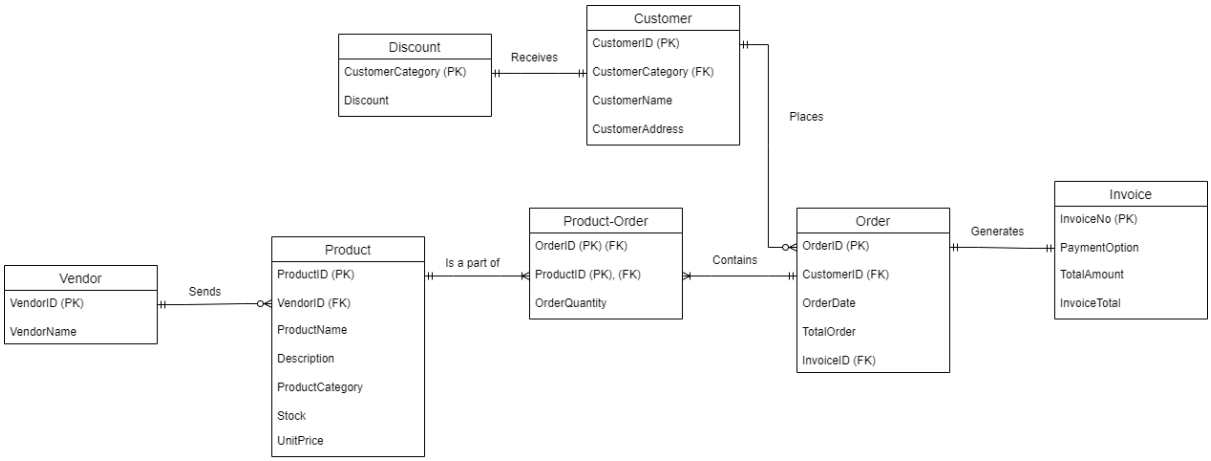


Figure 2: Final ERD.

Data Dictionary

Vendor

Attributes	Data Type	Constraints
VendorID (PK)	Number	Primary Key
VendorName	VARCHAR	NOT NULL

Table 4: Data Dictionary for Vendor.

Discount

Attributes	Data Type	Constraints
CustomerCategory (PK)	VARCHAR	Primary Key
Discount	Number	Not Null

Table 5: Data Dictionary for Discount.

Invoice

Attributes	Data Type	Constraints
InvoiceNo(PK)	Number	Primary Key
PaymentOption	VARCHAR	Not Null
TotalAmount	Number	Not Null
InvoiceTotal	Number	Not Null

Table 6: Data Dictionary for Invoice.

Customer

Attributes	Data Types	Constraints
CustomerID(PK)	Number	Primary Key
CustomerCategory(FK)	VARCHAR	Foreign Key
CustomerName	VARCHAR	Not Null
CustomerAddress	VARCHAR	Not Null

Table 7: Data Dictionary for Customer.

Product

Attributes	Data Types	Constraints
ProductID (PK)	Number	Primary Key
VendorID (FK)	Number	Foreign Key
ProductName	VARCHAR	Not Null
Description	VARCHAR	Not Null
ProductCategory	VARCHAR	Not Null
Stock	Number	Not Null
UnitPrice	Number	Not Null

Table 8: Data Dictionary for Product.

Order

Attributes	Data Types	Constraints
OrderID (PK)	Number	Primary Key
CustomerID (FK)	Number	Foreign Key
OrderDate	Number	Not Null
TotalOrder	Number	Not Null
InvoiceID (FK)	Number	Foreign Key

Table 9: Data Dictionary for Order.

ProductOrder

Attributes	Data Types	Constraints
OrderID (PK)(FK)	Number	Primary Key, Foreign Key
ProductID (PK)(FK)	Number	Primary Key, Foreign Key
OrderQuantity	Number	Not Null

Table 10: Data Dictionary for ProductOrder.

5. Implementation

5.1 Creating Tables

5.1.1 Creating Invoice Table

```
SQL> CREATE TABLE Invoice (
  2  InvoiceNo NUMBER PRIMARY KEY,
  3  PaymentOption VARCHAR(100) NOT NULL,
  4  TotalAmount NUMBER NOT NULL,
  5  InvoiceTotal NUMBER NOT NULL
  6  );

Table created.
```

Figure 3: Creating the Invoice Table.

```
SQL> DESC Invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INVOICENO                                 NOT NULL NUMBER
 PAYMENTOPTION                             NOT NULL VARCHAR2(100)
 TOTALAMOUNT                               NOT NULL NUMBER
 INVOICETOTAL                              NOT NULL NUMBER
```

Figure 4: Description of the Invoice Table.

5.1.2 Creating Vendor Table

```
SQL> CREATE TABLE Vendor (
  2  VendorID NUMBER PRIMARY KEY,
  3  VendorName VARCHAR(50) NOT NULL
  4  );

Table created.
```

Figure 5: Creating the Vendor Table.

```
SQL> DESC Vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VENDORID                                  NOT NULL NUMBER
 VENDORNAME                                NOT NULL VARCHAR2(50)
```

Figure 6: Describing the vendor Table.

5.1.3 Creating Discount Table

```
SQL> CREATE TABLE Discount (
  2  CustomerCategory VARCHAR (30) PRIMARY KEY,
  3  Discount NUMBER NOT NULL
  4  );

Table created.
```

Figure 7: Creating the Discount Table.

```
SQL> DESC Discount;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMERCATEGORY                          NOT NULL VARCHAR2(30)
 DISCOUNT                                  NOT NULL NUMBER
```

Figure 8: Describing the discount Table.

5.1.4 Creating Product table

```
SQL> CREATE TABLE Product (
  2  ProductID NUMBER PRIMARY KEY,
  3  ProductName VARCHAR(20) NOT NULL,
  4  ProductCategory VARCHAR(50) NOT NULL,
  5  ProductDescription VARCHAR(50) NOT NULL,
  6  Stock NUMBER NOT NULL,
  7  UnitPrice NUMBER NOT NULL,
  8  VendorID NUMBER NOT NULL,
  9  CONSTRAINT fk_vendor
 10  FOREIGN KEY (VendorID)
 11  REFERENCES Vendor(VendorID));

Table created.
```

Figure 9: Creating the Product table.

```
SQL> DESC Product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCTID                                 NOT NULL NUMBER
 PRODUCTNAME                               NOT NULL VARCHAR2(20)
 PRODUCTCATEGORY                           NOT NULL VARCHAR2(50)
 PRODUCTDESCRIPTION                        NOT NULL VARCHAR2(50)
 STOCK                                     NOT NULL NUMBER
 UNITPRICE                                 NOT NULL NUMBER
 VENDORID                                  NOT NULL NUMBER
```

Figure 10: Describing Product table.

5.1.5 Creating table Customer Table

```
SQL> CREATE TABLE Customer (
  2  CustomerID NUMBER PRIMARY KEY,
  3  CustomerName VARCHAR (50) NOT NULL,
  4  CustomerAddress VARCHAR(50) NOT NULL,
  5  CustomerCategory VARCHAR(30) NOT NULL,
  6  CONSTRAINT fk_discount
  7  FOREIGN KEY (CustomerCategory)
  8  REFERENCES Discount(CustomerCategory));

Table created.
```

Figure 11: Creating the customer table.

```
SQL> DESC Customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMERID                                NOT NULL NUMBER
 CUSTOMERNAME                              NOT NULL VARCHAR2(50)
 CUSTOMERADDRESS                           NOT NULL VARCHAR2(50)
 CUSTOMERCATEGORY                          NOT NULL VARCHAR2(30)
```

Figure 12: Describing the Customer table.

5.1.6 Creating Orders Table

```
SQL> CREATE TABLE Orders (
  2  OrderID NUMBER PRIMARY KEY,
  3  OrderDate DATE NOT NULL,
  4  TotalOrder NUMBER NOT NULL,
  5  CustomerID NUMBER NOT NULL,
  6  CONSTRAINT fk_customer
  7  FOREIGN KEY (CustomerID)
  8  REFERENCES Customer(CustomerID),
  9  InvoiceNo NUMBER NOT NULL,
 10  CONSTRAINT fk_invoice
 11  FOREIGN KEY (InvoiceNo)
 12  REFERENCES Invoice(InvoiceNo));

Table created.
```

Figure 13: Creating the Orders table.

```
SQL> DESC Orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDERID                                   NOT NULL NUMBER
 ORDERDATE                                 NOT NULL DATE
 TOTALORDER                                NOT NULL NUMBER
 CUSTOMERID                                NOT NULL NUMBER
 INVOICENO                                 NOT NULL NUMBER
```

Figure 14: Describing the Orders table.

5.1.7 Creating ProductOrder Table

```
SQL> CREATE TABLE ProductOrder (
  2  OrderID NUMBER,
  3  ProductID NUMBER,
  4  OrderQuantity NUMBER NOT NULL,
  5  PRIMARY KEY (OrderID, ProductID),
  6  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
  7  FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
  8  );

Table created.
```

Figure 15: Creating the ProductOrder table.

```
SQL> DESC ProductOrder;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDERID                                   NOT NULL NUMBER
 PRODUCTID                                 NOT NULL NUMBER
 ORDERQUANTITY                             NOT NULL NUMBER
```

Figure 16: Describing the ProductOrder table.

5.2 Inserting Values

5.2.1 Inserting Values into Vendor

```
SQL> INSERT INTO VENDOR (VendorID, VendorName)
  2  VALUES (1, 'Retro Electronics');

1 row created.

SQL> INSERT INTO VENDOR (VendorID, VendorName)
  2  VALUES (2, 'Jett Suppliers');

1 row created.

SQL> INSERT INTO VENDOR (VendorID, VendorName)
  2  VALUES (3, 'Krobar Distributor');

1 row created.

SQL> INSERT INTO VENDOR (VendorID, VendorName)
  2  VALUES (4, 'Nipurna Electronics');

1 row created.

SQL> INSERT INTO VENDOR (VendorID, VendorName)
  2  VALUES (5, 'User Suppliers');

1 row created.

SQL> INSERT INTO VENDOR (VendorID, VendorName)
  2  VALUES (6, 'Ender Electronics');

1 row created.

SQL> INSERT INTO VENDOR (VendorID, VendorName)
  2  VALUES (7, 'Lalit Traders');

1 row created.
```

Figure 17: Inserting Values into Vendor.

```
SQL> SELECT * FROM Vendor;

  VENDORID VENDORNAME
---------- --------------------------------------------------
         1 Retro Electronics
         2 Jett Suppliers
         3 Krobar Distributor
         4 Nipurna Electronics
         5 User Suppliers
         6 Ender Electronics
         7 Lalit Traders

7 rows selected.
```

Figure 18: Showing the inserted Values.

5.2.2 Inserting Values into Discount

```
SQL> INSERT INTO Discount (CustomerCategory, Discount)
  2  VALUES
  3  ('R', 0);

1 row created.

SQL> INSERT INTO Discount (CustomerCategory, Discount)
  2  VALUES
  3  ('S', 5);

1 row created.

SQL> INSERT INTO Discount (CustomerCategory, Discount)
  2  VALUES
  3  ('VIP', 10);

1 row created.

SQL> Select * FROM Discount;

CUSTOMERCATEGORY                DISCOUNT
------------------------------ ----------
R                                       0
S                                       5
VIP                                    10
```

Figure 19: Inserting values into Discount table and Showing the inserted Values.

5.2.3 Inserting Values into Customer

```
SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, CustomerCategory)
  2  VALUES
  3  (1, 'Tyler Shift', 'Kathmandu', 'VIP');

1 row created.

SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, CustomerCategory)
  2  VALUES
  3  (2, 'Lana Del Rai', 'Dharan', 'S');

1 row created.

SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, CustomerCategory)
  2  VALUES
  3  (3, 'Harry Singh', 'Dharan', 'R');

1 row created.

SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, CustomerCategory)
  2  VALUES
  3  (4, 'John Smith', 'Pokhara', 'VIP');

1 row created.

SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, CustomerCategory)
  2  VALUES
  3  (5, 'Henry Stickmen', 'Nepalgunj', 'VIP');

1 row created.

SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, CustomerCategory)
  2  VALUES
  3  (6, 'Charli Lee', 'Doti', 'R');

1 row created.

SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, CustomerCategory)
  2  VALUES
  3  (7, 'Justine Bastola', 'Hetauda', 'S');

1 row created.
```

Figure 20: Inserting Values into Customer.

```
SQL> SELECT * FROM Customer;

CUSTOMERID CUSTOMERNAME                                       CUSTOMERADDRESS                                       CUSTOMERCATEGORY
---------- -------------------------------------------------- ----------------------------------------------------- ------------------------------
         1 Tyler Shift                                        Kathmandu                                             VIP
         2 Lana Del Rai                                       Dharan                                                S
         3 Harry Singh                                        Dharan                                                R
         4 John Smith                                         Pokhara                                               VIP
         5 Henry Stickmen                                     Nepalgunj                                             VIP
         6 Charli Lee                                         Doti                                                  R
         7 Justine Bastola                                    Hetauda                                               S

7 rows selected.
```

Figure 21: Showing the inserted Values.

5.2.4 Inserting values into Product

```
SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (1, 1, 'Xbox 330', 'This is an Xbox 330', 'Electronics', 8, 200);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (2, 1, 'Xbox Controller', 'This is an Xbox Controller', 'Accessories', 6, 40);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (3, 2, 'DELL XPS', 'DELL XPS is a good laptop', 'Electronics', 4, 600);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (4, 3, 'Mouse', 'This a Mouse', 'Accessories', 14, 20);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (5, 3, 'Keyboard', 'This a mechanical Keyboard', 'Accessories', 12, 35);

1 row created.

SQL> VALUES (6, 4, 'Microphone', 'This a dynamic mic', 'Accessories', 10, 50);
SP2-0734: unknown command beginning "VALUES (6,..." - rest of line ignored.
SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (6, 4, 'Microphone', 'This a dynamic mic', 'Accessories', 10, 50);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (7, 5, 'ASUS ROG STRIX G', 'This a gaming laptop with 16GB RAM', 'Electronics', 11, 900);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (8, 6, 'Racing Wheel', 'This a racing wheel for virtual racers', 'Accessories', 5, 400);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES (9, 7, 'Headphone', 'This a stereo headphone with Dolby Atmos', 'Accessories', 8, 150);

1 row created.
```

Figure 22: Inserting Values into Product.

```
SQL> UPDATE Product
  2  SET Stock = Stock + 50
  3  WHERE ProductID = 8;

1 row updated.
```

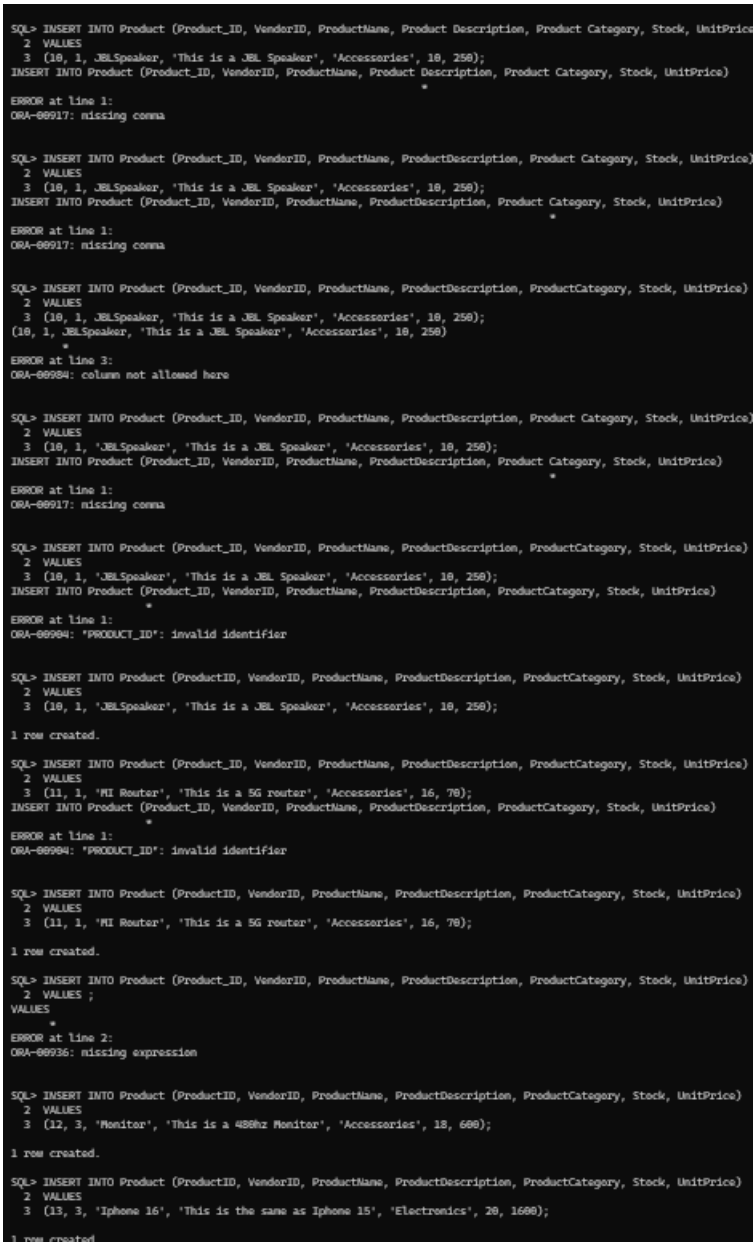
Figure 23: Updating the Stock in Product table.

```
SQL> SELECT * FROM Product;

 PRODUCTID PRODUCTNAME          PRODUCTCATEGORY                                  PRODUCTDESCRIPTION                                 STOCK  UNITPRICE   VENDORID
---------- -------------------- ------------------------------------------------ -------------------------------------------------- ---------- ---------- ----------
         1 Xbox 330             Electronics                                      This is an Xbox 330                                     8        200          1
         2 Xbox Controller      Accessories                                      This is an Xbox Controller                              6         40          1
         3 DELL XPS             Electronics                                      DELL XPS is a good laptop                               4        600          2
         4 Mouse                Accessories                                      This a Mouse                                           14         20          3
         5 Keyboard             Accessories                                      This a mechanical Keyboard                             12         35          3
         6 Microphone           Accessories                                      This a dynamic mic                                     10         50          4
         7 ASUS ROG STRIX G     Electronics                                      This a gaming laptop with 16GB RAM                     11        900          5
         8 Racing Wheel         Accessories                                      This a racing wheel for virtual racers                 55        400          6
         9 Headphone            Accessories                                      This a stereo headphone with Dolby Atmos                8        150          7

9 rows selected.
```

Figure 24: Showing the inserted Values.



```
SQL> INSERT INTO Product (Product_ID, VendorID, ProductName, Product Description, Product Category, Stock, UnitPrice
  2  VALUES
  3  (10, 1, JBLSpeaker, 'This is a JBL Speaker', 'Accesseries', 10, 250);
INSERT INTO Product (Product_ID, VendorID, ProductName, Product Description, Product Category, Stock, UnitPrice)
                                                     *
ERROR at line 1:
ORA-00917: missing comma


SQL> INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, Product Category, Stock, UnitPrice)
  2  VALUES
  3  (10, 1, JBLSpeaker, 'This is a JBL Speaker', 'Accesseries', 10, 250);
INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, Product Category, Stock, UnitPrice)
                                                                            *
ERROR at line 1:
ORA-00917: missing comma


SQL> INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES
  3  (10, 1, JBLSpeaker, 'This is a JBL Speaker', 'Accesseries', 10, 250);
(10, 1, JBLSpeaker, 'This is a JBL Speaker', 'Accesseries', 10, 250)
 *
ERROR at line 3:
ORA-00984: column not allowed here


SQL> INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, Product Category, Stock, UnitPrice)
  2  VALUES
  3  (10, 1, 'JBLSpeaker', 'This is a JBL Speaker', 'Accesseries', 10, 250);
INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, Product Category, Stock, UnitPrice)
                                                                            *
ERROR at line 1:
ORA-00917: missing comma


SQL> INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES
  3  (10, 1, 'JBLSpeaker', 'This is a JBL Speaker', 'Accesseries', 10, 250);
INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
                    *
ERROR at line 1:
ORA-00904: "PRODUCT_ID": invalid identifier


SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES
  3  (10, 1, 'JBLSpeaker', 'This is a JBL Speaker', 'Accesseries', 10, 250);

1 row created.

SQL> INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES
  3  (11, 1, 'MI Router', 'This is a 5G router', 'Accesseries', 16, 70);
INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
                    *
ERROR at line 1:
ORA-00904: "PRODUCT_ID": invalid identifier


SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES
  3  (11, 1, 'MI Router', 'This is a 5G router', 'Accesseries', 16, 70);

1 row created.

SQL> INSERT INTO Product (Product_ID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES ;
VALUES
       *
ERROR at line 2:
ORA-00936: missing expression


SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES
  3  (12, 3, 'Monitor', 'This is a 480hz Monitor', 'Accesseries', 18, 600);

1 row created.

SQL> INSERT INTO Product (ProductID, VendorID, ProductName, ProductDescription, ProductCategory, Stock, UnitPrice)
  2  VALUES
  3  (13, 3, 'Iphone 16', 'This is the same as Iphone 15', 'Electronics', 20, 1600);

1 row created.
```

Figure 25: Inserting more values into Product.

```
SQL> SELECT * FROM Product;

 PRODUCTID PRODUCTNAME          PRODUCTCATEGORY                                  PRODUCTDESCRIPTION                                 STOCK  UNITPRICE   VENDORID
---------- -------------------- ------------------------------------------------ -------------------------------------------------- ---------- ---------- ----------
         1 Xbox 330             Electronics                                      This is an Xbox 330                                     8        200          1
         2 Xbox Controller      Accessories                                      This is an Xbox Controller                              6         40          1
         3 DELL XPS             Electronics                                      DELL XPS is a good laptop                               4        600          2
         4 Mouse                Accessories                                      This a Mouse                                           14         20          3
         5 Keyboard             Accessories                                      This a mechanical Keyboard                             12         35          3
         6 Microphone           Accessories                                      This a dynamic mic                                     10         50          4
         7 ASUS ROG STRIX G     Electronics                                      This a gaming laptop with 16GB RAM                     11        900          5
         8 Racing Wheel         Accessories                                      This a racing wheel for virtual racers                 55        400          6
         9 Headphone            Accessories                                      This a stereo headphone with Dolby Atmos                8        150          7
        10 JBLSpeaker           Accessories                                      This is a JBL Speaker                                  10        250          1
        11 MI Router            Accessories                                      This is a 5G router                                    16         70          1

 PRODUCTID PRODUCTNAME          PRODUCTCATEGORY                                  PRODUCTDESCRIPTION                                 STOCK  UNITPRICE   VENDORID
---------- -------------------- ------------------------------------------------ -------------------------------------------------- ---------- ---------- ----------
        12 Monitor              Accessories                                      This is a 480hz Monitor                                18        600          3
        13 Iphone 16            Electronics                                      This is the same as Iphone 15                          20       1600          3

13 rows selected.
```

Figure 26: Showing the inserted values.

5.2.5 Inserting Values into Invoice

```
SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (101, 'Card', 240, 240);

1 row created.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (102, 'E-Wallet', 240, 240);

1 row created.

SQL> DELETE FROM INVOICE;

2 rows deleted.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (101, 'Card', 240, 240);

1 row created.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (102, 'E-Wallet', 620, 589);

1 row created.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (103, 'Cash', 800, 720);

1 row created.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (104, 'Cash', 350, 315);

1 row created.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (105, 'E-Wallet', 35, 35);

1 row created.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (106, 'Card', 1400, 1330);

1 row created.

SQL> INSERT INTO Invoice (InvoiceNo, PaymentOption, TotalAmount, InvoiceTotal)
  2  VALUES
  3  (107, 'Card', 40, 36);

1 row created.
```

Figure 27: Inserting Values into Invoice.

```
 INVOICENO PAYMENTOPTION                                                                                          TOTALAMOUNT INVOICETOTAL
---------- ---------------------------------------------------------------------------------------------------- ----------- ------------
       101 Card                                                                                                         240          240
       102 E-Wallet                                                                                                     620          589
       103 Cash                                                                                                         800          720
       104 Cash                                                                                                         350          315
       105 E-Wallet                                                                                                      35           35
       106 Card                                                                                                        1400         1330
       107 Card                                                                                                          40           36

7 rows selected.
```

Figure 28: Showing the inserted Values.

5.2.6 Inserting Values into Orders

```
SQL> INSERT INTO Orders (OrderID, OrderDate, TotalOrder, CustomerID, InvoiceNo)
  2  VALUES
  3  (1,  TO_DATE ('03-05-2023', 'DD-MM-YY'), 3, 3, 101);

1 row created.

SQL> INSERT INTO Orders (OrderID, OrderDate, TotalOrder, CustomerID, InvoiceNo)
  2  VALUES
  3  (2,  TO_DATE ('15-05-2023', 'DD-MM-YY'), 2, 2, 102);

1 row created.

SQL> INSERT INTO Orders (OrderID, OrderDate, TotalOrder, CustomerID, InvoiceNo)
  2  VALUES
  3  (3,  TO_DATE ('15-05-2023', 'DD-MM-YY'), 1, 4, 103);

1 row created.

SQL> INSERT INTO Orders (OrderID, OrderDate, TotalOrder, CustomerID, InvoiceNo)
  2  VALUES
  3  (4,  TO_DATE ('17-05-2023', 'DD-MM-YY'), 2, 1, 104);

1 row created.

SQL> INSERT INTO Orders (OrderID, OrderDate, TotalOrder, CustomerID, InvoiceNo)
  2  VALUES
  3  (5,  TO_DATE ('02-06-2023', 'DD-MM-YY'), 1, 6, 105);

1 row created.

SQL> INSERT INTO Orders (OrderID, OrderDate, TotalOrder, CustomerID, InvoiceNo)
  2  VALUES
  3  (6,  TO_DATE ('05-06-2023', 'DD-MM-YY'), 2, 7, 106);

1 row created.

SQL> INSERT INTO Orders (OrderID, OrderDate, TotalOrder, CustomerID, InvoiceNo)
  2  VALUES
  3  (7,  TO_DATE ('08-06-2023', 'DD-MM-YY'), 1, 5, 107);

1 row created.
```

Figure 29: Inserting Values into Orders.

```
SQL> Update Orders
  2  SET CustomerID = 3 WHERE OrderID = 4;

1 row updated.
```

Figure 30:The CustomerID is later updated in the Orders table..

```
SQL> SELECT * FROM Orders;

   ORDERID ORDERDATE TOTALORDER CUSTOMERID  INVOICENO
---------- --------- ---------- ---------- ----------
         1 03/MAY/23          3          3        101
         2 15/MAY/23          2          2        102
         3 15/MAY/23          1          4        103
         4 17/MAY/23          2          1        104
         5 02/JUN/23          1          6        105
         6 05/JUN/23          2          7        106
         7 08/JUN/23          1          5        107
```

Figure 31: Showing the inserted Values.

```
SQL> UPDATE ORDERS
  2  SET OrderDate = TO_DATE('03-08-2023', 'DD-MM-YYYY')
  3  WHERE OrderID = 7;

1 row updated.
```

Figure 32: The date in order Table is later updated.

```
SQL> SELECT * FROM ORDERS;

   ORDERID ORDERDATE TOTALORDER CUSTOMERID  INVOICENO
---------- --------- ---------- ---------- ----------
         1 03/MAY/23          3          3        101
         2 15/MAY/23          2          2        102
         3 15/MAY/23          1          4        103
         4 17/MAY/23          2          3        104
         5 02/JUN/23          1          6        105
         6 05/JUN/23          2          7        106
         7 03/AUG/23          1          5        107

7 rows selected.
```

Figure 33: Showing the updated table.

5.2.7 Inserting Values into ProductOrder

```
SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (1, 1, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (1, 2, 2);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (2, 3, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (2, 4, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (3, 7, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (4, 1, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (4, 9, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (5, 5, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (6, 8, 2);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (6, 3, 1);

1 row created.

SQL> INSERT INTO ProductOrder (OrderID, ProductID, OrderQuantity)
  2  VALUES
  3  (7, 2, 1);
```

Figure 34: Inserting Values into ProductOrder.

```
SQL> SELECT * FROM ProductOrder;

   ORDERID  PRODUCTID ORDERQUANTITY
---------- ---------- -------------
         1          1             1
         1          2             2
         2          3             1
         2          4             1
         3          7             1
         4          1             1
         4          9             1
         5          5             1
         6          8             2
         6          3             1
         7          2             1
```

Figure 35: Showing all the Inserted Values.

6. Database Querying

6.1 Information Query

a. List all the customers that are also staff of the company.

```
SQL> SELECT *
  2  FROM Customer
  3  WHERE CustomerCategory = 'S';

CUSTOMERID CUSTOMERNAME                                       CUSTOMERADDRESS                                       CUSTOMERCATEGORY
---------- -------------------------------------------------- ----------------------------------------------------- ------------------------------
         2 Lana Del Rai                                       Dharan                                                S
         7 Justine Bastola                                    Hetauda                                               S
```

Figure 36: Customers who are the staffs of the company.

b. List all the orders made for any particular product between the dates 01-05-2023 till 28- 05-2023.

```
SQL> SELECT * FROM Orders
  2  WHERE OrderDate BETWEEN TO_DATE('01-05-2023', 'DD-MM-YYYY') AND TO_DATE('28-05-2023', 'DD-MM-YYYY');

   ORDERID ORDERDATE TOTALORDER CUSTOMERID  INVOICENO
---------- --------- ---------- ---------- ----------
         1 03/MAY/23          3          3        101
         2 15/MAY/23          2          2        102
         3 15/MAY/23          1          4        103
         4 17/MAY/23          2          3        104
```

Figure 37: Orders made for products between May 1 and May 28 of 2023.

c. List all the customers with their order details and also the customers who have not ordered any products yet.

```
SQL> SELECT C.CustomerID, C.CustomerName, O.OrderDate, O.OrderID, O.TotalOrder, O.InvoiceNo
  2  FROM Customer C LEFT JOIN Orders O ON C.CustomerID = O.CustomerID;

CUSTOMERID CUSTOMERNAME                                       ORDERDATE    ORDERID TOTALORDER  INVOICENO
---------- -------------------------------------------------- --------- ---------- ---------- ----------
         3 Harry Singh                                        03/MAY/23          1          3        101
         2 Lana Del Rai                                       15/MAY/23          2          2        102
         4 John Smith                                         15/MAY/23          3          1        103
         3 Harry Singh                                        17/MAY/23          4          2        104
         6 Charli Lee                                         02/JUN/23          5          1        105
         7 Justine Bastola                                    05/JUN/23          6          2        106
         5 Henry Stickmen                                     08/JUN/23          7          1        107
         1 Tyler Shift

8 rows selected.
```

Figure 38: Customers with their order details including the customers with no orders.

d. List all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.

```
SQL> SELECT * FROM PRODUCT
  2  WHERE ProductName LIKE '_a%'
  3  AND STOCK > 50;

 PRODUCTID PRODUCTNAME          PRODUCTCATEGORY                                   PRODUCTDESCRIPTION                                    STOCK  UNITPRICE   VENDORID
---------- -------------------- ------------------------------------------------- ----------------------------------------------------- ---------- ---------- ----------
         8 Racing Wheel         Accessories                                       This a racing wheel for virtual racers                    55        400          6
```

Figure 39: Products that have more than 50 stock quantity and the second letter in their word is "a".

e. Find out the customer who has ordered recently.

```
SQL> SELECT *
  2  FROM
  3  (SELECT c.CustomerID,c.CustomerName,MAX(o.OrderDate) AS RecentOrderDate
  4  FROM Customer c
  5  JOIN Orders o ON c.CustomerID = o.CustomerID
  6  GROUP BY c.CustomerID, c.CustomerName
  7  ORDER BY RecentOrderDate DESC)
  8  WHERE ROWNUM = 1;

CUSTOMERID CUSTOMERNAME                                       RECENTORD
---------- -------------------------------------------------- ---------
         5 Henry Stickmen                                     08/JUN/23
```

Figure 40: Listing the customer with the most recent order.

6.2 Transaction Query

a. Show the total revenue of the company for each month.

```
SQL> SELECT TO_CHAR(OrderDate, 'MM-YYYY') AS Month,
  2  SUM(T.InvoiceTotal) AS TotalRevenue
  3  FROM Orders o
  4  JOIN Invoice T ON o.InvoiceNo = T.InvoiceNo
  5  GROUP BY TO_CHAR(OrderDate, 'MM-YYYY');

MONTH   TOTALREVENUE
------- ------------
06-2023         1401
05-2023         1864
```

Figure 41: Listing the total revenue for each month.

b. Find those orders that are equal or higher than the average order total value.

```
SQL> SELECT * FROM Orders o
  2  WHERE o.TotalOrder >= (SELECT AVG(TotalOrder)
  3  FROM Orders);

   ORDERID ORDERDATE TOTALORDER CUSTOMERID  INVOICENO
---------- --------- ---------- ---------- ----------
         1 03/MAY/23          3          3        101
         2 15/MAY/23          2          2        102
         4 17/MAY/23          2          3        104
         6 05/JUN/23          2          7        106

SQL>
```

Figure 42: Listing the orders that are equal or higher than average order total value.

c. List the details of vendors who have supplied more than 3 products to the company.

```
SQL> SELECT v.VendorID, v.VendorName,
  2  COUNT(p.ProductID) AS ProductCount
  3  FROM Vendor v
  4  JOIN Product p ON v.VendorID = p.VendorID
  5  GROUP BY v.VendorID, v.VendorName
  6  HAVING
  7  COUNT(p.ProductID) > 3;

  VENDORID VENDORNAME                                        PRODUCTCOUNT
---------- -------------------------------------------------- ------------
         3 Krobar Distributor                                            4
         1 Retro Electronics                                             4
```

Figure 43: Showing the Vendors who have provided more than 3 products.

d. Show the top 3 product details that have been ordered the most.

```
SQL> SELECT ROWNUM AS RANK,  ProductID, ProductName, OrderQuantity
  2  FROM
  3  (SELECT p.ProductID, p.ProductName, SUM(od.OrderQuantity) AS OrderQuantity
  4  FROM Product p
  5  JOIN ProductOrder od ON p.ProductID = od.ProductID
  6  GROUP BY P.ProductID, P.ProductName
  7  ORDER BY OrderQuantity DESC)
  8  WHERE ROWNUM <= 3;

      RANK  PRODUCTID PRODUCTNAME          ORDERQUANTITY
---------- ---------- -------------------- -------------
         1          2 Xbox Controller                  3
         2          3 DELL XPS                         2
         3          1 Xbox 330                         2
```

Figure 44: Showing the 3 most ordered products.

e. Find out the customer who has ordered the most in August with his/her total spending on that month.

```
SQL> SELECT *
  2  FROM (
  3  SELECT C.CustomerID, C.CustomerName,
  4  SUM(I.InvoiceTotal) AS AmountSpent,
  5  TO_CHAR(OrderDate, 'MM-YYYY') AS Month
  6  FROM Orders o
  7  JOIN Customer C ON o.CustomerID = c.CustomerID
  8  JOIN Invoice i ON o.InvoiceNo = i.InvoiceNo
  9  WHERE OrderDate BETWEEN TO_DATE('01-08-2023', 'DD-MM-YYYY') AND TO_DATE('30-08-2023', 'DD-MM-YYYY')
 10  GROUP BY
 11  C.CustomerID, C.CustomerName,
 12  TO_CHAR(OrderDate, 'MM-YYYY')
 13  ORDER BY
 14  AmountSpent DESC
 15  )
 16  WHERE ROWNUM = 1;

CUSTOMERID CUSTOMERNAME                                       AMOUNTSPENT MONTH
---------- -------------------------------------------------- ----------- -------
         5 Henry Stickmen                                              36 08-2023
```

Figure 45: Showing the Customer with the most orders in the month of August.

7. Critical Evaluation

In conclusion this coursework describes a thorough and well-considered plan for creating an e-commerce database system named "Gadget Emporium," which is an online marketplace that specializes in electronic devices and accessories. The proposed system contains features such as order processing, product management, categorization of customers, payment processing and discount structures. It offers a clear understanding of business rules and requirement essentials for the operation of an e-commerce platform. This is essential to create an efficient design and implementation of the database.

This coursework has taught me the importance of normalization in a database. Normalizing is necessary as it helps reduce redundancy which makes data organization easier. It makes maintaining a database easier. It has also helped understand Entity Relationship Diagrams. The Entity Relationship Diagram (ERD) helps visualize the database structure which helps understand the design and layout of the database. Before creating the database, identifying the entities and attributes is crucial, ERD helps in identifying the entities and attributes which lays a foundation in creating a database. This coursework also helped learn the different queries necessary to create a database, insert values into the database and extract the necessary values into the database. Furthermore, entry of unintentional data gave me a learning opportunity to research and fix the incorrect data that had been inserted into the database.

8. Drop Tables

```
SQL> DROP TABLE ProductOrder;

Table dropped.

SQL> DROP TABLE Product;

Table dropped.

SQL> DROP TABLE Vendor;

Table dropped.

SQL> DROP TABLE Order;
DROP TABLE Order
           *
ERROR at line 1:
ORA-00903: invalid table name


SQL> DROP TABLE Orders;

Table dropped.

SQL> DROP TABLE Invoice;

Table dropped.

SQL> DROP TABLE Customer;

Table dropped.

SQL> DROP TABLE Discount;

Table dropped.

SQL> SELECT * FROM TAB;

no rows selected
```

Figure 46: Dropping all the Tables.

9. Creation of Dump File

```
EXP-00056: ORACLE error 1017 encountered
ORA-01017: invalid username/password; logon denied
Username: "Gadget Emporium"
Password:

Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user Gadget Emporium
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user Gadget Emporium
About to export Gadget Emporium's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export Gadget Emporium's tables via Conventional Path ...
. . exporting table                       CUSTOMER          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                       DISCOUNT          3 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        INVOICE          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                         ORDERS          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        PRODUCT         13 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                   PRODUCTORDER         11 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        VENDOR          7 rows exported
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.

D:\>
```

Figure 47: The dump file is created in the D drive.

References

Rouse, M., 2023. *Normalization.* [Online] Available at: https://www.techopedia.com/definition/1221/normalization [Accessed 8 January 2024].