CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: Anjan Giri

London Met ID: 22085496

Assignment Submission Date: 14th January 2023

Word Count: 6460

Table of Contents

1. INTRODUCTION 1

1.2 Current Business Activities and Operations 2

1.3 Business Rules 3

1.3.1 Assumptions 4

1.4 Identification of Entities and Attributes 5

Customer Table 5

Order Table 6

Product Table 7

2. Initial ERD 9

3. Normalization 10

3.1 UNF (Un-normalized Form) 10

3.2 1NF (First Normal Form) 11

3.3 2NF (Second Normal Form) 12

3.4 3NF (Third Normal Form) 15

4. Final ERD 20

5. IMPLEMENTATION 22

5.1 Creating and connecting to new user. 22

5.2 Creating all of the tables with required columns. 24

5.2.1 Creating Customer_Category table. 24

5.2.2 Creating Customer table. 25

5.2.3 Creating Invoice table. 27

5.2.4 Creating Order table. 28

5.2.5 Creating Vendor table. 30

5.2.6 Creating Product_Category table. 31

5.2.7 Creating Product table. 32

5.2.8 Creating Product_Customer_Order table. 34

5.3 Inserting values in the tables. 36

5.3.1 Inserting values in Customer_Category table. 36

5.3.2 Inserting values in Customer table. 38

5.3.3 Inserting values in Invoice table. 40

5.3.4 Inserting values in Order table. 41

5.3.5 Inserting values in Vendor table. 44

5.3.6 Inserting values in Product_Category table. 45

5.3.7 Inserting values in Product table. 47

5.3.8 Inserting values in Product_Customer_Order table. 49

6. QUERIES 51

6.1 INFORMATION QUERIES 51

6.1.1 List all the customers that are also staff of the company. 51

6.1.2 List all the orders made for any particular product between the dates 01-05-2023 till 28-05-2023. 52

6.1.3 List all the customers with their order details and also the customers who have not ordered any products yet. 53

6.1.4 List all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50. 54

6.1.5 Find out the customer who has ordered recently. 55

6.2 TRANSACTION QUERIES 56

6.2.1 Show the total revenue of the company for each month. 56

6.2.2 Find those orders that are equal or higher than the average order total value. 57

6.2.3 List the details of vendors who have supplied more than 3 products to the company. 58

6.2.4 Show the top 3 product details that have been ordered the most. 59

6.2.5 Find out the customer who has ordered the most in August with his/her total spending on that month. 61

7. CRITICAL EVALUATION 63

8. DROP QUERY AND DUMP FILE CREATION 64

References 67

Table of Figures

Figure 1 : Initial ERD 9
Figure 2 : Final ERD 20
Figure 3 : creating a new user 22
Figure 4 : granting privileges to the user 23
Figure 5 : creating Customer_Category table 24
Figure 6 : checking details of Customer_Category table 24
Figure 7 : creating Customer table 25
Figure 8 : checking details of Customer table 26
Figure 9 : creating Invoice table 27
Figure 10 : checking details of Invoice table 27
Figure 11 : creating Order table 28
Figure 12 : checking details of Order table 29
Figure 13 : creating Vendor table 30
Figure 14 : checking details of Vendor table 30
Figure 15 : creating Product_Category table 31
Figure 16 : checking details of Product_Category table 31
Figure 17 : creating Product table 33
Figure 18 : checking details of Product table 33
Figure 19 : creating Product_Customer_Order table 34
Figure 20 : checking details of Product_Customer_Order table 35
Figure 21 : inserting values in Customer_Category table 36
Figure 22 : checking the values inserted in Customer_Category table 37
Figure 23 : inserting values in Customer table 39

Figure 24 : checking the values inserted in Customer table 39

Figure 25 : inserting values in Invoice table 40

Figure 26 : checking the values inserted in Invoice table 40

Figure 27 : inserting values in Order table... 41

Figure 28 : updating the values in Order table 43

Figure 29 : checking the values inserted in Order table 43

Figure 30 : inserting values in Vendor table 44

Figure 31 : checking the values inserted in Vendor table 44

Figure 32 : inserting values in Product_Category table 45

Figure 33 : checking the inserted values in Product_Category table............................. 46

Figure 34 : inserting values in Product table 48

Figure 35 : checking the values inserted in Product table 48

Figure 36 : inserting values in Product_Customer_Order table 49

Figure 37 : checking the values inserted in Product_Customer_Order table................. 50

Figure 38 : listing the customers who are also staff at the company 51

Figure 39 : listing all the orders made for 'Earpods MAX' between the dates 01-05-2023 till 28-05-2023. 52

Figure 40 : listing all customers with their order details along with the customers who have not ordered any products yet.. 53

Figure 41 : listing all product details that have second letter 'a' in their name and have stock more than 50... 54

Figure 42 : customer who has ordered recently along with the order date 55

Figure 43 : total revenue of the company for each month 56

Figure 44 : orders that are equal or higher than average order total value.................... 57

Figure 45 : listing details of vendors who have supplied more than 3 products to the company 58

Figure 46 : showing the top three products that have been ordered the most along with their details 60

Figure 47 : listing the customer who has ordered the most in August month with his/her total spending for that month 62

Figure 48 : tables present in the system 64

Figure 49 : drop product_customer_order, customer and customer_category tables 64

Figure 50 : drop order and invoice tables 65

Figure 51 : drop product, vendor and product_category tables 65

Figure 52 : dump file 66

Table of Tables

Table 1 : Attributes of Customer table.. 5

Table 2 : Attributes of Order table 6

Table 3 : Attributes of Product table.. 8

1. INTRODUCTION

‘Mr. John’ is a big entrepreneur with many successful businesses and facilities in Nepal. One of the businesses owned by him is a store named ‘Gadget Emporium’. The store deals with the operations related to various electric devices.

Located in the heart of Kathmandu valley i.e., Durbar Marg, Kathmandu, the store is expanding day by day and is taking over the market of electronics with its large selection of electronic devices along with the top-notch quality in the store’s products. As the store is getting bigger, Mr. John has hired hundreds of employees to maintain the smooth flow of operations in the store.

The physical store has been running and doing well for many years now but the increase in online platforms has built a new path for many businesses to flourish and Mr. John wants to expand his business through online medium as well. As the store is growing and expanding, Mr. John is planning to establish Gadget Emporium online marketplace. He wants to provide both private customers and business organizations large selection of gadgets without visiting the store physically. The establishment of the online marketplace also aims to make it more convenient and easily accessible to new as well as old customers to choose and buy the gadgets through online medium.

As a database designer, the task at hand is designing and making the database system for the ‘Gadget Emporium’ online marketplace fulfilling the requirements of Mr. John abiding by the provided business rules. The system should be able to perform all operations as required for the business. The main aim of designing the system is to make every task performed in the system convenient and effective. The system should keep proper record of every customer’s detail, their orders and the products purchased by them along with the products availability in the inventory. This should contribute to smooth operation of business activities and help in maintaining the data of the store in proper manner.

1.2 Current Business Activities and Operations

The store is currently operating through its physical outlet only. Customers in high numbers visit the store daily and take services from the store. To ensure smooth operations in every section, Mr. John has hired many employees that are actively working in the store. The store provides various gadgets, and it also addresses any issued faced by the customers along with different maintenance facilities. The store is actively involved in the market and ensures high sales every day.

Database system to maintain proper collection and management od data is a must in every business. The store keeps proper record of each customer, their orders and the products purchased by the customers in a database system of the store. Employees working in the store ensure that the data are well entered and properly maintained in the database system of the store. Various details regarding the products such as its name, unit price and availability are maintained properly and are checked in regular interval. Along with it, details regarding the orders are also maintained in the system. After a certain order is made, the ordered products are then sold to a customer after which an invoice is generated carrying the details of the purchase in it.

Although the system for data collection and execution may be lacking in some department, it has been remarkable for collecting and maintaining data required in the store. But now, as the store is planning to run an online marketplace database system required for it should be vaster and more efficient than the system that has been in use in the store throughout the years.

1.3 Business Rules

The business rules to be followed are stated as follows:

- The system should be able to keep records of customer, order and product details.

- Customers should be categorized as Regular, Staff and VIP and should be provided with a discount accordingly.

- Addresses of the customers should be recorded which facilitates in delivery of the products.

- A product can fall under only one product category whereas multiple products can be included in a single product category.

- Records of vendors supplying the products should be maintained.

- Each product should be associated with a single vendor and each vendor can supply multiple products.

- An order can contain multiple products and any one type of product can be included in multiple orders as any type of product can be purchased multiple times.

- Inventory details about the products should be properly maintained.

- The system should have various payment options to facilitate effective and smooth transactions of every orders.

- An invoice should be issued when the order is confirmed with proper payment details.

1.3.1 Assumptions

The assumptions made to ensure that the system is effective and efficient in handling all records and data related to every transaction along with the necessary details are stated as follows:

- Each entity must be assigned to a unique id of its own to reduce data duplication.

- Customer category id gives the customer category in which the customers are included in.

- Customer id provides the details of the customer.

- A customer can place multiple orders, but each order is associated with a single customer who places the order.

- Invoice id gives the detail about the payment type.

- Order id provides every detail related to the order made.

- The payment types accepted are debit/credit card, cash on delivery and e-wallet.

- Every price or amount stated in the database system are denominated in a single currency i.e., USD ($). So, all transaction details involving monetary transactions are expressed in USD value.

- Product id gives details about the products in the marketplace.

- Vendor id provides the name of the vendor/supplier.

- Product category id gives the category in which the product is included.

- All columns of every table must have some value in them. Values in any columns cannot be empty (NULL).

1.4 Identification of Entities and Attributes

Customer Table

The customer table consists of different attributes. Customer Id is the Primary Key of the table. The remaining attributes are Customer Name, Contact, Address, Customer Category Id, Customer Category Name and Discount Rate.

Attribute	Constraint	Datatype	Description
Customer Id (PK)	NOT NULL	varchar	Customer Id is the primary key of the table. It gives unique id to each customer.
Customer Name	NOT NULL	varchar	Customer Name gives the name of the customer.
Contact	NOT NULL	varchar	Contact provides the contact details of a customer.
Address	NOT NULL	varchar	It gives the address of the customer.
Customer Category Id	NOT NULL	varchar	Customer Category Id provides the unique id which are assigned to the categories of the customer.
Customer Category Name	NOT NULL	varchar	Customer Category Name gives the name of the category in which a customer falls into.
Discount Rate	NOT NULL	varchar	Discount Rate gives the discount percentage given to the customers according to their categories.

Table 1 : Attributes of Customer table

Order Table

Order table contains Order Id as its Primary Key. Other than that, the attributes present in the table are Date, Quantity, Total Amount, Invoice Id and Payment Type. The table also contains Customer Id as a Foreign Key.

Attribute	Constraint	Datatype	Description
Order Id (PK)	NOT NULL	varchar	Order Id is the primary key of the order table. It gives unique id to each order made by the customer.
Date	NOT NULL	date	Date gives the date on which the order has been made.
Quantity	NOT NULL	number	Quantity provides the total number of products in a order.
Total Amount	NOT NULL	number	Total Amount provides the total price of the order.
Invoice Id	NOT NULL	varchar	Invoice Id gives unique id to the invoice made after the order is made.
Payment Type	NOT NULL	varchar	Payment Type gives the method of payment chosen by the customer for doing the payment.
Customer Id (FK)	NOT NULL	varchar	Customer Id is a foreign key which relates the order table with the customer table.

Table 2 : Attributes of Order table

Product Table

Product Id is the Primary Key of the Product Table. Along with it, Product Name, Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id and Vendor Name are other attributes present in the table. It also contains Order Id as a Foreign Key.

Attribute	Constraint	Datatype	Description
Product Id (PK)	NOT NULL	varchar	It is the primary key of product table. It gives unique id to each product that the store deals with.
Product Name	NOT NULL	varchar	Product Name gives the name of the product.
Unit Price	NOT NULL	number	Unit Price gives the unit wise price of the products contained in the order.
Stock	NOT NULL		Stock gives the inventory data related to the products.
Description		varchar	It gives short description about the product.
Product Category Id	NOT NULL	varchar	Product Category Id is the unique id provided to the categories in which the products belong.
Product Category Name	NOT NULL	varchar	It provides the name of the categories that the products fall under.
Vendor Id	NOT NULL	varchar	Vendor Id is the unique id assigned to the vendors who supply the products to the store.
Vendor Name	NOT NULL	varchar	It gives the name of the vendors.

Order Id (FK)	NOT NULL	varchar	Order Id is a foreign key which relates the order and product table.

Table 3 : Attributes of Product table

2. Initial ERD

Initial ERD is an entity relationship diagram which is created at the start of any database system design. It is the start point and is created before doing normalization and finalizing an actual ERD for the required case. There may be many-to-many relation while creating an initial ERD which should be removed in the process of normalization before making final ERD of the case.

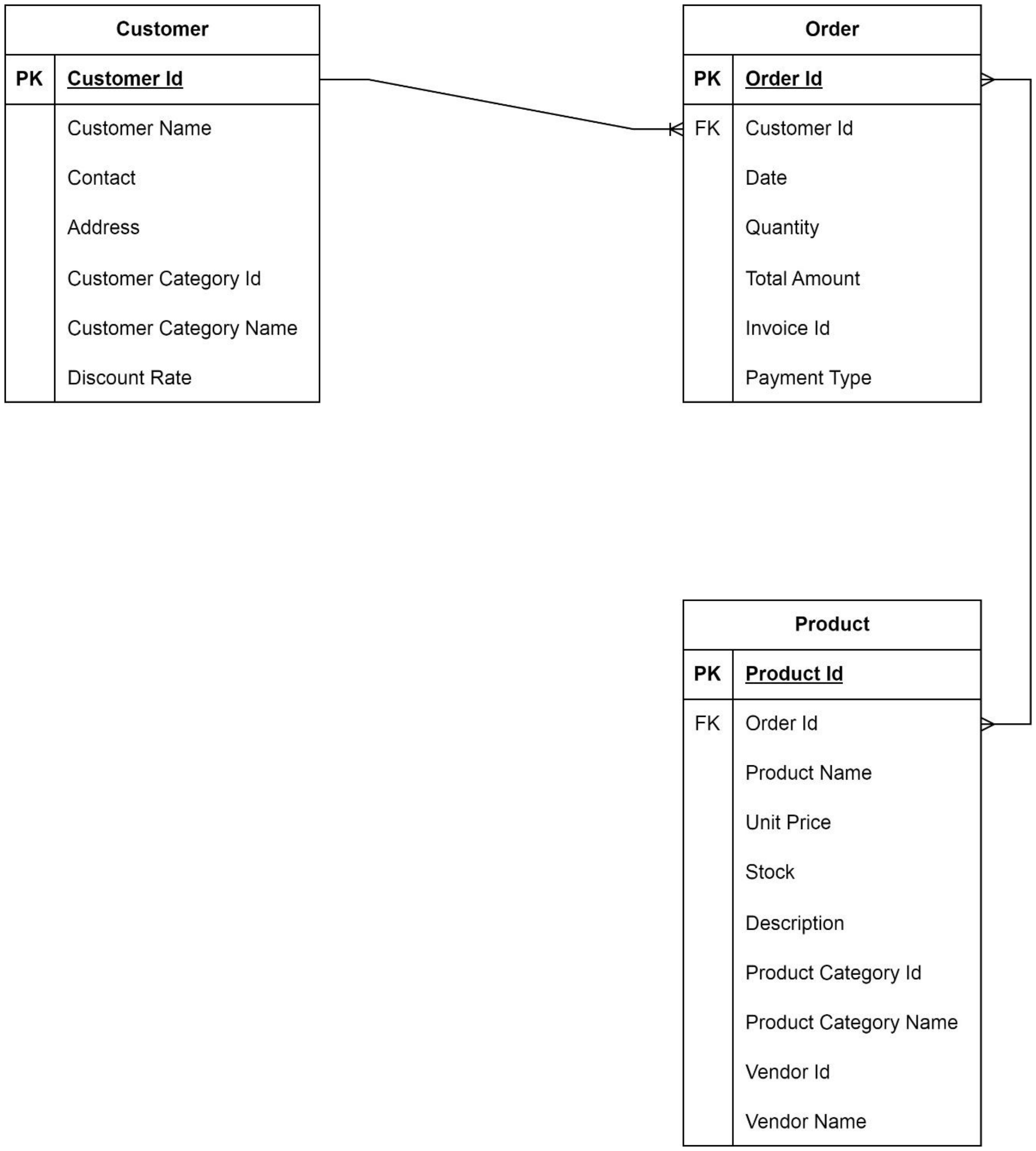


Figure 1 : Initial ERD

3. Normalization

3.1 UNF (Un-normalized Form)

In UNF, we need to identify the repeating data and repeating groups from the entities and attributes of the initial ERD. It is the initial phase of the normalization. The repeating data are written inside the small brackets whereas the repeating groups are written inside curly brackets with no repeating attributes which makes it easier to identify the repeating data and groups.

The un-normalized form of the system is:

Customer - (Customer Id, Customer Name, Contact, Address, Customer Category Id, Customer Category Name, Discount Rate,

{Order Id, Date, Quantity, Total Amount, Invoice Id, Payment Type

{Product Id, Product Name, Unit Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id, Vendor Name}})

Here, in the above un-normalized form, Customer Id is a Primary Key in which all other attributes are depended upon.

Customer Id (PK), Customer Name, Contact, Address, Customer Category Id, Customer Category Name and Discount Rate are repeating data.

{Order Id, Date, Quantity, Total Amount, Invoice Id, Payment Type {Product Id, Product Name, Unit Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id, Vendor Name}} are repeating groups.

3.2 1NF (First Normal Form)

In 1NF, the repeating data and repeating groups are separated from each other and are presented in separate tables of their own. It is the first step of normalization which ensures elimination of duplicate data. It helps to minimize data redundancy and dependency.

Here, repeating groups make new entities and tables which are connected to previous tables through a foreign key.

The first normal form of the system is:

Customer-1 – (<u>Customer Id</u>, Customer Name, Contact, Address, Customer Category Id, Customer Category Name, Discount Rate)

Order-1 – (<u>Order Id</u>, Date, Quantity, Total Amount, Invoice Id, Payment Type, Customer Id*)

Product-1 – (<u>Product Id</u>, Product Name, Unit Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id, Vendor Name, Customer Id*, Order Id*)

Here in the above 1NF form, three tables are formed where each table have their own Primary Key and are related to each other.

In ‘Customer-1’ table, ‘Customer Id’ is a Primary Key.

In ‘Order-1’ table, ‘Order Id’ is a Primary Key and the table also contains a Foreign Key which is ‘Customer Id’ that relates the ‘Order-1’ and ‘Customer-1’ tables.

In ‘Product-1’ table, ‘Product Id’ is a Primary Key. It also contains ‘Customer Id’ and ‘Order Id’ as two Foreign Keys that relates the tables with each other.

3.3 2NF (Second Normal Form)

It is the second step of normalization. The partial dependencies are checked in the 1NF form of the system and if found any, are removed to convert the normalization into 2NF form.

First, let’s check the dependency in ‘Customer-1’ table:

Customer-1 – (<u>Customer Id</u>, Customer Name, Contact, Address, Customer Category Id, Customer Category Name, Discount Rate)

Number of keys = 1

So, $2^n - 1 = 2^1 - 1 = 1$

Here, ‘Customer Id’ as a Primary Key gives other attributes and there is no partial dependency in the ‘Customer-1’ table and therefore, it remains same as it was in the first normal form.

‘Customer-1’ table in 2NF:

Customer-2 – (<u>Customer Id</u>, Customer Name, Contact, Address, Customer Category Id, Customer Category Name, Discount Rate)

Now, checking the ‘Order-1’ table:

Order-1 – (<u>Order Id</u>, Date, Quantity, Total Amount, Invoice Id, Payment Type, Customer Id*)

Number of keys = 2

So, $2^n - 1 = 2^2 - 1 = 3$

Checking for dependencies:

Order Id → Date, Quantity, Total Amount, Invoice Id, Payment Type

Customer Id → X

(No table is formed as there is only one column ‘Customer Id’)

Order Id, Customer Id → X

Now, checking the ‘Product-1’ table:

Product-1 – (Product Id, Product Name, Unit Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id, Vendor Name, Customer Id*, Order Id*)

Number of keys = 3

So, $2^n – 1 = 2^3 – 1 = 7$

Checking for dependencies:

Product Id → Product Name, Unit Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id, Vendor Name

Customer Id → X

(No table is formed as there is only one column ‘Customer Id’)

Order Id → X

(No table is formed as there is only one column ‘Order Id’)

Product Id, Customer Id → X

Product Id, Order Id → X

Customer Id, Order Id → X

Product Id, Customer Id, Order Id → X

The second normal form of the system is:

Customer-2 – (<u>Customer Id</u>, Customer Name, Contact, Address, Customer Category Id, Customer Category Name, Discount Rate)

Order-2 – (<u>Order Id</u>, Customer Id*, Date, Quantity, Total Amount, Invoice Id, Payment Type)

Product-2 – (<u>Product Id</u>, Customer Id*, Order Id*, Product Name, Unit Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id, Vendor Name)

Order_Customer-2 – (Order Id*, Customer Id*)

Product_Customer-2 – (Product Id*, Customer Id*)

Product_Order-2 – (Product Id*, Order Id*)

Product_Customer_Order-2 – (Product Id*, Customer Id*, Order Id*)

3.4 3NF (Third Normal Form)

It is the third and last step of normalization. The transitive dependencies are checked and removed from the tables to convert them from second to third normal form. To be in third normal form, a table should fulfill the condition where if A gives B, then B should not give C. It means that if a primary key gives a non-key value, then that non-key attribute should not give another non-key value.

A → B → C (here is transitive dependency)

To remove the dependency, the relation should be:

A → B

B → C

Checking the transitive dependency in 'Customer-2' table:

Customer-2 – (<u>Customer Id</u>, Customer Name, Contact, Address, Customer Category Id, Customer Category Name, Discount Rate)

Customer Id → Customer Name

Customer Id → Contact

Customer Id → Address

Customer Id → Customer Category Id → Customer Category Name, Discount Rate

To remove the transitive dependency encountered, we need to separate the dependent attribute.

So,

Customer Id → Customer Category Id

Customer Category Id → Customer Category Name

Customer Category Id → Discount Rate

Here, a new table ‘Customer Category’ is created.

Now, checking ‘Order-2’ table:

Order-2 – (<u>Order Id</u>, Customer Id*, Date, Quantity, Total Amount, Invoice Id, Payment Type)

Order Id → Date

Order Id → Quantity

Order Id → Total Amount

Order Id → Invoice Id → Payment Type

Separating the attributes to remove dependency.

Order Id → Invoice Id

Invoice Id → Payment Type

Here, a new table ‘Invoice’ is created.

Checking ‘Product-2’ table:

Product-2 – (Product Id, Customer Id*, Order Id*, Product Name, Unit Price, Stock, Description, Product Category Id, Product Category Name, Vendor Id, Vendor Name)

Product Id → Product Name

Product Id → Unit Price

Product Id → Stock

Product Id → Description

Product Id → Product Category Id → Product Category Name

To remove the transitive dependency, separate the dependent attribute.

Now,

Product Id → Product Category Id

Product Category Id → Product Category Name

Here, a new table 'Product Category' is created.

Product Id → Vendor Id → Vendor Name

Again, to remove transitive dependency:

Product Id → Vendor Id

Vendor Id → Vendor Name

Another table 'Vendor' is created.

So, the final tables after 3NF are:

Customer-3 – (<u>Customer Id</u>, Customer Category Id*, Customer Name, Contact, Address)

Customer_Category-3 – (<u>Customer Category Id</u>, Customer Category Name, Discount Rate)

Order-3 – (<u>Order Id</u>, Customer Id*, Invoice Id*, Date, Quantity, Total Amount)

Invoice-3 – (<u>Invoice Id</u>, Payment Type)

Product-3 – (<u>Product Id</u>, Customer Id*, Order Id*, Product Category Id*, Vendor Id*, Product Name, Unit Price, Stock, Description)

Product_Category-3 – (<u>Product Category Id</u>, Product Category Name)

Vendor-3 – (<u>Vendor Id</u>, Vendor Name)

Product_Customer_Order-3 – (Product Id*, Customer Id*, Order Id*)

Order_Customer-3 – (Order Id*, Customer Id*)

(This table is not made in Final ERD as its attributes which are 'Order Id' and 'Customer Id' are already given by 'Product_Customer_Order' table and it does not give any other non-key attributes.)

Product_Customer-3 – (Product Id*, Customer Id*)

(This table is not made in Final ERD as its attribues which are ‘Product Id’ and ‘Customer Id’ are already given by ‘Product_Customer_Order’ table and it does not give any other non-key attributes.)

Product_Order-3 – (Product Id*, Order Id*)

(This table is not made in Final ERD as its attributes which are ‘Product Id’ and ‘Order Id’ are already given by ‘Product_Customer_Order’ table and it does not give any other non-key attributes.)

4. Final ERD

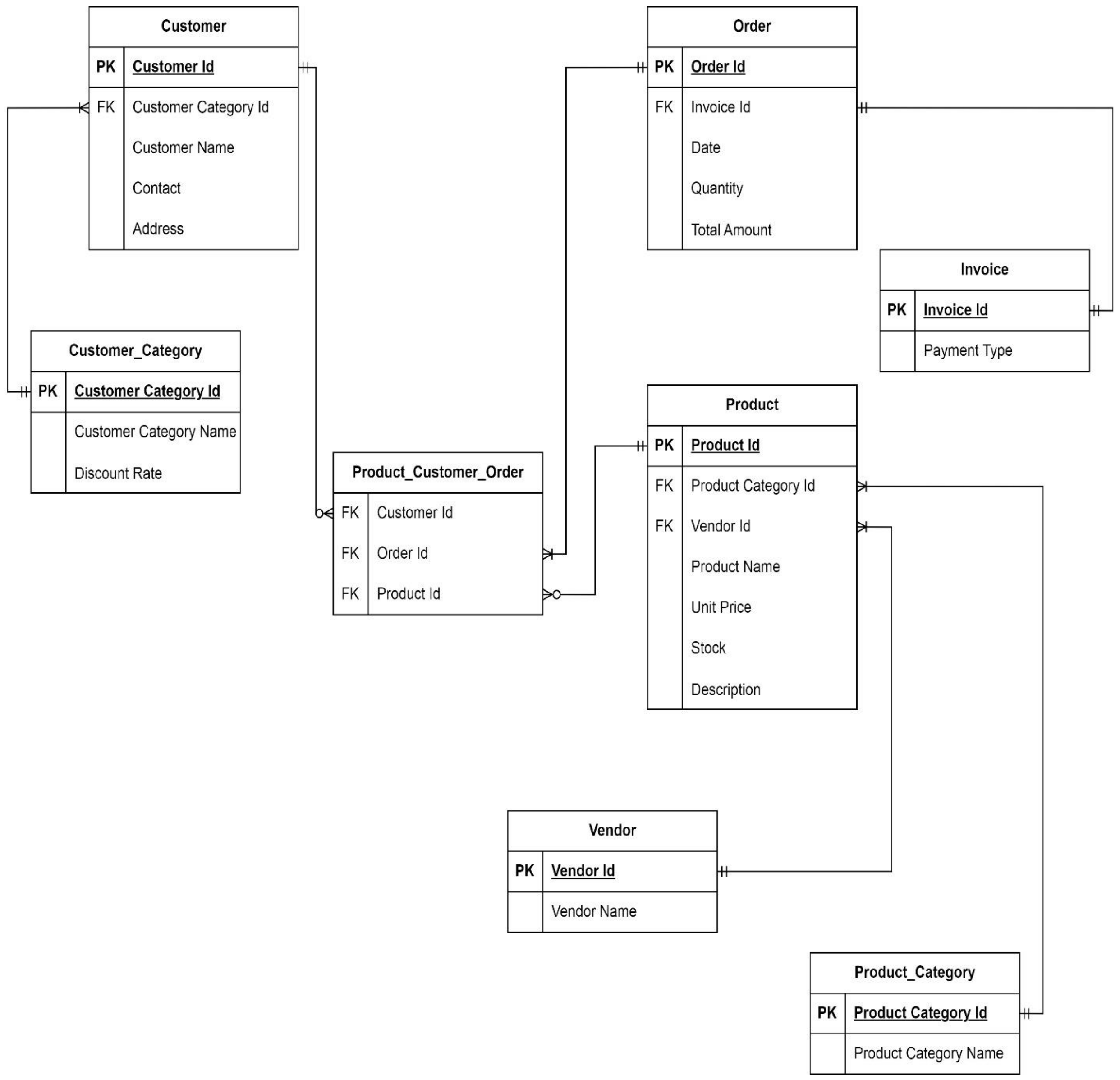


Figure 2 : Final ERD

The final ERD consists of eight tables ‘Customer’, ‘Customer_Category’, ‘Order’, ‘Invoice’, ‘Product’, ‘Product_Category’, ‘Vendor’ and ‘Product_Customer_Order’. Each tables provide various attributes and are defined through proper relation among each other. The Customer, Order and Product tables have one-to-many relation with Product_Customer_Order table which relates the tables with each other through the foreign keys. The Customer_Category and Product_Category tables also have one-to-many relations with Customer table and Product table respectively. Similarly, Vendor table has one-to-many relation with Product table and Invoice table has one-to-one relation with Order table.

5. IMPLEMENTATION

5.1 Creating and connecting to new user.

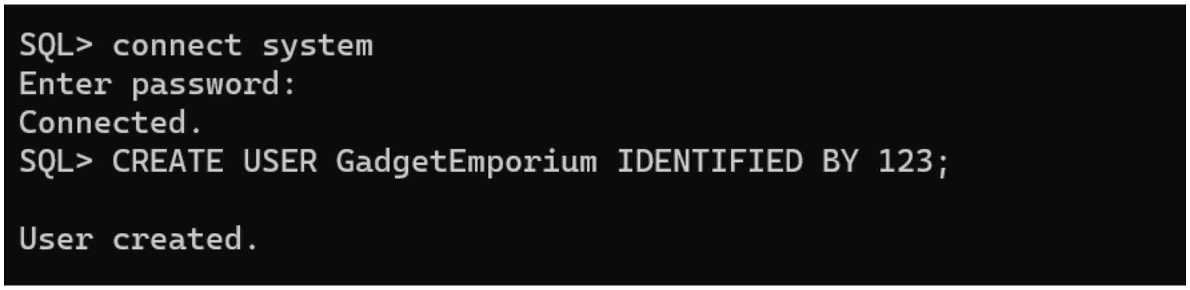
To create a new user, first we should connect to the system and enter the command as shown below:

Query:

connect system

enter password

CREATE USER GadgetEmporium IDENTIFIED BY 123;



```
SQL> connect system
Enter password:
Connected.
SQL> CREATE USER GadgetEmporium IDENTIFIED BY 123;

User created.
```

Figure 3 : creating a new user

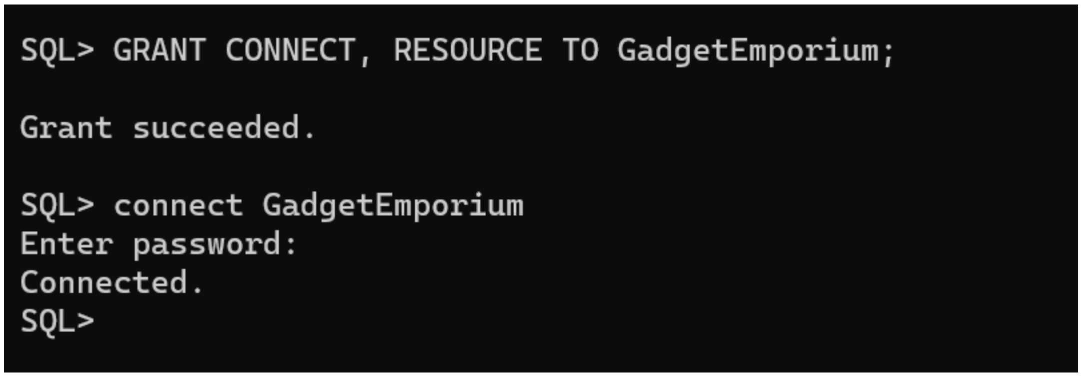
To work in a certain user, the user must be granted privileges to perform various tasks. So, to grant privileges to the user, we should enter the *GRANT CONNECT, RESOURCE TO GadgetEmporium* command as shown below:

Query:

GRANT CONNECT, RESOURCE TO GadgetEmporium;

connect GadgetEmporium

enter password



```
SQL> GRANT CONNECT, RESOURCE TO GadgetEmporium;

Grant succeeded.

SQL> connect GadgetEmporium
Enter password:
Connected.
SQL>
```

Figure 4 : granting privileges to the user

5.2 Creating all of the tables with required columns.

5.2.1 Creating Customer_Category table.

Customer_Category table contains customer category id as its primary key along with customer category name and discount rate as its other non-key attributes. The datatype for all columns is varchar.

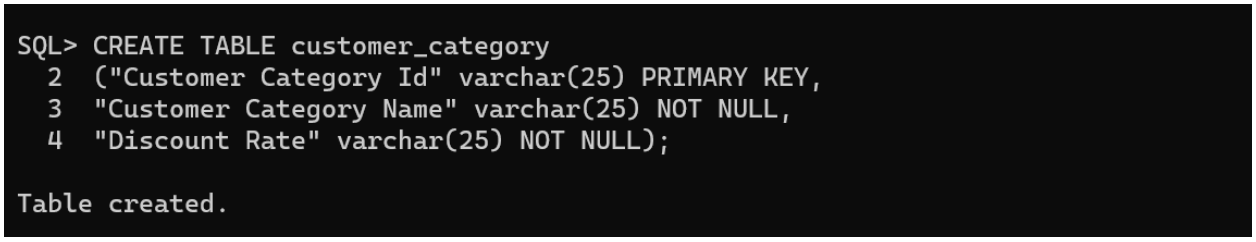
Query:

CREATE TABLE customer_category

(“Customer Category Id” varchar(25) PRIMARY KEY,

“Customer Category Name” varchar(25) NOT NULL,

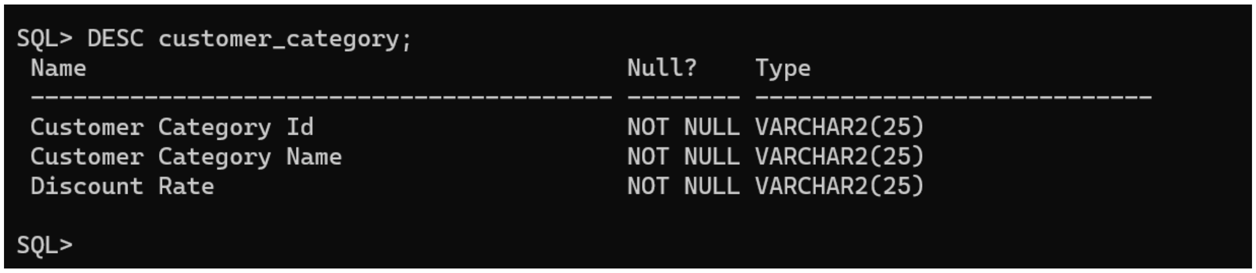
“Discount Rate” varchar(25) NOT NULL);



```
SQL> CREATE TABLE customer_category
  2  ("Customer Category Id" varchar(25) PRIMARY KEY,
  3  "Customer Category Name" varchar(25) NOT NULL,
  4  "Discount Rate" varchar(25) NOT NULL);

Table created.
```

Figure 5 : creating Customer_Category table

Use DESC command to see details of the created table.



```
SQL> DESC customer_category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Customer Category Id                      NOT NULL VARCHAR2(25)
 Customer Category Name                    NOT NULL VARCHAR2(25)
 Discount Rate                             NOT NULL VARCHAR2(25)

SQL>
```

Figure 6 : checking details of Customer_Category table

5.2.2 Creating Customer table.

Customer table contains customer id as its primary key. The table also contains other non-key attributes like customer name, contact and address. All columns have datatype of varchar. The table also has customer category id as a foreign key.

Query:

CREATE TABLE customer

("Customer Id" varchar(25) PRIMARY KEY,

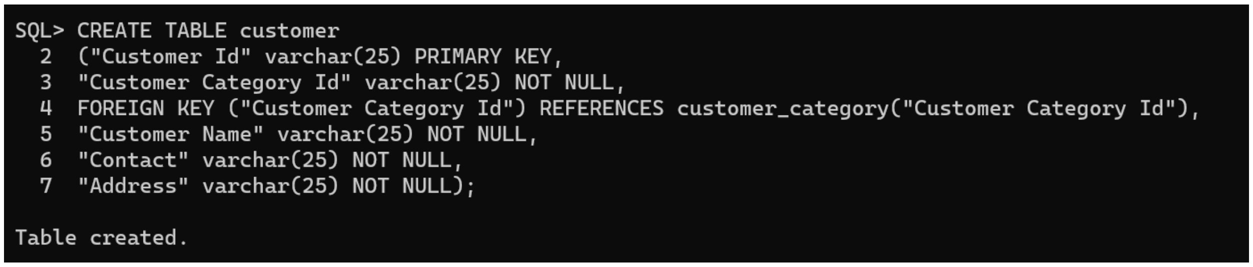
"Customer Category Id" varchar(25) NOT NULL,

FOREIGN KEY ("Customer Category Id") REFERENCES customer_category("Customer Category Id"),

"Customer Name" varchar(25) NOT NULL,

"Contact" varchar(25) NOT NULL,

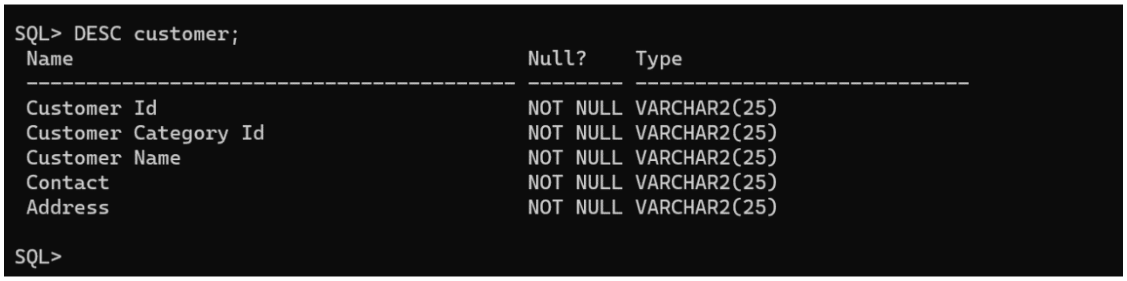
"Address" varchar(25) NOT NULL);



```
SQL> CREATE TABLE customer
  2  ("Customer Id" varchar(25) PRIMARY KEY,
  3  "Customer Category Id" varchar(25) NOT NULL,
  4  FOREIGN KEY ("Customer Category Id") REFERENCES customer_category("Customer Category Id"),
  5  "Customer Name" varchar(25) NOT NULL,
  6  "Contact" varchar(25) NOT NULL,
  7  "Address" varchar(25) NOT NULL);

Table created.
```

Figure 7 : creating Customer table

Use DESC command to see details of the created table.



```
SQL> DESC customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Customer Id                               NOT NULL VARCHAR2(25)
 Customer Category Id                      NOT NULL VARCHAR2(25)
 Customer Name                             NOT NULL VARCHAR2(25)
 Contact                                   NOT NULL VARCHAR2(25)
 Address                                   NOT NULL VARCHAR2(25)

SQL>
```

Figure 8 : checking details of Customer table

5.2.3 Creating Invoice table.

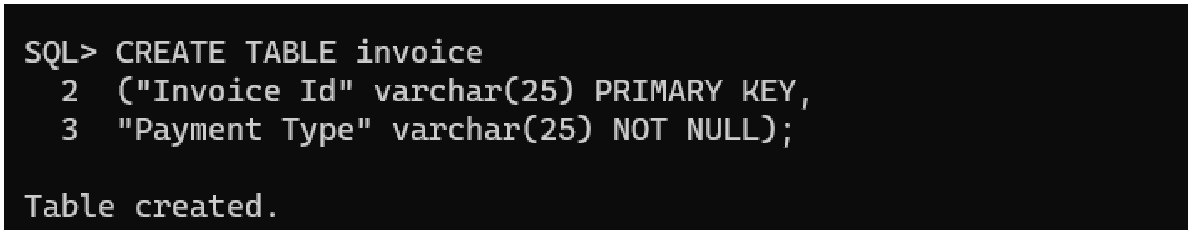
Invoice id is the primary key for invoice table with the datatype varchar. The table consists of another column named payment type with varchar as its datatype.

Query:

CREATE TABLE invoice

(“Invoice Id” varchar(25) PRIMARY KEY,

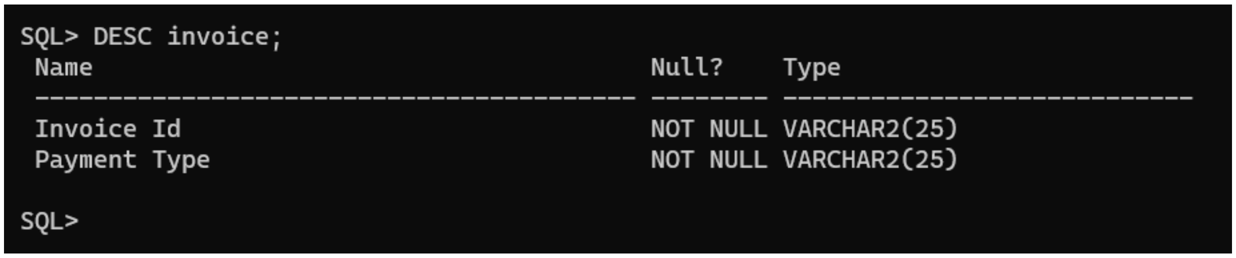
“Payment Type” varchar(25) NOT NULL);



```
SQL> CREATE TABLE invoice
  2  ("Invoice Id" varchar(25) PRIMARY KEY,
  3  "Payment Type" varchar(25) NOT NULL);

Table created.
```

Figure 9 : creating Invoice table

Use DESC command to see details of the created table.



```
SQL> DESC invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Invoice Id                                NOT NULL VARCHAR2(25)
 Payment Type                              NOT NULL VARCHAR2(25)

SQL>
```

Figure 10 : checking details of Invoice table

5.2.4 Creating Order table.

Order table has order id with the datatype varchar as its primary key. It also contains invoice id as a foreign key and has other non-key columns like date with datatype date and quantity and total amount columns with datatypes number.

Query:

CREATE TABLE "ORDER"

("Order Id" varchar(25) PRIMARY KEY,

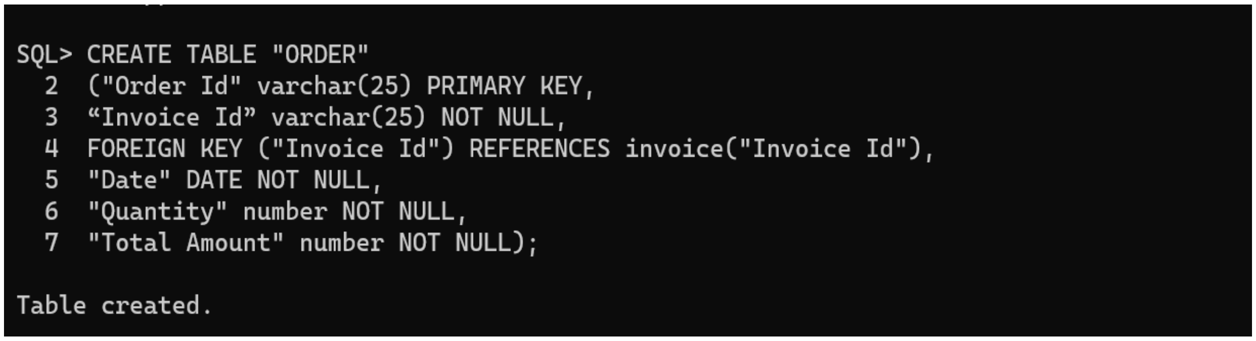
“Invoice Id” varchar(25) NOT NULL,

FOREIGN KEY ("Invoice Id") REFERENCES invoice("Invoice Id"),

"Date" DATE NOT NULL,

"Quantity" number NOT NULL,

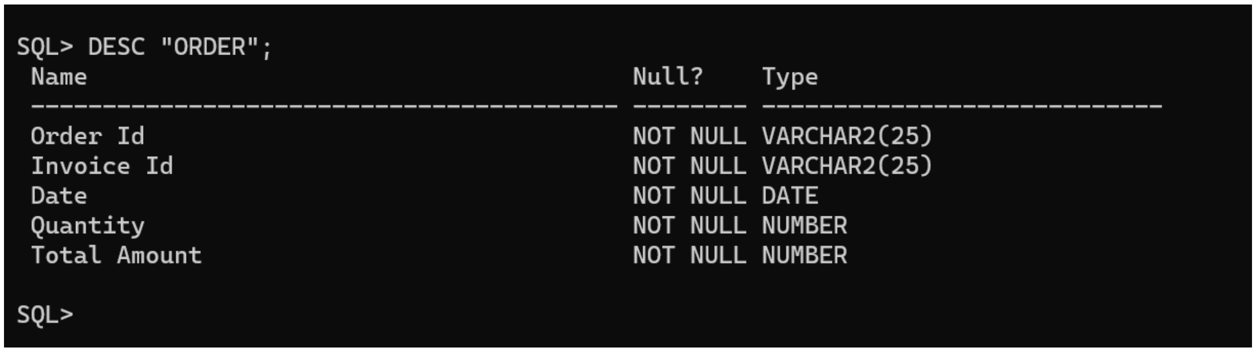
"Total Amount" number NOT NULL);



```
SQL> CREATE TABLE "ORDER"
  2  ("Order Id" varchar(25) PRIMARY KEY,
  3  “Invoice Id” varchar(25) NOT NULL,
  4  FOREIGN KEY ("Invoice Id") REFERENCES invoice("Invoice Id"),
  5  "Date" DATE NOT NULL,
  6  "Quantity" number NOT NULL,
  7  "Total Amount" number NOT NULL);

Table created.
```

Figure 11 : creating Order table

Use DESC command to see details of the created table.



```
SQL> DESC "ORDER";
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Order Id                                  NOT NULL VARCHAR2(25)
 Invoice Id                                NOT NULL VARCHAR2(25)
 Date                                      NOT NULL DATE
 Quantity                                  NOT NULL NUMBER
 Total Amount                              NOT NULL NUMBER

SQL>
```

Figure 12 : checking details of Order table

5.2.5 Creating Vendor table.

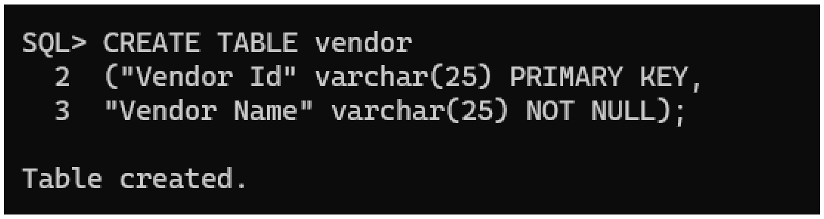
Vendor table contains of vendor id as a primary key and vendor name column with datatype varchar for both columns.

Query:

CREATE TABLE vendor

(“Vendor Id” varchar(25) PRIMARY KEY,

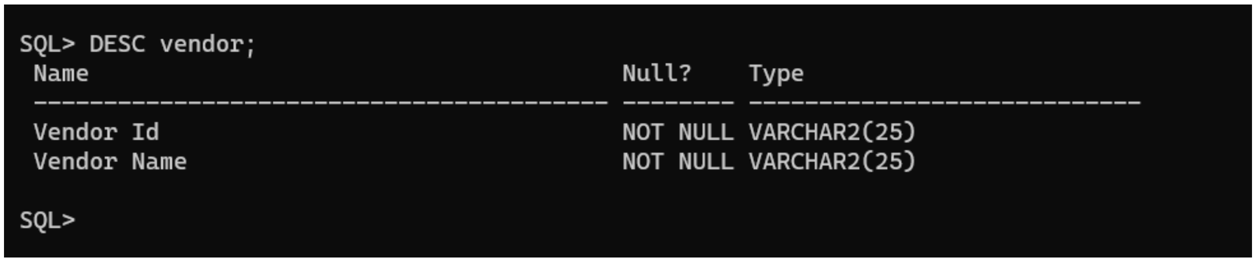
“Vendor Name” varchar(25) NOT NULL);



```
SQL> CREATE TABLE vendor
  2  ("Vendor Id" varchar(25) PRIMARY KEY,
  3  "Vendor Name" varchar(25) NOT NULL);

Table created.
```

Figure 13 : creating Vendor table

Use DESC command to see details of the created table.



```
SQL> DESC vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Vendor Id                                 NOT NULL VARCHAR2(25)
 Vendor Name                               NOT NULL VARCHAR2(25)

SQL>
```

Figure 14 : checking details of Vendor table

5.2.6 Creating Product_Category table.

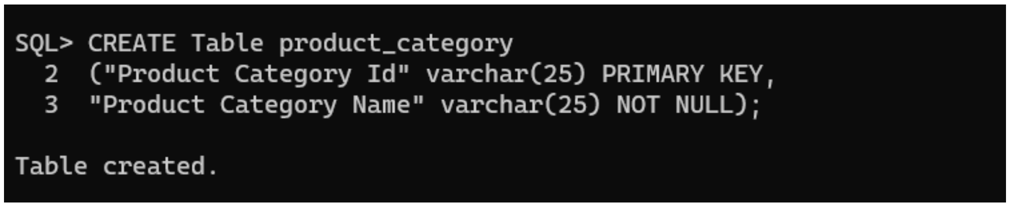
Product category id is the primary key of the Product_Category table. It also contains product category name column. Both columns have varchar datatype.

Query:

CREATE TABLE product_category

(“Product Category Id” varchar(25) PRIMARY KEY,

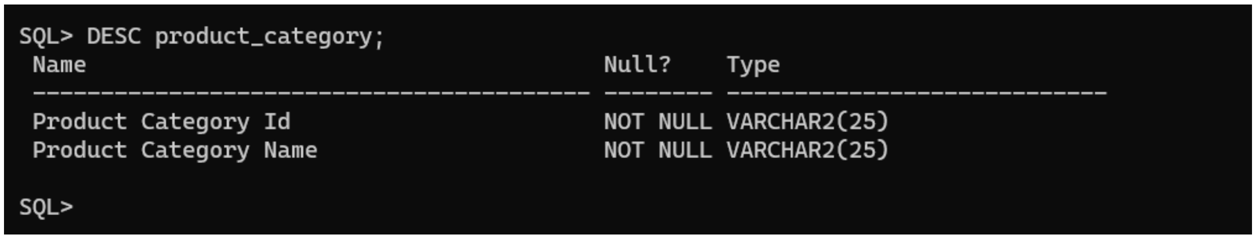
“Product Category Name” varchar(25) NOT NULL);



```
SQL> CREATE Table product_category
  2  ("Product Category Id" varchar(25) PRIMARY KEY,
  3  "Product Category Name" varchar(25) NOT NULL);

Table created.
```

Figure 15 : creating Product_Category table

Use DESC command to see details of the created table.



```
SQL> DESC product_category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Product Category Id                       NOT NULL VARCHAR2(25)
 Product Category Name                     NOT NULL VARCHAR2(25)

SQL>
```

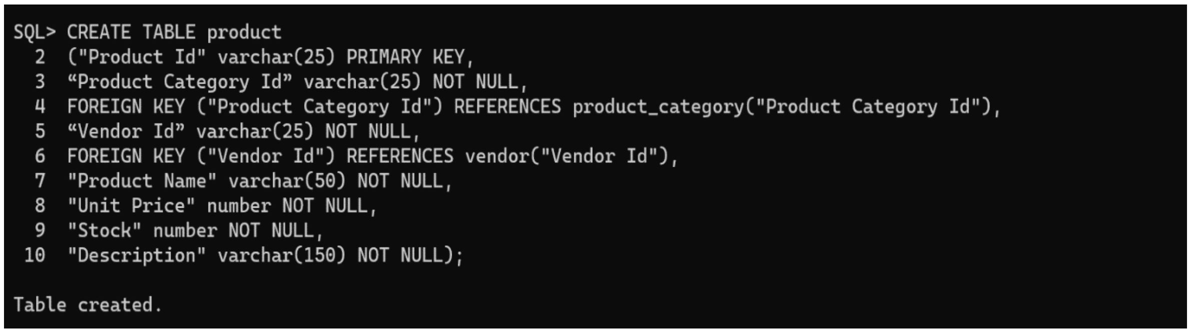
Figure 16 : checking details of Product_Category table

5.2.7 Creating Product table.

Product id is the primary key of the product table with datatype varchar. It contains product name and description columns with datatype varchar along with unit price and stock columns with datatype number. Product category id and vendor id columns are also present in the table as two foreign keys.

Query:

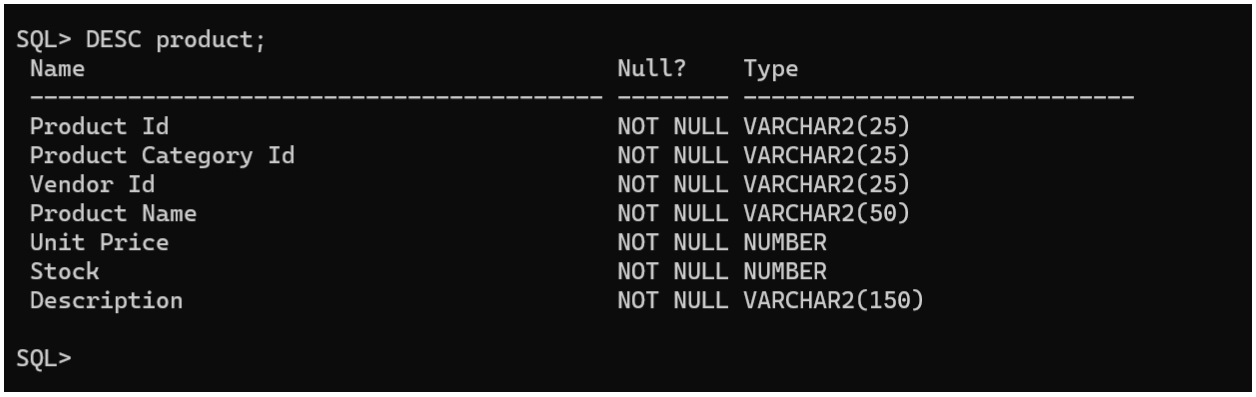
```
CREATE TABLE product
("Product Id" varchar(25) PRIMARY KEY,
“Product Category Id” varchar(25) NOT NULL,
FOREIGN KEY ("Product Category Id") REFERENCES product_category("Product Category Id"),
“Vendor Id” varchar(25) NOT NULL,
FOREIGN KEY ("Vendor Id") REFERENCES vendor("Vendor Id"),
"Product Name" varchar(50) NOT NULL,
"Unit Price" number NOT NULL,
"Stock" number NOT NULL,
"Description" varchar(150) NOT NULL);
```



```
SQL> CREATE TABLE product
("Product Id" varchar(25) PRIMARY KEY,
“Product Category Id” varchar(25) NOT NULL,
FOREIGN KEY ("Product Category Id") REFERENCES product_category("Product Category Id"),
“Vendor Id” varchar(25) NOT NULL,
FOREIGN KEY ("Vendor Id") REFERENCES vendor("Vendor Id"),
"Product Name" varchar(50) NOT NULL,
"Unit Price" number NOT NULL,
"Stock" number NOT NULL,
"Description" varchar(150) NOT NULL);

Table created.
```

Figure 17 : creating Product table

Use DESC command to see details of the created table.



```
SQL> DESC product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Product Id                                NOT NULL VARCHAR2(25)
 Product Category Id                       NOT NULL VARCHAR2(25)
 Vendor Id                                 NOT NULL VARCHAR2(25)
 Product Name                              NOT NULL VARCHAR2(50)
 Unit Price                                NOT NULL NUMBER
 Stock                                     NOT NULL NUMBER
 Description                               NOT NULL VARCHAR2(150)

SQL>
```

Figure 18 : checking details of Product table

5.2.8 Creating Product_Customer_Order table.

Product_Customer_Order table is a relational table which relates the customer, order and product tables. It contains three columns product id, customer id and order id with all being foreign keys.

Query:

CREATE TABLE product_customer_order

("Customer Id" varchar(25) NOT NULL,

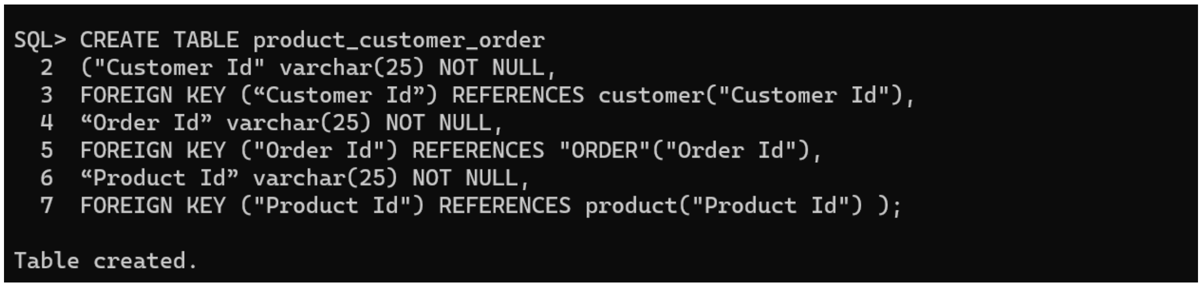
FOREIGN KEY (“Customer Id”) REFERENCES customer("Customer Id"),

“Order Id” varchar(25) NOT NULL,

FOREIGN KEY ("Order Id") REFERENCES "ORDER"("Order Id"),

“Product Id” varchar(25) NOT NULL,

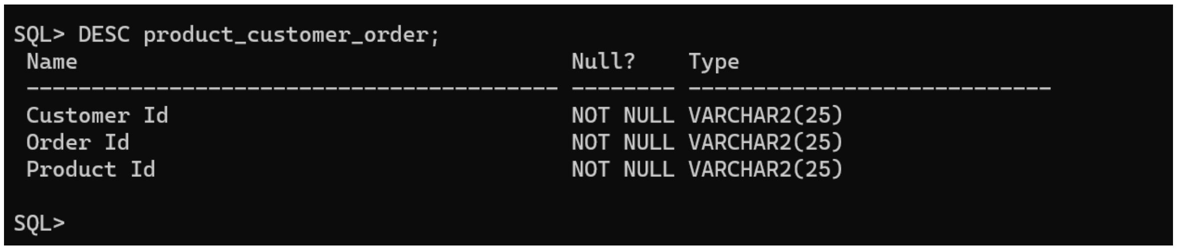
FOREIGN KEY ("Product Id") REFERENCES product("Product Id"));



```
SQL> CREATE TABLE product_customer_order
  2  ("Customer Id" varchar(25) NOT NULL,
  3  FOREIGN KEY (“Customer Id”) REFERENCES customer("Customer Id"),
  4  “Order Id” varchar(25) NOT NULL,
  5  FOREIGN KEY ("Order Id") REFERENCES "ORDER"("Order Id"),
  6  “Product Id” varchar(25) NOT NULL,
  7  FOREIGN KEY ("Product Id") REFERENCES product("Product Id") );

Table created.
```

Figure 19 : creating Product_Customer_Order table

Use DESC command to see details of the created table.



```
SQL> DESC product_customer_order;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 Customer Id                               NOT NULL VARCHAR2(25)
 Order Id                                  NOT NULL VARCHAR2(25)
 Product Id                                NOT NULL VARCHAR2(25)

SQL>
```

Figure 20 : checking details of Product_Customer_Order table

5.3 Inserting values in the tables.

5.3.1 Inserting values in Customer_Category table.

Insert the values in customer category table columns customer category id and customer category name.

.

Query:

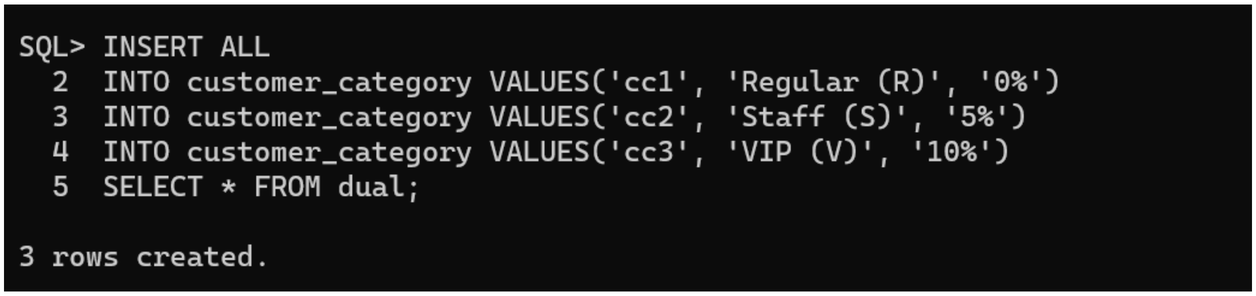
INSERT ALL

INTO customer_category VALUES('cc1', 'Regular (R)', '0%')

INTO customer_category VALUES('cc2', 'Staff (S)', '5%')

INTO customer_category VALUES('cc3', 'VIP (V)', '10%')

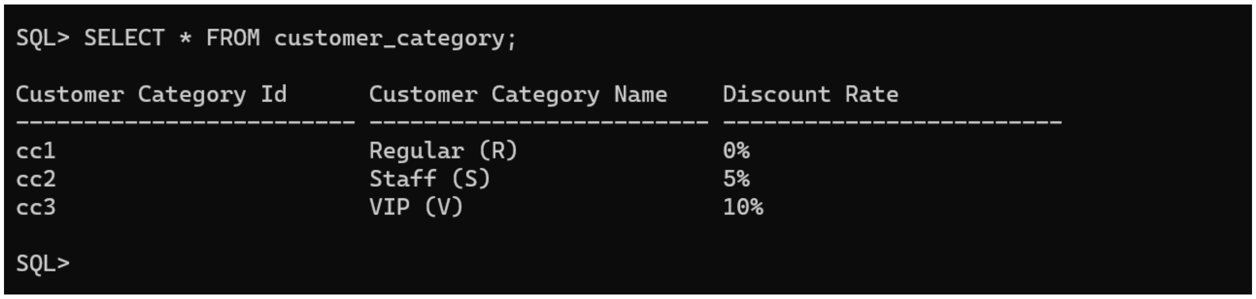
SELECT * FROM dual;



```
SQL> INSERT ALL
  2  INTO customer_category VALUES('cc1', 'Regular (R)', '0%')
  3  INTO customer_category VALUES('cc2', 'Staff (S)', '5%')
  4  INTO customer_category VALUES('cc3', 'VIP (V)', '10%')
  5  SELECT * FROM dual;

3 rows created.
```

Figure 21 : inserting values in Customer_Category table

Use SELECT * FROM tablename command to check the inserted values.



```
SQL> SELECT * FROM customer_category;

Customer Category Id      Customer Category Name    Discount Rate
------------------------- ------------------------- -------------------------
cc1                       Regular (R)               0%
cc2                       Staff (S)                 5%
cc3                       VIP (V)                   10%

SQL>
```

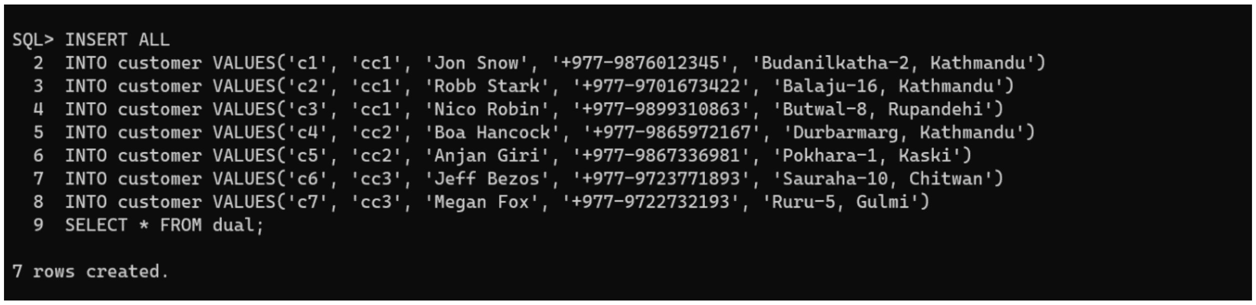
Figure 22 : checking the values inserted in Customer_Category table

5.3.2 Inserting values in Customer table.

Insert the suitable values customer id, customer category id, customer name, contact and address columns of customer table.

Query:

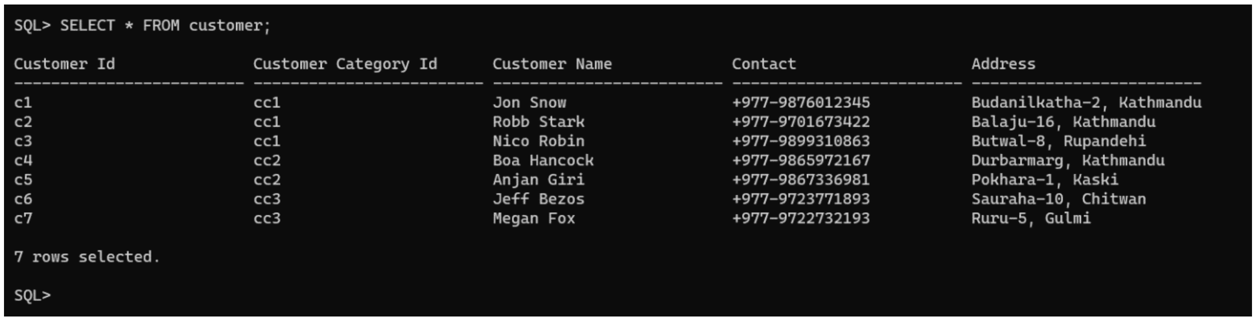
```
INSERT ALL
INTO customer VALUES('c1', 'cc1', 'Jon Snow', '+977-9876012345', 'Budanilkatha-2, Kathmandu')
INTO customer VALUES('c2', 'cc1', 'Robb Stark', '+977-9701673422', 'Balaju-16, Kathmandu')
INTO customer VALUES('c3', 'cc1', 'Nico Robin', '+977-9899310863', 'Butwal-8, Rupandehi')
INTO customer VALUES('c4', 'cc2', 'Boa Hancock', '+977-9865972167', 'Durbarmarg, Kathmandu')
INTO customer VALUES('c5', 'cc2', 'Anjan Giri', '+977-9867336981', 'Pokhara-1, Kaski')
INTO customer VALUES('c6', 'cc3', 'Jeff Bezos', '+977-9723771893', 'Sauraha-10, Chitwan')
INTO customer VALUES('c7', 'cc3', 'Megan Fox', '+977-9722732193', 'Ruru-5, Gulmi')
SELECT * FROM dual;
```



```
SQL> INSERT ALL
  INTO customer VALUES('c1', 'cc1', 'Jon Snow', '+977-9876012345', 'Budanilkatha-2, Kathmandu')
  INTO customer VALUES('c2', 'cc1', 'Robb Stark', '+977-9701673422', 'Balaju-16, Kathmandu')
  INTO customer VALUES('c3', 'cc1', 'Nico Robin', '+977-9899310863', 'Butwal-8, Rupandehi')
  INTO customer VALUES('c4', 'cc2', 'Boa Hancock', '+977-9865972167', 'Durbarmarg, Kathmandu')
  INTO customer VALUES('c5', 'cc2', 'Anjan Giri', '+977-9867336981', 'Pokhara-1, Kaski')
  INTO customer VALUES('c6', 'cc3', 'Jeff Bezos', '+977-9723771893', 'Sauraha-10, Chitwan')
  INTO customer VALUES('c7', 'cc3', 'Megan Fox', '+977-9722732193', 'Ruru-5, Gulmi')
  SELECT * FROM dual;

7 rows created.
```

Figure 23 : inserting values in Customer table

Use SELECT * FROM tablename command to check the inserted values.



```
SQL> SELECT * FROM customer;

Customer Id               Customer Category Id      Customer Name             Contact                   Address
------------------------- ------------------------- ------------------------- ------------------------- -------------------------
c1                        cc1                       Jon Snow                  +977-9876012345           Budanilkatha-2, Kathmandu
c2                        cc1                       Robb Stark                +977-9701673422           Balaju-16, Kathmandu
c3                        cc1                       Nico Robin                +977-9899310863           Butwal-8, Rupandehi
c4                        cc2                       Boa Hancock               +977-9865972167           Durbarmarg, Kathmandu
c5                        cc2                       Anjan Giri                +977-9867336981           Pokhara-1, Kaski
c6                        cc3                       Jeff Bezos                +977-9723771893           Sauraha-10, Chitwan
c7                        cc3                       Megan Fox                 +977-9722732193           Ruru-5, Gulmi

7 rows selected.

SQL>
```

Figure 24 : checking the values inserted in Customer table

5.3.3 Inserting values in Invoice table.

Suitable values in invoice id and payment type columns of invoice table are inserted.

Query:

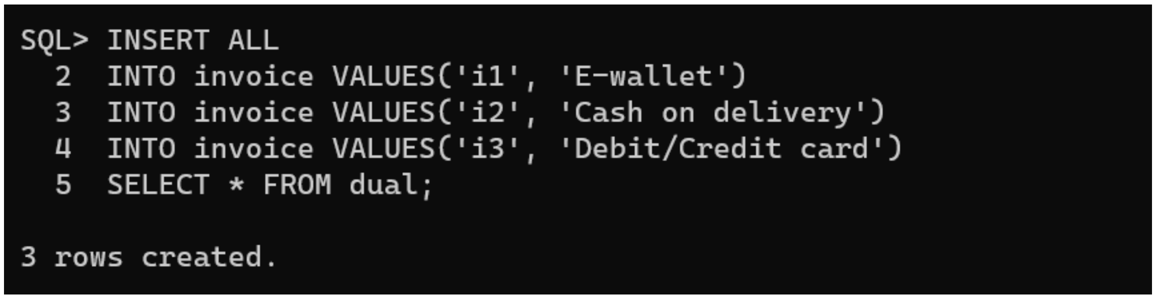
INSERT ALL

INTO invoice VALUES('i1', 'E-wallet')

INTO invoice VALUES('i2', 'Cash on delivery')

INTO invoice VALUES('i3', 'Debit/Credit card')

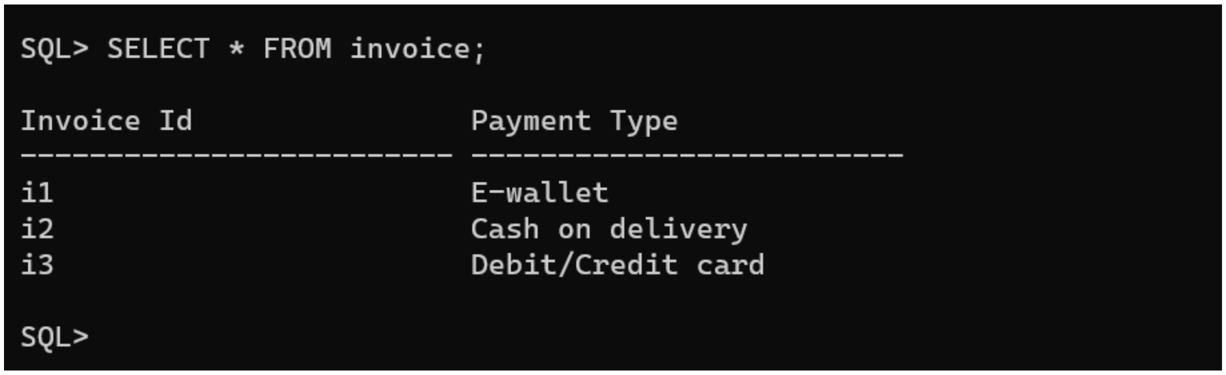
SELECT * FROM dual;



```
SQL> INSERT ALL
  2  INTO invoice VALUES('i1', 'E-wallet')
  3  INTO invoice VALUES('i2', 'Cash on delivery')
  4  INTO invoice VALUES('i3', 'Debit/Credit card')
  5  SELECT * FROM dual;

3 rows created.
```

Figure 25 : inserting values in Invoice table

Use SELECT * FROM tablename command to check the inserted values.



```
SQL> SELECT * FROM invoice;

Invoice Id                 Payment Type
-------------------------- -------------------------
i1                         E-wallet
i2                         Cash on delivery
i3                         Debit/Credit card

SQL>
```

Figure 26 : checking the values inserted in Invoice table

5.3.4 Inserting values in Order table.

Insert proper values in order id, invoice id, date, quantity and total amount columns of order table.

Query:

INSERT ALL

INTO "ORDER" VALUES('o1', 'i1', '03-JAN-2023', 5, 5000)

INTO "ORDER" VALUES('o2', 'i2', '22-MAR-2023', 8, 7200)

INTO "ORDER" VALUES('o3', 'i2', '12-AUG-2023', 2, 1200)

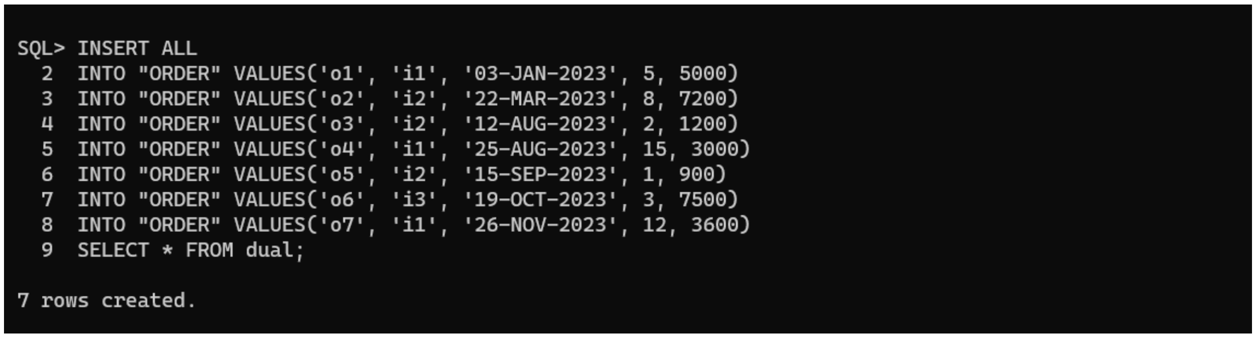
INTO "ORDER" VALUES('o4', 'i1', '25-AUG-2023', 15, 3000)

INTO "ORDER" VALUES('o5', 'i2', '15-SEP-2023', 1, 900)

INTO "ORDER" VALUES('o6', 'i3', '19-OCT-2023', 3, 7500)

INTO "ORDER" VALUES('o7', 'i1', '26-NOV-2023', 12, 3600)

SELECT * FROM dual;



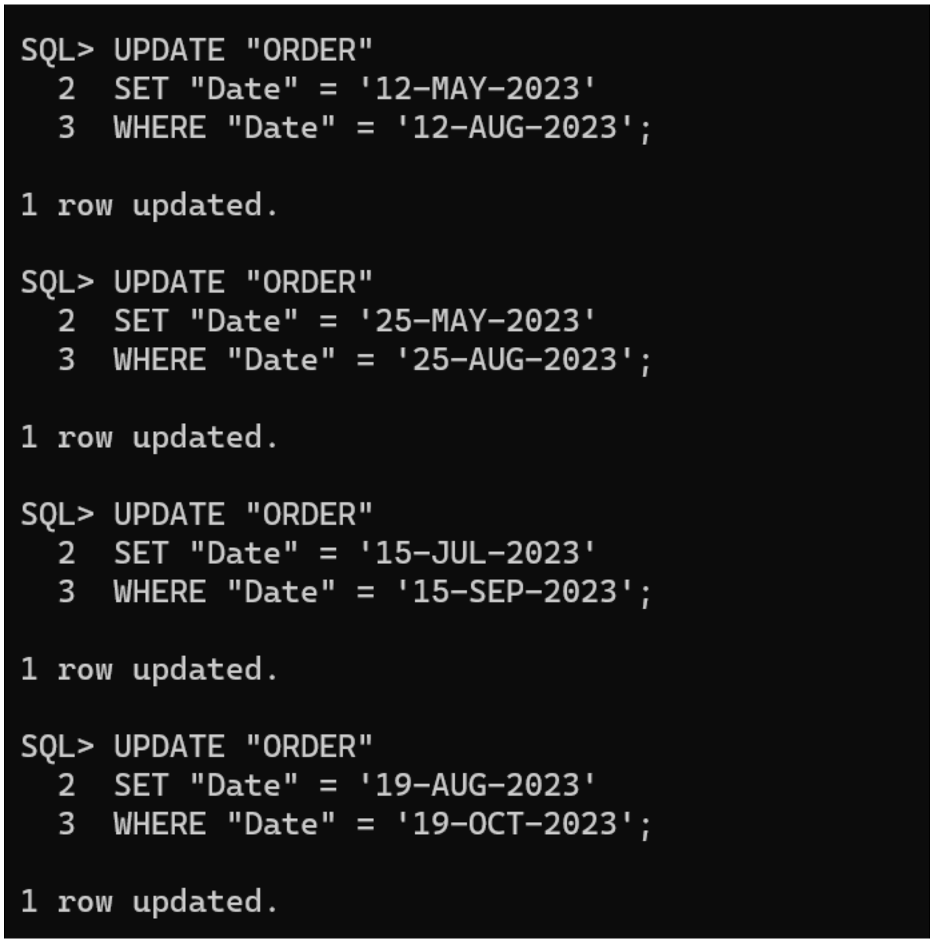
```
SQL> INSERT ALL
  2  INTO "ORDER" VALUES('o1', 'i1', '03-JAN-2023', 5, 5000)
  3  INTO "ORDER" VALUES('o2', 'i2', '22-MAR-2023', 8, 7200)
  4  INTO "ORDER" VALUES('o3', 'i2', '12-AUG-2023', 2, 1200)
  5  INTO "ORDER" VALUES('o4', 'i1', '25-AUG-2023', 15, 3000)
  6  INTO "ORDER" VALUES('o5', 'i2', '15-SEP-2023', 1, 900)
  7  INTO "ORDER" VALUES('o6', 'i3', '19-OCT-2023', 3, 7500)
  8  INTO "ORDER" VALUES('o7', 'i1', '26-NOV-2023', 12, 3600)
  9  SELECT * FROM dual;

7 rows created.
```

Figure 27 : inserting values in Order table

Now, as some data needs to be changed from the entered values, we should update the table.

Queries:

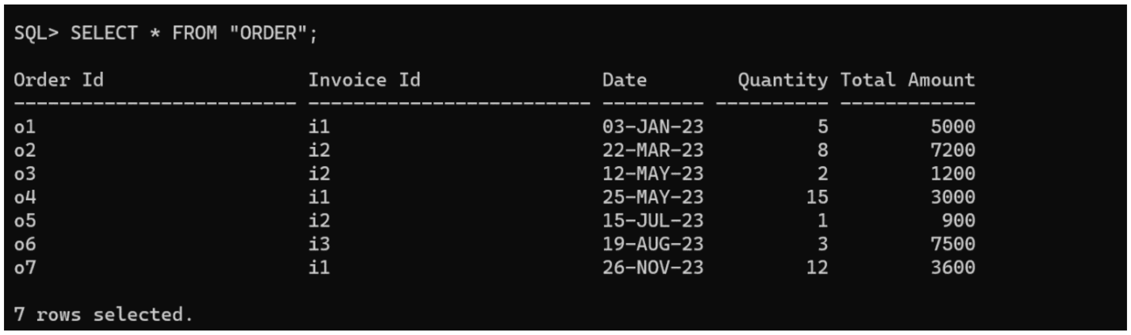
```
UPDATE "ORDER"

SET "Date" = '12-MAY-2023'

WHERE "Date" = '12-AUG-2023';


UPDATE "ORDER"

SET "Date" = '25-MAY-2023'

WHERE "Date" = '25-AUG-2023';


UPDATE "ORDER"

SET "Date" = '15-JUL-2023'

WHERE "Date" = '15-SEP-2023';


UPDATE "ORDER"

SET "Date" = '19-AUG-2023'

WHERE "Date" = '19-OCT-2023';
```



```
SQL> UPDATE "ORDER"
  2  SET "Date" = '12-MAY-2023'
  3  WHERE "Date" = '12-AUG-2023';

1 row updated.

SQL> UPDATE "ORDER"
  2  SET "Date" = '25-MAY-2023'
  3  WHERE "Date" = '25-AUG-2023';

1 row updated.

SQL> UPDATE "ORDER"
  2  SET "Date" = '15-JUL-2023'
  3  WHERE "Date" = '15-SEP-2023';

1 row updated.

SQL> UPDATE "ORDER"
  2  SET "Date" = '19-AUG-2023'
  3  WHERE "Date" = '19-OCT-2023';

1 row updated.
```

Figure 28 : updating the values in Order table

Use SELECT * FROM tablename command to check the inserted values.



```
SQL> SELECT * FROM "ORDER";

Order Id                  Invoice Id                Date        Quantity Total Amount
------------------------- ------------------------- --------- ---------- ------------
o1                        i1                        03-JAN-23          5         5000
o2                        i2                        22-MAR-23          8         7200
o3                        i2                        12-MAY-23          2         1200
o4                        i1                        25-MAY-23         15         3000
o5                        i2                        15-JUL-23          1          900
o6                        i3                        19-AUG-23          3         7500
o7                        i1                        26-NOV-23         12         3600

7 rows selected.
```

Figure 29 : checking the values inserted in Order table

5.3.5 Inserting values in Vendor table.

Insert the values in vendor table columns vendor id and vendor name.

Query:

INSERT ALL

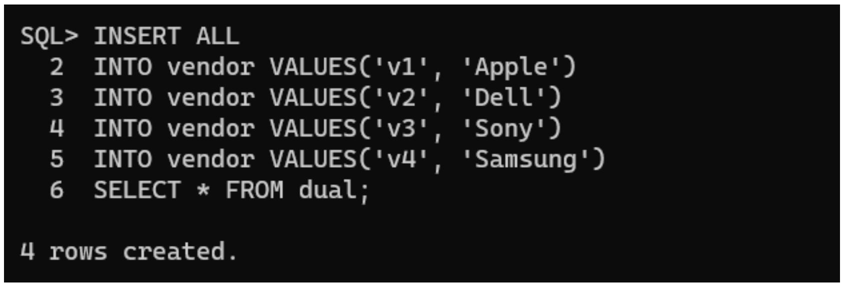
INTO vendor VALUES('v1', 'Apple')

INTO vendor VALUES('v2', 'Dell')

INTO vendor VALUES('v3', 'Sony')

INTO vendor VALUES('v4', 'Samsung')

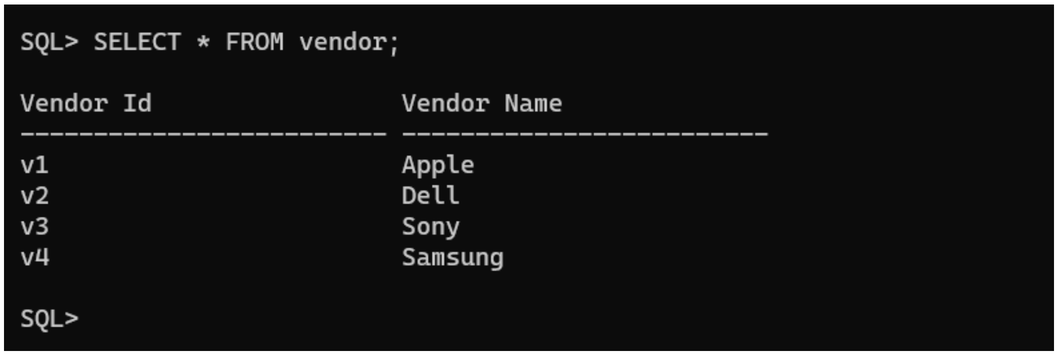
SELECT * FROM dual;



```
SQL> INSERT ALL
  2  INTO vendor VALUES('v1', 'Apple')
  3  INTO vendor VALUES('v2', 'Dell')
  4  INTO vendor VALUES('v3', 'Sony')
  5  INTO vendor VALUES('v4', 'Samsung')
  6  SELECT * FROM dual;

4 rows created.
```

Figure 30 : inserting values in Vendor table

Use SELECT * FROM tablename command to check the inserted values.



```
SQL> SELECT * FROM vendor;

Vendor Id                 Vendor Name
------------------------- -------------------------
v1                        Apple
v2                        Dell
v3                        Sony
v4                        Samsung

SQL>
```

Figure 31 : checking the values inserted in Vendor table

5.3.6 Inserting values in Product_Category table.

Insert proper values in product category id and product category name columns.

Query:

INSERT ALL

INTO product_category VALUES('pc1', 'Phone')

INTO product_category VALUES('pc2', 'Laptop')

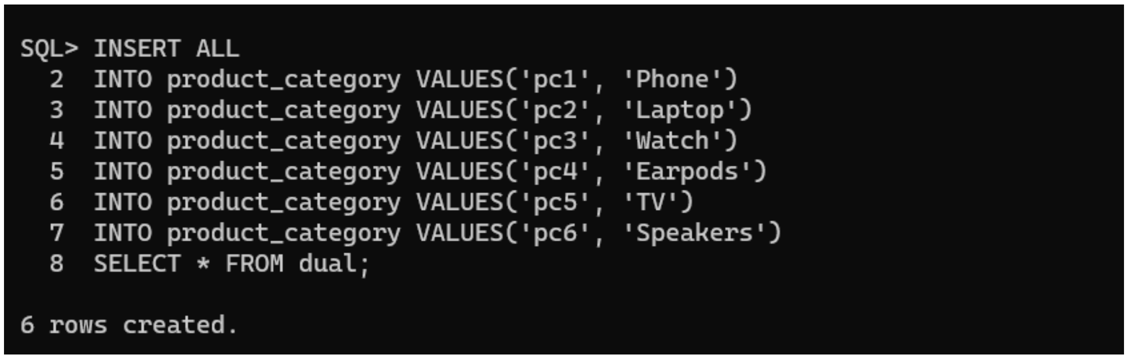
INTO product_category VALUES('pc3', 'Watch')

INTO product_category VALUES('pc4', 'Earpods')

INTO product_category VALUES('pc5', 'TV')

INTO product_category VALUES('pc6', 'Speakers')

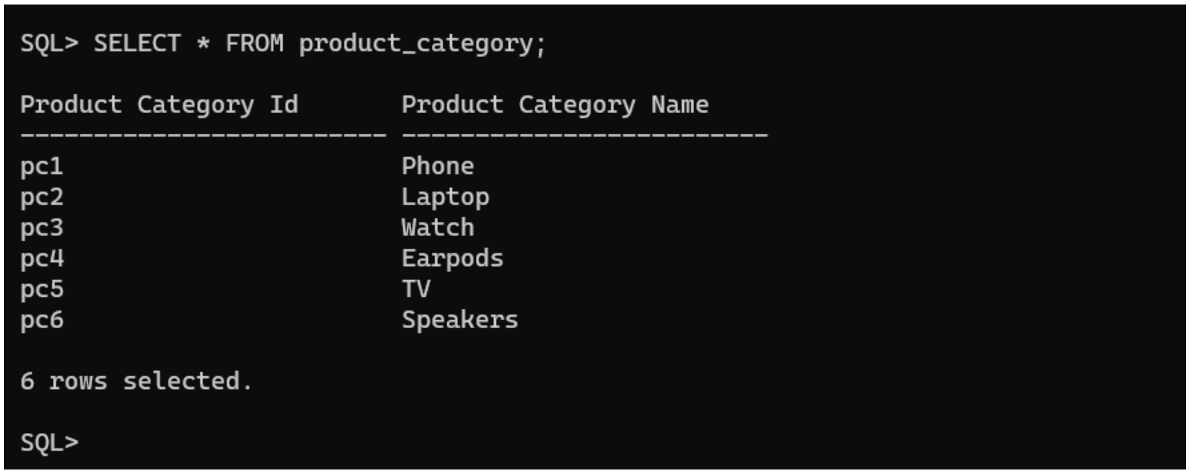
SELECT * FROM dual;



```
SQL> INSERT ALL
INTO product_category VALUES('pc1', 'Phone')
INTO product_category VALUES('pc2', 'Laptop')
INTO product_category VALUES('pc3', 'Watch')
INTO product_category VALUES('pc4', 'Earpods')
INTO product_category VALUES('pc5', 'TV')
INTO product_category VALUES('pc6', 'Speakers')
SELECT * FROM dual;

6 rows created.
```

Figure 32 : inserting values in Product_Category table

Use SELECT * FROM tablename command to check the inserted values.



```
SQL> SELECT * FROM product_category;

Product Category Id       Product Category Name
------------------------- -------------------------
pc1                       Phone
pc2                       Laptop
pc3                       Watch
pc4                       Earpods
pc5                       TV
pc6                       Speakers

6 rows selected.

SQL>
```

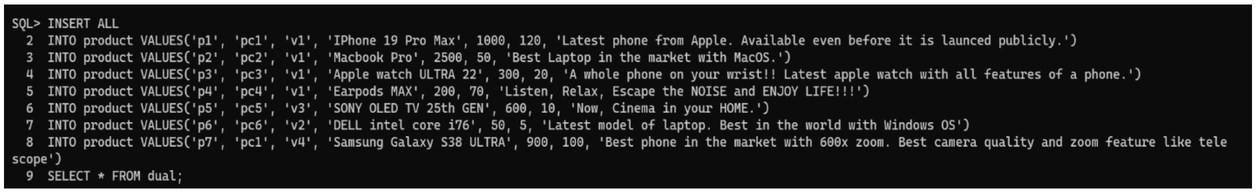
Figure 33 : checking the inserted values in Product_Category table

5.3.7 Inserting values in Product table.

Insert suitable values in product id, product category id, vendor id, product name, unit price and description columns of product table.

Query:

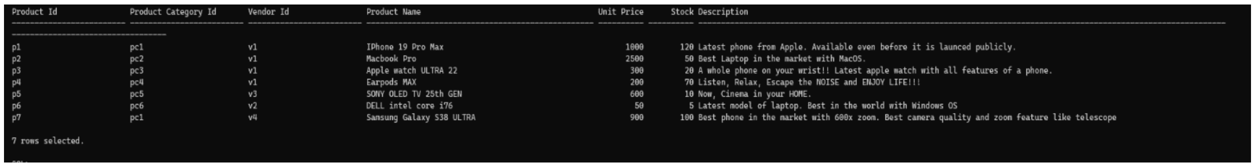
```
INSERT ALL

INTO product VALUES('p1', 'pc1', 'v1', 'IPhone 19 Pro Max', 1000, 120, 'Latest phone from Apple. Available even before it is launced publicly.')

INTO product VALUES('p2', 'pc2', 'v1', 'Macbook Pro', 2500, 50, 'Best Laptop in the market with MacOS.')

INTO product VALUES('p3', 'pc3', 'v1', 'Apple watch ULTRA 22', 300, 20, 'A whole phone on your wrist!! Latest apple watch with all features of a phone.')

INTO product VALUES('p4', 'pc4', 'v1', 'Earpods MAX', 200, 70, 'Listen, Relax, Escape the NOISE and ENJOY LIFE!!!')

INTO product VALUES('p5', 'pc5', 'v3', 'SONY OLED TV 25th GEN', 600, 10, 'Now, Cinema in your HOME.')

INTO product VALUES('p6', 'pc6', 'v2', 'DELL intel core i76', 50, 5, 'Latest model of laptop. Best in the world with Windows OS')

INTO product VALUES('p7', 'pc1', 'v4', 'Samsung Galaxy S38 ULTRA', 900, 100, 'Best phone in the market with 600x zoom. Best camera quality and zoom feature like telescope')

SELECT * FROM dual;
```



```
SQL> INSERT ALL
  INTO product VALUES('p1', 'pc1', 'v1', 'IPhone 19 Pro Max', 1000, 120, 'Latest phone from Apple. Available even before it is launced publicly.')
  INTO product VALUES('p2', 'pc2', 'v1', 'Macbook Pro', 2500, 50, 'Best Laptop in the market with MacOS.')
  INTO product VALUES('p3', 'pc3', 'v1', 'Apple watch ULTRA 22', 300, 20, 'A whole phone on your wrist!! Latest apple watch with all features of a phone.')
  INTO product VALUES('p4', 'pc4', 'v1', 'Earpods MAX', 200, 70, 'Listen, Relax, Escape the NOISE and ENJOY LIFE!!!')
  INTO product VALUES('p5', 'pc5', 'v3', 'SONY OLED TV 25th GEN', 600, 10, 'Now, Cinema in your HOME.')
  INTO product VALUES('p6', 'pc6', 'v2', 'DELL intel core i76', 50, 5, 'Latest model of laptop. Best in the world with Windows OS')
  INTO product VALUES('p7', 'pc1', 'v4', 'Samsung Galaxy S38 ULTRA', 900, 100, 'Best phone in the market with 600x zoom. Best camera quality and zoom feature like tele
scope')
  SELECT * FROM dual;
```

Figure 34 : inserting values in Product table

Use SELECT * FROM tablename command to check the inserted values.



```
Product Id               Product Category Id      Vendor Id                Product Name                                       Unit Price      Stock Description
------------------------ ------------------------ ------------------------ -------------------------------------------------- ---------- ---------- --------------------------------------------------------------------------------
--------------------------------

p1                       pc1                      v1                       IPhone 19 Pro Max                                        1000        120 Latest phone from Apple. Available even before it is launced publicly.
p2                       pc2                      v1                       Macbook Pro                                              2500         50 Best Laptop in the market with MacOS.
p3                       pc3                      v1                       Apple watch ULTRA 22                                      300         20 A whole phone on your wrist!! Latest apple watch with all features of a phone.
p4                       pc4                      v1                       Earpods MAX                                               200         70 Listen, Relax, Escape the NOISE and ENJOY LIFE!!!
p5                       pc5                      v3                       SONY OLED TV 25th GEN                                     600         10 Now, Cinema in your HOME.
p6                       pc6                      v2                       DELL intel core i76                                        50          5 Latest model of laptop. Best in the world with Windows OS
p7                       pc1                      v4                       Samsung Galaxy S38 ULTRA                                  900        100 Best phone in the market with 600x zoom. Best camera quality and zoom feature like telescope

7 rows selected.
```

Figure 35 : checking the values inserted in Product table

5.3.8 Inserting values in Product_Customer_Order table.

Insert appropriate values in customer id, order id and product id columns.

Query:

INSERT ALL

INTO product_customer_order VALUES(‘c1’, ‘o3’, ‘p5’)

INTO product_customer_order VALUES(‘c2’, ‘o2’, ‘p7’)

INTO product_customer_order VALUES(‘c3’, ‘o6’, ‘p2’)

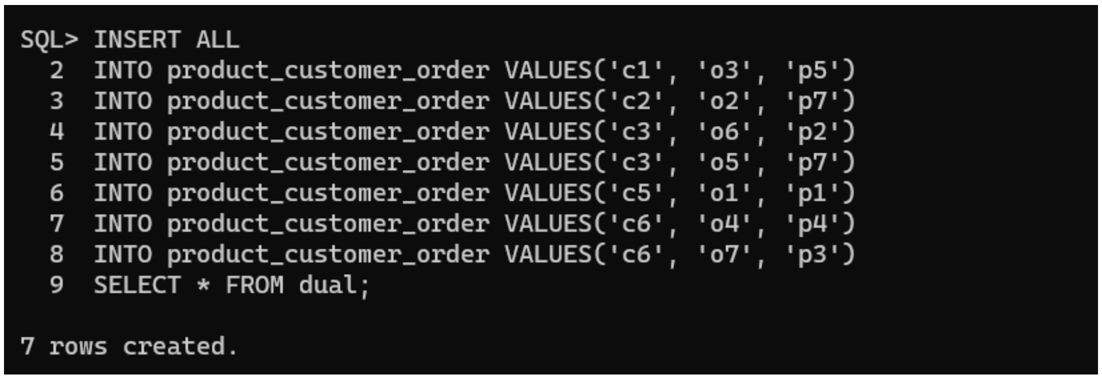
INTO product_customer_order VALUES(‘c3’, ‘o5’, ‘p7’)

INTO product_customer_order VALUES(‘c5’, ‘o1’, ‘p1’)

INTO product_customer_order VALUES(‘c6’, ‘o4’, ‘p4’)

INTO product_customer_order VALUES(‘c6’, ‘o7’, ‘p3’)

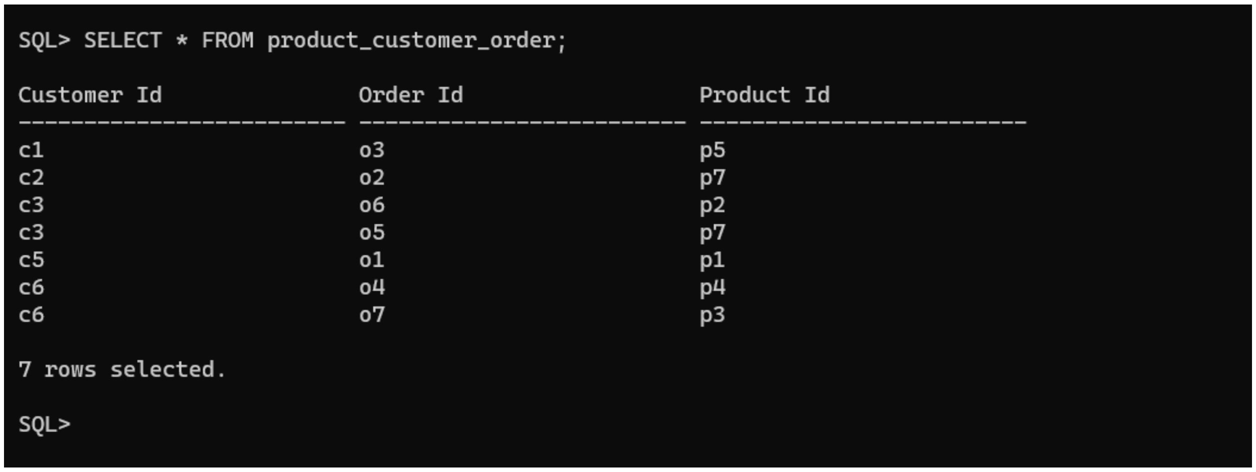
SELECT * FROM dual;



```
SQL> INSERT ALL
  2  INTO product_customer_order VALUES('c1', 'o3', 'p5')
  3  INTO product_customer_order VALUES('c2', 'o2', 'p7')
  4  INTO product_customer_order VALUES('c3', 'o6', 'p2')
  5  INTO product_customer_order VALUES('c3', 'o5', 'p7')
  6  INTO product_customer_order VALUES('c5', 'o1', 'p1')
  7  INTO product_customer_order VALUES('c6', 'o4', 'p4')
  8  INTO product_customer_order VALUES('c6', 'o7', 'p3')
  9  SELECT * FROM dual;

7 rows created.
```

Figure 36 : inserting values in Product_Customer_Order table

Use SELECT * FROM tablename command to check the inserted values.



```
SQL> SELECT * FROM product_customer_order;

Customer Id               Order Id                  Product Id
------------------------- ------------------------- -------------------------
c1                        o3                        p5
c2                        o2                        p7
c3                        o6                        p2
c3                        o5                        p7
c5                        o1                        p1
c6                        o4                        p4
c6                        o7                        p3

7 rows selected.

SQL>
```

Figure 37 : checking the values inserted in Product_Customer_Order table

6. QUERIES

6.1 INFORMATION QUERIES

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

Query:

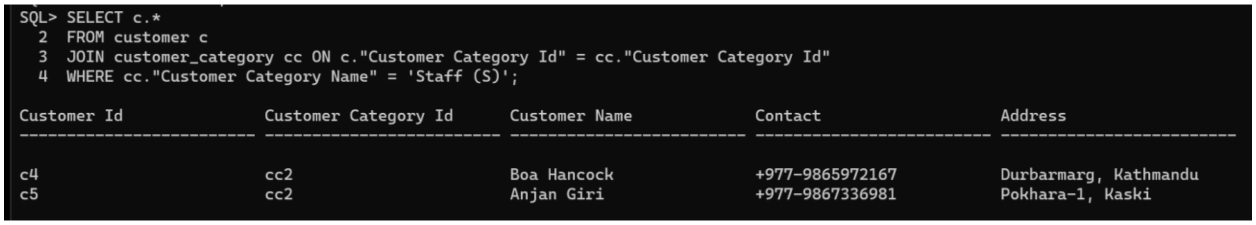
SELECT c.*

FROM customer c

JOIN customer_category cc ON c."Customer Category Id" = cc."Customer Category Id"

WHERE cc."Customer Category Name" = 'Staff (S)';

Here, the command lists all customers who are also the staff of the company. The 'c' in the command acts as an alias for Customer table and 'cc' acts as the alias for Customer_Category table.



```
SQL> SELECT c.*
  2  FROM customer c
  3  JOIN customer_category cc ON c."Customer Category Id" = cc."Customer Category Id"
  4  WHERE cc."Customer Category Name" = 'Staff (S)';

Customer Id               Customer Category Id      Customer Name             Contact                   Address
------------------------- ------------------------- ------------------------- ------------------------- -------------------------

c4                        cc2                       Boa Hancock               +977-9865972167           Durbarmarg, Kathmandu
c5                        cc2                       Anjan Giri                +977-9867336981           Pokhara-1, Kaski
```

Figure 38 : listing the customers who are also staff at the company

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

Query:

SELECT *

FROM "ORDER"

WHERE "Order Id" IN (

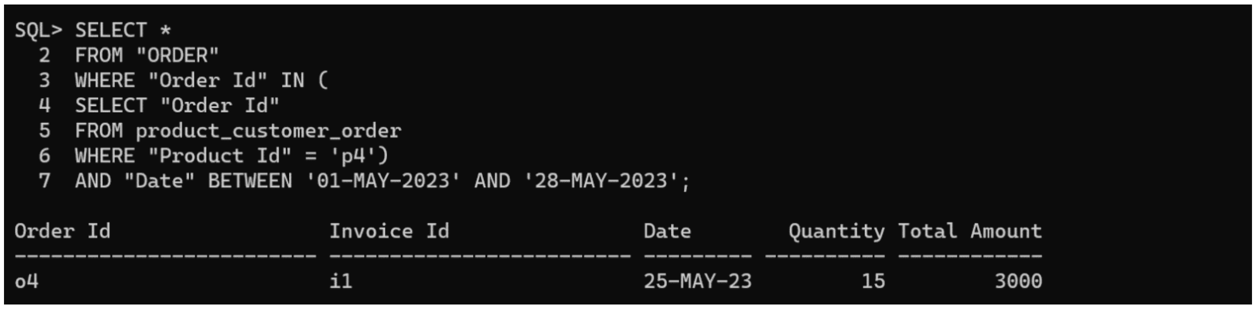
SELECT "Order Id"

FROM product_customer_order

WHERE "Product Id" = 'p4')

AND "Date" BETWEEN '01-MAY-2023' AND '28-MAY-2023';

Here, the query lists all orders made for ‘Earpods MAX’ product which has the product id ‘p4’ between the dates ’01-05-2023’ and ’28-05-2023’. First order id where the earpods max (product id ‘p4’) is ordered is selected from product_customer_order table and then in order table, the order id is selected between the provided dates to give the required result.



```
SQL> SELECT *
  2  FROM "ORDER"
  3  WHERE "Order Id" IN (
  4  SELECT "Order Id"
  5  FROM product_customer_order
  6  WHERE "Product Id" = 'p4')
  7  AND "Date" BETWEEN '01-MAY-2023' AND '28-MAY-2023';

Order Id                  Invoice Id                Date        Quantity Total Amount
------------------------- ------------------------- --------- ---------- ------------
o4                        i1                        25-MAY-23         15         3000
```

Figure 39 : listing all the orders made for 'Earpods MAX' between the dates 01-05-2023 till 28-05-2023.

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

Query:

SELECT c."Customer Id", c."Customer Name", o."Order Id", o."Date", o."Quantity", o."Total Amount"

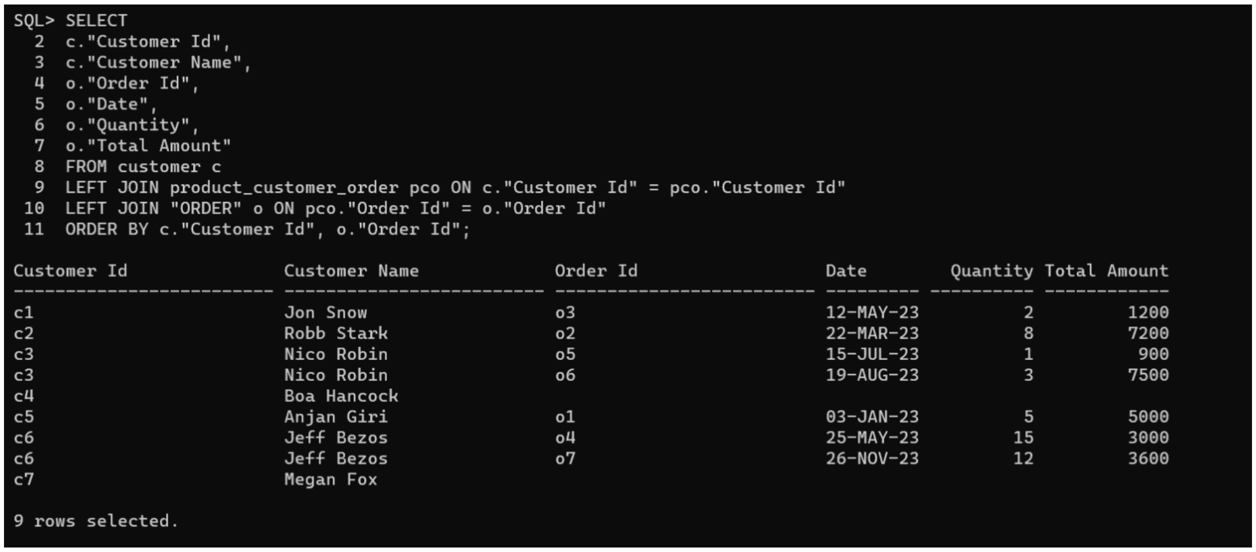
FROM customer c

LEFT JOIN product_customer_order pco ON c."Customer Id" = pco."Customer Id"

LEFT JOIN "ORDER" o ON pco."Order Id" = o."Order Id"

ORDER BY c."Customer Id", o."Order Id";

The query lists all customers with their order details along with the customers who have not ordered any products yet. Here, ‘c’, ‘o’, and ‘pco’ are the aliases for customer, order and product_customer_order tables respectively. The columns relate with each other through the relation table product_customer_order table.



```
SQL> SELECT
c."Customer Id",
c."Customer Name",
o."Order Id",
o."Date",
o."Quantity",
o."Total Amount"
FROM customer c
LEFT JOIN product_customer_order pco ON c."Customer Id" = pco."Customer Id"
LEFT JOIN "ORDER" o ON pco."Order Id" = o."Order Id"
ORDER BY c."Customer Id", o."Order Id";

Customer Id               Customer Name             Order Id                  Date        Quantity Total Amount
------------------------- ------------------------- ------------------------- --------- ---------- ------------
c1                        Jon Snow                  o3                        12-MAY-23          2         1200
c2                        Robb Stark                o2                        22-MAR-23          8         7200
c3                        Nico Robin                o5                        15-JUL-23          1          900
c3                        Nico Robin                o6                        19-AUG-23          3         7500
c4                        Boa Hancock
c5                        Anjan Giri                o1                        03-JAN-23          5         5000
c6                        Jeff Bezos                o4                        25-MAY-23         15         3000
c6                        Jeff Bezos                o7                        26-NOV-23         12         3600
c7                        Megan Fox

9 rows selected.
```

Figure 40 : listing all customers with their order details along with the customers who have not ordered any products yet

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

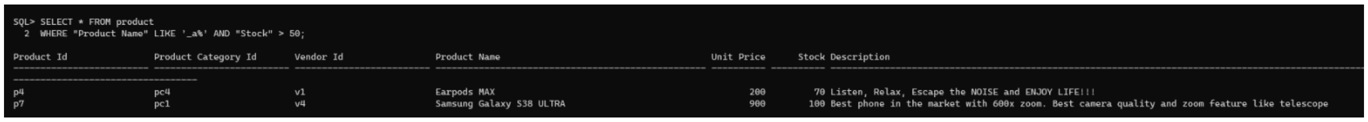
Query:

SELECT * FROM product

WHERE "Product Name" LIKE '_a%' AND "Stock" > 50;

Here, the query lists the product details that have second letter ‘a’ in its name and have a stock quantity more than 50.

_ denotes one character and % denotes 0 or many characters.



```
SQL> SELECT * FROM product
  2  WHERE "Product Name" LIKE '_a%' AND "Stock" > 50;

Product Id                Product Category Id       Vendor Id                 Product Name                                       Unit Price      Stock Description
------------------------- ------------------------- ------------------------- -------------------------------------------------- ---------- ---------- ----------------------------------------------------------------------------------------------------
----------------------------------
p4                        pc4                       v1                        Earpods MAX                                               200         70 Listen, Relax, Escape the NOISE and ENJOY LIFE!!!
p7                        pc1                       v4                        Samsung Galaxy S38 ULTRA                                  900        100 Best phone in the market with 600x zoom. Best camera quality and zoom feature like telescope
```

Figure 41 : listing all product details that have second letter 'a' in their name and have stock more than 50

6.1.5 Find out the customer who has ordered recently.

Query:

SELECT * FROM (

SELECT customer."Customer Id", customer."Customer Name",

"ORDER"."Order Id", "ORDER"."Date"

FROM customer

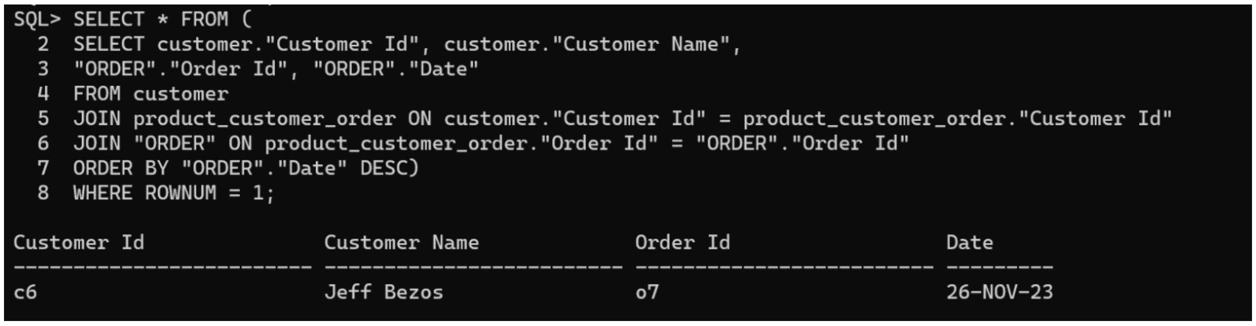
JOIN product_customer_order ON customer."Customer Id" = product_customer_order."Customer Id"

JOIN "ORDER" ON product_customer_order."Order Id" = "ORDER"."Order Id"

ORDER BY "ORDER"."Date" DESC)

WHERE ROWNUM = 1;

The query finds the customer who has ordered most recently and provides the ordered date along with name and id of the customer. The ORDER BY clause is used to sort the results in descending order and ROWNUM =1 is written to ensure that only the most recent order date is displayed in result.



```
SQL> SELECT * FROM (
  2  SELECT customer."Customer Id", customer."Customer Name",
  3  "ORDER"."Order Id", "ORDER"."Date"
  4  FROM customer
  5  JOIN product_customer_order ON customer."Customer Id" = product_customer_order."Customer Id"
  6  JOIN "ORDER" ON product_customer_order."Order Id" = "ORDER"."Order Id"
  7  ORDER BY "ORDER"."Date" DESC)
  8  WHERE ROWNUM = 1;

Customer Id               Customer Name             Order Id                  Date
------------------------- ------------------------- ------------------------- ---------
c6                        Jeff Bezos                o7                        26-NOV-23
```

Figure 42 : customer who has ordered recently along with the order date

6.2 TRANSACTION QUERIES

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

Query:

SELECT EXTRACT(YEAR FROM "Date") AS y,

EXTRACT(MONTH FROM "Date") AS m,

SUM("Total Amount") AS Total_Revenue

FROM "ORDER"

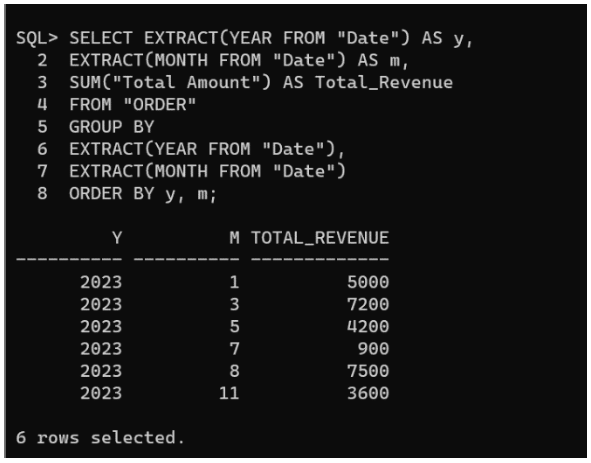
GROUP BY

EXTRACT(YEAR FROM "Date"),

EXTRACT(MONTH FROM "Date")

ORDER BY y, m;

The query shows the total revenue of the company for each month. Here, SUM is used to calculate the amount of each month and display the amount of the month as total_amount. GROUP BY groups the results by the date and ORDER BY orders the result based on the date.



```
SQL> SELECT EXTRACT(YEAR FROM "Date") AS y,
  2  EXTRACT(MONTH FROM "Date") AS m,
  3  SUM("Total Amount") AS Total_Revenue
  4  FROM "ORDER"
  5  GROUP BY
  6  EXTRACT(YEAR FROM "Date"),
  7  EXTRACT(MONTH FROM "Date")
  8  ORDER BY y, m;

         Y          M TOTAL_REVENUE
---------- ---------- -------------
      2023          1          5000
      2023          3          7200
      2023          5          4200
      2023          7           900
      2023          8          7500
      2023         11          3600

6 rows selected.
```

Figure 43 : total revenue of the company for each month

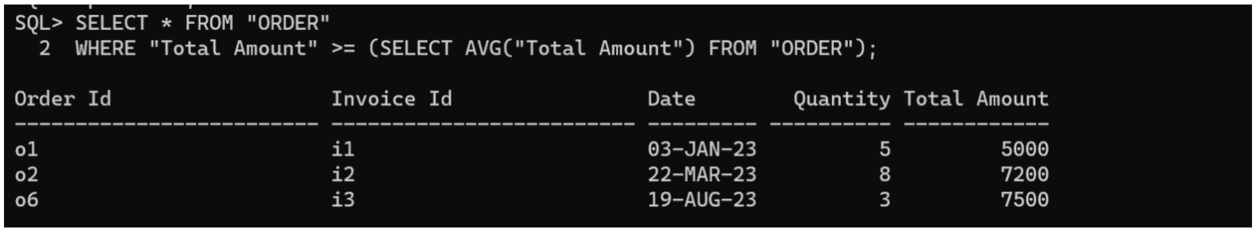
6.2.2 Find those orders that are equal or higher than the average order total value.

Query:

SELECT * FROM "ORDER"

WHERE "Total Amount" >= (SELECT AVG("Total Amount") FROM "ORDER");

This query lists the orders that are equal or higher than average order total value. AVG is used to determine the average value of Total Amount column of the order table.



```
SQL> SELECT * FROM "ORDER"
  2  WHERE "Total Amount" >= (SELECT AVG("Total Amount") FROM "ORDER");

Order Id                 Invoice Id               Date        Quantity Total Amount
------------------------ ------------------------ --------- ---------- ------------
o1                       i1                       03-JAN-23          5         5000
o2                       i2                       22-MAR-23          8         7200
o6                       i3                       19-AUG-23          3         7500
```

Figure 44 : orders that are equal or higher than average order total value

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

Query:

SELECT v."Vendor Id", v."Vendor Name",

COUNT(p."Product Id") AS Supplied_Products

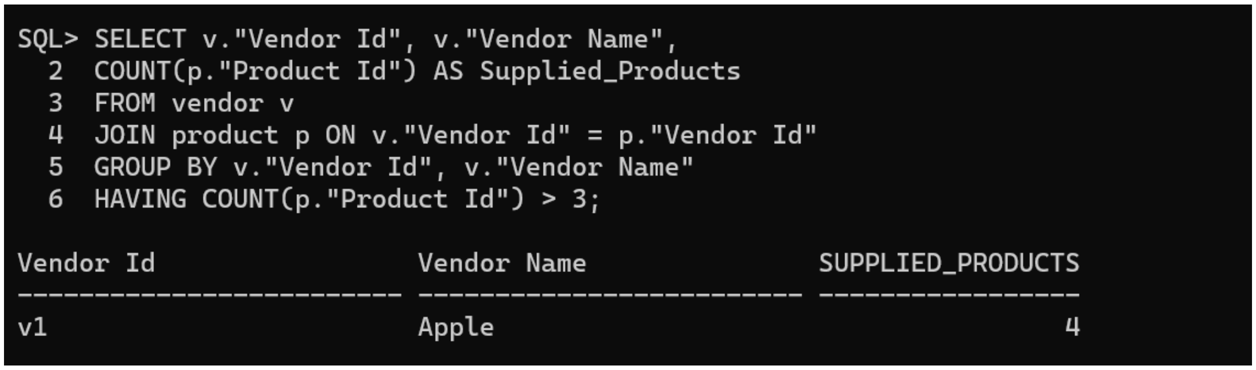
FROM vendor v

JOIN product p ON v."Vendor Id" = p."Vendor Id"

GROUP BY v."Vendor Id", v."Vendor Name"

HAVING COUNT(p."Product Id") > 3;

The query lists the details of vendors who have supplied more than three products to the company. Here, ‘v’ and ‘p’ are the aliases for vendor and product respectively. COUNT is used to calculate number of products supplied by the vendor and GROUP BY groups the result by vendor id and name. Then, HAVING is used filter the result to include the vendors who have supplied more than three products. In this case, the value of supplied products is 4 which includes phones, laptops, watches and earpods.



```
SQL> SELECT v."Vendor Id", v."Vendor Name",
  2  COUNT(p."Product Id") AS Supplied_Products
  3  FROM vendor v
  4  JOIN product p ON v."Vendor Id" = p."Vendor Id"
  5  GROUP BY v."Vendor Id", v."Vendor Name"
  6  HAVING COUNT(p."Product Id") > 3;

Vendor Id                 Vendor Name               SUPPLIED_PRODUCTS
------------------------- ------------------------- -----------------
v1                        Apple                                     4
```

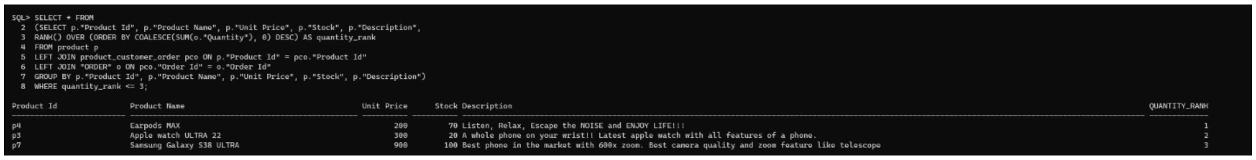
Figure 45 : listing details of vendors who have supplied more than 3 products to the company

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

Query:

```
SELECT * FROM
(SELECT p."Product Id", p."Product Name", p."Unit Price", p."Stock", p."Description",
RANK() OVER (ORDER BY COALESCE(SUM(o."Quantity"), 0) DESC) AS quantity_rank
FROM product p
LEFT JOIN product_customer_order pco ON p."Product Id" = pco."Product Id"
LEFT JOIN "ORDER" o ON pco."Order Id" = o."Order Id"
GROUP BY p."Product Id", p."Product Name", p."Unit Price", p."Stock", p."Description")
WHERE quantity_rank <= 3;
```

This query shows the top three products which have been ordered the most along with their details. Here, ‘p’, ‘o’ and ‘pco’ are the aliases for product, order and product_customer_order tables respectively. RANK and SUM are used to rank the products depending on the sum of their ordered quantities. LEFT JOIN gives the information from the tables and COALESCE is used for such cases where a certain product is not ordered. For example, Dell laptop from products table is not included in any orders so it cannot be included in the quantity of ordered products. GROUP BY groups the results and after the quantity rank’s condition is defined to give the top three results, the required result is displayed where earpods max is ordered most which is followed by Apple watch ULTRA 22 and Samsung Galaxy S38 ULTRA.



```
SQL> SELECT * FROM
(SELECT p."Product Id", p."Product Name", p."Unit Price", p."Stock", p."Description",
RANK() OVER (ORDER BY COALESCE(SUM(o."Quantity"), 0) DESC) AS quantity_rank
FROM product p
LEFT JOIN product_customer_order pco ON p."Product Id" = pco."Product Id"
LEFT JOIN "ORDER" o ON pco."Order Id" = o."Order Id"
GROUP BY p."Product Id", p."Product Name", p."Unit Price", p."Stock", p."Description")
WHERE quantity_rank <= 3;

Product Id           Product Name                              Unit Price      Stock Description                                                                        QUANTITY_RANK
-------------------- ----------------------------------------- ---------- ---------- ---------------------------------------------------------------------------------- -------------
p4                   Earpods MAX                                      200         70 Listen, Relax, Escape the NOISE and ENJOY LIFE!!!                                              1
p3                   Apple watch ULTRA 22                             300         20 A whole phone on your wrist!! Latest apple watch with all features of a phone.                 2
p7                   Samsung Galaxy S38 ULTRA                         900        100 Best phone in the market with 600x zoom. Best camera quality and zoom feature like telescope   3
```

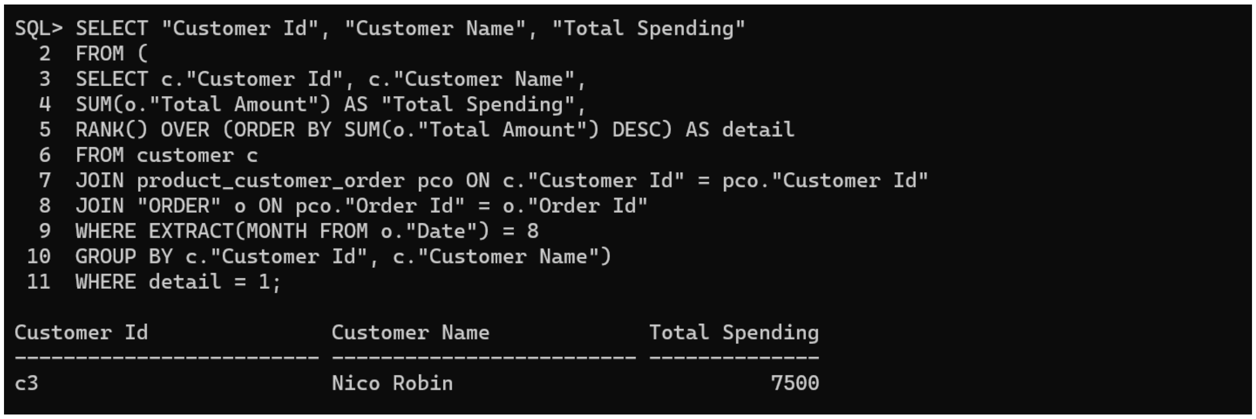
Figure 46 : showing the top three products that have been ordered the most along with their details

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

Query:

```
SELECT "Customer Id", "Customer Name", "Total Spending"

FROM (

SELECT c."Customer Id", c."Customer Name",

SUM(o."Total Amount") AS "Total Spending",

RANK() OVER (ORDER BY SUM(o."Total Amount") DESC) AS detail

FROM customer c

JOIN product_customer_order pco ON c."Customer Id" = pco."Customer Id"

JOIN "ORDER" o ON pco."Order Id" = o."Order Id"

WHERE EXTRACT(MONTH FROM o."Date") = 8

GROUP BY c."Customer Id", c."Customer Name")

WHERE detail = 1;
```

This query gives the customer who has ordered the most in the month of August along with his/her total spending on the August month. Here, ‘c’, ‘o’, and ‘pco’ are the aliases for customer, order and product_customer_order tables respectively. WHERE filters the results obtained according to the condition applied. SUM adds the total amount of order and RANK is used to assign position based on total spending in descending order which is which subquery. Then the result is filtered by outer query which gives the top total spending value. GROUP BY groups the result by customer and ORDER BY sorts the result. JOIN clause connects the tables based on their relations.



```
SQL> SELECT "Customer Id", "Customer Name", "Total Spending"
FROM (
SELECT c."Customer Id", c."Customer Name",
SUM(o."Total Amount") AS "Total Spending",
RANK() OVER (ORDER BY SUM(o."Total Amount") DESC) AS detail
FROM customer c
JOIN product_customer_order pco ON c."Customer Id" = pco."Customer Id"
JOIN "ORDER" o ON pco."Order Id" = o."Order Id"
WHERE EXTRACT(MONTH FROM o."Date") = 8
GROUP BY c."Customer Id", c."Customer Name")
WHERE detail = 1;

Customer Id               Customer Name             Total Spending
------------------------- ------------------------- --------------
c3                        Nico Robin                          7500
```

Figure 47 : listing the customer who has ordered the most in August month with his/her total spending for that month

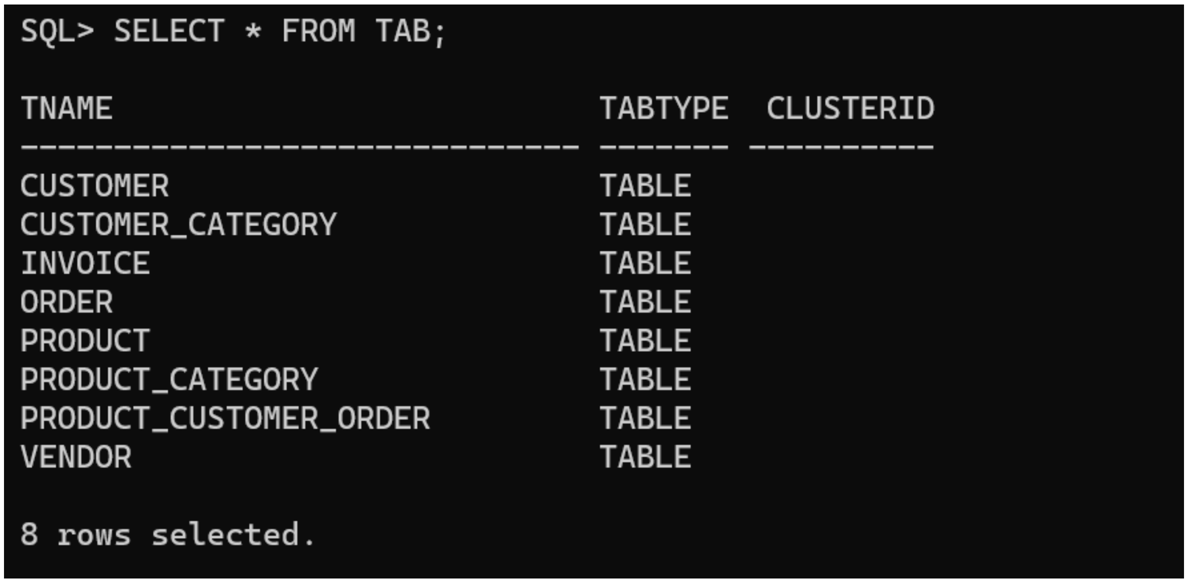
7. CRITICAL EVALUATION

Database is a systematic collection of structured information or data, usually stored digitally on a computer system and managed by a database management system (DBMS). In modern databases, data is typically organized into tables of rows and columns to optimize processing and query efficiency. This arrangement makes it easy to access, control, modify, update, and organize data. Structured Query Language (SQL) is widely used in most databases for tasks such as writing and querying data. (Oracle, 2023) The database module aims in making the students familiar working on the database systems and provides proper materials and course structure to make students knowledgeable and to give proper understanding of the module properly.

The given coursework of Database module (module code: CC5051NI) is the first coursework of the module which accounts for 50% of the overall module grades. The coursework required us to prepare a database design and system for the 'Gadget Emporium' online marketplace to help Mr. John create and implement a strong database system to support his new e-commerce endeavor. For the completion of the coursework, we needed to prepare an Initial Entity Relationship Diagram (ERD), do normalization of it and determine the final ERD based on the scenario provided in the question of the coursework. We also had to make a database system in SQL using proper queries and data to meet the requirement of the coursework. The system is completed and is able to keep track of all customers, products and orders.

8. DROP QUERY AND DUMP FILE CREATION

TABLES LIST:

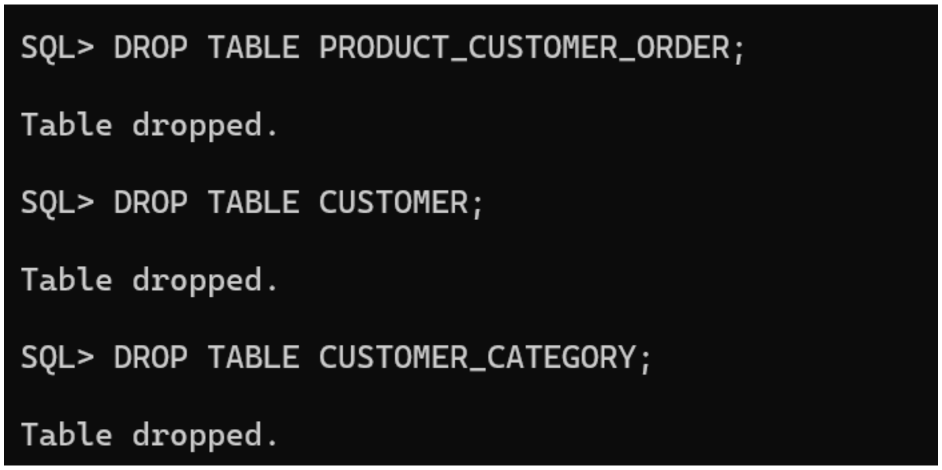


```
SQL> SELECT * FROM TAB;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
CUSTOMER                       TABLE
CUSTOMER_CATEGORY              TABLE
INVOICE                        TABLE
ORDER                          TABLE
PRODUCT                        TABLE
PRODUCT_CATEGORY               TABLE
PRODUCT_CUSTOMER_ORDER         TABLE
VENDOR                         TABLE

8 rows selected.
```

Figure 48 : tables present in the system

DROPPING TABLES:

PRODUCT_CUSTOMER_ORDER, CUSTOMER and CUSTOMER_CATEGORY

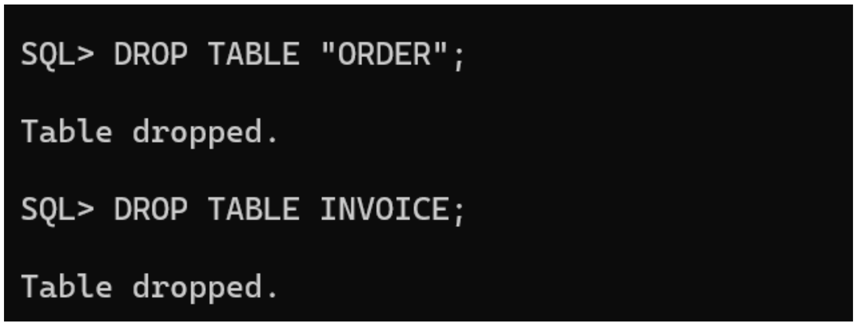


```
SQL> DROP TABLE PRODUCT_CUSTOMER_ORDER;

Table dropped.

SQL> DROP TABLE CUSTOMER;

Table dropped.

SQL> DROP TABLE CUSTOMER_CATEGORY;

Table dropped.
```

Figure 49 : drop product_customer_order, customer and customer_category tables

DROPPING TABLES:

ORDER and INVOICE

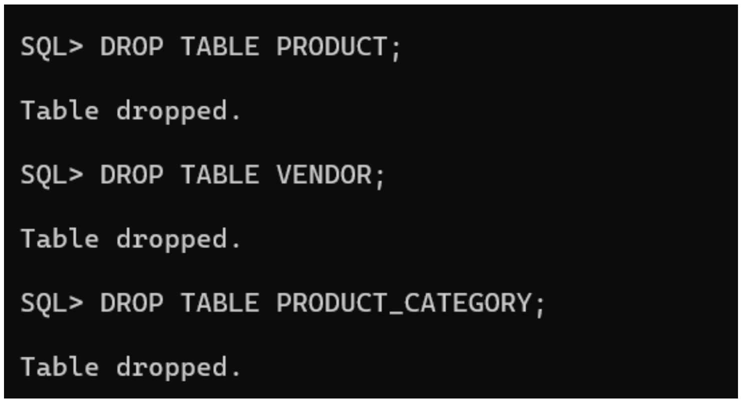


```
SQL> DROP TABLE "ORDER";

Table dropped.

SQL> DROP TABLE INVOICE;

Table dropped.
```

Figure 50 : drop order and invoice tables

DROPPING TABLES:

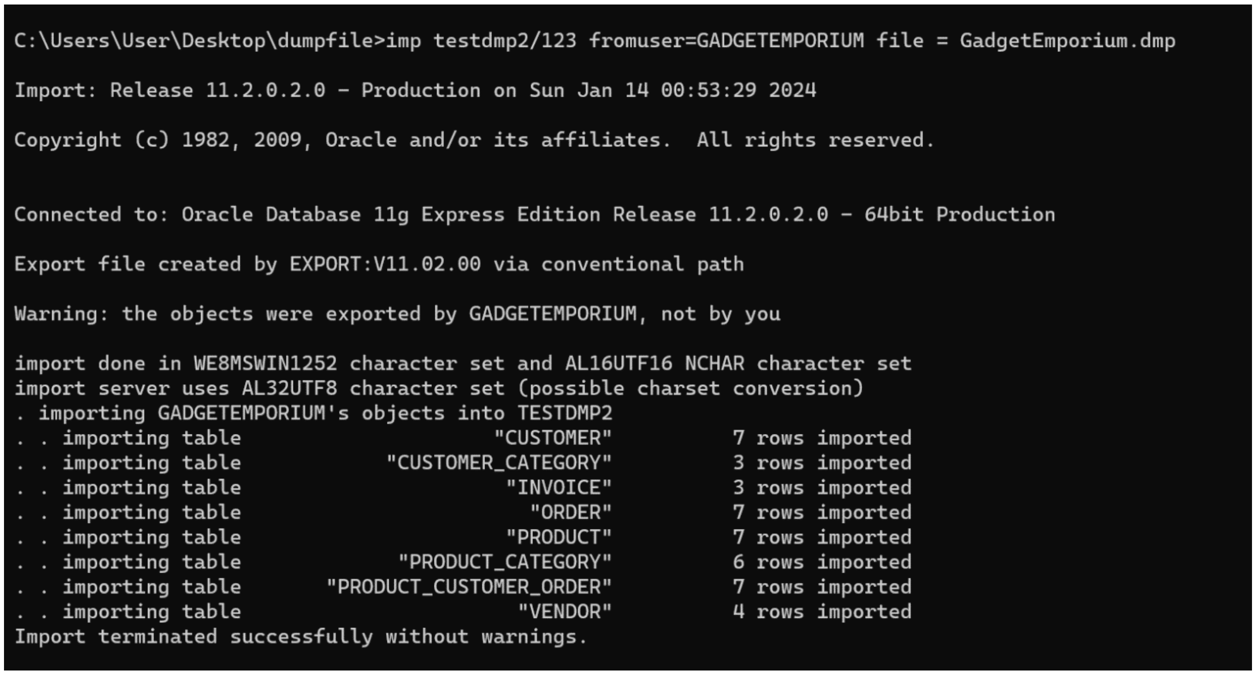
PRODUCT, VENDOR and PRODUCT_CATEGORY



```
SQL> DROP TABLE PRODUCT;

Table dropped.

SQL> DROP TABLE VENDOR;

Table dropped.

SQL> DROP TABLE PRODUCT_CATEGORY;

Table dropped.
```

Figure 51 : drop product, vendor and product_category tables

Below is the screenshot of the dumpfile created and imported:



```
C:\Users\User\Desktop\dumpfile>imp testdmp2/123 fromuser=GADGETEMPORIUM file = GadgetEmporium.dmp

Import: Release 11.2.0.2.0 - Production on Sun Jan 14 00:53:29 2024

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

Export file created by EXPORT:V11.02.00 via conventional path

Warning: the objects were exported by GADGETEMPORIUM, not by you

import done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
import server uses AL32UTF8 character set (possible charset conversion)
. importing GADGETEMPORIUM's objects into TESTDMP2
. . importing table                   "CUSTOMER"          7 rows imported
. . importing table          "CUSTOMER_CATEGORY"          3 rows imported
. . importing table                    "INVOICE"          3 rows imported
. . importing table                      "ORDER"          7 rows imported
. . importing table                    "PRODUCT"          7 rows imported
. . importing table           "PRODUCT_CATEGORY"          6 rows imported
. . importing table     "PRODUCT_CUSTOMER_ORDER"          7 rows imported
. . importing table                     "VENDOR"          4 rows imported
Import terminated successfully without warnings.
```

Figure 52 : dump file

References

Oracle. (2023, 12 25). *Database*. Retrieved from Oracle: https://www.oracle.com/database/what-is-database/#:~:text=Is%20a%20Database%3F-,Database%20defined,database%20management%20system%20(DBMS).