LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

Autumn 2024

Student Name: Deepshikha Sharma

London Met ID: 22085522

Assignment Submission Date: 15th January, 2024

Word Count: 5750

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

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Gadget Emporium is a massive corporation that focuses on electronic gadgets and accessories such as smartphones, TVs, computers, and smart watches. All of the products sold by this well-known company are imported from the US, the UK, China, India, and several other nations. The company is well known for offering premium products together with services and guarantees. The company sells products at retail from several device manufacturers, such as LG, Samsung, Xiaomi, Dell, Acer, and Lenovo. Mr. John is the CEO of this firm. Even though he started Gadget Emporium during the pandemic, a few years after its founding, it achieved remarkable success.

Now, Mr. John plans to open an online business with the goal of delivering the same services right to customers' doorsteps. His goal is to offer a wide range of electronic products for online purchasing to both private individuals and corporate entities. Mr. John wants to turn Gadget Emporium into a business that gives customers the ability to purchase a wide range of gadgets at their fingertips since the world is changing and we can see that services that consumers used to receive by visiting a physical store are now available right at their doorsteps. Although he had this goal from the start, he lacked the funds, ideas, and personnel to carry out such a significant move. However, he chose to confront other companies in the same industry and steer his own firm in that direction after seeing the potential of internet enterprises during the epidemic.

1.1. Current business operation

Gadget Emporium is currently working in supplying electronic gadgets and devices to a wide range of population. It is a well-known gadget firm that provides products with quality and god price range. The products are ordered from Japan, USA, China, India and South Korea. Its physical shop is located in Lalitpur, Nepal. The customers go to their shop and can choose from a wide range of products to buy from. They also get loyalty points per purchase which is a good thing. The owner of the business have hired around 10 staffs working in the shop to facilitate the customers. The business is not yet online and is running purely because of the customers who go back to the shop to buy again.

The customers are satisfied with their purchase and want an online shopping website to ease their purchase process.

1.2. Current Business Rules

Product Management: The system stores all the details of product. Each product must be of only one category. Each category can have one or many products

Customer Categories and Discounts: The customers are divided into staff, regular and VIP. Each category has different discount rates.

Order Processing: Customers can order one or more products. The system stores all the order details.

Management System of Vendor: The vendor supplies products. The vendor can supply one or more products. One product should be associated with single vendor.

Availability of Products: The products are stored in the inventory. The inventory can have one or more products.

Payment process: The customers can pay their bills using any method. Each order has one payment method.

Invoice: An invoice is created after transaction which contains details of order, product and customer

1.3. Assumptions

- Each order detail must have one payment option
- The customers are provided discounts on the basis of their category
- The availability status of the product depends on the inventory

1.4. Identification of entities and attributes

1.4.1. Entities

When referring to a real-world item in a database management system (DBMS), an entity is one that possesses specific features that characterize its nature. Entities are distinct, meaning that each of the two entities in a pair possesses a characteristic that sets them apart from the other. A white piece and a black piece, for instance, may be distinguished from one another on a chessboard because of their different colors. The traits that characterize an entity's distinctive qualities or features make them up (Arya, 2022).

1.4.2. Attributes

An attribute in a database management system (DBMS) can be used to characterize the features or attributes of an item or component, such a field or database table. When comparing a spreadsheet to a database, the attribute is only one -- non-null -- cell or the intersection of a certain column and row (Awati, 2022).

The entities and attributes used in this coursework are:

Entities	**Attributes**
Customer	**Customer_ID(PK),** Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone
Orders	**Order_ID(PK),** Order_date, Total_order_amount, Products_purchased,_quantity, Unit_price, Payment_number, Payment_type, Invoice_number, Invoice_date
Product	**Product_ID(PK),**Product_name, Product_description, Product_category_name, Product_category_number,

	Product_price, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount, Stock_qty

Table 1 Entities and attributes table

1.5. Relationship between entities

The relationship between Entities and the relation between them are:

Entities	**Relation**
Customer to Orders	One to Many
Orders to Product	Many to Many

Table 2 Entities and relation table

1.6. Identification of Primary Keys and Foreign Keys

Tables	**Primary Key**	**Foreign Key**
Customer	**Customer_ID**	-
Orders	**Order_ID**	**Customer_ID**
Product	**Product_ID**	**Customer_ID, Order_ID**

1.7. Identification of Constraints

1.7.1. For Customer

The attributes, their datatype, constraint and explanation of Customer table are as follows.

Attribute	**Data Type**	**Constraint**	**Explanation**
Customer_ID	VARCHAR(50)	PRIMARY KEY	It stores IDs of customer
Customer_category_number	VARCHAR(50)	NOT NULL UNIQUE	It stores category number of the customer

Customer_category_name	VARCHAR(50)	NOT NULL	It stores category name of the customer
Customer_name	VARCHAR(50)	NOT NULL	It stores name to the customers
Customer_discount	DECIMAL(5,2)	NOT NULL	It stores discount percentage of customers
Customer_address	VARCHAR(50)	NOT NULL	It stores address of the customers
Customer_phone	INT, UNIQUE	NOT NULL, UNIQUE	It stores the phone number of the customer

Table 3 Identification and explanation of constraints of Customer table

In the above table, all the values in the customer table are not null which means empty value cannot be inserted into it. The customer_category_number and customer_phone is unique.

1.7.2. For Orders

The attributes, their datatype, constraint and explanation of Orders table are as follows.

Attribute	**Data Type**	**Constraint**	**Explanation**
Order_ID	VARCHAR(50)	PRIMARY KEY	It stores IDs of orders made my customer
Order_date	DATE	NOT NULL	It stores date on which the order was made
Total_order_amount	INT	NOT NULL	It stores the total amount after order
Products_purchased,_quantity	INT	NOT NULL	It stores data of total products

			purchased during the order
Unit_price	DECIMAL(5,2)	NOT NULL	It stores unit price of the product
Payment_number	VARCHAR(50)	NOT NULL UNIQUE	It stores payment ID
Payment_type	VARCHAR(50)	NOT NULL	It stores the type of payment
Invoice_number	VARCHAR(50)	NOT NULL, UNIQUE	It stores the invoice ID
Invoice_date	DATE	NOT NULL	It stores the date of the invoice

Table 4 Identification and explanation of constraints of Orders table

In the above table, all the values in the Orders table are not null which means empty value cannot be inserted into it. The payment_number and invoice_number are unique.

1.7.3. For Product

The attributes, their datatype, constraint and explanation of Product table are as follows.

Attribute	Data Type	Constraint	Explanation
Product_ID	VARCHAR(50)	PRIMARY KEY	It stores IDs of the products
Product_name	VARCHAR(50)	NOT NULL	It stores name of the products
Product_description	VARCHAR(50)	NOT NULL	It stores the description of the product
Product_category_name	VARCHAR(50)	NOT NULL	It stores the name of category of the product

Product_category_number	VARCHAR(50)	NOT NULL UNIQUE	It stores the category number of the product
Stock_qty	INT	NOT NULL	It stores the quantity of stock of the product
Product_price	INT	NOT NULL	It stores price of the product
Inventory_number	VARCHAR(50)	NOT NULL UNIQUE	It stores the inventory number of the product
Inventory_name	VARCHAR(50)	NOT NULL	It stores the inventory name of the product
Availability_status	VARCHAR(50)	NOT NULL	It stores the availability status of weather the product is available or not of the product
Vendor_number	VARCHAR(50)	NOT NULL, UNIQUE	It stores the vendor ID
Vendor_name	VARCHAR(50)	NOT NULL	It stores the vendor’s name
Vendor_address	VARCHAR(50)	NOT NULL	It stores the vendor’s address
Vendor_supply_amount	INT	NOT NULL	It stores the vendor’s supply amount of product

Table 5 Identification and explanation of constraints of Product table

In the above table, all the values in the Product table are not null which means empty value cannot be inserted into it. The vendor_number product_category_number and inventory_number are unique.

2. Initial ERD

The Initial ERD below shows the entities their attributes and the relationship between the entities. This is the un-normalized form of ERD which shows data redundancy. This can be resolved with the help of normalization.

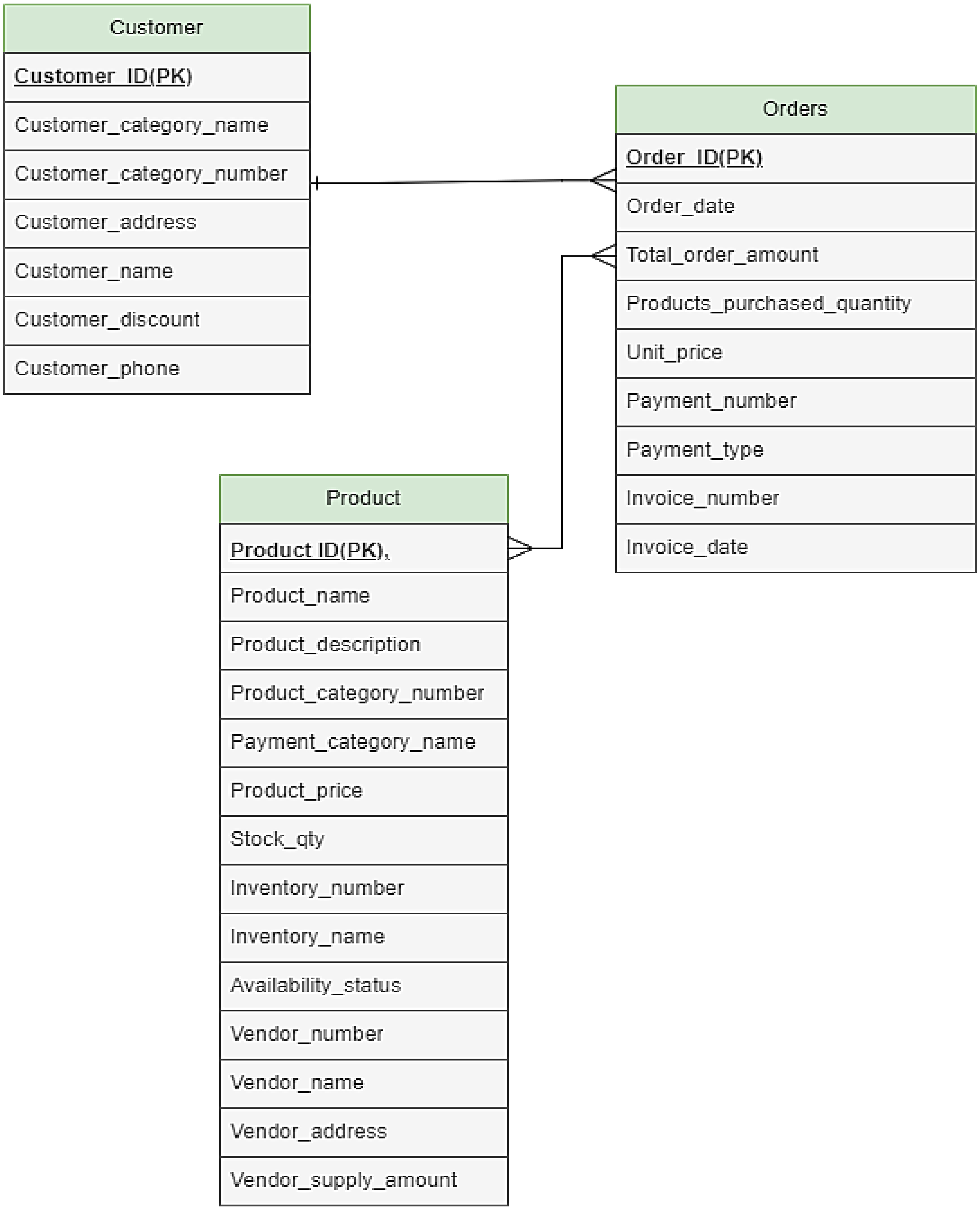


Figure 1 Initial ERD

3. Normalization

The procedure known as "normalization" is used to improve data integrity and remove redundant data from tables. Data organization in the database is further aided by normalization. The procedure involves many steps to convert the data into tabular format and eliminate redundant information from relational tables. In order to guarantee that database integrity constraints carry out their requirements correctly, normalization arranges a database's columns and tables. It is an organized method of breaking down tables to get rid of repetitive data and unwanted features like Insertion, Update, and Deletion anomalies (S, 2022).

3.1. UNF (Un-normalized Form)

Un-normalized form (UNF), often referred to as un-normalized relation or non-first normalized form (N1NF or NF2), is a database data model (the way that data is organized in a database) that does not satisfy any of the relational model's specified database normalization requirements. Non-relational, or NoSQL, databases are database systems that allow de-normalized data. Non-normalized connections can be viewed as the initial state of the normalization process in relational models. This is not the same as de-normalization. In relational databases, de-normalization purposefully compromises normalization for some tables (Anon., 2023).

The UNF form for this coursework is:

(**Customer_ID(PK),** Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone **{ Order_ID(PK),** Order_date, Total_order_amount, Products_purchased,_quantity, Unit_price, Payment_number, Payment_type, Invoice_number, Invoice_date**{ Product_ID(PK),** Product_name, Product_description, Product_category, Product_category_number, Product_price, Stock_qty, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount **}})**

3.2. 1NF (First Normal Form)

It is a DBMS normalization level. When a relation has an atomic value, it is said to be in 1 normal form (also known as 1NF) in DBMS. To put it another way, 1NF specifies that an attribute of a table may only have one value and cannot carry multiple values. A relation would be in violation of the First Network Framework (1NF) if it had any multi-valued or composite attributes, as the 1NF forbids the use of any of these attributes alone or in combination (Anon., 2022).

Tables after 1NF

Customer-1(**Customer_ID(PK),** Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone)

Orders-1(**Order_ID(PK), Customer_ID*(FK)**, Order_date, Total_order_amount, Products_purchased,_quantity, Unit_price, Payment_number, Payment_type, Invoice_number, Invoice_date)

Product-1(**Product_ID(PK), Order_ID*(FK), Customer_ID*(FK)** Product_name, Product_description, Product_category, Product_category_number, Product_price, Stock_qty, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount)

Explanation:

To transform UNF into 1NF, the repeating data and repeating groups are kept in the separate tables. Here, the repeating data is customer and repeating groups are orders and customer. Each table consists a Primary Key and the Order and Product table consists of Foreign Keys. The tables are related to each other with the help of Foreign Keys.

3.3. 2NF (Second Normal Form)

Subgroups of data that appear in several rows of tables must be eliminated in order to comply with the second normal form, and they must be represented in a new table with links established between them. Basically, distinct tables should include all subsets of data that are possible to exist in multiple rows. Relationships between the newly generated tables (the rearranged subgroups of the data) and new key labels may then be established (Morris, 2022).

A table is said to be in 2NF when:

- The table is in 1NF
- When there is no partial dependency

- **For Customer**

Assumption:

- **Customer_ID(PK)**→ Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone

The table is:

Customer-2(Customer_ID(PK), Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone)

Explanation:

The Customer_ID gives Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone. There is no partial dependency in this table and it is already in 1NF.

- **For Order**

  Assumptions:

  - **Order_ID(PK)**→Order_date, Total_order_amount, Products_purchased,_quantity, Unit_price, Payment_number, Payment_type, Invoice_number, Invoice_date
  - **Customer_ID*(FK)**→ **X**
  - **Order_ID*(FK), Customer_ID*(FK)** → **X**

  The tables are:

  Orders-2 (**Order_ID(PK),** Order_date, Total_order_amount, Products_purchased,_quantity, Unit_price, Payment_number, Payment_type, Invoice_number, Invoice_date)

  Order-Customer-2(Order_ID*(FK), Customer_ID*(FK))

  Explanation:

  The Order_ID gives Order_date, Total_order_amount, Products_purchased,_quantity, Unit_price, Payment_number, Payment_type, Invoice_number, Invoice_date but there is Customer_ID as foreign key in that table. Since Order_ID is the reference from Orders table, a bridge entity is created to avoid any kind of anomalies and partial dependencies.

- **For Product**

  Assumptions:

  - **Product_ID(PK)**→ Product_name, Product_description, Product_category, Product_category_number, Product_price, Stock_qty, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount
  - **Customer_ID** → **X**
  - **Order_ID** → **X**
  - **Order_ID, Product_ID** → Unit_price
  - **Order_ID, Customer_ID** → **X**

- **Product_ID, Customer_ID → X**
- **Order ID, Product ID, Vendor ID → X**

The tables are:

Product-2(**Product_ID(PK), Order_ID*(FK), Customer_ID*(FK)** Product_name, Product_description, Product_category, Product_category_number, Product_price, Stock_qty, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount)

Order-Product-2(Order_ID*(FK), Product_ID*(FK),Unit_price)

Product-Customer-2(**Product_ID*(FK), Customer_ID*(FK))**
Order-Product-Customer-2(Order_ID*(FK), Product_ID*(FK), Customer_ID*(FK))

Explanation:

The Product_ID gives Product_name, Product_description, Product_category, Product_category_number, Product_price, Stock_qty, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount but there is Customer_ID and Order_ID as foreign keys in that table. Since Order_ID and Customer_ID is the reference from Orders and Customer table, bridge entities are created to avoid any kind of anomalies and partial dependencies. Also Unit_price depends on both Customer_ID and Order_ID which creates anomaly. So, a bridge table is created and Unit_price is kept there along with Customer_ID and Order_ID.

Final Tables after 2NF

The following are the final tables after 2NF:

Customer-2(Customer_ID(PK), Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone)

Orders-2 (**Order_ID(PK),** Order_date, Total_order_amount, Products_purchased,_quantity, Unit_price, Payment_number, Payment_type, Invoice_number, Invoice_date)

Order-Customer-2(Order_ID*(FK), Customer_ID*(FK))

Product-2(Product_ID(PK), Order_ID*(FK), Customer_ID*(FK) Product_name, Product_description, Product_category, Product_category_number, Product_price, Stock_qty, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount)

Order-Product-2(Order_ID*(FK), Product_ID*(FK), Unit_price)

Product-Customer-2(Product_ID*(FK), Customer_ID*(FK))

Order-Product-Customer-2(Order_ID*(FK), Product_ID*(FK), Customer_ID*(FK)

3.4. 3NF (Third Normal Form)

When a relation is in 2NF but lacks transitive partial dependence, it is said to be in its third normal form. A relation is said to be in 3NF if there is no transitive dependency for the non-prime characteristics. In order to prevent data duplication and maintain data integrity in databases, we employ the 3NF. It is appropriate to create normal relational databases using the third normal form. The reason for this is because most 3NF tables are not affected by insertion, update, or deletion abnormalities (Karthik, 2023).

A table is said to be in 3NF when:

- The table is in 2NF
- When there are no transitive dependencies

In Customer-2

Customer-2(**Customer_ID(PK),** Customer_category_number, Customer_category_name, Customer_name, Customer_discount, Customer_address, Customer_phone)

Explanation:

Here,

Customer_ID(PK)→Customer_name,Customer_address,Customer_category_number, Customer_category_name, Customer_phone

But,

Customer_category_number→Customer_category_name, Customer_discount

Which means there is transitive dependency as

A → B, B→C and A→ C

To avoid this, we have to separate the tables. After separation of the tables, the primary key of the new table becomes foreign table in the previous table.

Customer-3(**Customer_ID(PK), Customer_category_number*(FK),** Customer_name, Customer_address, Customer_phone)

Customer-Category-3(**Customer_category_number(PK),** Customer_category_name, Customer_discount)

- **In Order-2**
  Order-2 (**Order_ID(PK),** Order_date, Total_order_amount, Products_purchased,_quantity, Payment_number, Payment_type, Invoice_number, Invoice_date)

  Here,

Order_ID(PK)→Order_date, Total_order_amount, Products_purchased,_quantity, Payment_number, Payment_type, Invoice_number, Invoice_date

But,

Payment_number → Payment_type

Invoice_number → Invoice_date

Which means there is transitive dependency as

A → B, B→C and A→ C

To avoid this, we have to separate the tables. After separation of the tables, the primary key of the new table becomes foreign table in the previous table.

Order-3 (**Order_ID(PK),** Order_date, Total_order_amount, Products_purchased,_quantity, **Payment_number*(FK)**, **Invoice_number*(FK)**)

Payment-3(**Payment_number(PK),** Payment_type)

Invoice-3(**Invoice_number(PK),** Invoice_date)

- **In Product-2**

  Product-2 (**Product_ID(PK), Order_ID*(FK), Customer_ID*(FK)** Product_name, Product_description, Product_category, Product_category_number, Product_price, Stock_qty, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount)

  Here,

  Product_ID(PK)→ Product_name, Product_description, Product_category, Product_category_number, Product_price, Product_stock_quantity, Inventory_number, Inventory_name, Availability_status, Vendor_number, Vendor_name, Vendor_address, Vendor_supply_amount)

But,

Product_category_number → Product_category

Inventory_number → Inventory_name,Availability_status, Stock_qty

Vendor_number → Vendor_name, Vendor_address, Vendor_supply_amount

Which means there is transitive dependency as

A → B, B→C and A→ C

To avoid this, we have to separate the tables. After separation of the tables, the primary key of the new table becomes foreign table in the previous table.

Product-3(**Product_ID(PK),** Product_name, Product_description, Product_price **Product_category_number*(FK), Inventory_number*(FK), Vendor_number*(FK))**

Product-Category-3(**Product_category_number(PK),** Product_category) **Inventory-3**(**Inventory_number(PK),**Inventory_name,Availability_status, Stock_qty)

Vendor-3(Vendor_number(PK),Vendor_name, Vendor_address, Vendor_supply_amount**)**

Final Tables after 3NF

Customer-3(**Customer_ID(PK), Customer_category_number*(FK),** Customer_name, Customer_address, Customer_phone)

Customer-Category-3(**Customer_category_number(PK),** Customer_category_name, Customer_discount)

Order-3 (**Order_ID(PK), Payment_number*(FK), Invoice_number*(FK),** Order_date, Total_order_amount, Products_purchased_quantity)

Payment-3(**Payment_number(PK),** Payment_type)

Invoice-3(**Invoice_number(PK),** Invoice_date)

Product-3(**Product_ID(PK), Product_category_number*(FK), Inventory_number*(FK), Vendor_number*(FK)** Product_name, Product_description, Product_price)

Product-Category-3(**Product_category_number(PK),** Product_category)

Inventory-3(**Inventory_number(PK),**Inventory_name, Availability_status, Stock_qty)

Vendor-3(Vendor_number(PK),Vendor_name, Vendor_address, Vendor_supply_amount**)**

Order-Customer-3(Order_ID*(FK), Customer_ID*(FK))

Order-Product-3(Order_ID*(FK), Product_ID*(FK), Unit_price)

Product-Customer-3(**Product_ID*(FK), Customer_ID*(FK))**

Order-Product-Customer-3(Order_ID*(FK), Product_ID*(FK), Customer_ID*(FK))

4. Final ERD

The below diagram shows the final entity relation diagram after normalization. After normalization the bridge entities are created, the partial and transitive are removed and the foreign keys are used to reference the tables.

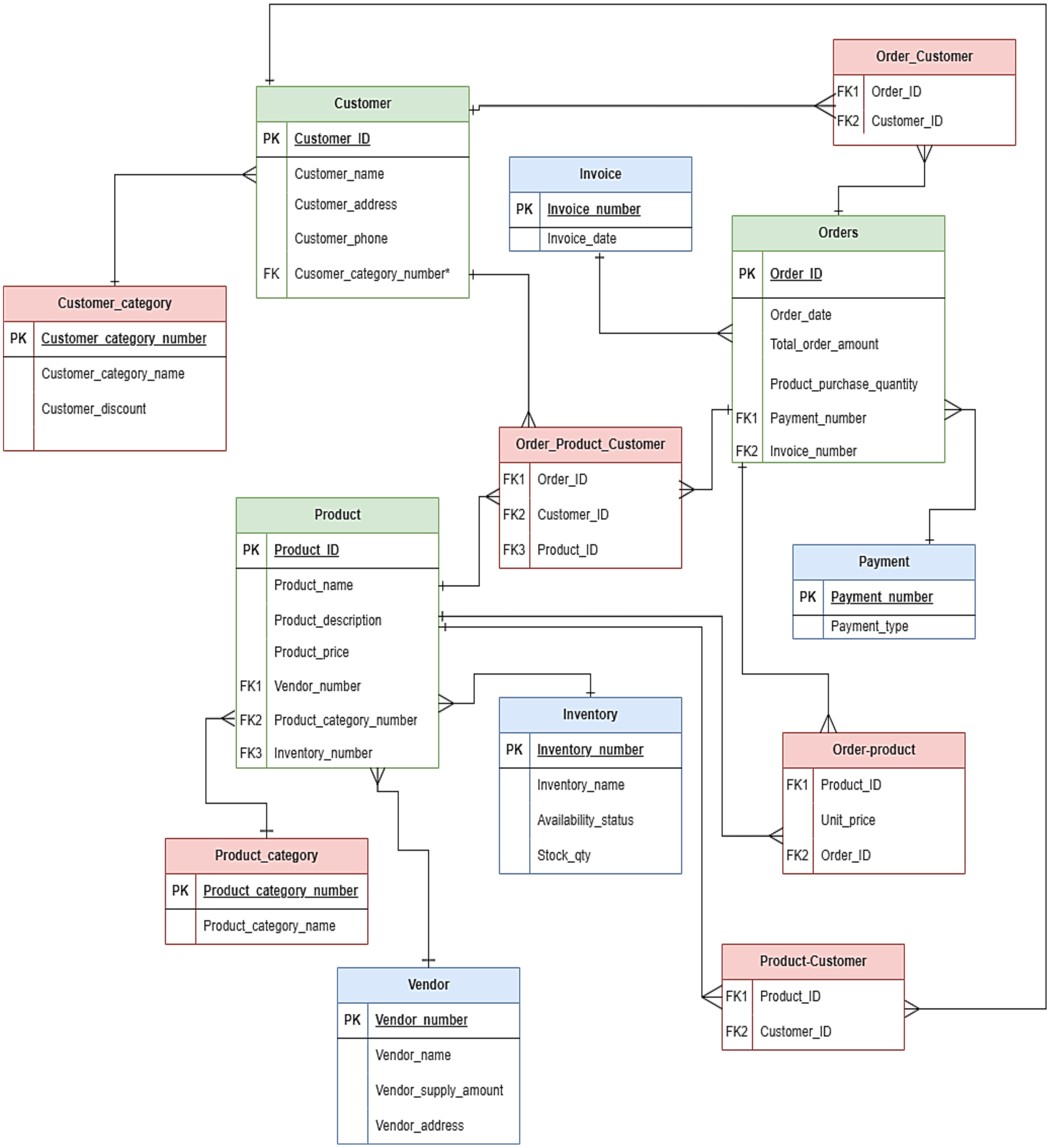


Figure 2 Final ERD

5. Implementation

After normalization, the data redundancies are removed. Now the tables after 3NF are created in SQL, the values are inserted into it and some queries are also carried out.

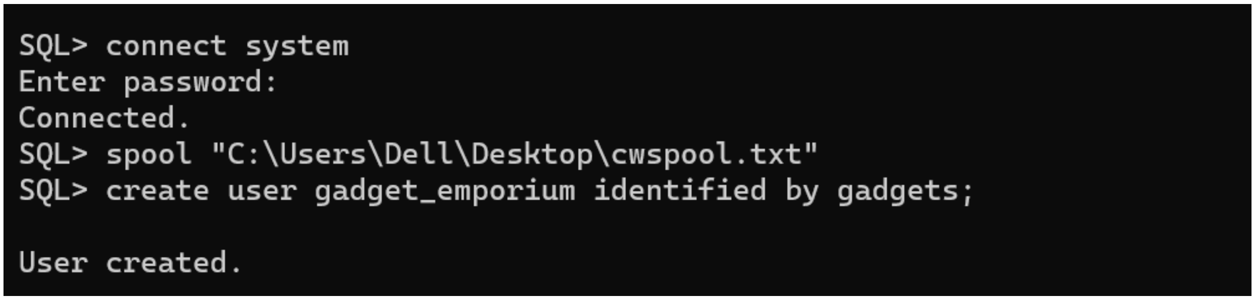
5.1. Creating and describing the tables

- **Connecting to system and creating a spool file**

Syntax:

spool "C:\Users\Dell\Desktop\cwspool.txt"

create user gadget_emporium identified by gadgets;



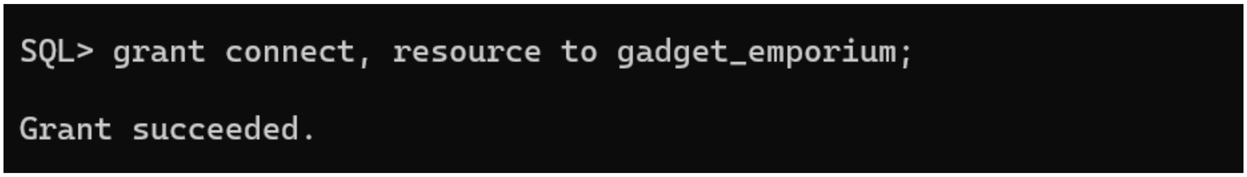
```
SQL> connect system
Enter password:
Connected.
SQL> spool "C:\Users\Dell\Desktop\cwspool.txt"
SQL> create user gadget_emporium identified by gadgets;

User created.
```

Figure 3 Connecting system and creating spool file

- **Granting access to resources to gadget_emporium**

Syntax

grant connect, resource to gadget_emporium;



```
SQL> grant connect, resource to gadget_emporium;

Grant succeeded.
```

Figure 4 Granting access

- **Creating table Product_category**

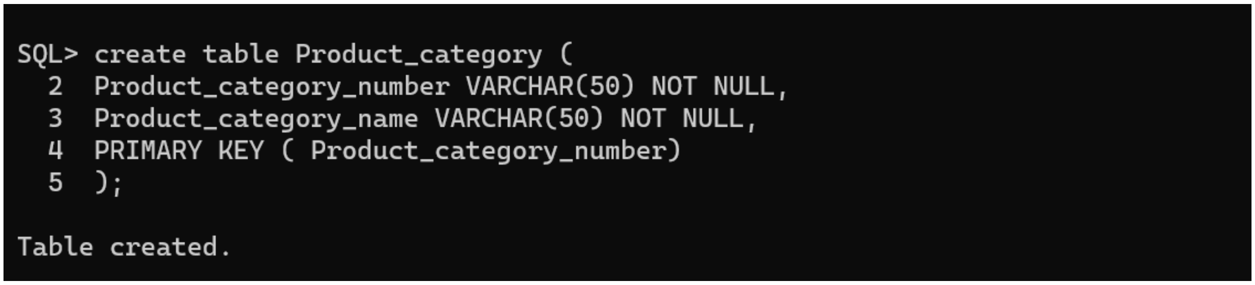
Syntax

create table Product_category (

Product_category_number VARCHAR(50) NOT NULL, Product_category_name VARCHAR(50) NOT NULL,

PRIMARY KEY (Product_category_number)

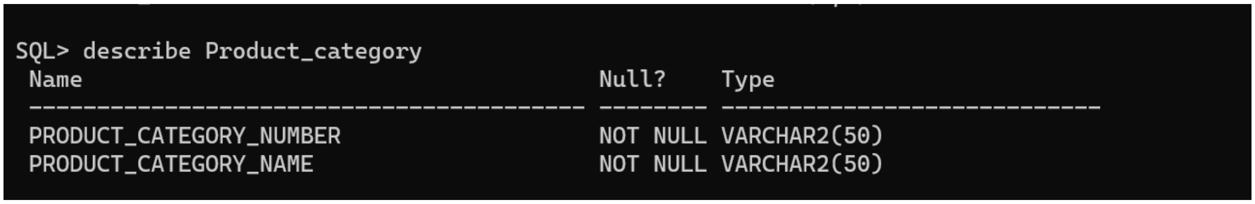
);



```
SQL> create table Product_category (
     Product_category_number VARCHAR(50) NOT NULL,
     Product_category_name VARCHAR(50) NOT NULL,
     PRIMARY KEY ( Product_category_number)
     );

Table created.
```

Figure 5 Creating Product_category table

- **Describing the table**



```
SQL> describe Product_category
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCT_CATEGORY_NUMBER                   NOT NULL VARCHAR2(50)
 PRODUCT_CATEGORY_NAME                     NOT NULL VARCHAR2(50)
```

Figure 6 Describing Product_category table

Table Description:

The table stores product category number and product category names of products and has one to many relationship with product table.

- **Creating and describing Customer_category table**

  <u>Syntax</u>

  create table Customer_category (

  Customer_category_number VARCHAR(50) NOT NULL,

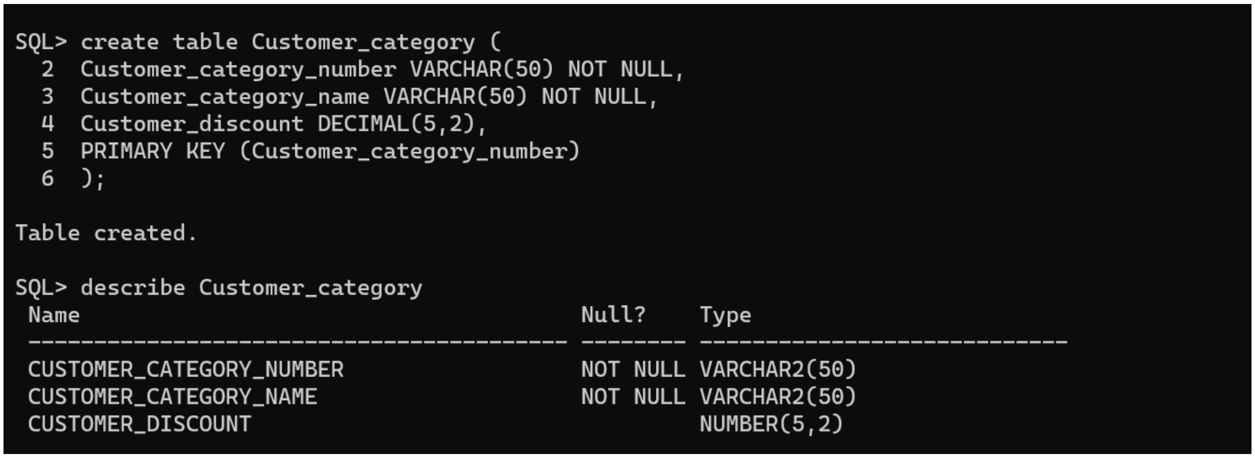
  Customer_category_name VARCHAR(50) NOT NULL,

  Customer_discount DECIMAL(5,2),

  PRIMARY KEY (Customer_category_number)

);

describe Customer_category



```
SQL> create table Customer_category (
  2  Customer_category_number VARCHAR(50) NOT NULL,
  3  Customer_category_name VARCHAR(50) NOT NULL,
  4  Customer_discount DECIMAL(5,2),
  5  PRIMARY KEY (Customer_category_number)
  6  );

Table created.

SQL> describe Customer_category
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_CATEGORY_NUMBER                  NOT NULL VARCHAR2(50)
 CUSTOMER_CATEGORY_NAME                    NOT NULL VARCHAR2(50)
 CUSTOMER_DISCOUNT                                  NUMBER(5,2)
```

Figure 7 Creating and describing Customer_category table

Table Description:

This table stores information such as customer category number, customer category name and customer discount. The customer category number, customer category name are not null. It has one to many relationship with customer table

- **Creating and describing Inventory table**

  <u>Syntax</u>

  create table Inventory (

  Inventory_number VARCHAR(50) NOT NULL,

  Inventory_name VARCHAR(50) NOT NULL,

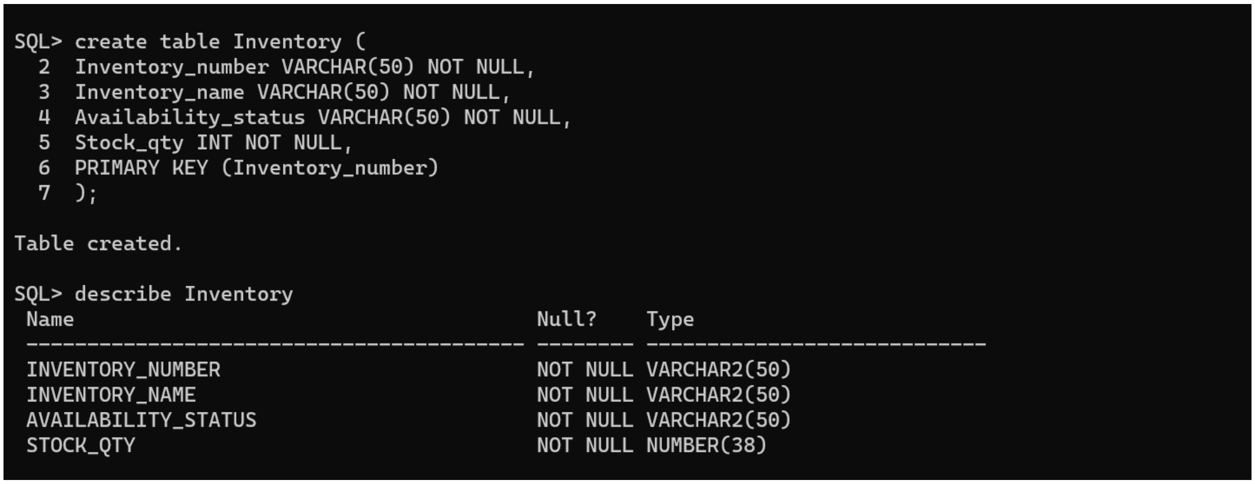
  Availability_status VARCHAR(50) NOT NULL,

  Stock_qty INT NOT NULL,

  PRIMARY KEY (Inventory_number)

  );

  describe Inventory



```
SQL> create table Inventory (
Inventory_number VARCHAR(50) NOT NULL,
Inventory_name VARCHAR(50) NOT NULL,
Availability_status VARCHAR(50) NOT NULL,
Stock_qty INT NOT NULL,
PRIMARY KEY (Inventory_number)
);

Table created.

SQL> describe Inventory
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INVENTORY_NUMBER                          NOT NULL VARCHAR2(50)
 INVENTORY_NAME                            NOT NULL VARCHAR2(50)
 AVAILABILITY_STATUS                       NOT NULL VARCHAR2(50)
 STOCK_QTY                                 NOT NULL NUMBER(38)
```

Figure 8 Creating and describing Inventory table

Table Description:

This table stores the values such as Inventory number, inventory name, availability status and stock quantity. All the values in it are not null. It has one to many relationship with product table.

- **Creating and describing Payment table**
  <u>Syntax</u>

  create table Payment (

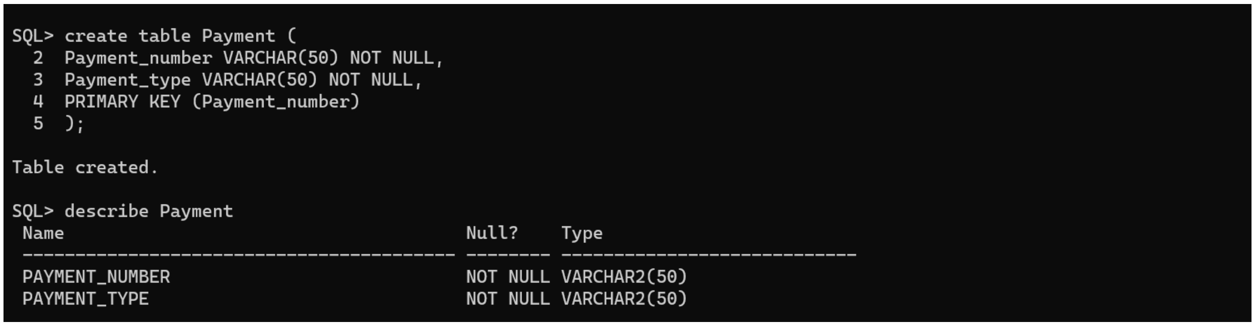
  Payment_number VARCHAR(50) NOT NULL,

  Payment_type VARCHAR(50) NOT NULL,

  PRIMARY KEY (Payment_number)

  );

  describe Payment



```
SQL> create table Payment (
Payment_number VARCHAR(50) NOT NULL,
Payment_type VARCHAR(50) NOT NULL,
PRIMARY KEY (Payment_number)
);

Table created.

SQL> describe Payment
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PAYMENT_NUMBER                            NOT NULL VARCHAR2(50)
 PAYMENT_TYPE                              NOT NULL VARCHAR2(50)
```

Figure 9 Creating and describing Payment table

Table Description:

This table stores payment number, payment type of payment. The values are not null. It has one to many relationship with Order table.

- **Creating and describing Invoice table**

Syntax

create table Invoice (

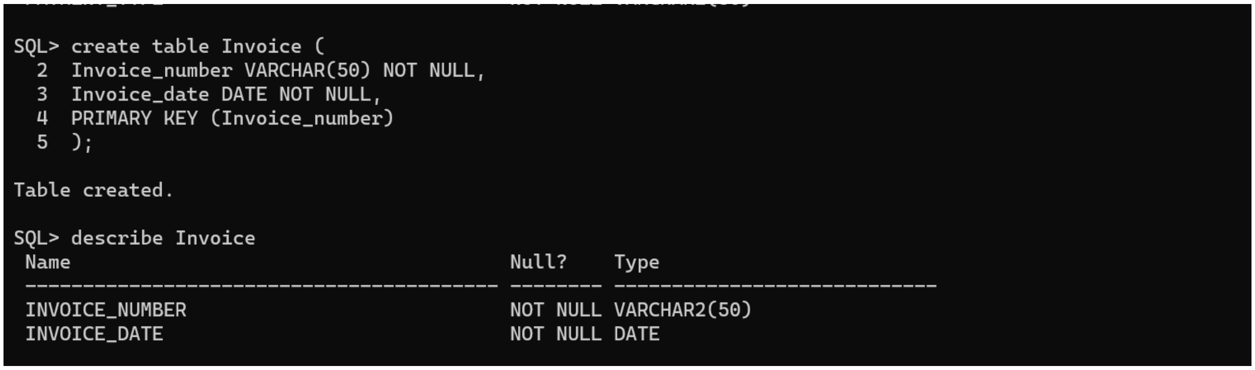
Invoice_number VARCHAR(50) NOT NULL,

Invoice_date DATE NOT NULL,

PRIMARY KEY (Invoice_number)

);

describe Invoice



```
SQL> create table Invoice (
  2  Invoice_number VARCHAR(50) NOT NULL,
  3  Invoice_date DATE NOT NULL,
  4  PRIMARY KEY (Invoice_number)
  5  );

Table created.

SQL> describe Invoice
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INVOICE_NUMBER                            NOT NULL VARCHAR2(50)
 INVOICE_DATE                              NOT NULL DATE
```

Figure 10 Creating and describing Invoice table

Table Description:

This table stores Invoice number and invoice date. The values are not null. It has one to many relationship with Order table.

- **Creating and describing Vendor table**

Syntax

create table Vendor (

Vendor_number VARCHAR(50) NOT NULL,

Vendor_name VARCHAR(50) NOT NULL,

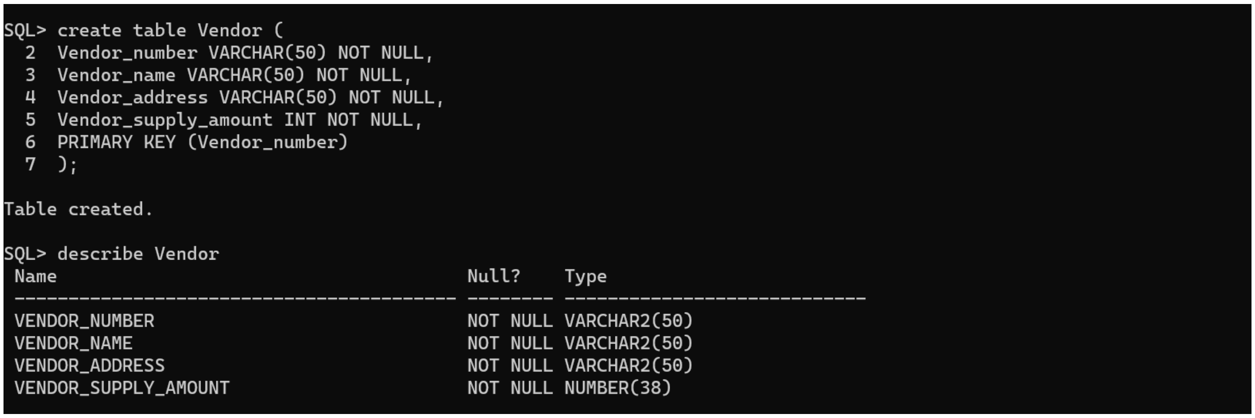
Vendor_address VARCHAR(50) NOT NULL,

Vendor_supply_amount INT NOT NULL,

PRIMARY KEY (Vendor_number)

);

describe Vendor



```
SQL> create table Vendor (
  2  Vendor_number VARCHAR(50) NOT NULL,
  3  Vendor_name VARCHAR(50) NOT NULL,
  4  Vendor_address VARCHAR(50) NOT NULL,
  5  Vendor_supply_amount INT NOT NULL,
  6  PRIMARY KEY (Vendor_number)
  7  );

Table created.

SQL> describe Vendor
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VENDOR_NUMBER                             NOT NULL VARCHAR2(50)
 VENDOR_NAME                               NOT NULL VARCHAR2(50)
 VENDOR_ADDRESS                            NOT NULL VARCHAR2(50)
 VENDOR_SUPPLY_AMOUNT                      NOT NULL NUMBER(38)
```

Figure 11 Creating and describing Vendor table

Table description:

Vendor table stores the values like vendor number vendor name, vendor address and vendor supply amount. The values are not null and the table has one to many relationship with product.

- **Creating and describing Product table**
  Syntax

create table Product (

Product_ID VARCHAR(50) NOT NULL,

Product_category_number VARCHAR(50) NOT NULL,

Inventory_number VARCHAR(50) NOT NULL,

Vendor_number VARCHAR(50) NOT NULL,

Product_name VARCHAR(50) NOT NULL,

Product_description VARCHAR(50) NOT NULL,

Product_price INT NOT NULL,

PRIMARY KEY (Product_ID),

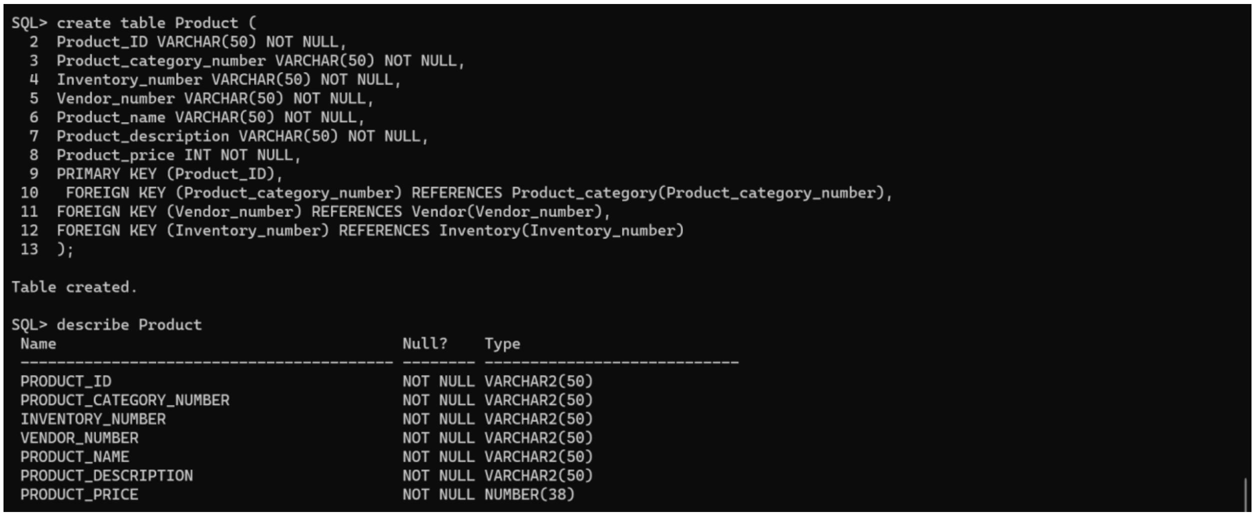
FOREIGN KEY (Product_category_number) REFERENCES Product_category(Product_category_number),

FOREIGN KEY (Vendor_number) REFERENCES Vendor(Vendor_number),

FOREIGN KEY (Inventory_number) REFERENCES Inventory(Inventory_number)

);

describe Product



```
SQL> create table Product (
  2  Product_ID VARCHAR(50) NOT NULL,
  3  Product_category_number VARCHAR(50) NOT NULL,
  4  Inventory_number VARCHAR(50) NOT NULL,
  5  Vendor_number VARCHAR(50) NOT NULL,
  6  Product_name VARCHAR(50) NOT NULL,
  7  Product_description VARCHAR(50) NOT NULL,
  8  Product_price INT NOT NULL,
  9  PRIMARY KEY (Product_ID),
 10   FOREIGN KEY (Product_category_number) REFERENCES Product_category(Product_category_number),
 11  FOREIGN KEY (Vendor_number) REFERENCES Vendor(Vendor_number),
 12  FOREIGN KEY (Inventory_number) REFERENCES Inventory(Inventory_number)
 13  );

Table created.

SQL> describe Product
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCT_ID                                NOT NULL VARCHAR2(50)
 PRODUCT_CATEGORY_NUMBER                   NOT NULL VARCHAR2(50)
 INVENTORY_NUMBER                          NOT NULL VARCHAR2(50)
 VENDOR_NUMBER                             NOT NULL VARCHAR2(50)
 PRODUCT_NAME                              NOT NULL VARCHAR2(50)
 PRODUCT_DESCRIPTION                       NOT NULL VARCHAR2(50)
 PRODUCT_PRICE                             NOT NULL NUMBER(38)
```

Figure 12 Creating and describing Product table

Table Description:

This table stores Product values such as Product ID, Product category number, Inventory number, Vendor number, Product name, product number, product description and product prices. They are not null.

- **Creating and describing Customer table**
  <u>Syntax</u>

  create table Customer (

  Customer_ID VARCHAR(50) NOT NULL,

  Customer_category_number VARCHAR(50) NOT NULL,

  Customer_name VARCHAR(50) NOT NULL,

  Customer_address VARCHAR(50) NOT NULL,

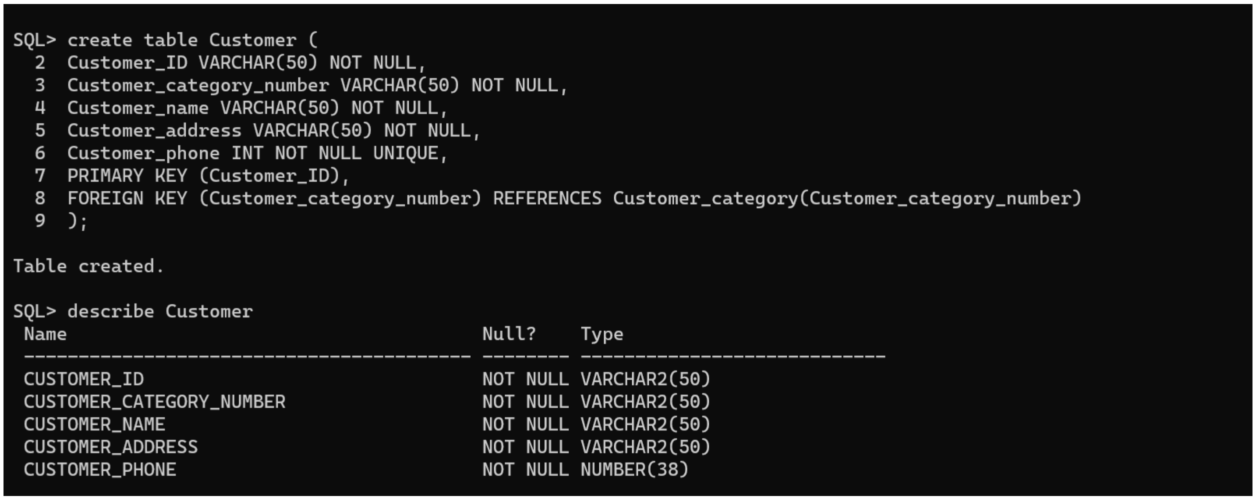
  Customer_phone INT NOT NULL UNIQUE,

  PRIMARY KEY (Customer_ID),

  FOREIGN KEY (Customer_category_number) REFERENCES Customer_category(Customer_category_number)

  );

  describe Customer



```
SQL> create table Customer (
  Customer_ID VARCHAR(50) NOT NULL,
  Customer_category_number VARCHAR(50) NOT NULL,
  Customer_name VARCHAR(50) NOT NULL,
  Customer_address VARCHAR(50) NOT NULL,
  Customer_phone INT NOT NULL UNIQUE,
  PRIMARY KEY (Customer_ID),
  FOREIGN KEY (Customer_category_number) REFERENCES Customer_category(Customer_category_number)
  );

Table created.

SQL> describe Customer
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL VARCHAR2(50)
 CUSTOMER_CATEGORY_NUMBER                  NOT NULL VARCHAR2(50)
 CUSTOMER_NAME                             NOT NULL VARCHAR2(50)
 CUSTOMER_ADDRESS                          NOT NULL VARCHAR2(50)
 CUSTOMER_PHONE                            NOT NULL NUMBER(38)
```

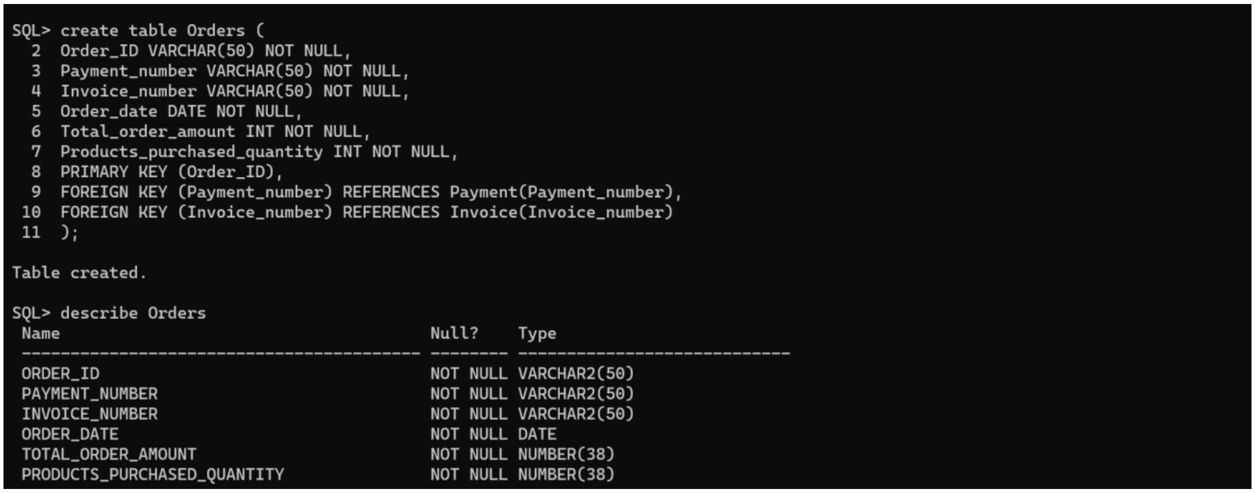
Figure 13 Creating and describing Customer table

Table description:

The customer table contains customer id, customer category number, customer name and customer address. The values are not null. It has one to many relationship with Product_customer order_customer and Order_product_customer table. It has many to one relationship with Customer_category table.

- **Creating and describing Order table**
  <u>Syntax</u>

```
create table Orders (
Order_ID VARCHAR(50) NOT NULL,
 Payment_number VARCHAR(50) NOT NULL,
 Invoice_number VARCHAR(50) NOT NULL,
 Order_date DATE NOT NULL,
Total_order_amount INT NOT NULL,
Products_purchased_quantity INT NOT NULL,
PRIMARY KEY (Order_ID),
FOREIGN KEY (Payment_number) REFERENCES Payment(Payment_number),
FOREIGN KEY (Invoice_number) REFERENCES Invoice(Invoice_number)
                    );
describe Orders
```



```
SQL> create table Orders (
  Order_ID VARCHAR(50) NOT NULL,
  Payment_number VARCHAR(50) NOT NULL,
  Invoice_number VARCHAR(50) NOT NULL,
  Order_date DATE NOT NULL,
  Total_order_amount INT NOT NULL,
  Products_purchased_quantity INT NOT NULL,
  PRIMARY KEY (Order_ID),
  FOREIGN KEY (Payment_number) REFERENCES Payment(Payment_number),
  FOREIGN KEY (Invoice_number) REFERENCES Invoice(Invoice_number)
  );

Table created.

SQL> describe Orders
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL VARCHAR2(50)
 PAYMENT_NUMBER                            NOT NULL VARCHAR2(50)
 INVOICE_NUMBER                            NOT NULL VARCHAR2(50)
 ORDER_DATE                                NOT NULL DATE
 TOTAL_ORDER_AMOUNT                        NOT NULL NUMBER(38)
 PRODUCTS_PURCHASED_QUANTITY               NOT NULL NUMBER(38)
```

Figure 14 Creating and describing Orders table

Table description:

The orders table contain values like order id, payment number, invoice number, order date, total order amount and products purchased quantity. This table has all not null values. It has one to many relationship with order_customer, order_product and order_product_customer table.It has many to one relationship with payment and invoive tables.

- **Creating and describing Order_Customer table**
  <u>Syntax</u>

create table Order_Customer (

Order_ID VARCHAR(50) NOT NULL,

Customer_ID VARCHAR(50) NOT NULL,

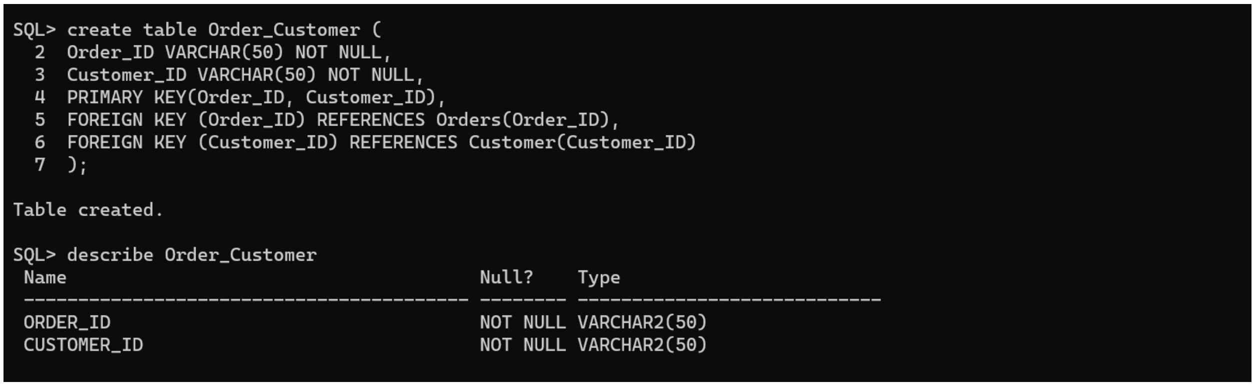
PRIMARY KEY(Order_ID, Customer_ID),

FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),

FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)

);

describe Order_Customer



```
SQL> create table Order_Customer (
  2  Order_ID VARCHAR(50) NOT NULL,
  3  Customer_ID VARCHAR(50) NOT NULL,
  4  PRIMARY KEY(Order_ID, Customer_ID),
  5  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
  6  FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
  7  );

Table created.

SQL> describe Order_Customer
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL VARCHAR2(50)
 CUSTOMER_ID                               NOT NULL VARCHAR2(50)
```

Figure 15 Creating and describing Order_Customer table

Table Description:

This table consists values like Order id and Customer Id as foreign keys. It has many to one relationship with Order and Customer tables.

- **Creating and describing Order_Product table**
  Syntax

create table Order_Product (

Order_ID VARCHAR(50) NOT NULL,

Product_ID VARCHAR(50) NOT NULL,

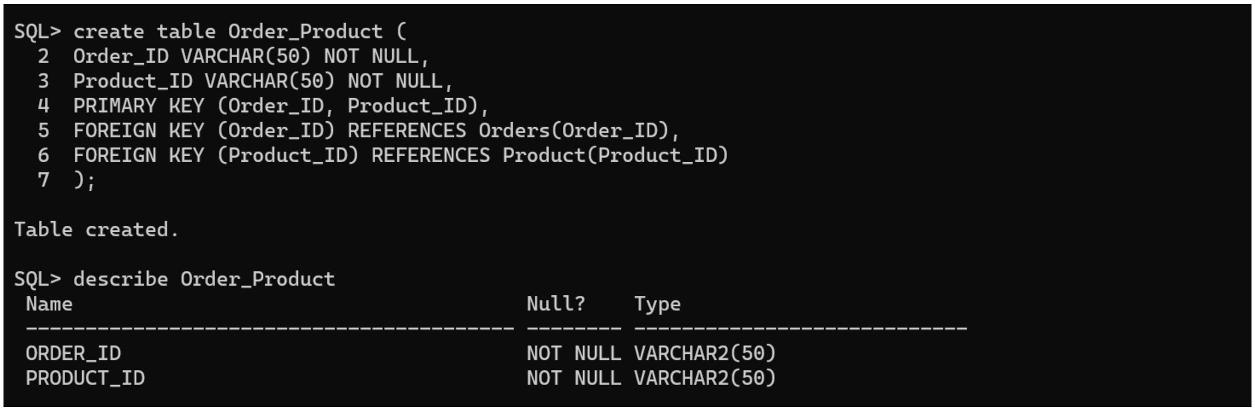
PRIMARY KEY (Order_ID, Product_ID),

FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),

FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)

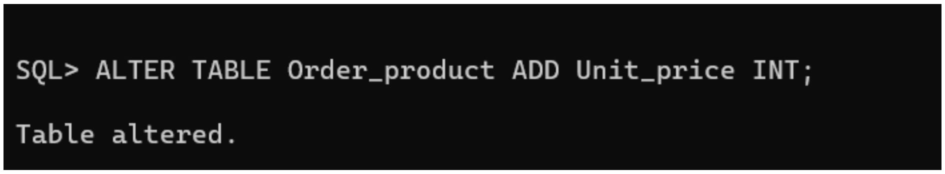
);

describe Order_Product



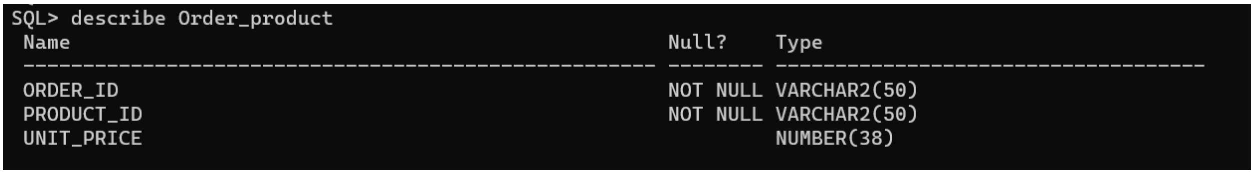
```
SQL> create table Order_Product (
  2  Order_ID VARCHAR(50) NOT NULL,
  3  Product_ID VARCHAR(50) NOT NULL,
  4  PRIMARY KEY (Order_ID, Product_ID),
  5  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
  6  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
  7  );

Table created.

SQL> describe Order_Product
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL VARCHAR2(50)
 PRODUCT_ID                                NOT NULL VARCHAR2(50)
```

Figure 16 Creating and describing Order_Product table



```
SQL> ALTER TABLE Order_product ADD Unit_price INT;

Table altered.
```

Figure 17 Altering Order_Product to add Unit price



```
SQL> describe Order_product
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ORDER_ID                                              NOT NULL VARCHAR2(50)
 PRODUCT_ID                                            NOT NULL VARCHAR2(50)
 UNIT_PRICE                                                     NUMBER(38)
```

Figure 18 Describing Order_product again

Table Description:

This table contains Order ID and Product ID as foreign keys and Unit price as a data. It has many to one relationship with Order and Product tables.

- **Creating Product_customer table**
  Syntax

  create table Product_Customer (

Product_ID VARCHAR(50) NOT NULL,

Customer_ID VARCHAR(50) NOT NULL,

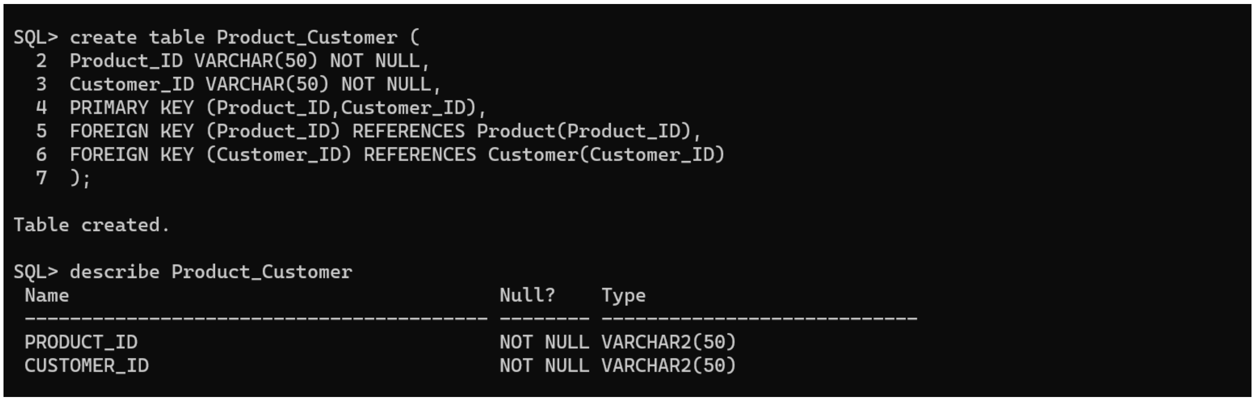
PRIMARY KEY (Product_ID,Customer_ID),

FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),

FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)

);

describe Product_Customer



```
SQL> create table Product_Customer (
  2  Product_ID VARCHAR(50) NOT NULL,
  3  Customer_ID VARCHAR(50) NOT NULL,
  4  PRIMARY KEY (Product_ID,Customer_ID),
  5  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
  6  FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
  7  );

Table created.

SQL> describe Product_Customer
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCT_ID                                NOT NULL VARCHAR2(50)
 CUSTOMER_ID                               NOT NULL VARCHAR2(50)
```

Figure 19 Creating and describing Product_customer table

Table Description and relation:

The Table contains Product ID and Customer ID as foreign keys. It has many to one relationship with Product and Customer tables**.**

- **Creating table Order_Product_Customer**
  <u>Syntax</u>

  create table Order_Product_Customer (

  Order_ID VARCHAR(50) NOT NULL,

  Product_ID VARCHAR(50) NOT NULL,

  Customer_ID VARCHAR(50) NOT NULL,

  PRIMARY KEY (Order_ID, Product_ID, Customer_ID),

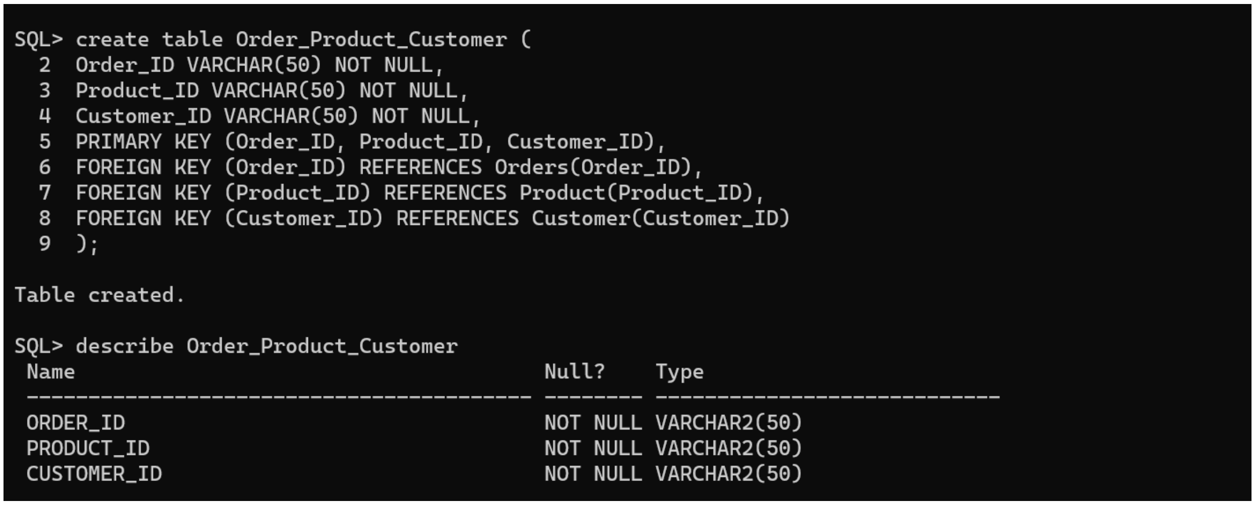
  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),

  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),

  FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)

  );

  describe Order_Product_Customer



```
SQL> create table Order_Product_Customer (
  2  Order_ID VARCHAR(50) NOT NULL,
  3  Product_ID VARCHAR(50) NOT NULL,
  4  Customer_ID VARCHAR(50) NOT NULL,
  5  PRIMARY KEY (Order_ID, Product_ID, Customer_ID),
  6  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
  7  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
  8  FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
  9  );

Table created.

SQL> describe Order_Product_Customer
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL VARCHAR2(50)
 PRODUCT_ID                                NOT NULL VARCHAR2(50)
 CUSTOMER_ID                               NOT NULL VARCHAR2(50)
```

Figure 20 Creating and describing Order_Product_Customer table

Table Description and relation:

This table contains Order ID, Product ID and Customer ID as foreign keys. It has many to one relationship with Order, Product and Customer tables.

5.2. Inserting values in the tables and checking

- **Inserting values in Product_category table and checking**

Syntax

insert into Product_category values

('PC01','Laptop');

insert into Product_category values

('PC02','Mobiles');

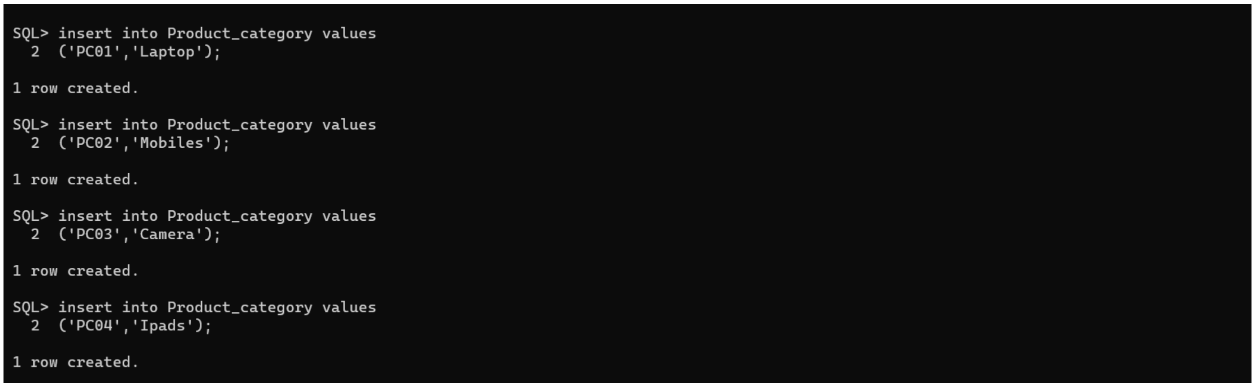
insert into Product_category values

('PC03','Camera');

insert into Product_category values

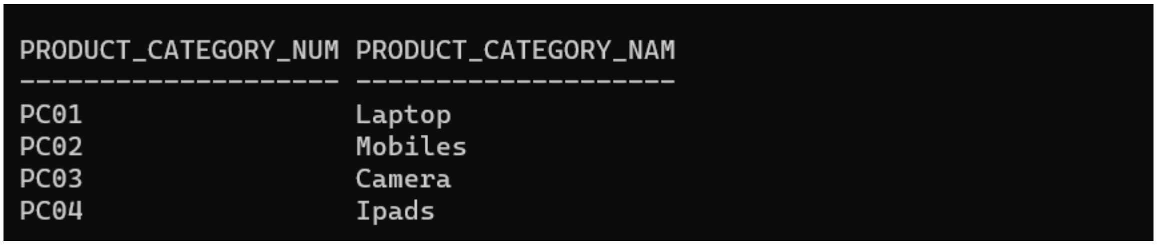
('PC04','Ipads');

Select * from Product_category;



```
SQL> insert into Product_category values
  2  ('PC01','Laptop');

1 row created.

SQL> insert into Product_category values
  2  ('PC02','Mobiles');

1 row created.

SQL> insert into Product_category values
  2  ('PC03','Camera');

1 row created.

SQL> insert into Product_category values
  2  ('PC04','Ipads');

1 row created.
```

Figure 21 Inserting values in Product_category table



```
PRODUCT_CATEGORY_NUM PRODUCT_CATEGORY_NAM
-------------------- --------------------
PC01                 Laptop
PC02                 Mobiles
PC03                 Camera
PC04                 Ipads
```

Figure 22 Checking the values

- **Inserting into Customer_category table and checking**
  Syntax

insert into Customer_category values

('CC01','Regular',0.00);

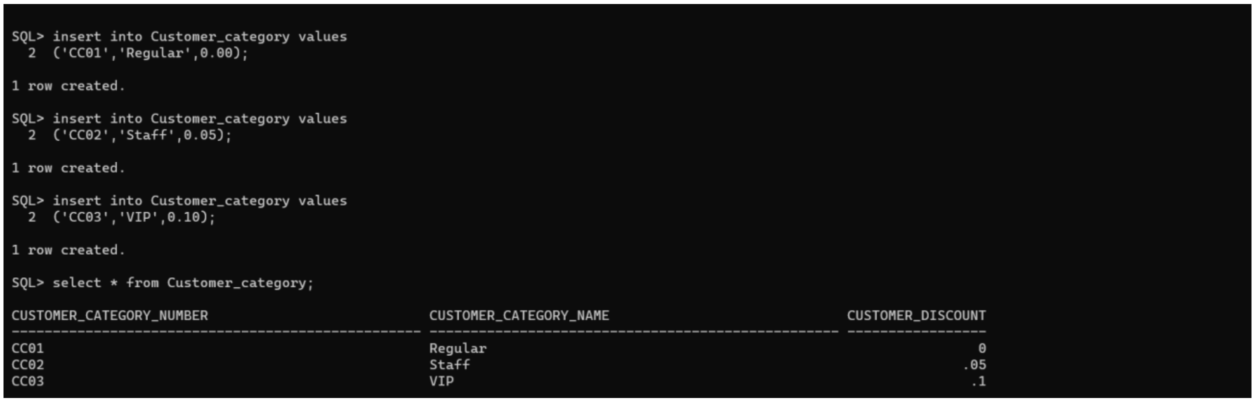
insert into Customer_category values

('CC02','Staff',0.05);

insert into Customer_category values

('CC03','VIP',0.10);

select * from Customer_category;



```
SQL> insert into Customer_category values
  2  ('CC01','Regular',0.00);

1 row created.

SQL> insert into Customer_category values
  2  ('CC02','Staff',0.05);

1 row created.

SQL> insert into Customer_category values
  2  ('CC03','VIP',0.10);

1 row created.

SQL> select * from Customer_category;

CUSTOMER_CATEGORY_NUMBER                           CUSTOMER_CATEGORY_NAME                             CUSTOMER_DISCOUNT
-------------------------------------------------- -------------------------------------------------- -----------------
CC01                                               Regular                                                            0
CC02                                               Staff                                                            .05
CC03                                               VIP                                                               .1
```

Figure 23 Inserting into Customer_category table and checking

- **Inserting into Inventory values and checking**
  Syntax

  insert into Inventory values

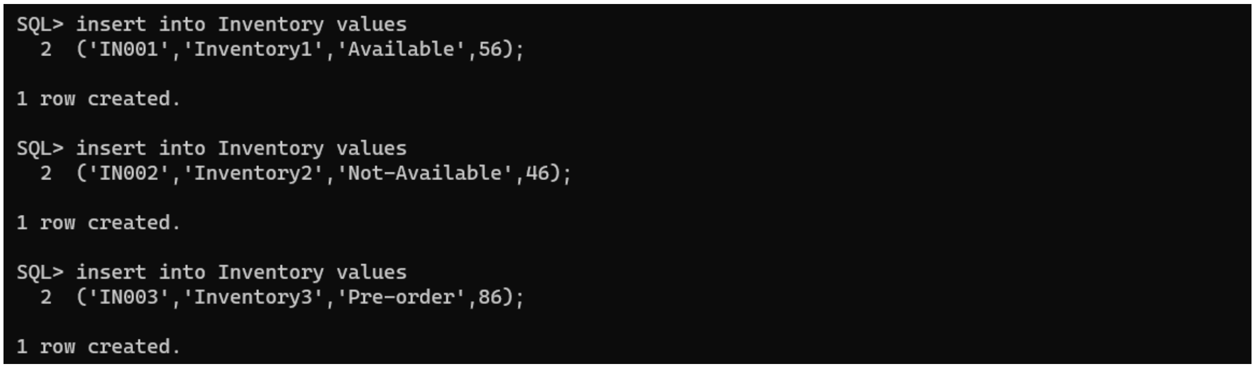
  ('IN001','Inventory1','Available',56);

  insert into Inventory values

  ('IN002','Inventory2','Not-Available',46);

  insert into Inventory values

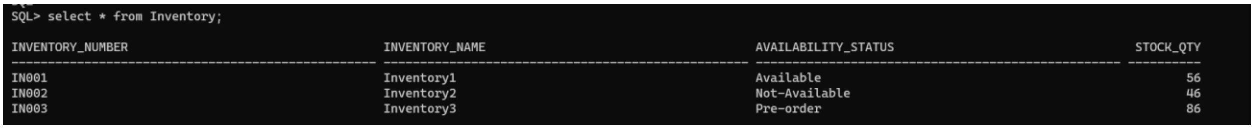
  ('IN003','Inventory3','Pre-order',86);



```
SQL> insert into Inventory values
  2  ('IN001','Inventory1','Available',56);

1 row created.

SQL> insert into Inventory values
  2  ('IN002','Inventory2','Not-Available',46);

1 row created.

SQL> insert into Inventory values
  2  ('IN003','Inventory3','Pre-order',86);

1 row created.
```

Figure 24 Inserting into inventory table

select * from Inventory;



```
SQL> select * from Inventory;

INVENTORY_NUMBER                                   INVENTORY_NAME                                     AVAILABILITY_STATUS                                 STOCK_QTY
-------------------------------------------------- -------------------------------------------------- -------------------------------------------------- ----------
IN001                                              Inventory1                                         Available                                                  56
IN002                                              Inventory2                                         Not-Available                                              46
IN003                                              Inventory3                                         Pre-order                                                  86
```

Figure 25 Checking Inventory values

- **Inserting into Payment values and checking**
  <u>Syntax</u>

  insert into payment values

  ('PAY1','COD');

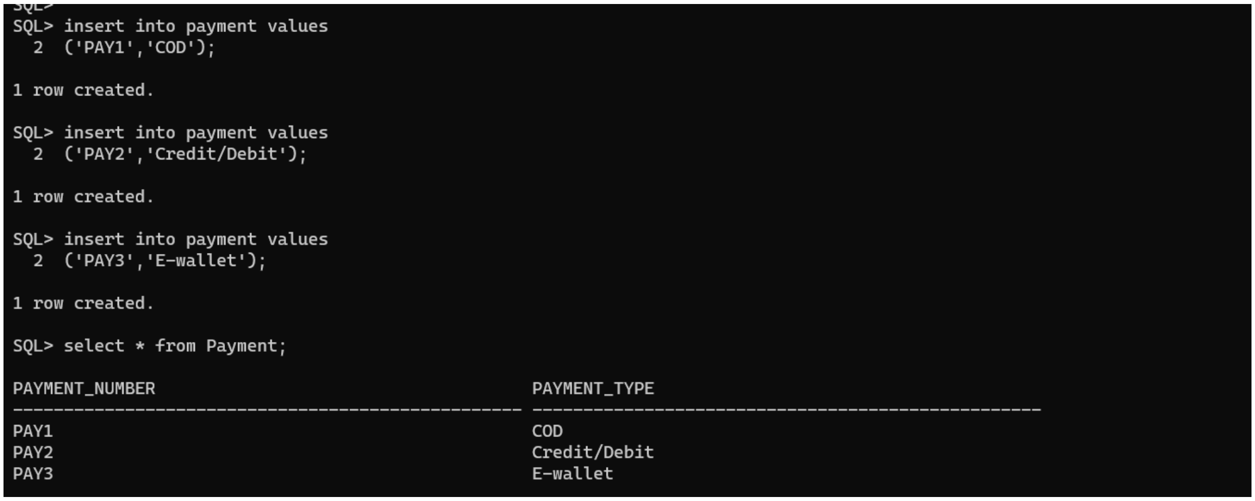
  insert into payment values

  ('PAY2','Credit/Debit');

  insert into payment values

  ('PAY3','E-wallet');

  select * from Payment;



```
SQL> insert into payment values
  2  ('PAY1','COD');

1 row created.

SQL> insert into payment values
  2  ('PAY2','Credit/Debit');

1 row created.

SQL> insert into payment values
  2  ('PAY3','E-wallet');

1 row created.

SQL> select * from Payment;

PAYMENT_NUMBER                                     PAYMENT_TYPE
-------------------------------------------------- --------------------------------------------------
PAY1                                               COD
PAY2                                               Credit/Debit
PAY3                                               E-wallet
```

Figure 26 Inserting into payment table ad checking

- **Inserting values in Invoice table and checking**
  <u>Syntax</u>

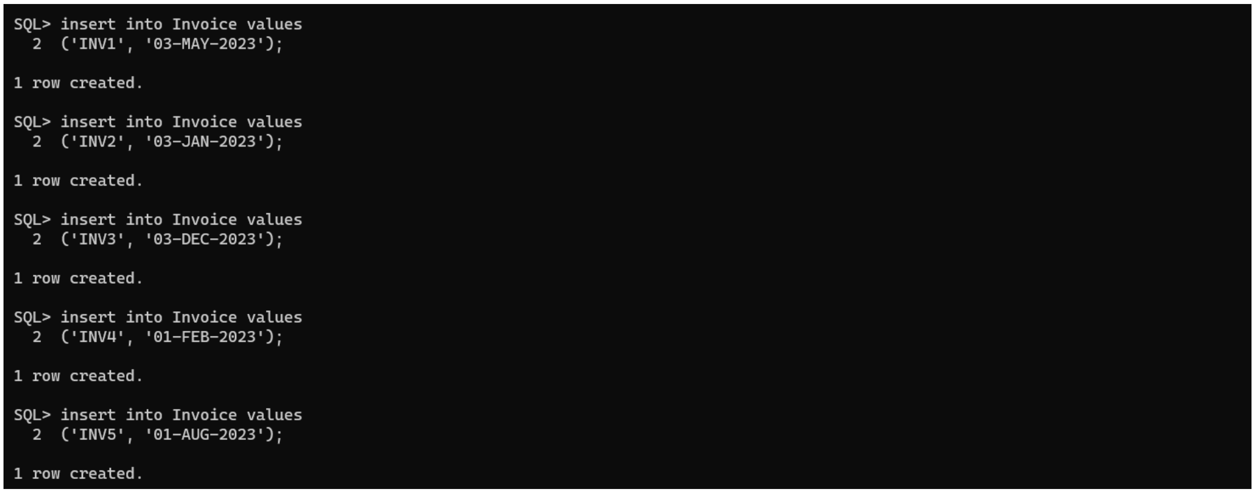
  insert into Invoice values

  ('INV1', '03-MAY-2023');

  insert into Invoice values

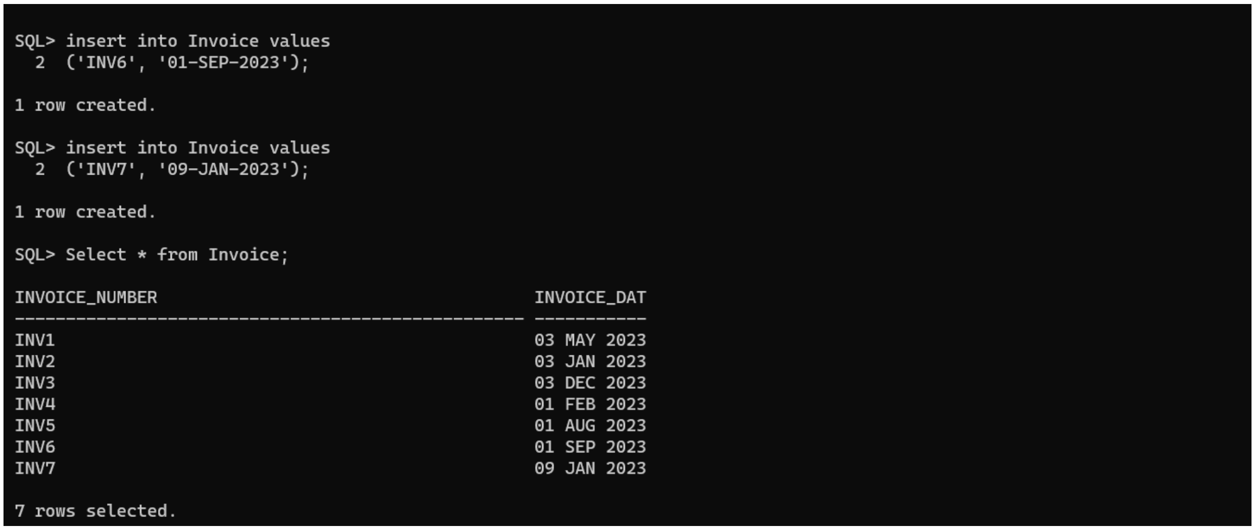
  ('INV2', '03-JAN-2023');

```
insert into Invoice values
 ('INV3', '03-DEC-2023');
insert into Invoice values
 ('INV4', '01-FEB-2023');
insert into Invoice values
 ('INV5', '01-AUG-2023');
insert into Invoice values
 ('INV6', '01-SEP-2023');
insert into Invoice values
 ('INV7', '09-JAN-2023');
Select * from Invoice;
```



```
SQL> insert into Invoice values
  2  ('INV1', '03-MAY-2023');

1 row created.

SQL> insert into Invoice values
  2  ('INV2', '03-JAN-2023');

1 row created.

SQL> insert into Invoice values
  2  ('INV3', '03-DEC-2023');

1 row created.

SQL> insert into Invoice values
  2  ('INV4', '01-FEB-2023');

1 row created.

SQL> insert into Invoice values
  2  ('INV5', '01-AUG-2023');

1 row created.
```

Figure 27 Inserting values in Invoice values



```
SQL> insert into Invoice values
  2  ('INV6', '01-SEP-2023');

1 row created.

SQL> insert into Invoice values
  2  ('INV7', '09-JAN-2023');

1 row created.

SQL> Select * from Invoice;

INVOICE_NUMBER                                     INVOICE_DAT
-------------------------------------------------- -----------
INV1                                               03 MAY 2023
INV2                                               03 JAN 2023
INV3                                               03 DEC 2023
INV4                                               01 FEB 2023
INV5                                               01 AUG 2023
INV6                                               01 SEP 2023
INV7                                               09 JAN 2023

7 rows selected.
```

Figure 28 Checking Invoice values

- **Inserting values in Vendor table and checking**
  Syntax

  insert into Vendor values

  ('V001','ASUS','Austin',2);

  insert into Vendor values

  ('V002','Samsung','South Korea',9);

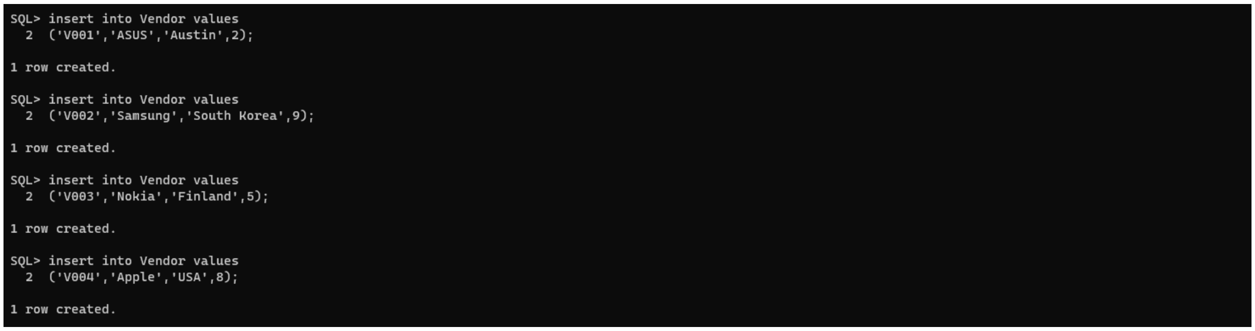
  insert into Vendor values

  ('V003','Nokia','Finland',5);

  insert into Vendor values

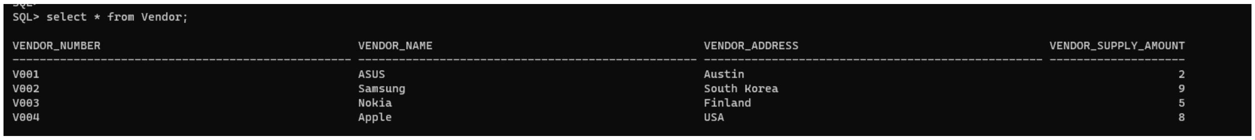
  ('V004','Apple','USA',8);

  select * from Vendor;



```
SQL> insert into Vendor values
  2  ('V001','ASUS','Austin',2);

1 row created.

SQL> insert into Vendor values
  2  ('V002','Samsung','South Korea',9);

1 row created.

SQL> insert into Vendor values
  2  ('V003','Nokia','Finland',5);

1 row created.

SQL> insert into Vendor values
  2  ('V004','Apple','USA',8);

1 row created.
```

Figure 29 Inserting values in vendor



```
SQL> select * from Vendor;

VENDOR_NUMBER                                      VENDOR_NAME                                        VENDOR_ADDRESS                                     VENDOR_SUPPLY_AMOUNT
-------------------------------------------------- -------------------------------------------------- -------------------------------------------------- --------------------
V001                                               ASUS                                               Austin                                                                2
V002                                               Samsung                                            South Korea                                                           9
V003                                               Nokia                                              Finland                                                               5
V004                                               Apple                                              USA                                                                   8
```

Figure 30 Displaying values

- **Inserting values in product and checking**

  <u>Syntax</u>

  insert into product values

  ('P01','PC01','IN001','V001','Asus X5','DES01',128000);

  insert into product values

  ('P02','PC02','IN002','V002','Sansung galaxy X','DES02',95000);

  insert into product values

  ('P03','PC03','IN003','V003','Nokia N5','DES03',20000);

  insert into product values

  ('P04','PC04','IN003','V004','Mackbook','DES04',32000);

  insert into product values

  ('P05','PC01','IN001','V001','Asus X6','DES05',196000);

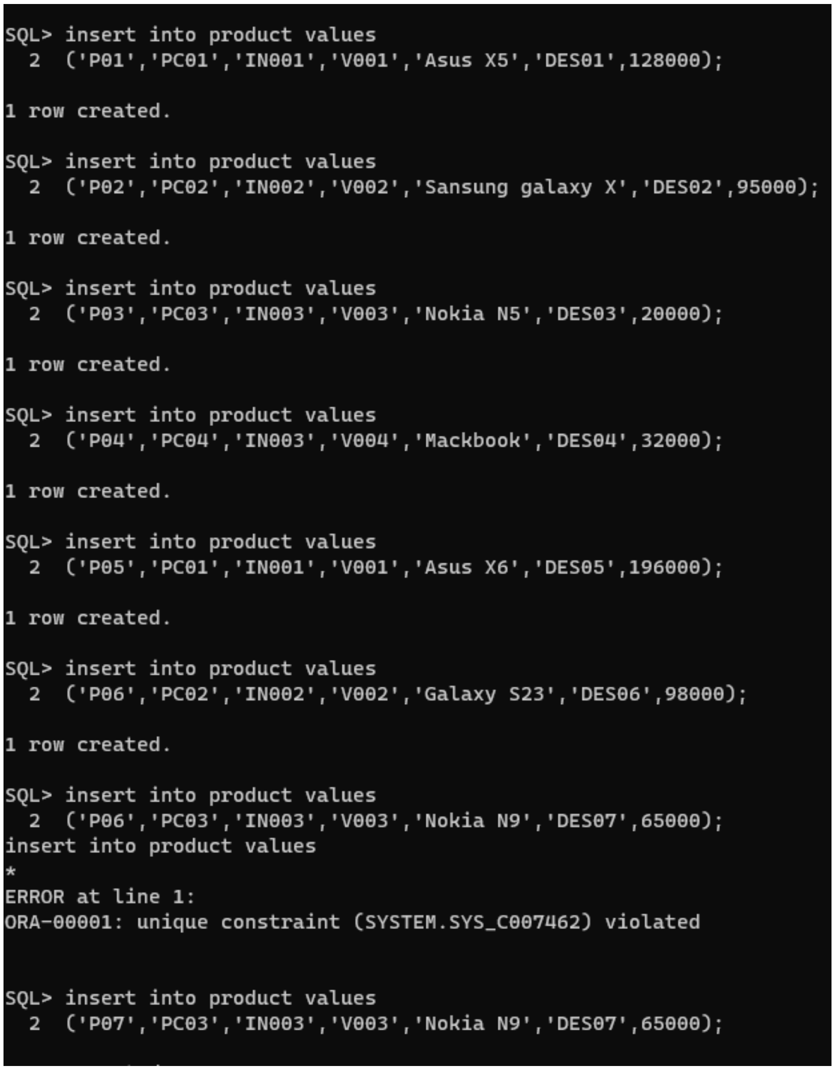
  insert into product values

  ('P06','PC02','IN002','V002','Galaxy S23','DES06',98000);

  insert into product values

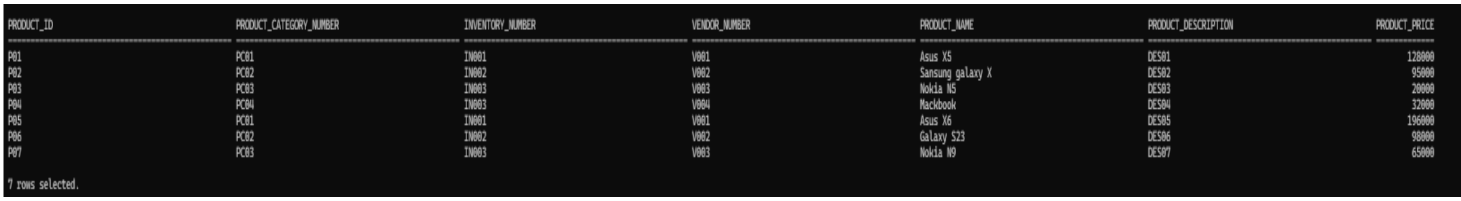
  ('P07','PC03','IN003','V003','Nokia N9','DES07',65000);

  SQL> select * from product;



```
SQL> insert into product values
  2  ('P01','PC01','IN001','V001','Asus X5','DES01',128000);

1 row created.

SQL> insert into product values
  2  ('P02','PC02','IN002','V002','Sansung galaxy X','DES02',95000);

1 row created.

SQL> insert into product values
  2  ('P03','PC03','IN003','V003','Nokia N5','DES03',20000);

1 row created.

SQL> insert into product values
  2  ('P04','PC04','IN003','V004','Mackbook','DES04',32000);

1 row created.

SQL> insert into product values
  2  ('P05','PC01','IN001','V001','Asus X6','DES05',196000);

1 row created.

SQL> insert into product values
  2  ('P06','PC02','IN002','V002','Galaxy S23','DES06',98000);

1 row created.

SQL> insert into product values
  2  ('P06','PC03','IN003','V003','Nokia N9','DES07',65000);
insert into product values
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C007462) violated


SQL> insert into product values
  2  ('P07','PC03','IN003','V003','Nokia N9','DES07',65000);
```

Figure 31 Inserting values in product



```
PRODUCT_ID  PRODUCT_CATEGORY_NUMBER  INVENTORY_NUMBER  VENDOR_NUMBER  PRODUCT_NAME      PRODUCT_DESCRIPTION  PRODUCT_PRICE
----------  -----------------------  ----------------  -------------  ----------------  -------------------  -------------
P01         PC01                     IN001             V001           Asus X5           DES01                       128000
P02         PC02                     IN002             V002           Sansung galaxy X  DES02                        95000
P03         PC03                     IN003             V003           Nokia N5          DES03                        20000
P04         PC04                     IN003             V004           Mackbook          DES04                        32000
P05         PC01                     IN001             V001           Asus X6           DES05                       196000
P06         PC02                     IN002             V002           Galaxy S23        DES06                        98000
P07         PC03                     IN003             V003           Nokia N9          DES07                        65000

7 rows selected.
```

Figure 32 Checking the values

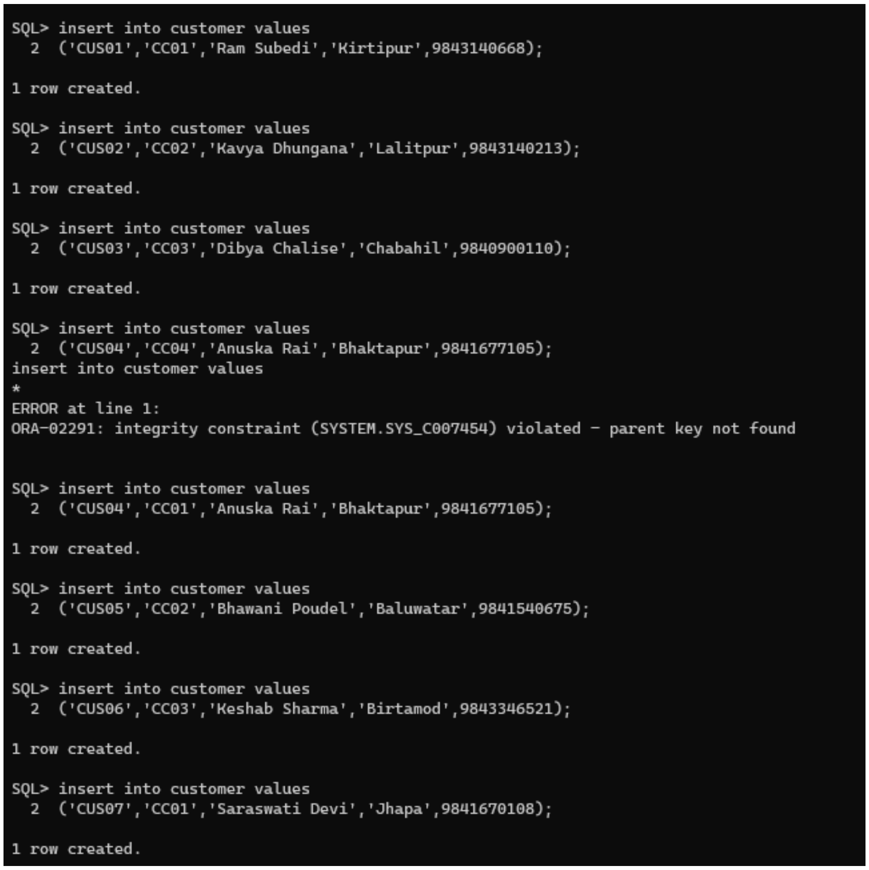
- **Inserting values in Customer table and checking**
  <u>Syntax</u>

  insert into customer values

  ('CUS01','CC01','Ram Subedi','Kirtipur',9843140668);

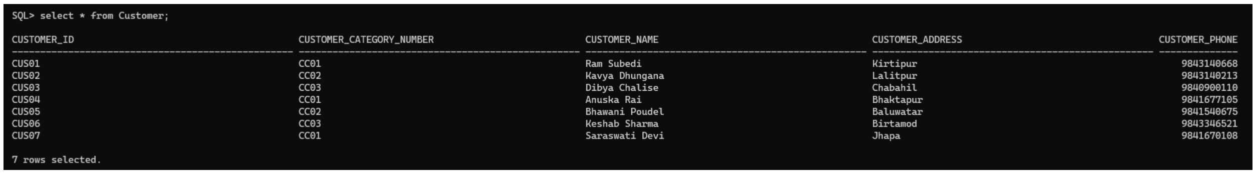
  insert into customer values

```
 ('CUS02','CC02','Kavya Dhungana','Lalitpur',9843140213);
insert into customer values
 ('CUS03','CC03','Dibya Chalise','Chabahil',9840900110);
insert into customer values
('CUS04','CC01','Anuska Rai','Bhaktapur',9841677105);
insert into customer values
('CUS05','CC02','Bhawani Poudel','Baluwatar',9841540675);
insert into customer values
('CUS06','CC03','Keshab Sharma','Birtamod',9843346521);
insert into customer values
 ('CUS07','CC01','Saraswati Devi','Jhapa',9841670108);
```



```
SQL> insert into customer values
  2  ('CUS01','CC01','Ram Subedi','Kirtipur',9843140668);

1 row created.

SQL> insert into customer values
  2  ('CUS02','CC02','Kavya Dhungana','Lalitpur',9843140213);

1 row created.

SQL> insert into customer values
  2  ('CUS03','CC03','Dibya Chalise','Chabahil',9840900110);

1 row created.

SQL> insert into customer values
  2  ('CUS04','CC04','Anuska Rai','Bhaktapur',9841677105);
insert into customer values
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.SYS_C007454) violated - parent key not found


SQL> insert into customer values
  2  ('CUS04','CC01','Anuska Rai','Bhaktapur',9841677105);

1 row created.

SQL> insert into customer values
  2  ('CUS05','CC02','Bhawani Poudel','Baluwatar',9841540675);

1 row created.

SQL> insert into customer values
  2  ('CUS06','CC03','Keshab Sharma','Birtamod',9843346521);

1 row created.

SQL> insert into customer values
  2  ('CUS07','CC01','Saraswati Devi','Jhapa',9841670108);

1 row created.
```

Figure 33 Inserting values in Customer table



```
SQL> select * from Customer;

CUSTOMER_ID                                        CUSTOMER_CATEGORY_NUMBER                           CUSTOMER_NAME                                      CUSTOMER_ADDRESS                                   CUSTOMER_PHONE
-------------------------------------------------- -------------------------------------------------- -------------------------------------------------- -------------------------------------------------- --------------
CUS01                                              CC01                                               Ram Subedi                                         Kirtipur                                               9843140668
CUS02                                              CC02                                               Kavya Dhungana                                     Lalitpur                                               9843140213
CUS03                                              CC03                                               Dibya Chalise                                      Chabahil                                               9840900110
CUS04                                              CC01                                               Anuska Rai                                         Bhaktapur                                              9841677105
CUS05                                              CC02                                               Bhawani Poudel                                     Baluwatar                                              9841540675
CUS06                                              CC03                                               Keshab Sharma                                      Birtamod                                               9843346521
CUS07                                              CC01                                               Saraswati Devi                                     Jhapa                                                  9841670108

7 rows selected.
```

Figure 34 Checking values of customer

- **Inserting values in orders table**

  <u>Syntax</u>

  insert into orders values

  ('ORD01','PAY1','INV1','01-MAY-2023','65000',5);

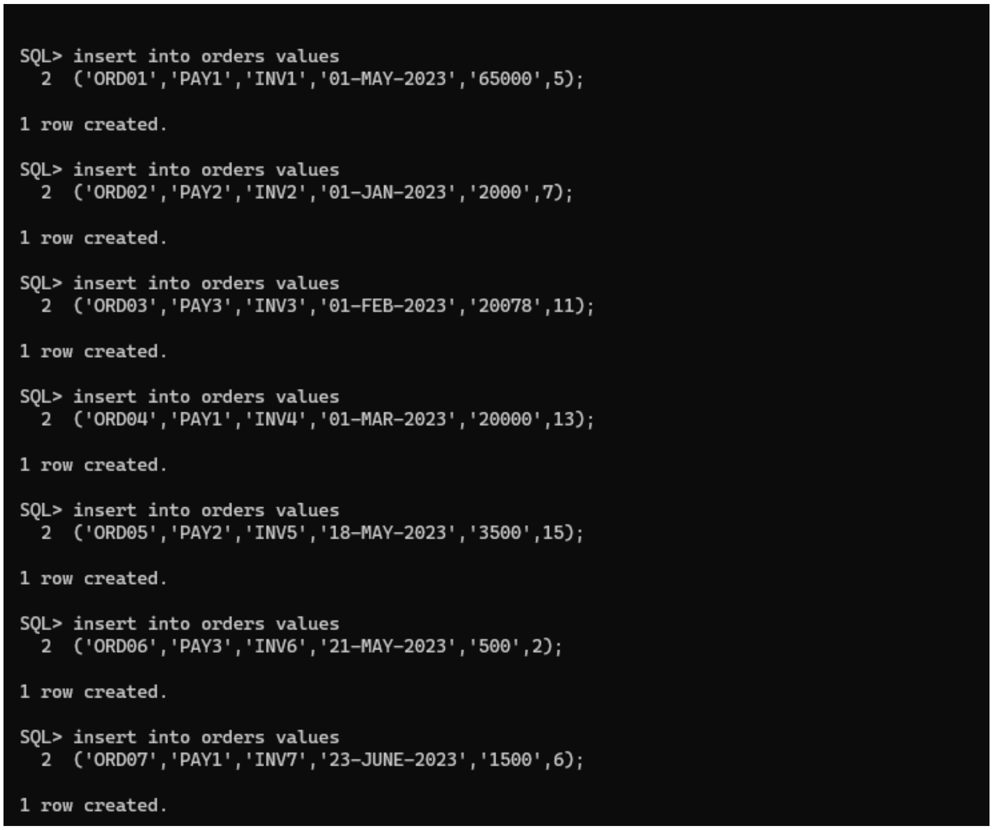
  insert into orders values

  ('ORD02','PAY2','INV2','01-JAN-2023','2000',7);

  insert into orders values
  ('ORD03','PAY3','INV3','01-FEB-2023','20078',11);

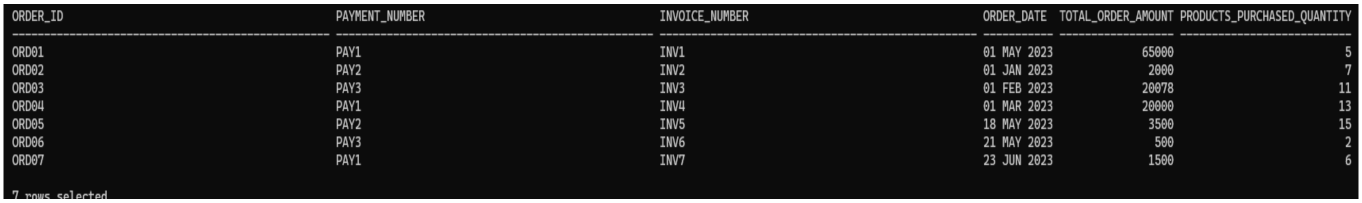
  insert into orders values

```
 ('ORD04','PAY1','INV4','01-MAR-2023','20000',13);
insert into orders values
 ('ORD05','PAY2','INV5','18-MAY-2023','3500',15);
insert into orders values
 ('ORD06','PAY3','INV6','21-MAY-2023','500',2);
insert into orders values
 ('ORD07','PAY1','INV7','23-JUNE-2023','1500',6);
select * from Orders;
```



```
SQL> insert into orders values
  2  ('ORD01','PAY1','INV1','01-MAY-2023','65000',5);

1 row created.

SQL> insert into orders values
  2  ('ORD02','PAY2','INV2','01-JAN-2023','2000',7);

1 row created.

SQL> insert into orders values
  2  ('ORD03','PAY3','INV3','01-FEB-2023','20078',11);

1 row created.

SQL> insert into orders values
  2  ('ORD04','PAY1','INV4','01-MAR-2023','20000',13);

1 row created.

SQL> insert into orders values
  2  ('ORD05','PAY2','INV5','18-MAY-2023','3500',15);

1 row created.

SQL> insert into orders values
  2  ('ORD06','PAY3','INV6','21-MAY-2023','500',2);

1 row created.

SQL> insert into orders values
  2  ('ORD07','PAY1','INV7','23-JUNE-2023','1500',6);

1 row created.
```

Figure 35 Inserting values in Orders table

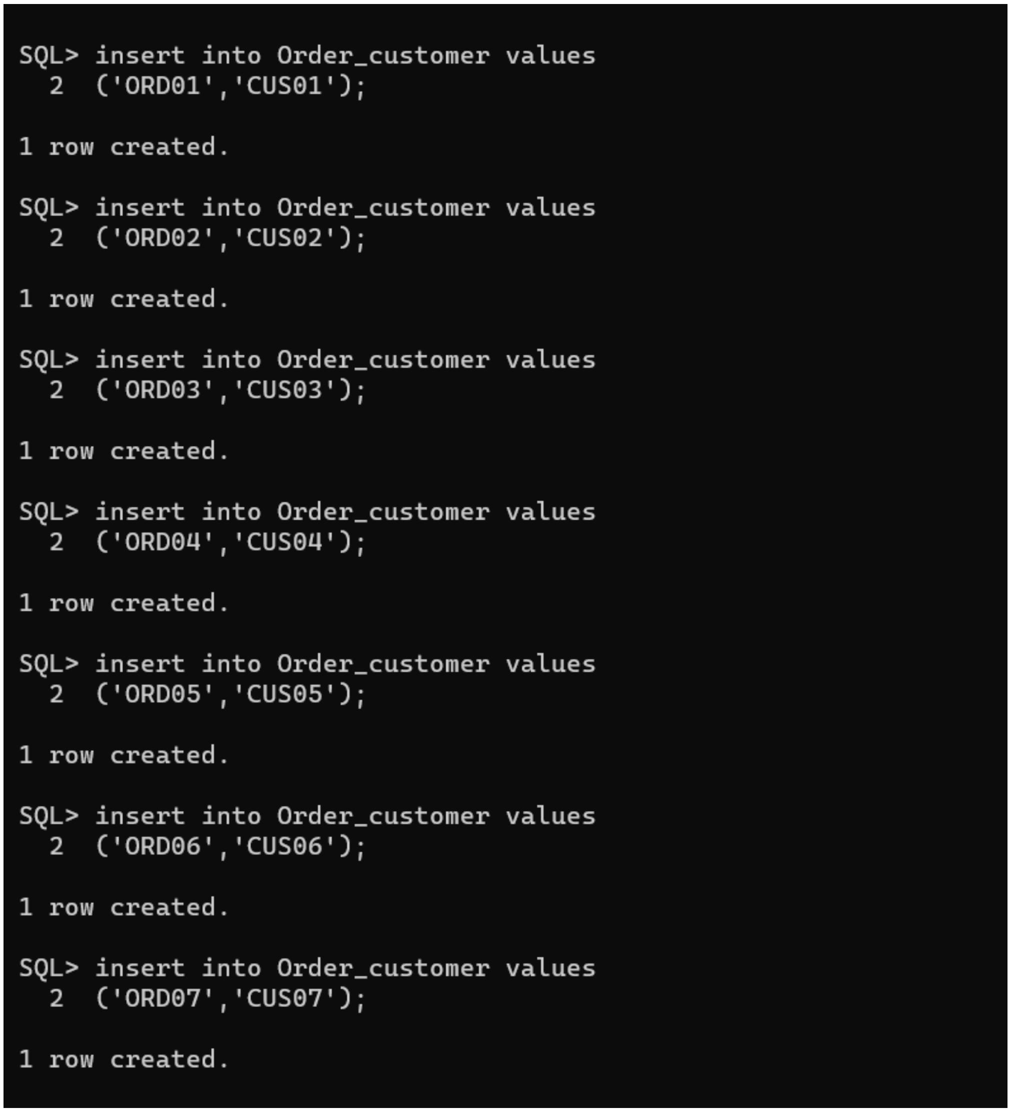


```
ORDER_ID                                           PAYMENT_NUMBER                                     INVOICE_NUMBER                                     ORDER_DATE  TOTAL_ORDER_AMOUNT PRODUCTS_PURCHASED_QUANTITY
-------------------------------------------------- -------------------------------------------------- -------------------------------------------------- ----------- ------------------ ---------------------------
ORD01                                              PAY1                                               INV1                                               01 MAY 2023              65000                           5
ORD02                                              PAY2                                               INV2                                               01 JAN 2023               2000                           7
ORD03                                              PAY3                                               INV3                                               01 FEB 2023              20078                          11
ORD04                                              PAY1                                               INV4                                               01 MAR 2023              20000                          13
ORD05                                              PAY2                                               INV5                                               18 MAY 2023               3500                          15
ORD06                                              PAY3                                               INV6                                               21 MAY 2023                500                           2
ORD07                                              PAY1                                               INV7                                               23 JUN 2023               1500                           6

7 rows selected
```

Figure 36 Checking the values of Orders table

- **Inserting value in Order_customer table and checking**
  <u>Syntax</u>

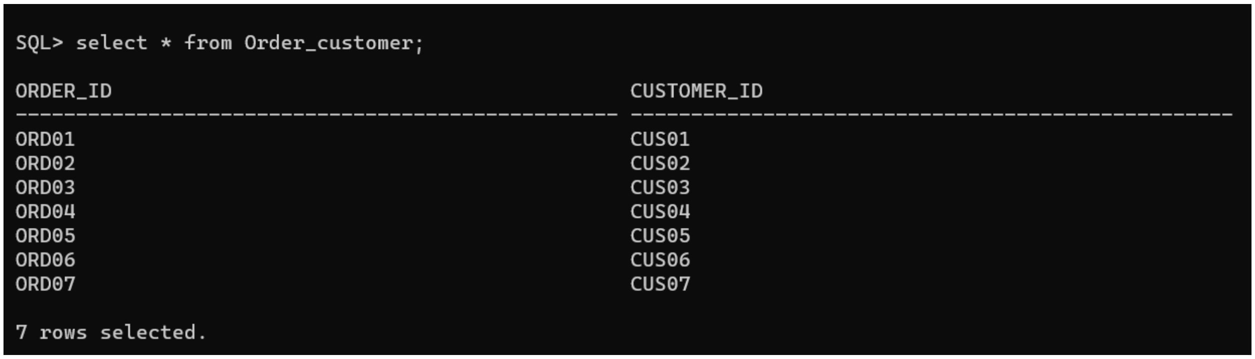
```
insert into Order_customer values
 ('ORD01','CUS01');
insert into Order_customer values
 ('ORD02','CUS02');
insert into Order_customer values
 ('ORD03','CUS03');
insert into Order_customer values
 ('ORD04','CUS04');
insert into Order_customer values
 ('ORD05','CUS05');
insert into Order_customer values
 ('ORD06','CUS06');
insert into Order_customer values
 ('ORD07','CUS07');
```



```
SQL> insert into Order_customer values
  2  ('ORD01','CUS01');

1 row created.

SQL> insert into Order_customer values
  2  ('ORD02','CUS02');

1 row created.

SQL> insert into Order_customer values
  2  ('ORD03','CUS03');

1 row created.

SQL> insert into Order_customer values
  2  ('ORD04','CUS04');

1 row created.

SQL> insert into Order_customer values
  2  ('ORD05','CUS05');

1 row created.

SQL> insert into Order_customer values
  2  ('ORD06','CUS06');

1 row created.

SQL> insert into Order_customer values
  2  ('ORD07','CUS07');

1 row created.
```

Figure 37 Inserting values in Order_customer table

select * from Order_customer;

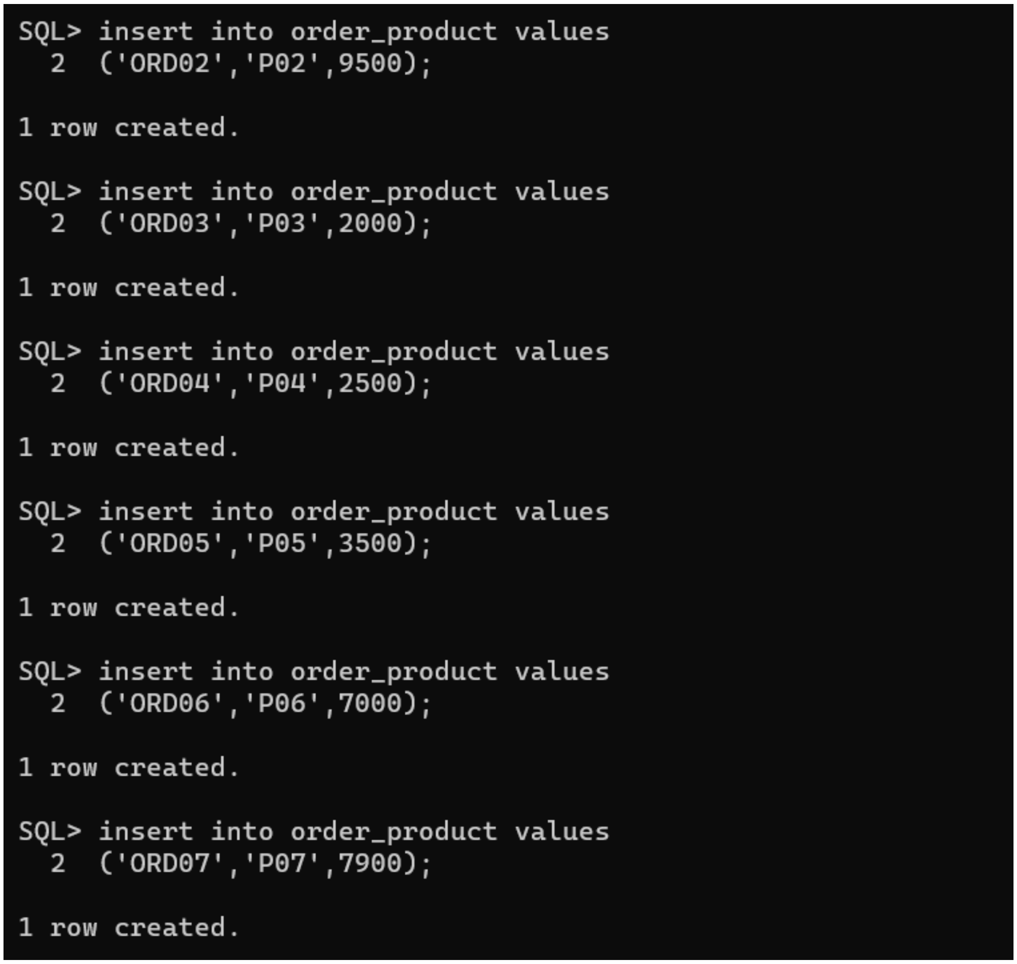


```
SQL> select * from Order_customer;

ORDER_ID                                           CUSTOMER_ID
-------------------------------------------------- --------------------------------------------------
ORD01                                              CUS01
ORD02                                              CUS02
ORD03                                              CUS03
ORD04                                              CUS04
ORD05                                              CUS05
ORD06                                              CUS06
ORD07                                              CUS07

7 rows selected.
```

Figure 38 Checking values of Order_customer table

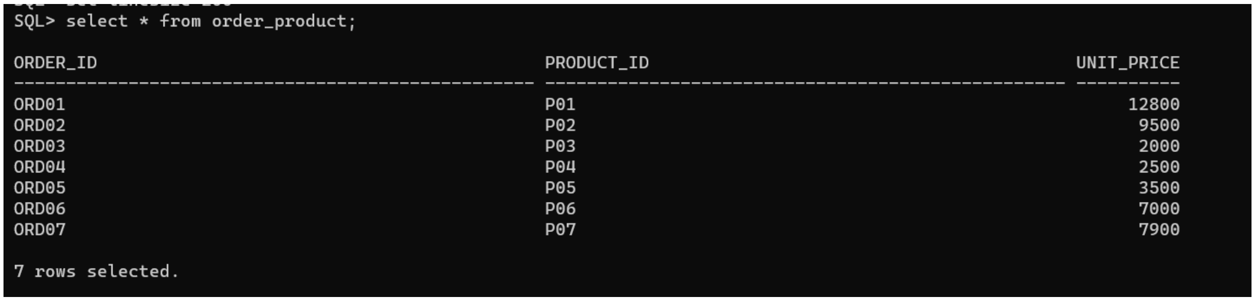
- **Inserting values in Order_product table**
  Syntax

```
insert into order_product values
 ('ORD02','P02',9500);
insert into order_product values
 ('ORD03','P03',2000);
insert into order_product values
 ('ORD04','P04',2500);
insert into order_product values
('ORD05','P05',3500);
insert into order_product values
 ('ORD06','P06',7000);
insert into order_product values
('ORD07','P07',7900);
```



```
SQL> insert into order_product values
  2  ('ORD02','P02',9500);

1 row created.

SQL> insert into order_product values
  2  ('ORD03','P03',2000);

1 row created.

SQL> insert into order_product values
  2  ('ORD04','P04',2500);

1 row created.

SQL> insert into order_product values
  2  ('ORD05','P05',3500);

1 row created.

SQL> insert into order_product values
  2  ('ORD06','P06',7000);

1 row created.

SQL> insert into order_product values
  2  ('ORD07','P07',7900);

1 row created.
```

Figure 39 Inserting values in Order_product table



```
SQL> select * from order_product;

ORDER_ID                                           PRODUCT_ID                                         UNIT_PRICE
-------------------------------------------------- -------------------------------------------------- ----------
ORD01                                              P01                                                     12800
ORD02                                              P02                                                      9500
ORD03                                              P03                                                      2000
ORD04                                              P04                                                      2500
ORD05                                              P05                                                      3500
ORD06                                              P06                                                      7000
ORD07                                              P07                                                      7900

7 rows selected.
```

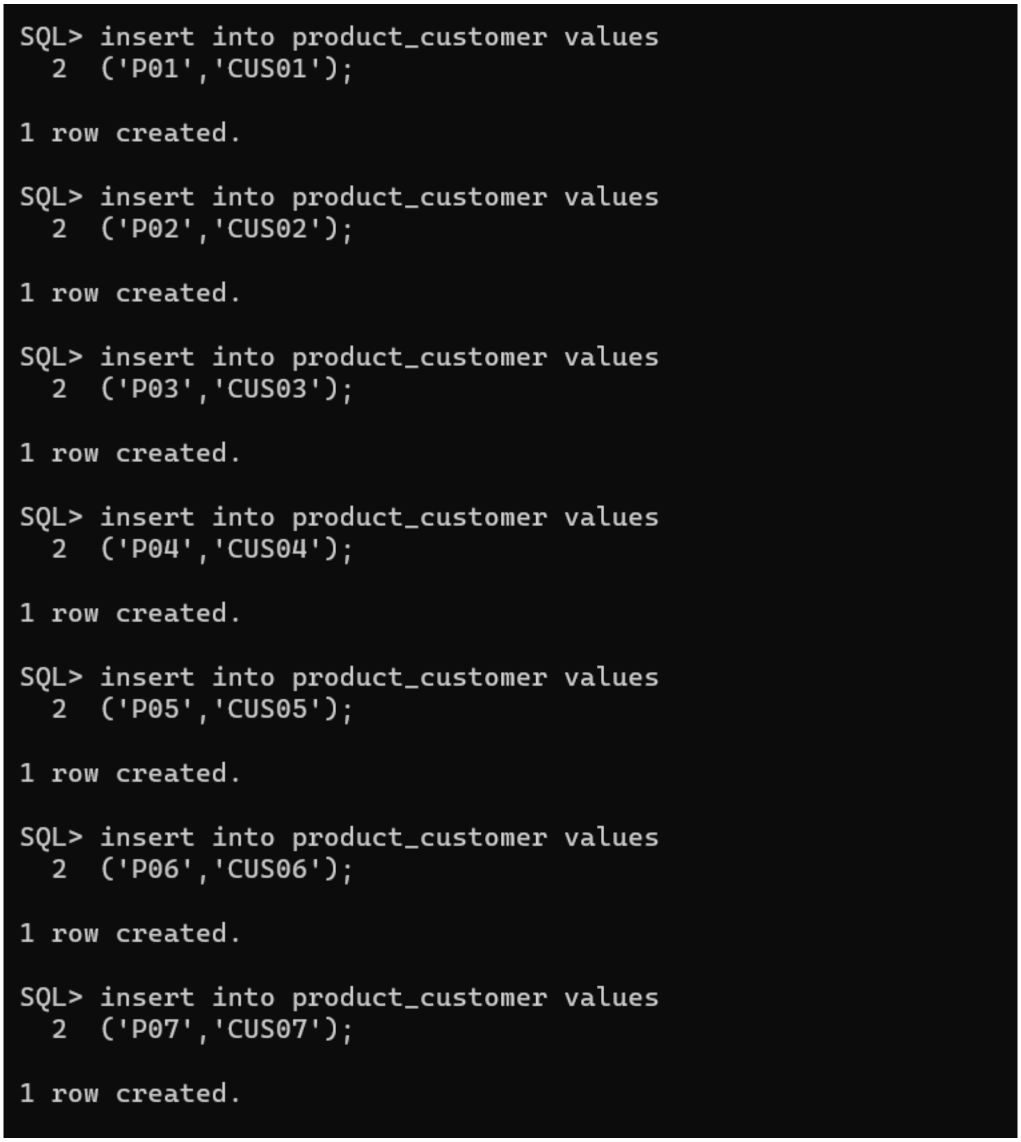
Figure 40 Checking values of Order_product table

- **Inserting values in Product_customer table and checking**
  Syntax

insert into product_customer values

('P01','CUS01');

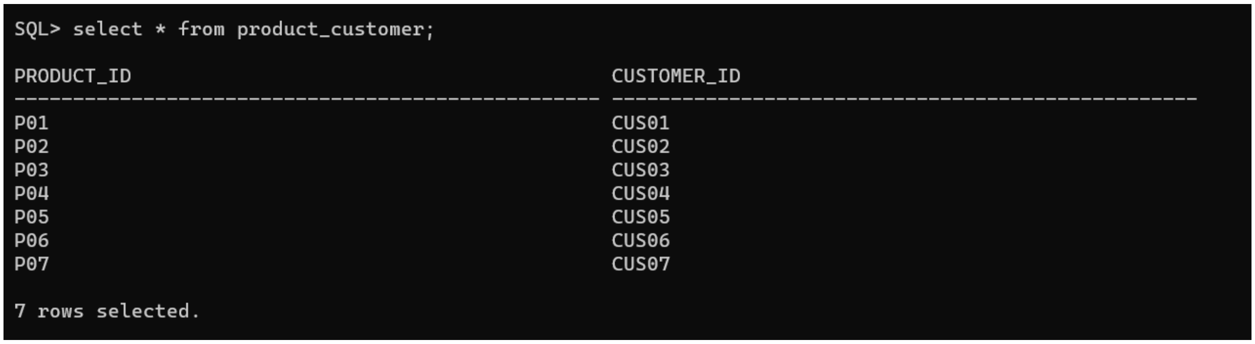
```
insert into product_customer values
 ('P02','CUS02');
insert into product_customer values
('P03','CUS03');
insert into product_customer values
('P04','CUS04');
insert into product_customer values
 ('P05','CUS05');
insert into product_customer values
 ('P06','CUS06');
insert into product_customer values
 ('P07','CUS07');
```



```
SQL> insert into product_customer values
  2  ('P01','CUS01');

1 row created.

SQL> insert into product_customer values
  2  ('P02','CUS02');

1 row created.

SQL> insert into product_customer values
  2  ('P03','CUS03');

1 row created.

SQL> insert into product_customer values
  2  ('P04','CUS04');

1 row created.

SQL> insert into product_customer values
  2  ('P05','CUS05');

1 row created.

SQL> insert into product_customer values
  2  ('P06','CUS06');

1 row created.

SQL> insert into product_customer values
  2  ('P07','CUS07');

1 row created.
```

Figure 41 Inserting value in Product_customer table

select * from product_customer

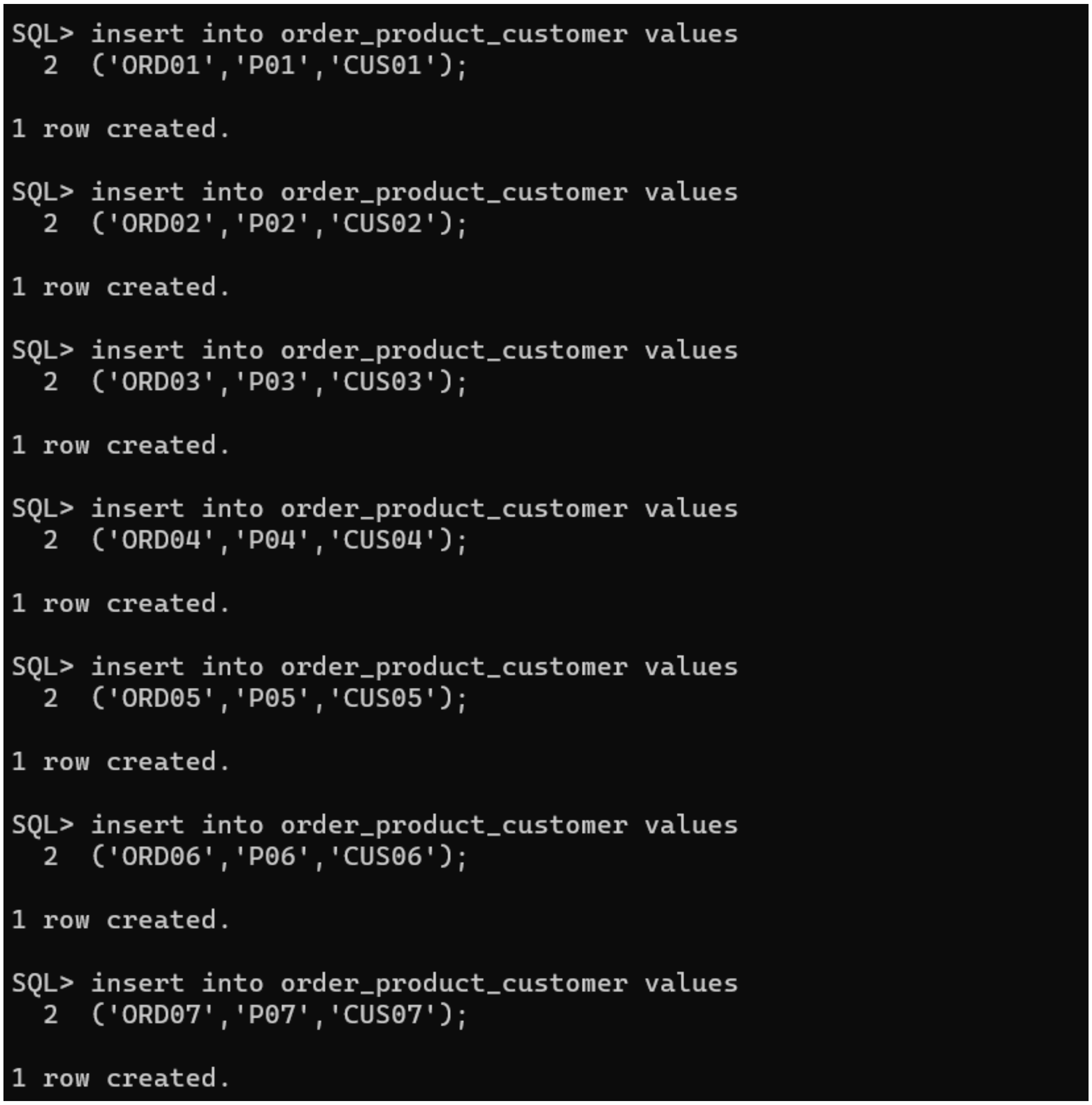


```
SQL> select * from product_customer;

PRODUCT_ID                                         CUSTOMER_ID
-------------------------------------------------- --------------------------------------------------
P01                                                CUS01
P02                                                CUS02
P03                                                CUS03
P04                                                CUS04
P05                                                CUS05
P06                                                CUS06
P07                                                CUS07

7 rows selected.
```

Figure 42 Checking the values of Product_customer table

- **Inserting value in Order_product_customer table**

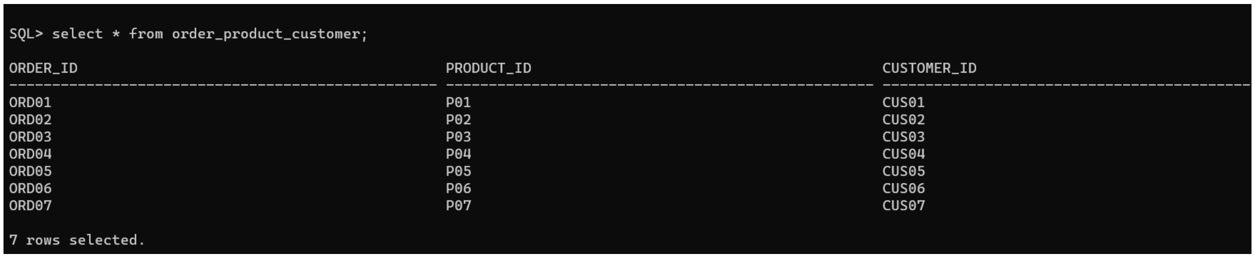
```
insert into order_product_customer values
 ('ORD01','P01','CUS01');
insert into order_product_customer values
 ('ORD02','P02','CUS02');
insert into order_product_customer values
 ('ORD03','P03','CUS03');
insert into order_product_customer values
 ('ORD04','P04','CUS04');
insert into order_product_customer values
 ('ORD05','P05','CUS05');
insert into order_product_customer values
 ('ORD06','P06','CUS06');
insert into order_product_customer values
 ('ORD07','P07','CUS07');
```



```
SQL> insert into order_product_customer values
  2  ('ORD01','P01','CUS01');

1 row created.

SQL> insert into order_product_customer values
  2  ('ORD02','P02','CUS02');

1 row created.

SQL> insert into order_product_customer values
  2  ('ORD03','P03','CUS03');

1 row created.

SQL> insert into order_product_customer values
  2  ('ORD04','P04','CUS04');

1 row created.

SQL> insert into order_product_customer values
  2  ('ORD05','P05','CUS05');

1 row created.

SQL> insert into order_product_customer values
  2  ('ORD06','P06','CUS06');

1 row created.

SQL> insert into order_product_customer values
  2  ('ORD07','P07','CUS07');

1 row created.
```

Figure 43 Inserting values in Order_product_customer table

select * from order_product_customer;



```
SQL> select * from order_product_customer;

ORDER_ID                                           PRODUCT_ID                                         CUSTOMER_ID
-------------------------------------------------- -------------------------------------------------- --------------------------------------------------
ORD01                                              P01                                                CUS01
ORD02                                              P02                                                CUS02
ORD03                                              P03                                                CUS03
ORD04                                              P04                                                CUS04
ORD05                                              P05                                                CUS05
ORD06                                              P06                                                CUS06
ORD07                                              P07                                                CUS07

7 rows selected.
```

Figure 44 Checking the values

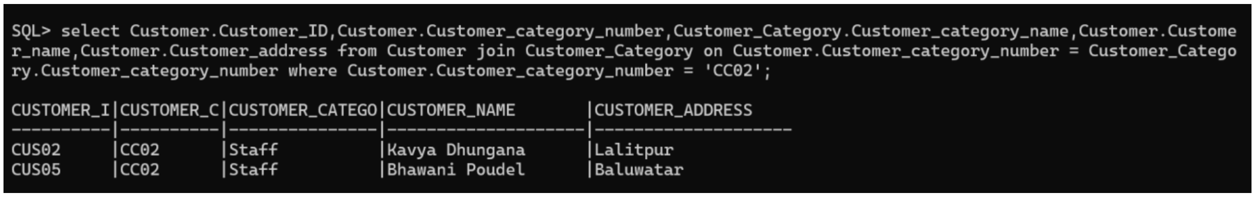
6. Querying

6.1. Information Query

- **List all the customers that are also staff of the company.**

Syntax

select Customer.Customer_ID,Customer.Customer_category_number,Customer_Category.Customer_category_name,Customer.Customer_name,Customer.Customer_address from Customer join Customer_Category on Customer.Customer_category_number = Customer_Category.Customer_category_number where Customer.Customer_category_number = 'CC02';



```
SQL> select Customer.Customer_ID,Customer.Customer_category_number,Customer_Category.Customer_category_name,Customer.Custome
r_name,Customer.Customer_address from Customer join Customer_Category on Customer.Customer_category_number = Customer_Catego
ry.Customer_category_number where Customer.Customer_category_number = 'CC02';

CUSTOMER_I|CUSTOMER_C|CUSTOMER_CATEGO|CUSTOMER_NAME       |CUSTOMER_ADDRESS
----------|----------|---------------|--------------------|--------------------
CUS02     |CC02      |Staff          |Kavya Dhungana      |Lalitpur
CUS05     |CC02      |Staff          |Bhawani Poudel      |Baluwatar
```

Figure 45 Listing the customers that are also staff of the company

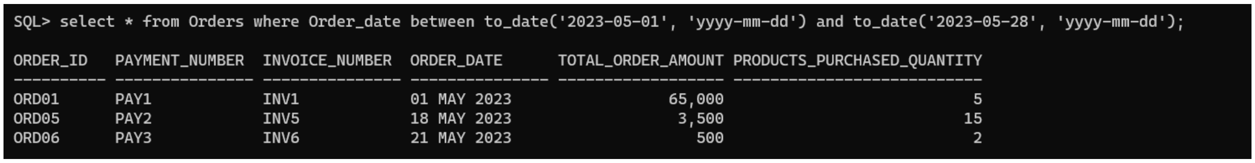
Explanation:

All the customers details is selected and the customer_category column is used to join the table to customer_category table and the information of the customer who is also staff is displayed.

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

Syntax

select * from Orders where Order_date between to_date('2023-05-01', 'yyyy-mm-dd') and to_date('2023-05-28', 'yyyy-mm-dd');



```
SQL> select * from Orders where Order_date between to_date('2023-05-01', 'yyyy-mm-dd') and to_date('2023-05-28', 'yyyy-mm-dd');

ORDER_ID   PAYMENT_NUMBER  INVOICE_NUMBER  ORDER_DATE      TOTAL_ORDER_AMOUNT PRODUCTS_PURCHASED_QUANTITY
---------- --------------- --------------- --------------- ------------------ ---------------------------
ORD01      PAY1            INV1            01 MAY 2023                 65,000                           5
ORD05      PAY2            INV5            18 MAY 2023                  3,500                          15
ORD06      PAY3            INV6            21 MAY 2023                    500                           2
```

Figure 46 Listing orders between the particular dates

Explanation:

The Order date between 2023-05-01 and 2023-05-28 is selected and displayed

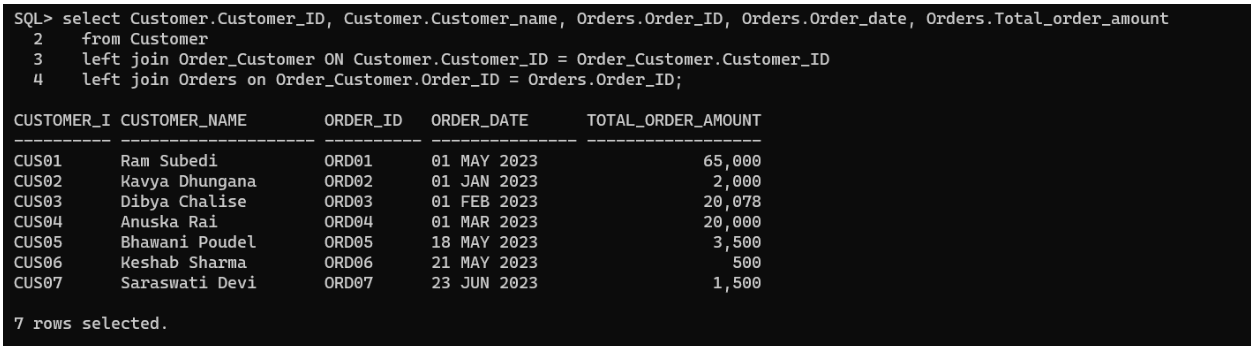
- **List all the customers with their order details and also the customers who have not ordered any products yet.**
  <u>Syntax</u>

select Customer.Customer_ID, Customer.Customer_name, Orders.Order_ID, Orders.Order_date, Orders.Total_order_amount

from Customer

left join Order_Customer ON Customer.Customer_ID = Order_Customer.Customer_ID

left join Orders on Order_Customer.Order_ID = Orders.Order_ID;



```
SQL> select Customer.Customer_ID, Customer.Customer_name, Orders.Order_ID, Orders.Order_date, Orders.Total_order_amount
     from Customer
     left join Order_Customer ON Customer.Customer_ID = Order_Customer.Customer_ID
     left join Orders on Order_Customer.Order_ID = Orders.Order_ID;

CUSTOMER_I CUSTOMER_NAME        ORDER_ID   ORDER_DATE      TOTAL_ORDER_AMOUNT
---------- -------------------- ---------- --------------- ------------------
CUS01      Ram Subedi           ORD01      01 MAY 2023                 65,000
CUS02      Kavya Dhungana       ORD02      01 JAN 2023                  2,000
CUS03      Dibya Chalise        ORD03      01 FEB 2023                 20,078
CUS04      Anuska Rai           ORD04      01 MAR 2023                 20,000
CUS05      Bhawani Poudel       ORD05      18 MAY 2023                  3,500
CUS06      Keshab Sharma        ORD06      21 MAY 2023                    500
CUS07      Saraswati Devi       ORD07      23 JUN 2023                  1,500

7 rows selected.
```

Figure 47 Listing customer order details

Explanation:

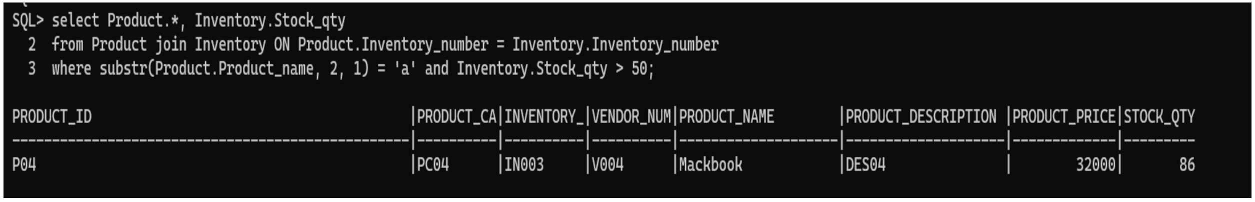
The customer ID, customer name from customer table and order ID, order date, Total order amount from order table is selected. Order_customer is joined to the left on customer ID where Customer ID is common and Orders table is joined to the left of the same Order_Customer table. Everything is displayed at last.

- **List all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.**
  <u>Syntax</u>

select Product.*, Inventory.Stock_qty

from Product join Inventory ON Product.Inventory_number = Inventory.Inventory_number

where substr(Product.Product_name, 2, 1) = 'a' and Inventory.Stock_qty > 50;



```
SQL> select Product.*, Inventory.Stock_qty
     from Product join Inventory ON Product.Inventory_number = Inventory.Inventory_number
     where substr(Product.Product_name, 2, 1) = 'a' and Inventory.Stock_qty > 50;

PRODUCT_ID                                        |PRODUCT_CA|INVENTORY_|VENDOR_NUM|PRODUCT_NAME        |PRODUCT_DESCRIPTION |PRODUCT_PRICE|STOCK_QTY
--------------------------------------------------|----------|----------|----------|--------------------|--------------------|-------------|---------
P04                                               |PC04      |IN003     |V004      |Mackbook            |DES04               |        32000|       86
```

Figure 48 Listing products with second latter 'a'

Explanation:

All the columns from the product table along with the stock quantity from inventory is selected and the Product and Inventory tables are joined. The product details of the products having ‘a’ in their product name and stock quantity more than 50 is displayed.

- **Find out the customer who has ordered recently.**

Syntax

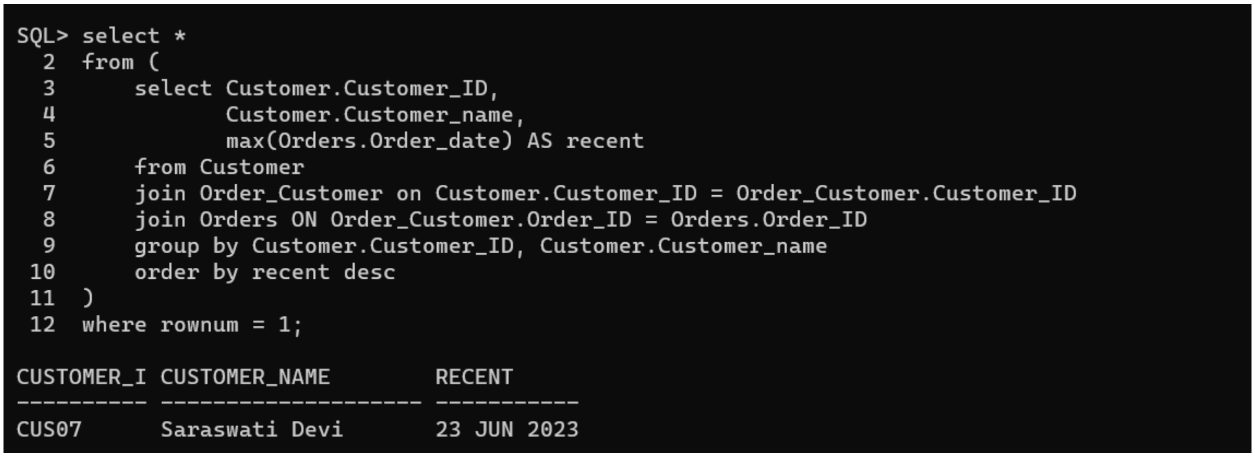
select *

from (

select Customer.Customer_ID, Customer.Customer_name, max(Orders.Order_date) as recent from Customer join Order_Customer on Customer.Customer_ID =   Order_Customer.Customer_ID join Orders ON Order_Customer.Order_ID = Orders.Order_ID group by Customer.Customer_ID, Customer.Customer_name order by recent desc

)

where rownum = 1;



```
SQL> select *
     from (
         select Customer.Customer_ID,
                Customer.Customer_name,
                max(Orders.Order_date) AS recent
         from Customer
         join Order_Customer on Customer.Customer_ID = Order_Customer.Customer_ID
         join Orders ON Order_Customer.Order_ID = Orders.Order_ID
         group by Customer.Customer_ID, Customer.Customer_name
         order by recent desc
     )
     where rownum = 1;

CUSTOMER_I CUSTOMER_NAME        RECENT
---------- -------------------- -----------
CUS07      Saraswati Devi       23 JUN 2023
```

Figure 49 Finding Customers who has ordered recently

Explanation:

The customer ID, customer name is selected of the customers who has ordered recently by joining the customer table to order table and retrieving its date.

6.2. Transaction Query

- **Show the total revenue of the company for each month.**
  <u>Syntax</u>

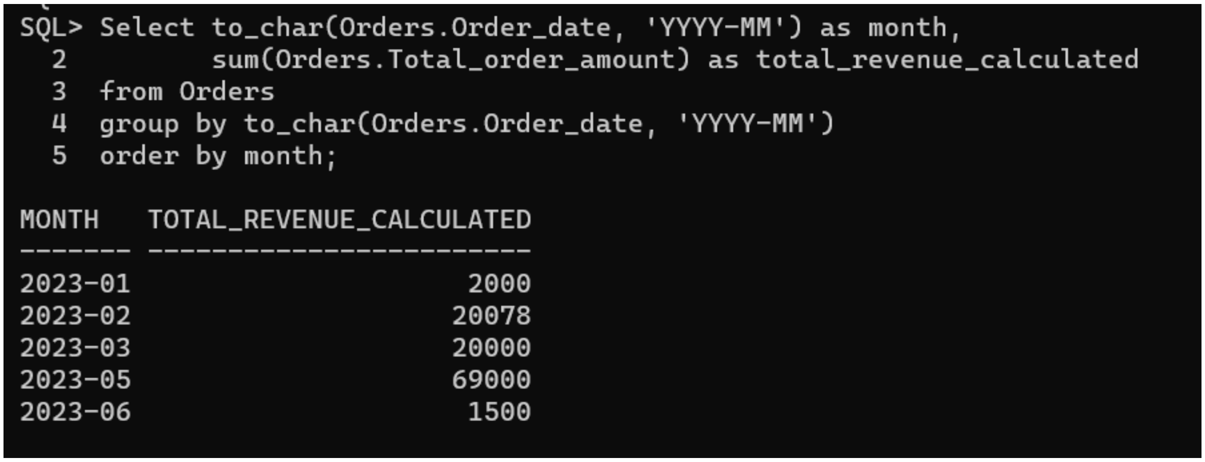
  Select to_char(Orders.Order_date, 'YYYY-MM') as month,

  sum(Orders.Total_order_amount) as total_revenue_calculated

  from Orders

  group by to_char(Orders.Order_date, 'YYYY-MM')

  order by month;



```
SQL> Select to_char(Orders.Order_date, 'YYYY-MM') as month,
            sum(Orders.Total_order_amount) as total_revenue_calculated
     from Orders
     group by to_char(Orders.Order_date, 'YYYY-MM')
     order by month;

MONTH   TOTAL_REVENUE_CALCULATED
------- ------------------------
2023-01                     2000
2023-02                    20078
2023-03                    20000
2023-05                    69000
2023-06                     1500
```

Figure 50 Showing total revenue of company for each month

Explanation:

The dates are extracted from orders table and the total order amount of each month is summed to get the total revenue of each month. The result is displayed

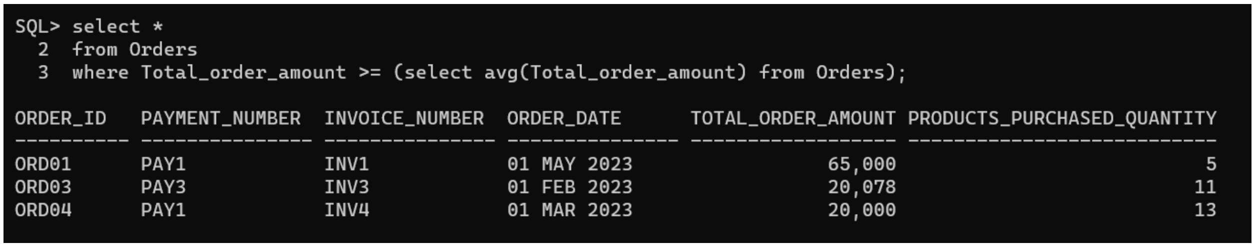
- **Find those orders that are equal or higher than the average order total value.**

  <u>Syntax</u>

  select *

  from Orders

  where Total_order_amount >= (select avg(Total_order_amount) from Orders);



```
SQL> select *
     from Orders
     where Total_order_amount >= (select avg(Total_order_amount) from Orders);

ORDER_ID   PAYMENT_NUMBER  INVOICE_NUMBER  ORDER_DATE      TOTAL_ORDER_AMOUNT PRODUCTS_PURCHASED_QUANTITY
---------- --------------- --------------- --------------- ------------------ ---------------------------
ORD01      PAY1            INV1            01 MAY 2023                 65,000                           5
ORD03      PAY3            INV3            01 FEB 2023                 20,078                          11
ORD04      PAY1            INV4            01 MAR 2023                 20,000                          13
```

Figure 51 Showing orders higher than average values

Explanation:

The orders that are equal or higher than the average order total value is displayed along with order id, payment number, invoice number, order date and products purchased quantity.

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

Syntax

select

Vendor.Vendor_number,

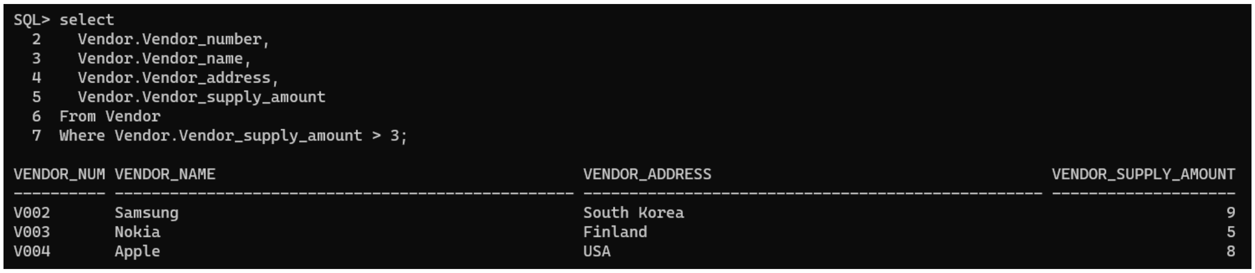
Vendor.Vendor_name,

Vendor.Vendor_address,

Vendor.Vendor_supply_amount

From Vendor

Where Vendor.Vendor_supply_amount > 3;



```
SQL> select
       Vendor.Vendor_number,
       Vendor.Vendor_name,
       Vendor.Vendor_address,
       Vendor.Vendor_supply_amount
     From Vendor
     Where Vendor.Vendor_supply_amount > 3;

VENDOR_NUM VENDOR_NAME                                         VENDOR_ADDRESS                                      VENDOR_SUPPLY_AMOUNT
---------- --------------------------------------------------- --------------------------------------------------- --------------------
V002       Samsung                                             South Korea                                                            9
V003       Nokia                                               Finland                                                                5
V004       Apple                                               USA                                                                    8
```

Figure 52 Details of vendor supplying more than 3 products

Explanation:

The details of vendor supplying more than three products is displayed.

- **Show the top 3 product details that have been ordered the most.**
  <u>Syntax</u>

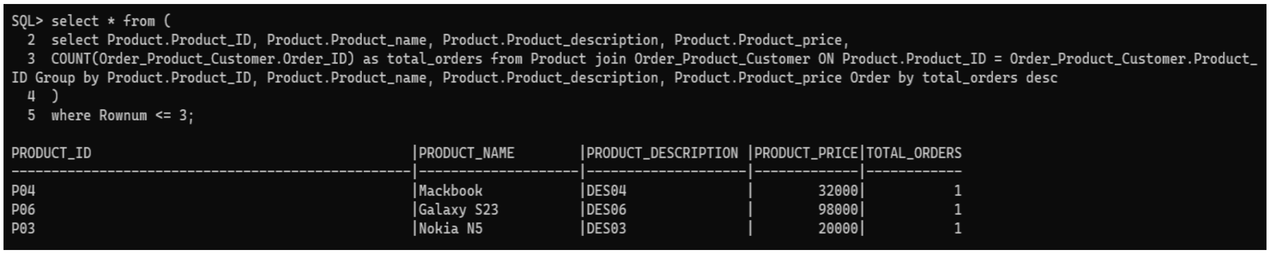
select * from (

select Product.Product_ID, Product.Product_name, Product.Product_description, Product.Product_price,

COUNT(Order_Product_Customer.Order_ID) as total_orders from Product join Order_Product_Customer ON Product.Product_ID = Order_Product_Customer.Product_ID Group by Product.Product_ID, Product.Product_name, Product.Product_description, Product.Product_price Order by total_orders desc

)

where Rownum <= 3;



```
SQL> select * from (
  2  select Product.Product_ID, Product.Product_name, Product.Product_description, Product.Product_price,
  3  COUNT(Order_Product_Customer.Order_ID) as total_orders from Product join Order_Product_Customer ON Product.Product_ID = Order_Product_Customer.Product_
ID Group by Product.Product_ID, Product.Product_name, Product.Product_description, Product.Product_price Order by total_orders desc
  4  )
  5  where Rownum <= 3;

PRODUCT_ID                                        |PRODUCT_NAME        |PRODUCT_DESCRIPTION |PRODUCT_PRICE|TOTAL_ORDERS
--------------------------------------------------|--------------------|--------------------|-------------|------------
P04                                               |Mackbook            |DES04               |        32000|           1
P06                                               |Galaxy S23          |DES06               |        98000|           1
P03                                               |Nokia N5            |DES03               |        20000|           1
```

Figure 53 Details of the 3 products that has been ordered most

Explanation:

The product details of the top three products ordered the most are displayed.

- **Find out the customer who has ordered the most in August with his/her total spending on that month.**
  <u>Syntax</u>

select

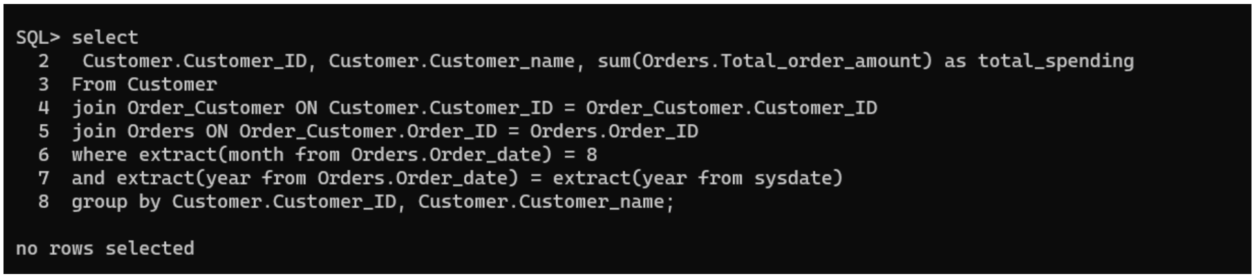
Customer.Customer_ID, Customer.Customer_name, sum(Orders.Total_order_amount) as total_spending From Customer

join Order_Customer on Customer.Customer_ID = Order_Customer.Customer_ID join Orders ON Order_Customer.Order_ID = Orders.Order_ID

where extract(month from Orders.Order_date) = 8

and extract(year from Orders.Order_date) = extract(year from sysdate)

group by Customer.Customer_ID, Customer.Customer_name;



```
SQL> select
  2   Customer.Customer_ID, Customer.Customer_name, sum(Orders.Total_order_amount) as total_spending
  3  From Customer
  4  join Order_Customer ON Customer.Customer_ID = Order_Customer.Customer_ID
  5  join Orders ON Order_Customer.Order_ID = Orders.Order_ID
  6  where extract(month from Orders.Order_date) = 8
  7  and extract(year from Orders.Order_date) = extract(year from sysdate)
  8  group by Customer.Customer_ID, Customer.Customer_name;

no rows selected
```

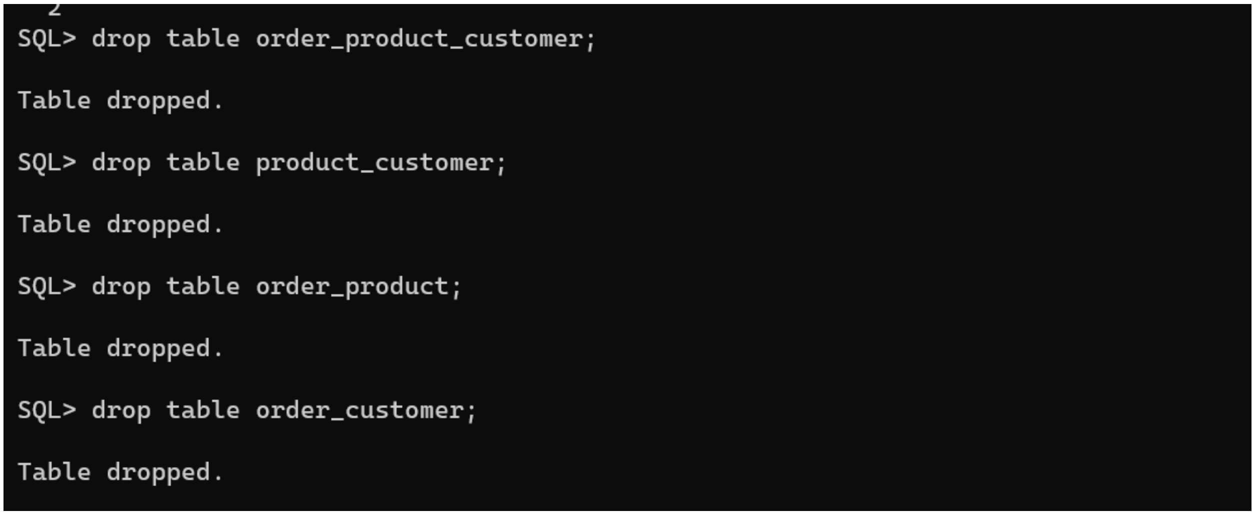
Figure 54 Details of customer who has ordered in August

Explanation:

All the customer’s details is extracted and the total order amount from orders table is summed. The Order_Customer table is joined on Customer and the Orders table is joined on Order_Customer table. Then the month from order date in orders tables is checked whether it is august or not and the customer who has ordered the most in August with his/her total spending on that month is displayed. Since the database has no customers who has ordered in August, no rows were printed.

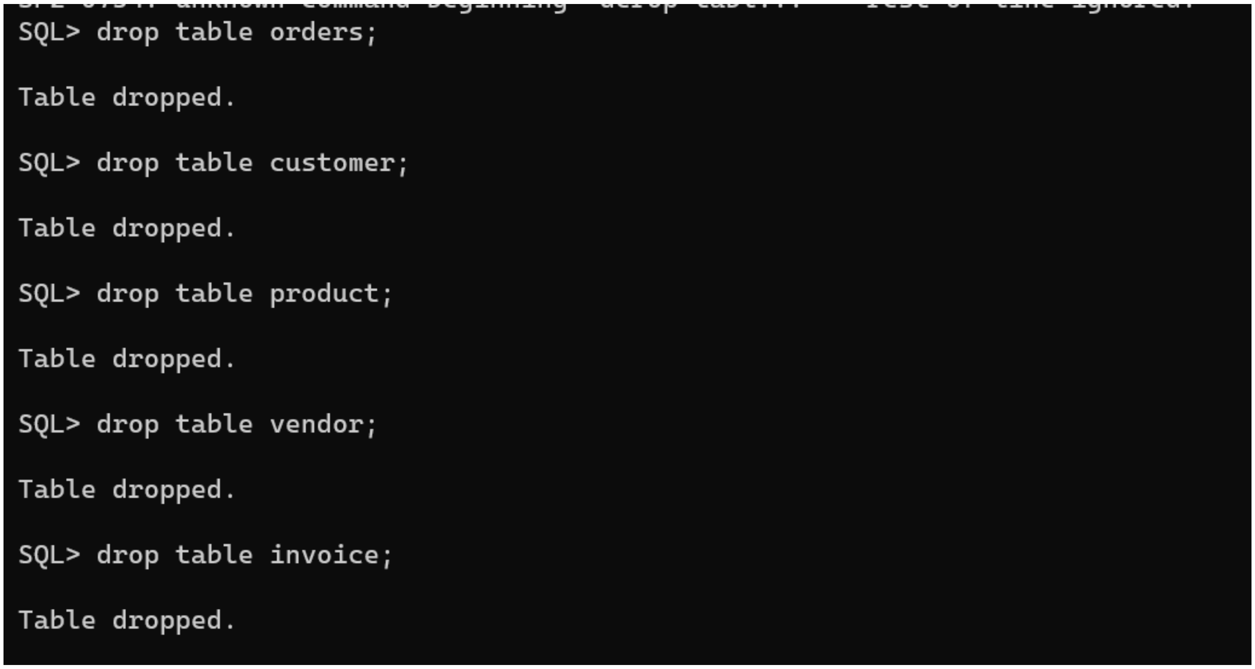
7. Dropping tables

After everything, the tables are dropped one by one. The screenshots of the table drop are provided below.



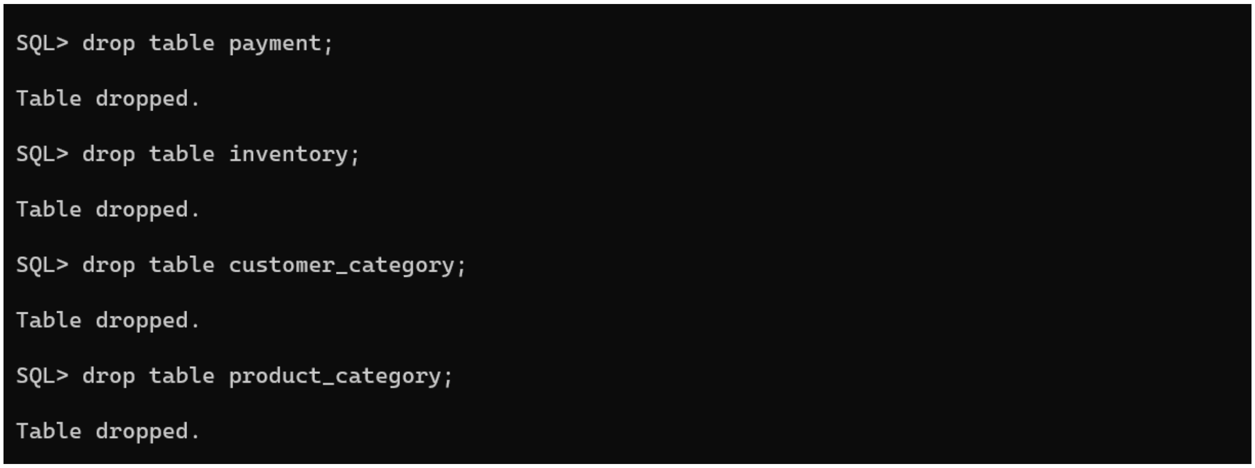
```
  2
SQL> drop table order_product_customer;

Table dropped.

SQL> drop table product_customer;

Table dropped.

SQL> drop table order_product;

Table dropped.

SQL> drop table order_customer;

Table dropped.
```

Figure 55 Dropping tables I



```
SP2-0734: unknown command beginning "derop tabl..." - rest of line ignored.
SQL> drop table orders;

Table dropped.

SQL> drop table customer;

Table dropped.

SQL> drop table product;

Table dropped.

SQL> drop table vendor;

Table dropped.

SQL> drop table invoice;

Table dropped.
```

Figure 56 Dropping tables II



```
SQL> drop table payment;

Table dropped.

SQL> drop table inventory;

Table dropped.

SQL> drop table customer_category;

Table dropped.

SQL> drop table product_category;

Table dropped.
```

Figure 57 Dropping tables III

8. Critical Evaluation

8.1. Critical evaluation of Module

A database is essential for storing a lot of data. In any sector, data protection and analysis are crucial. Organizations in the real world utilize it to improve operational efficiency. Data management is crucial in many sectors, including banks, workplaces, schools, and universities. Personal health information about patients is encrypted and stored in hospitals using this technology. Banks use it to maintain transaction records. Report cards and student information are stored on it at schools and institutions. Improving company processes is crucial in offices. Databases are also useful for concurrency control, data integration and migration, and database backup and recovery.

8.2. Critical Assessment of Coursework

In summary, all of the concepts related to databases, queries, and normalization were understood after completion of this coursework. An example was provided within the framework of the Gadget Emporium firm. For this firm, a database has to be made. There were seven business guidelines that needed to be adhered to during the normalization process. As database designers, we had to assist Mr. John, the store's owner, in developing and putting into place a robust database system that would support his enterprise. All of the items that consumers have ordered must be tracked via the system that was developed. The purpose of normalization was to eliminate redundant data. Tables were built and the database was developed in SQL after redundant data was eliminated. The information was stored in tables. Then the queries were carried out to demonstrate the outcome of the database created.

9. Conclusion

In a nutshell, this coursework was a great learning process regarding database, its importance and its power. The fact that database helps to maintain data integrity and prevent any loss or misuse of data was also clear. Similarly, normalization was the main highlight of this coursework. The entire coursework was related to use of normalization to remove data redundancy. We had to normalize till 3NF and implement the tables in SQL. Information and transaction queries were also carried out in order to check the normalization.

Various kinds of difficulties were encountered during the completion of this coursework. The concept and implementation of normalization was difficult to understand at first. Also, querying and creating tables was a bit confusing. Difficulties were also encountered during the creation of spool and dump files. However, all these difficulties were resolved with the help of our tutors and lecturers. They helped us to clear our confusions and work accordingly to successfully complete our coursework. They also provided us with guidance and helped us to deal with all the possible difficulties that could arise in the future. They even provided us with revisions on normalization to clear our doubts.

10. References

Anon., 2022. *Byjus.* [Online]
Available at: https://byjus.com/gate/first-normal-form-in-dbms/
[Accessed 4 January 2024].

Anon., 2023. *Academic Accelerator.* [Online]
Available at: https://academic-accelerator.com/encyclopedia/unnormalized-form
[Accessed 12 January 2024].

Arya, S., 2022. *Scaler.* [Online]
Available at: https://www.scaler.com/topics/entity-in-dbms/
[Accessed 1 January 2024].

Awati, R., 2022. *Tech Target.* [Online]
Available at: https://www.techtarget.com/whatis/definition/attribute
[Accessed 1 January 2024].

Karthik, 2023. *Byjus.* [Online]
Available at: https://byjus.com/gate/third-normal-form-in-dbms/
[Accessed 13 January 2024].

Morris, S., 2022. *Coresignal.* [Online]
Available at: https://coresignal.com/blog/data-normalization/
[Accessed 12 January 2024].

S, R. A., 2022. *Simplile arn.* [Online]
Available at: https://www.simplilearn.com/tutorials/sql-tutorial/what-is-normalization-in-sql
[Accessed 1 January 2024].