LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: Atal Gyawali

London Met ID: 22067674

Submitted To : Yunisha Bajracharya

Assignment Submission Date: 01/15/2024

Word Count:

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.

Contents

1.Introduction 1

1.1 Introduction of the Business and Its Forte 1

1.2 Description of Current Business Activities and Operations 1

1.3 Aims and Objectives 1

1.4 Business Rules 2

1.5 Assumptions 2

1.5 Entities and Attributes 3

Customer 3

Order 3

Products 4

2.Initial ERD 6

3.Normalization 8

Un-Normal Form 8

First Normal Form 8

Second Normal Form 9

Third Normal Form 10

4.Final ERD 12

5.Implementation 14

5.1 Creating Entities and Establishing Relations. 14

5.1.1 Customer Category 14

5.1.2 Customer 16

5.1.3 Orders 17

5.1.4 Invoice 18

5.1.5 OrderDetails 20

5.1.6 PurchasedProduct 21

5.1.7 Vendor 23

5.1.8 ProductDetails 24

5.1.9 Tables in the Database. 26

5.2 Inserting Data in the tables and verifying. 27

Inserting Data into CustomerCategory table. 27

Inserting Data in Customer table. 28
Inserting Data in Orders table. 29
Inserting Data in Invoice table. 30
Inserting Data in OrderDetails table. 31
Inserting Data in PurchasedProduct table. 32
Inserting Data in Vendor table. 33
Inserting Data in ProdcutDetails table. 34
6.Database Querying 35
1.1 Information Query 35
6.2 Transaction Query 37
7.Critical Evaluation 40
8.References 41

Table of figures

Figure 1: Initial Entity Relationship Diagram. 6
Figure 2: Final Entity Relationship Diagram. 12
Figure 3: Creating CustomerCategory Table. 14
Figure 4: Describe CustomerCategory Table. 15
Figure 5: Creating Customer Table. 16
Figure 6: Describe Customer Table. 17
Figure 7: Creating Orders Table. 18
Figure 8: Describe Orders Table. 18
Figure 9: Creating Invoice Table. 19
Figure 10: Describe Invoice Table. 20
Figure 11: Creating OrderDetails Table. 21
Figure 12: Describe OrderDetails Table. 21
Figure 13: Creating PurchasedProduct Table. 22
Figure 14:Describe PurchasedProduct Table. 23
Figure 15: Creating Vendor Table. 24
Figure 16: Describe Vendor Table. 24
Figure 17: Creating ProductDetails Table. 25
Figure 18: Describe ProductDetails Table. 26
Figure 19: List of tables in the database 26
Figure 20: Inserting values into CustomerCategory table. 27
Figure 21: Verifying the data in CustomerCategory table. 27
Figure 22: Inserting values into Customer Table. 28
Figure 23:Verifying the data in Customer table. 28
Figure 24 Inserting values into Orders Table. 29
Figure 25 Verifying the data in Orders table. 29
Figure 26 Inserting values into Invoice Table. 30
Figure 27 Verifying the data in Orders table. 30
Figure 28 Inserting values into OrderDetails Table. 31
Figure 29 Verifying the data in OrderDetails table. 31
Figure 30 Inserting values into PurchasedProdcut Table. 32
Figure 31 Verifying the data in PurchasedProduct table. 32
Figure 32 Inserting values into Vendor Table. 33
Figure 33 Verifying the data in Vendor table. 33
Figure 34 Inserting values into ProductDetails Table. 34
Figure 35 Verifying the data in ProductDetails table. 34
Figure 36: Information Query Q no 1 Ans. 35
Figure 37: Information Query Q no 2 Ans. 35
Figure 38: Information Query Q no 3 Ans. 36
Figure 39: Information Query Q no 4 Ans. 36

Figure 40: Information Query Q no 5 Ans. 37
Figure 41:Transaction Query Q no 1 Ans. 37
Figure 42:Transaction Query Q no 2 Ans. 38
Figure 43:Transaction Query Q no 3 Ans. 38
Figure 44:Transaction Query Q no 4 Ans. 39
Figure 45:Transaction Query Q no 5 Ans. 39

Table of Tables.

Table 1: Customer table before normalization..... 3
Table 2: Order table before normalization..... 4
Table 3: Product table before normalization..... 5
Table 4: Customer Category table before normalization..... 14
Table 5:Customer table after normalization..... 16
Table 6: Order table after normalization..... 17
Table 7: Invoice table after normalization..... 19
Table 8: Orderdetails table after normalization..... 20
Table 9: PurchasedProduct table after normalization..... 22
Table 10: Vendor table after normalization..... 23
Table 11: ProductDetails after normalization..... 25

1.Introduction

1.1 Introduction of the Business and Its Forte

The new online marketplace "Gadget Emporium" was started by Mr. John, an electronics enthusiast and businessman. The company, which specializes in electrical equipment and accessories, wants to offer a wide range of goods to both individual customers and corporate entities. Offering a large selection of top-notch electronic devices and accessories, Gadget Emporium establishes itself as a one-stop shop for professionals and aficionados of technology.

1.2 Description of Current Business Activities and Operations

The company functions as an internet store where clients can peruse, choose, and buy electrical devices and accessories. Product management, order processing, vendor management, real-time inventory control, payment processing, and automated invoice production are among the main tasks. Clients are divided into three categories: VIP, Staff, and Regular. Each group is entitled to a set percentage off. Different items are supplied by vendors, and inventory is controlled to avoid overselling.

1.3 Aims and Objectives

The main goal of this coursework is to design and implement a strong database system for Mr. John’s online store that efficiently manages electronic devices, customers, orders, and invoices. This database should help the employees and admin of the company to keep records of everything happening in the store. The objective of

The objectives are given below:

- To develop a schema for comprehensive order information and handle orders with multiple products.
- To create a customer database with categories and addresses.
- Handle different payment options securely and efficiently.

1.4 Business Rules

Some of the business rule of the “Gadget Emporium” online marketplace are given below:

- The system should handle all the details of electronic gadgets and accessories.
- The system should be able to keep track of all its customers.
- Customers can browse and purchase one or more electronic gadgets online and the system must record the details of each order.
- The records of vendors or suppliers that provide gadgets and accessories are also maintained.
- The system should track real-time product availability to prevent overselling and maintain accurate stock level.
- There should be multiple payment options cash on delivery, card or e-wallet.
- Invoice should be created after the customer checks out the order and should store the details of order.

1.5 Assumptions

Here are some of the assumptions made while designing the databas.

- Customer can make multiple order.
- Customer can either pay online or cash on delivery for their purchase.
- Customer can get discounts based on their category.
- The owners of the system should hire technicians to maintain the database properly.
- The system should be regularly updated and maintained.

1.5 Entities and Attributes

We can initially define the following entities and their properties based on the given question and business rules. These are not the complete entities, during the normalization process they can even produce more entities.

Customer

Attributes	Data Type	Constraints	Description
CustomerID	INTEGER	Primary Key	This field stores the Id of the customer.
CustomerName	VARCHAR		This field stores the name of the customer.
Address	VARCHAR		This field stores the address of the customer.
CustomerCategoryID	INTEGER		This field stores the Customer Category ID.
CustomerCategory	VARCHAR		This field stores the category in which a customer belongs.
DiscountRate	DECIMAL		This field stores discount rates a customer can get on the basis of their category.

Table 1: Customer table before normalization

Order

Attributes	Data Type	Constraints	Description
OrderID	Integer	Primary Key	This field stores the order ID which is the primary key of this table.
DeliveryStatus	VARCHAR		This field stores if the order is delivered to the customer or not.
OrderDate	DATE		This field stores the date on which the order was made.

TotalOrderAmount	DECIMAL		This field stores the total amount of the ordered product.
PurasedQuantity	INTEGER		This field stores the quantity of the ordered product.
PurasedQuantityPrice	DECIMAL		This field stores the total amount of one purchased product.
InvoiceID	INTEGER		This field stores the invoice id.
PaymentOption	VARCHAR		This field stores the method used for payment.
PaymentStatus	VARCHAR		This field stores the payment status (paid, not paid) of the order.
TotalPaidAmount	DECIMAL		This field stores the total amount paid by the customer after discount.
DiscountAmount	DECIMAL		This field stores the discount amount.

Table 2: Order table before normalization

Products

Attributes	Data Type	Constraints	Description
ProductID	INTEGER	Primary Key	This field stores the Id of the customer.
ProductName	VARCHAR		This field stores the name of the customer.
Description	VARCHAR		This field stores the address of the customer.
ProductCategory	VARCHAR		This field stores the category in which a customer belongs.
Price	DECIMAL		This field stores the unit price of the product.
StockLevel	INTEGER		This field stores the number of products left in stock.
VendorID	INTEGER		This field stores the vendor id.
Vendor	VARCHAR		This field stores the name of the vendor

VendorContact	INTEGER		This field stores contact details of the vendor.

Table 3: Product table before normalization

2.Initial ERD

The Entity Relational Model is like a blueprint for organizing information in a database. It helps figure out what things, or entities, should be in the database and how they're connected. This model lays out the structure of the database in a visual way, showing how different pieces of information relate to each other. The Entity Relationship Diagram, which is part of this model, is like a map explaining how these different things in the database are linked. These models are used to represent real-life things like people, cars, or companies, and how they're connected in the database. Essentially, the ER Diagram gives you a clear picture of how everything is structured in the database (GeeksforGeeks, 2023).

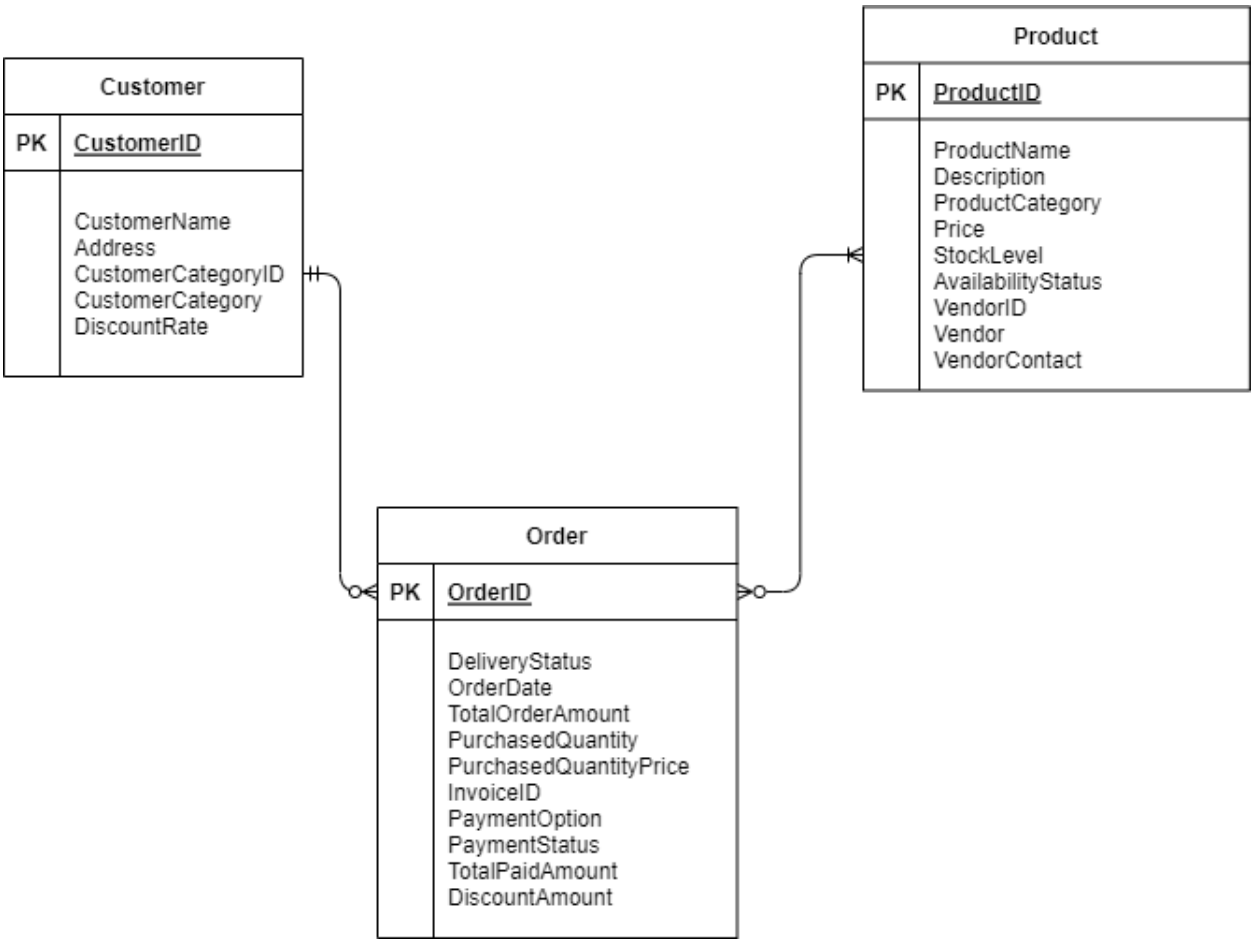

Figure 1: Initial Entity Relationship Diagram.

The above figure represents the Initial entity relationship diagram before normalization. Here we have three entities Customer, Order and Product with their attributes.

- A customer can make multiple orders and can choose to not make any orders at all.
- A single order belongs to only one customer.
- An order can have multiple or single product in it.
- A product can belong to multiple or none of the orders.

3.Normalization

Normalization is the process of organizing data into tables in such a way that results of using the database are always unambiguous and as intended. The primary goal of normalization is to eliminate data anomalies and ensure data integrity by breaking down large tables into smaller, related tables (Anon., 2023).

Un-Normal Form

The following table is the Un Normal Form:

Customer(CustomerID, CustomerName, Address, CustomerCategoryID, CustomerCategory, DiscountRate {OrderdID, DeliveryStatus, OrderDate, TotalOrderAmount, InvoiceID, PaymentOption, PaymentStatus, TotalPaidAmount, DiscountAmount {ProductID, PurchasedQuantity, PurchasedQuantityPrice, ProductName, Description, ProductCategory, Price, StockLevel, AvailabiltiyStatus, VendorID, Vendor, VendorContact } })

All the repeating groups in this relation are within {}.

All the attributes with repeating groups are included in a single relation.

First Normal Form

Customer-1 (CustomerID, CustomerName, Address, CustomerCategoryID, CustomerCategory, DiscountRate)

Order-1 (CustomerID, OrderdID, DeliveryStatus, OrderDate, TotalOrderAmount, InvoiceID, PaymentOption, PaymentStatus, TotalPaidAmount, DiscountAmount)

Product-1 (CustomerID, OrderdID, ProductID, PurchasedQuantity, PurchasedQuantityPrice, ProductName, Description, ProductCategory, Price, StockLevel, AvailabiltiyStatus, VendorID, Vendor, VendorContact)

All the repeating groups have been removed, so that there is only a single value at the intersection of each row and column of the relation.

Second Normal Form

Separating Composite keys

From Customer-1

CustomerID -> CustomerName, Address, CustomerCategoryID, CustomerCategory, DiscountRate

From Order-1

CustomerID, OrderID -> X

OrderID -> DeliveryStatus, OrderDate, TotalOrderAmount, InvoiceID, PaymentOption, PaymentStatus, TotalPaidAmount, DiscountAmount

From Product-1

CustomerID, OrderdID, ProductID -> PurchasedQuantity, PurchasedQuantityPrice

ProductID -> ProductName, Description, ProductCategory, Price, StockLevel, AvailabiltiyStatus, VendorID,Vendor, VendorContact

Final Second Normal Form (2NF)

Customer-2(CustomerID, CustomerName, Address, CustomerCategoryID, CustomerCategory, DiscountRate)

Order-2 (CustomerID, OrderdID)

OrderDetails-2 (OrderID, DeliveryStatus, OrderDate, TotalOrderAmount, InvoiceID, PaymentOption, PaymentStatus,TotalPaidAmount, DiscountAmount)

PurchaseProduct-2 (CustomerID, OrderdID, ProductID, PurchasedQuantity, PurchasedQuantityPrice)

ProductDetails-2 (ProductID, ProductName, Description, ProductCategory, Price, StockLevel, AvailabiltiyStatus, VendorID, Vendor, VendorContact)

All the partial dependencies of the relation have been removed and the relation with only key attributes have been established. Hence, it's in first normal form.

Third Normal Form

From Customer-2

CustomerID -> CustomerName, Address, CustomerCategoryID, CustomerCategory, DiscountRate

CustomerCategoryID -> CustomerCategory, DiscountRate

From OrderDetails-2

OrderID -> DeliveryStatus, OrderDate, TotalOrderAmount, InvoiceID, PaymentOption, PaymentStatus, TotalPaidAmount, DiscountAmount

InvoiceID -> PaymentOption, PaymentStatus, TotalPaidAmount, DiscountAmount

From ProductDetails-2

ProductID -> ProductName, Description, ProductCategory, Price, StockLevel, AvailabiltiyStatus, VendorID, Vendor, VendorContact

VendorID -> Vendor, VendorContact

Final Third Normal form

Customer-3(CustomerID, CustomerName, Address, CustomerCategoryID*)

CustomerCategory-3(CustomerCategoryID, CustomerCategory, DiscountRate)

Order-3(OrderID,CustomerID)

OrderDetails-3 (OrderID, DeliveryStatus, OrderDate, TotalOrderAmount, InvoiceID*)

Invoice-3(InvoiceID, PaymentOption, PaymentStatus,TotalPaidAmount, DiscountAmount)

PurchaseProduct-3(CustomerID, OrderdID, ProductID, PurchasedQuantity, PurchasedQuantityPrice)

ProductDetails-3(ProductID, ProductName, Description, ProductCategory, Price, StockLevel, AvailabiltiyStatus, VendorID*)

Vendor-3 (VendorID, Vendor, VendorContact)

All the transitive dependencies have been removed from the relation. Hence, its in third normal form.

4.Final ERD

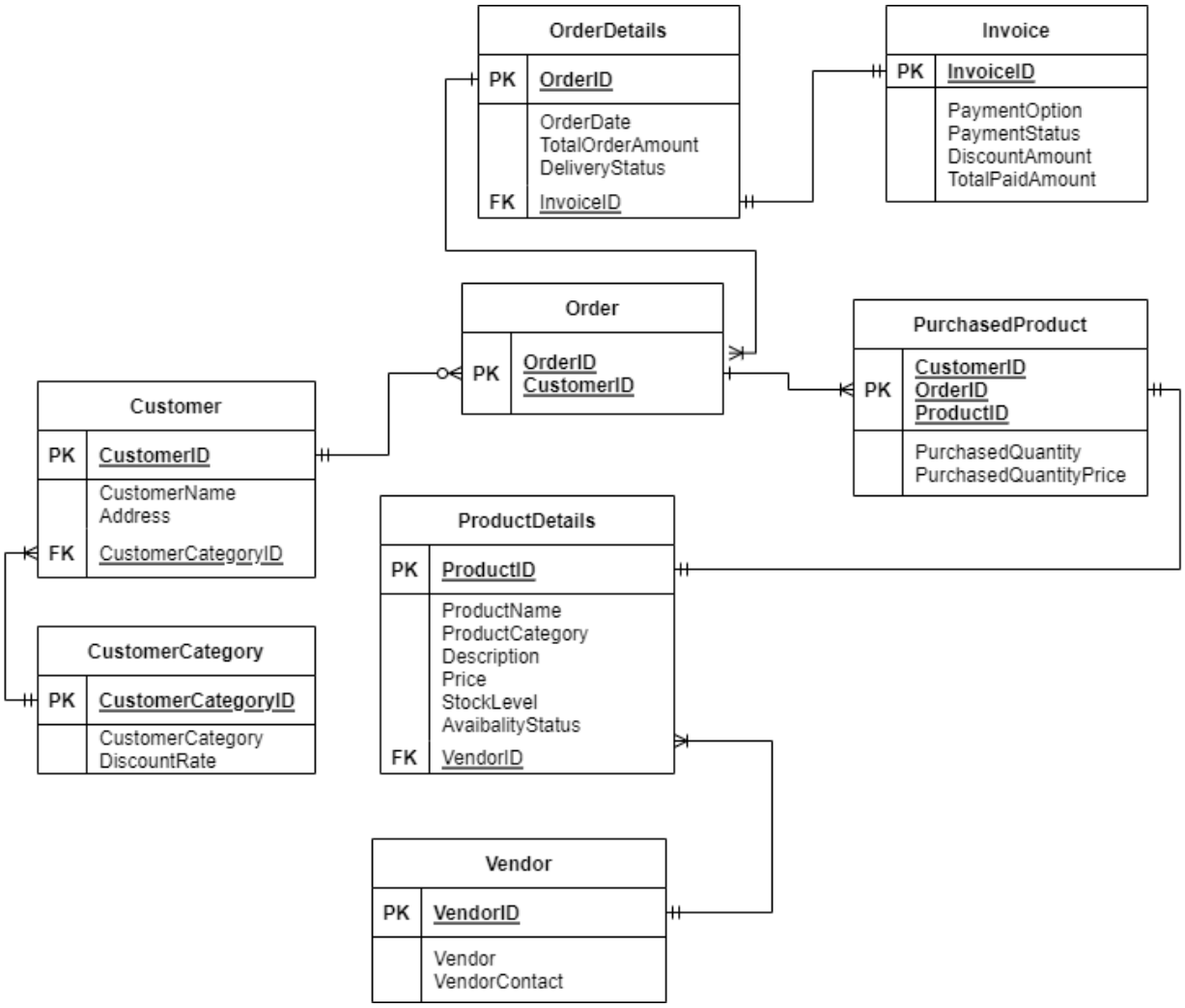

Figure 2: Final Entity Relationship Diagram.

The above diagram shows the Final Entity Relationship Diagram after normalization. Here we can see that after normalization the three initial entities has been separated into eight different entities, but the number of attributes has stayed the same. In this ERD:

- A customer belongs to only one category, but a single category can have one or multiple customers.
- A single customer can make multiple orders, but a single order belongs to only one customer.
- A single order has multiple purchased products, but a single purchased product belongs to only one order. (One product can belong in multiple

order but one product that has been purchased can only belong to one order).

- A single product can have a unique product detail and a unique detail will belong to only one product.
- A single product can have only one vendor, but a single vendor can supply multiple products.
- A single order can have a single order detail, but the same details can be found in multiple orders.
- A single order can have a single invoice and a single invoice belongs to only one order.

5.Implementation

5.1 Creating Entities and Establishing Relations.

5.1.1 Customer Category

The Customer Category table after normalization is given below:

Attributes	Data Type	Constraints	Description
CustomerCategoryID	INTEGER	Primary Key	This column stores the Id of the customer category.
CustomerCategory	VARCHAR2(15)	NOT NULL	This column stores name of the customer category on the basis of customer category Id.
DiscountRate	DECIMAL	NOT NULL	This column stores discount rates a customer can get on the basis of their category.

Table 4: Customer Category table before normalization

Creating Customer Category table in SQL.

Here, I have created the Customer Category table using the (Create table) command. It consists a primary key CustomerCategory and stores the data related to the category of the customer.

```
SQL> SPOOL D:\Database\Atal.sql;
SQL> CREATE TABLE CustomerCategory(
  2  CustomerCategoryID INT NOT NULL PRIMARY KEY,
  3  CustomerCategory VARCHAR2(15) NOT NULL,
  4  DiscountRate DECIMAL NOT NULL);

Table created.

SQL>
```

Figure 3: Creating CustomerCategory Table.

After creating the table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
Table created.

SQL> Desc CustomerCategory;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
---
 CUSTOMERCATEGORYID                        NOT NULL NUMBER(38)
 CUSTOMERCATEGORY                          NOT NULL VARCHAR2(15)
 DISCOUNTRATE                              NOT NULL NUMBER(38)

SQL>
```

Figure 4: Describe CustomerCategory Table.

5.1.2 Customer

The Customer table after normalization is given below:

Attributes	Data Type	Constraints	Description
CustomerID	INTEGER	Primary Key	This column stores the Id of the customer.
CustomerName	VARCHAR2(30)	NOT NULL	This column stores the full name of the customer.
Address	VARCHAR2(50)	NOT NULL	This column stores the
CustomerCategoryID	INTEGER	FOREIGN KEY	This column stores the CustomerCategoryID which is linked to the column by the same name in the CustomerCategory table.

Table 5:Customer table after normalization

Creating Customer table in SQL.

Here, I have created the Customer table using the (Create table) command. It consists of a primary key CustomerID, a foreign key CustomerCategoeyID and stores the data related to the details of the customer.

```
SQL> CREATE TABLE Customer(
  2  CustomerID INT NOT NULL PRIMARY KEY,
  3  CustomerName VARCHAR2(30) NOT NULL,
  4  Address VARCHAR2(50) NOT NULL,
  5  CustomerCategoryID INT NOT NULL,
  6  FOREIGN KEY (CustomerCategoryID) REFERENCES CustomerCategory(CustomerCa
tegoryID));

Table created.

SQL>
```

Figure 5: Creating Customer Table.

After creating the customer table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
SQL> Desc Customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
----
 CUSTOMERID                                NOT NULL NUMBER(38)
 CUSTOMERNAME                              NOT NULL VARCHAR2(30)
 ADDRESS                                   NOT NULL VARCHAR2(50)
 CUSTOMERCATEGORYID                        NOT NULL NUMBER(38)

SQL>
```

Figure 6: Describe Customer Table.

5.1.3 Orders

The Orders table after normalization is given below:

Attributes	Data Type	Constraints	Description
OrderID	INTEGER	Primary Key (Composite)	This column stores the Id of the Order.
CustomerID	INTEGER	Primary Key (Composite)	This column stores the customer id.

Table 6: Order table after normalization.

Creating Orders table in SQL.

I have created the Orders table using the (Create table) command. CustomerID and OrderID forms a composite primary key in this table.

```
SQL> CREATE TABLE Orders(
  2  OrderID INT NOT NULL,
  3  CustomerID INT NOT NULL,
  4  PRIMARY KEY (OrderID, CustomerID));

Table created.

SQL>
```

Figure 7: Creating Orders Table.

After creating the Orders table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
SQL> Desc Orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
----
 ORDERID                                   NOT NULL NUMBER(38)
 CUSTOMERID                                NOT NULL NUMBER(38)

SQL>
```

Figure 8: Describe Orders Table.

5.1.4 Invoice

The Invoice table after normalization is given below:

Attributes	Data Type	Constraints	Description
InvoiceID	INTEGER	Primary Key	This column stores the Id of the Invoice.
PaymentOption	VARCHAR2(30)	NOT NULL	This column stores the way customer made payment. Like cash on delivery or online payment.
PaymentStatus	VARCHAR2(30)	NOT NULL	This column stores the status of the

			payment. If the customer has paid or not.
DiscountAmount	DECIMAL	NOT NULL	This column stores the discount amount given to the customer.
CustomerCategoryID	DECIMAL	NOT NULL	This column stores the total paid amount by the customer after discount.

Table 7: Invoice table after normalization.

Creating Invoice table in SQL.

Here, I have created the Invoice table using the (Create table) command. It consists of a primary key InvoiceID stores the data related to the invoice.

```
SQL> CREATE TABLE Invoice(
  2  InvoiceID INT NOT NULL PRIMARY KEY,
  3  PaymentOption VARCHAR2(30) NOT NULL,
  4  PaymentStatus VARCHAR2(30) NOT NULL,
  5  DiscountAmount DECIMAL NOT NULL,
  6  TotalPaidAmount DECIMAL NOT NULL);

Table created.

SQL>
```

Figure 9: Creating Invoice Table.

After creating the Invoice table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
SQL> Desc Invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
----
 INVOICEID                                 NOT NULL NUMBER(38)
 PAYMENTOPTION                             NOT NULL VARCHAR2(30)
 PAYMENTSTATUS                             NOT NULL VARCHAR2(30)
 DISCOUNTAMOUNT                            NOT NULL NUMBER(38)
 TOTALPAIDAMOUNT                           NOT NULL NUMBER(38)

SQL>
```

Figure 10: Describe Invoice Table.

5.1.5 OrderDetails

The OrderDetails table after normalization is given below:

Attributes	Data Type	Constraints	Description
OrderID	INTEGER	Primary Key	This column stores the Id of the order.
OrderDate	DATE	NOT NULL	This column stores the Date on which the order was made.
TotalOrderAmount	DECIMAL	NOT NULL	This column stores the total amount of the order before discount.
InvoiceID	INTEGER	FOREIGN KEY	This column stores InvoiceID which is linked to the InvoiceID in Invoice table.
DeliveryStatus	VARCHAR2(30)	NOT NULL	This column store the information if the order is delivered or not.

Table 8: Orderdetails table after normalization.

Creating OrderDetails table in SQL.

Here, I have created the OrderDetails table using the (Create table) command. It consists of a primary key OrderID, a foreign key InvoiceID and stores the data of the orders made by the customer.

```
SQL> CREATE TABLE OrderDetails(
  2  OrderID INT NOT NULL PRIMARY KEY,
  3  OrderDate DATE NOT NULL,
  4  TotalOrderAmount DECIMAL NOT NULL,
  5  InvoiceID INT NOT NULL,
  6  DeliveryStatus VARCHAR2(30) NOT NULL,
  7  FOREIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceID));

Table created.

SQL>
```

Figure 11: Creating OrderDetails Table.

After creating the OrderDetails table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
SQL> Desc OrderDetails;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
----
 ORDERID                                   NOT NULL NUMBER(38)
 ORDERDATE                                 NOT NULL DATE
 TOTALORDERAMOUNT                          NOT NULL NUMBER(38)
 INVOICEID                                 NOT NULL NUMBER(38)
 DELIVERYSTATUS                            NOT NULL VARCHAR2(30)

SQL>
```

Figure 12: Describe OrderDetails Table.

5.1.6 PurchasedProduct

The PurchasedProduct table after normalization is given below:

Attributes	Data Type	Constraints	Description

CustomerID	INTEGER	Primary Key (Composite)	This column stores the Id of the customer that made the purchase.
OrderID	INTEGER	Primary Key (Composite)	This column stores the order id.
ProductID	INTEGER	Primary Key (Composite)	This column stores the purchase product id.
PurchasedQuantity	INTEGER	NOT NULL	This column stores the quantity of the purchased product.
PurchasedQuantityPrice	DECIMAL	NOT NULL	This column stores the CustomerCategoryID which is linked to the column by the same name in the CustomerCategory table.

Table 9: PurchasedProduct table after normalization.

Creating PurchasedProduct table in SQL.

Here, I have created the PurchaseProduct table using the (Create table) command. It consists of a composite primary key made by the combination of Customer ID, Order ID and Product ID.

```
SQL> CREATE TABLE PurchasedProduct(
  2  CustomerID INT NOT NULL,
  3  OrderID INT NOT NULL,
  4  ProductID INT NOT NULL,
  5  PurchasedQuantity INT NOT NULL,
  6  PurchasedQuantityPrice DECIMAL NOT NULL,
  7  PRIMARY KEY (CustomerID, OrderID, ProductID));

Table created.

SQL>
```

Figure 13: Creating PurchasedProduct Table.

After creating the PurchasedProduct table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
SQL> Desc PurchasedProduct;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 CUSTOMERID                                NOT NULL NUMBER(38)
 ORDERID                                   NOT NULL NUMBER(38)
 PRODUCTID                                 NOT NULL NUMBER(38)
 PURCHASEDQUANTITY                         NOT NULL NUMBER(38)
 PURCHASEDQUANTITYPRICE                    NOT NULL NUMBER(38)

SQL>
```

Figure 14:Describe PurchasedProduct Table.

5.1.7 Vendor

The Vendor table after normalization is given below:

Attributes	Data Type	Constraints	Description
VendorID	INTEGER	Primary Key	This column stores the Id of the vendor.
Vendor	VARCHAR2(30)	NOT NULL	This column stores the name of the vendor.
VendorContact	VARCHAR2(30)	NOT NULL	This column stores the contact details of the vendor.

Table 10: Vendor table after normalization.

Creating Vendor table in SQL.

Here, I have created the Vendor table using the (Create table) command. It consists of a primary key VendorID and stores the data of the vendor that supplies products to our shop.

```
SQL> CREATE TABLE Vendor(
  2  VendorID INT NOT NULL PRIMARY KEY,
  3  Vendor VARCHAR2(30) NOT NULL,
  4  VendorContact VARCHAR2(30) NOT NULL);

Table created.
```

Figure 15: Creating Vendor Table.

After creating the Vendor table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
SQL> Desc Vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 VENDORID                                  NOT NULL NUMBER(38)
 VENDOR                                    NOT NULL VARCHAR2(30)
 VENDORCONTACT                             NOT NULL VARCHAR2(30)

SQL>
```

Figure 16: Describe Vendor Table.

5.1.8 ProductDetails

The ProdcutDetails table after normalization is given below:

Attributes	Data Type	Constraints	Description
ProductID	INTEGER	Primary Key	This column stores the Id of the Product.
ProductName	VARCHAR2(30)	NOT NULL	This column stores the name of the product.
ProductCategory	VARCHAR2(30)	NOT NULL	This column stores the information about which

			category a product belongs.
Description	VARCHAR2(100)	NOT NULL	This column stores the description of the product.
Price	INTEGER	NOT NULL	This column stores the unit price of the product.
StockLevel	INTEGER	NOT NULL	This column stores the stock level of the product.
AvailabilityStatus	VARCHAR2(30)	NOT NULL	This column stores the availability status of the product.
VendorID	INTEGER	FOREIGN KEY	This column stores the ID of the vendor that links with the VendorID in Vendor table.

Table 11: ProductDetails after normalization.

Creating ProdcutDetails table in SQL.

Here, I have created the ProductDetails table using the (Create table) command. It consists of a primary key ProductID, a foreign key VendorID and stores the data related to the details of the product.

```
SQL> CREATE TABLE ProductDetails(
  2  ProductID INT NOT NULL PRIMARY KEY,
  3  ProductName VARCHAR2(30) NOT NULL,
  4  ProductCategory VARCHAR2(30) NOT NULL,
  5  Description VARCHAR2(100) NOT NULL,
  6  Price INT NOT NULL,
  7  StockLevel INT NOT NULL,
  8  AvailabilityStatus VARCHAR2(30) NOT NULL,
  9  VendorID INT NOT NULL,
 10  FOREIGN KEY (VendorID) REFERENCES Vendor(VendorID));

Table created.

SQL>
```

Figure 17: Creating ProductDetails Table.

After creating the ProductDetails table, I used (Desc) command to describe the created table to see if all the columns are created properly with proper constraints.

```
SQL> Desc ProductDetails;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
----
 PRODUCTID                                 NOT NULL NUMBER(38)
 PRODUCTNAME                               NOT NULL VARCHAR2(30)
 PRODUCTCATEGORY                           NOT NULL VARCHAR2(30)
 DESCRIPTION                               NOT NULL VARCHAR2(100)
 PRICE                                     NOT NULL NUMBER(38)
 STOCKLEVEL                                NOT NULL NUMBER(38)
 AVAILABILITYSTATUS                        NOT NULL VARCHAR2(30)
 VENDORID                                  NOT NULL NUMBER(38)

SQL>
```

Figure 18: Describe ProductDetails Table.

5.1.9 Tables in the Database.

To see all the tables in the database I used (SELECT * FROM TAB;) command.

```
SQL> SELECT * FROM TAB;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
CUSTOMER                       TABLE
CUSTOMERCATEGORY               TABLE
INVOICE                        TABLE
ORDERDETAILS                   TABLE
ORDERS                         TABLE
PRODUCTDETAILS                 TABLE
PURCHASEDPRODUCT               TABLE
VENDOR                         TABLE

8 rows selected.

SQL>
```

Figure 19: List of tables in the database.

5.2 Inserting Data in the tables and verifying.

Inserting Data into CustomerCategory table.

The following data is stored in the CustomerCategory table using the (INSERT) command.

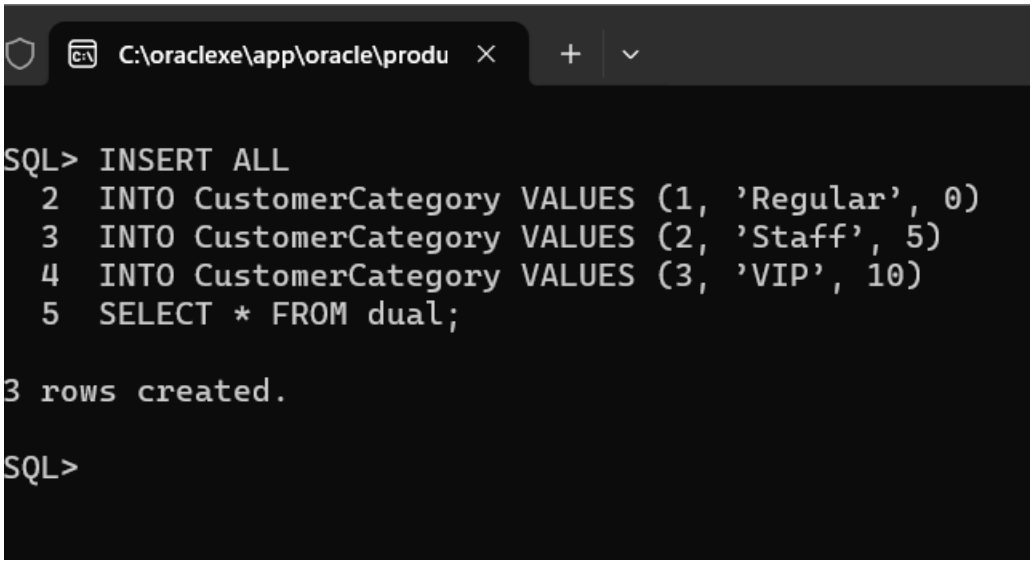

Figure 20: Inserting values into CustomerCategory table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

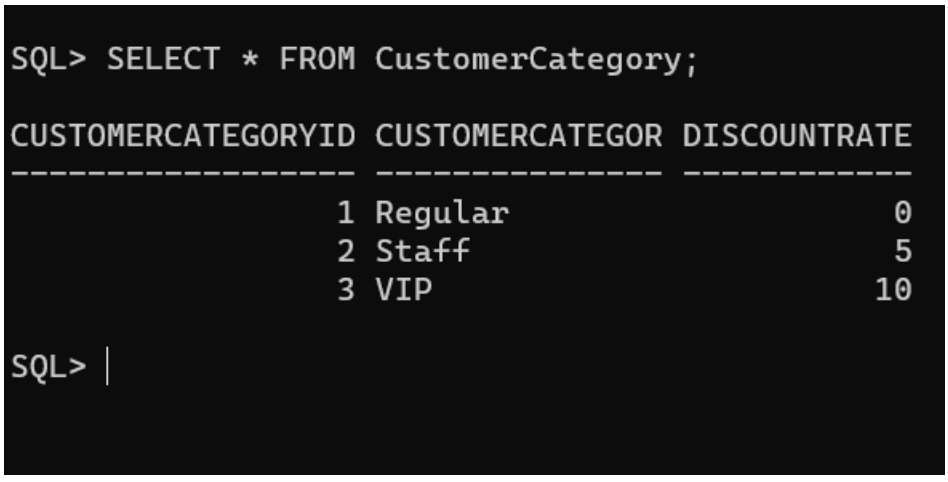

Figure 21: Verifying the data in CustomerCategory table.

Inserting Data in Customer table.

The following data is stored in the Customer table using the (INSERT) command.

```
SQL> INSERT ALL
  2  INTO Customer VALUES (1,'Atal','Kathmandu', 1)
  3  INTO Customer VALUES (2,'Hari','Jhapa', 2)
  4  INTO Customer VALUES (3,'Ram','Illam', 3)
  5  INTO Customer VALUES (4,'John','Pokhara', 2)
  6  INTO Customer VALUES (5,'Sita','Dhangadi', 1)
  7  INTO Customer VALUES (6,'Shyam','Jumla', 1)
  8  INTO Customer VALUES (7,'Priya','Nepalgunj', 2)
  9  INTO Customer VALUES (8,'Laxmi','Dhangadi', 1)
 10  INTO Customer VALUES (9,'Saksham','Surkhet', 3)
 11  INTO Customer VALUES (10,'Laxman','Chitwan', 3)
 12  SELECT * FROM dual;

10 rows created.

SQL>
```

Figure 22: Inserting values into Customer Table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

```
SQL> SELECT * FROM Customer;

CUSTOMERID CUSTOMERNAME    ADDRESS         CUSTOMERCATEGORYID
---------- --------------- --------------- ------------------
         1 Atal            Kathmandu                       1
         2 Hari            Jhapa                           2
         3 Ram             Illam                           3
         4 John            Pokhara                         2
         5 Sita            Dhangadi                        1
         6 Shyam           Jumla                           1
         7 Priya           Nepalgunj                       2
         8 Laxmi           Dhangadi                        1
         9 Saksham         Surkhet                         3
        10 Laxman          Chitwan                         3

10 rows selected.

SQL>
```

Figure 23:Verifying the data in Customer table.

Inserting Data in Orders table.

The following data is stored in the Orders table using the (INSERT) command.

```
SQL> INSERT ALL
  2  INTO Orders VALUES(1,1)
  3  INTO Orders VALUES (2,4)
  4  INTO Orders VALUES (3,5)
  5  INTO Orders VALUES (4,3)
  6  INTO Orders VALUES (5,3)
  7  INTO Orders VALUES (6,6)
  8  INTO Orders VALUES (7,7)
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 24 Inserting values into Orders Table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

```
SQL> SELECT * FROM Orders;

   ORDERID CUSTOMERID
---------- ----------
         1          1
         2          4
         3          5
         4          3
         5          3
         6          6
         7          7

7 rows selected.

SQL>
```

Figure 25 Verifying the data in Orders table.

Inserting Data in Invoice table.

The following data is stored in the invoice table using the (INSERT) command.

```
SQL> INSERT ALL
  2  INTO Invoice VALUES (1,'Online','Paid', 0, 3000)
  3  INTO Invoice VALUES (2,'COD','Paid', 100, 1900)
  4  INTO Invoice VALUES (3,'Online','Paid', 0, 5000)
  5  INTO Invoice VALUES (4,'Online','Paid', 200, 1800)
  6  INTO Invoice VALUES (5,'COD','Paid', 300, 2700)
  7  INTO Invoice VALUES (6,'COD','Paid', 0, 2500)
  8  INTO Invoice VALUES (7,'Online','Paid', 50, 950)
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 26 Inserting values into Invoice Table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

```
SQL> SELECT * FROM Invoice;

INVOICEID PAYMENTOPTION   PAYMENTSTATUS   DISCOUNTAMOUNT TOTALPAIDAMOUNT
--------- --------------- --------------- -------------- ---------------
        1 Online          Paid                         0            3000
        2 COD             Paid                       100            1900
        3 Online          Paid                         0            5000
        4 Online          Paid                       200            1800
        5 COD             Paid                       300            2700
        6 COD             Paid                         0            2500
        7 Online          Paid                        50             950

7 rows selected.

SQL>
```

Figure 27 Verifying the data in Orders table.

Inserting Data in OrderDetails table.

The following data is stored in the OrderDetails table using the (INSERT) command.

```
SQL> INSERT ALL
  2  INTO OrderDetails VALUES (1, DATE '2023-05-01', 3000, 1, 'Delivered')
  3  INTO OrderDetails VALUES (2, DATE '2023-05-12', 2000, 2, 'Delivered')
  4  INTO OrderDetails VALUES (3, DATE '2023-05-24', 5000, 3, 'Delivered')
  5  INTO OrderDetails VALUES (4, DATE '2023-06-05', 2000, 4, 'Delivered')
  6  INTO OrderDetails VALUES (5, DATE '2023-06-15', 3000, 5, 'Delivered')
  7  INTO OrderDetails VALUES (6, DATE '2023-06-17', 2500, 6, 'Delivered')
  8  INTO OrderDetails VALUES (7, DATE '2023-08-02', 1000, 7, 'Delivered')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 28 Inserting values into OrderDetails Table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

```
SQL> SELECT * FROM OrderDetails;

   ORDERID ORDERDATE TOTALORDERAMOUNT INVOICEID DELIVERYSTATUS
---------- --------- --------------- --------- ------------------------
         1 01-MAY-23            3000         1 Delivered
         2 12-MAY-23            2000         2 Delivered
         3 24-MAY-23            5000         3 Delivered
         4 05-JUN-23            2000         4 Delivered
         5 15-JUN-23            3000         5 Delivered
         6 17-JUN-23            2500         6 Delivered
         7 02-AUG-23            1000         7 Delivered

7 rows selected.

SQL>
```

Figure 29 Verifying the data in OrderDetails table.

Inserting Data in PurchasedProduct table.

The following data is stored in the PurchasedProduct table using the (INSERT) command.

```
SQL> INSERT ALL
  2  INTO PurchasedProduct VALUES (1, 1, 1, 1, 1500)
  3  INTO PurchasedProduct VALUES (1, 1, 2, 1, 1500)
  4  INTO PurchasedProduct VALUES (4, 2, 4, 2, 2000)
  5  INTO PurchasedProduct VALUES (5, 3, 3, 2, 5000)
  6  INTO PurchasedProduct VALUES (3, 4, 2, 2, 2000)
  7  INTO PurchasedProduct VALUES (3, 5, 4, 4, 3000)
  8  INTO PurchasedProduct VALUES (6, 6, 7, 5, 2500)
  9  INTO PurchasedProduct VALUES (7, 7, 7, 2, 1000)
 10  SELECT * FROM dual;

8 rows created.

SQL>
```

Figure 30 Inserting values into PurchasedProdcut Table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

```
SQL> SELECT * FROM PurchasedProduct;

CUSTOMERID    ORDERID  PRODUCTID PURCHASEDQUANTITY PURCHASEDQUANTITYPRICE
---------- ---------- ---------- ----------------- ----------------------
         1          1          1                 1                   1500
         1          1          2                 1                   1500
         4          2          4                 2                   2000
         5          3          3                 2                   5000
         3          4          2                 2                   2000
         3          5          4                 4                   3000
         6          6          7                 5                   2500
         7          7          7                 2                   1000

8 rows selected.

SQL>
```

Figure 31 Verifying the data in PurchasedProduct table.

Inserting Data in Vendor table.

The following data is stored in the Vendor table using the (INSERT) command.

```
SQL> INSERT ALL
  2  INTO Vendor VALUES(1,'Samsung','Samsung@gmail.com')
  3  INTO Vendor VALUES(2,'Apple','Apple @gmail.com')
  4  INTO Vendor VALUES(3,'Oppo','Oppo @gmail.com')
  5  INTO Vendor VALUES(4,'Microsoft','Microsoft @gmail.com')
  6  INTO Vendor VALUES(5,'Sony','Sony@gmail.com')
  7  INTO Vendor VALUES(6,'Canon','Canon@gmail.com')
  8  INTO Vendor VALUES(7,'Acer','Acer@gmail.com')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 32 Inserting values into Vendor Table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

```
SQL> SELECT * FROM Vendor;

  VENDORID VENDOR                         VENDORCONTACT
---------- ---------------------------- ----------------------
         1 Samsung                      Samsung@gmail.com
         2 Apple                        Apple @gmail.com
         3 Oppo                         Oppo @gmail.com
         4 Microsoft                    Microsoft @gmail.com
         5 Sony                         Sony@gmail.com
         6 Canon                        Canon@gmail.com
         7 Acer                         Acer@gmail.com

7 rows selected.
```

Figure 33 Verifying the data in Vendor table.

Inserting Data in ProdcutDetails table.

The following data is stored in the ProductDetails table using the (INSERT) command.

```
SQL> INSERT ALL
  2  INTO ProductDetails VALUES (1, 'Iphone15', 'Smartphone', 'Apple Iphone 15', 1500, 16, 'Available', 2)
  3  INTO ProductDetails VALUES (2, 'TabS4', 'Tablet', 'Samsung Galaxy tab s4', 1000, 60, 'Available', 1)
  4  INTO ProductDetails VALUES (3, 'MacM1', 'Laptop', 'Apple Mac', 2500, 16, 'Available', 2)
  5  INTO ProductDetails VALUES (4, 'A15', 'Smartphone', 'Oppo A15', 750, 54, 'Available', 3)
  6  INTO ProductDetails VALUES (5, 'M50', 'Camera', 'Canon M50', 800, 12, 'Available', 6)
  7  INTO ProductDetails VALUES (6, 'Xbox', 'Gaming Console', 'Xboxone', 500, 15, 'Available', 4)
  8  INTO ProductDetails VALUES (7, 'PS5', 'Gaming Console', 'Playstation', 500, 22, 'Available', 5)
  9  INTO ProductDetails VALUES (8, 'Airpods Pro', 'Headphone', 'Apple Airpods', 200, 32, 'Available', 2)
 10  INTO ProductDetails VALUES (9, 'Nitro5', 'Laptop', 'Acer Nitro5', 1650, 5, 'Available', 7)
 11  INTO ProductDetails VALUES (10, 'Ipad5', 'Tablet', 'Apple Ipad5', 1200, 12, 'Available', 2)
 12  SELECT * FROM dual;

10 rows created.
```

Figure 34 Inserting values into ProductDetails Table.

(SELECT * FROM table_name) command is used to see the inserted data in the table.

```
SQL> SELECT * FROM ProductDetails;

PRODUCTID PRODUCTNAME          PRODUCTCATEGORY      DESCRIPTION                              PRICE STOCKLEVEL AVAILABILITYSTA VENDORID
--------- -------------------- -------------------- ---------------------------------------- ----- ---------- --------------- --------
        1 Iphone15             Smartphone           Apple Iphone 15                           1500         16 Available              2
        2 TabS4                Tablet               Samsung Galaxy tab s4                     1000         60 Available              1
        3 MacM1                Laptop               Apple Mac                                 2500         16 Available              2
        4 A15                  Smartphone           Oppo A15                                   750         54 Available              3
        5 M50                  Camera               Canon M50                                  800         12 Available              6
        6 Xbox                 Gaming Console       Xboxone                                    500         15 Available              4
        7 PS5                  Gaming Console       Playstation                                500         22 Available              5
        8 Airpods Pro          Headphone            Apple Airpods                              200         32 Available              2
        9 Nitro5               Laptop               Acer Nitro5                               1650          5 Available              7
       10 Ipad5                Tablet               Apple Ipad5                               1200         12 Available              2

10 rows selected.

SQL>
```

Figure 35 Verifying the data in ProductDetails table.

6.Database Querying

1.1 Information Query

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

Ans: I have entered the following command to get the list of all the customers that are also staff of the company. We can see in the picture below that there are three customers Hari, John and Priya who are also staff of our company.

```
SQL> SELECT *
  2  FROM Customer
  3  WHERE CustomerCategoryID = (SELECT CustomerCategoryID FROM CustomerCategory WHERE CustomerCategory = 'Staff');

CUSTOMERID CUSTOMERNAME                   ADDRESS                                            CUSTOMERCATEGORYID
---------- ------------------------------ -------------------------------------------------- ------------------
         2 Hari                           Jhapa                                                               2
         4 John                           Pokhara                                                             2
         7 Priya                          Nepalgunj                                                           2

SQL>
```

Figure 36: Information Query Q no 1 Ans.

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

Ans: I have entered the following command to get the list all the orders made for any ‘A15’ between the dates 01-05-2023 till 28- 05-2023.

```
SQL> SELECT OrderDetails.*
  2  FROM OrderDetails
  3  JOIN PurchasedProduct ON OrderDetails.OrderID = PurchasedProduct.OrderID
  4  JOIN ProductDetails ON PurchasedProduct.ProductID = ProductDetails.ProductID
  5  WHERE OrderDetails.OrderDate BETWEEN TO_DATE('2023-05-01', 'YYYY-MM-DD') AND TO_DATE('2023-05-28', 'YYYY-MM-DD')
  6  AND ProductDetails.ProductName = 'A15';

   ORDERID ORDERDATE TOTALORDERAMOUNT  INVOICEID DELIVERYSTATUS
---------- --------- ---------------- ---------- ------------------------------
         2 12-MAY-23             2000          2 Delivered

SQL>
```

Figure 37: Information Query Q no 2 Ans.

We can see in the screen shot below that the order no 2 was made between the given date and has the product ‘A15’ in it.

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

Ans: I have entered the following command to get the list all the customers with their order details and also the customers who have not ordered any products yet**.**

```
SQL> SELECT Customer.*, Orders.*, OrderDetails.*
  2  FROM Customer
  3  LEFT JOIN Orders ON Customer.CustomerID = Orders.CustomerID
  4  LEFT JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
  5  ORDER BY Customer.CustomerID;

CUSTOMERID CUSTOMERNAME                 ADDRESS                                            CUSTOMERCATEGORYID    ORDERID CUSTOMERID    ORDERID ORDERDATE TOTALORDERAMOUNT  INVOICEID DELIVERYSTATUS
---------- ---------------------------- -------------------------------------------------- ------------------ ---------- ---------- ---------- --------- ----------------- ---------- ---------------
-----
         1 Atal                         Kathmandu                                                         1          1          1          1 01-MAY-23         3000          1 Delivered
         2 Hari                         Jhapa                                                             2
         3 Ram                          Illam                                                             3          4          3          4 05-JUN-23         2000          4 Delivered
         3 Ram                          Illam                                                             3          5          3          5 15-JUN-23         3000          5 Delivered
         4 John                         Pokhara                                                           2          2          4          2 12-MAY-23         2000          2 Delivered
         5 Sita                         Dhangadi                                                          1          3          5          3 24-MAY-23         5000          3 Delivered
         6 Shyam                        Jumla                                                             1          6          6          6 17-JUN-23         2500          6 Delivered
         7 Priya                        Nepalgunj                                                         2          7          7          7 02-AUG-23         1000          7 Delivered
         8 Laxmi                        Dhangadi                                                          1
         9 Sakshan                      Surkhet                                                           3
        10 Laxman                       Chitwan                                                           3

11 rows selected.

SQL>
```

Figure 38: Information Query Q no 3 Ans.

We can see in the picture below that only seven customers have ordered so far.

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

Ans: I have entered the following command to list all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.

```
SQL> SELECT *
  2  FROM ProductDetails
  3  WHERE SUBSTR(ProductName, 2, 1) = 'a' AND StockLevel > 50;

 PRODUCTID PRODUCTNAME                    PRODUCTCATEGORY               DESCRIPTION                    PRICE STOCKLEVEL AVAILABILITYSTATUS                VENDORID
---------- ------------------------------ ------------------------------ ------------------------------ ---------- ---------- ------------------------------ ----------
         2 TabS4                          Tablet                         Samsung Galaxy tab s4            1000         60 Available                               1

SQL>
```

Figure 39: Information Query Q no 4 Ans.

We can see in the picture above that only TabS4 is a product that have quantity more than 50 and has the second letter ‘a’ in its name.

5. Find out the customer who has ordered recently.

Ans: I have entered the following command to find out the customer who has ordered recently.

Figure 40: Information Query Q no 5 Ans.

We can see in the above picture that Priya is the customer who has ordered recently.

6.2 Transaction Query

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

Ans: I have entered the following command to show the total revenue of the company for each month.

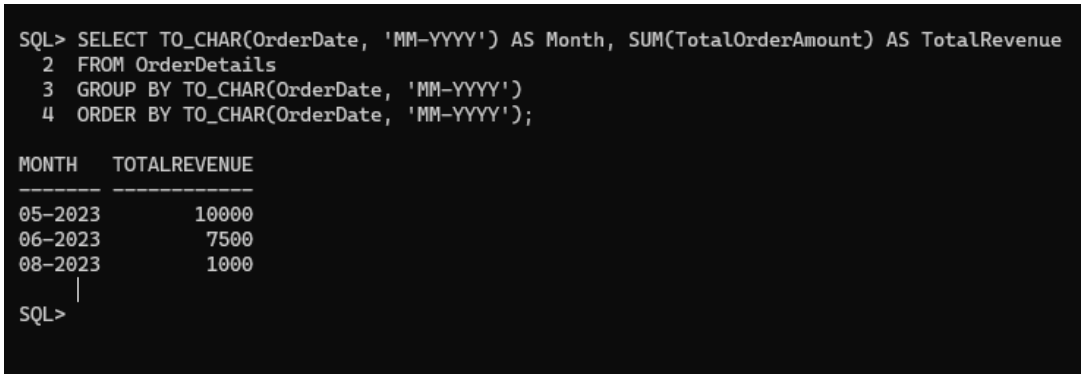

Figure 41:Transaction Query Q no 1 Ans.

We can see in the above figure the total revenue of three months.

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

Ans: I have entered the following command to find those orders that are equal or higher than the average order total value.

```
SQL> SELECT *
  2  FROM OrderDetails
  3  WHERE TotalOrderAmount >= (SELECT AVG(TotalOrderAmount) FROM OrderDetails);

   ORDERID ORDERDATE TOTALORDERAMOUNT  INVOICEID DELIVERYSTATUS
---------- --------- ---------------- ---------- ------------------------------
         1 01-MAY-23             3000          1 Delivered
         3 24-MAY-23             5000          3 Delivered
         5 15-JUN-23             3000          5 Delivered

SQL>
```

Figure 42:Transaction Query Q no 2 Ans.

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

Ans: I have entered the following command to list the details of vendors who have supplied more than 3 products to the company.

```
SQL> SELECT Vendor.*, COUNT(ProductDetails.ProductID) AS TotalProductsSupplied
  2  FROM Vendor
  3  JOIN ProductDetails ON Vendor.VendorID = ProductDetails.VendorID
  4  GROUP BY Vendor.VendorID, Vendor.Vendor, Vendor.VendorContact
  5  HAVING COUNT(ProductDetails.ProductID) > 3;

  VENDORID VENDOR                         VENDORCONTACT                  TOTALPRODUCTSSUPPLIED
---------- ------------------------------ ------------------------------ ---------------------
         2 Apple                          Apple @gmail.com                                   4

SQL>
```

Figure 43:Transaction Query Q no 3 Ans.

We can see in the above diagram that Apple provides four products to our shop.

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

Ans: I have entered the following command to show the top 3 product details that have been ordered the most.

```
SQL> SELECT *
FROM (
    SELECT
        ProductDetails.ProductID,
        ProductDetails.ProductName,
        ProductDetails.ProductCategory,
        ProductDetails.Description,
        ProductDetails.Price,
        ProductDetails.StockLevel,
        ProductDetails.AvailabilityStatus,
        ProductDetails.VendorID,
        COUNT(PurchasedProduct.OrderID) AS OrderCount
    FROM
        PurchasedProduct
    JOIN
        ProductDetails ON PurchasedProduct.ProductID = ProductDetails.ProductID
    GROUP BY
        ProductDetails.ProductID, ProductDetails.ProductName, ProductDetails.ProductCategory, ProductDetails.Description, ProductDetails.Price, ProductDetails.StockLevel, ProductDetails.AvailabilityStatus, ProductDetails.VendorID
    ORDER BY
        OrderCount DESC
)
WHERE ROWNUM <= 3;

PRODUCTID PRODUCTNAME  PRODUCTCATEGORY  DESCRIPTION            PRICE STOCKLEVEL AVAILABILITYSTATUS  VENDORID ORDERCOUNT
--------- ------------ ---------------- ---------------------- ----- ---------- ------------------- -------- ----------
        2 TabS4        Tablet           Samsung Galaxy tab s4   1000         60 Available                  1          2
        7 PS5          Gaming Console   Playstation              500         22 Available                  5          2
        4 A15          Smartphone       Oppo A15                 750         54 Available                  3          2

SQL> |
```

Figure 44:Transaction Query Q no 4 Ans.

We can see in the above picture that TabS4, PS5 and A15 are the top 3 product that have been ordered the most.

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

Ans: I have entered the following command to find out the customer who has ordered the most in August with his/her total spending on that month.

```
SQL> SELECT *
FROM (
    SELECT
        Customer.CustomerID,
        Customer.CustomerName,
        SUM(Invoice.TotalPaidAmount) AS TotalSpending
    FROM
        Customer
    JOIN
        PurchasedProduct ON Customer.CustomerID = PurchasedProduct.CustomerID
    JOIN
        OrderDetails ON PurchasedProduct.OrderID = OrderDetails.OrderID
    JOIN
        Invoice ON OrderDetails.InvoiceID = Invoice.InvoiceID
    WHERE
        TO_CHAR(OrderDetails.OrderDate, 'MM') = '08'
    GROUP BY
        Customer.CustomerID, Customer.CustomerName
    ORDER BY
        TotalSpending DESC
)
WHERE ROWNUM <= 1;

CUSTOMERID CUSTOMERNAME                   TOTALSPENDING
---------- ------------------------------ -------------
         7 Priya                                    950

SQL>
```

Figure 45:Transaction Query Q no 5 Ans.

We can see in the above picture that Priya is the customer who has ordered the most in August and her total spending on that month is 950.

7.Critical Evaluation

This is one of the most important modules I have studied in Information technology. It has helped me a lot in understanding how data works and is used especially for online stores like this one. Its not just a theoretical module, the majority of this modules is focused on real life use and that shines through in coursework, where we gain hands-on experience in designing a strong database system for “Gadget Emporium”. This module can be used for managing data in real life but it is not fully complete without intergrading other modules in it. For example to fully launch our online store we need to enhance our security measures, create a proper GUI for the store,etc.

I was able to learn a lot about the during the course of this module. To create a database ‘” Gadget Emporium” I had to learn a lot about data and manipulation of data in the database. I had to learn about normalization, entity relationship diagram, business rules and various sql commands to create a table, insert into that table, view the table and show the table data in a certain way. I have gained a better understanding of how to design and manage databases, especially in the context of an online store like this one. I have not only learned the technical aspects of database design but also the practical skills for building a successful and scalable e-commerce platform.

8.References

Anon., 2023. *Margaret Rouse.* [Online]
Available at:
https://www.techtarget.com/searchdatamanagement/definition/normalization
[Accessed 13 Janaury 2023].

GeeksforGeeks, 2023. *Introduction to ER Model.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-of-er-model/