CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: Sujal Nakarmi

London Met ID: 22067808

College Id: NP01CP4A220192

Group: L2C5

Assignment Submission Date: 15 January, 2024

Word Count: 6442

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 and its forte

This project includes the database design for an online ecommerce. The name of the online ecommerce is Gadget Emporium. The main desire of the system is to keep track of all the customer, order, and products. Gadget Emporium stores many gadgets required for the customer where people can browse for varieties of products. The customer orders to purchase the products. The customer can browse any product of their choice and make an order if there is any doubt about the product, they can view the review and ask help from the customer service. The products include gadget like Smart Phone, Smart TV, Smart Watch, Washing Machines, Cameras etc. The revenue is generated by the sales of the products and a discount rate is imposed based on the customer. There is a stock availability check so that the company may run out of stock and disappoint the customer. The company future plan and goals is to expand the business in various places and becomes the no 1 trusted ecommerce as having different showrooms for various brands such as Apple, Samsung, LG, Panasonic which can also help the customer convenient to browse and order specific type product. It aims to deliver the products as soon as possible for customer satisfaction. There are some offers which occurs every year in the festive season which can encourage and attract people to purchase the products such as 15% off, Buy One Get One Free and Exchange Old gadget with new Gadget by adding some amount. The ecommerce platform is available in all the devices like laptop, mobile phone, tablet which also help in the customer satisfaction. Advertisement of the ecommerce will be on the website where every year on a specifc date there would be a lucky draw where customer can awarded with some awesome rewards.

This project includes the normalization method to reduce data redundancy. Amount of Data should be removed. We use normalization as design has to show relevant relationship between Entitles.

1.2 Current Business Activities and Operations

The system stores the Customer details in the customer database. Details such as customer id, customer name, customer address, and customer categories. These details are confidential. Therefore, database is used to store it more securely. Customer Address are used for the delivery purpose. The customer are categorized into VIP, Staff and Regular. The VIP are the customers who get the most discount rate and the staff as who works can also be customer getting some rate of discount, The regular customers are given the least discount order, but they can increase the rate by promoting the membership. The order details are stored in the order database details like order date of the order, total order amount of each order id, payment options like credit/debit card, cash on delivery, e-wallet to facilitate secure and seamless transaction. The product details are stored in product database details like product id, product name, product description, product categories, unit price. Stock Level and Availability Status is checked to keep the record or sales of the product. The vendors details are kept in the vendor database where details like vendor id, vendor name, vendor address, vendor address to request for the supply of the product and to keep record of the products supplied by the vendors.

If the product is damaged unfortunately, the system will replace the same product with the new one for the customer convenience. But in some cases, if the customer wants to refund their money, the refund service is also available. An invoice will be generated after the customer confirms his/her order which include details of the customer, order, payment along with the discount as proof product has been sold to the customer from this ecommerce.

After receiving products and after purchasing customer can give review about the system which will promote the company. Customers can also mention the areas where they want some improvement and also suggest some new ideas.

2. List of Business Rules

Business Rules are used to demonstrate the representation and manipulation of data in many different aspects. (Raipurkar & Deokate, 2012) The first step for a database design is to set up Business Rule. (SolutionGlobal, 2023)

- Customer can browse and order many gadgets of their choice and the order details must be recorded by the system.
- Each order can have multiple product and one particular product can be included into multiple order which placed by the Customer which defines Many to Many Relationships.
- Customers can be categorized into VIP, REGULAR and STAFF which also determines the discount rate such as 10% to VIP, 0 % to REGULAR and 5% to STAFF.
- Each product is supplied by one specific vendor and one vendor can supply many products to the company which defines One to Many Relationships.
- From the payment options like Cash on Delivery, e-wallet, credit card, debit card one of them must be recorded in the order details to facilitate secure and seamless transaction.
- Each category can have many products, but one product is included in one category.
- For keeping track of the products, the product should have stock quantity or availability status so that the products might not run out.
- Customer Address should be maintained or recorded properly and should be check which will help in the delivery process.
- An invoice must be generated after the payment of the confirmed product purchased by the customer reflecting deducted discount.

Assumptions

- One Customer may purchase many products, but each product must be associated with one specific customer.
- One order may have many products and one specific product may be included in one order.
- Each product must be supplied by one specific vendor and one specific vendor must supply many products to the company.
- Discount Depends upon customer categories as customer is categorized into three category which are VIP, REGULAR and STAFF and discount depends on the title entitled to each Customer.
- Order date, Total order amount and Payment Option depends upon the order id.
- Order Quantity and Line Total depends upon both product id and order id as order give the number of the product customer desire and product determines the description of the product. Line total is calculated by Unit price from the product and order quantity from the order.
- Product Name, Product Description, Product Categories, Unit Price, Stock Level Depends upon product id.
- Vendor name, address and contact number is dependent on Vendor id.

3. Identification of the Entities and Attributes

Entities are a real-world object which are stored in the database. Data representation and managing is the role of entity, and it should be separable from the group. (JavaTpoint, 2023)

Attributes are the properties which describes the entity. It is a table column which is a database component. The types of attributes are Composite attribute, Multivalued attribute, Key attribute, and Derived attribute. (Rouse, 2023). Integer, string, or date is the datatype of attributes. (Sugandhi, 2023)

3.1 Entities

The entities which I identified from the study case are listed below:

- Customer
- Order
- Product
- Vendor

3.2 Relationship

The following table shows the relationships of Entities with each other.

Entities	Relationship
Customer and Order	One to Many
Order and Product	Many to Many
Product and Vendor	Many to One

Table 1 Relation of Entities

3.3 Entities and Attributes

Entities	Attribute
Customer	Customer Id Customer Name Customer Address Customer Categories Discount

Table 2 Customer Entity

Entities	Attribute
Order	Order Id Order Date Total Order Amount Payment Option

Table 3 Order Entity

Entities	Attribute
Product	Product Id Product Name Product Description Product Categories Unit Price Stock Level Line Total Order Quantity

Table 4 Product Entity

Entities	Attribute
Vendor	Vendor Id Vendor Name Vendor Address Vendor Contact Number

Table 5 Vendor Table

3.4 Datatype of Attribute

For Customer

Attribute	Datatype	Constraints	Description
Customer Id	Number	Primary Key, Not Null	This field stores the id of customer which must be unique.
Customer Name	Varchar (20)		This field stores the name of the customer.
Customer Address	Varchar (50)		This field stores the address of the customer.
Customer Categories	Varchar (10)		This field store the category of customer which are Regular (R), Staff (S) and VIP (V).
Discount	Number		This field store the discount rate given to the customer.

Table 6 Data Dictionary for Customer

For Order

Attribute	Datatype	Constraints	Description
Order id	Number	Primary Key, Not Null	This field stores the id of order which must be unique.
Order date	Date		This field stores the date of the order being processed.

Total Order Amount	Number		This field stores the sum of all the product order made by one customer.
Payment Option	Varchar (30)		This field store the method of the payment done by the customer like Cash on delivery, Credit Card, Debit Card, and e-wallet.
Customer Id	Number	Foreign Key	This field store the id of the customer who have made the order.
Product Id	Number	Foreign Key	This field store the id of the product being ordered.

Table 7 Data Dictionary For Order

For Product

Attribute	Datatype	Constraints	Description
Product Id	Number	Primary Key, Not Null	This field stores the id of each product which must be unique.
Product Name	Varchar (20)		This field stores the name of the product.
Product Description	Varchar (50)		This field stores the description of the product.
Product Categories	Varchar (20)		This field stores the category of product like which type of product.
Unit Price	Number		This field store price of one specific product.
Stock Level	Number		This field store the quantity of product available in the stock.
Order Quantity	Number		This field stores the quantity of product ordered by the customer.

Line Total	Number		This field stores the value after the multiplication of unit price and order quantity.
Order Id	Number	Foreign Key	This field store the id of the order for products.
Vendor Id	Number	Foreign Key	This field stores the id of the vendor who have supplied the products.

Table 8 Data Dictionary For Product

For Vendor

Attribute	Datatype	Constraints	Description
Vendor Id	Number	Primary Key, Not Null	This field stores the id of vendor which must be unique.
Vendor Name	Varchar (20)		This field stores the name of the vendor.
Vendor Address	Varchar (50)		This field stores the address of the vendor.
Vendor Contact Number	Number		This field store the phone number of vendor.

Table 9 Data Dictionary for Vendor

4. Initial ERD

ERD is the design tool which shows the relationship between many entities. The full form of ERD is Entity Relationship Diagram. (Gibbs, 2021) ERD design is based on logic and business rule. Entities relating to each other in the system is decorated by ERD. (LucidChart, 2023)

4.1 List of the created objects

The objects Entities and Attribute which are created are as follows:

Entities	Attribute
Customer	Customer Id Customer Name Customer Address Customer Categories Discount

Table 10 Customer Table

Entities	Attribute
Order	Order Id Order Date Total Order Amount Payment Option

Table 11 Order Table

Entities	Attribute
Product	Product Id Product Name Product Description Product Categories Unit Price Stock Level Line Total Order Quantity

Table 12 Product Table

Entities	Attribute
Vendor	Vendor Id Vendor Name Vendor Address Vendor Contact Number

Table 13 Vendor Table

4.2 Identification and representation of Primary key and Foreign key

Since Primary Key is distinctive within the table its value should not repeat and it cannot be 0, empty or null. For deleting, inserting, Updating and Restoring data in the table there must be the presence of primary key. (IBM , 2023)

A field in one table that reference primary key and in another table is called a foreign key. To prevent from demolishing links between tables foreign key are used. (w3schools, 2023)

For Customer

Here, Customer id is identified as Primary Key, also be called as unique identifier which means value will not repeat also the value is not be empty or 0.

Attribute	Datatype	Constraint
Customer id	Number	Primary Key, Not Null

Table 14 Primary key in customer

For Order

Attribute	Datatype	Constraint
Order id	Number	Primary Key, Not Null
Customer id	Number	Foreign Key (Customer id) Reference Customer (Customer id)
Product id	Number	Foreign Key (Product id) Reference Product (Product id)

Table 15 Primary Key and Foreign Key in order

For Product

Attribute	Datatype	Constraint
Product id	Number	Primary Key, Not Null
Order id	Number	Foreign Key (Order id) Reference Customer (Order id)
Vendor id	Number	Foreign Key (Vendor id) Reference Vendor (Vendor id)

Table 16 Primary Key and Foreign Key in Product

For Vendor

Attribute	Datatype	Constraint
Vendor id	Number	Primary Key, Not Null

Table 17 Primary Key in Vendor

4.3 Initial ERD

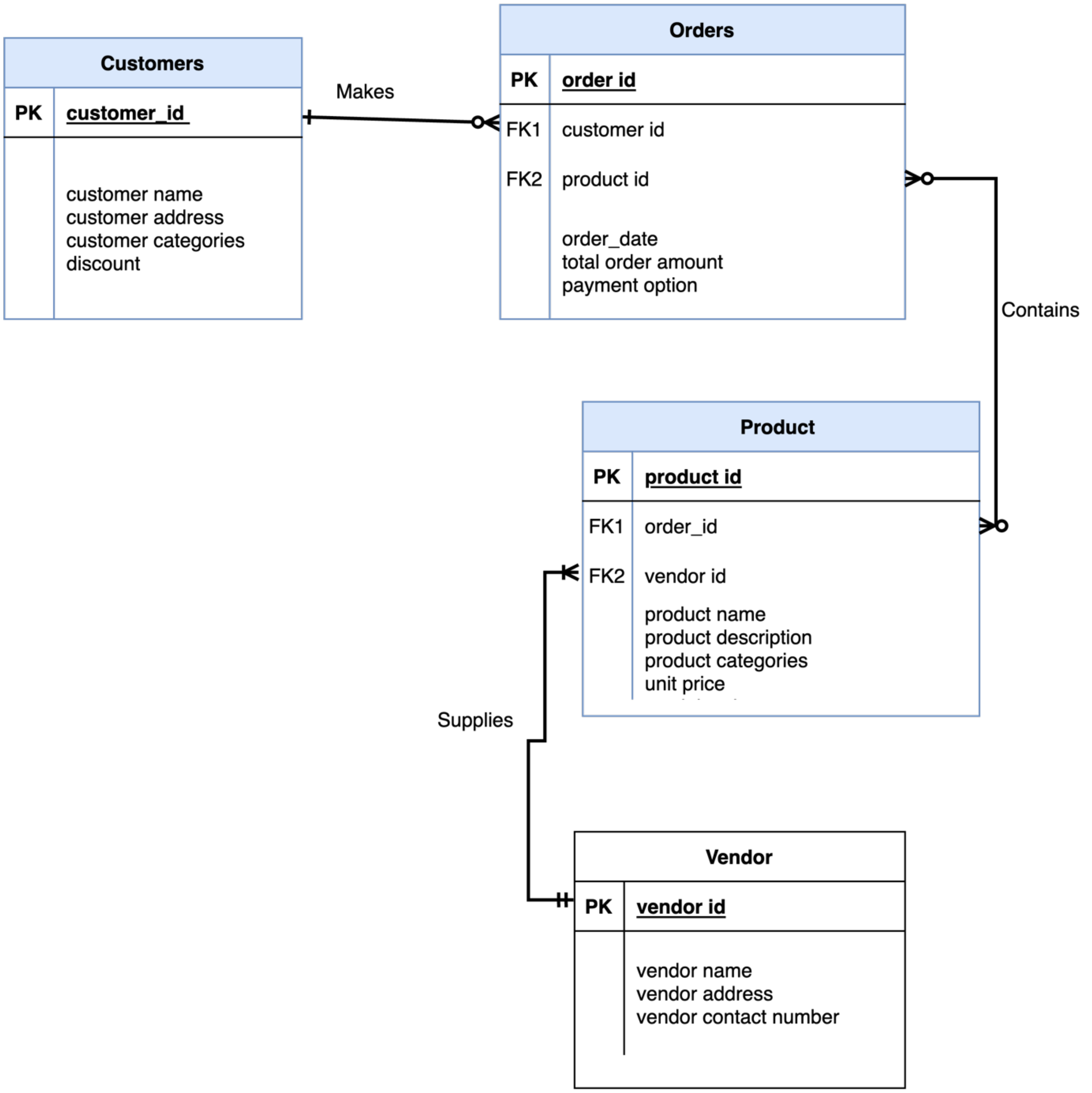


Figure 1 Initial ERD

Customer and Order = One Mandatory to Many Optional

One Customer may purchase many products, but each product must be associated with one specific customer.

Order and Product = Many to Many Optional

One order may have many products and one specific product may be included in one order.

Product and Vendor = One to Many Mandatory

Each product must be supplied by one specific vendor and one specific vendor must supply many products to the company.

5. Normalization

To terminate data redundancy and unpleasant characteristics like Insertion, Anomalies, Update and Deletion normalization is used. To logically store the data without any repetition is normalization, in this coursework we are using Normal forms as UNF, 1NF, 2NF and 3NF. (Peterson, 2023)

- **Customer**- Customer id, Customer Categories, Customer name, Customer Address, Discount
- **Order**- Order id, Order date, Total Amount Order, Payment Option
- **Product**- Product id, Product name, Product Description, Product Categories, Unit Price, Stock Quantity, Order Quantity, Line Total
- **Vendor**- Vendor Id, Vendor Name, Vendor Address, Vendor Contact Number

5.1 UNF (Un-normalized form)

In UNF all the attributes with repeating groups are included together in a single Relation.

STEP 1: **List all the attributes from and name the relation(entity).**

Business (customer id, customer name, customer address, customer categories, discount, order id, order date, total order amount, payment option, product id, product name, product description, product categories, stock level, unit price, line total, order quantity, vendor id, vendor name, vendor address, vendor contact number)

Explanation

The Name of the relation is Business and, in the bracket, there is a list of all the attributes which are listed down together.

STEP 2: Choose a suitable unique identifier for the relation(entity).

Business (<u>customer id</u>, customer name, customer address, customer categories, discount, order id, order date, total order amount, payment option, product id, product name, product description, product categories, stock level, unit price, line total, order quantity, vendor id, vendor name, vendor address, vendor contact number)

Explanation

Customer id is chosen to be the unique identifier and is being underlined.

STEP 3: Show Repeating Group within { }.

Business (<u>customer id</u>, customer name, customer address, customer categories, discount, {order id, order date, total order amount, payment option, {product id, product name, product description, product categories, stock level, unit price, line total, order quantity, vendor id, vendor name, vendor address, vendor contact number}})

Explanation

The repeating group are placed with the curly bracket. Here there exist a repeating group inside a repeating group.

Customer {order, {product, vendor}}

One customer can order multiple order and each order can have multiple products. There is no relation of vendor with customer and order but there is a relation of vendor with product as products are being received by vendor. Therefore, vendor and product are placed together inside same curly bracket.

5.2 1NF (First Normal Form)

In 1NF all the repeating group we identified in UNF is removed or separated as a new relation (entity). Advantage of 1NF is simplicity and uniform access.

STEP 1: Remove Repeating Group to form new relation (Entity), Name it.

CustomerOrder (order id, order date, total order amount, payment option)

ProductVendor (product name, product description, product categories, stock level, unit price, line total, order quantity, vendor id, vendor name, vendor address, vendor contact number)

Explanation

In this step we removed the repeating group which were shown in the curly bracket and separated them into two relation (Entity) and gave them a name.

STEP 2: Carry Forward the Unique Identifier to this Relation (Entity)

CustomerOrder (customer id *, order id, order date, total order amount, payment option)

ProductVendor (customer id *, order id *, product name, product description, product categories, stock level, unit price, line total, order quantity, vendor id, vendor name, vendor address, vendor contact number)

Explanation

In this step we carry forwarded the unique identifier from the previous relation. We must carry forward every unique identifier from the previous relation as soon as there is a creation of new relation. The carry forwarded attributes are shown with underline and asterisk (*).

STEP 3: Choose the unique identifier for the new relation.

CustomerOrder (customer id *, order id, order date, total order amount, payment option)

ProductVendor (customer id *, order id *, product id, product name, product description, product categories, stock level, unit price, line total, order quantity, vendor id, vendor name, vendor address, vendor contact number)

Explanation

After creation of the new relation, we need to choose a unique identifier. Here I have chosen order id for customerorder relation (entity) and product id for productvendor relation (entity).

Tables Created After 1NF.

1. **Customer -1** (customer id, customer name, customer address, customer categories, discount)
2. **CustomerOrder -1** (customer id *, order id, order date, total order amount, payment option)
3. **ProductVendor -1** (customer id *, order id *, product id, product name, product description, product categories, stock level, unit price, line total, order quantity, vendor id, vendor name, vendor address, vendor contact number)

Explanation

From the UNF, moving forward to the 1NF removing repeating groups, carry forwarding the unique identifier and choosing new identifier in the new relation there are 3 tables being created. Unique Identifier are being underlined in each relation and carry forward identifiers are being underlined with presence of asterisk.

5.3 2NF (Second Normal Form)

In 2NF we remove the Partial Functional Dependency and Full Functional Dependency. We use Full Functional Dependency if it is necessary to use all attributes of Composite Determinant to identify its object uniquely. Partial Functional Dependency Exist when if it is necessary to use only subset of Attributes of a composite determinant to identify object uniquely.

- **Checking Partial Functional Dependency and Full Functional Dependency**

  From CustomerOrder Relation

- Customer id, order id → ❌

  (There is nothing dependent both on customer id and order id)

- Customer id → ❌

  (There is no partial dependency here.)

- Order id → order date, total order amount, payment option

  Here order date, total order amount and payment option are partial dependent on order id as order id give order date, the total order amount is calculated based on order id and payment option, or method also depends upon the order id.

  Therefore, Order id a Composite Key and order date, total order amount and payment option depends on Order id.

  Now we must separate the partial dependency into new relation and name it.

  CustomerOrder (customer id*, order id*)

  Order (order id, order date, total order amount, payment option)

  Explanation

  The partial dependency is separated into a new relation and named as Order. In the new relation Order the order id will be the primary key and the in the CustomerOrder relation the order id will be the foreign key.

  For ProductVendor Relation

- Customer id, Order id, Product id → ❌

There is nothing dependent on Customer id, order id and product id

- Customer id, Product id → ❌

(There is nothing dependent on Customer id and Product id)

- Order id, Product id → order quantity, line total

(Assumption: The order quantity is dependent both on order and product as the order id gives the quantity number and product id gives which product and for the line total order quantity is required from the order id and product unit price is required from the product id.)

- Customer id → ❌

(There is no partial dependency here.)

- Order id → ❌

(There is no partial dependency here.)

Product id → product name, product description, product categories, stock level, unit price, vendor id, vendor name, vendor address, vendor contact number

(Assumption: the product name, product description, stock, unit price all is dependent on product id as product id can give product name, product id can give the description of the product, categories of product, stock level can be identified by the product id as well as the unit price. The vendor's name, address, contact number is also dependent on product id as product is supplied by vendor.)

Again we need to separate the FFD and PFD into a new relation and name them accordingly.

OrderItemLine (order id *, product id *, order quantity, line total)

CustomerOrderProduct – (customer id *, order id *, product id *)

ProductVendor (product id, product name, product description, product categories, stock level, unit price, vendor Id, vendor name, vendor address)

Explanation

Order quantity and Line total from product relation is '**Fully Functional Dependent**' on order id and product id from the above assumption.

Product name, description, categories, stock level, unit price, vendor name, address, contact number is '**Partially Functional Dependent**' on product id from the above assumption.

Initial Second Normal Form

CustomerOrder (customer id*, order id*)

Order (order id, order date, total order amount, payment option)

OrderItemLine (order id *, product id *, order quantity, line total)

CustomerOrderProduct (customer id *, order id *, product id *)

ProductVendor (product id, product name, product description, product categories, stock level, unit price, vendor Id, vendor name, vendor address, vendor contact number)

Final Second Normal Form Table

1. **Customer - 2** (customer id, customer name, customer address, customer categories, discount)
2. **CustomerOrder –2** (customer id*, order id*)
3. **Order -2** (order id, order date, total order amount, payment option)
4. **OrderItemLine -2** (order id *, product id *, order quantity, line total)
5. **CustomerOrderProduct -2** (customer id *, order id *, product id *)
6. **ProductVendor -2** (product id, product name, product description, product categories, stock level, unit price, vendor Id, vendor name, vendor address, vendor contact number)

Explanation

From 1NF, the partial functional dependency and full functional dependencies are identified and removed into a new relation. The unique identifiers are underlined, carry forwarded and left over identifiers are under lined with asterisk. There are total 6 table created after 2NF.

5.4 3NF (Third Normal Form)

In 3NF the Transitive Functional Dependency are removed. Existence of intermediate dependency occurs Transitive Functional Dependency. Attributes that are wholly dependent upon another attribute should be removed and separated to a new relation.

We need to check for Transitive Functional Dependency in all Relation which have more than one non-key attribute.

Note (Relation with only one non-key or no non-key is already in 3NF)

Transitive Dependency between Non-keys

In Customer Relation

customer categories → discount

customer id → customer categories → discount

(Assumption: Here customer categories gives discount rate as the discount rate may be differ based on categories as Regular Customer gets some rate, VIP can get other rate and Regular Customer can get different rate)

Now we need to separate it and name it also choose a unique identifier which is underlined. The customer categories in the customer relation is left over becoming foreign key.

Customer (customer id, customer name, customer address, customer categories *)

CustomerDisount (customer categories, discount)

In ProductVendor relation

product id → vendor id

product id → vendor id → vendor name, vendor address, vendor contact number

(Assumption: product Id gives vendor id as from which vendor the product is being supplied and the vendor id gives the details of the vendor like vendor name, address, contact number)

Now we need to separate it and name it also choose a unique identifier which is underlined. The vendor id in the product relation is left over becoming foreign key.

Product (product id, product name, product description, product categories, vendor id * , stock level, unit price)

Vendor (vendor id, vendor name, vendor address, vendor contact number)

No Transitive Dependency between Non-Keys

In Customer Relation

customer name → ✗ customer address

Here customer name does not give customer address as there can be a lot of customers having the same name but the two people having same name lives in different location. For example there are two customer having name Ram but their living place can be different like first Ram can live in Kathmandu and second Ram can live in Bhaktapur.

In order relation

order date → ✗ total order amount

Here order date doesn't give total order amount because in a day multiple order can be processed and each order can have different total order amount.

order date → ✗ payment option

Here order date doesn't give payment option as in a day multiple customers can order multiple products and do multiple payment with different payment option. For example on August 5 2023 many customer can make payment with different methods like cash on delivery, credit card, e-wallet so order 1 can have cash on delivery method, order 2 can have credit card method on the same day.

In ProductVendor relation

product name → ❌ product description

Here product name doesn't give product description as many products can have same name, but different description example name of the product is Laptop the description can be different such as Mac Book, Windows, Dell etc.

product name → ❌ product categories

Here product name doesn't give product category as two different product name can have same category for example Samsung Galaxy S23 Ultra and iPhone 13 pro max are the two different name of the product having same product categories which is Smart Phone.

product name → ❌ stock level

Here the product name doesn't give the stock level as name of the product can be same and repeat but the stock level of the product can be different. For example, the name of the product is Phone for two products but the description is different like iPhone and Android so the stock level can vary.

product name → ❌ unit price

Here the product name doesn't give unit price as name of the product can be same and repeat but the price of the product can be different. For example, the name of the product is TV for two products but the description is different like Samsung 55 inch and Sony 42 inch so the price is different.

vendor name → ❌ vendor address

Here vendor name does not give vendor address as there can be a lot of vendors having the same name but the two vendors having same name can be located in different part. For example, there are two vendors having name Apple but their location can be different like first Apple vendor can be located in Kathmandu and second Apple vendor can be located in Pokhara.

vendor name → ❌ vendor contact number

Here vendor name does not give vendor contact number as there can be a lot of vendors having the same name but the two vendors can have different contact number. For example, there are two vendors having name Apple but their contact number be different as they are located in different location.

CustomerOrder –2 (customer id*, order id*)

In this relation there aren't any non-key. Therefore, the relation is already in 3NF. We don't have to make any changes.

CustomerOrderProduct -2 (customer id *, order id *, product id *)

Same goes to this relation there aren't any non-key. Therefore, the relation is already in 3NF. We don't have to make any changes.

OrderItemLine -2 (order id *, product id *, order quantity, line total)

order quantity → ❌ line total

Here order quantity doesn't give line total as line total is calculated by multiplying order quantity and unit price of the product.

Initial Third Normal Form

Customer (customer id, customer name, customer address, customer categories *)

CustomerDisount (customer categories, discount)

Product (product id, product name, product description, product categories, vendor id *, stock level, unit price)

Vendor (vendor id, vendor name, vendor address, vendor contact number)

Final Third Normal Form

1. **Customer - 3** (customer id, customer name, customer address, customer categories *)
2. **CustomerDisount -3** (customer categories, discount)
3. **CustomerOrder –2** (customer id*, order id*)
4. **Order -3** (order id, order date, total order amount, payment option)
5. **OrderItemLine -3** (order id *, product id *, order quantity, line total)
6. **CustomerOrderProduct -3** (customer id *, order id *, product id *)
7. **Product-3** (product id, product name, product description, product categories, vendor id *, stock level, unit price)
8. **Vendor -3** (vendor id, vendor name, vendor address, vendor contact number)

Explanation

After removing the transitive dependency and separating them into a new relation choosing a new unique identifier in the new relation made, the 3NF is complete. Now we have altogether 8 tables.

6. Final ERD

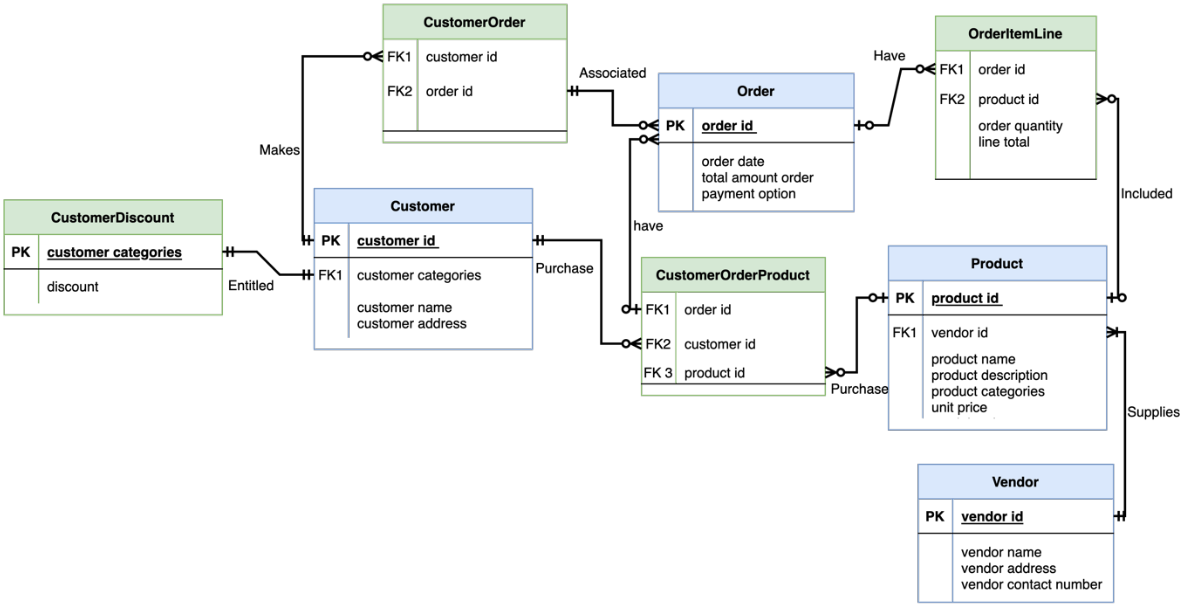


Figure 2 Final ERD

Here the entities are coloured in blue and the bridging entities and entities made after the normalization process are coloured in Green.

CustomerDiscount and Customer = One to One Mandatory

A customer must be entitled a discount rate based on their categories and One category must have one customer.

Customer and CustomerOrder = One Mandatory to Many Optional

One customer may have many orders but each order must be associated to one specific customer.

Customer and CustomerOrderProduct = One Mandatory to Many Optional

One Customer may purchase many products and may have many products but each order and purchased product is associated with one specific customer.

Order and Customer Order = Many Optional To One Mandatory

Many orders must be associated with one customer and one customer may have many orders.

Order and OrderItemLine = One to Many Optional

One Order may have multiple products and Each Orders may be included in one order.

Order and CustomerOrderProduct = Many To One Optional

Many Order can be placed by a customer and many order may include one specific product and one specific product may be included in many order placed by customer and one customer may order may have many order.

Product and OrderItemline = One to Many Optional

One product may be included in many orders and many orders may have that specific product,

Product and CustomerOrderProduct = One Mandatory to Many Optional

One Specific Product may be purchased by many customers and may be included into many orders and many customers and order may purchase or include that specific product.

Product and Vendor = One to Many Mandatory

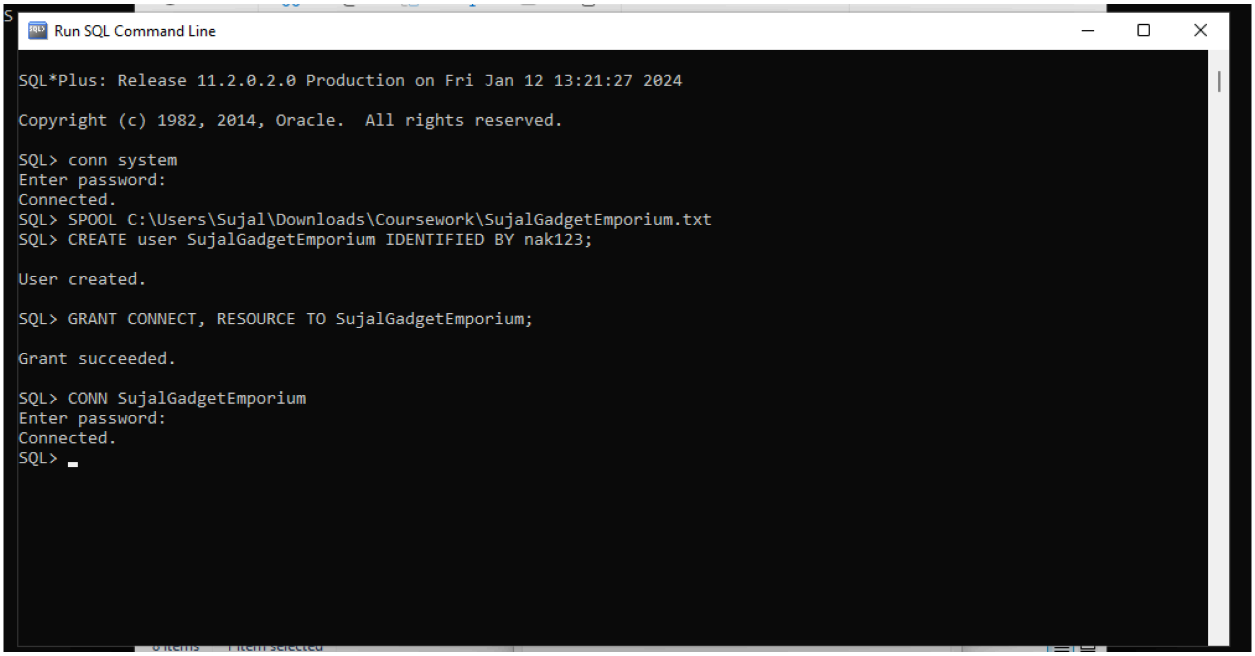
Each product must be supplied by one specific vendor and one specific vendor must supply many products to the company.

7. Implementation

- **Relations and tables for the "Gadget Emporium” database with the SQL Command and list the snapshot of its resulting output. Ensure that referential integrity is established between related tables.**

First we need to connect system and give password then we need to create a new user here I have given name SujalGadgetEmporium also we need to create a password here I have given nak123. After that we need to conn to that user from the system and Start Creating, Inserting and giving queries.

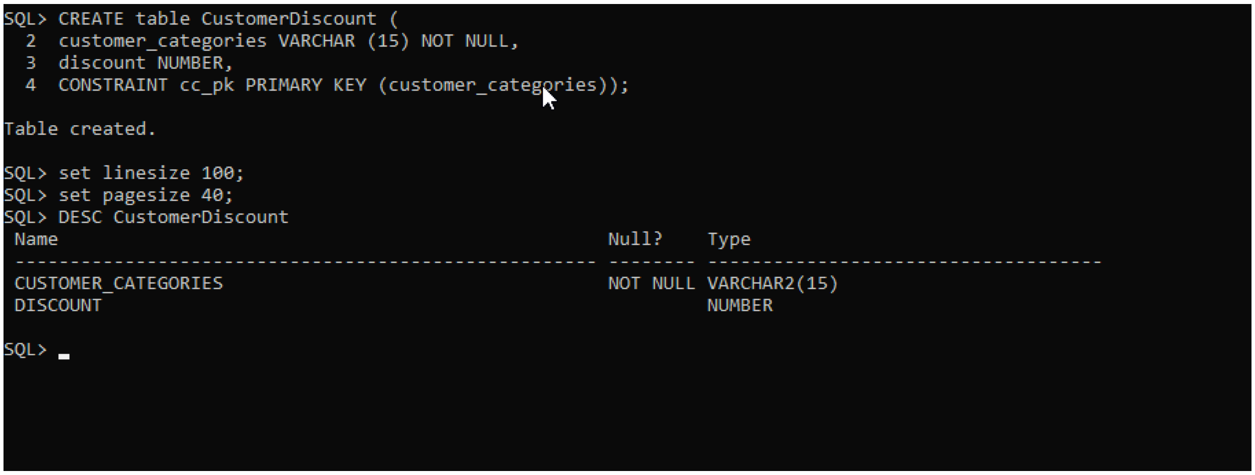
The following are the screenshot of the values being inserted in each table along with their Constraint Like Foreign Key and Primary Key.



```
SQL*Plus: Release 11.2.0.2.0 Production on Fri Jan 12 13:21:27 2024

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> conn system
Enter password:
Connected.
SQL> SPOOL C:\Users\Sujal\Downloads\Coursework\SujalGadgetEmporium.txt
SQL> CREATE user SujalGadgetEmporium IDENTIFIED BY nak123;

User created.

SQL> GRANT CONNECT, RESOURCE TO SujalGadgetEmporium;

Grant succeeded.

SQL> CONN SujalGadgetEmporium
Enter password:
Connected.
SQL>
```

Figure 3 Connecting To System and Creating New User and Granting Resource

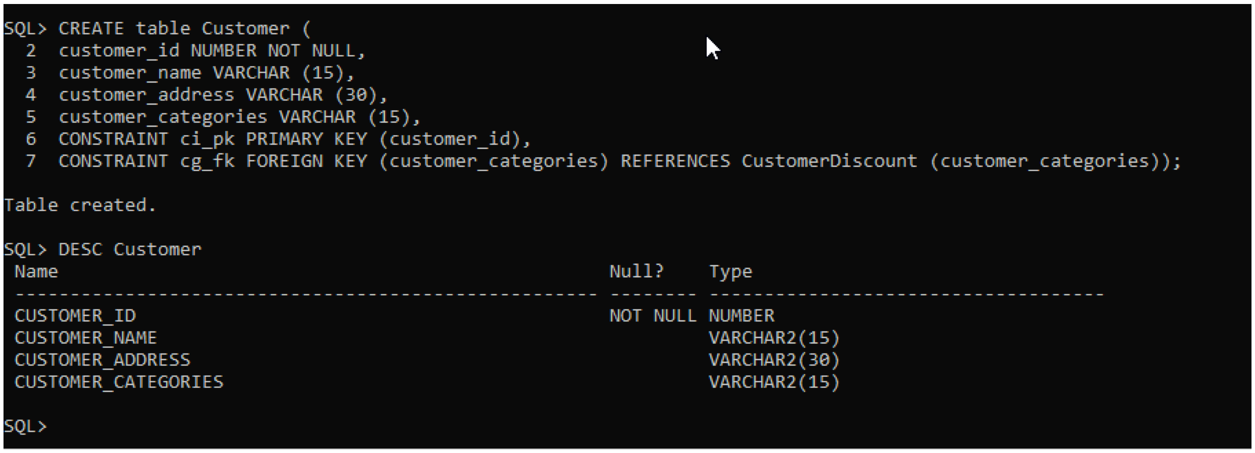
Creating CustomerDiscount table



```
SQL> CREATE table CustomerDiscount (
  2  customer_categories VARCHAR (15) NOT NULL,
  3  discount NUMBER,
  4  CONSTRAINT cc_pk PRIMARY KEY (customer_categories));

Table created.

SQL> set linesize 100;
SQL> set pagesize 40;
SQL> DESC CustomerDiscount
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 CUSTOMER_CATEGORIES                                   NOT NULL VARCHAR2(15)
 DISCOUNT                                                       NUMBER

SQL> _
```

Figure 4 Creating Table CustomerDiscount

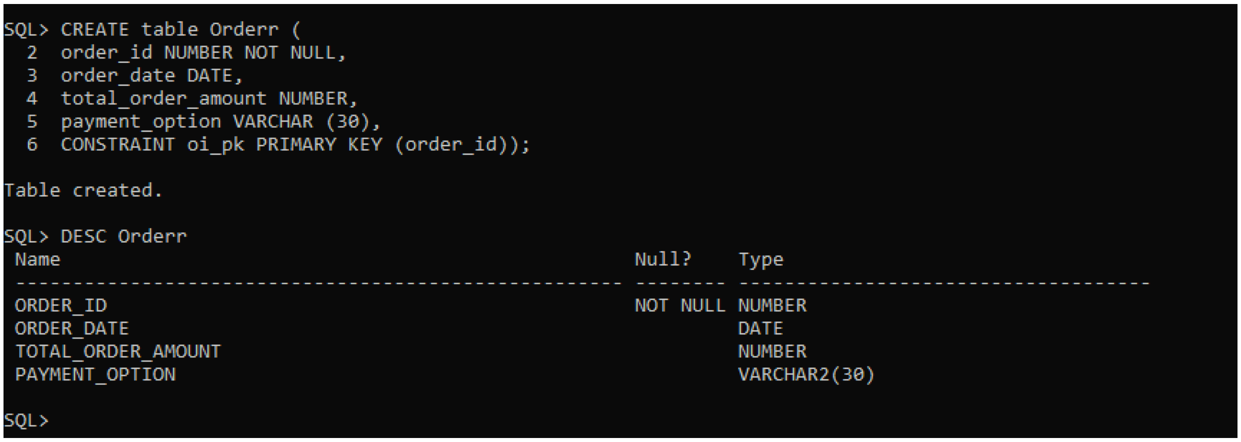
Creating Customer table



```
SQL> CREATE table Customer (
  2  customer_id NUMBER NOT NULL,
  3  customer_name VARCHAR (15),
  4  customer_address VARCHAR (30),
  5  customer_categories VARCHAR (15),
  6  CONSTRAINT ci_pk PRIMARY KEY (customer_id),
  7  CONSTRAINT cg_fk FOREIGN KEY (customer_categories) REFERENCES CustomerDiscount (customer_categories));

Table created.

SQL> DESC Customer
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 CUSTOMER_ID                                           NOT NULL NUMBER
 CUSTOMER_NAME                                                  VARCHAR2(15)
 CUSTOMER_ADDRESS                                               VARCHAR2(30)
 CUSTOMER_CATEGORIES                                            VARCHAR2(15)

SQL>
```

Figure 5 Creating Table Customer

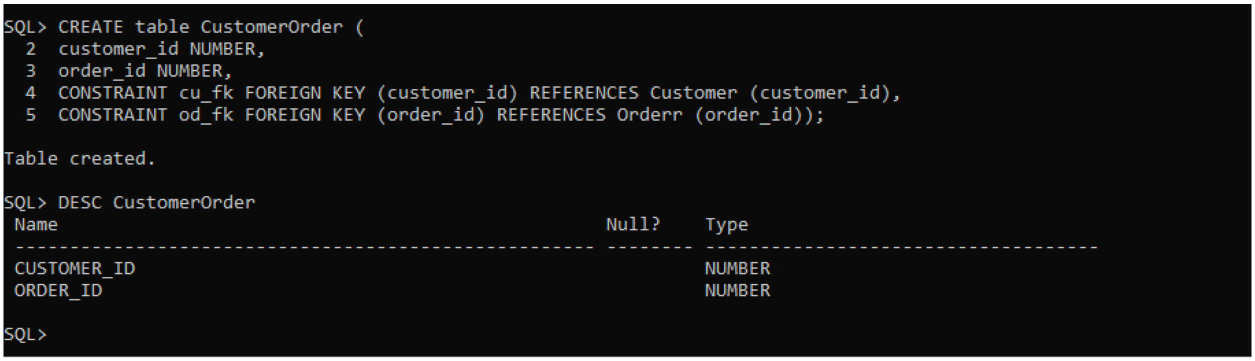
Creating Order table Since order is a keyword I have given additional r . Orderr



```
SQL> CREATE table Orderr (
     order_id NUMBER NOT NULL,
     order_date DATE,
     total_order_amount NUMBER,
     payment_option VARCHAR (30),
     CONSTRAINT oi_pk PRIMARY KEY (order_id));

Table created.

SQL> DESC Orderr
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ORDER_ID                                              NOT NULL NUMBER
 ORDER_DATE                                                     DATE
 TOTAL_ORDER_AMOUNT                                             NUMBER
 PAYMENT_OPTION                                                 VARCHAR2(30)

SQL>
```

Figure 6 Creating Table Orderr

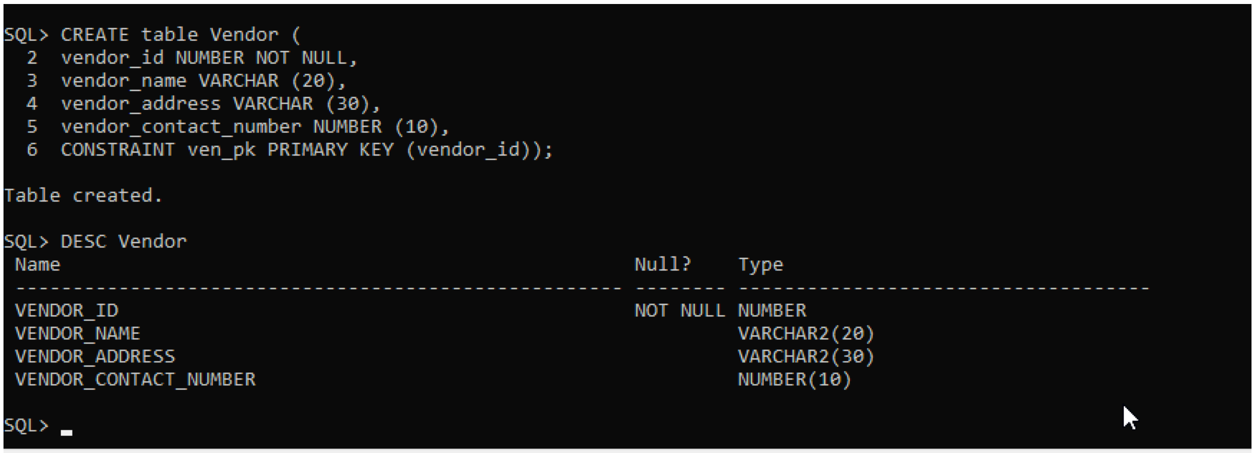
Creating CustomerOrder table



```
SQL> CREATE table CustomerOrder (
     customer_id NUMBER,
     order_id NUMBER,
     CONSTRAINT cu_fk FOREIGN KEY (customer_id) REFERENCES Customer (customer_id),
     CONSTRAINT od_fk FOREIGN KEY (order_id) REFERENCES Orderr (order_id));

Table created.

SQL> DESC CustomerOrder
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 CUSTOMER_ID                                                    NUMBER
 ORDER_ID                                                       NUMBER

SQL>
```

Figure 7 Creating Table CustomerOrder

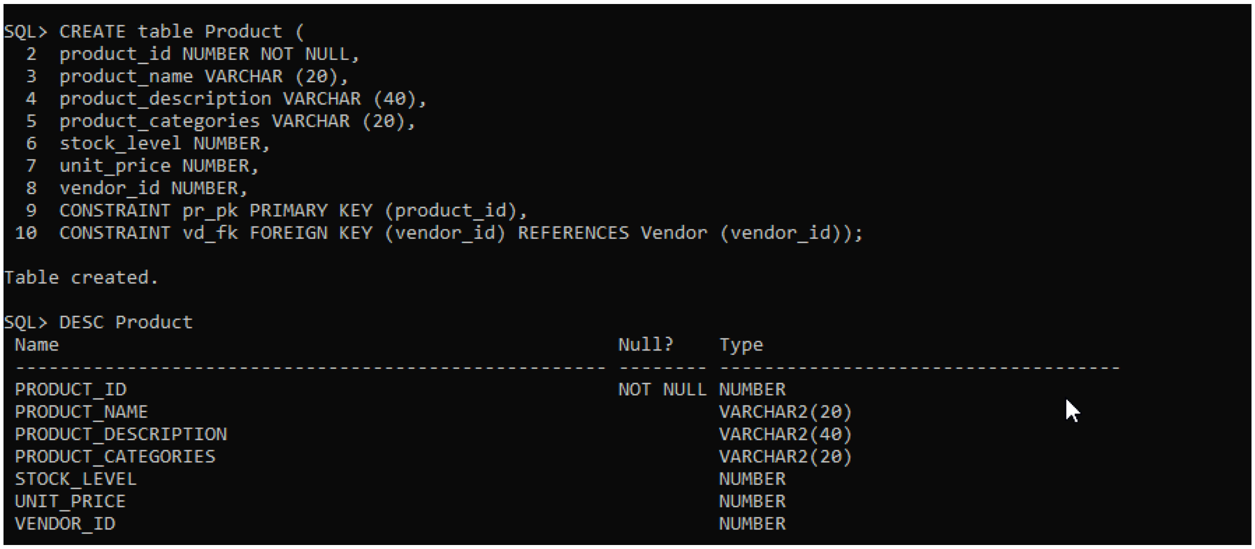
Creating Vendor table



```
SQL> CREATE table Vendor (
     vendor_id NUMBER NOT NULL,
     vendor_name VARCHAR (20),
     vendor_address VARCHAR (30),
     vendor_contact_number NUMBER (10),
     CONSTRAINT ven_pk PRIMARY KEY (vendor_id));

Table created.

SQL> DESC Vendor
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 VENDOR_ID                                             NOT NULL NUMBER
 VENDOR_NAME                                                    VARCHAR2(20)
 VENDOR_ADDRESS                                                 VARCHAR2(30)
 VENDOR_CONTACT_NUMBER                                          NUMBER(10)

SQL>
```

Figure 8 Creating Table Vendor

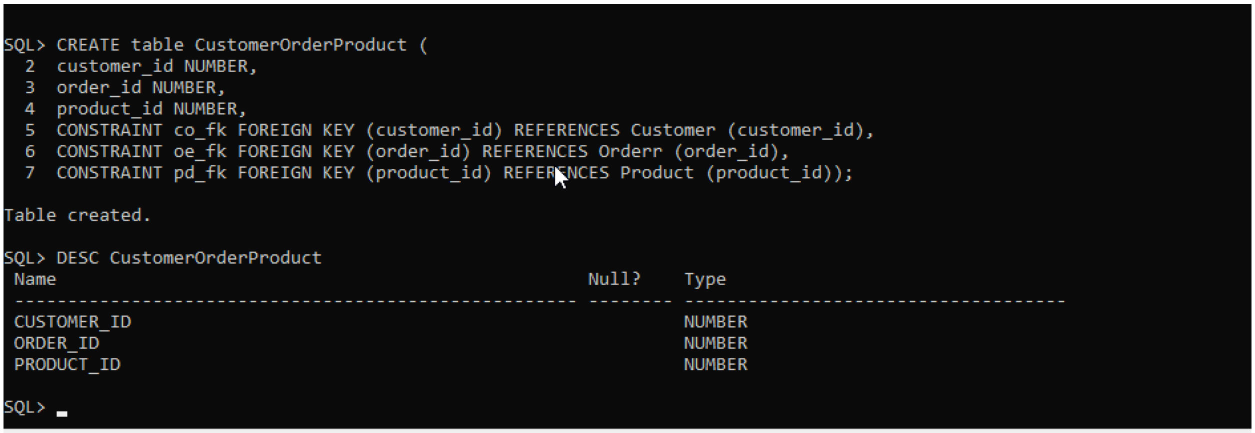
Creating Vendor Table



```
SQL> CREATE table Product (
  2  product_id NUMBER NOT NULL,
  3  product_name VARCHAR (20),
  4  product_description VARCHAR (40),
  5  product_categories VARCHAR (20),
  6  stock_level NUMBER,
  7  unit_price NUMBER,
  8  vendor_id NUMBER,
  9  CONSTRAINT pr_pk PRIMARY KEY (product_id),
 10  CONSTRAINT vd_fk FOREIGN KEY (vendor_id) REFERENCES Vendor (vendor_id));

Table created.

SQL> DESC Product
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 PRODUCT_ID                                            NOT NULL NUMBER
 PRODUCT_NAME                                                   VARCHAR2(20)
 PRODUCT_DESCRIPTION                                            VARCHAR2(40)
 PRODUCT_CATEGORIES                                             VARCHAR2(20)
 STOCK_LEVEL                                                    NUMBER
 UNIT_PRICE                                                     NUMBER
 VENDOR_ID                                                      NUMBER
```

Figure 9 Creating Table Vendor

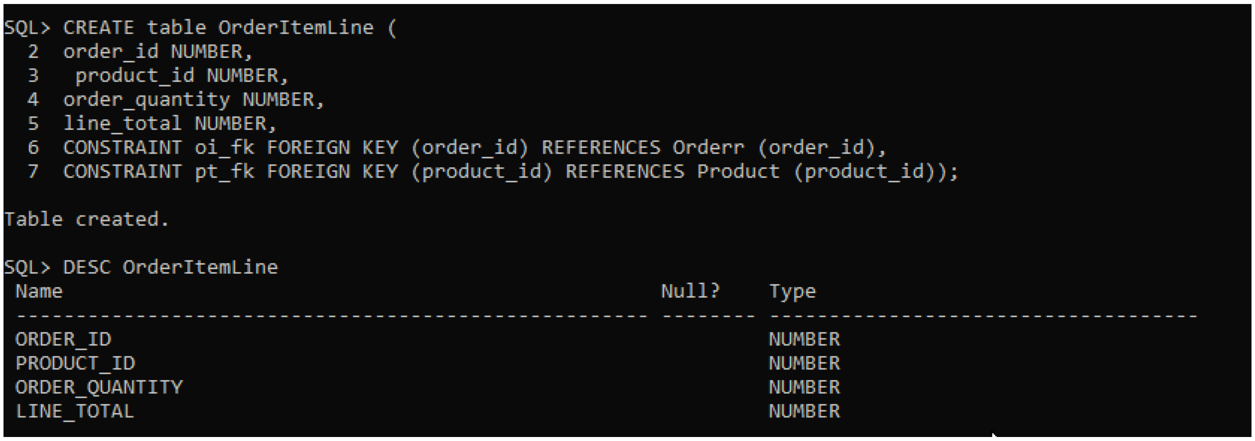
Creating CustomerOrderProduct table



```
SQL> CREATE table CustomerOrderProduct (
  2  customer_id NUMBER,
  3  order_id NUMBER,
  4  product_id NUMBER,
  5  CONSTRAINT co_fk FOREIGN KEY (customer_id) REFERENCES Customer (customer_id),
  6  CONSTRAINT oe_fk FOREIGN KEY (order_id) REFERENCES Orderr (order_id),
  7  CONSTRAINT pd_fk FOREIGN KEY (product_id) REFERENCES Product (product_id));

Table created.

SQL> DESC CustomerOrderProduct
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 CUSTOMER_ID                                                    NUMBER
 ORDER_ID                                                       NUMBER
 PRODUCT_ID                                                     NUMBER

SQL> _
```

Figure 10 Creating table CustomerOrderProduct

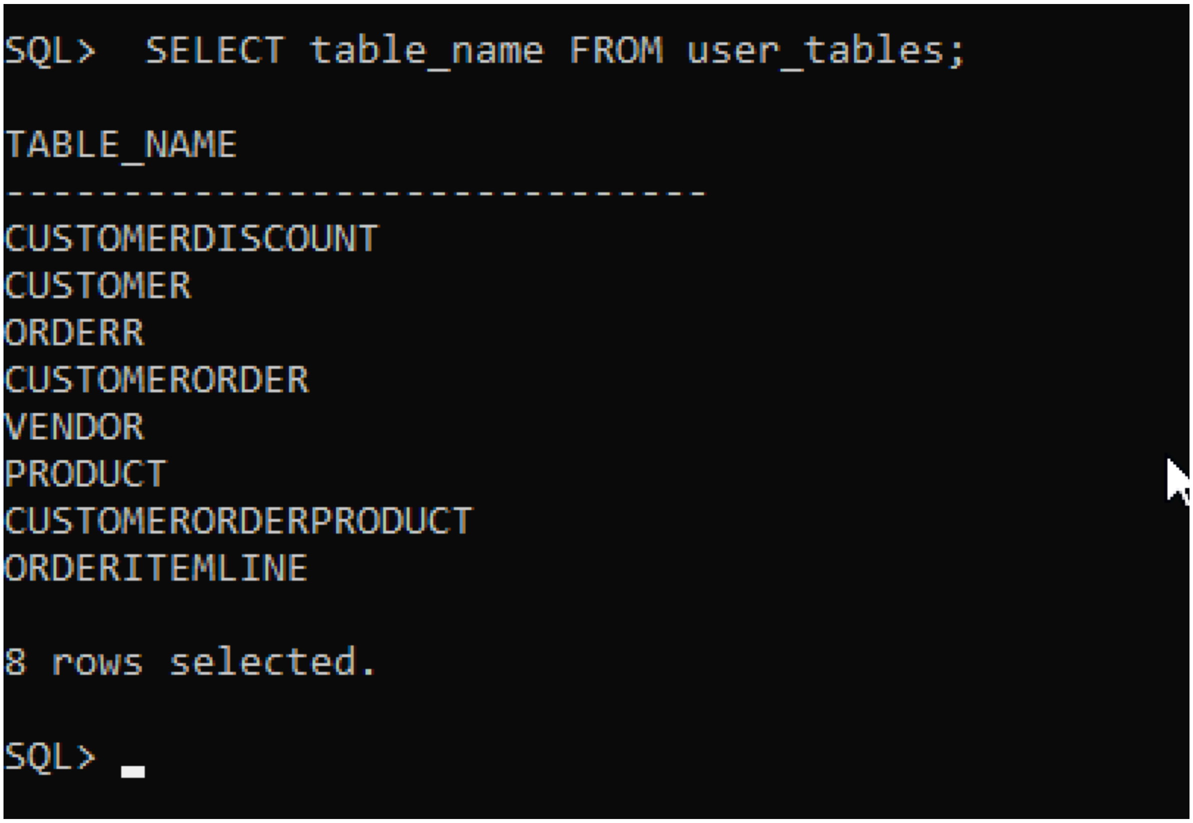
Creating OrderItemLine table



```
SQL> CREATE table OrderItemLine (
order_id NUMBER,
 product_id NUMBER,
order_quantity NUMBER,
line_total NUMBER,
CONSTRAINT oi_fk FOREIGN KEY (order_id) REFERENCES Orderr (order_id),
CONSTRAINT pt_fk FOREIGN KEY (product_id) REFERENCES Product (product_id));

Table created.

SQL> DESC OrderItemLine
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 ORDER_ID                                                       NUMBER
 PRODUCT_ID                                                     NUMBER
 ORDER_QUANTITY                                                 NUMBER
 LINE_TOTAL                                                     NUMBER
```

Figure 11 Creating Table OrderItemLine

This is a query to list all the table we created.



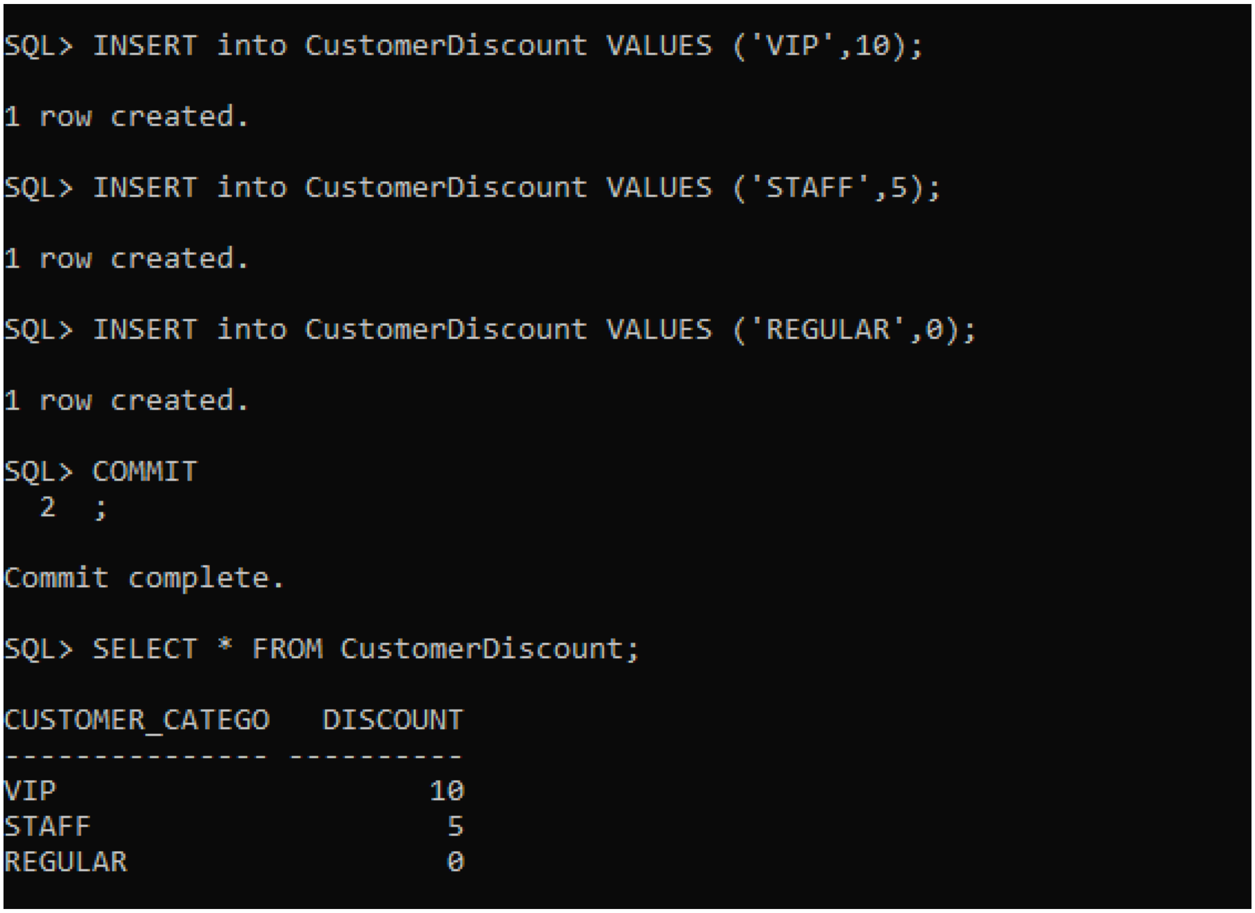
```
SQL>  SELECT table_name FROM user_tables;

TABLE_NAME
------------------------------
CUSTOMERDISCOUNT
CUSTOMER
ORDERR
CUSTOMERORDER
VENDOR
PRODUCT
CUSTOMERORDERPRODUCT
ORDERITEMLINE

8 rows selected.

SQL> _
```

Figure 12 Viewing all the table made in the user

- **Populate them with appropriate test data that is relevant to the questions listed below. List the screenshots of the SQL Command used and the overall rows of the table with an image of its resulting output. Enter at least 7 rows in each table.**

  Include the screenshot of the INSERT SQL Statement used to populate table data, along with the TABLE's CONTENT displayed using SELECT statements.

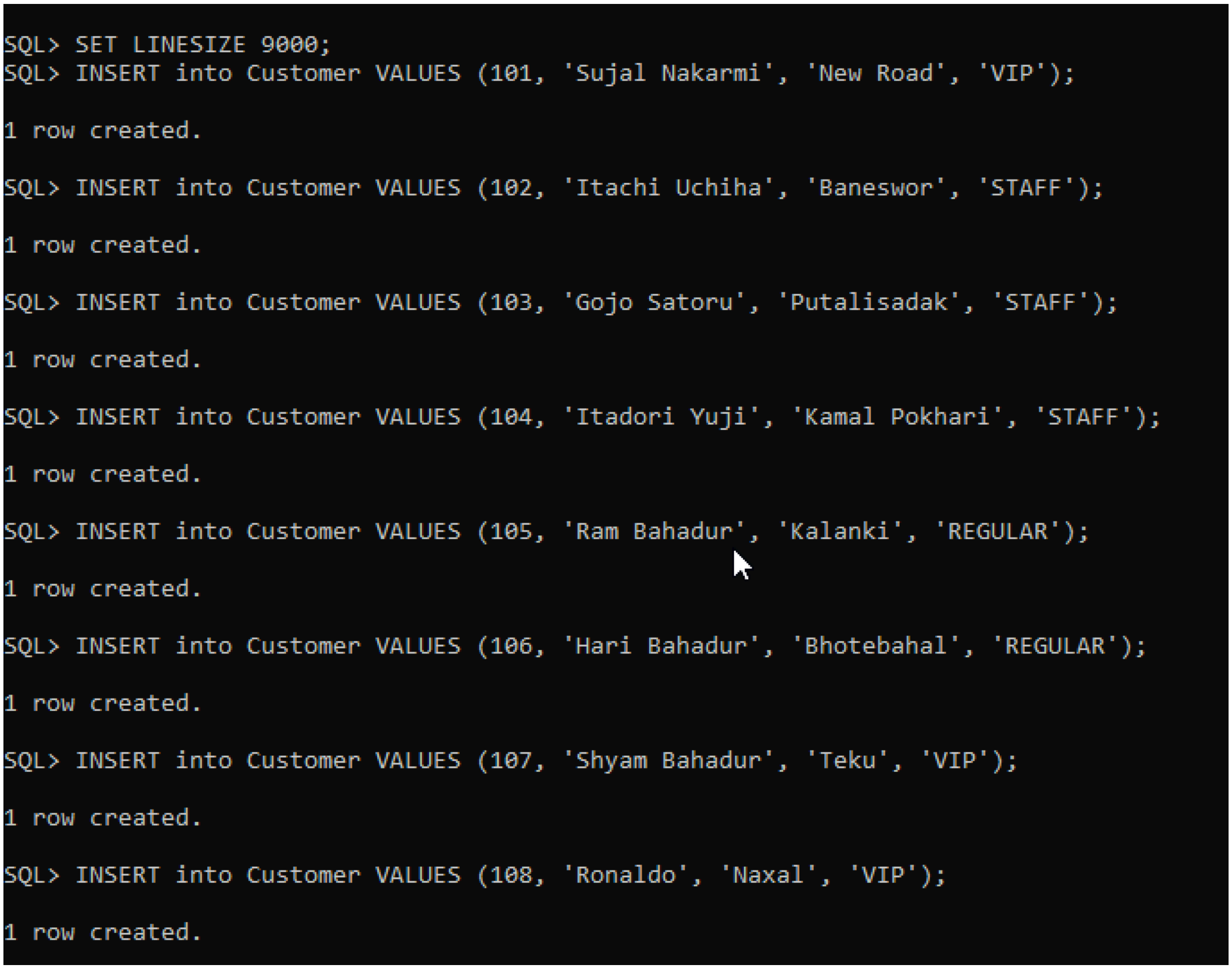
  Inserting Value and Viewing in The CustomerDiscount Table



```
SQL> INSERT into CustomerDiscount VALUES ('VIP',10);

1 row created.

SQL> INSERT into CustomerDiscount VALUES ('STAFF',5);

1 row created.

SQL> INSERT into CustomerDiscount VALUES ('REGULAR',0);

1 row created.

SQL> COMMIT
  2  ;

Commit complete.

SQL> SELECT * FROM CustomerDiscount;

CUSTOMER_CATEGO   DISCOUNT
--------------- ----------
VIP                     10
STAFF                    5
REGULAR                  0
```

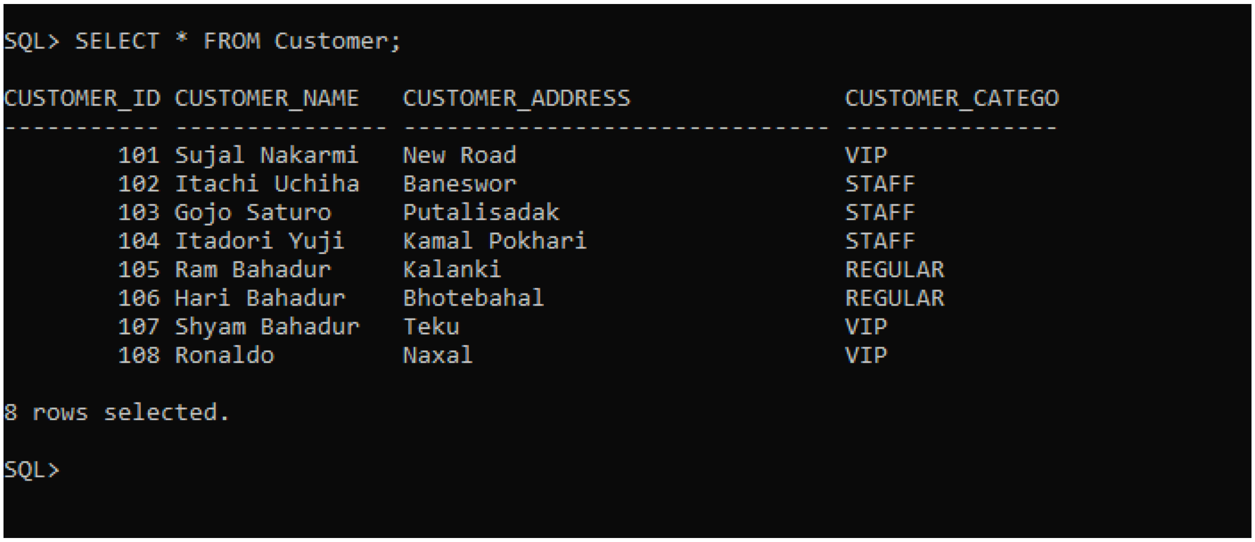
Figure 13 Inserting Value and Viewing in The CustomerDiscount Table

Inserting Value in Customer Table



```
SQL> SET LINESIZE 9000;
SQL> INSERT into Customer VALUES (101, 'Sujal Nakarmi', 'New Road', 'VIP');

1 row created.

SQL> INSERT into Customer VALUES (102, 'Itachi Uchiha', 'Baneswor', 'STAFF');

1 row created.

SQL> INSERT into Customer VALUES (103, 'Gojo Satoru', 'Putalisadak', 'STAFF');

1 row created.

SQL> INSERT into Customer VALUES (104, 'Itadori Yuji', 'Kamal Pokhari', 'STAFF');

1 row created.

SQL> INSERT into Customer VALUES (105, 'Ram Bahadur', 'Kalanki', 'REGULAR');

1 row created.

SQL> INSERT into Customer VALUES (106, 'Hari Bahadur', 'Bhotebahal', 'REGULAR');

1 row created.

SQL> INSERT into Customer VALUES (107, 'Shyam Bahadur', 'Teku', 'VIP');

1 row created.

SQL> INSERT into Customer VALUES (108, 'Ronaldo', 'Naxal', 'VIP');

1 row created.
```

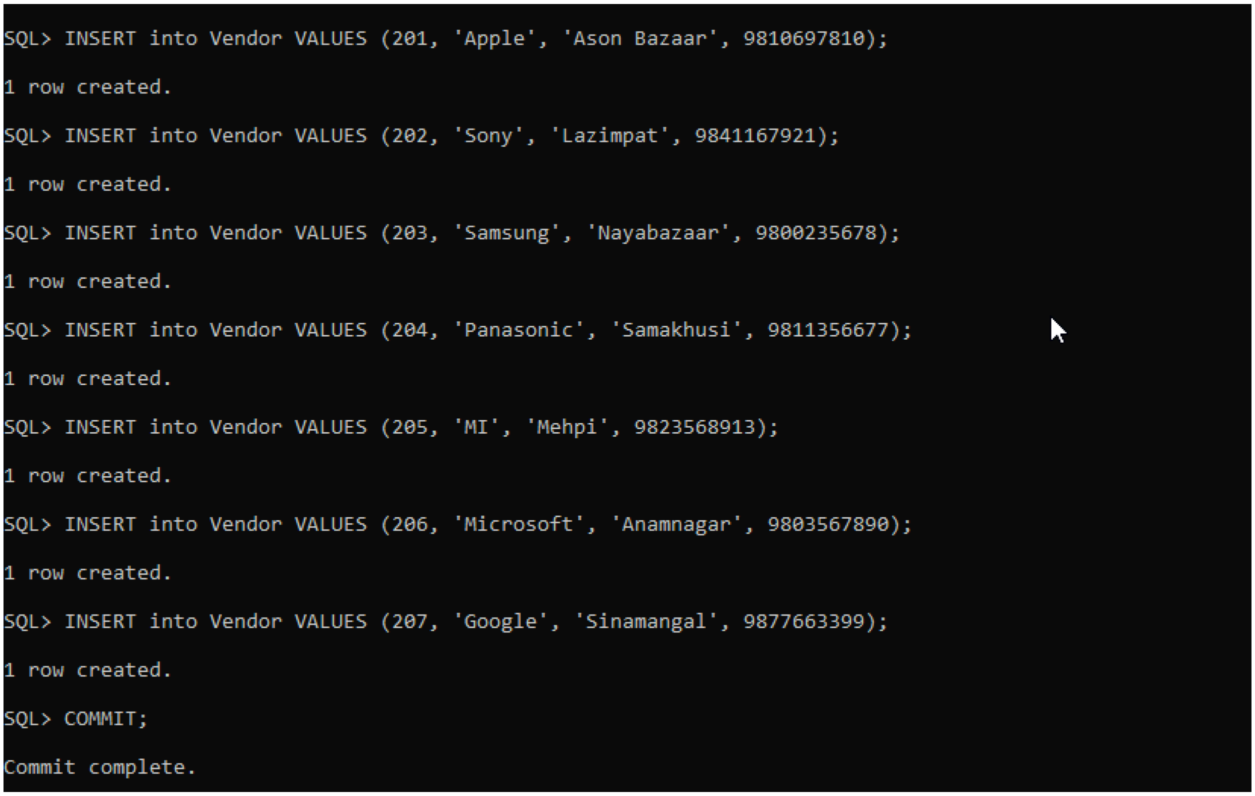
Figure 14 Inserting Value in Customer Table



```
SQL> SELECT * FROM Customer;

CUSTOMER_ID CUSTOMER_NAME   CUSTOMER_ADDRESS              CUSTOMER_CATEGO
----------- --------------- ----------------------------- ---------------
        101 Sujal Nakarmi   New Road                      VIP
        102 Itachi Uchiha   Baneswor                      STAFF
        103 Gojo Saturo     Putalisadak                   STAFF
        104 Itadori Yuji    Kamal Pokhari                 STAFF
        105 Ram Bahadur     Kalanki                       REGULAR
        106 Hari Bahadur    Bhotebahal                    REGULAR
        107 Shyam Bahadur   Teku                          VIP
        108 Ronaldo         Naxal                         VIP

8 rows selected.

SQL>
```

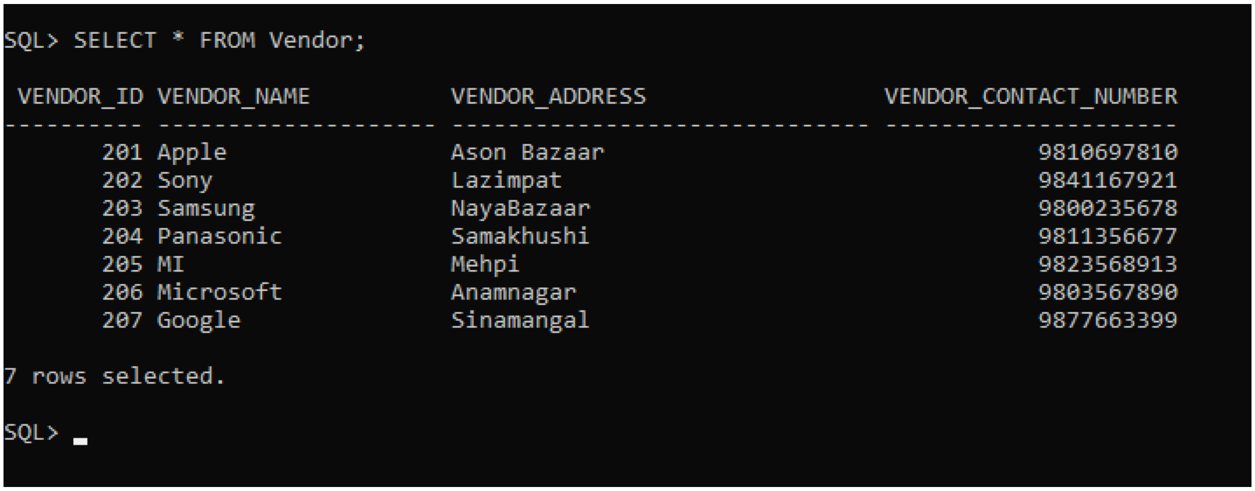
Figure 15 Viewing Inserted Value in Customer

Inserting Value in Vendor Table



```
SQL> INSERT into Vendor VALUES (201, 'Apple', 'Ason Bazaar', 9810697810);

1 row created.

SQL> INSERT into Vendor VALUES (202, 'Sony', 'Lazimpat', 9841167921);

1 row created.

SQL> INSERT into Vendor VALUES (203, 'Samsung', 'Nayabazaar', 9800235678);

1 row created.

SQL> INSERT into Vendor VALUES (204, 'Panasonic', 'Samakhusi', 9811356677);

1 row created.

SQL> INSERT into Vendor VALUES (205, 'MI', 'Mehpi', 9823568913);

1 row created.

SQL> INSERT into Vendor VALUES (206, 'Microsoft', 'Anamnagar', 9803567890);

1 row created.

SQL> INSERT into Vendor VALUES (207, 'Google', 'Sinamangal', 9877663399);

1 row created.

SQL> COMMIT;

Commit complete.
```

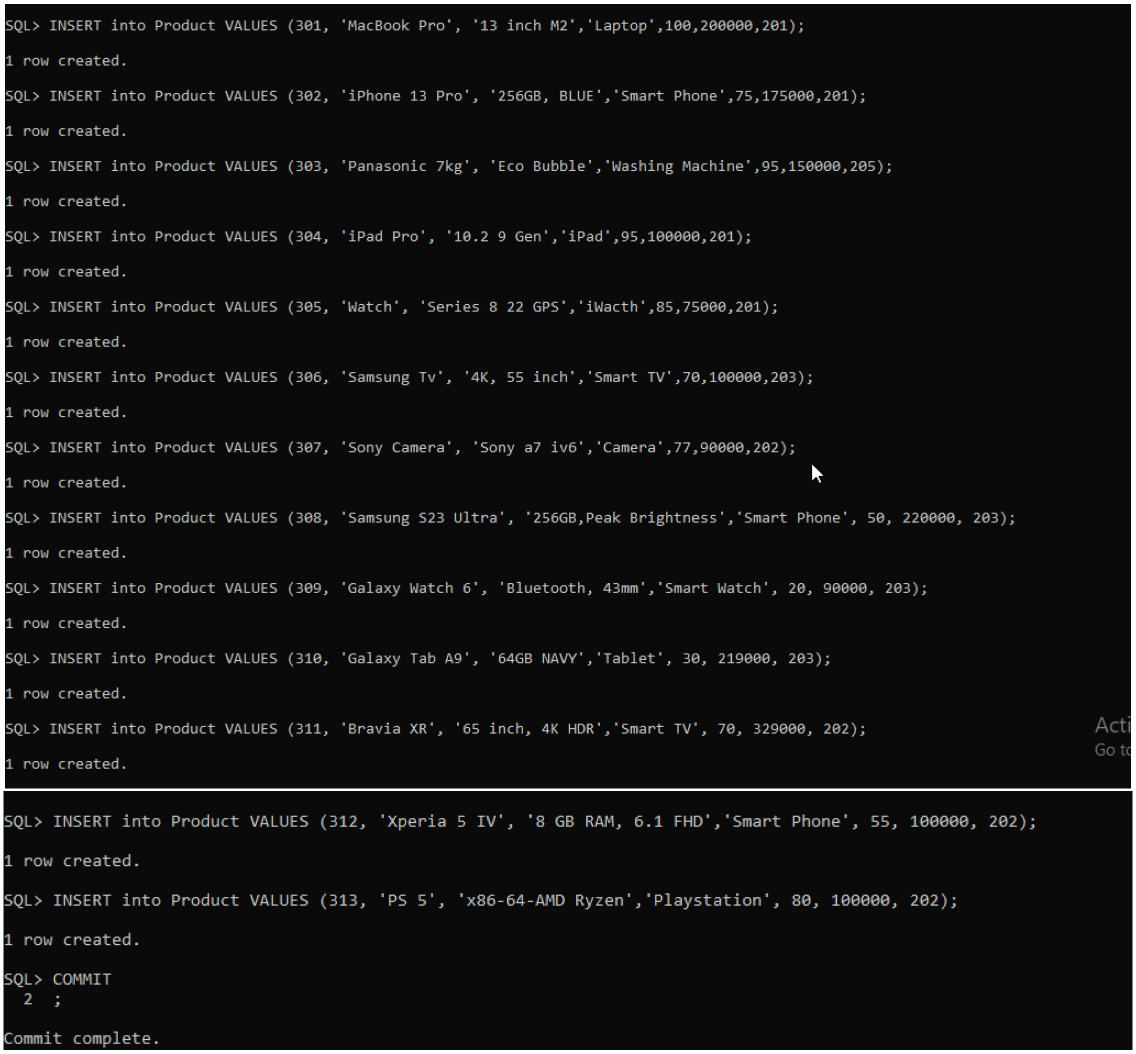
Figure 16 Inserting Value in Vendor Table



```
SQL> SELECT * FROM Vendor;

 VENDOR_ID VENDOR_NAME          VENDOR_ADDRESS                 VENDOR_CONTACT_NUMBER
---------- -------------------- ------------------------------ ---------------------
       201 Apple                Ason Bazaar                               9810697810
       202 Sony                 Lazimpat                                  9841167921
       203 Samsung              NayaBazaar                                9800235678
       204 Panasonic            Samakhushi                                9811356677
       205 MI                   Mehpi                                     9823568913
       206 Microsoft            Anamnagar                                 9803567890
       207 Google               Sinamangal                                9877663399

7 rows selected.

SQL>
```

Figure 17 Viewing Value Inserted in Vendor Table

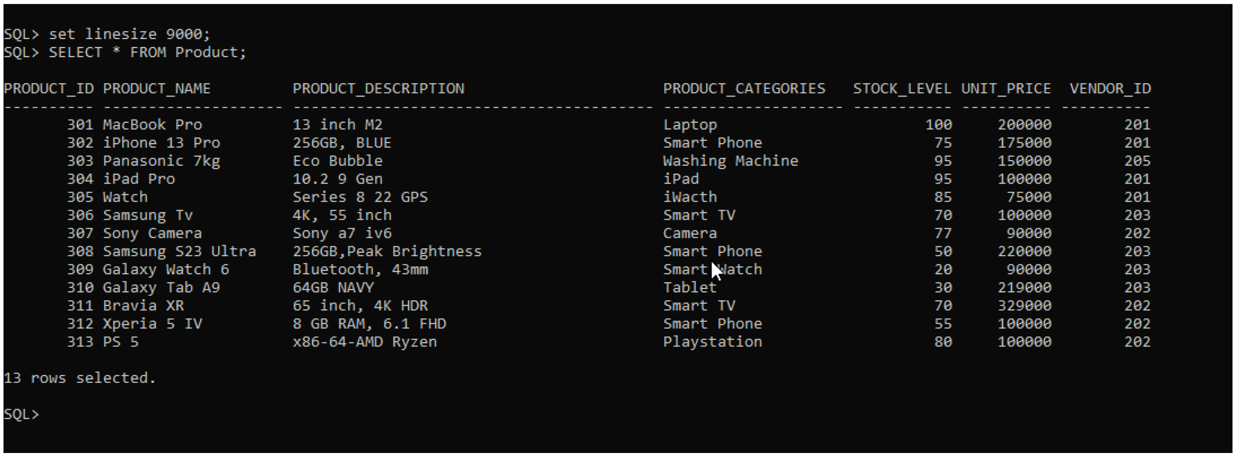
Inserting into Product table



```
SQL> INSERT into Product VALUES (301, 'MacBook Pro', '13 inch M2','Laptop',100,200000,201);

1 row created.

SQL> INSERT into Product VALUES (302, 'iPhone 13 Pro', '256GB, BLUE','Smart Phone',75,175000,201);

1 row created.

SQL> INSERT into Product VALUES (303, 'Panasonic 7kg', 'Eco Bubble','Washing Machine',95,150000,205);

1 row created.

SQL> INSERT into Product VALUES (304, 'iPad Pro', '10.2 9 Gen','iPad',95,100000,201);

1 row created.

SQL> INSERT into Product VALUES (305, 'Watch', 'Series 8 22 GPS','iWacth',85,75000,201);

1 row created.

SQL> INSERT into Product VALUES (306, 'Samsung Tv', '4K, 55 inch','Smart TV',70,100000,203);

1 row created.

SQL> INSERT into Product VALUES (307, 'Sony Camera', 'Sony a7 iv6','Camera',77,90000,202);

1 row created.

SQL> INSERT into Product VALUES (308, 'Samsung S23 Ultra', '256GB,Peak Brightness','Smart Phone', 50, 220000, 203);

1 row created.

SQL> INSERT into Product VALUES (309, 'Galaxy Watch 6', 'Bluetooth, 43mm','Smart Watch', 20, 90000, 203);

1 row created.

SQL> INSERT into Product VALUES (310, 'Galaxy Tab A9', '64GB NAVY','Tablet', 30, 219000, 203);

1 row created.

SQL> INSERT into Product VALUES (311, 'Bravia XR', '65 inch, 4K HDR','Smart TV', 70, 329000, 202);

1 row created.
```

```
SQL> INSERT into Product VALUES (312, 'Xperia 5 IV', '8 GB RAM, 6.1 FHD','Smart Phone', 55, 100000, 202);

1 row created.

SQL> INSERT into Product VALUES (313, 'PS 5', 'x86-64-AMD Ryzen','Playstation', 80, 100000, 202);

1 row created.

SQL> COMMIT
  2  ;

Commit complete.
```

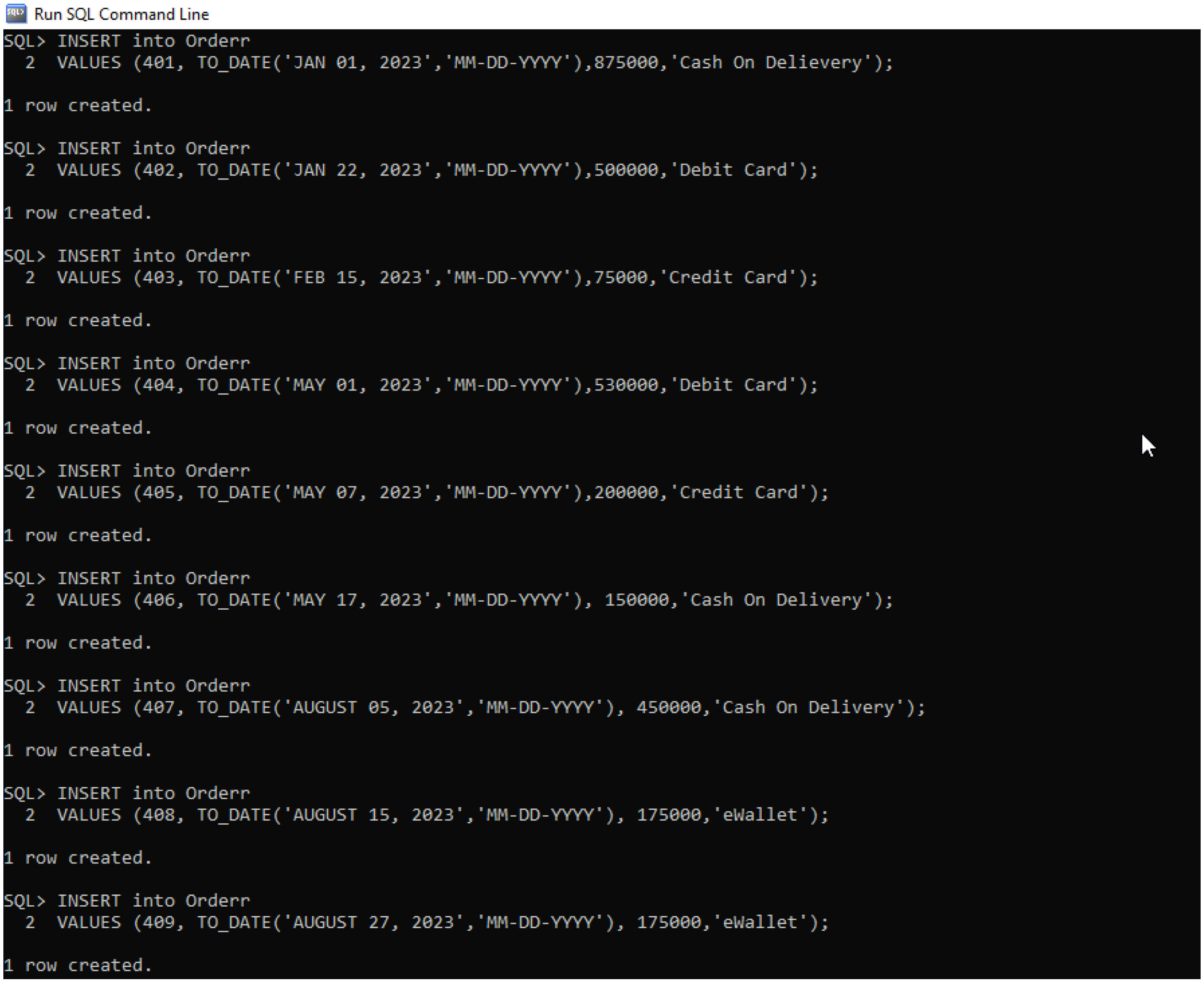
Figure 18 Inserting Value in Product Table



```
SQL> set linesize 9000;
SQL> SELECT * FROM Product;

PRODUCT_ID PRODUCT_NAME         PRODUCT_DESCRIPTION                      PRODUCT_CATEGORIES   STOCK_LEVEL UNIT_PRICE  VENDOR_ID
---------- -------------------- ---------------------------------------- -------------------- ----------- ---------- ----------
       301 MacBook Pro          13 inch M2                               Laptop                       100     200000        201
       302 iPhone 13 Pro        256GB, BLUE                              Smart Phone                   75     175000        201
       303 Panasonic 7kg        Eco Bubble                               Washing Machine               95     150000        205
       304 iPad Pro             10.2 9 Gen                               iPad                          95     100000        201
       305 Watch                Series 8 22 GPS                          iWacth                        85      75000        201
       306 Samsung Tv           4K, 55 inch                              Smart TV                      70     100000        203
       307 Sony Camera          Sony a7 iv6                              Camera                        77      90000        202
       308 Samsung S23 Ultra    256GB,Peak Brightness                    Smart Phone                   50     220000        203
       309 Galaxy Watch 6       Bluetooth, 43mm                          Smart Watch                   20      90000        203
       310 Galaxy Tab A9        64GB NAVY                                Tablet                        30     219000        203
       311 Bravia XR            65 inch, 4K HDR                          Smart TV                      70     329000        202
       312 Xperia 5 IV          8 GB RAM, 6.1 FHD                        Smart Phone                   55     100000        202
       313 PS 5                 x86-64-AMD Ryzen                         Playstation                   80     100000        202

13 rows selected.

SQL>
```

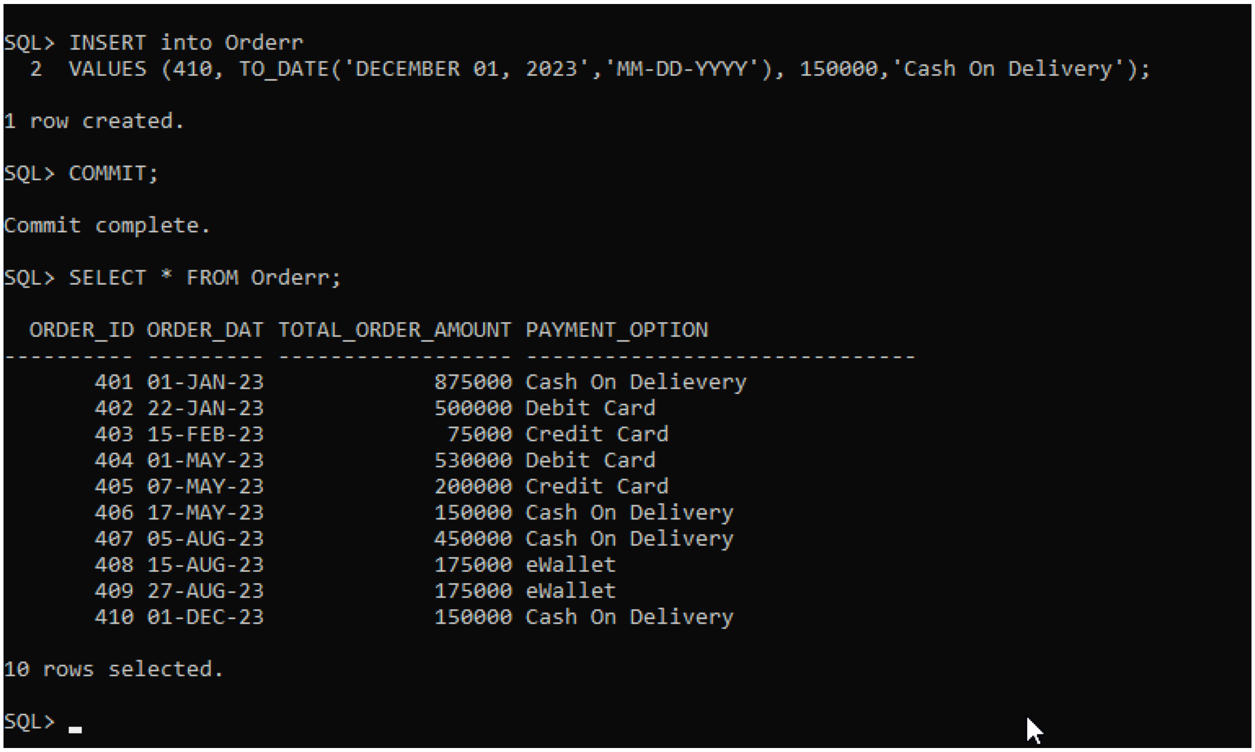
Figure 19 Viewing Inserted Value in Product Table

Inserting into Orderr Table



```
SQL> INSERT into Orderr
  2  VALUES (401, TO_DATE('JAN 01, 2023','MM-DD-YYYY'),875000,'Cash On Delievery');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (402, TO_DATE('JAN 22, 2023','MM-DD-YYYY'),500000,'Debit Card');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (403, TO_DATE('FEB 15, 2023','MM-DD-YYYY'),75000,'Credit Card');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (404, TO_DATE('MAY 01, 2023','MM-DD-YYYY'),530000,'Debit Card');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (405, TO_DATE('MAY 07, 2023','MM-DD-YYYY'),200000,'Credit Card');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (406, TO_DATE('MAY 17, 2023','MM-DD-YYYY'), 150000,'Cash On Delivery');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (407, TO_DATE('AUGUST 05, 2023','MM-DD-YYYY'), 450000,'Cash On Delivery');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (408, TO_DATE('AUGUST 15, 2023','MM-DD-YYYY'), 175000,'eWallet');

1 row created.

SQL> INSERT into Orderr
  2  VALUES (409, TO_DATE('AUGUST 27, 2023','MM-DD-YYYY'), 175000,'eWallet');

1 row created.
```

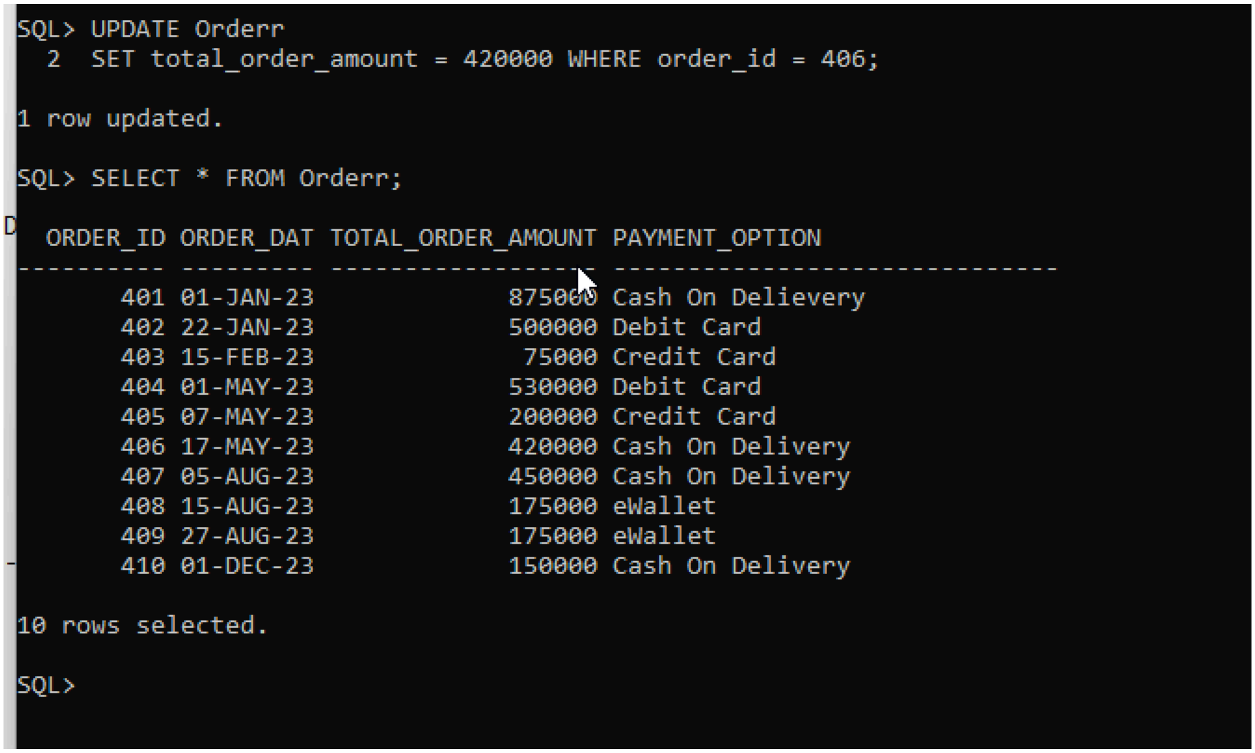
Figure 20 Inserting Values in Orderr Table



```
SQL> INSERT into Orderr
  2  VALUES (410, TO_DATE('DECEMBER 01, 2023','MM-DD-YYYY'), 150000,'Cash On Delivery');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM Orderr;

  ORDER_ID ORDER_DAT TOTAL_ORDER_AMOUNT PAYMENT_OPTION
---------- --------- ------------------ ------------------------------
       401 01-JAN-23             875000 Cash On Delievery
       402 22-JAN-23             500000 Debit Card
       403 15-FEB-23              75000 Credit Card
       404 01-MAY-23             530000 Debit Card
       405 07-MAY-23             200000 Credit Card
       406 17-MAY-23             150000 Cash On Delivery
       407 05-AUG-23             450000 Cash On Delivery
       408 15-AUG-23             175000 eWallet
       409 27-AUG-23             175000 eWallet
       410 01-DEC-23             150000 Cash On Delivery

10 rows selected.

SQL>
```

Figure 21 Inserting and Viewing inserted Value in Orderr Table

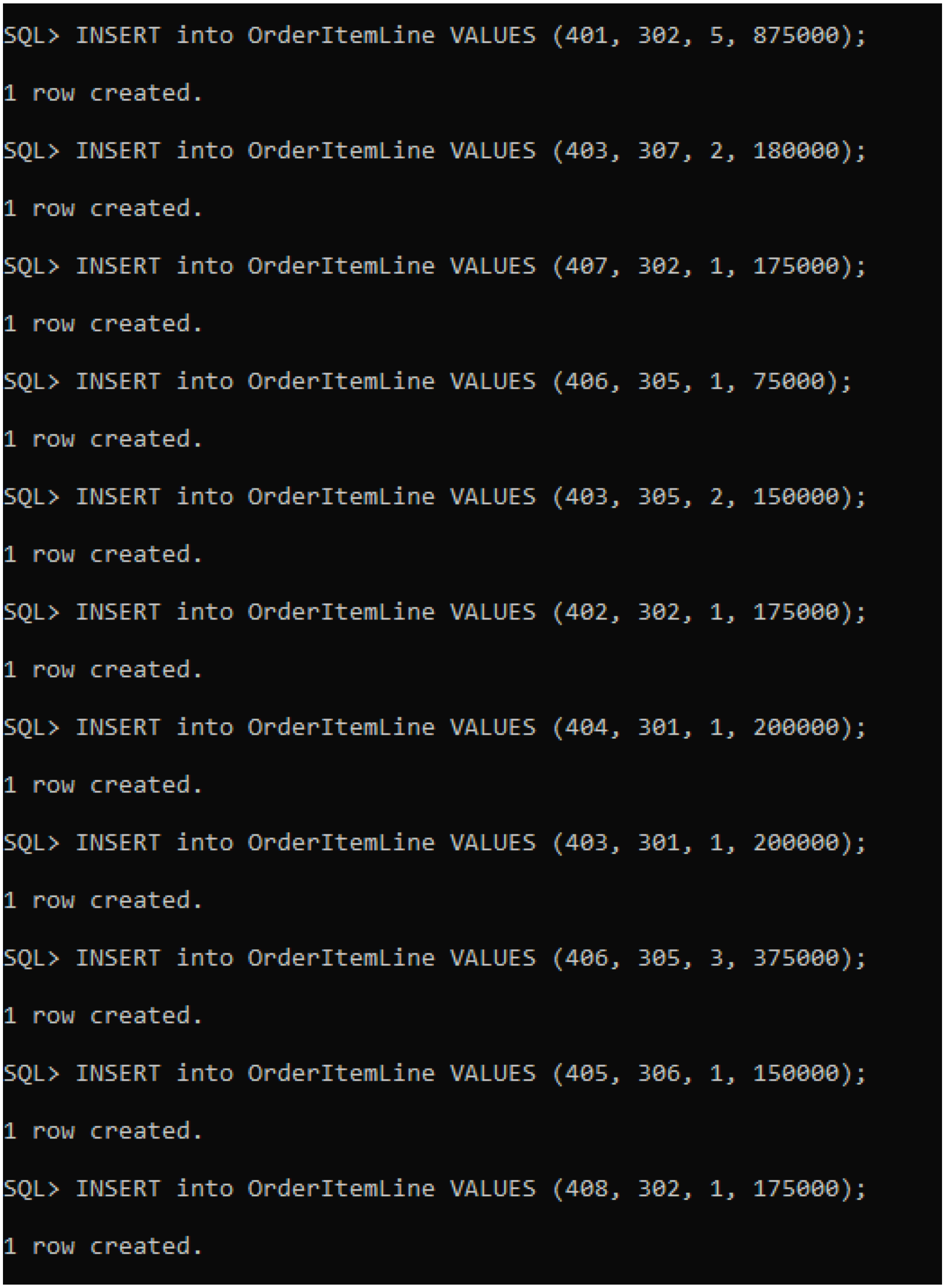
Here I had mistakenly given the wrong total_order_amount. Therefore, I have updated it.



```
SQL> UPDATE Orderr
  2  SET total_order_amount = 420000 WHERE order_id = 406;

1 row updated.

SQL> SELECT * FROM Orderr;

  ORDER_ID ORDER_DAT TOTAL_ORDER_AMOUNT PAYMENT_OPTION
---------- --------- ------------------ ------------------------------
       401 01-JAN-23             875000 Cash On Delievery
       402 22-JAN-23             500000 Debit Card
       403 15-FEB-23              75000 Credit Card
       404 01-MAY-23             530000 Debit Card
       405 07-MAY-23             200000 Credit Card
       406 17-MAY-23             420000 Cash On Delivery
       407 05-AUG-23             450000 Cash On Delivery
       408 15-AUG-23             175000 eWallet
       409 27-AUG-23             175000 eWallet
       410 01-DEC-23             150000 Cash On Delivery

10 rows selected.

SQL>
```

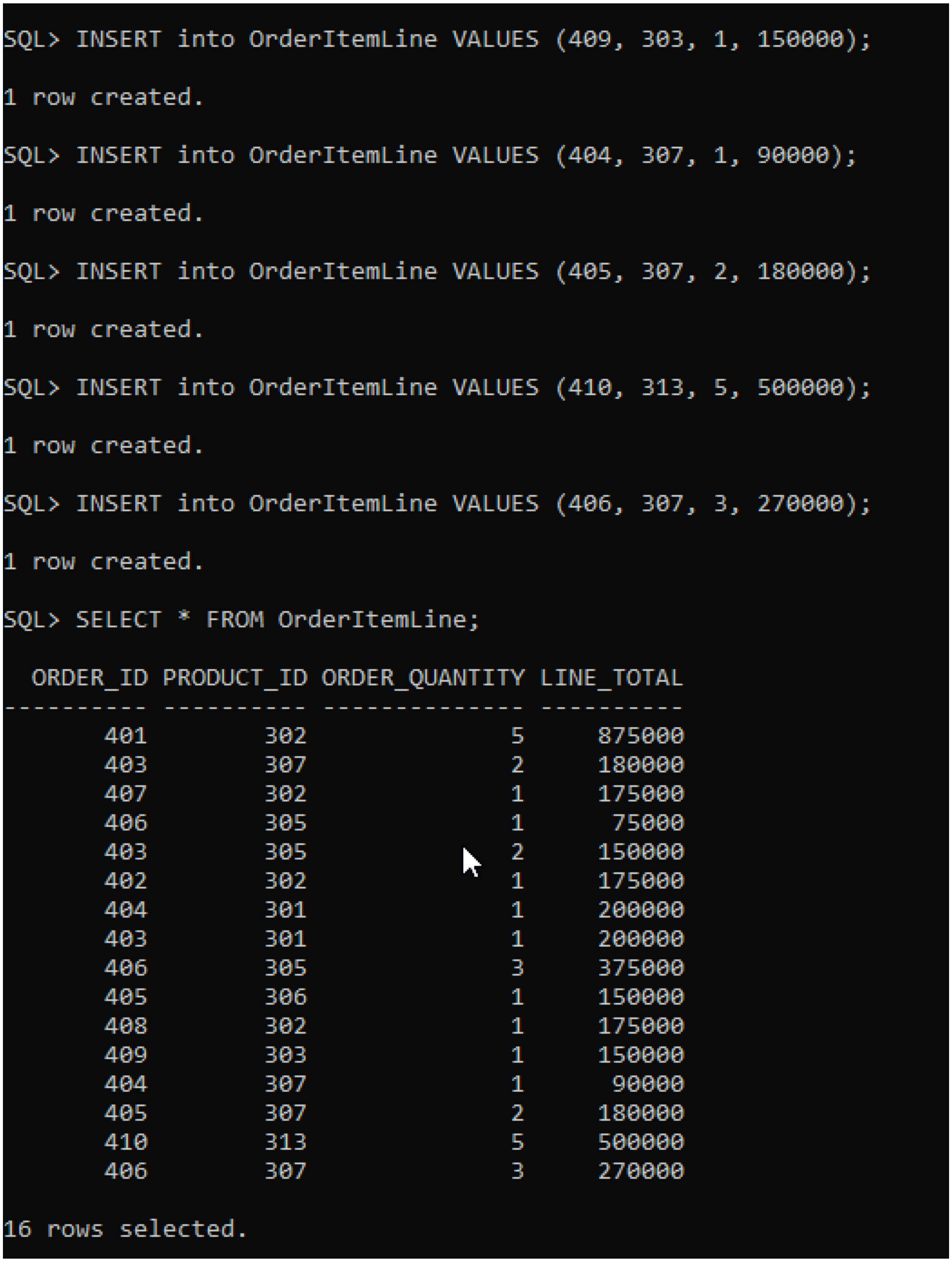
Figure 22 Updating the total_order-amount in Orderr Table

Inserting into OrderItemLine table



```
SQL> INSERT into OrderItemLine VALUES (401, 302, 5, 875000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (403, 307, 2, 180000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (407, 302, 1, 175000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (406, 305, 1, 75000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (403, 305, 2, 150000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (402, 302, 1, 175000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (404, 301, 1, 200000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (403, 301, 1, 200000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (406, 305, 3, 375000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (405, 306, 1, 150000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (408, 302, 1, 175000);

1 row created.
```

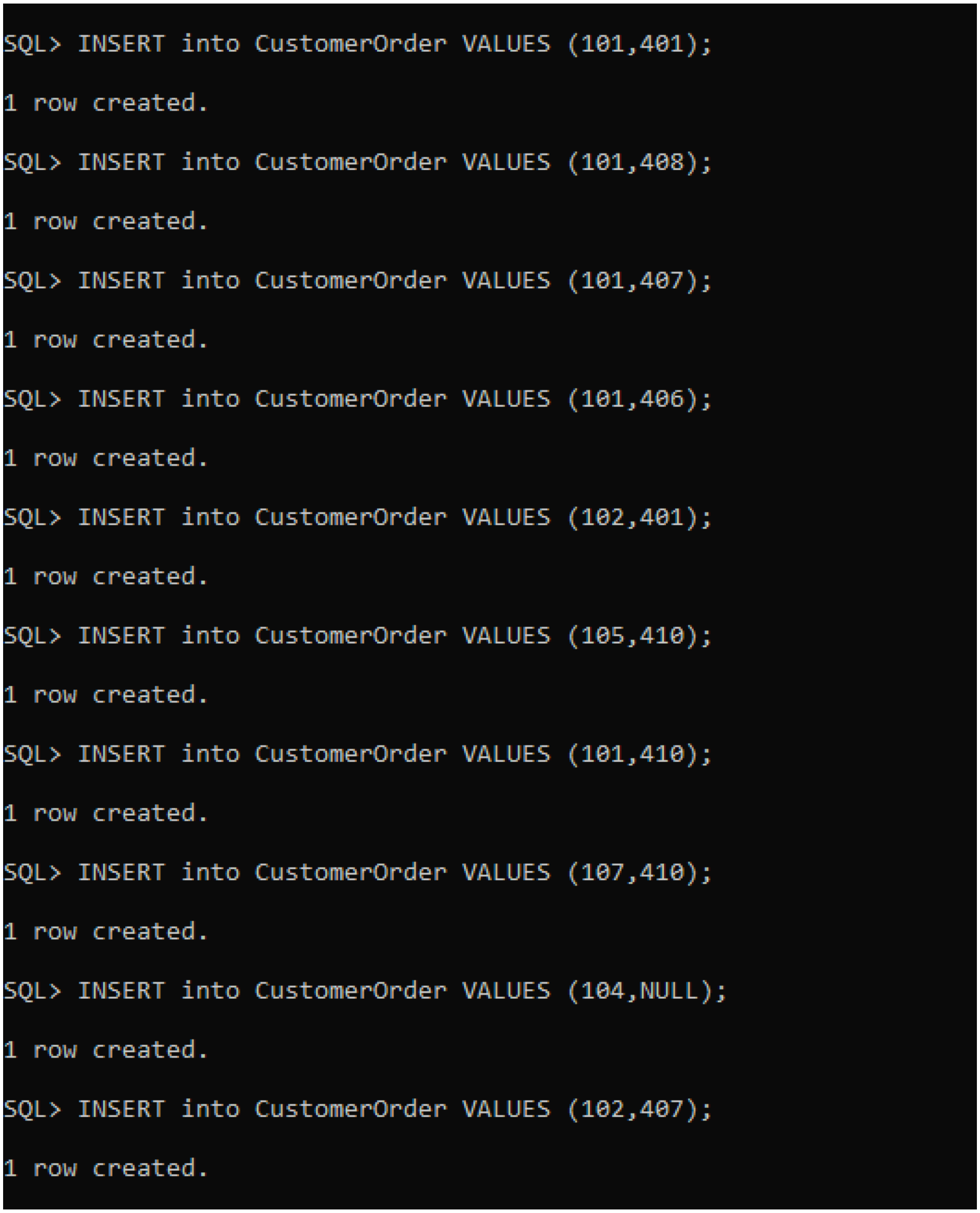
Figure 23 Inserting Values in the OrderItemLine Table



```
SQL> INSERT into OrderItemLine VALUES (409, 303, 1, 150000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (404, 307, 1, 90000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (405, 307, 2, 180000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (410, 313, 5, 500000);

1 row created.

SQL> INSERT into OrderItemLine VALUES (406, 307, 3, 270000);

1 row created.

SQL> SELECT * FROM OrderItemLine;

  ORDER_ID PRODUCT_ID ORDER_QUANTITY LINE_TOTAL
---------- ---------- -------------- ----------
       401        302              5     875000
       403        307              2     180000
       407        302              1     175000
       406        305              1      75000
       403        305              2     150000
       402        302              1     175000
       404        301              1     200000
       403        301              1     200000
       406        305              3     375000
       405        306              1     150000
       408        302              1     175000
       409        303              1     150000
       404        307              1      90000
       405        307              2     180000
       410        313              5     500000
       406        307              3     270000

16 rows selected.
```

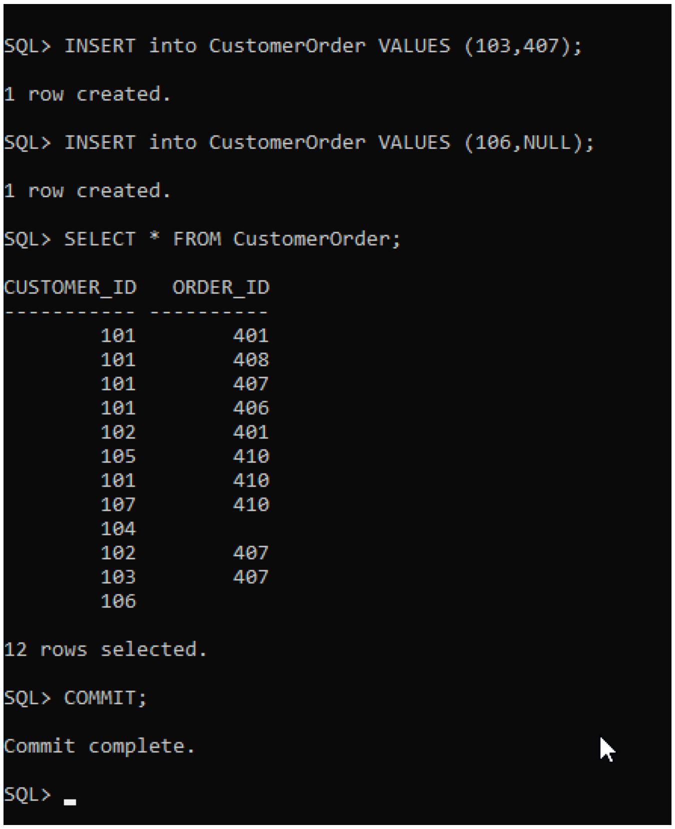
Figure 24 Inserting Value and Viewing inserted value in OrderItemLine table

Inserting into CustomerOrder



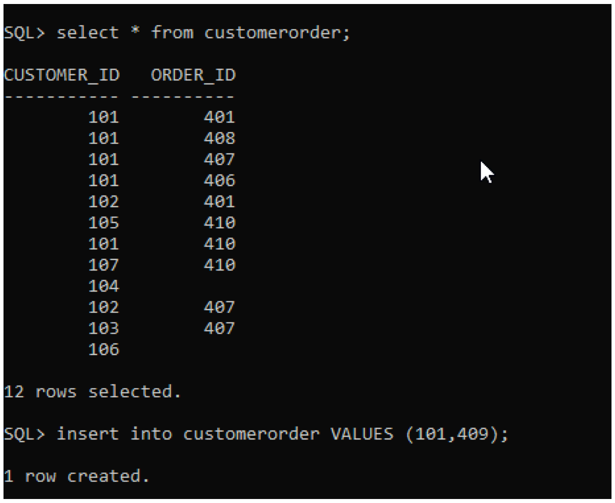
```
SQL> INSERT into CustomerOrder VALUES (101,401);

1 row created.

SQL> INSERT into CustomerOrder VALUES (101,408);

1 row created.

SQL> INSERT into CustomerOrder VALUES (101,407);

1 row created.

SQL> INSERT into CustomerOrder VALUES (101,406);

1 row created.

SQL> INSERT into CustomerOrder VALUES (102,401);

1 row created.

SQL> INSERT into CustomerOrder VALUES (105,410);

1 row created.

SQL> INSERT into CustomerOrder VALUES (101,410);

1 row created.

SQL> INSERT into CustomerOrder VALUES (107,410);

1 row created.

SQL> INSERT into CustomerOrder VALUES (104,NULL);

1 row created.

SQL> INSERT into CustomerOrder VALUES (102,407);

1 row created.
```

Figure 25 Inserting Value in CustomerOrder table



```
SQL> INSERT into CustomerOrder VALUES (103,407);

1 row created.

SQL> INSERT into CustomerOrder VALUES (106,NULL);

1 row created.

SQL> SELECT * FROM CustomerOrder;

CUSTOMER_ID   ORDER_ID
----------- ----------
        101        401
        101        408
        101        407
        101        406
        102        401
        105        410
        101        410
        107        410
        104
        102        407
        103        407
        106

12 rows selected.

SQL> COMMIT;

Commit complete.

SQL> _
```

Figure 26 Inserting Value and Viewing inserted value in CustomerProduct. Table

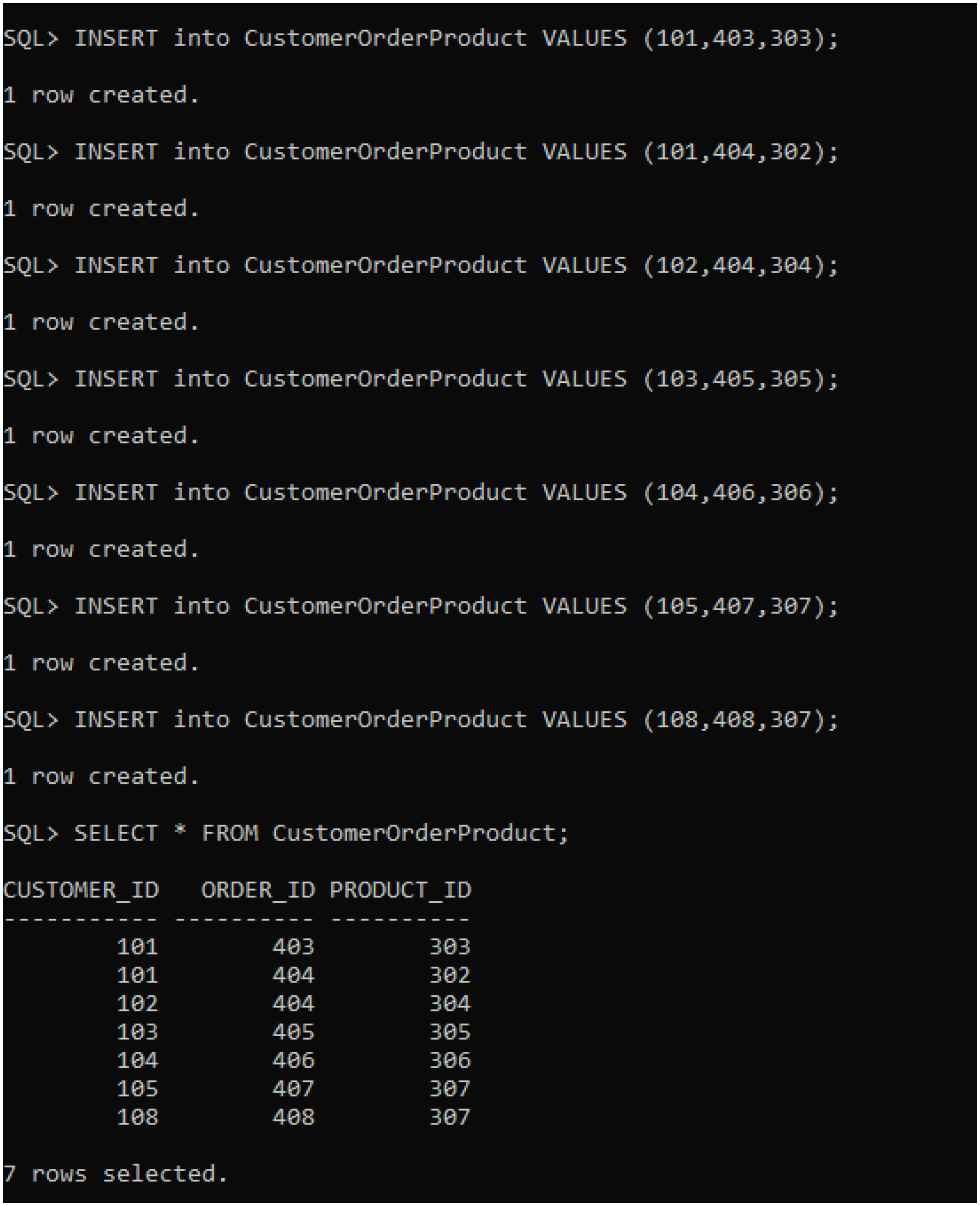


```
SQL> select * from customerorder;

CUSTOMER_ID   ORDER_ID
----------- ----------
        101        401
        101        408
        101        407
        101        406
        102        401
        105        410
        101        410
        107        410
        104
        102        407
        103        407
        106

12 rows selected.

SQL> insert into customerorder VALUES (101,409);

1 row created.
```

Figure 27 Updating CustomerOrder

Here I have inserted one value because the query was not giving the output. I had forgotten to give one value.

Inserting into CustomerOrderProduct



```
SQL> INSERT into CustomerOrderProduct VALUES (101,403,303);

1 row created.

SQL> INSERT into CustomerOrderProduct VALUES (101,404,302);

1 row created.

SQL> INSERT into CustomerOrderProduct VALUES (102,404,304);

1 row created.

SQL> INSERT into CustomerOrderProduct VALUES (103,405,305);

1 row created.

SQL> INSERT into CustomerOrderProduct VALUES (104,406,306);

1 row created.

SQL> INSERT into CustomerOrderProduct VALUES (105,407,307);

1 row created.

SQL> INSERT into CustomerOrderProduct VALUES (108,408,307);

1 row created.

SQL> SELECT * FROM CustomerOrderProduct;

CUSTOMER_ID   ORDER_ID PRODUCT_ID
----------- ---------- ----------
        101        403        303
        101        404        302
        102        404        304
        103        405        305
        104        406        306
        105        407        307
        108        408        307

7 rows selected.
```

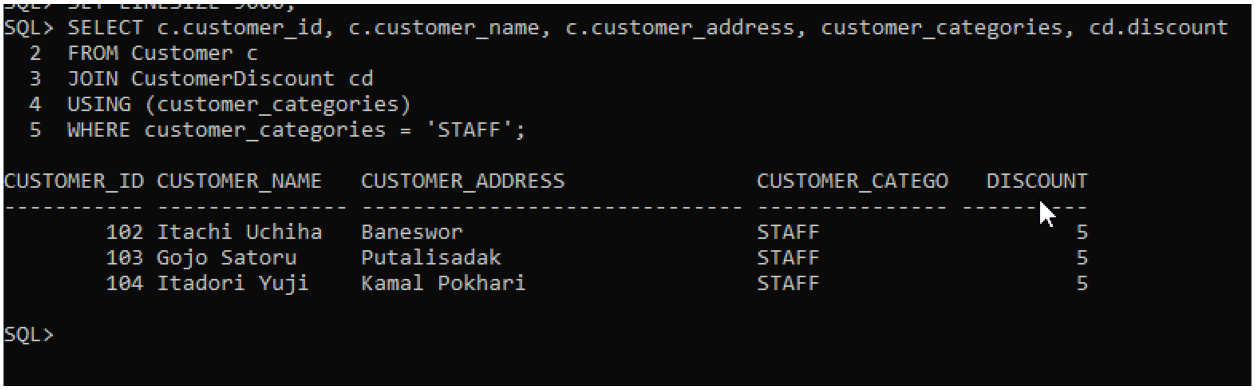
Figure 28 Inserting Value and Viewing inserted value in CustomerOrderProduct table

8. Database Query

Requesting data from the database, basic question compared to a collection of data is called Database Query. (Gibbs, 2023) In database using a specific syntax writing database query helps you to access, manipulate, update, delete, insert data in relational database. (SOLARWIND, 2023)

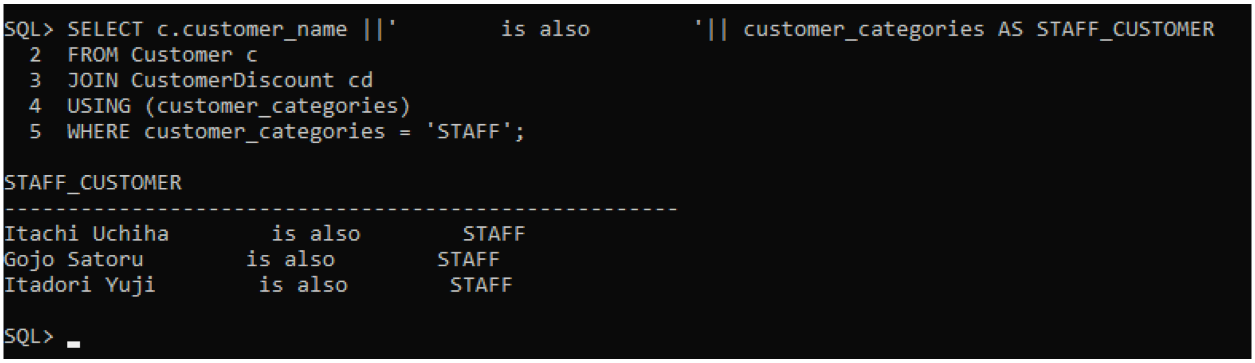
8.1 Information query

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



```
SQL> SET LINESIZE 3000;
SQL> SELECT c.customer_id, c.customer_name, c.customer_address, customer_categories, cd.discount
     FROM Customer c
     JOIN CustomerDiscount cd
     USING (customer_categories)
     WHERE customer_categories = 'STAFF';

CUSTOMER_ID CUSTOMER_NAME   CUSTOMER_ADDRESS               CUSTOMER_CATEGO   DISCOUNT
----------- --------------- ------------------------------ --------------- ----------
        102 Itachi Uchiha   Baneswor                       STAFF                    5
        103 Gojo Satoru     Putalisadak                    STAFF                    5
        104 Itadori Yuji    Kamal Pokhari                  STAFF                    5

SQL>
```

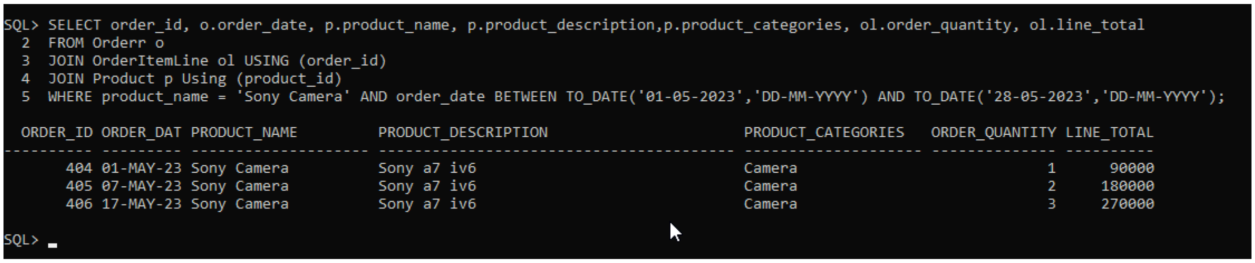
Figure 29 Information Query No.1



```
SQL> SELECT c.customer_name ||'        is also        '|| customer_categories AS STAFF_CUSTOMER
     FROM Customer c
     JOIN CustomerDiscount cd
     USING (customer_categories)
     WHERE customer_categories = 'STAFF';

STAFF_CUSTOMER
-----------------------------------------------------
Itachi Uchiha        is also        STAFF
Gojo Satoru        is also        STAFF
Itadori Yuji        is also        STAFF

SQL>
```

Figure 30 Information Query No 1 Using Concatenation Operator

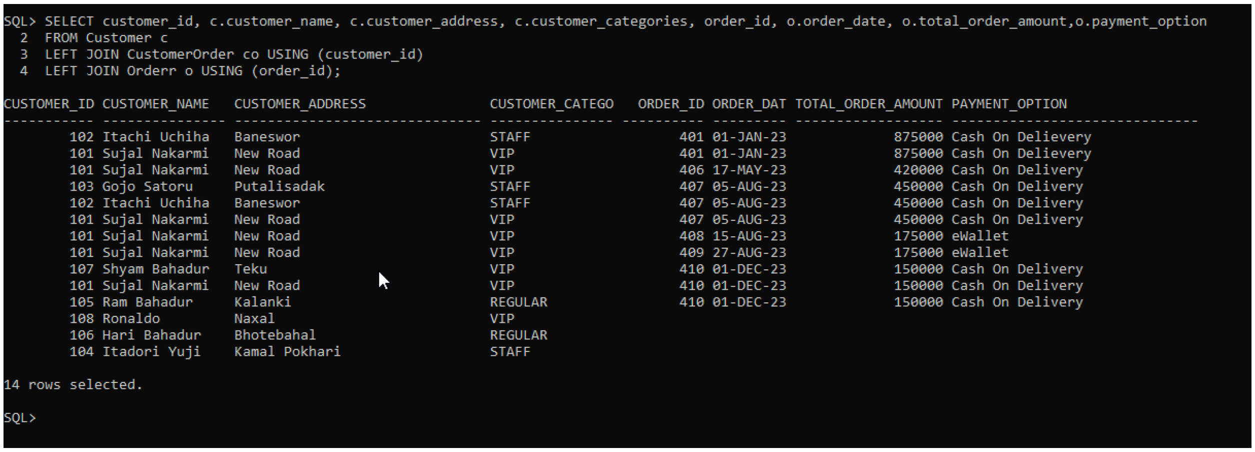
2. List all the orders made for any particular product between the dates 01-05-2023 till 28-05-2023.



```
SQL> SELECT order_id, o.order_date, p.product_name, p.product_description,p.product_categories, ol.order_quantity, ol.line_total
     FROM Orderr o
     JOIN OrderItemLine ol USING (order_id)
     JOIN Product p Using (product_id)
     WHERE product_name = 'Sony Camera' AND order_date BETWEEN TO_DATE('01-05-2023','DD-MM-YYYY') AND TO_DATE('28-05-2023','DD-MM-YYYY');

  ORDER_ID ORDER_DAT PRODUCT_NAME         PRODUCT_DESCRIPTION                      PRODUCT_CATEGORIES   ORDER_QUANTITY LINE_TOTAL
---------- --------- -------------------- ---------------------------------------- -------------------- -------------- ----------
       404 01-MAY-23 Sony Camera          Sony a7 iv6                              Camera                            1      90000
       405 07-MAY-23 Sony Camera          Sony a7 iv6                              Camera                            2     180000
       406 17-MAY-23 Sony Camera          Sony a7 iv6                              Camera                            3     270000

SQL>
```

Figure 31 Listing All the order made between May 1 to May 28 of product name Sony

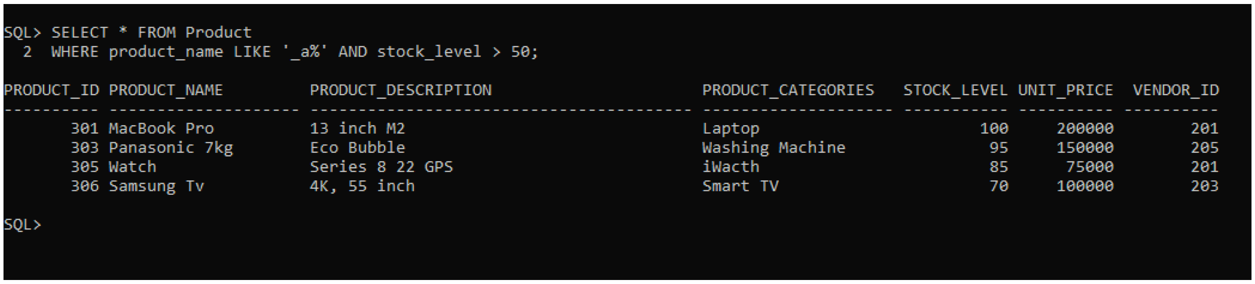
3. List all the customers with their order details and also the customers who have not ordered any products yet.



```
SQL> SELECT customer_id, c.customer_name, c.customer_address, c.customer_categories, order_id, o.order_date, o.total_order_amount,o.payment_option
     FROM Customer c
     LEFT JOIN CustomerOrder co USING (customer_id)
     LEFT JOIN Orderr o USING (order_id);

CUSTOMER_ID CUSTOMER_NAME   CUSTOMER_ADDRESS               CUSTOMER_CATEGO   ORDER_ID ORDER_DAT TOTAL_ORDER_AMOUNT PAYMENT_OPTION
----------- --------------- ------------------------------ --------------- ---------- --------- ------------------ ------------------------------
        102 Itachi Uchiha   Baneswor                       STAFF                  401 01-JAN-23             875000 Cash On Delievery
        101 Sujal Nakarmi   New Road                       VIP                    401 01-JAN-23             875000 Cash On Delievery
        101 Sujal Nakarmi   New Road                       VIP                    406 17-MAY-23             420000 Cash On Delivery
        103 Gojo Satoru     Putalisadak                    STAFF                  407 05-AUG-23             450000 Cash On Delivery
        102 Itachi Uchiha   Baneswor                       STAFF                  407 05-AUG-23             450000 Cash On Delivery
        101 Sujal Nakarmi   New Road                       VIP                    407 05-AUG-23             450000 Cash On Delivery
        101 Sujal Nakarmi   New Road                       VIP                    408 15-AUG-23             175000 eWallet
        101 Sujal Nakarmi   New Road                       VIP                    409 27-AUG-23             175000 eWallet
        107 Shyam Bahadur   Teku                           VIP                    410 01-DEC-23             150000 Cash On Delivery
        101 Sujal Nakarmi   New Road                       VIP                    410 01-DEC-23             150000 Cash On Delivery
        105 Ram Bahadur     Kalanki                        REGULAR                410 01-DEC-23             150000 Cash On Delivery
        108 Ronaldo         Naxal                          VIP
        106 Hari Bahadur    Bhotebahal                     REGULAR
        104 Itadori Yuji    Kamal Pokhari                  STAFF

14 rows selected.

SQL>
```

Figure 32 Listing all the customer with their order details and also customer who haven't ordered any products

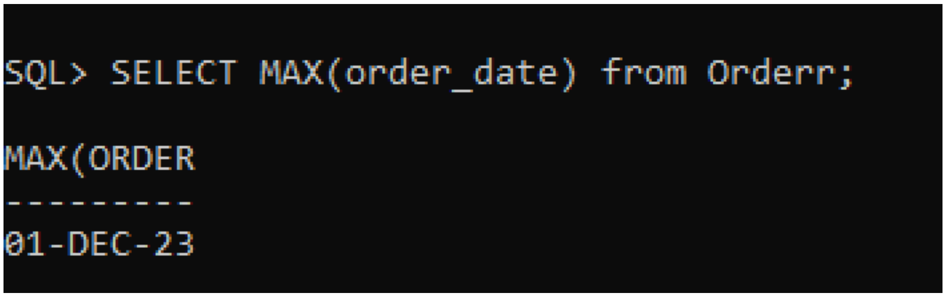
4. List all product details that have the second letter 'a' in their product name and have a stock quantity more than 50.



```
SQL> SELECT * FROM Product
     WHERE product_name LIKE '_a%' AND stock_level > 50;

PRODUCT_ID PRODUCT_NAME         PRODUCT_DESCRIPTION                      PRODUCT_CATEGORIES   STOCK_LEVEL UNIT_PRICE  VENDOR_ID
---------- -------------------- ---------------------------------------- -------------------- ----------- ---------- ----------
       301 MacBook Pro          13 inch M2                               Laptop                       100     200000        201
       303 Panasonic 7kg        Eco Bubble                               Washing Machine               95     150000        205
       305 Watch                Series 8 22 GPS                          iWacth                        85      75000        201
       306 Samsung Tv           4K, 55 inch                              Smart TV                      70     100000        203

SQL>
```

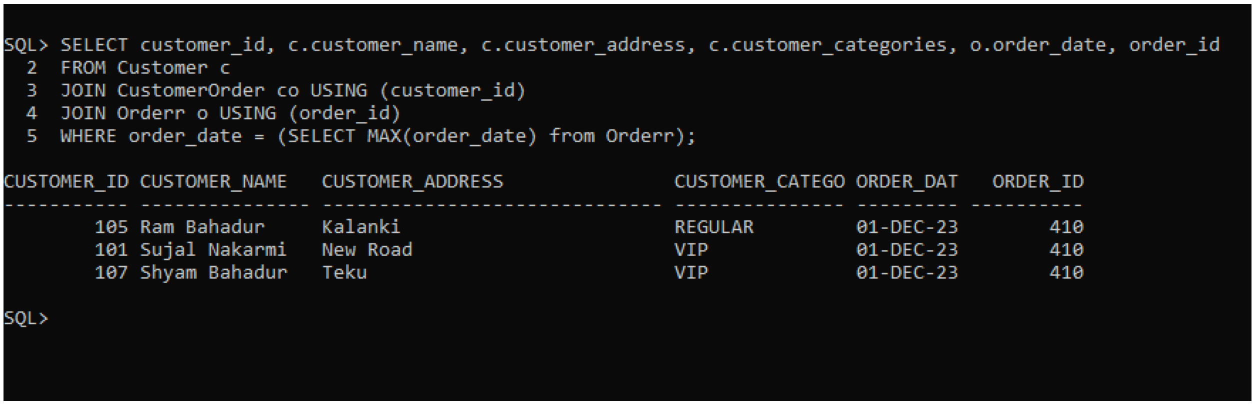
Figure 33 Listing all the product having second letter 'a' and stock greater than 50

5. Find out the customer who has ordered recently.



```
SQL> SELECT MAX(order_date) from Orderr;

MAX(ORDER
---------
01-DEC-23
```

Figure 34 Checking Most Recent Order Date

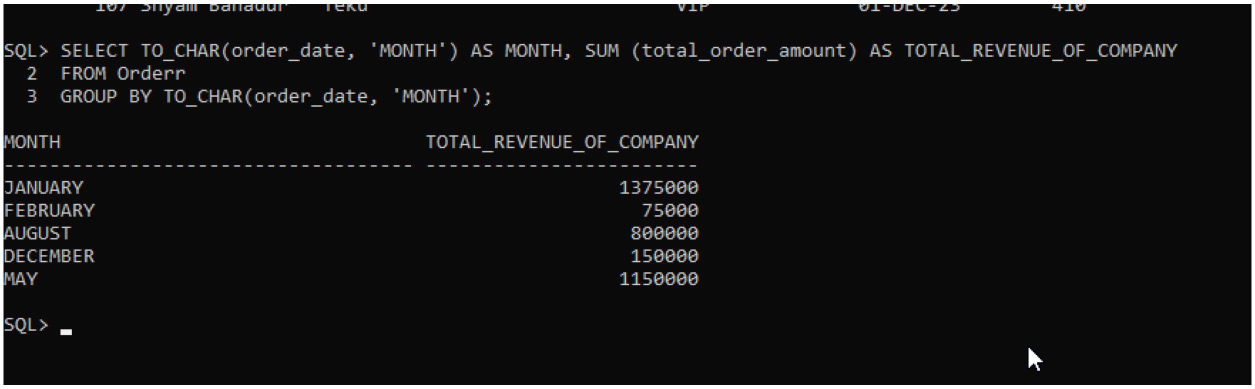


```
SQL> SELECT customer_id, c.customer_name, c.customer_address, c.customer_categories, o.order_date, order_id
  2  FROM Customer c
  3  JOIN CustomerOrder co USING (customer_id)
  4  JOIN Orderr o USING (order_id)
  5  WHERE order_date = (SELECT MAX(order_date) from Orderr);

CUSTOMER_ID CUSTOMER_NAME   CUSTOMER_ADDRESS               CUSTOMER_CATEGO ORDER_DAT   ORDER_ID
----------- --------------- ------------------------------ --------------- --------- ----------
        105 Ram Bahadur     Kalanki                        REGULAR         01-DEC-23        410
        101 Sujal Nakarmi   New Road                       VIP             01-DEC-23        410
        107 Shyam Bahadur   Teku                           VIP             01-DEC-23        410

SQL>
```

Figure 35 Selecting those customers who have ordered recently

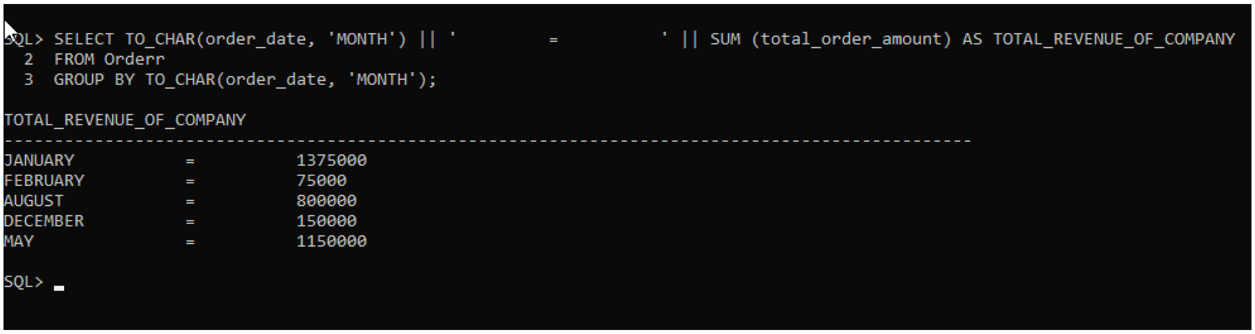
8.2 Transaction query

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



```
       107 Shyam Bahadur   Teku                         VIP            01-DEC-23        410

SQL> SELECT TO_CHAR(order_date, 'MONTH') AS MONTH, SUM (total_order_amount) AS TOTAL_REVENUE_OF_COMPANY
     FROM Orderr
     GROUP BY TO_CHAR(order_date, 'MONTH');

MONTH                                TOTAL_REVENUE_OF_COMPANY
------------------------------------ ------------------------
JANUARY                                               1375000
FEBRUARY                                                75000
AUGUST                                                 800000
DECEMBER                                               150000
MAY                                                   1150000

SQL> _
```

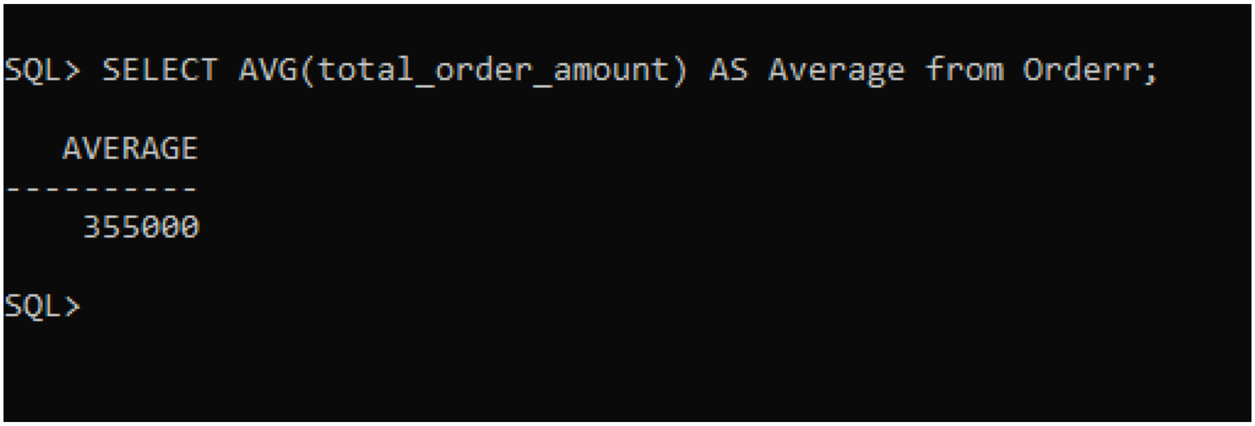
Figure 36 Total Revenue of the company from each month



```
SQL> SELECT TO_CHAR(order_date, 'MONTH') || '         =          ' || SUM (total_order_amount) AS TOTAL_REVENUE_OF_COMPANY
     FROM Orderr
     GROUP BY TO_CHAR(order_date, 'MONTH');

TOTAL_REVENUE_OF_COMPANY
--------------------------------------------------------------------------------------------
JANUARY          =          1375000
FEBRUARY         =          75000
AUGUST           =          800000
DECEMBER         =          150000
MAY              =          1150000

SQL> _
```

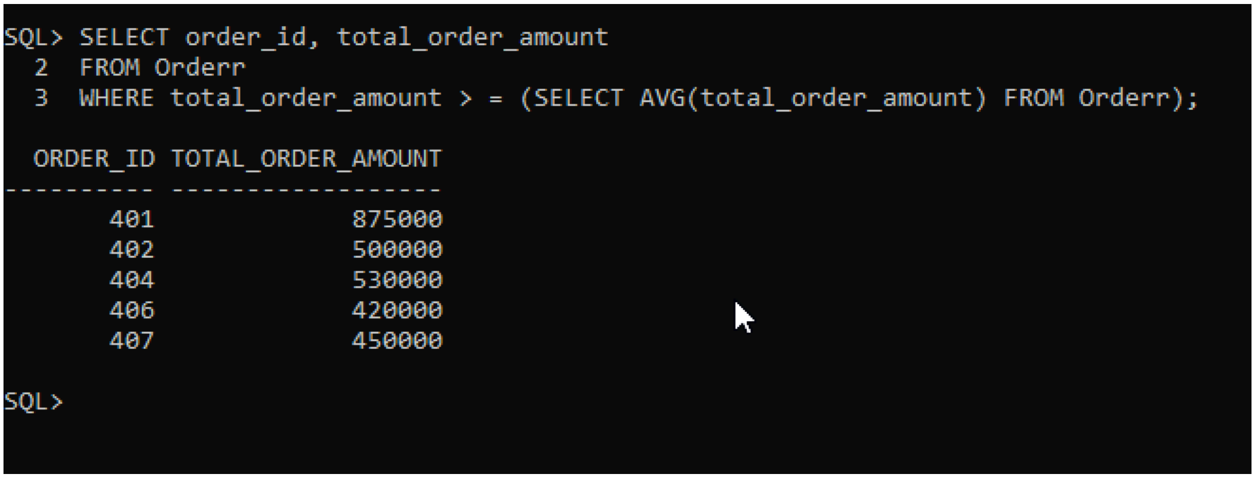
Figure 37 Total Revenue of Company From Each Month Using Concatenation Operator

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



```
SQL> SELECT AVG(total_order_amount) AS Average from Orderr;

   AVERAGE
----------
    355000

SQL>
```

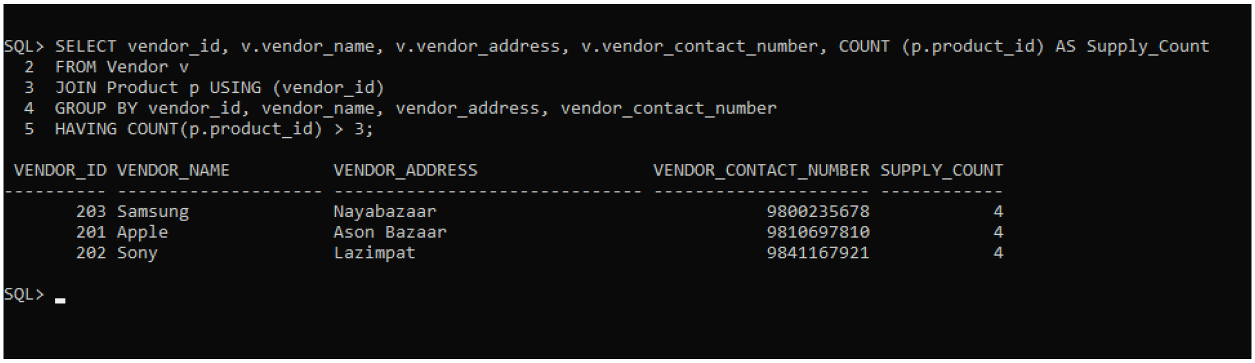
Figure 38 Calculating Average of Total Order Amount



```
SQL> SELECT order_id, total_order_amount
  2  FROM Orderr
  3  WHERE total_order_amount > = (SELECT AVG(total_order_amount) FROM Orderr);

  ORDER_ID TOTAL_ORDER_AMOUNT
---------- ------------------
       401             875000
       402             500000
       404             530000
       406             420000
       407             450000

SQL>
```

Figure 39 Listing those total order amount which are greater or equal to average of total order amount

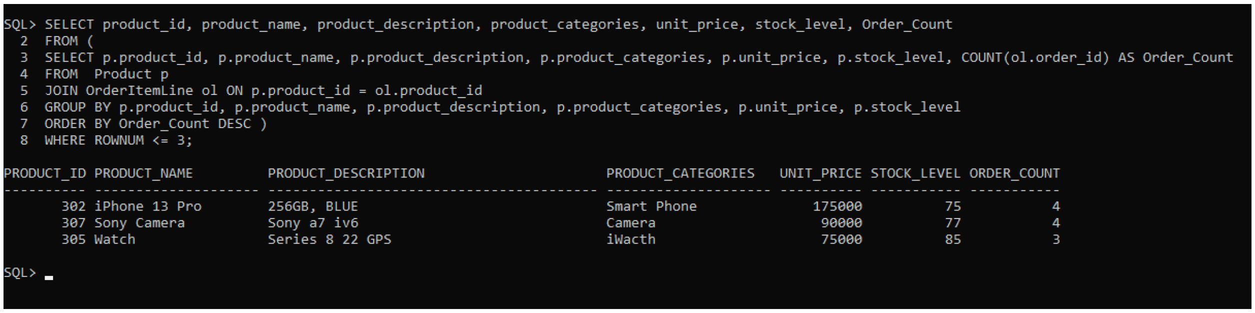
3. List the details of vendors who have supplied more than 3 products to the company.



```
SQL> SELECT vendor_id, v.vendor_name, v.vendor_address, v.vendor_contact_number, COUNT (p.product_id) AS Supply_Count
  2  FROM Vendor v
  3  JOIN Product p USING (vendor_id)
  4  GROUP BY vendor_id, vendor_name, vendor_address, vendor_contact_number
  5  HAVING COUNT(p.product_id) > 3;

 VENDOR_ID VENDOR_NAME          VENDOR_ADDRESS                 VENDOR_CONTACT_NUMBER SUPPLY_COUNT
---------- -------------------- ------------------------------ --------------------- ------------
       203 Samsung              Nayabazaar                                9800235678            4
       201 Apple                Ason Bazaar                               9810697810            4
       202 Sony                 Lazimpat                                  9841167921            4

SQL>
```

Figure 40 Listing all the vendor details who have supplied more than 3 products to company

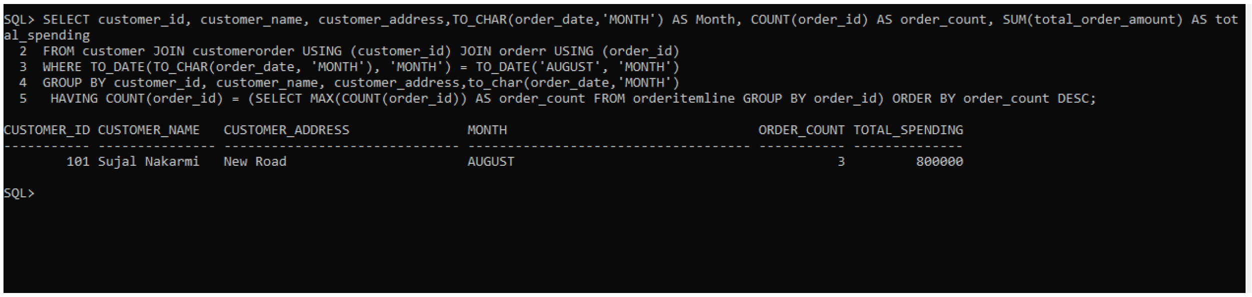
4. Show the top 3 product details that have been ordered the most.



```
SQL> SELECT product_id, product_name, product_description, product_categories, unit_price, stock_level, Order_Count
  2  FROM (
  3  SELECT p.product_id, p.product_name, p.product_description, p.product_categories, p.unit_price, p.stock_level, COUNT(ol.order_id) AS Order_Count
  4  FROM  Product p
  5  JOIN OrderItemLine ol ON p.product_id = ol.product_id
  6  GROUP BY p.product_id, p.product_name, p.product_description, p.product_categories, p.unit_price, p.stock_level
  7  ORDER BY Order_Count DESC )
  8  WHERE ROWNUM <= 3;

PRODUCT_ID PRODUCT_NAME         PRODUCT_DESCRIPTION                       PRODUCT_CATEGORIES    UNIT_PRICE STOCK_LEVEL ORDER_COUNT
---------- -------------------- ----------------------------------------- -------------------- ---------- ----------- -----------
       302 iPhone 13 Pro        256GB, BLUE                               Smart Phone              175000          75           4
       307 Sony Camera          Sony a7 iv6                               Camera                    90000          77           4
       305 Watch                Series 8 22 GPS                           iWacth                    75000          85           3

SQL>
```

Figure 41 Showing the top three product which has 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_id, customer_name, customer_address,TO_CHAR(order_date,'MONTH') AS Month, COUNT(order_id) AS order_count, SUM(total_order_amount) AS tot
al_spending
  2  FROM customer JOIN customerorder USING (customer_id) JOIN orderr USING (order_id)
  3  WHERE TO_DATE(TO_CHAR(order_date, 'MONTH'), 'MONTH') = TO_DATE('AUGUST', 'MONTH')
  4  GROUP BY customer_id, customer_name, customer_address,to_char(order_date,'MONTH')
  5   HAVING COUNT(order_id) = (SELECT MAX(COUNT(order_id)) AS order_count FROM orderitemline GROUP BY order_id) ORDER BY order_count DESC;

CUSTOMER_ID CUSTOMER_NAME   CUSTOMER_ADDRESS               MONTH                                ORDER_COUNT TOTAL_SPENDING
----------- --------------- ------------------------------ ------------------------------------ ----------- --------------
        101 Sujal Nakarmi   New Road                       AUGUST                                         3         800000

SQL>
```

Figure 42 Displaying the Customer who have order the most in AUGUST and total spending on that month

9. Critical Evaluation

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

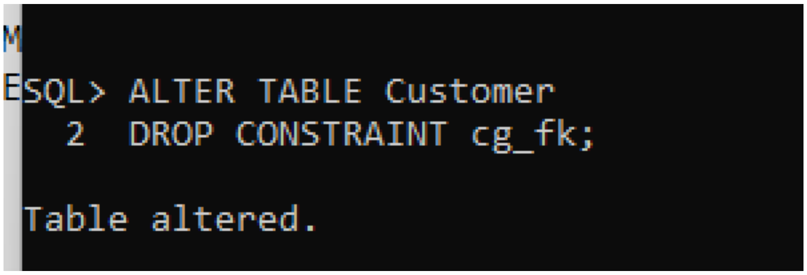
The name of the module is database. Database stores and manages large amount of data in a categorized way so that whenever we need to access some data, we can easy get it. (DatabaseTown, 2023) . For Example, Contact on our phone stores contact numbers of large numbers of people which can be easily be access from us as we can simply search the name of the person we are trying to contact. E banking, social media are the application where database plays a crucial role. As we have learnt how to make ERD and also the rules to follow while making ERD, it helped us in completion of Coursework of our Module Software Engineering. Different IOT devices transfer data which is stored in the IOT Database System. File system and Managing files are some common databases used in Operating System.

9.2 Critical Assessment of coursework.

The coursework was about to work as a Database Designer, to create a strong database design for the ecommerce which stores the details of the customer their order details as well as the products details. Customer details are crucial and personal for each customer so to store the details securely database is used. There can be a lot of orders made by a customer in one day, there can be a lot of products and details to be stored, therefore, to store huge data efficiently database is used. Normalization and Query part were well researched on the online platform like Google and YouTube for the completion of the coursework. Reviews were received from our tutor/lecturer which helped us moving forward. The knowledge about the Normalization and SQL has been gone to more dept and it also helped us to practically understand the concepts of Database. By communicating with seniors as asking them to review work has somehow improved my Communication Skills.

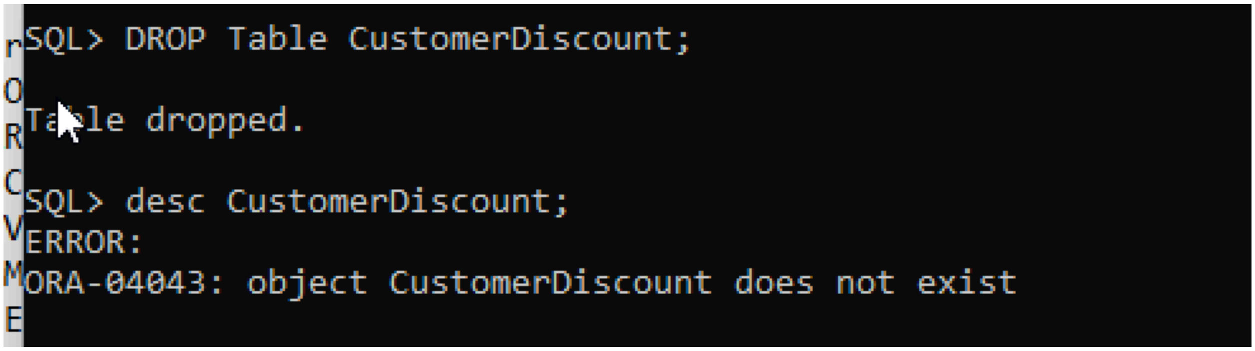
10. Dropping Table And Creating Dump File

To drop a table first we have delete all the foreign key from the respective table which are shown below.



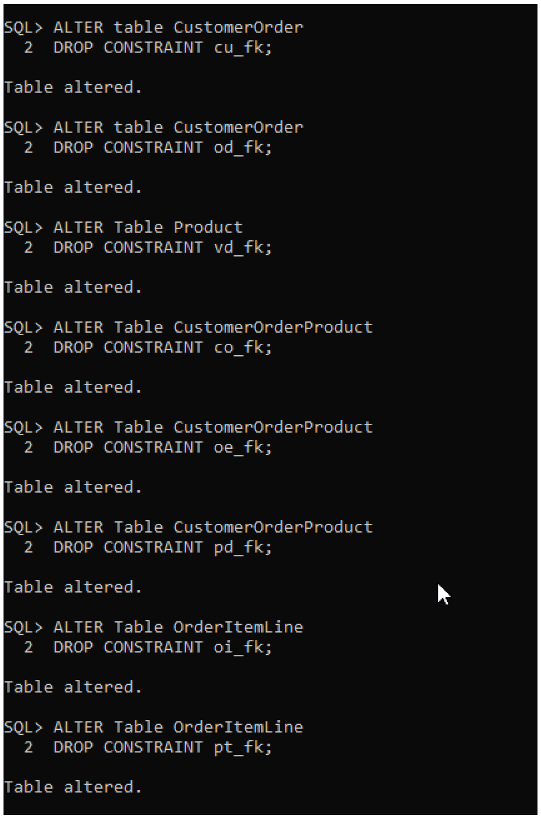
```
SQL> ALTER TABLE Customer
  2  DROP CONSTRAINT cg_fk;

Table altered.
```

Figure 43 Droping Foreign Key in Customer Table



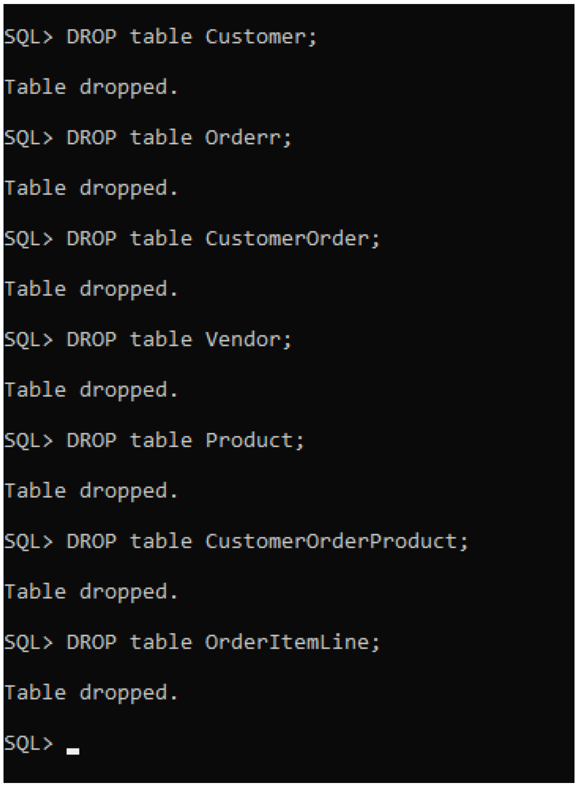
```
SQL> DROP Table CustomerDiscount;

Table dropped.

SQL> desc CustomerDiscount;
ERROR:
ORA-04043: object CustomerDiscount does not exist
```

Figure 44 Droppin CustomerDiscount Table



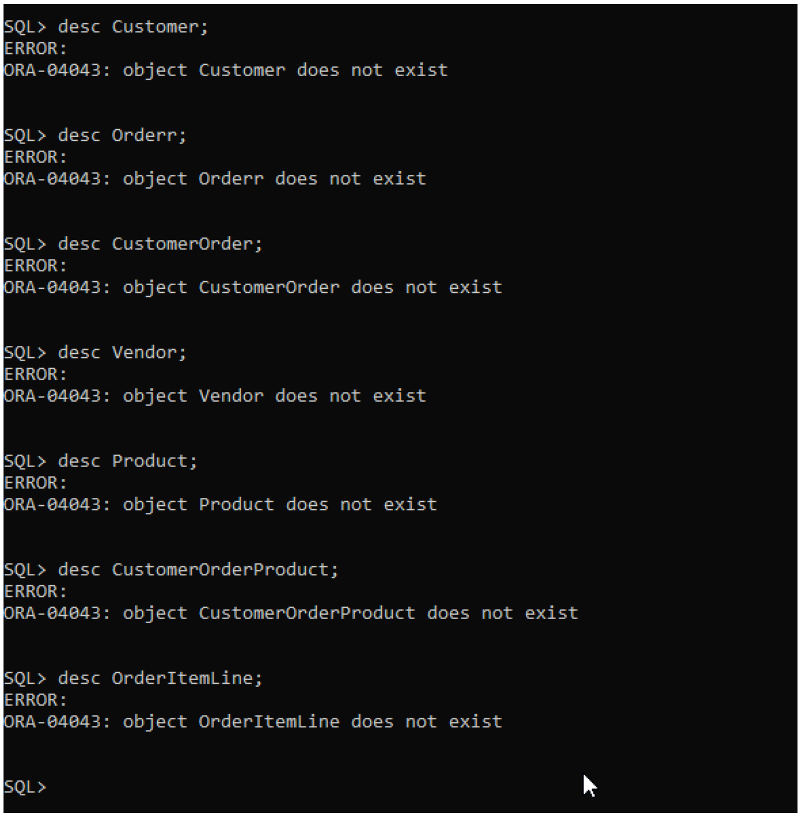
```
SQL> ALTER table CustomerOrder
  2  DROP CONSTRAINT cu_fk;

Table altered.

SQL> ALTER table CustomerOrder
  2  DROP CONSTRAINT od_fk;

Table altered.

SQL> ALTER Table Product
  2  DROP CONSTRAINT vd_fk;

Table altered.

SQL> ALTER Table CustomerOrderProduct
  2  DROP CONSTRAINT co_fk;

Table altered.

SQL> ALTER Table CustomerOrderProduct
  2  DROP CONSTRAINT oe_fk;

Table altered.

SQL> ALTER Table CustomerOrderProduct
  2  DROP CONSTRAINT pd_fk;

Table altered.

SQL> ALTER Table OrderItemLine
  2  DROP CONSTRAINT oi_fk;

Table altered.

SQL> ALTER Table OrderItemLine
  2  DROP CONSTRAINT pt_fk;

Table altered.
```

Figure 45 Dropping all foreign key from respective tables



```
SQL> DROP table Customer;

Table dropped.

SQL> DROP table Orderr;

Table dropped.

SQL> DROP table CustomerOrder;

Table dropped.

SQL> DROP table Vendor;

Table dropped.

SQL> DROP table Product;

Table dropped.

SQL> DROP table CustomerOrderProduct;

Table dropped.

SQL> DROP table OrderItemLine;

Table dropped.

SQL> _
```

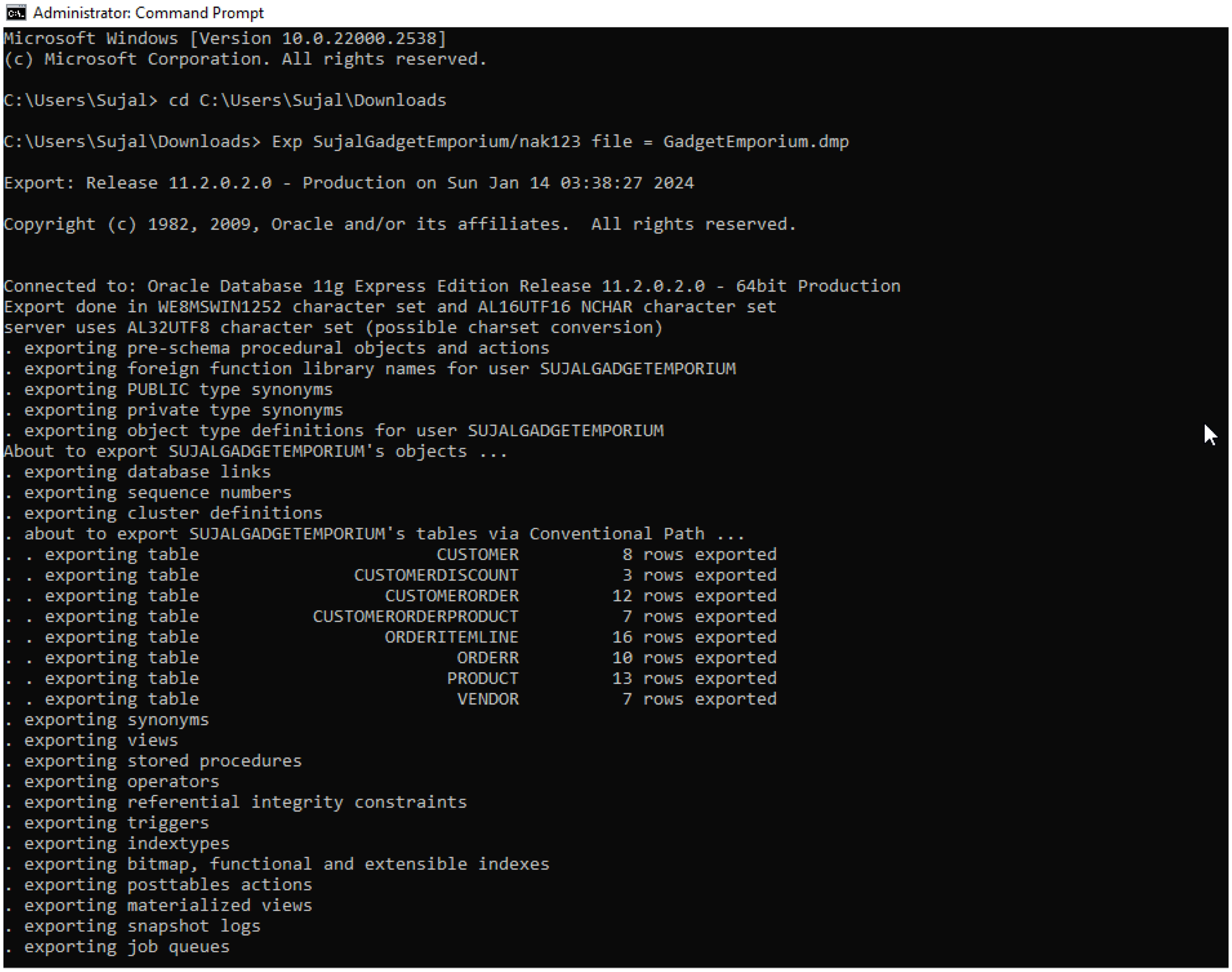
Figure 46 Dropping all tables



```
SQL> desc Customer;
ERROR:
ORA-04043: object Customer does not exist


SQL> desc Orderr;
ERROR:
ORA-04043: object Orderr does not exist


SQL> desc CustomerOrder;
ERROR:
ORA-04043: object CustomerOrder does not exist


SQL> desc Vendor;
ERROR:
ORA-04043: object Vendor does not exist


SQL> desc Product;
ERROR:
ORA-04043: object Product does not exist


SQL> desc CustomerOrderProduct;
ERROR:
ORA-04043: object CustomerOrderProduct does not exist


SQL> desc OrderItemLine;
ERROR:
ORA-04043: object OrderItemLine does not exist


SQL>
```

Figure 47 Checking if the tables has been dropped or no

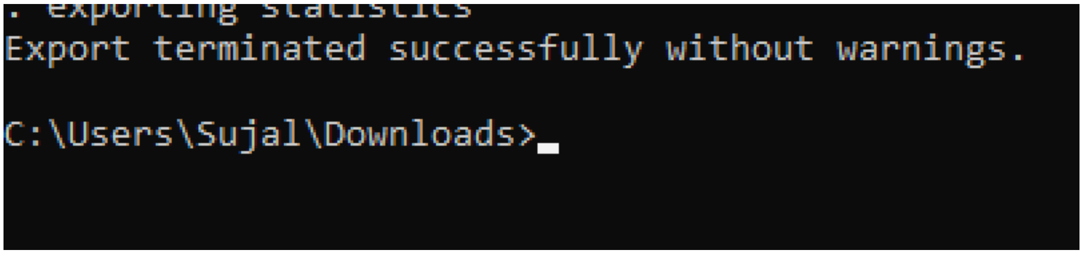
Creating a Dump File



Administrator: Command Prompt

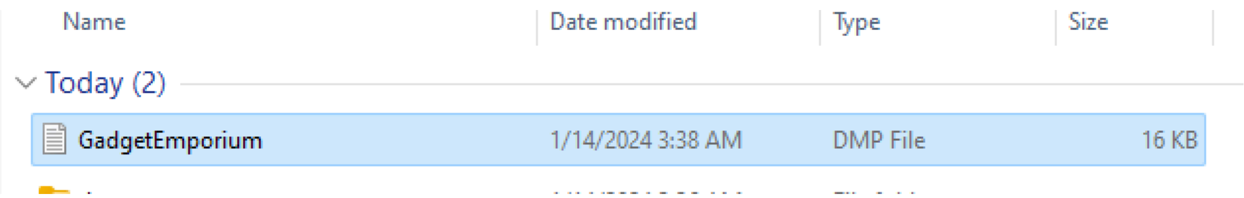
```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sujal> cd C:\Users\Sujal\Downloads

C:\Users\Sujal\Downloads> Exp SujalGadgetEmporium/nak123 file = GadgetEmporium.dmp

Export: Release 11.2.0.2.0 - Production on Sun Jan 14 03:38:27 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 SUJALGADGETEMPORIUM
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user SUJALGADGETEMPORIUM
About to export SUJALGADGETEMPORIUM's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export SUJALGADGETEMPORIUM's tables via Conventional Path ...
. . exporting table                       CUSTOMER          8 rows exported
. . exporting table               CUSTOMERDISCOUNT          3 rows exported
. . exporting table                  CUSTOMERORDER         12 rows exported
. . exporting table           CUSTOMERORDERPRODUCT          7 rows exported
. . exporting table                  ORDERITEMLINE         16 rows exported
. . exporting table                         ORDERR         10 rows exported
. . exporting table                        PRODUCT         13 rows exported
. . exporting table                         VENDOR          7 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
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
```

Figure 48 Writing Syntax For Creating dump file



```
. exporting statistics
Export terminated successfully without warnings.

C:\Users\Sujal\Downloads>_
```

Figure 49 Dump File Successfully Created



Name	Date modified	Type	Size
Today (2)			
GadgetEmporium	1/14/2024 3:38 AM	DMP File	16 KB

Figure 50 Image of Dump File being Created

11. Conclusion

After completion of the coursework, I have learnt many concepts of the Database Module such as Normalization, many SQL query. I have gained more knowledge about UNF, 1NF, 2NF, 3NF, how to use select, insert, create, alter, update, drop, queries and function, group by clause, order by clause, JOIN the table using (using operator and or operator), about the alias, concatenation operators. I have reviewed my progress each week to our lecturer and Tutor completion of coursework without their guidance wasn't possible.

Structing the report also have developed my ability in creating the documentation more beautiful. I have developed time management skill doing the coursework and communication skills were also developed. To complete the coursework I had to do a lot of research which also increase my research skills.

12. References

Raipurkar, A. & Deokate, G., 2012. Business Rules in DBMS. *International Journal of Advanced Research in Computer Science,* 3(3), p. 693.

JavaTpoint, 2023. *Entity in DBMS.* [Online]
Available at: https://www.javatpoint.com/entity-in-dbms
[Accessed 22 December 2023].

Rouse, M., 2023. *What Does Attribute Mean?.* [Online]
Available at: https://www.techopedia.com/definition/1164/attribute-database-systems#:~:text=In%20relational%20databases%2C%20attributes%20are,to%20uniquely%20identify%20each%20item.
[Accessed 23 December 2023].

Sugandhi, A., 2023. *Types of Attributes in DBMS.* [Online]
Available at: https://www.knowledgehut.com/blog/database/attributes-in-dbms
[Accessed 23 December 2023].

Gibbs, M., 2021. *What is an Entity in a Database?.* [Online]
Available at: https://study.com/academy/lesson/what-is-an-entity-in-a-database.html#:~:text=An%20entity%20is%20an%20object%20about%20which%20data%20is%20to,domain%20type%20is%20the%20key.
[Accessed 23 December 2023].

LucidChart, 2023. *What is an Entity Relationship Diagram (ERD)?.* [Online]
Available at: https://www.lucidchart.com/pages/er-diagrams
[Accessed 23 December 2023].

IBM , 2023. *Primary Keys.* [Online]
Available at: https://www.ibm.com/docs/en/iodg/11.3?topic=reference-primary-keys
[Accessed 23 December 2023].

w3schools, 2023. *SQL FOREIGN KEY Constraint.* [Online]
Available at: https://www.w3schools.com/sql/sql_foreignkey.asp
[Accessed 23 December 2023].

Peterson, R., 2023. *What is Normalization in DBMS (SQL)? 1NF, 2NF, 3NF Example.* [Online]
Available at: https://www.guru99.com/database-normalization.html
[Accessed 23 December 2023].

Gibbs, M., 2023. *Database Query: Definition & Tools.* [Online]
Available at: https://study.com/academy/lesson/database-query-definition-tools.html#:~:text=A%20database%20query%20is%20a%20request%20for%20data%2

0from%20a,tables%2C%20or%20even%20other%20queries.
[Accessed 23 December 2023].

SolutionGlobal, 2023. *Business Rules and How They Affect Database Design.* [Online]
Available at: https://www.soutron.com/blog/general/business-rules-database-design/
[Accessed 23 December 2023].

SOLARWIND, 2023. *What is a Database Query?.* [Online]
Available at: https://www.solarwinds.com/resources/it-glossary/database-query
[Accessed 23 December 2023].

DatabaseTown, 2023. *Why are Databases Important?.* [Online]
Available at: https://databasetown.com/why-are-databases-important/#:~:text=Databases%20provide%20a%20centralized%20location,an%20extended%20period%20of%20time.
[Accessed 26 December 2023].