LONDON
METROPOLITAN
UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: Sikum Hangma Madi

London Met ID: 22085627

Assignment Submission Date: Monday, January 15, 2024

Word Count: 4047

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

1.1. Introduction of the Business

Gadget Emporium is an online marketplace founded by Mr. John F. Kennedy in 2017, who is a passionate entrepreneur and electronics enthusiast. The company is focused on offering a wide range of electronic gadgets and accessories to businesses as well as private customers. The company takes great satisfaction in being ahead of technology trends and making sure that clients have access to the newest and most advanced devices available. Gadget Emporium aims to be a go to go store for every electronics, whether it's computers, smartphones, smart home appliances, or smart electronic appliances.

The foundation of Gadget Emporium's concept is its dedication to building a tech enthusiast community. The organisation regularly arranges online gatherings, in-person product launches, and interactive discussion boards where people can exchange insights and talk about the newest developments in technology. Gadget Emporium stands out for its dedication to developing a sense of community among its customers, offering an engaging and cooperative setting for those who share a love for remaining at the top of electronic advancements.

Gadget Emporium is currently an online retailer that offers customers an easy-to-use interface for browsing, selecting, and purchasing gadgets. The company receives the products in its inventory from reliable vendors and manufacturers, guaranteeing the authenticity and quality of the goods provided.

1.2. Aims and Objectives

The main aim of Gadget Emporium is to be the top online source for the newest and most advanced electronic devices and accessories, serving both corporate and individual customers. With a focus on optimising the browsing, selection, and purchasing procedures for customers, Gadget Emporium is dedicated to provide an easy-to-use online shopping experience.

- Creative and advanced leadership
- Make it simple for customers to browse, purchase the products.
- User Friendly interface.
- Guarantee authenticity and quality of the products.
- Focus on both private and business customers.

1.3. Current Business activities and Operation

As a online retailer of electrical devices, Gadget Emporium takes part in a variety of activities to keep up its standing. In order to guarantee authenticity and client happiness, the business concentrates on obtaining premium products from reliable suppliers. Gadget Emporium carries out smooth transactions, effective logistics, and prompt customer service through its user-friendly e-commerce platform. In the continuously changing industry of electronics, Gadget Emporium aims to offer both businesses and individual consumers a comprehensive user experience through the system.

1.4. Business Rules

- Each product must be of only one category and each category can have one or many products.
- Customers can browse and purchase one or many electronic gadgets online
- The customers are categorized as Regular(R), Staff(S) and VIP(V) with their respective discount rates of 0%, 5% and 10%. Discount Rates are identified by the type of Customer Category a customer falls under. The customer's address information are to be stored properly for better shipping process.
- Each order must record details of the product purchased, including its quantities, unit prices and total order amount. An order can consist of many products and any one type of products can be included in multiple orders. So, there is many to many relationship between order and product.

- Each product should be associated with a single vendor and each vendor can supply one or many products. Therefore, there is one to many relationship between product and vendor.
- Products must be tracked according to it's real-time inventory details that must include stock quantity and availability in order to prevent overselling.
- The system must be open with various payment options like cash on delivery, credit/debit card, apple pay or pay pal. Each order detail must have one payment option.
- An invoice is generated after the customer's order confirmation which includes details of the customer, order and payment. The cash on delivery payment option offers two choices for the customer, one for paying half the price of total order amount and another for just paying the shipping fee in order to prevent order cancellation after the shipment. Hence, there is one to one relationship between payment and invoice.

1.5. Identification of Entities and Attributes

In the initial phase of identification of entities and attributes of Gadget's Emporium design, five primary key tables have been identified. They are Category, Vendor, Product, Order and Customer table, better clarification and clear concept on the attributes of these entities are explained on the tables below:-

1.5.1. Category Entity

These entities manages the classification and distribution of different product which has been supplied by the vendors, in order to prevent misplace and data loss insuring organized inventory of the system.

Attributes	**Data Type**	**Constraints**	**Description**
CategoryID	INT	PRIMARY KEY, NOT NULL	Unique Identifier for each category
CategoryName	VARCHAR(50)	NOT NLL	Names for each categories

Table 1 Category Entity

1.5.2. Vendor Entity

These entities contains information of vendors such as VendorName and its contact details.

Attributes	**Data Type**	**Constraints**	**Description**
VendorID	INT	PRIMARYKEY, NOT NULL	Unique Identifier for each vendor
VendorName	VARCHAR(50)	NOT NULL	Names of the vendors
PhoneNumber	INT	NOT NULL	Phone Numbers of Vendors
EmailAddress	VARCHAR(30)	NOT NUL	Email of each Vendors

Table 2 Vendor Entity

1.5.3. Product Entity

One of the most important entity, Product contains primary information of products, its description along with real time stock level and availability status.

Attributes	**Data type**	**Constraints**	**Description**
ProductID	INT	PRIMARY KEY, NOT NULL	Unique Identifier for each Product
ProductName	VARCHAR(50)	NOT NULL	Names for the products
ModelNo	INT	NOT NULL	Model Numbers for each product
ProductDescription	VARCHAR(100)	NOT NULL	Description of products
UnitPrice$	INT	NOT NULL	Price of the products

StockLevel	INT	NOT NULL	Stock Levels of the product
AvailabilityStatus	VARCHAR(50)	NOT NULL	States the availability status of the product
LastUpdated	DATE	NOT NULL	Date when the inventory details were updated
CategoryID	INT	FOREIGN KEY NOT NULL	References Category table in Product table
VendorID	INT	FOREIGN KEY NOT NULL	References Vendor Table in Product Table
OrderID	INT	FOREIGN KEY	References Order table in Product Table

Table 3 Product Entity

1.5.4. Order Entity

The order table acts as the junction for storing order relates details, tracking information, dates and product information.

Attributes	**Data types**	**Constraint**	**Description**
OrderID	INT	PRIMARY KEY NOT NULL	Uniquely Identifies Order
OrderDate	DATE	NOT NULL	Specifies the order date
OrderQuantity	INT	NOT NULL	Stores the quantity of the order
TotalOrderAmount	VARCHAR(10)	NOT NULL	Stores the total order amount of the customer
InvoiceNO	INT	NOT NULL	Identifies Invoice Number
InvoiceDate	VARCHAR(10)	NOT NULL	Specifies the date the invoice was generated
PaymentNO	INT	NOT NULL	Identifies Payment
PaymentMethod	VARCHAR(50)	NOT NULL	Payment Method of the order

PaymentStatus	VARCHAR(10)	NOT NULL	Tracks the payment status of the order
ConfigurationNO	INT	NOT NULL	Confirms the Payment status by configuration number
ProductID	INT	FOREIGN KEY NOT NULL	References Product Table in Order table
CustomerID	INT	FOREIGN KEY NOT NULL	References Customer table in Order tablae

Table 4 Order Entity

1.5.5. Customer Entity

The customer table consists of all the data related to a customer including their name, number, address and category.

Attributes	**Data Types**	**Constraints**	**Description**
CustomerID	INT	PRIMARY KEY NOT NULL	Uniquely Identifies Customer
CustomerName	VARCHAR(50)	NOT NULL	Name of the customer
CustomerAddress	VARCHAR(50)	NOT NULL	Address of the customer
PhoneNumber	INT	NOT NULL	Phone number of the customer
EmailAddress	VARCHAR(50)	NOT NULL	Email Address of the customer
CustomerCategoryID	INT	NOT NULL	Identifies Customer Category
CustomerCategory	VARCHAR(50)	NOT NULL	Types of Category the customer fall under
Discount	VARCHAR(5)	NOT NULL	Stores the discount rate of that specific customer category

Table 5 Customer Entity

2. Initial ERD

The Entity-Relationship Diagram (ERD) is a fundamental component of Gadget Emporium's database architecture that acts as a roadmap for structuring important business components. Entities like "Product," "Category," "Vendor," "Customer," and "Order" are included in the ERD and are connected to create relationships that show how data moves through the system.

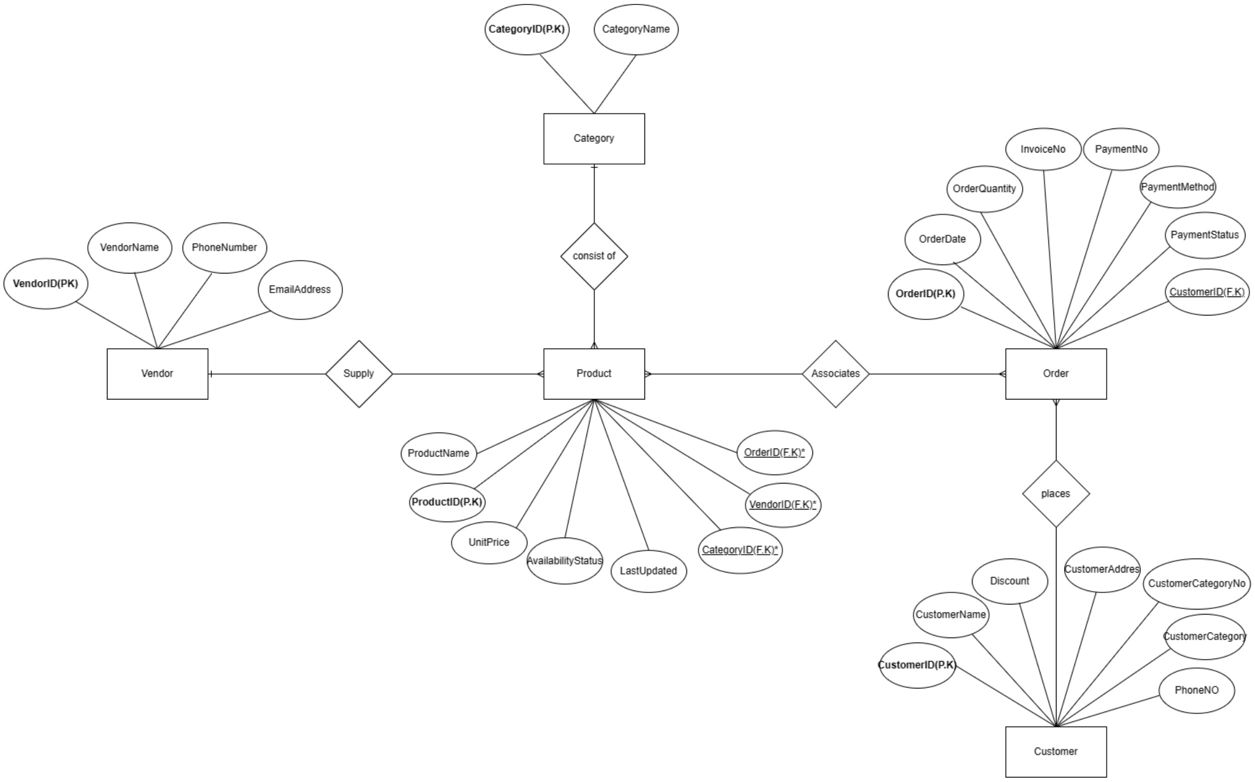


Figure 1 Initial ERD-Chen Model

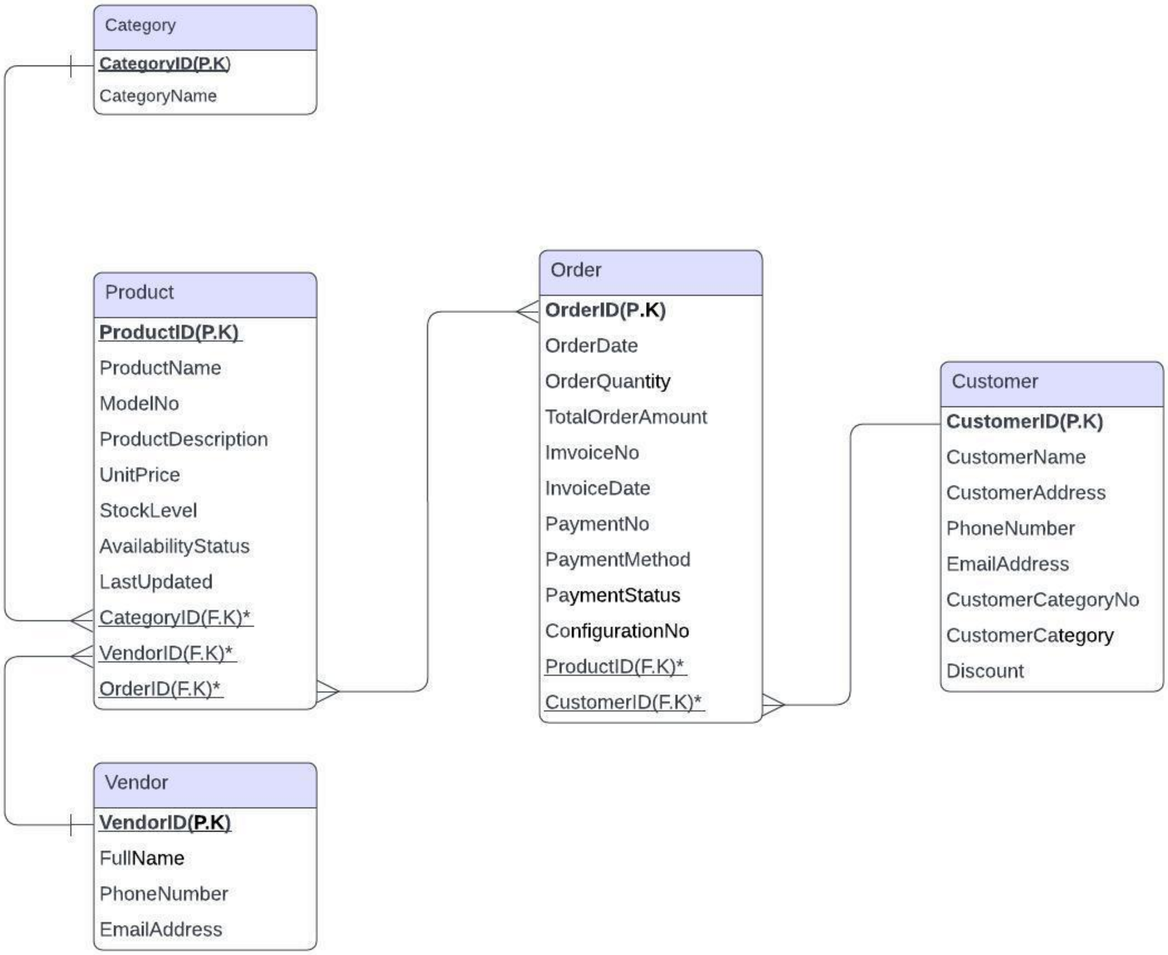


Figure 2 Initial ERD-Crow's foot Notation model

3. Normalization

Normalization in database is a designing approach for organizing data, reducing data redundancy and eliminating anomalies. There are 3 stages in the process of normalization they are, **1NF, 2NF** and **3NF**.

3.1. UNF

“Unnormalized for”(UNF) refers to the initial state of a database before any steps of normalization is applied. It is unstructured/denormalized stage where the database might contain redundancies and anomalies.

The UNF of Gadget Emporium is given below:-

Customer- (<u>CustomerID</u>(PK), CustomeName, CustomerAddress, PhoneNo, EmailAddress, CustomerCategoryID, CustomerCategory ,Discount.{OrderID, OrderDate, OrderQuantity, TotalOrderAmount, InvoiceNo, InvoiceDate, PaymentNo, PaymentMethod, PaymentStatus, ConfigurationNo {ProductID, ProductName, ModelNO, ProductDescription, UnitPrice, StockLevel, AvailabilityStatus, LastUpdated, VendorID, VendorName, PhoneNo, EmailAddress, CategoryID, CategoryName} })

Under the assumption:-

- The ‘Customer’ is an entity with primary key CustomerID, while there might be other foreign keys in these entity but for this initial stage, CustomerID serves as a single primary key in the ‘Customer’ table.
- It is assumed that Customer is Repeating Data because a single customer can place one or many orders with different attributes like OrderID, OrderDate and a customer can engage in multiple transactions, having ‘Order’ entity as its Repeating Groups, showing the purchasing activities of a customer.
- Product is Repeating Groups of Order. Since, An order can have multiple products and any one type of product might be included in multiple orders placed by various customers.

Here, two repeating groups were identified nested within each other resulting in data redundancy and anomalies. So, further normalization steps are listed below.

3.2. 1NF

A table is in 1NF if it contains an atomic value or repeating groups. So 1NF is all about eliminating the Repeating Groups from UNF.

Customer -1 (CustomerID(PK), CustomerName, CustomerAddress, PhoneNo, EmailAddress, CustomerCategoryID, CustomerCategory, Discount.)

- The 'Customer' table represents a single customers with information like customer's name, address, contacts and category stored in it. 'CustomerID' serves as the primary key of this table.

Customer_Order -1 (CustomerID*, OrderID*, OrderDate, OrderQuantity, TotalOrderAmount, InvoiceNumber, InvoiceDate, PaymentNo, PaymentMethod, PaymentStatus, ConfigurationNo)

- The 'Customer_Order' table is a junction table which represents the relationship between customers and their orders. 'CustomerID' and 'OrderID' are two primary keys of these table where 'CustomerID' is the foreign key referencing the 'Çustomer' table.

Customer_Order_Product -1 (CustomerID*, Order ID*, Product ID*, ProductName, ModelNO, ProductDescription, UnitPrice, StockLevel, AvailabilityStatus, LastUpdated, VendorID, VendorName, PhoneNo, EmailAddress, CategoryID, CategoryName)

- The 'Customer_Order_Product' table is a junction table that links customer, order and product together. The 'CustomerID', 'OrderID', 'ProductID' are three primary keys of these table where 'CustomerID' and 'OrderID' are foreign keys referencing the 'Customer_Order' table, establishing a proper hierarchical relationship.

3.3. 2NF

A table will be in 2NF if is in 1NF and all non-key attributes are fully functional dependent on the primary key.

Customer -1 (CustomerID(PK), CustomerName, CustomerAddress, PhoneNo, EmailAddress, CustomerCategoryID, CustomerCategory, Discount.)

From Customer -1 entity:-

Customer ID -> CustomerName, CustomerAddress, PhoneNo, EmailAddress, CustomerCategoryID, CustomerCategory, Discount

Under Assumption Customer -1 :-

- There are no partial dependencies as all non prime key fully functionally depends on the primary key CustomerID.

Customer_Order -1 (CustomerID*, OrderID*, OrderDate, OrderQuantity, TotalOrderAmount, InvoiceNumber, InvoiceDate, PaymentNo, PaymentMethod, PaymentStatus, ConfigurationNo)

Form Customer_Order -1 entity:-

Customer ID -> none

Order ID -> OrderDate, OrderQuantity, TotalOrderAmount,

Customer ID , Order ID -> PaymentNo, PaymentMethod, PaymentStatus, ConfigurationNo, InvoiceNumber, InvoiceDate

Under Assumption Customer _ Order -1 :-

- It is assumed that OrderDate, OrderQuantity and TotalOrderAmount attributes are fully dependent on Order ID(PK).

- It is assumed that PaymentNo, PaymentMethod, PaymentStatus, ConfigurationNo, InvoiceNo, InvoiceDate attributes are dependent on both Customer ID and Order ID(Primary key).
  - This is because each order that is made by the customer may result having unique payment and the payment's configuration details.
  - The dependency on Customer ID is important because they are related with a certain customer.
  - The dependency on Order ID is important because the information is specific to a certain order that was made by the customer.

Customer_Order_Product -1 (<u>CustomerID*, Order ID*, Product ID*,</u> ProductName, ModelNO, ProductDescription, UnitPrice, StockLevel, AvailabilityStatus, LastUpdated, VendorID, VendorName, PhoneNo, EmailAddress, CategoryID, CategoryName)

<u>Form Customer_ Order_ Product -1 entity:-</u>

Customer ID -> none

Order ID -> none

Product ID -> ProductName, Model NO, Product Description, Unit Price, InvenotryNO, Stock Level, Availability Status, Last Updated, Vendor ID*, Name, Phone, Enail, Category ID*, Category Name

Customer ID, Order ID-> none

Order ID, Product ID -> none

Customer ID , Product ID -> none

Customer ID , Order ID, Product ID -> none

Under Assumption Customer_ Order_ Product :-

1) There were partial dependencies because every non key attributes on the table Customer_ Order_ Product are fully dependent on ProductID so we form a separate entity called 'Product' where ProductID serves as the primary key .

Putting into 2NF, the table formed are:-

Customer 2 - (CustomerID(PK), CustomerName, CustomerAddress, PhoneNo, EmailAddress, CustomerCategoryID, CustomerCategory, Discount)

Customer_Order 2 – (CustomerID*, OrderID*, PaymentNo, PaymentMethod, PaymentStatus, ConfigurationNo, InvoiceNumber, InvoiceDate)

Order 2 – (OrderID(PK), OrderDate, OrderQuantity, TotalOrderAmount,)

Product 2 – (ProductID(PK) , ProductName, ModelNO, ProductDescription, UnitPrice, StockLevel, AvailabilityStatus, LastUpdated, VendorID, VendorName, PhoneNo, EmailAddress, CategoryID, CategoryName)

Customer_Order_ Product 2 – (CustomerID*, OrderID*, ProductID*)

3.4. 3NF

A table is in 3NF if it is in 2NF and if there are no transitive dependencies.

Customer 2 - (CustomerID(PK), CustomerName, CustomerAddress, PhoneNo, EmailAddress)

Form Customer 2 – entity:-

Customer ID -> Customer Name, Shipping Address, Billing Address, House No, Phone No, Email Address, CustomerCategoryID, CustomerCategory, Discount

Customer Name -> none

CustomerCategoryID -> CustomerCategory, Discount

Under the assumption :-

- There were transitive dependencies found as CustomerCategory and Discount a non prime attribute tend to depend on another non prime attribute i.e. CustomerCategoryID so we form a separate entity named ‘CustomerCategory’.
- A customer name cannot determine its address and contact information because there might be some cases where customer’s name might be same but their address and contact information might be different.

Customer_Order 2 – (CustomerID*, OrderID*, PaymentNo, PaymentMethod, PaymentStatus, ConfigurationNo, InvoiceNumber, InvoiceDate)

From Customer_ Order – 2 Entity :-

Customer ID, Order ID -> PaymentNo, PaymentMethod, PaymentStatus, Configuration No

Payment No -> PaymentMethod, PaymentStatus, ConfigurationNo, InvoiceNO*, InvoiceDate

InvoiceNo -> InvoiceDate

Under the assumption :-

- Transitive Dependencies were found since some non prime attributes depended on another non prime attributes.
- The PaymentNo, PaymentMethod, PaymentStatus and ConfigurationNo attributes tend to depend on a non-key Attribute i.e. PaymentNo. So, separate entity 'Payment' were formed with Payment NO as a Primary key.
- The InvoiceDate also depends on InvocieNo as well as PaymentNo, CustomerID and OrderID. So separate entity 'Invoice' was formed with InvoiceNo as Primary key.

Order 2 – (<u>OrderID(PK), </u>OrderDate, OrderQuantity, TotalOrderAmount,)

<u>From Order -2 Entity:-</u>

Order ID -> Order Date, Order Quantity, Total Order Amount

Order Date -> X

Under the assumption:-

- There were no transitive dependencies found.
- There are no non key attributes (Order Date, Order Quantity, Total Order Amount) that are functionally dependent on one another.

Product 2 – (ProductID(PK) , ProductName, ModelNO, ProductDescription, UnitPrice$, StockLevel, AvailabilityStatus, LastUpdated, VendorID, VendorName, PhoneNo, EmailAddress, CategoryID, CategoryName)

From Product -2 Entity :-

Product ID -> ProductName, ModelNO, ProductDescription, UnitPrice$, StockLevel, AvailabilityStatus, LastUpdated

Vendor ID -> VendorName, PhoneNumber, EmailAddress

Category ID -> CategoryName

Under the assumption:-

- Transitive dependencies were found
- Attributes VendorName, phone and Email depends on the non key attribute Vendor ID. So different entity "Vendor" were formed with VendorID as the primary key. Each product should be associated with one vendor and each vendor can supply one or more products.
- Attribute CategoryName is also dependent on categoryID, a non key attribute so different entity 'Category' were formed with CategoryID as the primary key. Each product can be of only one category and each category can have one or many products.

Customer_Order_ Product 2 – (CustomerID*, OrderID*, ProductID*)

From Customer_ Order_ Product – 2 Entity:-

Under the assumption:-

- No transitive dependencies were found since all the attributes present are all keys referencing a different entities.

Putting into 3NF, the tables formed were,

Customer 3 - (CustomerID(PK), CustomerName, CustomerAddress, PhoneNo, Email Address, CustomerCategoryNo*)

CustomerCategory 3 – (CustomerCategoryID(PK), CustomerCategory, Discount)

Customer_Order 3 – (CustomerID(PK), OrderID(pk) CustomerID*, OrderID*, Payment No*)

Payment 3 – (PaymentNo(PK), PaymentMethod, PaymentStatus, ConfigurationNo, InvoiceNo*)

Invoice 3 - (InvoiceNo(PK), InvoiceDate)

Order 3 – (OrderID(PK), OrderDate, OrderQuantity, TotalOrderAmount)

Product 3 – (ProductID(PK) ,ProductName, ModelNO, ProductDescription, UnitPrice, StockLevel, AvailabilityStatus, LastUpdated, VendorID*, CategoryID*)

Vendor 3 – (VendorID(PK), VendorName, PhoneNo, EmailAddress)

Category 3 - (CategoryID(PK) , Category Name)

Customer_Order_Product 3 - (CustomerOrderProductID(PK) CustomerID(FK)*, OrderID(FK)*, ProductID(FK)*

4. Final ERD

The final ERD includes the necessary tables to represent customers, orders, payments, invoices, products, vendors, categories, and the relationships between them for Gadget Emporium. Orders, payments, invoices, and customer information are all organised well to reduce redundancy.

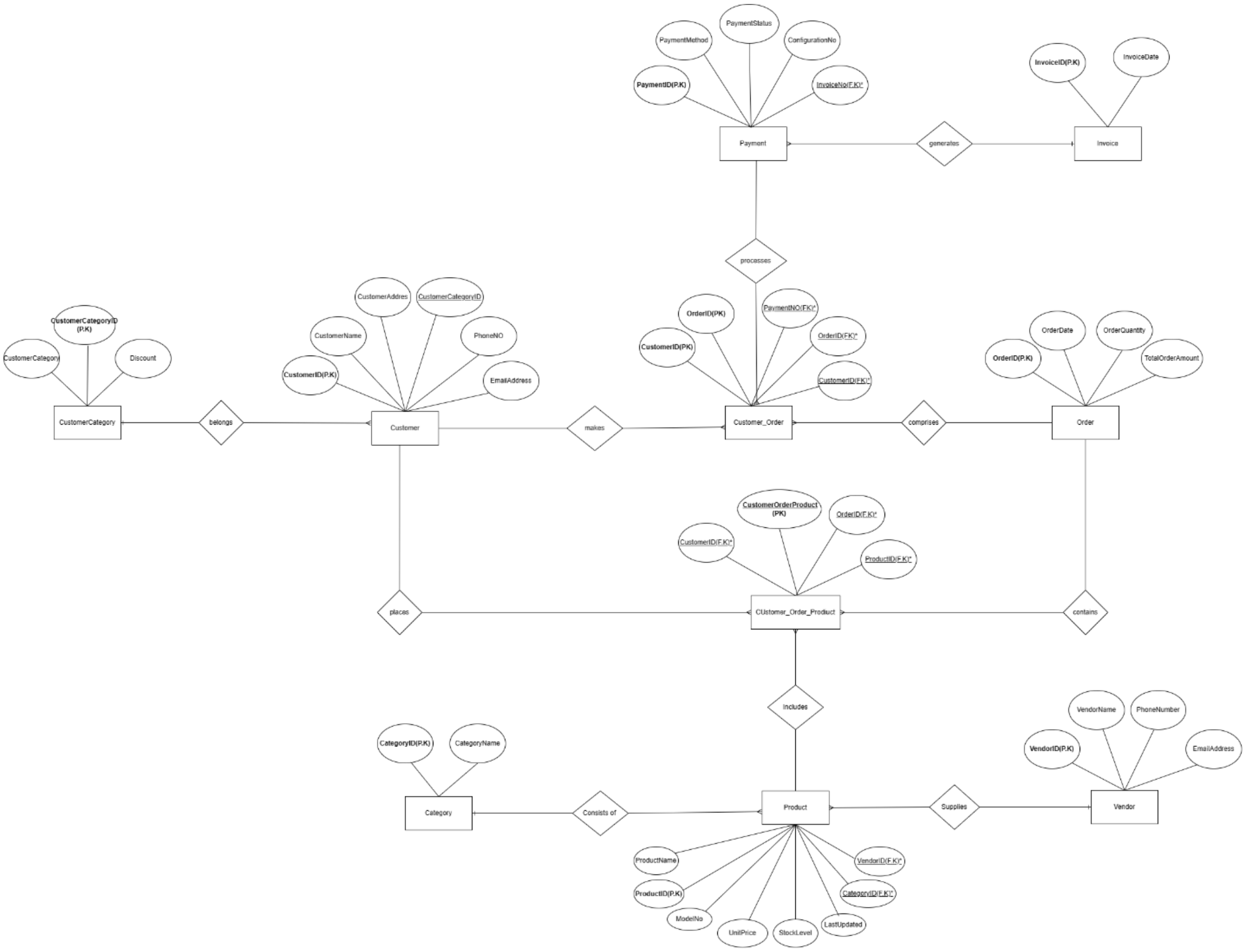


Figure 3 Final ERD-Chen Model

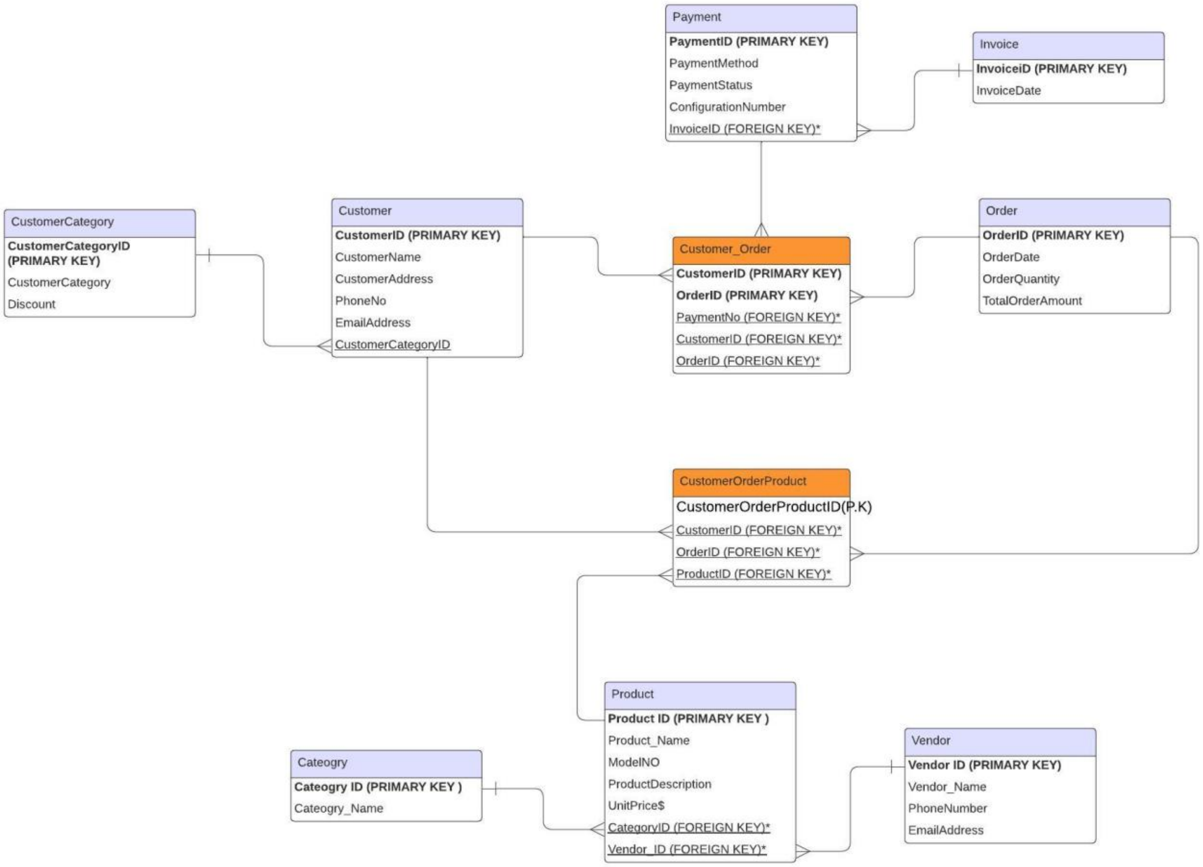
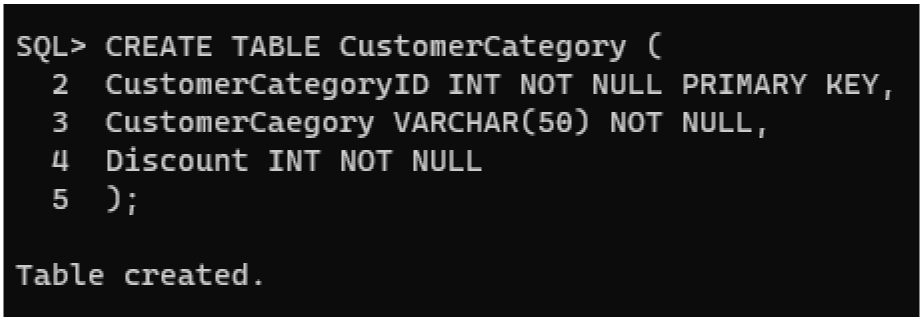


Figure 4 Final ERD-Crow's Foot Model

5. Implementation

5.1. Creation of tables

CustomerCategory



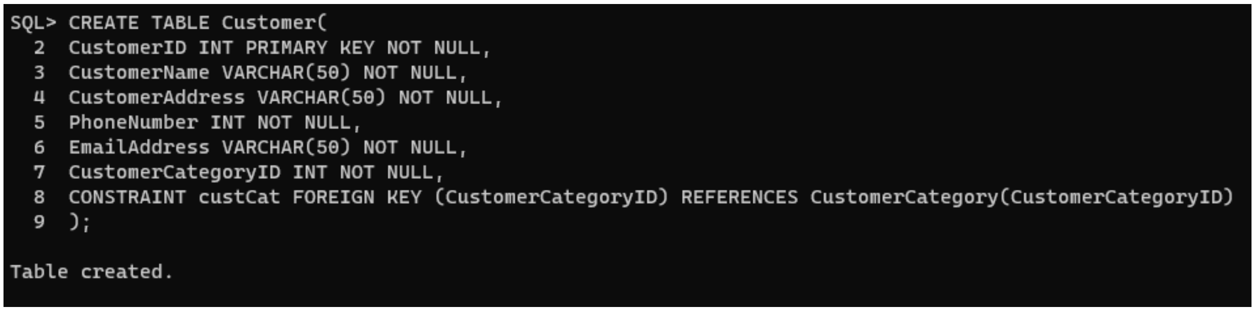
```
SQL> CREATE TABLE CustomerCategory (
     CustomerCategoryID INT NOT NULL PRIMARY KEY,
     CustomerCaegory VARCHAR(50) NOT NULL,
     Discount INT NOT NULL
     );

Table created.
```

Figure 5 Create table CustomerCategory

Attributes	Data Type	Constraints	Description
CustomerCategoryID	INT	PRIMARY KEY, NOT NULL	Unique Identifier for each category
CustomerCategory	VARCHAR(50)	NOT NULL	Names for customer categories
Discount	VARCHAR(3)	NOT NULL	Discount of customer category

Table 6 Final CustomerCategory table

Customer



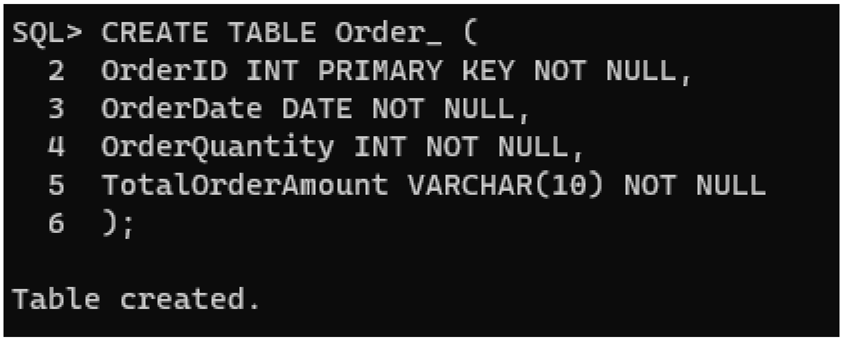
```
SQL> CREATE TABLE Customer(
     CustomerID INT PRIMARY KEY NOT NULL,
     CustomerName VARCHAR(50) NOT NULL,
     CustomerAddress VARCHAR(50) NOT NULL,
     PhoneNumber INT NOT NULL,
     EmailAddress VARCHAR(50) NOT NULL,
     CustomerCategoryID INT NOT NULL,
     CONSTRAINT custCat FOREIGN KEY (CustomerCategoryID) REFERENCES CustomerCategory(CustomerCategoryID)
     );

Table created.
```

Figure 6 Create table Customer

Attributes	Data Types	Constraints	Description
CustomerID	INT	PRIMARY KEY NOT NULL	Uniquely Identifies Customer
CustomerName	VARCHAR(50)	NOT NULL	Name of the customer
CustomerAddress	VARCHAR(50)	NOT NULL	Address of the customer
PhoneNumber	INT	NOT NULL	Phone number of the customer
EmailAddress	VARCHAR(50)	NOT NULL	Email Address of the customer
CustomerCategoryID	INT	FOREIGN KEY NOT NULL	References CustomerCategory Table

Table 7 Final Customer table

Order



```
SQL> CREATE TABLE Order_ (
     OrderID INT PRIMARY KEY NOT NULL,
     OrderDate DATE NOT NULL,
     OrderQuantity INT NOT NULL,
     TotalOrderAmount VARCHAR(10) NOT NULL
     );

Table created.
```

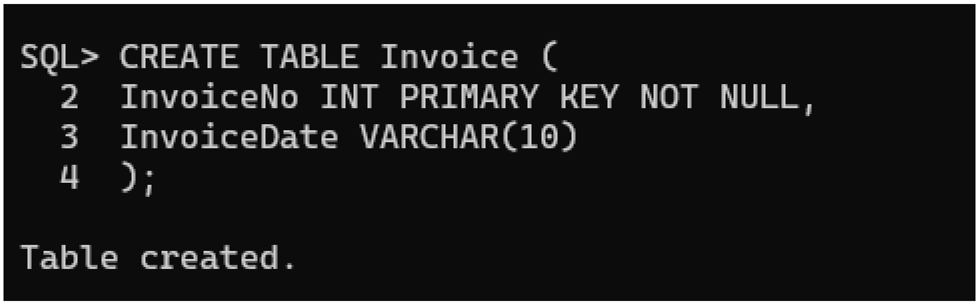
*Figure 7 Create table Order_*

Attributes	Data types	Constraint	Description
OrderID	INT	PRIMARY KEY NOT NULL	Uniquely Identifies Order
OrderDate	DATE	NOT NULL	Specifies the order date
OrderQuantity	INT	NOT NULL	Stores the quantity of the order

TotalOrderAmount	VARCHAR(10)	NOT NULL	Stores the total order amount of the customer

Table 8 Final Order_ table

Invoice



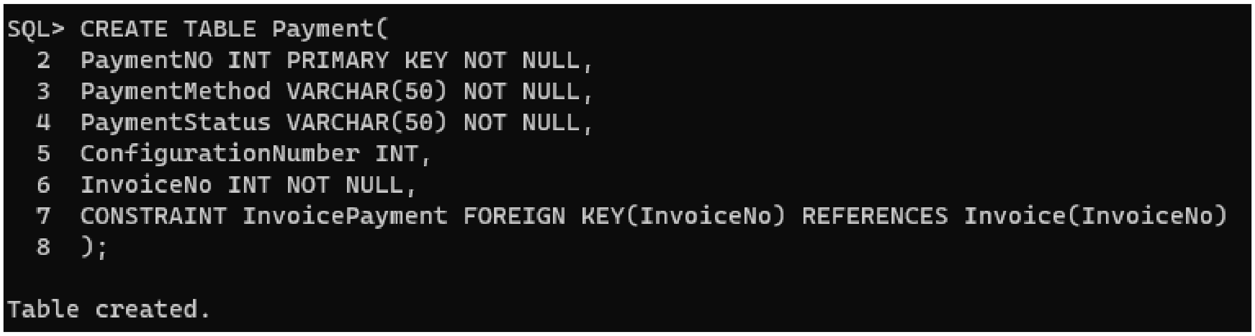
```
SQL> CREATE TABLE Invoice (
     InvoiceNo INT PRIMARY KEY NOT NULL,
     InvoiceDate VARCHAR(10)
     );

Table created.
```

Figure 8 Create table Invoice

Attributes	**Data types**	**Constraint**	**Description**
InvoiceNO	INT	NOT NULL	Identifies Invoice Number
InvoiceDate	VARCHAR(10)	NOT NULL	Specifies the date the invoice was generated

Table 9 Final Invoice table

Payment



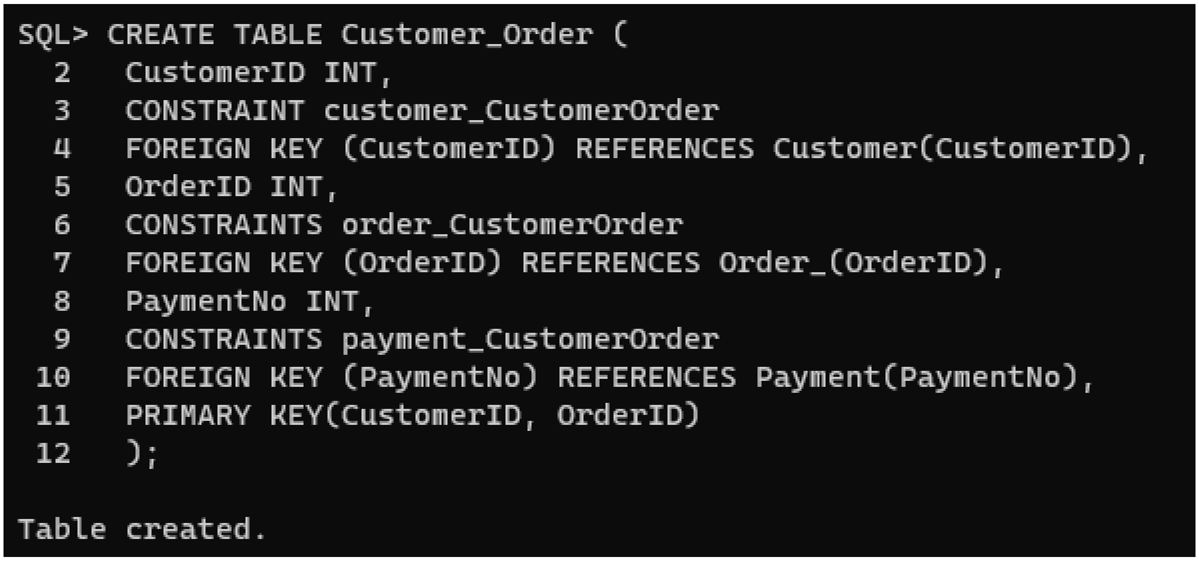
```
SQL> CREATE TABLE Payment(
     PaymentNO INT PRIMARY KEY NOT NULL,
     PaymentMethod VARCHAR(50) NOT NULL,
     PaymentStatus VARCHAR(50) NOT NULL,
     ConfigurationNumber INT,
     InvoiceNo INT NOT NULL,
     CONSTRAINT InvoicePayment FOREIGN KEY(InvoiceNo) REFERENCES Invoice(InvoiceNo)
     );

Table created.
```

Figure 9 Create table Payment

Attributes	Data types	Constraint	Description
PaymentNO	INT	NOT NULL	Identifies Payment
PaymentMethod	VARCHAR(50)	NOT NULL	Payment Method of the order
PaymentStatus	VARCHAR(10)	NOT NULL	Tracks the payment status of the order
ConfigurationNO	INT	NOT NULL	Confirms the Payment status by configuration number
InvoiceNO	INT	FOREIGN KEY NOT NULL	References Invoice Table in Payment table

Table 10 Final Payment table

Customer_Order



```
SQL> CREATE TABLE Customer_Order (
     CustomerID INT,
     CONSTRAINT customer_CustomerOrder
     FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
     OrderID INT,
     CONSTRAINTS order_CustomerOrder
     FOREIGN KEY (OrderID) REFERENCES Order_(OrderID),
     PaymentNo INT,
     CONSTRAINTS payment_CustomerOrder
     FOREIGN KEY (PaymentNo) REFERENCES Payment(PaymentNo),
     PRIMARY KEY(CustomerID, OrderID)
     );

Table created.
```

Figure 10 Create table Customer_Order

Attributes	Data types	Constraint	Description
CustomerID	INT	PRIMARY KEY NOT NULL	Uniquely Identifies Customer

OrderID	INT	PRIMARY KEY NOT NULL	Uniquely Identifies Order
PaymentNo	INT	FOREIGN KEY NOT NULL	References Payment Table in Customer_Order table
CustomerID	INT	FOREIGN KEY NOT NULL	References Customer Table in Customer_Order table
OrderID	INT	FOREIGN KEY NOT NULL	References Order Table in Customer_Order table

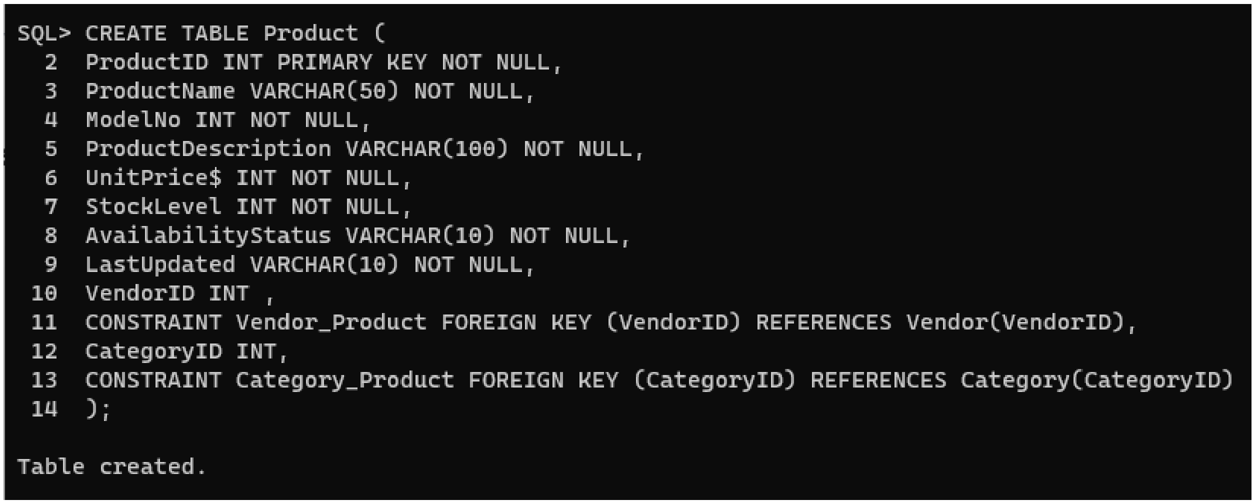
Table 11 Final Customer_Order table

Product

Attributes	**Data type**	**Constraints**	**Description**
ProductID	INT	PRIMARY KEY, NOT NULL	Unique Identifier for each Product
ProductName	VARCHAR(50)	NOT NULL	Names for the products
ModelNo	INT	NOT NULL	Model Numbers for each product
ProductDescription	VARCHAR(100)	NOT NULL	Description of products
UnitPrice$	INT	NOT NULL	Price of the products
StockLevel	INT	NOT NULL	Stock Levels of the product
AvailabilityStatus	VARCHAR(50)	NOT NULL	States the availability status of the product
LastUpdated	DATE	NOT NULL	Date when the inventory details were updated

CategoryID	INT	FOREIGN KEY NOT NULL	References Category table in Product table
VendorID	INT	FOREIGN KEY NOT NULL	References Vendor Table in Product Table

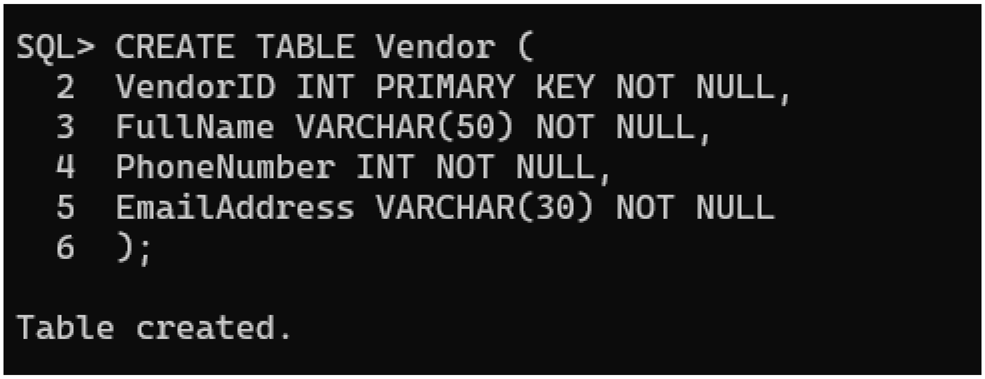
Table 12 Final Product table



```
SQL> CREATE TABLE Product (
  2  ProductID INT PRIMARY KEY NOT NULL,
  3  ProductName VARCHAR(50) NOT NULL,
  4  ModelNo INT NOT NULL,
  5  ProductDescription VARCHAR(100) NOT NULL,
  6  UnitPrice$ INT NOT NULL,
  7  StockLevel INT NOT NULL,
  8  AvailabilityStatus VARCHAR(10) NOT NULL,
  9  LastUpdated VARCHAR(10) NOT NULL,
 10  VendorID INT ,
 11  CONSTRAINT Vendor_Product FOREIGN KEY (VendorID) REFERENCES Vendor(VendorID),
 12  CategoryID INT,
 13  CONSTRAINT Category_Product FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)
 14  );

Table created.
```

Figure 11 Create table Product

Vendor



```
SQL> CREATE TABLE Vendor (
  2  VendorID INT PRIMARY KEY NOT NULL,
  3  FullName VARCHAR(50) NOT NULL,
  4  PhoneNumber INT NOT NULL,
  5  EmailAddress VARCHAR(30) NOT NULL
  6  );

Table created.
```

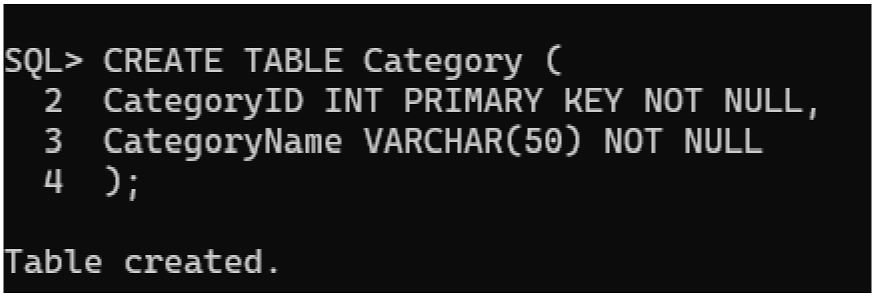
Figure 12 Create table Vendor

Attributes	**Data Type**	**Constraints**	**Description**
VendorID	INT	PRIMARYKEY, NOT NULL	Unique Identifier for each vendor
VendorName	VARCHAR(50)	NOT NULL	Names of the vendors

PhoneNumber	INT	NOT NULL	Phone Numbers of Vendors
EmailAddress	VARCHAR(30)	NOT NUL	Email of each Vendors

Table 13 Final Vendor table

Category



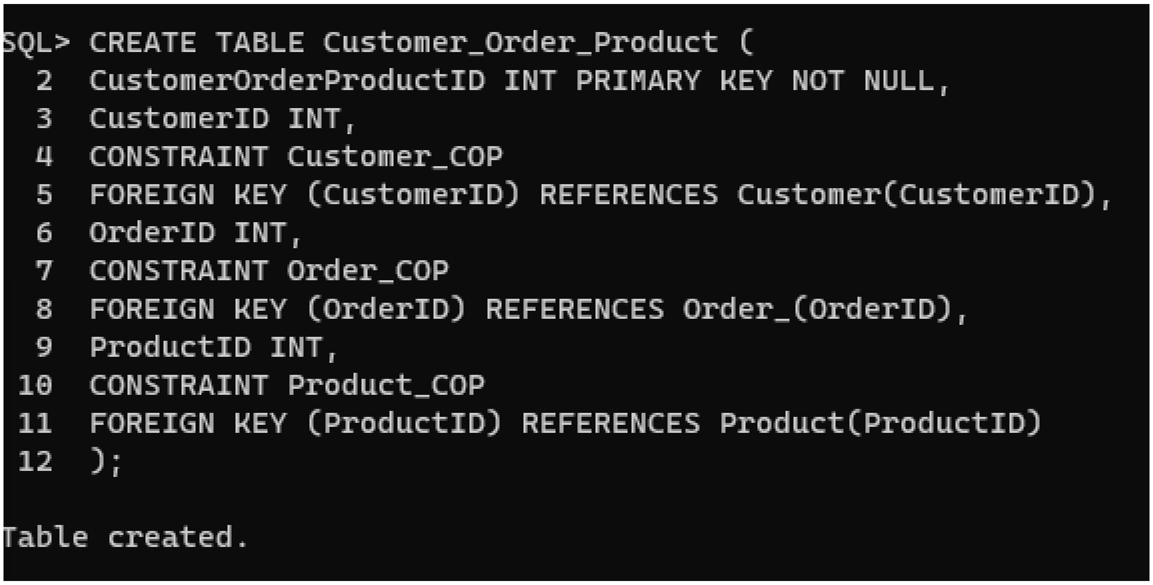
```
SQL> CREATE TABLE Category (
  2  CategoryID INT PRIMARY KEY NOT NULL,
  3  CategoryName VARCHAR(50) NOT NULL
  4  );

Table created.
```

Figure 13 Create table Category

Attributes	**Data Type**	**Constraints**	**Description**
CategoryID	INT	PRIMARY KEY, NOT NULL	Unique Identifier for each category
CategoryName	VARCHAR(50)	NOT NLL	Names for each categories

Table 14 Final Category table

Customer_Order_Product



```
SQL> CREATE TABLE Customer_Order_Product (
  2  CustomerOrderProductID INT PRIMARY KEY NOT NULL,
  3  CustomerID INT,
  4  CONSTRAINT Customer_COP
  5  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
  6  OrderID INT,
  7  CONSTRAINT Order_COP
  8  FOREIGN KEY (OrderID) REFERENCES Order_(OrderID),
  9  ProductID INT,
 10  CONSTRAINT Product_COP
 11  FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
 12  );

Table created.
```

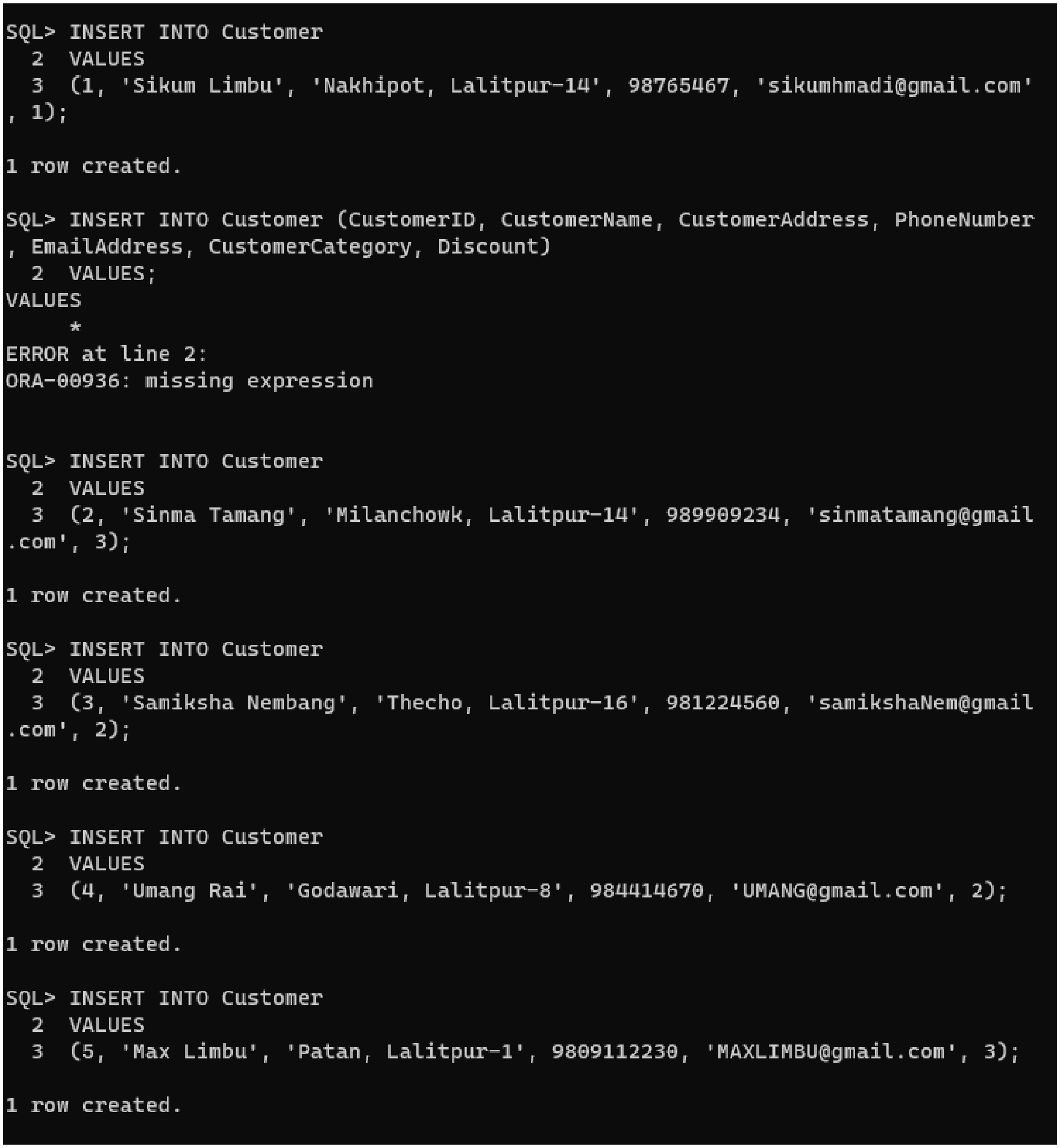
Figure 14 Create table Customer_Order_Product

Attributes	**Data types**	**Constraint**	**Description**
CustomerOrderProductID	INT	PRIMARY KEY NOT NULL	Uniquely Identifies the bridge table Customer_Order_Product
CustomerID	INT	FOREIGN KEY NOT NULL	References Customer Table in Customer_Order_Product table
OrderID	INT	FOREIGN KEY NOT NULL	References Order Table in Customer_Order_Product table
ProductIID	INT	FOREIGN KEY NOT NULL	References Product Table in Customer_Order_Product table

Table 15 Final Customer_Order_Product table

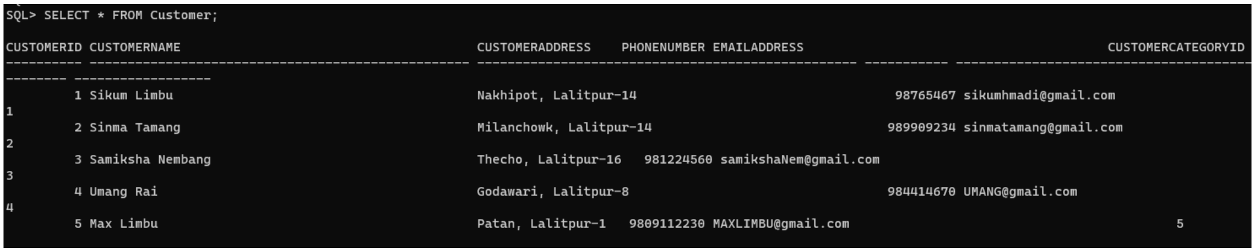
5.2. Insertion of Values in the tables

Insert value in Customer



```
SQL> INSERT INTO Customer
  2  VALUES
  3  (1, 'Sikum Limbu', 'Nakhipot, Lalitpur-14', 98765467, 'sikumhmadi@gmail.com'
, 1);

1 row created.

SQL> INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, PhoneNumber
, EmailAddress, CustomerCategory, Discount)
  2  VALUES;
VALUES
     *
ERROR at line 2:
ORA-00936: missing expression


SQL> INSERT INTO Customer
  2  VALUES
  3  (2, 'Sinma Tamang', 'Milanchowk, Lalitpur-14', 989909234, 'sinmatamang@gmail
.com', 3);

1 row created.

SQL> INSERT INTO Customer
  2  VALUES
  3  (3, 'Samiksha Nembang', 'Thecho, Lalitpur-16', 981224560, 'samikshaNem@gmail
.com', 2);

1 row created.

SQL> INSERT INTO Customer
  2  VALUES
  3  (4, 'Umang Rai', 'Godawari, Lalitpur-8', 984414670, 'UMANG@gmail.com', 2);

1 row created.

SQL> INSERT INTO Customer
  2  VALUES
  3  (5, 'Max Limbu', 'Patan, Lalitpur-1', 9809112230, 'MAXLIMBU@gmail.com', 3);

1 row created.
```

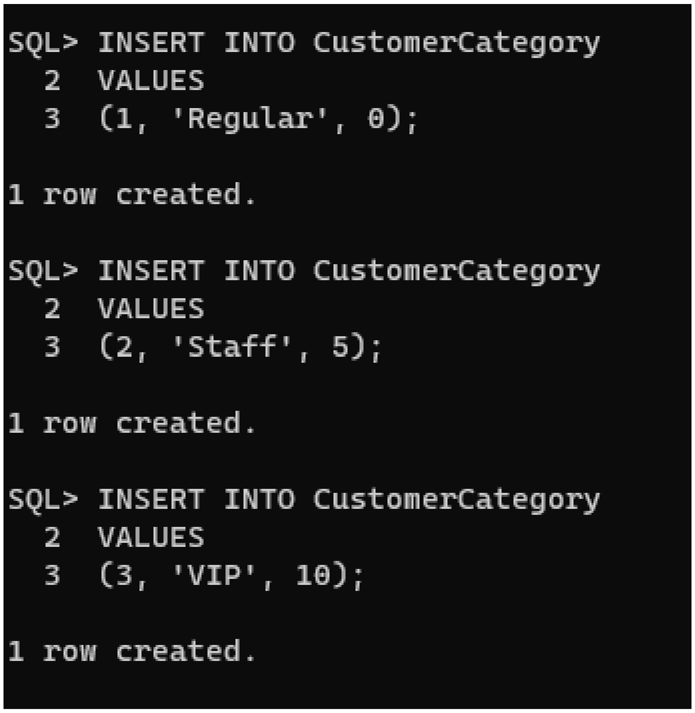
Figure 15 Insert Value into Customer



```
SQL> SELECT * FROM Customer;

CUSTOMERID CUSTOMERNAME                                       CUSTOMERADDRESS    PHONENUMBER EMAILADDRESS                               CUSTOMERCATEGORYID
---------- -------------------------------------------------- -------------------------------------------------- ----------- ---------------------------------------
-------- ------------------
         1 Sikum Limbu                                        Nakhipot, Lalitpur-14                                  98765467 sikumhmadi@gmail.com
1
         2 Sinma Tamang                                       Milanchowk, Lalitpur-14                               989909234 sinmatamang@gmail.com
2
         3 Samiksha Nembang                                   Thecho, Lalitpur-16   981224560 samikshaNem@gmail.com
3
         4 Umang Rai                                          Godawari, Lalitpur-8                                  984414670 UMANG@gmail.com
4
         5 Max Limbu                                          Patan, Lalitpur-1   9809112230 MAXLIMBU@gmail.com                                  5
```

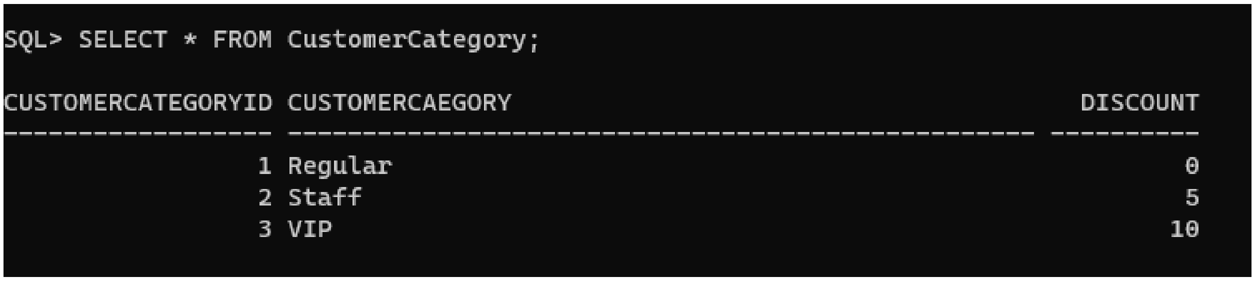
Figure 16 Values from Customer

Insert in CustomerCategory



```
SQL> INSERT INTO CustomerCategory
  2  VALUES
  3  (1, 'Regular', 0);

1 row created.

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

1 row created.

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

1 row created.
```

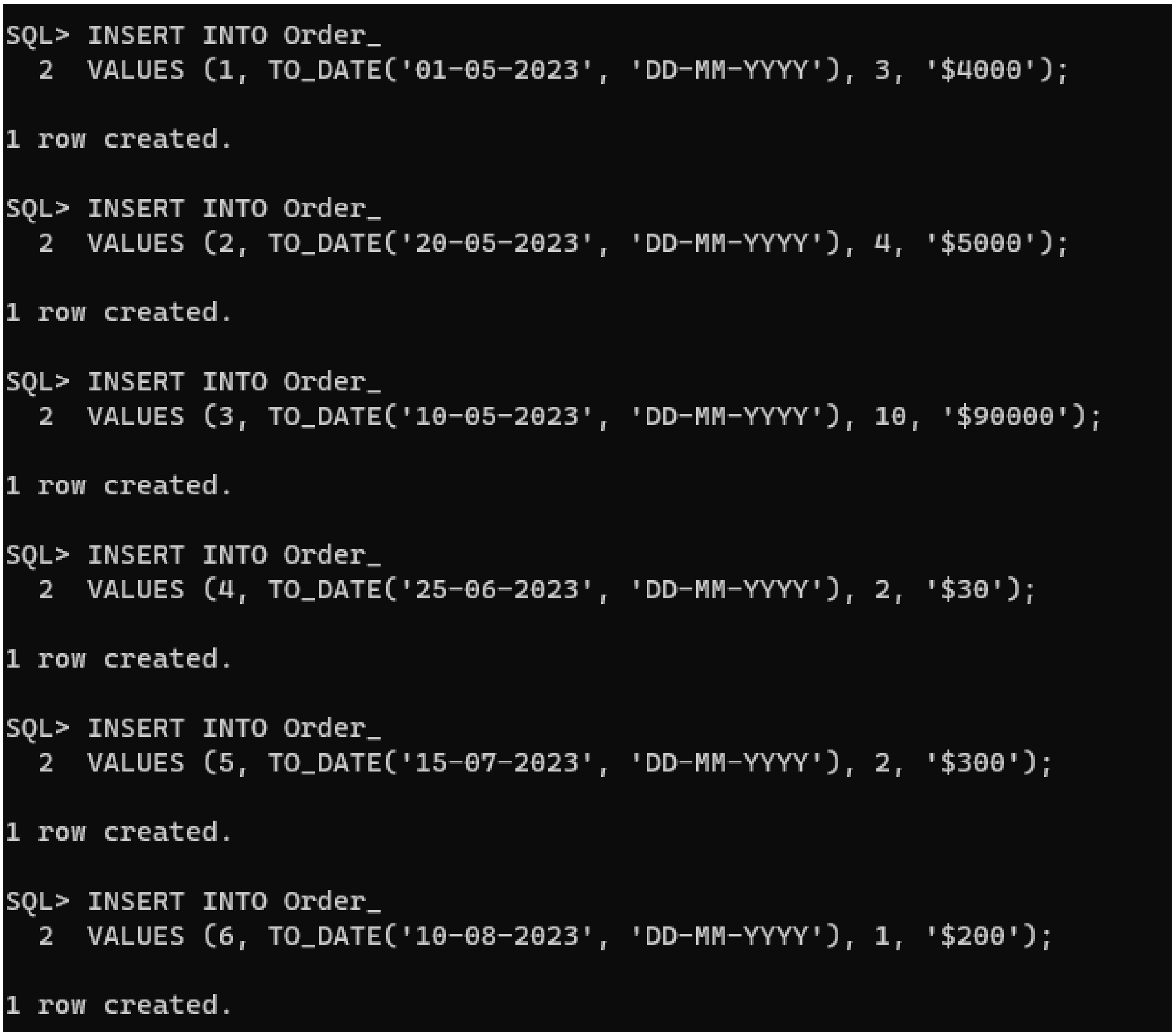
Figure 17 Insert value in CustomerCategory



```
SQL> SELECT * FROM CustomerCategory;

CUSTOMERCATEGORYID CUSTOMERCAEGORY                                      DISCOUNT
------------------ -------------------------------------------------- ----------
                 1 Regular                                                     0
                 2 Staff                                                       5
                 3 VIP                                                        10
```

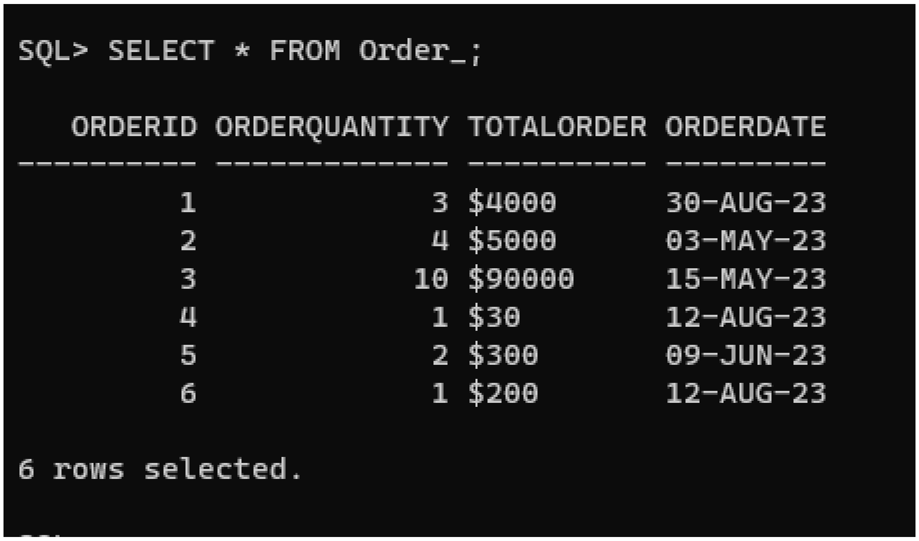
Figure 18 Value in CustomerCategory

Insert value in Order



```
SQL> INSERT INTO Order_
  2  VALUES (1, TO_DATE('01-05-2023', 'DD-MM-YYYY'), 3, '$4000');

1 row created.

SQL> INSERT INTO Order_
  2  VALUES (2, TO_DATE('20-05-2023', 'DD-MM-YYYY'), 4, '$5000');

1 row created.

SQL> INSERT INTO Order_
  2  VALUES (3, TO_DATE('10-05-2023', 'DD-MM-YYYY'), 10, '$90000');

1 row created.

SQL> INSERT INTO Order_
  2  VALUES (4, TO_DATE('25-06-2023', 'DD-MM-YYYY'), 2, '$30');

1 row created.

SQL> INSERT INTO Order_
  2  VALUES (5, TO_DATE('15-07-2023', 'DD-MM-YYYY'), 2, '$300');

1 row created.

SQL> INSERT INTO Order_
  2  VALUES (6, TO_DATE('10-08-2023', 'DD-MM-YYYY'), 1, '$200');

1 row created.
```

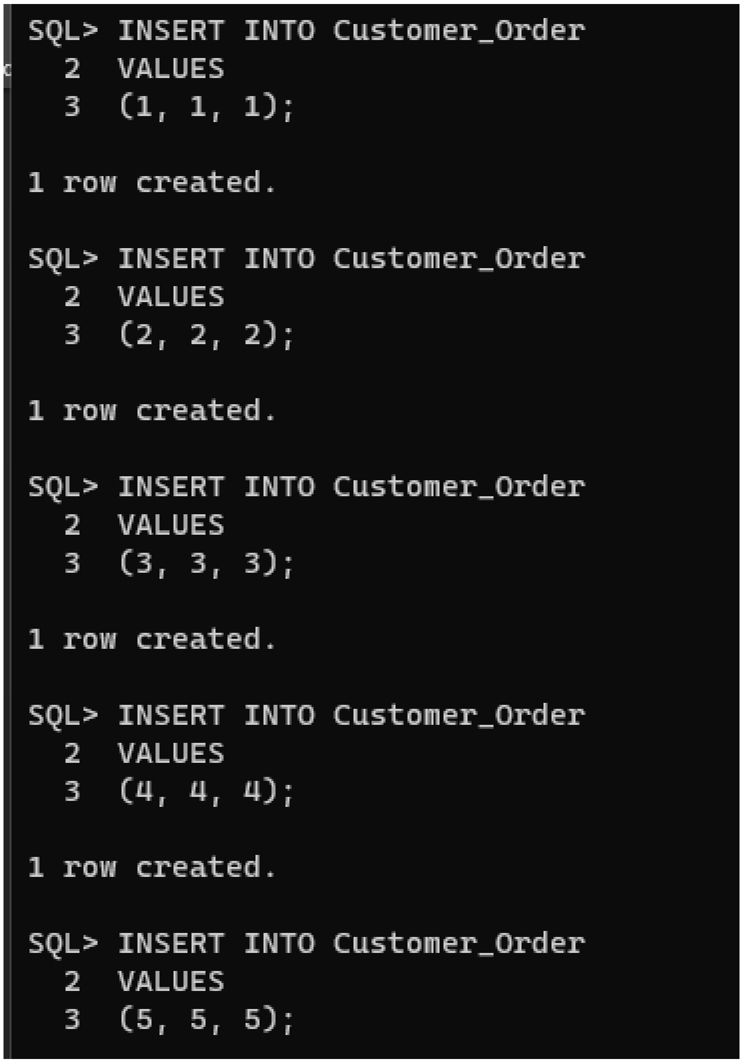
*Figure 19 Insert value in Order_*



```
SQL> SELECT * FROM Order_;

   ORDERID ORDERQUANTITY TOTALORDER ORDERDATE
---------- ------------- ---------- ---------
         1             3 $4000      30-AUG-23
         2             4 $5000      03-MAY-23
         3            10 $90000     15-MAY-23
         4             1 $30        12-AUG-23
         5             2 $300       09-JUN-23
         6             1 $200       12-AUG-23

6 rows selected.
```

Figure 20 values from Order

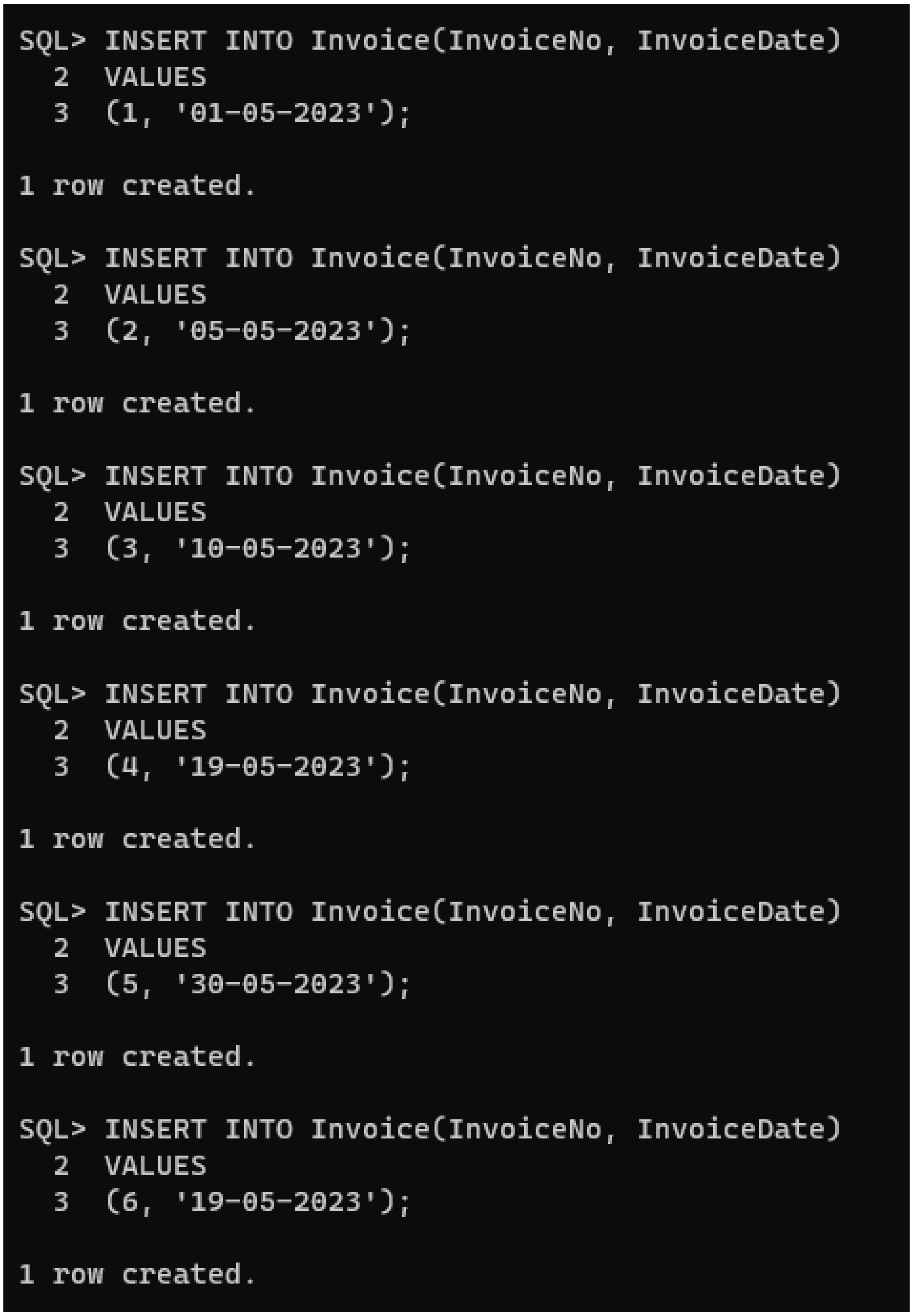
Insert value in Customer_Order



```
SQL> INSERT INTO Customer_Order
  2  VALUES
  3  (1, 1, 1);

1 row created.

SQL> INSERT INTO Customer_Order
  2  VALUES
  3  (2, 2, 2);

1 row created.

SQL> INSERT INTO Customer_Order
  2  VALUES
  3  (3, 3, 3);

1 row created.

SQL> INSERT INTO Customer_Order
  2  VALUES
  3  (4, 4, 4);

1 row created.

SQL> INSERT INTO Customer_Order
  2  VALUES
  3  (5, 5, 5);
```

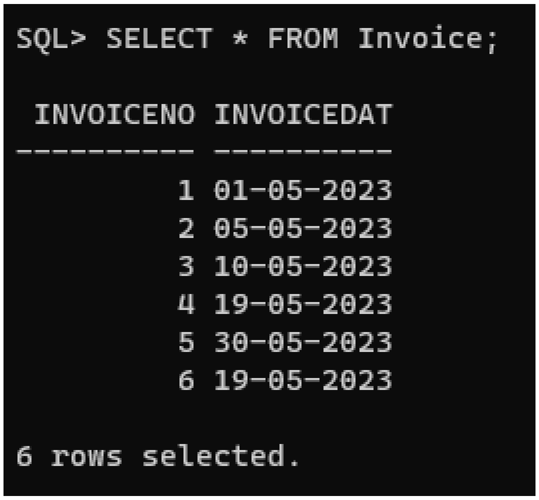
Figure 21 Insert value in Customer_Order

Insert value in Invoice



```
SQL> INSERT INTO Invoice(InvoiceNo, InvoiceDate)
  2  VALUES
  3  (1, '01-05-2023');

1 row created.

SQL> INSERT INTO Invoice(InvoiceNo, InvoiceDate)
  2  VALUES
  3  (2, '05-05-2023');

1 row created.

SQL> INSERT INTO Invoice(InvoiceNo, InvoiceDate)
  2  VALUES
  3  (3, '10-05-2023');

1 row created.

SQL> INSERT INTO Invoice(InvoiceNo, InvoiceDate)
  2  VALUES
  3  (4, '19-05-2023');

1 row created.

SQL> INSERT INTO Invoice(InvoiceNo, InvoiceDate)
  2  VALUES
  3  (5, '30-05-2023');

1 row created.

SQL> INSERT INTO Invoice(InvoiceNo, InvoiceDate)
  2  VALUES
  3  (6, '19-05-2023');

1 row created.
```

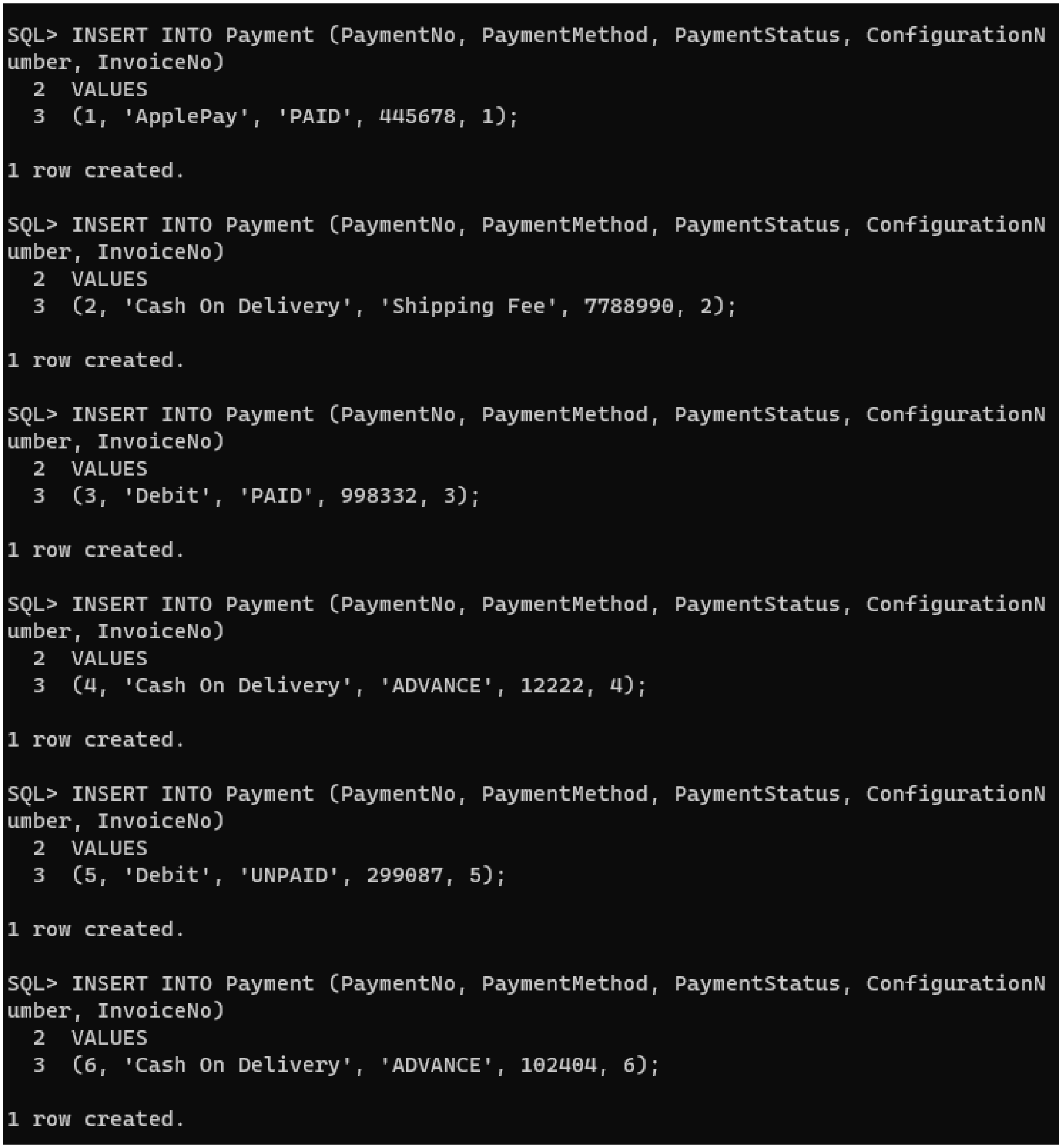
Figure 22 Insert values in Invoice



```
SQL> SELECT * FROM Invoice;

 INVOICENO INVOICEDAT
---------- ----------
         1 01-05-2023
         2 05-05-2023
         3 10-05-2023
         4 19-05-2023
         5 30-05-2023
         6 19-05-2023

6 rows selected.
```

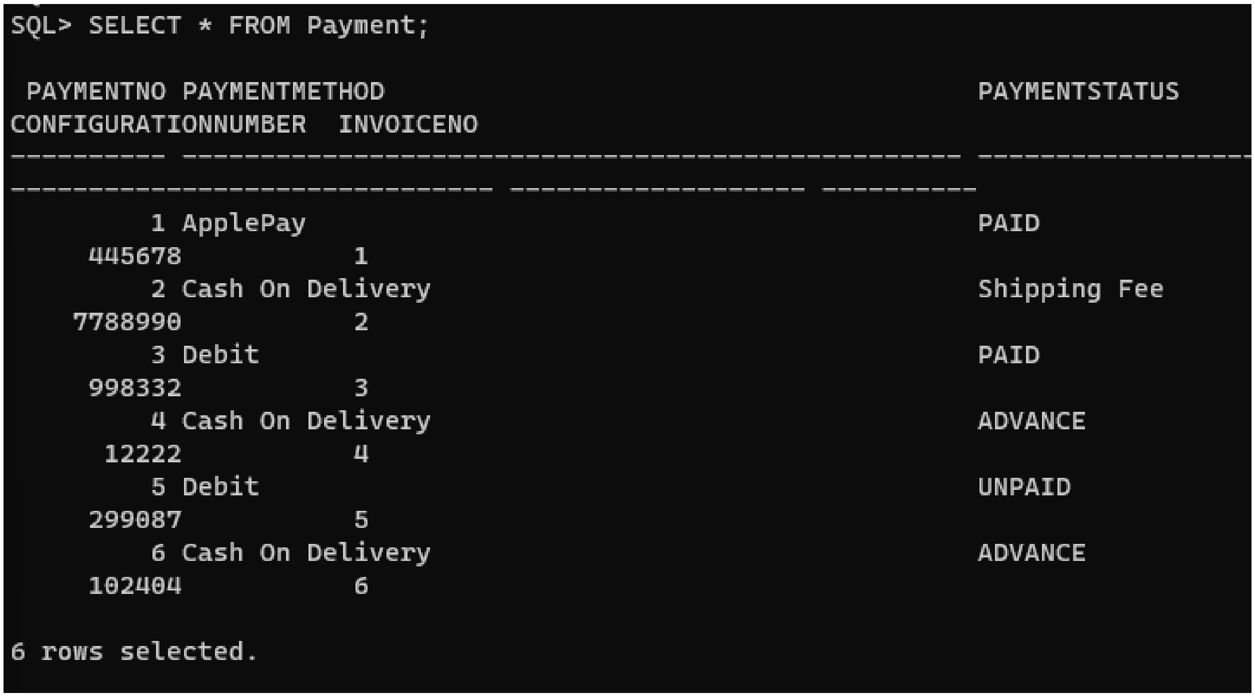
Figure 23 Values in Invoice

Insert value in Payment



```
SQL> INSERT INTO Payment (PaymentNo, PaymentMethod, PaymentStatus, ConfigurationN
umber, InvoiceNo)
  2  VALUES
  3  (1, 'ApplePay', 'PAID', 445678, 1);

1 row created.

SQL> INSERT INTO Payment (PaymentNo, PaymentMethod, PaymentStatus, ConfigurationN
umber, InvoiceNo)
  2  VALUES
  3  (2, 'Cash On Delivery', 'Shipping Fee', 7788990, 2);

1 row created.

SQL> INSERT INTO Payment (PaymentNo, PaymentMethod, PaymentStatus, ConfigurationN
umber, InvoiceNo)
  2  VALUES
  3  (3, 'Debit', 'PAID', 998332, 3);

1 row created.

SQL> INSERT INTO Payment (PaymentNo, PaymentMethod, PaymentStatus, ConfigurationN
umber, InvoiceNo)
  2  VALUES
  3  (4, 'Cash On Delivery', 'ADVANCE', 12222, 4);

1 row created.

SQL> INSERT INTO Payment (PaymentNo, PaymentMethod, PaymentStatus, ConfigurationN
umber, InvoiceNo)
  2  VALUES
  3  (5, 'Debit', 'UNPAID', 299087, 5);

1 row created.

SQL> INSERT INTO Payment (PaymentNo, PaymentMethod, PaymentStatus, ConfigurationN
umber, InvoiceNo)
  2  VALUES
  3  (6, 'Cash On Delivery', 'ADVANCE', 102404, 6);

1 row created.
```

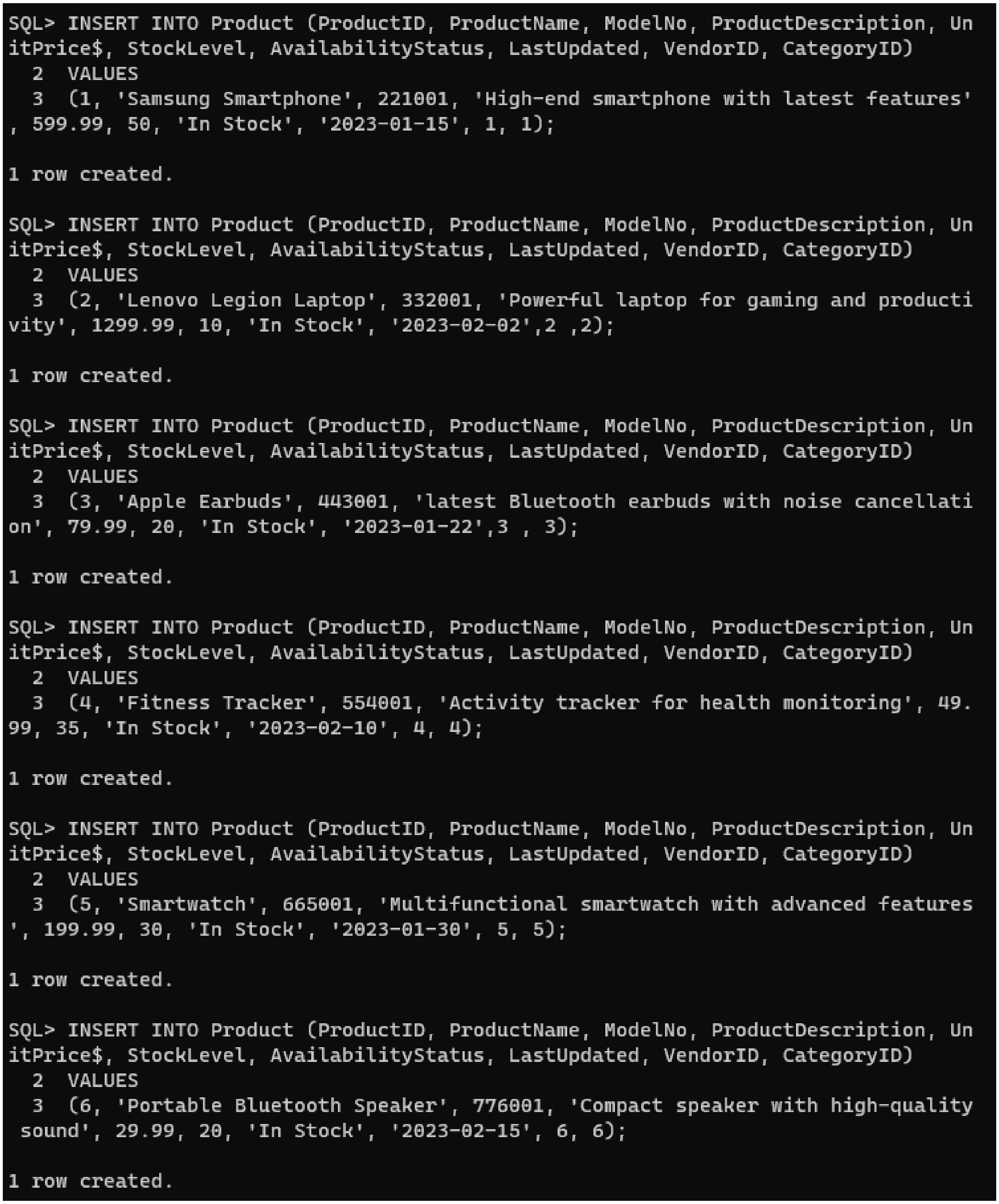
Figure 24 Insert values in Payment



```
SQL> SELECT * FROM Payment;

 PAYMENTNO PAYMENTMETHOD                                      PAYMENTSTATUS
CONFIGURATIONNUMBER  INVOICENO
---------- -------------------------------------------------- ----------------
------------------------------- ------------------- ----------
         1 ApplePay                                           PAID
     445678           1
         2 Cash On Delivery                                   Shipping Fee
    7788990           2
         3 Debit                                              PAID
     998332           3
         4 Cash On Delivery                                   ADVANCE
      12222           4
         5 Debit                                              UNPAID
     299087           5
         6 Cash On Delivery                                   ADVANCE
     102404           6

6 rows selected.
```

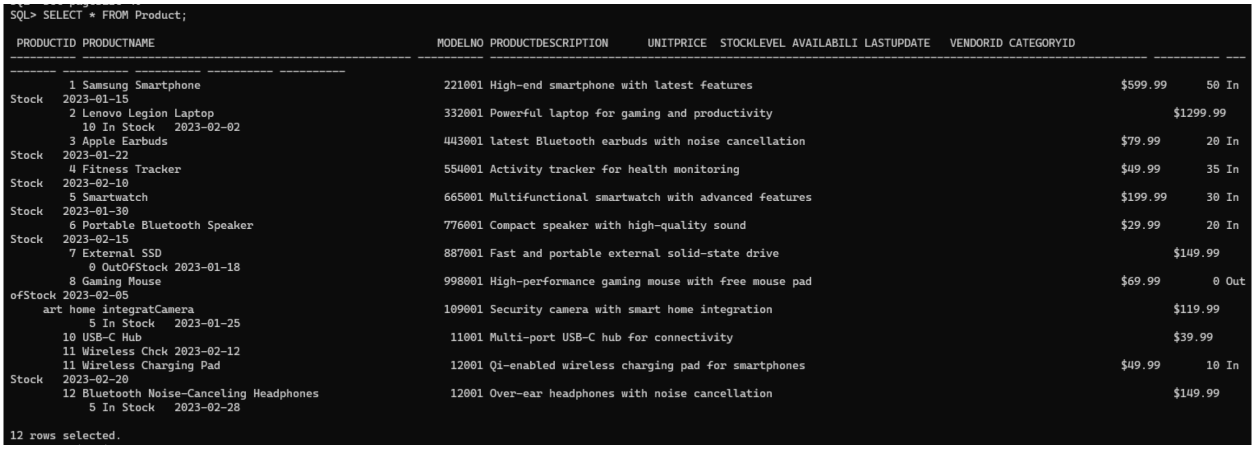
Figure 25 values in Payment table

Insert value in Product



```
SQL> INSERT INTO Product (ProductID, ProductName, ModelNo, ProductDescription, Un
itPrice$, StockLevel, AvailabilityStatus, LastUpdated, VendorID, CategoryID)
  2  VALUES
  3  (1, 'Samsung Smartphone', 221001, 'High-end smartphone with latest features'
, 599.99, 50, 'In Stock', '2023-01-15', 1, 1);

1 row created.

SQL> INSERT INTO Product (ProductID, ProductName, ModelNo, ProductDescription, Un
itPrice$, StockLevel, AvailabilityStatus, LastUpdated, VendorID, CategoryID)
  2  VALUES
  3  (2, 'Lenovo Legion Laptop', 332001, 'Powerful laptop for gaming and producti
vity', 1299.99, 10, 'In Stock', '2023-02-02',2 ,2);

1 row created.

SQL> INSERT INTO Product (ProductID, ProductName, ModelNo, ProductDescription, Un
itPrice$, StockLevel, AvailabilityStatus, LastUpdated, VendorID, CategoryID)
  2  VALUES
  3  (3, 'Apple Earbuds', 443001, 'latest Bluetooth earbuds with noise cancellati
on', 79.99, 20, 'In Stock', '2023-01-22',3 , 3);

1 row created.

SQL> INSERT INTO Product (ProductID, ProductName, ModelNo, ProductDescription, Un
itPrice$, StockLevel, AvailabilityStatus, LastUpdated, VendorID, CategoryID)
  2  VALUES
  3  (4, 'Fitness Tracker', 554001, 'Activity tracker for health monitoring', 49.
99, 35, 'In Stock', '2023-02-10', 4, 4);

1 row created.

SQL> INSERT INTO Product (ProductID, ProductName, ModelNo, ProductDescription, Un
itPrice$, StockLevel, AvailabilityStatus, LastUpdated, VendorID, CategoryID)
  2  VALUES
  3  (5, 'Smartwatch', 665001, 'Multifunctional smartwatch with advanced features
', 199.99, 30, 'In Stock', '2023-01-30', 5, 5);

1 row created.

SQL> INSERT INTO Product (ProductID, ProductName, ModelNo, ProductDescription, Un
itPrice$, StockLevel, AvailabilityStatus, LastUpdated, VendorID, CategoryID)
  2  VALUES
  3  (6, 'Portable Bluetooth Speaker', 776001, 'Compact speaker with high-quality
 sound', 29.99, 20, 'In Stock', '2023-02-15', 6, 6);

1 row created.
```

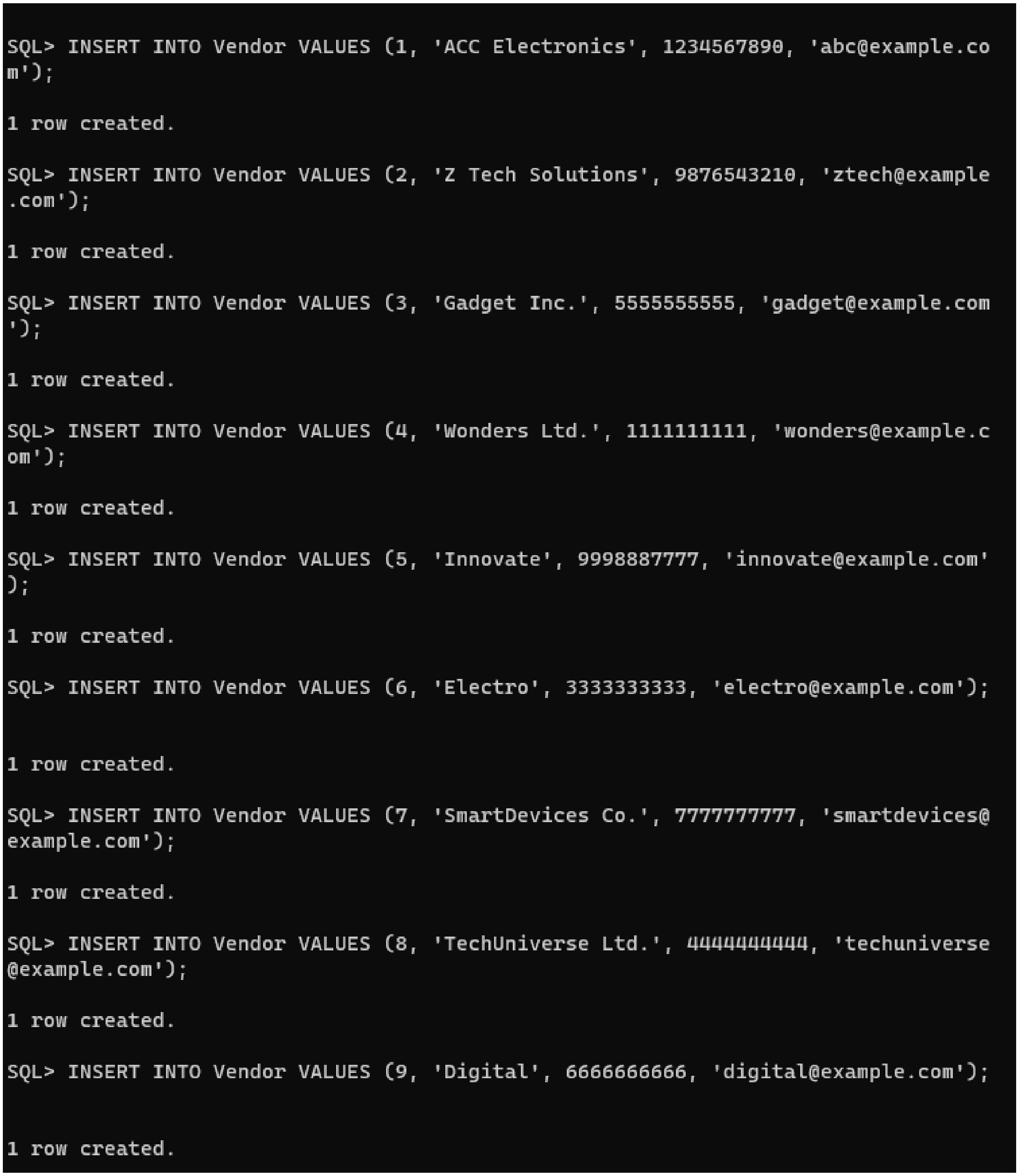
Figure 26 Inserted value in Product



```
SQL> SELECT * FROM Product;

 PRODUCTID PRODUCTNAME                                                MODELNO PRODUCTDESCRIPTION      UNITPRICE  STOCKLEVEL AVAILABILI LASTUPDATE   VENDORID CATEGORYID
---------- -------------------------------------------------- ---------- ---------------------------------------------------------------------------------------------------- ---------- ---
------- ---------- ---------- ---------- ----------
         1 Samsung Smartphone                                    221001 High-end smartphone with latest features                                                            $599.99      50 In
Stock   2023-01-15
         2 Lenovo Legion Laptop                                  332001 Powerful laptop for gaming and productivity                                                                 $1299.99
           10 In Stock   2023-02-02
         3 Apple Earbuds                                         443001 latest Bluetooth earbuds with noise cancellation                                                    $79.99       20 In
Stock   2023-01-22
         4 Fitness Tracker                                       554001 Activity tracker for health monitoring                                                              $49.99       35 In
Stock   2023-02-10
         5 Smartwatch                                            665001 Multifunctional smartwatch with advanced features                                                   $199.99      30 In
Stock   2023-01-30
         6 Portable Bluetooth Speaker                            776001 Compact speaker with high-quality sound                                                             $29.99       20 In
Stock   2023-02-15
         7 External SSD                                          887001 Fast and portable external solid-state drive                                                                $149.99
            0 OutOfStock 2023-01-18
         8 Gaming Mouse                                          998001 High-performance gaming mouse with free mouse pad                                                   $69.99        0 Out
ofStock 2023-02-05
     art home integratCamera                                     109001 Security camera with smart home integration                                                                 $119.99
            5 In Stock   2023-01-25
        10 USB-C Hub                                              11001 Multi-port USB-C hub for connectivity                                                                       $39.99
        11 Wireless Chck 2023-02-12
        11 Wireless Charging Pad                                  12001 Qi-enabled wireless charging pad for smartphones                                                    $49.99       10 In
Stock   2023-02-20
        12 Bluetooth Noise-Canceling Headphones                   12001 Over-ear headphones with noise cancellation                                                                 $149.99
            5 In Stock   2023-02-28

12 rows selected.
```

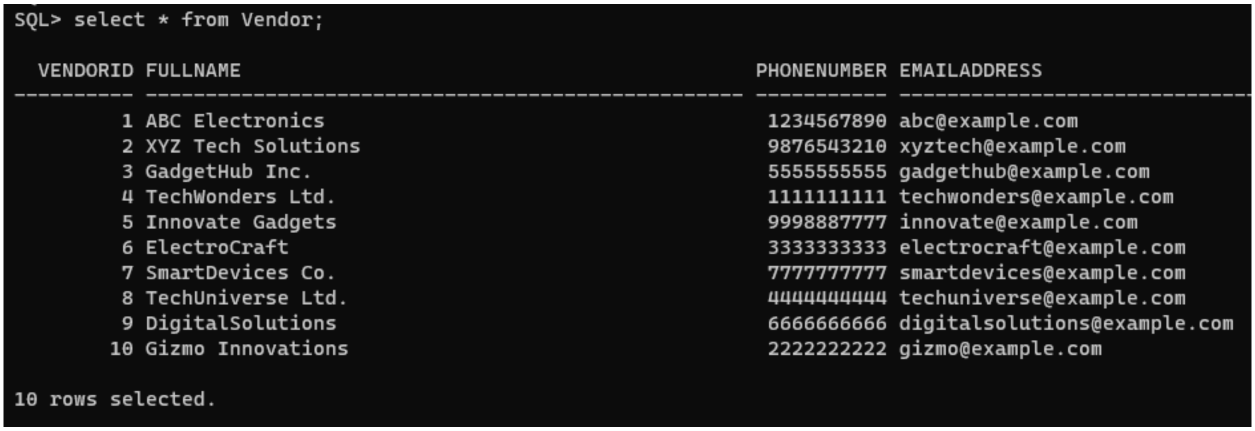
Figure 27 Values in Product Table

Insert value in Vendor



```
SQL> INSERT INTO Vendor VALUES (1, 'ACC Electronics', 1234567890, 'abc@example.co
m');

1 row created.

SQL> INSERT INTO Vendor VALUES (2, 'Z Tech Solutions', 9876543210, 'ztech@example
.com');

1 row created.

SQL> INSERT INTO Vendor VALUES (3, 'Gadget Inc.', 5555555555, 'gadget@example.com
');

1 row created.

SQL> INSERT INTO Vendor VALUES (4, 'Wonders Ltd.', 1111111111, 'wonders@example.c
om');

1 row created.

SQL> INSERT INTO Vendor VALUES (5, 'Innovate', 9998887777, 'innovate@example.com'
);

1 row created.

SQL> INSERT INTO Vendor VALUES (6, 'Electro', 3333333333, 'electro@example.com');


1 row created.

SQL> INSERT INTO Vendor VALUES (7, 'SmartDevices Co.', 7777777777, 'smartdevices@
example.com');

1 row created.

SQL> INSERT INTO Vendor VALUES (8, 'TechUniverse Ltd.', 4444444444, 'techuniverse
@example.com');

1 row created.

SQL> INSERT INTO Vendor VALUES (9, 'Digital', 6666666666, 'digital@example.com');


1 row created.
```

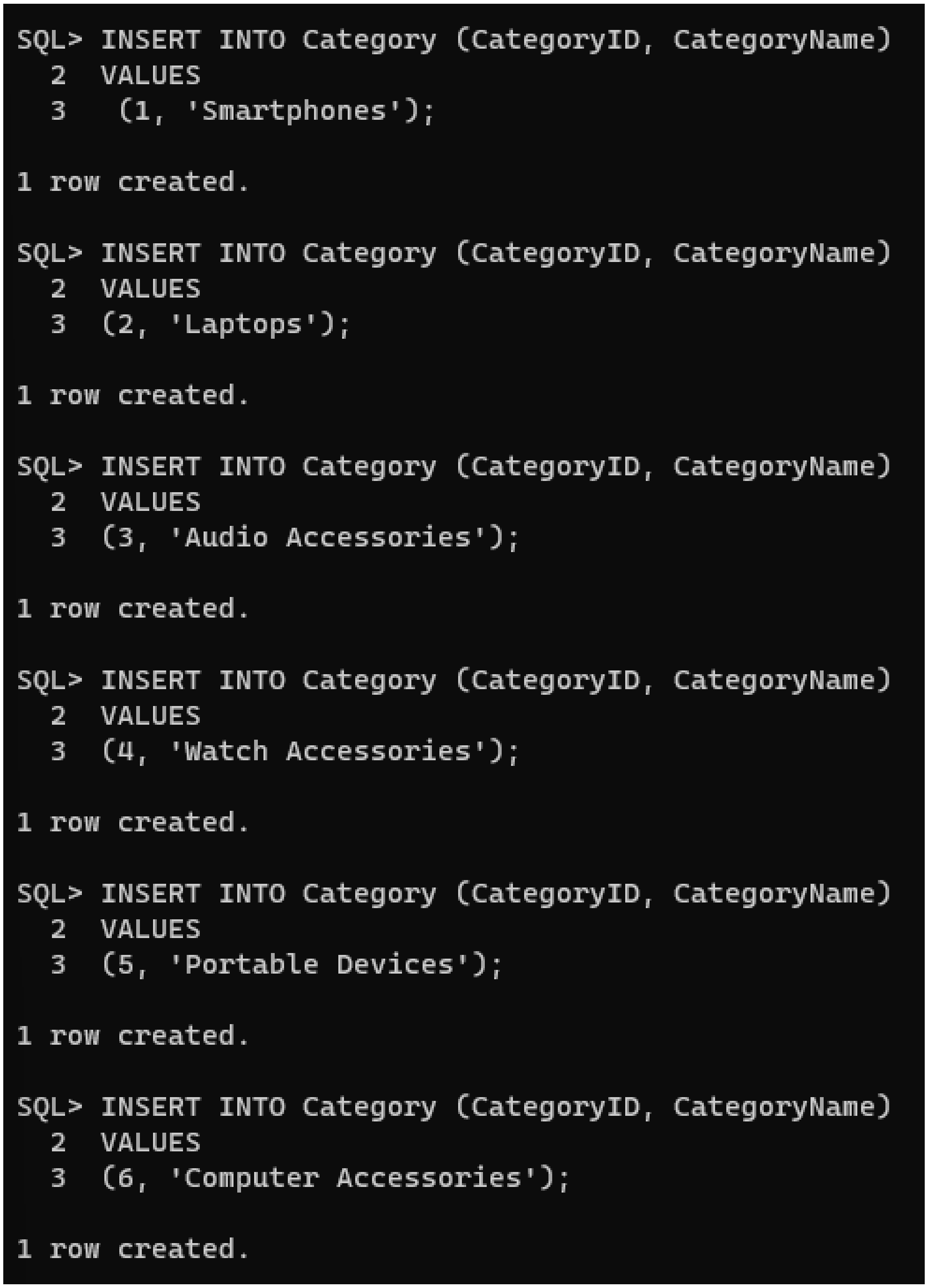
Figure 28 Insert value in Vendor



```
SQL> select * from Vendor;

  VENDORID FULLNAME                                           PHONENUMBER EMAILADDRESS
---------- -------------------------------------------------- ----------- ------------------------------
         1 ABC Electronics                                     1234567890 abc@example.com
         2 XYZ Tech Solutions                                  9876543210 xyztech@example.com
         3 GadgetHub Inc.                                      5555555555 gadgethub@example.com
         4 TechWonders Ltd.                                    1111111111 techwonders@example.com
         5 Innovate Gadgets                                    9998887777 innovate@example.com
         6 ElectroCraft                                        3333333333 electrocraft@example.com
         7 SmartDevices Co.                                    7777777777 smartdevices@example.com
         8 TechUniverse Ltd.                                   4444444444 techuniverse@example.com
         9 DigitalSolutions                                    6666666666 digitalsolutions@example.com
        10 Gizmo Innovations                                   2222222222 gizmo@example.com

10 rows selected.
```

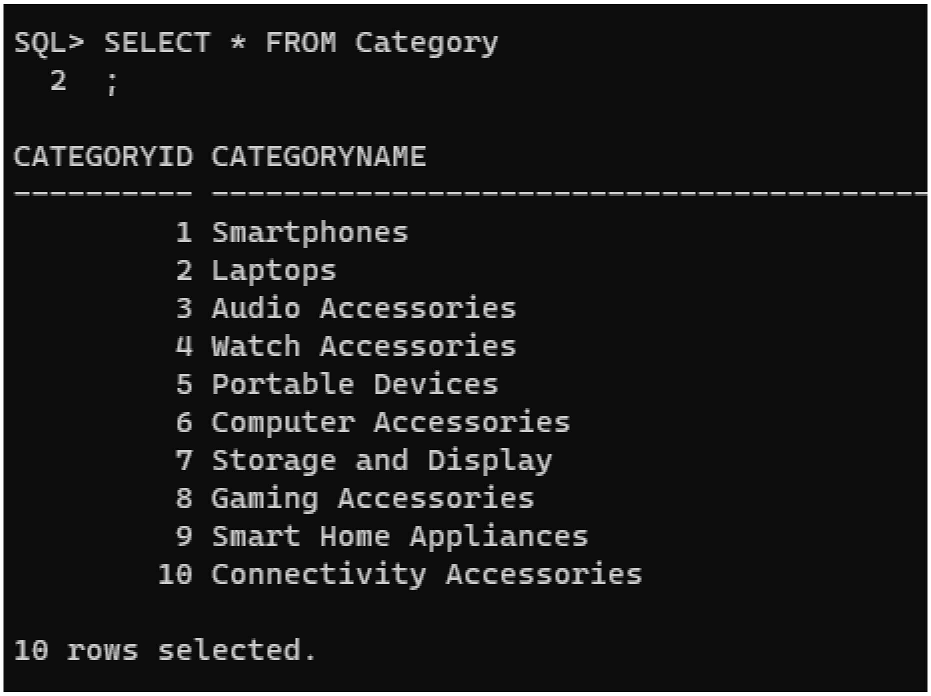
Figure 29 Value in Vendor table

Insert value in Category



```
SQL> INSERT INTO Category (CategoryID, CategoryName)
     VALUES
      (1, 'Smartphones');

1 row created.

SQL> INSERT INTO Category (CategoryID, CategoryName)
     VALUES
     (2, 'Laptops');

1 row created.

SQL> INSERT INTO Category (CategoryID, CategoryName)
     VALUES
     (3, 'Audio Accessories');

1 row created.

SQL> INSERT INTO Category (CategoryID, CategoryName)
     VALUES
     (4, 'Watch Accessories');

1 row created.

SQL> INSERT INTO Category (CategoryID, CategoryName)
     VALUES
     (5, 'Portable Devices');

1 row created.

SQL> INSERT INTO Category (CategoryID, CategoryName)
     VALUES
     (6, 'Computer Accessories');

1 row created.
```

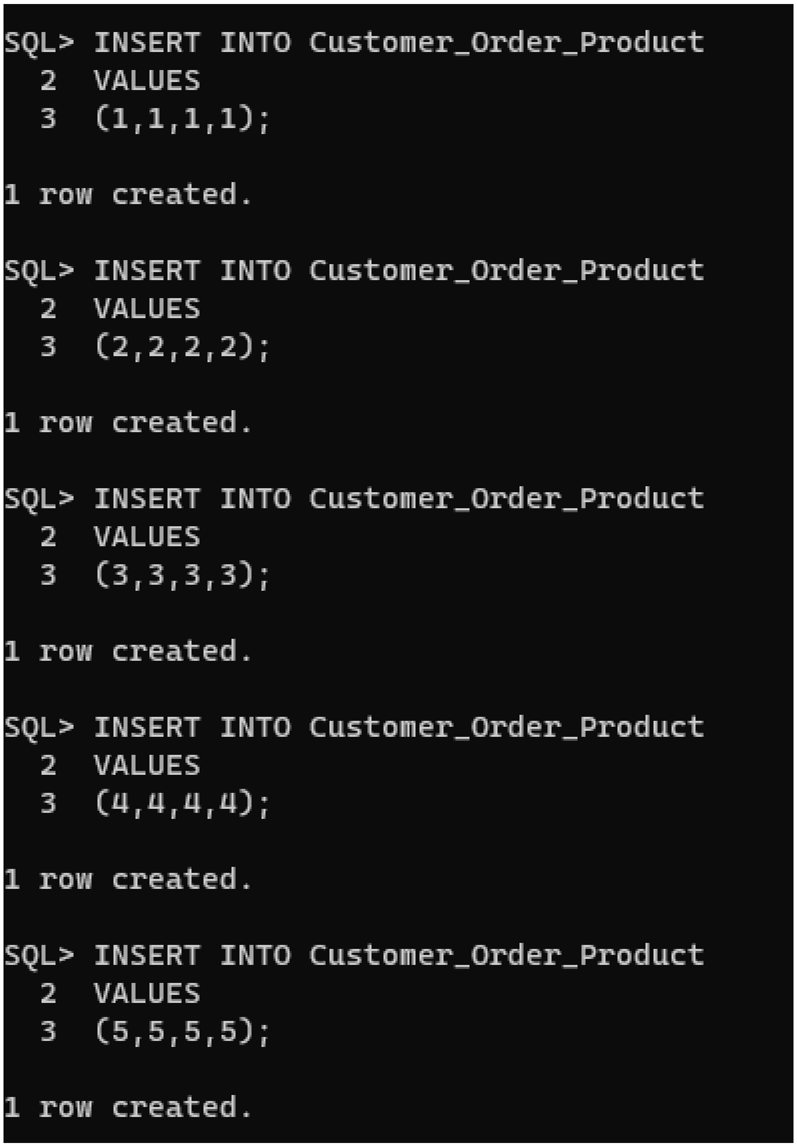
Figure 30 Insert value in Category



```
SQL> SELECT * FROM Category
  2  ;

CATEGORYID CATEGORYNAME
---------- ----------------------------------------
         1 Smartphones
         2 Laptops
         3 Audio Accessories
         4 Watch Accessories
         5 Portable Devices
         6 Computer Accessories
         7 Storage and Display
         8 Gaming Accessories
         9 Smart Home Appliances
        10 Connectivity Accessories

10 rows selected.
```

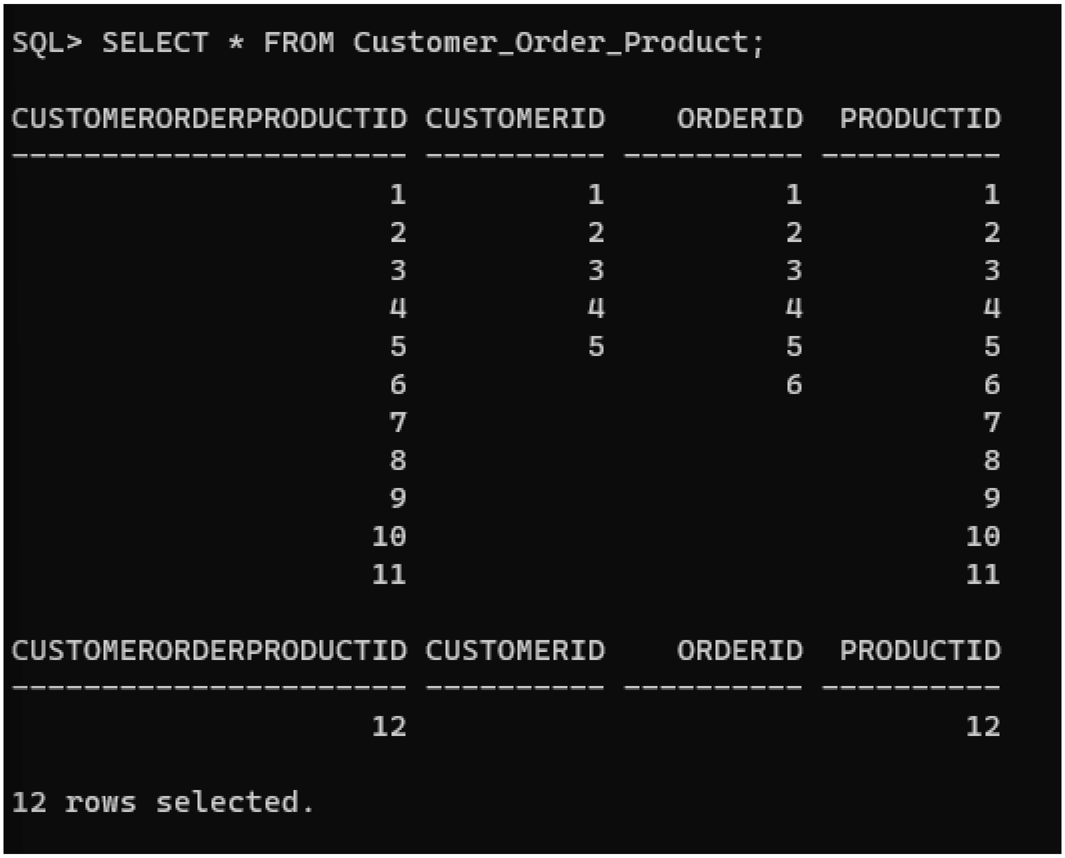
Figure 31 Values in Category

Insert value in Customer_Order_Product



```
SQL> INSERT INTO Customer_Order_Product
  2  VALUES
  3  (1,1,1,1);

1 row created.

SQL> INSERT INTO Customer_Order_Product
  2  VALUES
  3  (2,2,2,2);

1 row created.

SQL> INSERT INTO Customer_Order_Product
  2  VALUES
  3  (3,3,3,3);

1 row created.

SQL> INSERT INTO Customer_Order_Product
  2  VALUES
  3  (4,4,4,4);

1 row created.

SQL> INSERT INTO Customer_Order_Product
  2  VALUES
  3  (5,5,5,5);

1 row created.
```

Figure 32 Insert Value in Customer_Order_Product



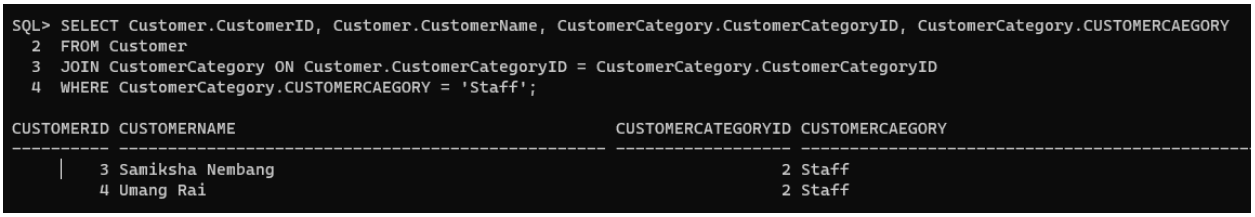
```
SQL> SELECT * FROM Customer_Order_Product;

CUSTOMERORDERPRODUCTID CUSTOMERID    ORDERID  PRODUCTID
---------------------- ---------- ---------- ----------
                     1          1          1          1
                     2          2          2          2
                     3          3          3          3
                     4          4          4          4
                     5          5          5          5
                     6                     6          6
                     7                                7
                     8                                8
                     9                                9
                    10                               10
                    11                               11

CUSTOMERORDERPRODUCTID CUSTOMERID    ORDERID  PRODUCTID
---------------------- ---------- ---------- ----------
                    12                               12

12 rows selected.
```

Figure 33 Values in Customer_Order_Product

6. Database Querying

6.1. Information Query

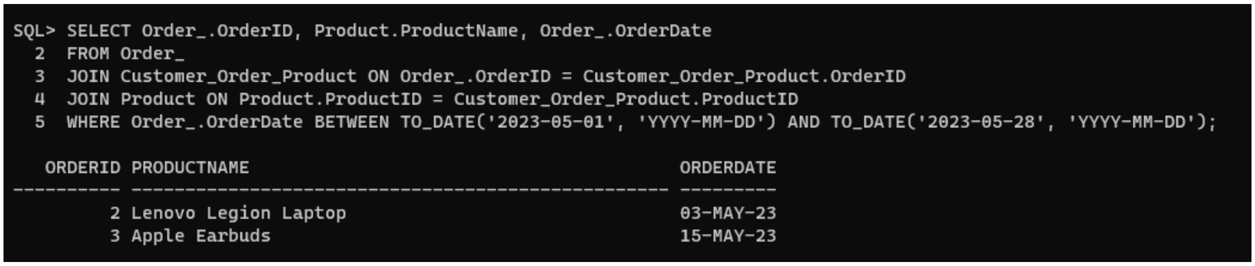
1) List all the customers that are also staff of the company.



```
SQL> SELECT Customer.CustomerID, Customer.CustomerName, CustomerCategory.CustomerCategoryID, CustomerCategory.CUSTOMERCAEGORY
     FROM Customer
     JOIN CustomerCategory ON Customer.CustomerCategoryID = CustomerCategory.CustomerCategoryID
     WHERE CustomerCategory.CUSTOMERCAEGORY = 'Staff';

CUSTOMERID CUSTOMERNAME                                       CUSTOMERCATEGORYID CUSTOMERCAEGORY
---------- -------------------------------------------------- ------------------ --------------------------------------------------
         3 Samiksha Nembang                                                    2 Staff
         4 Umang Rai                                                           2 Staff
```

Figure 34 All the customers who are also the Staff of the company

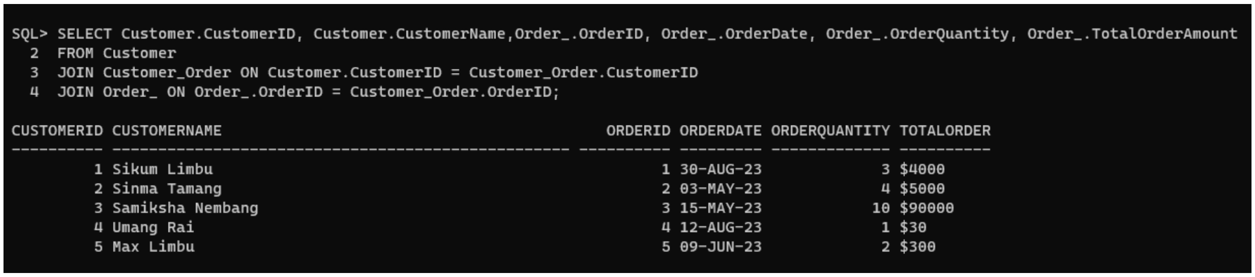
2) List all the orders made for any particular product between the dates 01-05-2023 till 28 05-2023.



```
SQL> SELECT Order_.OrderID, Product.ProductName, Order_.OrderDate
     FROM Order_
     JOIN Customer_Order_Product ON Order_.OrderID = Customer_Order_Product.OrderID
     JOIN Product ON Product.ProductID = Customer_Order_Product.ProductID
     WHERE Order_.OrderDate BETWEEN TO_DATE('2023-05-01', 'YYYY-MM-DD') AND TO_DATE('2023-05-28', 'YYYY-MM-DD');

   ORDERID PRODUCTNAME                                        ORDERDATE
---------- -------------------------------------------------- ---------
         2 Lenovo Legion Laptop                               03-MAY-23
         3 Apple Earbuds                                      15-MAY-23
```

Figure 35 orders made for any particular product between the dates 01-05-2023 till 28-05-2023

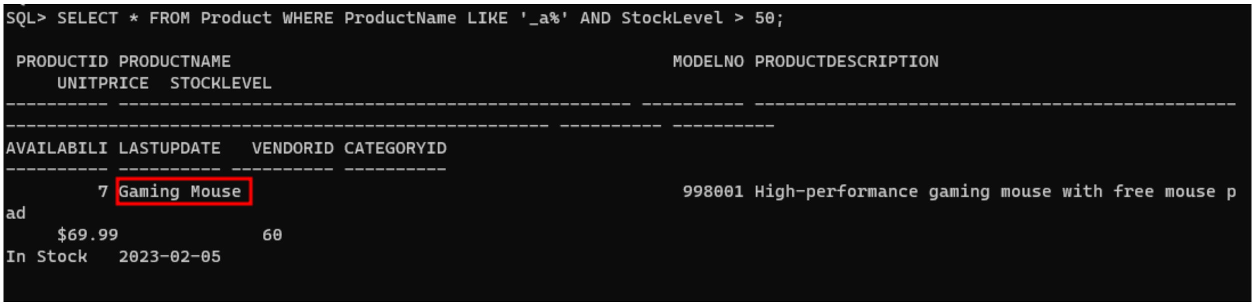
3) List all the customers with their order details and also the customers who have not ordered any products yet.



```
SQL> SELECT Customer.CustomerID, Customer.CustomerName,Order_.OrderID, Order_.OrderDate, Order_.OrderQuantity, Order_.TotalOrderAmount
     FROM Customer
     JOIN Customer_Order ON Customer.CustomerID = Customer_Order.CustomerID
     JOIN Order_ ON Order_.OrderID = Customer_Order.OrderID;

CUSTOMERID CUSTOMERNAME                                          ORDERID ORDERDATE ORDERQUANTITY TOTALORDER
---------- -------------------------------------------------- ---------- --------- ------------- ----------
         1 Sikum Limbu                                                 1 30-AUG-23             3 $4000
         2 Sinma Tamang                                                2 03-MAY-23             4 $5000
         3 Samiksha Nembang                                            3 15-MAY-23            10 $90000
         4 Umang Rai                                                   4 12-AUG-23             1 $30
         5 Max Limbu                                                   5 09-JUN-23             2 $300
```

Figure 36 customers with their order details and also the customers who have not ordered

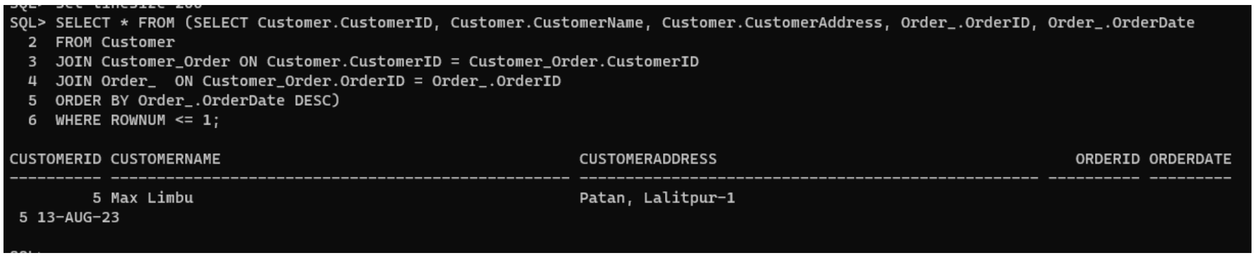
4) List all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.



```
SQL> SELECT * FROM Product WHERE ProductName LIKE '_a%' AND StockLevel > 50;

 PRODUCTID PRODUCTNAME                                           MODELNO PRODUCTDESCRIPTION
     UNITPRICE  STOCKLEVEL
---------- -------------------------------------------------- ---------- ----------------------------------------------
-------------------------------------------------- ---------- ----------
AVAILABILI LASTUPDATE   VENDORID CATEGORYID
---------- ---------- ---------- ----------
         7 Gaming Mouse                                           998001 High-performance gaming mouse with free mouse p
ad
     $69.99              60
In Stock   2023-02-05
```

Figure 37 all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.

5) Find out the customer who has ordered recently.

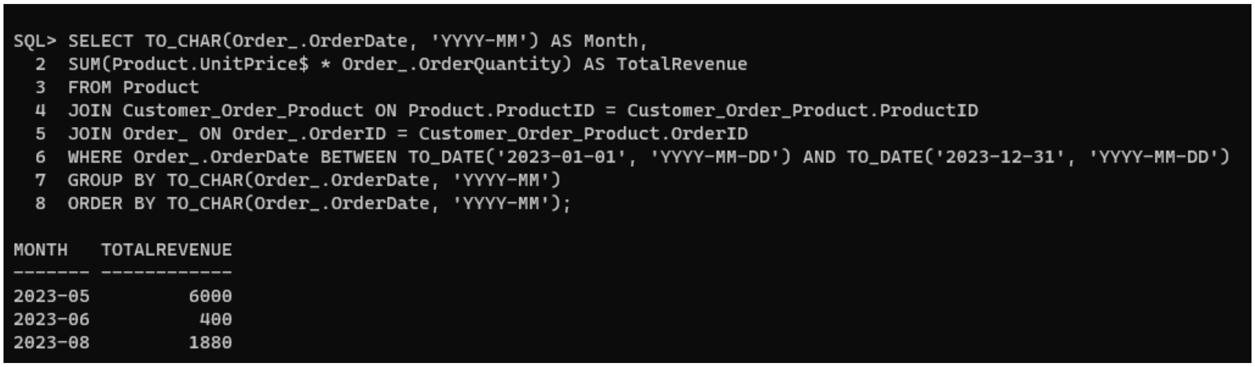


```
SQL> SELECT * FROM (SELECT Customer.CustomerID, Customer.CustomerName, Customer.CustomerAddress, Order_.OrderID, Order_.OrderDate
     FROM Customer
     JOIN Customer_Order ON Customer.CustomerID = Customer_Order.CustomerID
     JOIN Order_  ON Customer_Order.OrderID = Order_.OrderID
     ORDER BY Order_.OrderDate DESC)
     WHERE ROWNUM <= 1;

CUSTOMERID CUSTOMERNAME                                       CUSTOMERADDRESS                                       ORDERID ORDERDATE
---------- -------------------------------------------------- -------------------------------------------------- ---------- ---------
         5 Max Limbu                                          Patan, Lalitpur-1
 5 13-AUG-23
```

Figure 38 Customer who has ordered recently

6.2. Transaction Query

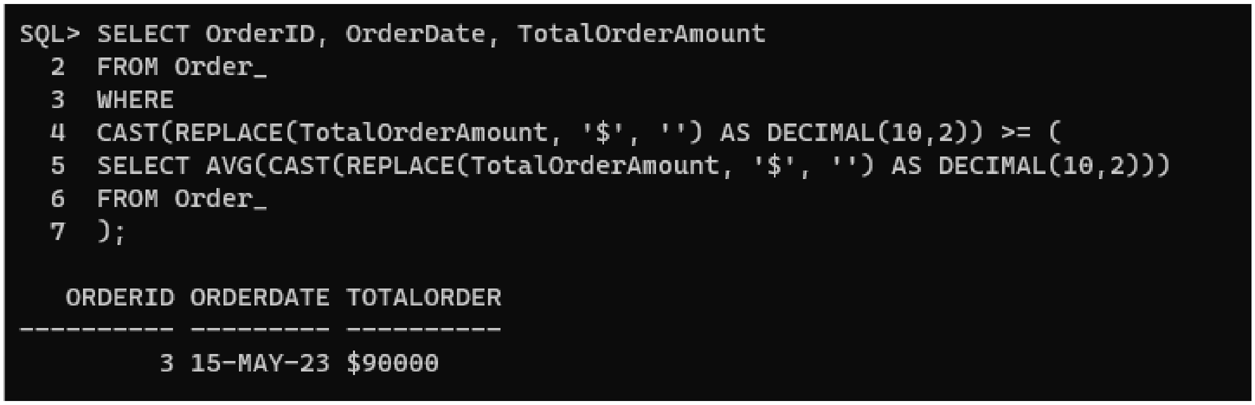
1) Show the total revenue of the company for each month.



```
SQL> SELECT TO_CHAR(Order_.OrderDate, 'YYYY-MM') AS Month,
     SUM(Product.UnitPrice$ * Order_.OrderQuantity) AS TotalRevenue
     FROM Product
     JOIN Customer_Order_Product ON Product.ProductID = Customer_Order_Product.ProductID
     JOIN Order_ ON Order_.OrderID = Customer_Order_Product.OrderID
     WHERE Order_.OrderDate BETWEEN TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD')
     GROUP BY TO_CHAR(Order_.OrderDate, 'YYYY-MM')
     ORDER BY TO_CHAR(Order_.OrderDate, 'YYYY-MM');

MONTH   TOTALREVENUE
------- ------------
2023-05         6000
2023-06          400
2023-08         1880
```

Figure 39 the total revenue of the company for each month

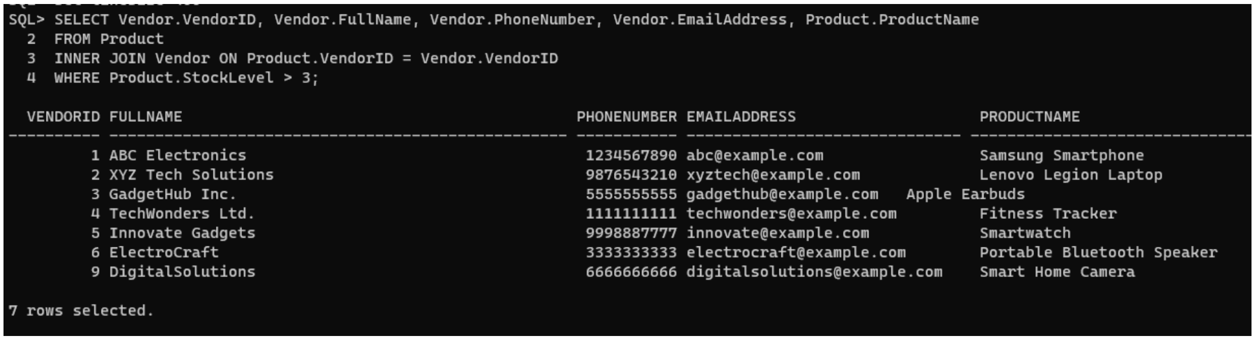
2) Find those orders that are equal or higher than the average order total value.



```
SQL> SELECT OrderID, OrderDate, TotalOrderAmount
     FROM Order_
     WHERE
     CAST(REPLACE(TotalOrderAmount, '$', '') AS DECIMAL(10,2)) >= (
     SELECT AVG(CAST(REPLACE(TotalOrderAmount, '$', '') AS DECIMAL(10,2)))
     FROM Order_
     );

   ORDERID ORDERDATE TOTALORDER
---------- --------- ----------
         3 15-MAY-23 $90000
```

Figure 40 orders that are equal or higher than the average order total value

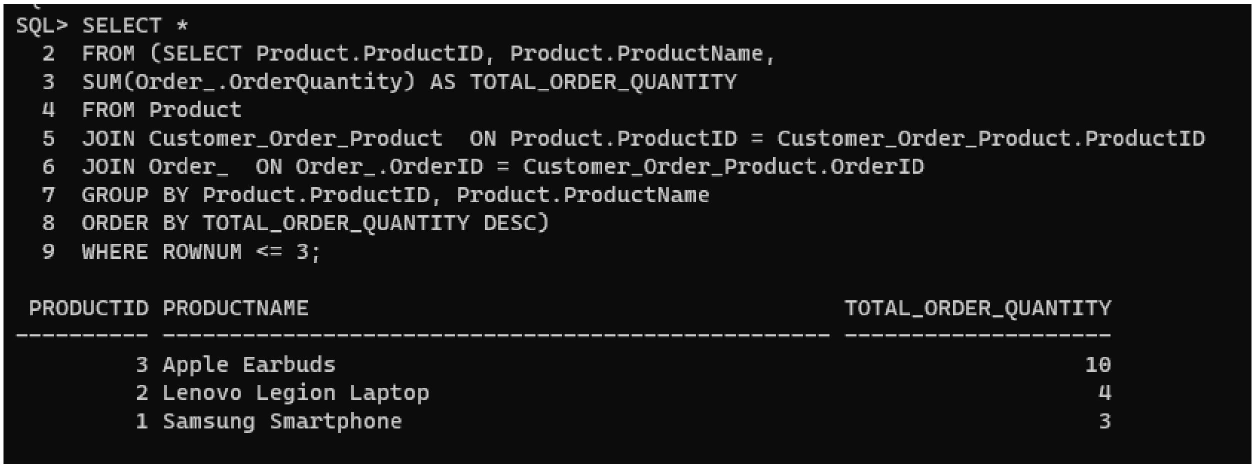
3) List the details of vendors who have supplied more than 3 products to the company.



```
SQL> SELECT Vendor.VendorID, Vendor.FullName, Vendor.PhoneNumber, Vendor.EmailAddress, Product.ProductName
     FROM Product
     INNER JOIN Vendor ON Product.VendorID = Vendor.VendorID
     WHERE Product.StockLevel > 3;

  VENDORID FULLNAME                                           PHONENUMBER EMAILADDRESS                   PRODUCTNAME
---------- -------------------------------------------------- ----------- ------------------------------ ------------------------------
         1 ABC Electronics                                     1234567890 abc@example.com                Samsung Smartphone
         2 XYZ Tech Solutions                                  9876543210 xyztech@example.com            Lenovo Legion Laptop
         3 GadgetHub Inc.                                      5555555555 gadgethub@example.com   Apple Earbuds
         4 TechWonders Ltd.                                    1111111111 techwonders@example.com        Fitness Tracker
         5 Innovate Gadgets                                    9998887777 innovate@example.com           Smartwatch
         6 ElectroCraft                                        3333333333 electrocraft@example.com       Portable Bluetooth Speaker
         9 DigitalSolutions                                    6666666666 digitalsolutions@example.com   Smart Home Camera

7 rows selected.
```

Figure 41 the details of vendors who have supplied more than 3 products to the company.

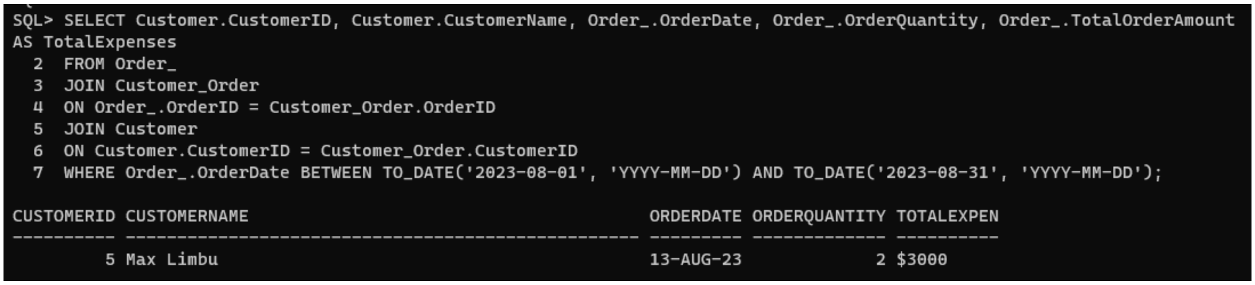
4) Show the top 3 product details that have been ordered the most.



```
SQL> SELECT *
  FROM (SELECT Product.ProductID, Product.ProductName,
  SUM(Order_.OrderQuantity) AS TOTAL_ORDER_QUANTITY
  FROM Product
  JOIN Customer_Order_Product  ON Product.ProductID = Customer_Order_Product.ProductID
  JOIN Order_  ON Order_.OrderID = Customer_Order_Product.OrderID
  GROUP BY Product.ProductID, Product.ProductName
  ORDER BY TOTAL_ORDER_QUANTITY DESC)
  WHERE ROWNUM <= 3;

 PRODUCTID PRODUCTNAME                                        TOTAL_ORDER_QUANTITY
---------- -------------------------------------------------- --------------------
         3 Apple Earbuds                                                      10
         2 Lenovo Legion Laptop                                                4
         1 Samsung Smartphone                                                  3
```

Figure 42 the top 3 product details that have been ordered the most

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



```
SQL> SELECT Customer.CustomerID, Customer.CustomerName, Order_.OrderDate, Order_.OrderQuantity, Order_.TotalOrderAmount
AS TotalExpenses
  FROM Order_
  JOIN Customer_Order
  ON Order_.OrderID = Customer_Order.OrderID
  JOIN Customer
  ON Customer.CustomerID = Customer_Order.CustomerID
  WHERE Order_.OrderDate BETWEEN TO_DATE('2023-08-01', 'YYYY-MM-DD') AND TO_DATE('2023-08-31', 'YYYY-MM-DD');

CUSTOMERID CUSTOMERNAME                                       ORDERDATE ORDERQUANTITY TOTALEXPEN
---------- -------------------------------------------------- --------- ------------- ----------
         5 Max Limbu                                          13-AUG-23             2 $3000
```

Figure 43 the customer who has ordered the most in August with his/her total spending on that month

7. Critical Evaluation

7.1. Critical Evaluation of module, its usage and relation with other subject

The module database which is under evaluation play a vital role in developing understanding of it subject matter. The way it connects with other related areas demonstrates its importance and practical usefulness. In addition to providing students with fundamental knowledge, the module acts as a link, forming relationships with related subjects. Students are able to integrate knowledge from multiple perspectives, which improves their overall understanding of the field. The usefulness of the module is derived from its capacity to work in combination with other courses, enhancing the overall educational experience and equipping students with the knowledge and skills necessary to comprehend the academic domain in a complex and related manner.

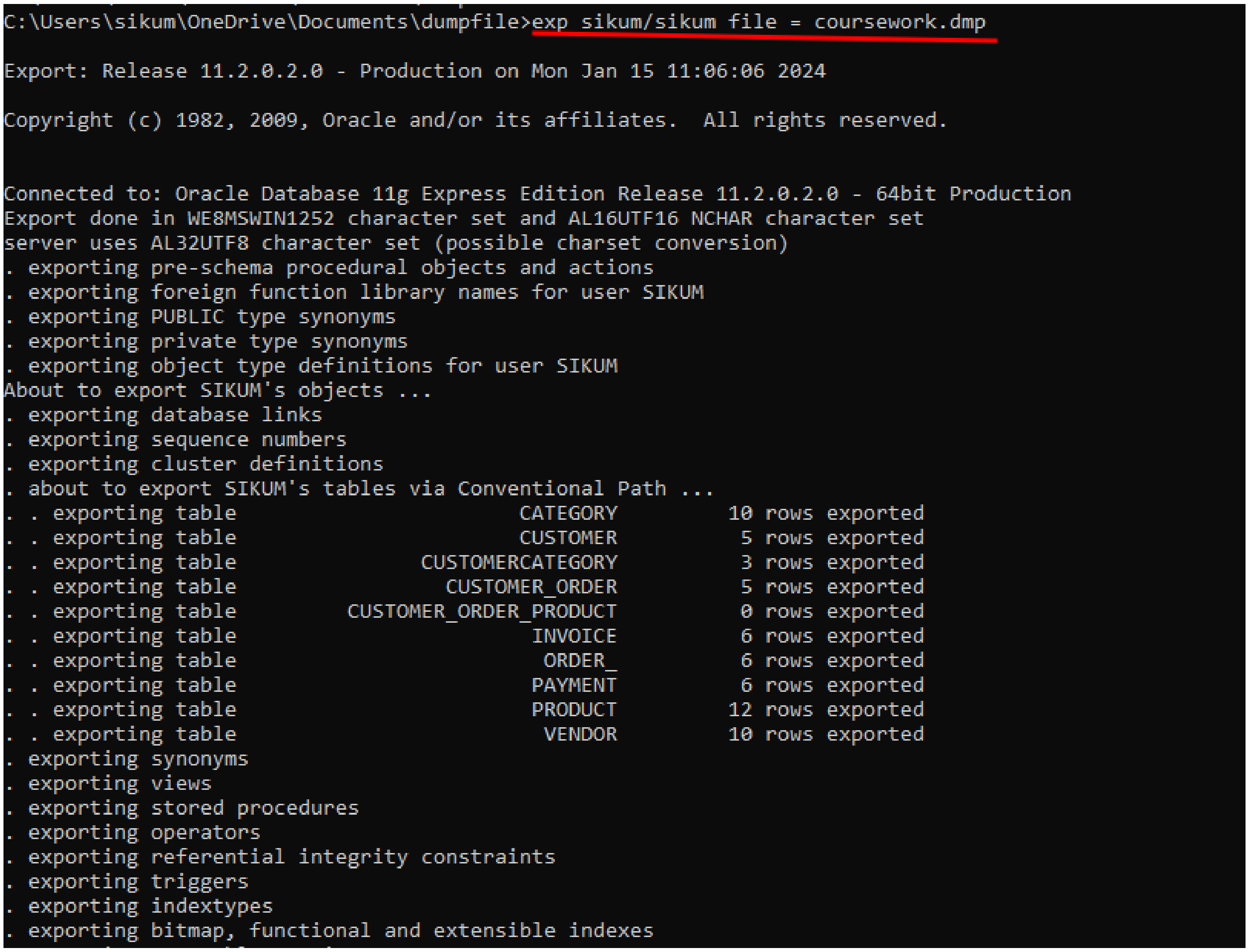
7.2. Critical Assessment of Coursework

There were many challenges while completing this coursework, it took me a whole month to get used to the phrase "normalisation," which seemed so unclear. Plenty of mistakes and lots of chances for improvement were there and Huge thanks to Siddharth Sir, my tutorial teacher, who helped me manage the confusion with patience. With his assistance, the confusing normalisation became reasonable for me.

The next challenge for me were querying, it was difficult to choose the appropriate phrases and ensure that everything made sense in sql command prompt. By the end, I had completed the coursework and gained an understanding of how databases function in the real world. Although it was a difficult process, I learned a many things about databases and database management which will be great help for me in the future.

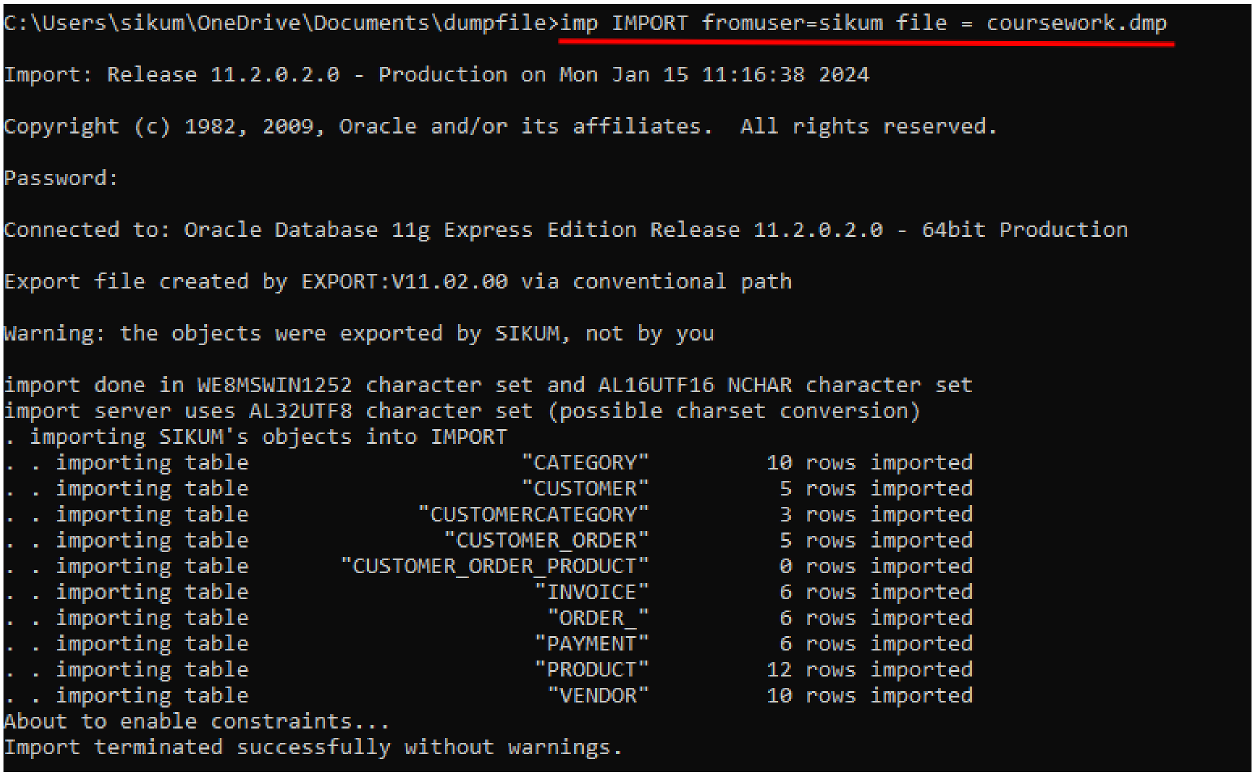
8. Drop Query and Database Dump file creation

8.1 Database Dump File Creation



```
C:\Users\sikum\OneDrive\Documents\dumpfile>exp sikum/sikum file = coursework.dmp

Export: Release 11.2.0.2.0 - Production on Mon Jan 15 11:06:06 2024

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


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user SIKUM
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user SIKUM
About to export SIKUM's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export SIKUM's tables via Conventional Path ...
. . exporting table                       CATEGORY         10 rows exported
. . exporting table                       CUSTOMER          5 rows exported
. . exporting table               CUSTOMERCATEGORY          3 rows exported
. . exporting table                 CUSTOMER_ORDER          5 rows exported
. . exporting table         CUSTOMER_ORDER_PRODUCT          0 rows exported
. . exporting table                        INVOICE          6 rows exported
. . exporting table                         ORDER_          6 rows exported
. . exporting table                        PAYMENT          6 rows exported
. . exporting table                        PRODUCT         12 rows exported
. . exporting table                         VENDOR         10 rows exported
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
```

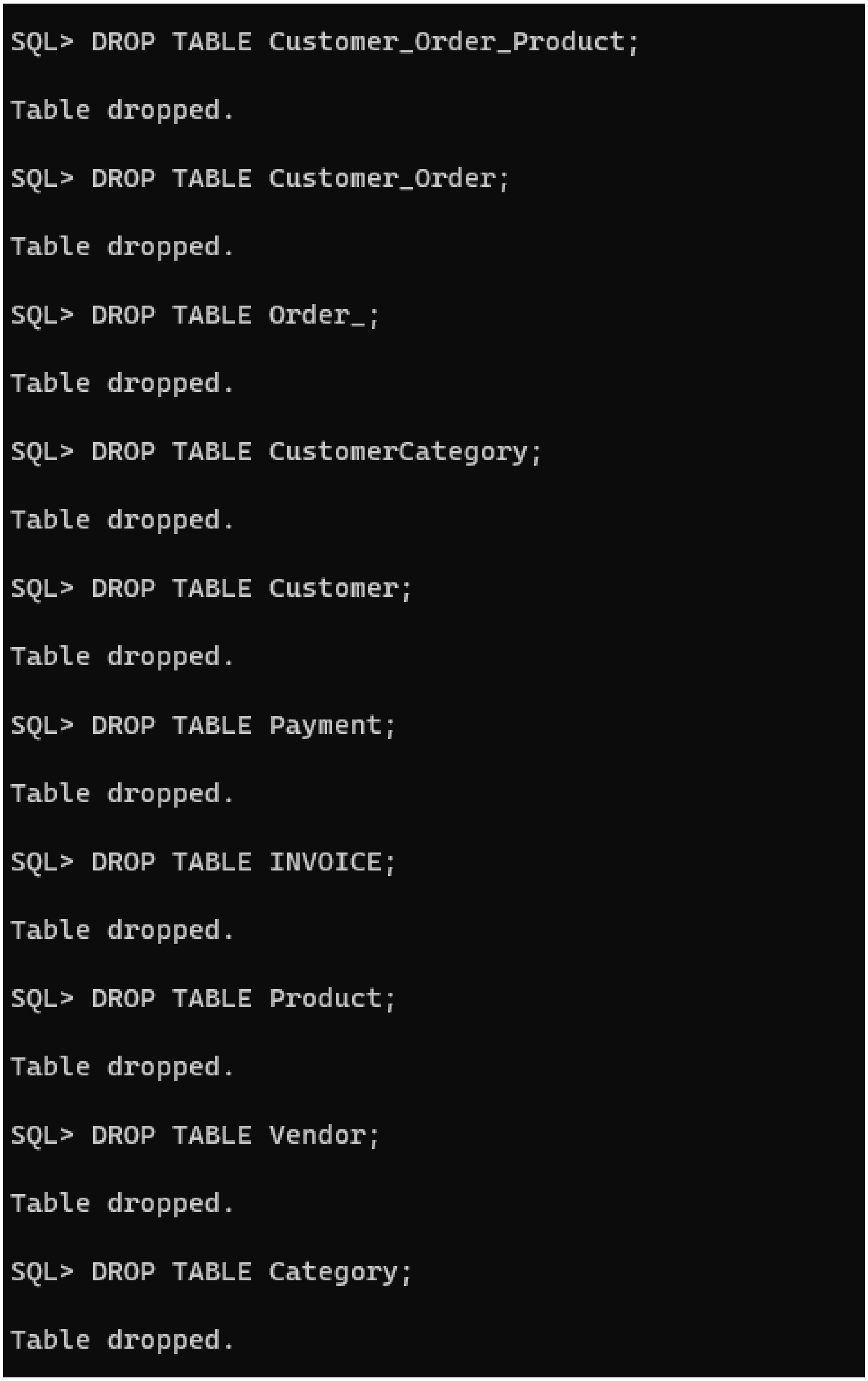
Figure 44 Creating Dump file 1.1



```
C:\Users\sikum\OneDrive\Documents\dumpfile>imp IMPORT fromuser=sikum file = coursework.dmp

Import: Release 11.2.0.2.0 - Production on Mon Jan 15 11:16:38 2024

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

Password:

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

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

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

import done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
import server uses AL32UTF8 character set (possible charset conversion)
. importing SIKUM's objects into IMPORT
. . importing table                     "CATEGORY"         10 rows imported
. . importing table                     "CUSTOMER"          5 rows imported
. . importing table             "CUSTOMERCATEGORY"          3 rows imported
. . importing table               "CUSTOMER_ORDER"          5 rows imported
. . importing table       "CUSTOMER_ORDER_PRODUCT"          0 rows imported
. . importing table                      "INVOICE"          6 rows imported
. . importing table                       "ORDER_"          6 rows imported
. . importing table                      "PAYMENT"          6 rows imported
. . importing table                      "PRODUCT"         12 rows imported
. . importing table                       "VENDOR"         10 rows imported
About to enable constraints...
Import terminated successfully without warnings.
```

Figure 45 Creating dump file 1.2

8.2. Drop Table



```
SQL> DROP TABLE Customer_Order_Product;

Table dropped.

SQL> DROP TABLE Customer_Order;

Table dropped.

SQL> DROP TABLE Order_;

Table dropped.

SQL> DROP TABLE CustomerCategory;

Table dropped.

SQL> DROP TABLE Customer;

Table dropped.

SQL> DROP TABLE Payment;

Table dropped.

SQL> DROP TABLE INVOICE;

Table dropped.

SQL> DROP TABLE Product;

Table dropped.

SQL> DROP TABLE Vendor;

Table dropped.

SQL> DROP TABLE Category;

Table dropped.
```

Figure 46 dropped Tables