LONDON
METROPOLITAN
UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: Sardul Ojha
London Met ID: 22067311
Assignment Submission Date: 15th January 2024
Word Count: 3762

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

Welcome to Gadget Emporium, the place where tech dreams come to life! Created by the visionary Mr. John, a passionate entrepreneur and electronics enthusiast, this online haven isn't just an e-commerce platform; it's a promise of an extraordinary shopping experience. Our curated store brings the world of electronic gadgets to your fingertips, whether you're a tech enthusiast or a business in search of cutting-edge solutions. With a wide range of products and a robust database system, we aim to offer a great shopping experience to our customers.

"Gadget Emporium" is not just an e-commerce platform; it's a promise to provide an outstanding shopping experience. Gadget Emporium is designed to meet your needs with a diverse and expansive selection of electronic devices. Our forte lies in the commitment to delivering an exceptional shopping experience for both private consumers and business organizations. With our strong database system crafted by our expert team, we keep track of important elements, including our valued customers, a variety of products, and ensure smooth order processing.

As you explore our online store, you'll discover a comprehensive system managing products, customers, orders, vendors, inventory, payments, and invoices. Join us at Gadget Emporium, the epitome of reliability, customer focus, and a great shopping experience. Here, innovation, quality, and convenience converge to redefine the way you shop for electronic devices. Welcome to a world where technology meets passion. Join us, and let's make your electronic dreams a reality!

1.2 Current Business Activities and Operations

The current business activities and operations are mentioned below:

- Gadget Emporium is an online e-commerce platform to sell electronic gadgets.
- It keeps track of products, their description, price, quantity and the vendor.

- The details of the customer are also stored. They are divided into three categories: Regular(R), Staff(S), and VIP(V) with 0, 5 and 10 percent discount respectively.
- It also monitors every orders. It keeps track of order date, quantity of product bought, total amount paid, customer details and payment.
- It also accepts different payment options like cash on delivery, cards or e-wallets.
- It also manages invoice for every transaction.

1.3 List of Business Rules

The business rules are as follows:

- A product is bought by many customers and a customer can place many orders.
- A vendor can supply one or more products. A product must have a single vendor.
- A customer must be categorized.
- Each order must have one payment option and a payment option can be in multiple order.
- An order must be associated with a single customer.
- An order can have multiple products and any product can be placed in multiple orders.
- A customer can exist without purchasing any products (with no orders).
- The order quantity must be associated with customer and product.
- Details of order and payment are associated with order and customer.
- Each customer category must have different discount rate.
- A payment must contain any one payment method.

1.4 Identification of Entities and Attributes

The following entities with their attributes, data type and description have been identified for the database :

Entities	Attributes	Data type	Description
Product	Product ID Product Name	Number VARCHAR	Uniquely identifies every product Name of the product

	Description	VARCHAR	Description of the product
	Product Category	VARCHAR	Category of the product
	Price	Number	Cost price of the product
	Stock Level	Number	Number of product present in stock
	Vendor ID	Number	Id of the product’s vendor
Vendor	Vendor ID	Number	Uniquely identifies every vendor
	Vendor Name	VARCHAR	Name of the vendor
	Vendor Address	VARCHAR	Address of the vendor
Customer	Customer ID	Number	Uniquely identifies every customer
	Customer Name	VARCHAR	Name of the customer
	Category	VARCHAR	Customer’s Category (R, S or V)
	Discount Rate	Number	Discount received by customer
	Address	VARCHAR	Address of the customer
	Phone Number	Number	Customer’s contact number
Order	Order ID	Number	Uniquely identifies every order
	Customer ID	Number	Id of order’s customer
	Product ID	Number	Id of product in the order
	Product Quantity	Number	Quantity of the product
	Order Date	Date	Date of order
	Total Order Amount	Number	Total amount of the order (without discount)
	Payment ID	Number	Id of payment
	Payment Option	VARCHAR	Type of payment
	Payment Option Details	Number	Details of payment (card no, phone number)

Table 1: Table of Entity and Attribute Identification

2. Initial ERD

2.1 List of Entities and Attributes

Entity: Product

Attributes: Product ID, Product Name, Description, Product Category, Price, Stock Level, Vendor ID

Entity: Customer

Attributes: Customer ID, Customer Name, Category, Discount Rate, Address, Phone Number

Entity: Order

Attributes: Order ID, Customer ID, Product ID, Product Quantity, Order Date, Total Order Amount, Payment ID, Payment Option, Payment Option Details

Entity: Vendor

Attributes: Vendor ID, Vendor Name, Vendor Address

2.2 Identification and representation of Primary Keys and Foreign Keys

a. Product Entity

- Product ID (Primary Key)
- Vendor ID (Foreign Key)

b. Vendor Entity

- Vendor ID (Primary Key)

c. Customer Entity

- Customer ID (Primary Key)

d. Order Entity

- Order ID (Primary Key)
- CustomerID (Foreign Key)
- ProductID (Foreign Key)

2.3 Entity Relationship Diagram

A simple ERD diagram is created from our business rules and identified entities. It helps us to preview the design of database before creating it. The initial ERD for gadget emporium:

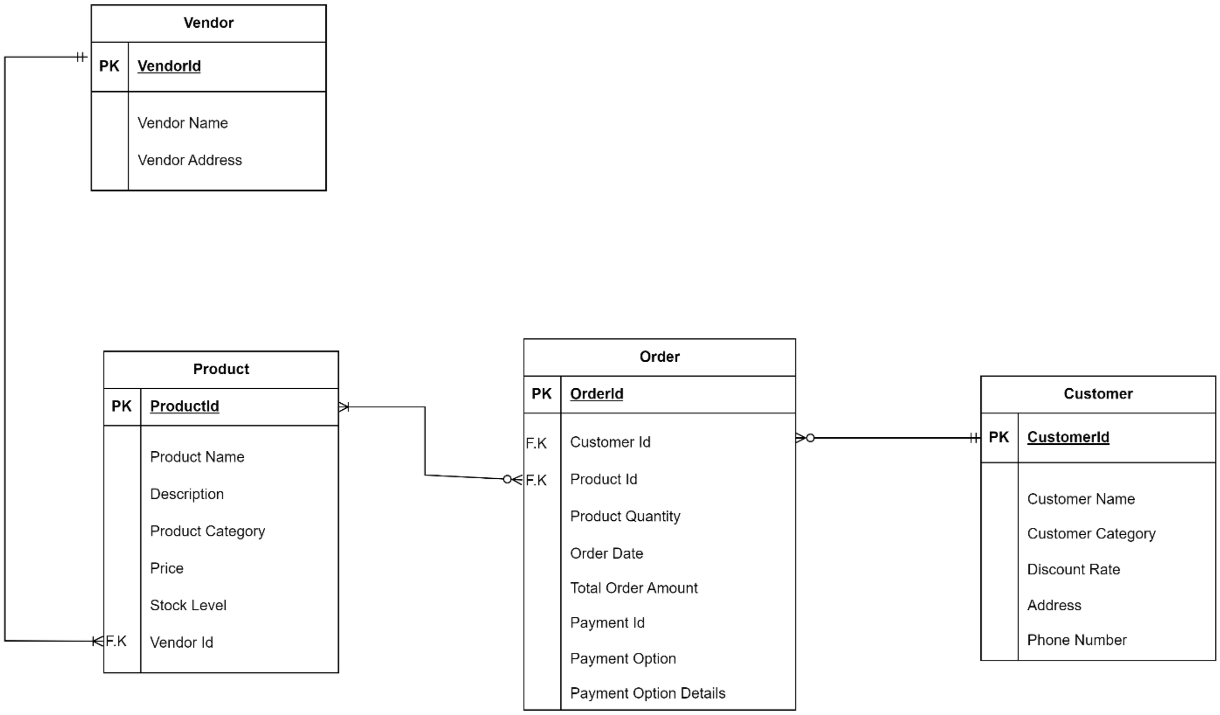


Figure 1: Initial ERD Diagram

3. Normalization

The normalization process of our database is as follows:

Unnormalized Form (UNF)

In this form, no rules of normalization are applied. All the attributes are represented in a single table and repeating data and repeating groups are identified. A single primary key is also identified.

The Table in UNF is:

Product (Product ID (P.K), Product Name, Description, Product Category, Price, Stock Level, Vendor ID, Vendor Name, {Customer ID, Customer Name, Customer Category, Discount, Address, Phone Number, {Order ID, Order Date, Order Quantity, Total Order Payment, Payment ID, Payment Type, Payment Option Detail })

In our case, a single table named Product is created where Product ID is the primary key. Repeating groups are separated using curly braces "{}". Here,

- Product ID, Product Name, Description, Product Category, Price, Stock Level, Vendor ID, Vendor Name, Vendor Address : **repeating data**
- Customer ID, Customer Name, Customer Category, Discount, Address, Phone Number : **repeating groups**
- Order ID, Order Date, Order Quantity, Total Order Payment, Payment ID, Payment Type, Payment Option Detail : **repeating groups of repeating groups**

First Normal Form (1NF)

For the first normal form, repeating data and repeating groups are separated into different tables. Each table has a primary key and foreign key is present in repeating group tables to create a relationship between them.

The Tables in 1NF are:

Product -1 (<u>ProductID</u> (P.K), Product Name, Description, Product Category, Price, Stock Level, Vendor ID, Vendor Name, Vendor Address)

Customer-1 (Product ID* (F.K), <u>Customer ID</u> (P.K), Customer Name, Customer Category, Discount, Address, Phone Number)

Order-1 (ProductID* (F.K), CustomerID* (F.K), <u>OrderID</u> (P.K), Order Date, Order Quantity, Total Order Payment, Payment ID, Payment Type, Payment Option Detail)

First, product-1 table is created for the repeating data. Here product Id is the primary key. Then customer-1 table with customer Id as primary key is created for the repeating group. Since it is a repeating group, product id is present as foreign key to create a relationship between customer and product. Similarly, order-1 table consists of two foreign keys (product id and customer id) as it is a repeating group inside a repeating group. Here, order id is the primary key as it uniquely identifies each order.

Second Normal Form (2NF)

A table is said to be in second normal form if it is in the first normal form and does not contain any partial dependency. If a table in 1NF consists of only one key attribute, then it is automatically converted to 2NF. For the table with 2 or more keys, dependencies are checked with the combination of keys. If there are more than one attributes present, then a table is formed. Here,

Product -1 table has only one key attribute, so it remains the same in second normal form. So,

Product -2 (ProductID (P.K), Product Name, Description, Product Category, Price, Stock Level, Vendor ID, Vendor Name, Vendor Address)

For Customer-1 Table (Two key columns), checking for partial dependencies for (2^2-1) = 3 combinations :

Product ID --> X

Customer ID --> Customer Name, Customer Category, Discount, Address, Phone Number

ProductID, Customer ID --> X

Similarly, For Order-1 Table (Three key columns), checking for partial dependencies for (2^3-1) = 7 combinations :

Customer ID --> X

Product ID --> X

Order ID --> X

Customer ID, Order ID --> Order Date, Total Order Payment, Payment ID, Payment Type, Payment Option Details

Product ID, Customer ID --> X

Product ID, Order ID --> Order Quantity

Customer ID, Product ID, Order ID --> X

The Tables in 2NF are:

Product -2 (ProductID (P.K), Product Name, Description, Product Category, Price, Stock Level, Vendor ID, Vendor Name, Vendor Address)

Customer-2 (Customer ID (P.K), Customer Name, Customer Category, Discount, Address, Phone Number)

Order-2 (Order ID (P.K), Order Date, Total Order Payment, Customer ID*, Payment ID, Payment Type, Payment Option Details)

Prod_cus_details-2 (Product ID*, Customer ID*)

Prod_order_details -2 (Product ID*, Order ID*, Order Quantity)

Prod_cus_order_details-2(Customer ID*, Product ID*, Order ID*)

Third Normal Form (3NF)

In third normal form, dependency between the non-key attributes are identified and they are removed to a new table. A primary key is decided for the new table and that key becomes a foreign key in the original table. This is done to remove data redundancy. Now,

Checking for transitive dependency for product-2 table, we get,

Product Name --> X

Description --> X

Product Category --> X

Price --> X

Stock Level --> X

Vendor ID --> Vendor Name, Vendor Address

Similarly, checking for transitive dependency for customer-2 table, we get,

Customer Name --> X

Customer Category --> Discount

Address --> X

Phone Number --> X

Also, checking for transitive dependency for order-2 table, we get,

Order Date --> X

Total Order Payment -- X

Payment ID --> Payment Type, Payment Option Details

The remaining tables Prod_cus_details-2, Order_quantity_details-2, Cus_order_detail-2 and Prod_cus_order_details-2 tables do not have transitive dependency as they have no or only one attribute, so they remain the same.

Here, the transitive dependency table is created for vendor, customer category and payment. A unique primary key is decided for each table and that becomes a foreign key in the parent table. So,

The Tables in 3NF are:

Product -3 (ProductID (P.K), Product Name, Description, Product Category, Price, Stock Level, Vendor ID*)

Vendor_details-3 (Vendor ID (P.K), Vendor Name, Vendor address)

Customer-3 (Customer ID (P.K), Customer Name, Customer Category*, Address, Phone Number)

Customer_category-3 (Customer Category (P.K), Discount)

Order-3 (Order ID (P.K), Order Date, Customer ID*, Total Order Payment, Payment ID*)

Payment-3 (Payment ID (P.K), Payment Type, Payment Option Details)

Prod_cus_details-3 (Product ID*, Customer ID*)

Cus_order_details-3 (Order ID*)

Prod_cus_order_details-3 (Customer ID*, Product ID*, Order ID*)

After 3NF, the tables are named properly for the creation of database. Also, the unwanted tables like Prod_cus_details-3, Prod_cus_order_details-3 are removed because we do not need the relationship between product and customer, and it does not have use in our database.

Hence, the final database tables are:

⇒ Product (ProductID (P.K), Product Name, Description, Product Category, Price, Stock Level, Vendor ID*)

⇒ Vendor_details (Vendor ID (P.K), Vendor Name, Vendor address)

⇒ Customer (Customer ID (P.K), Customer Name, Customer Category*, Address, Phone Number)

⇒ Customer Category Details (Customer Category (P.K), Discount)

⇒ Order (Order ID (P.K), Order Date, Total Order Payment, Customer ID*, Payment ID*)

⇒ Payment Details (Payment ID (P.K), Payment Type, Payment Option Details)

⇒ Order Quantity Details (Product ID*, Order ID*, Order Quantity)

4. Final ERD

After declaration of tables from normalization, a final ERD is created to show the relationship between the normalized tables.

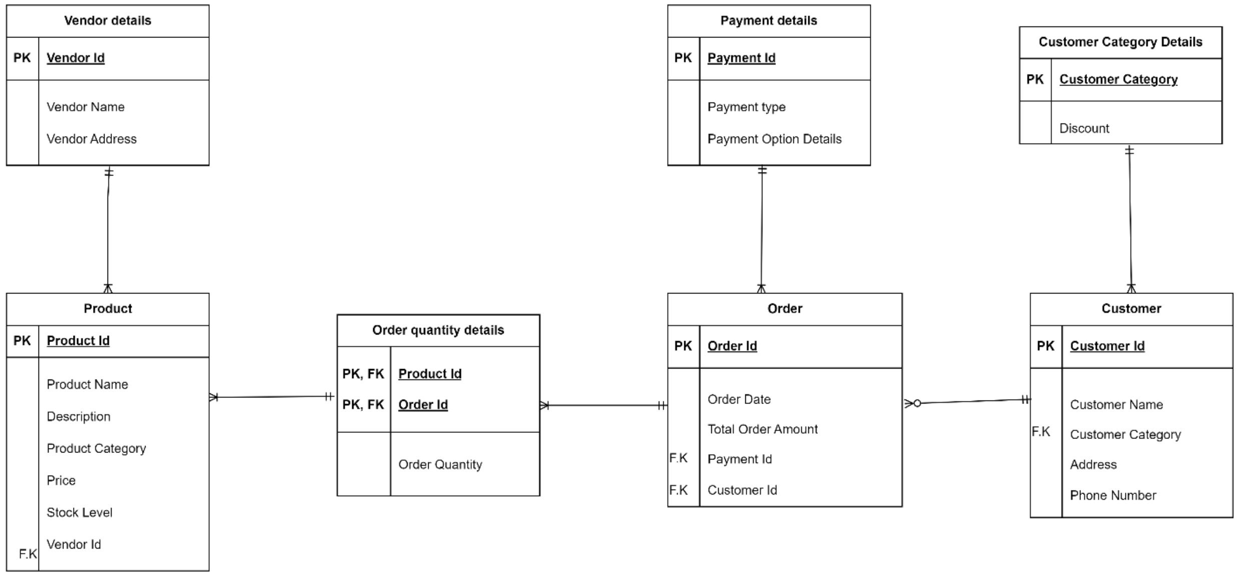


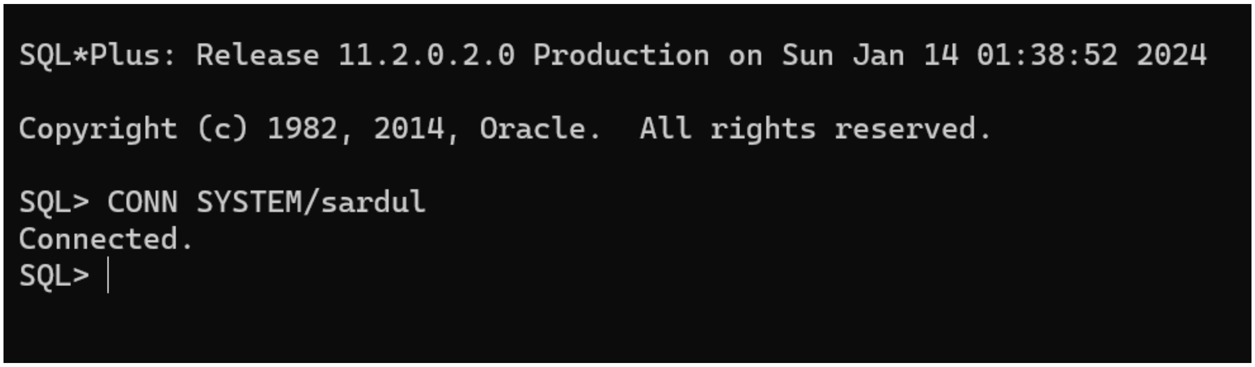
Figure 2: Final ERD Diagram

5. Implementation

The step-by-step creation of user, tables and insertion of values are shown below with the help of respective snapshots:

5.1 Creating User

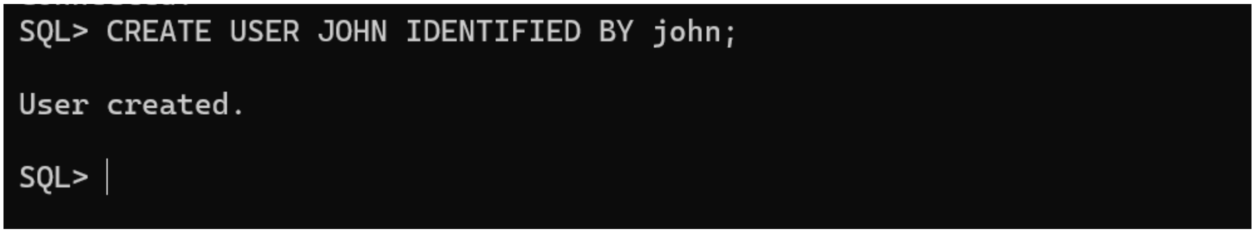
1. Connecting to the system.



```
SQL*Plus: Release 11.2.0.2.0 Production on Sun Jan 14 01:38:52 2024

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

SQL> CONN SYSTEM/sardul
Connected.
SQL> 
```

Figure 3: Screenshot of connecting to the system

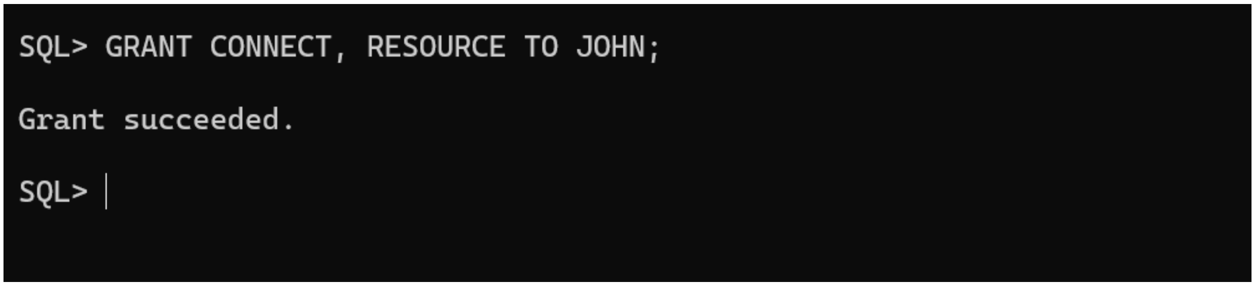
2. Creating user named "John" with john as password



```
SQL> CREATE USER JOHN IDENTIFIED BY john;

User created.

SQL> 
```

Figure 4: Screenshot of creating a user

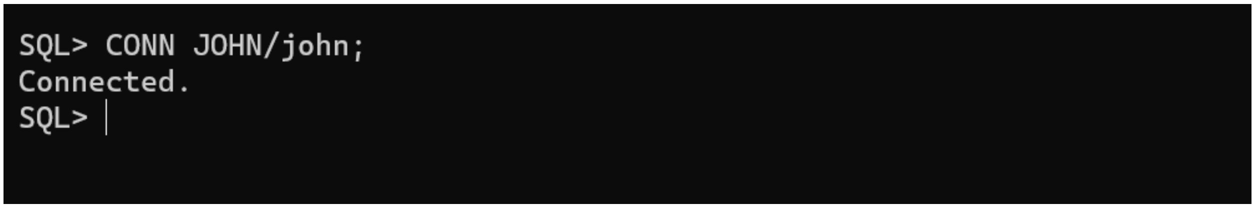
3. Granting access to user "John"



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

Grant succeeded.

SQL> 
```

Figure 5: Screenshot of granting access to John

4. Connection to "John" user.

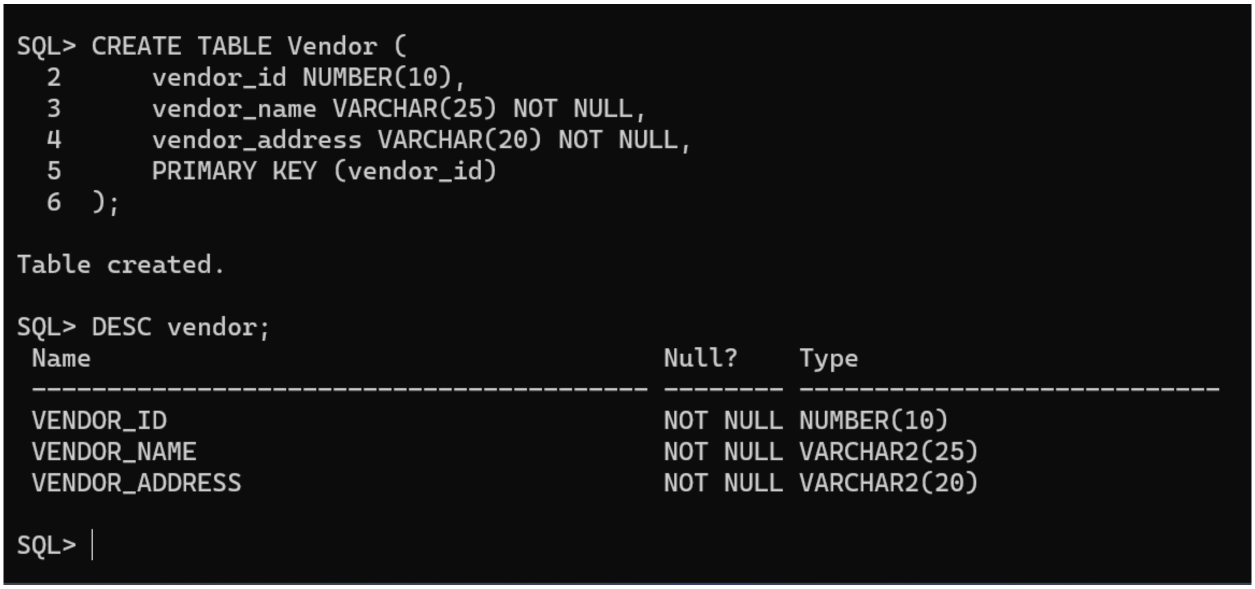


```
SQL> CONN JOHN/john;
Connected.
SQL>
```

Figure 6: Screenshot of connecting to John

5.2 Creating Tables

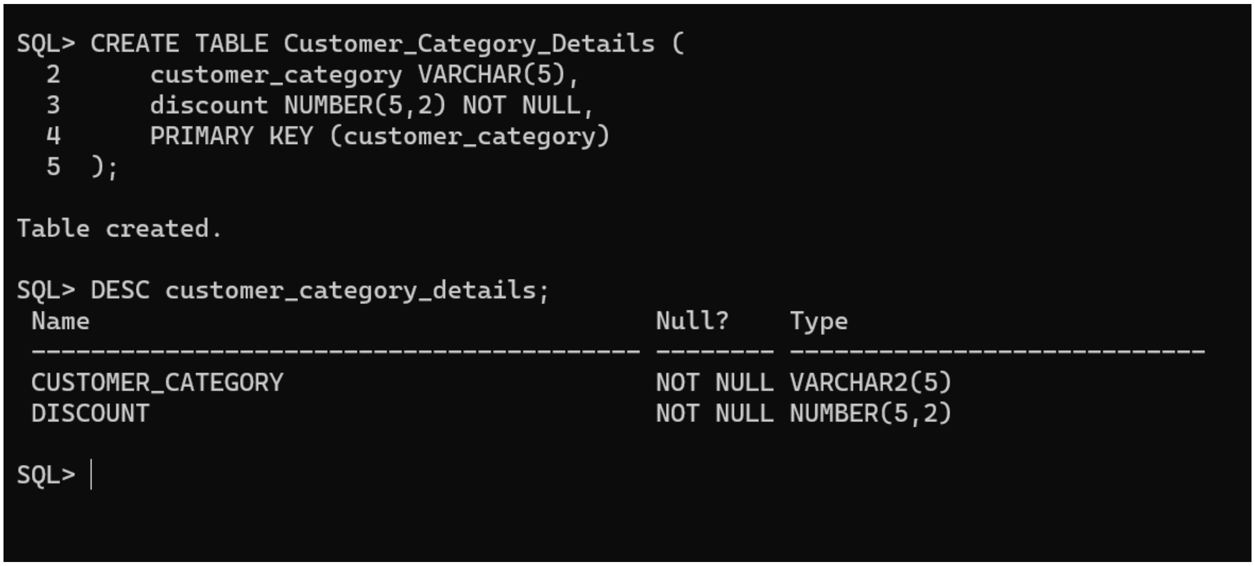
5. Creating vendor table.



```
SQL> CREATE TABLE Vendor (
        vendor_id NUMBER(10),
        vendor_name VARCHAR(25) NOT NULL,
        vendor_address VARCHAR(20) NOT NULL,
        PRIMARY KEY (vendor_id)
    );

Table created.

SQL> DESC vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VENDOR_ID                                 NOT NULL NUMBER(10)
 VENDOR_NAME                               NOT NULL VARCHAR2(25)
 VENDOR_ADDRESS                            NOT NULL VARCHAR2(20)

SQL>
```

Figure 7: Screenshot of vendor table created and described

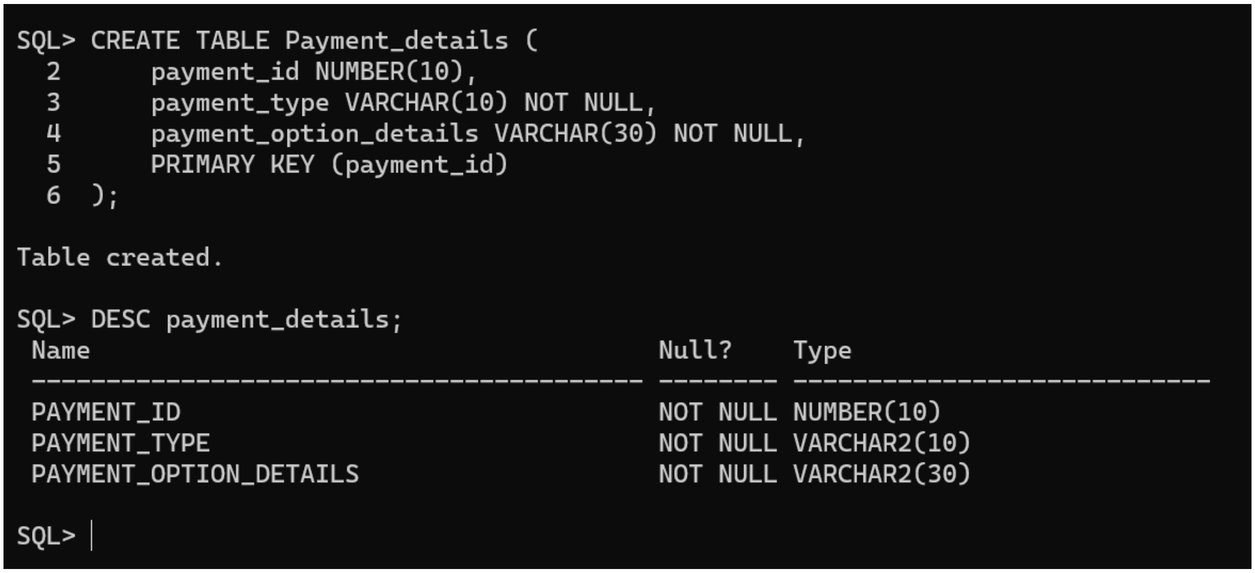
6. Creating customer_category_details table.



```
SQL> CREATE TABLE Customer_Category_Details (
         customer_category VARCHAR(5),
         discount NUMBER(5,2) NOT NULL,
         PRIMARY KEY (customer_category)
     );

Table created.

SQL> DESC customer_category_details;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_CATEGORY                         NOT NULL VARCHAR2(5)
 DISCOUNT                                  NOT NULL NUMBER(5,2)

SQL>
```

Figure 8: Screenshot of customer_category_details table created and described

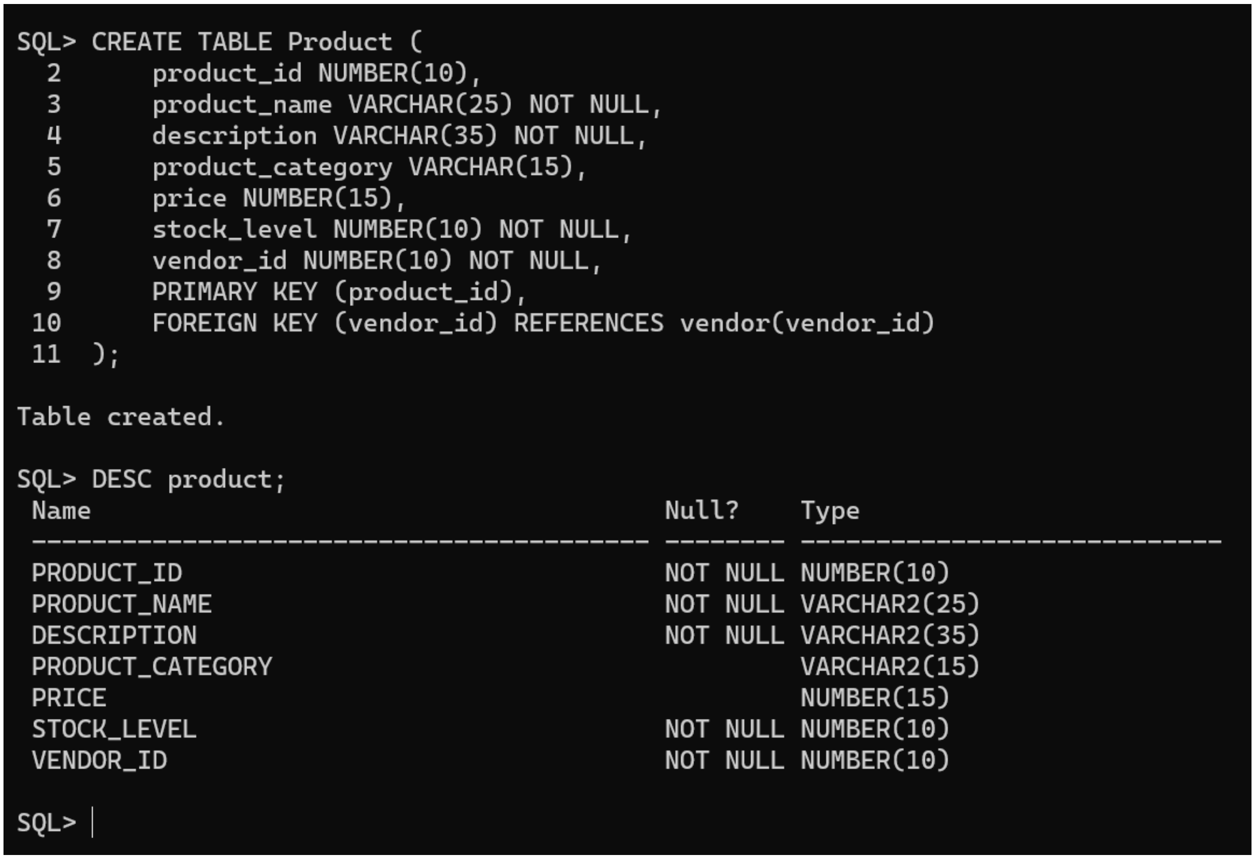
7. Creating payment_details table.



```
SQL> CREATE TABLE Payment_details (
         payment_id NUMBER(10),
         payment_type VARCHAR(10) NOT NULL,
         payment_option_details VARCHAR(30) NOT NULL,
         PRIMARY KEY (payment_id)
     );

Table created.

SQL> DESC payment_details;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PAYMENT_ID                                NOT NULL NUMBER(10)
 PAYMENT_TYPE                              NOT NULL VARCHAR2(10)
 PAYMENT_OPTION_DETAILS                    NOT NULL VARCHAR2(30)

SQL>
```

Figure 9: Screenshot of payment_details table created and described

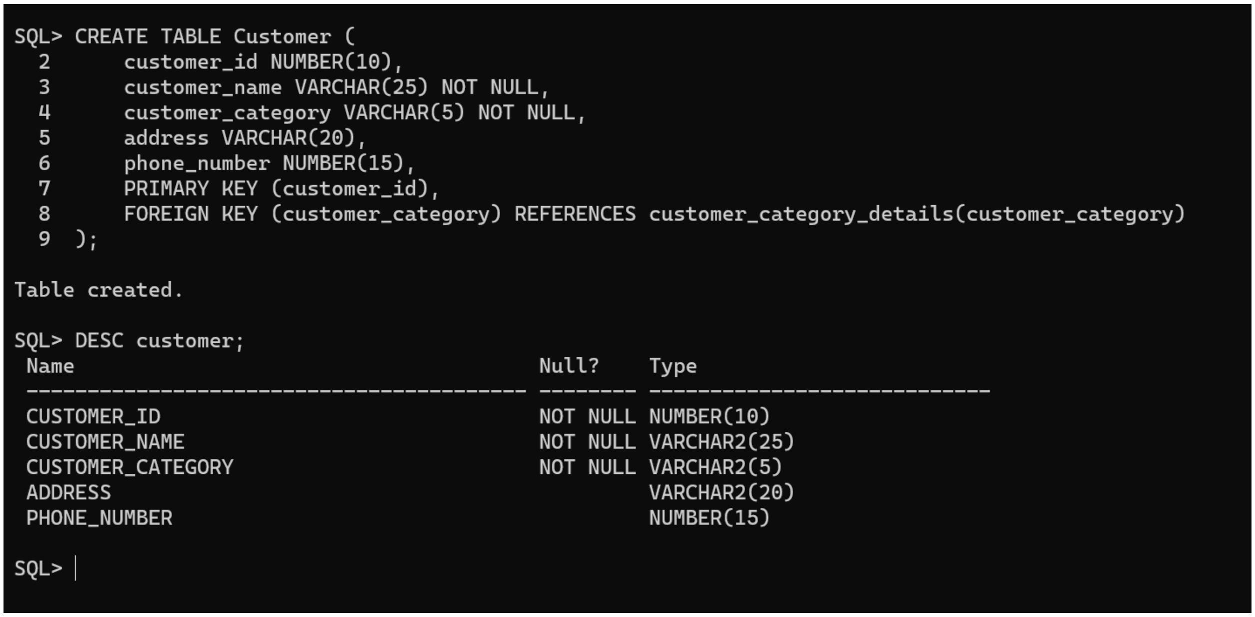
8. Creating product table.



```
SQL> CREATE TABLE Product (
        product_id NUMBER(10),
        product_name VARCHAR(25) NOT NULL,
        description VARCHAR(35) NOT NULL,
        product_category VARCHAR(15),
        price NUMBER(15),
        stock_level NUMBER(10) NOT NULL,
        vendor_id NUMBER(10) NOT NULL,
        PRIMARY KEY (product_id),
        FOREIGN KEY (vendor_id) REFERENCES vendor(vendor_id)
  );

Table created.

SQL> DESC product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCT_ID                                NOT NULL NUMBER(10)
 PRODUCT_NAME                              NOT NULL VARCHAR2(25)
 DESCRIPTION                               NOT NULL VARCHAR2(35)
 PRODUCT_CATEGORY                                   VARCHAR2(15)
 PRICE                                              NUMBER(15)
 STOCK_LEVEL                               NOT NULL NUMBER(10)
 VENDOR_ID                                 NOT NULL NUMBER(10)

SQL>
```

Figure 10: Screenshot of product table created and described

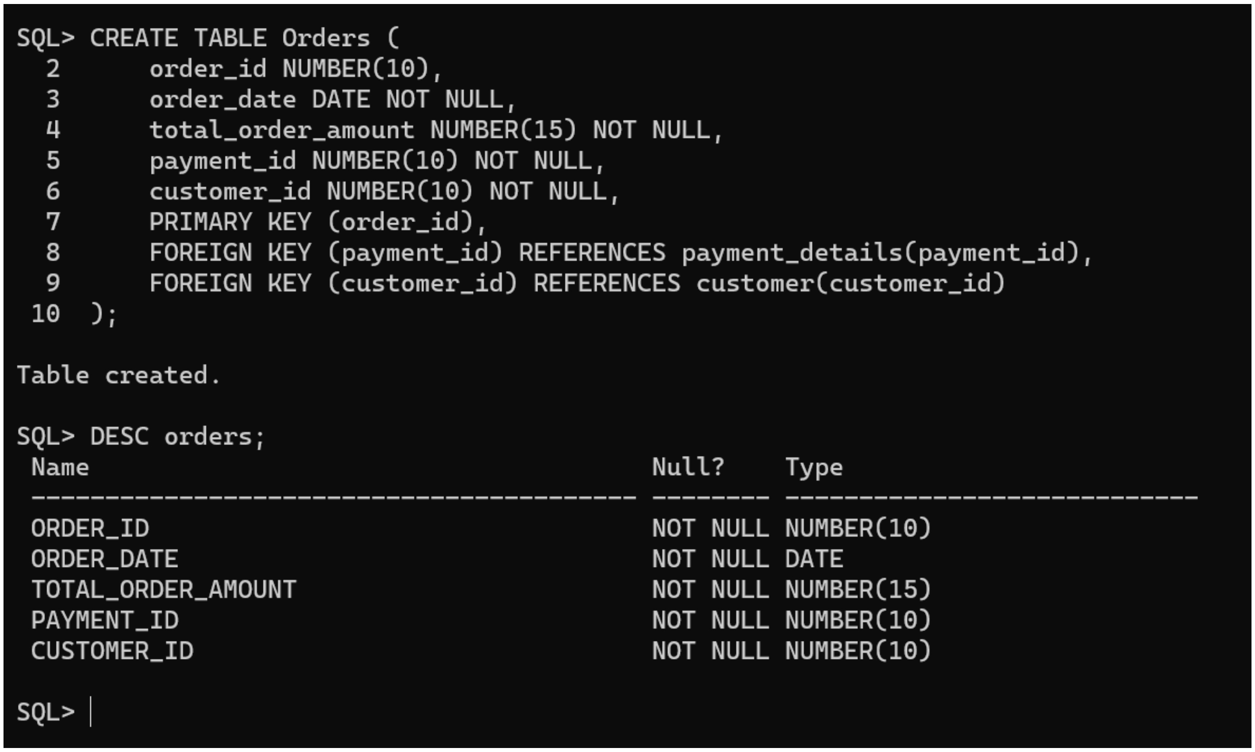
9. Creating customer table.



```
SQL> CREATE TABLE Customer (
         customer_id NUMBER(10),
         customer_name VARCHAR(25) NOT NULL,
         customer_category VARCHAR(5) NOT NULL,
         address VARCHAR(20),
         phone_number NUMBER(15),
         PRIMARY KEY (customer_id),
         FOREIGN KEY (customer_category) REFERENCES customer_category_details(customer_category)
     );

Table created.

SQL> DESC customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL NUMBER(10)
 CUSTOMER_NAME                             NOT NULL VARCHAR2(25)
 CUSTOMER_CATEGORY                         NOT NULL VARCHAR2(5)
 ADDRESS                                            VARCHAR2(20)
 PHONE_NUMBER                                       NUMBER(15)

SQL>
```

Figure 11: Screenshot of customer table created and described

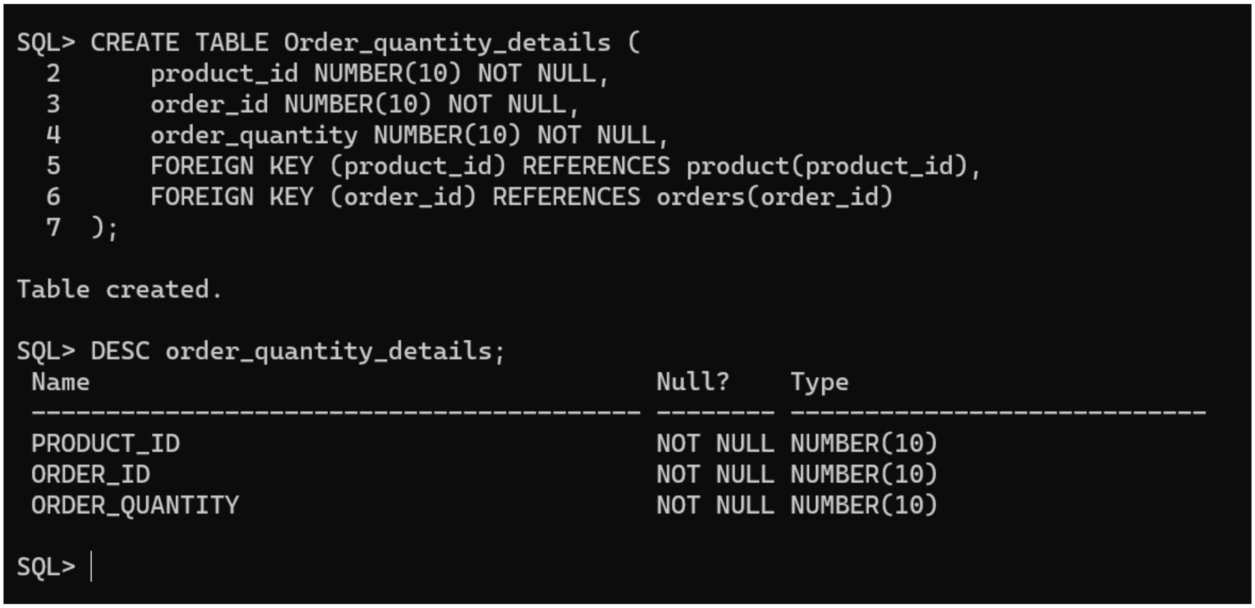
10. Creating orders table.



```
SQL> CREATE TABLE Orders (
         order_id NUMBER(10),
         order_date DATE NOT NULL,
         total_order_amount NUMBER(15) NOT NULL,
         payment_id NUMBER(10) NOT NULL,
         customer_id NUMBER(10) NOT NULL,
         PRIMARY KEY (order_id),
         FOREIGN KEY (payment_id) REFERENCES payment_details(payment_id),
         FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
     );

Table created.

SQL> DESC orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL NUMBER(10)
 ORDER_DATE                                NOT NULL DATE
 TOTAL_ORDER_AMOUNT                        NOT NULL NUMBER(15)
 PAYMENT_ID                                NOT NULL NUMBER(10)
 CUSTOMER_ID                               NOT NULL NUMBER(10)

SQL>
```

Figure 12: Screenshot of orders table created and described

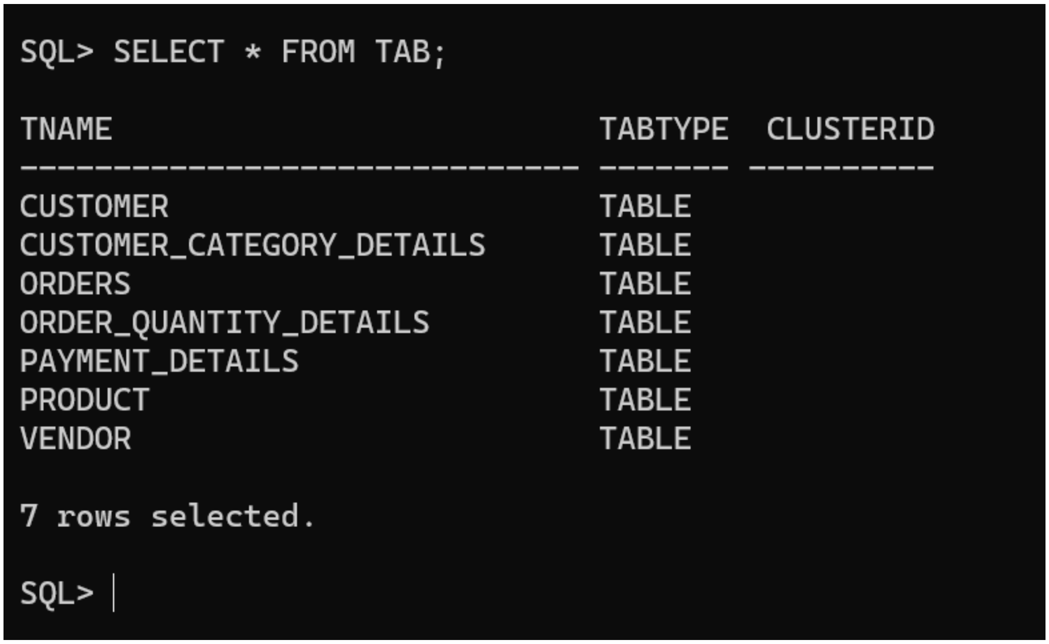
11. Creating order_quantity_details table.



```
SQL> CREATE TABLE Order_quantity_details (
        product_id NUMBER(10) NOT NULL,
        order_id NUMBER(10) NOT NULL,
        order_quantity NUMBER(10) NOT NULL,
        FOREIGN KEY (product_id) REFERENCES product(product_id),
        FOREIGN KEY (order_id) REFERENCES orders(order_id)
     );

Table created.

SQL> DESC order_quantity_details;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCT_ID                                NOT NULL NUMBER(10)
 ORDER_ID                                  NOT NULL NUMBER(10)
 ORDER_QUANTITY                            NOT NULL NUMBER(10)

SQL>
```

Figure 13: Screenshot of_order_quantity_details table created and described

12. Checking if all the tables are present.

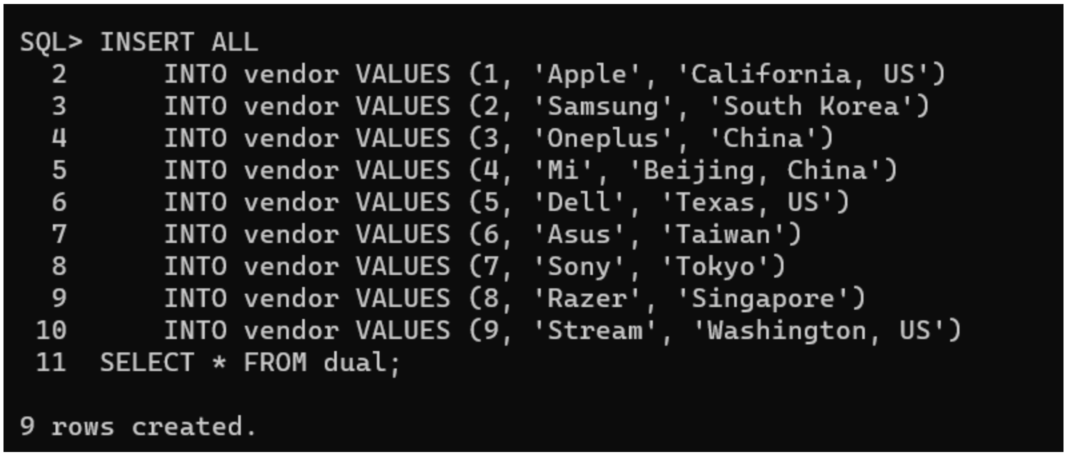


```
SQL> SELECT * FROM TAB;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
CUSTOMER                       TABLE
CUSTOMER_CATEGORY_DETAILS      TABLE
ORDERS                         TABLE
ORDER_QUANTITY_DETAILS         TABLE
PAYMENT_DETAILS                TABLE
PRODUCT                        TABLE
VENDOR                         TABLE

7 rows selected.

SQL>
```

Figure 14: Screenshot of displaying all tables

5.2 Inserting Values

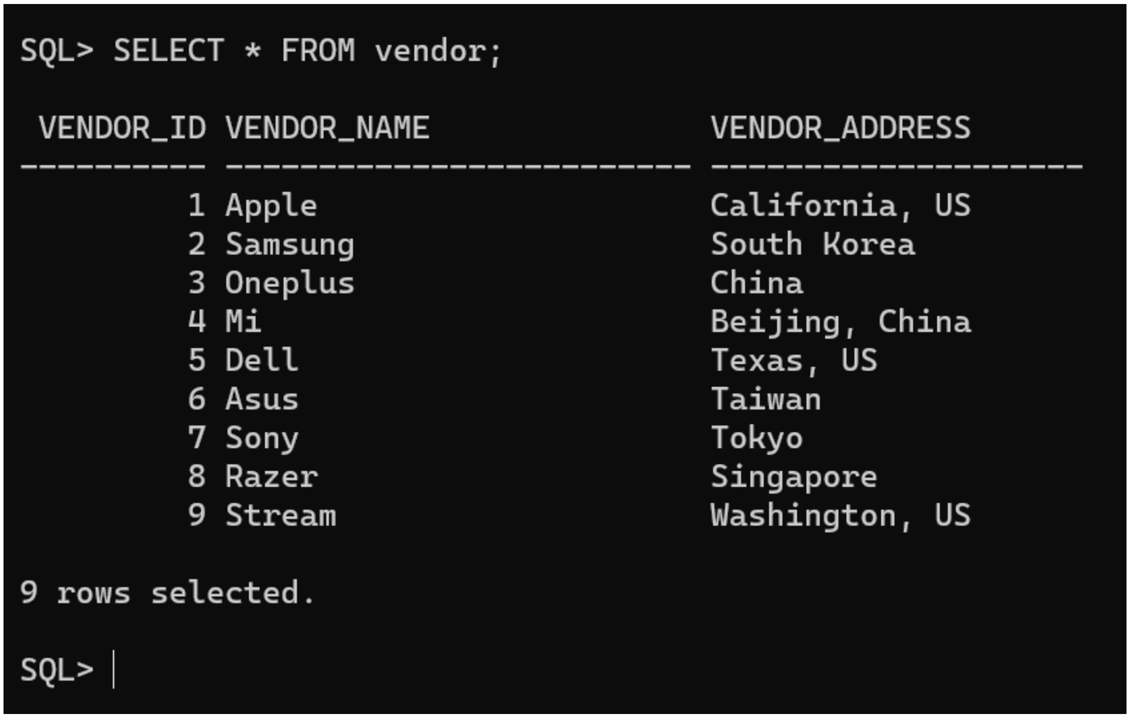
13. Inserting values in vendor table.



```
SQL> INSERT ALL
         INTO vendor VALUES (1, 'Apple', 'California, US')
         INTO vendor VALUES (2, 'Samsung', 'South Korea')
         INTO vendor VALUES (3, 'Oneplus', 'China')
         INTO vendor VALUES (4, 'Mi', 'Beijing, China')
         INTO vendor VALUES (5, 'Dell', 'Texas, US')
         INTO vendor VALUES (6, 'Asus', 'Taiwan')
         INTO vendor VALUES (7, 'Sony', 'Tokyo')
         INTO vendor VALUES (8, 'Razer', 'Singapore')
         INTO vendor VALUES (9, 'Stream', 'Washington, US')
     SELECT * FROM dual;

9 rows created.
```

Figure 15: Screenshot of inserting values in Vendor table

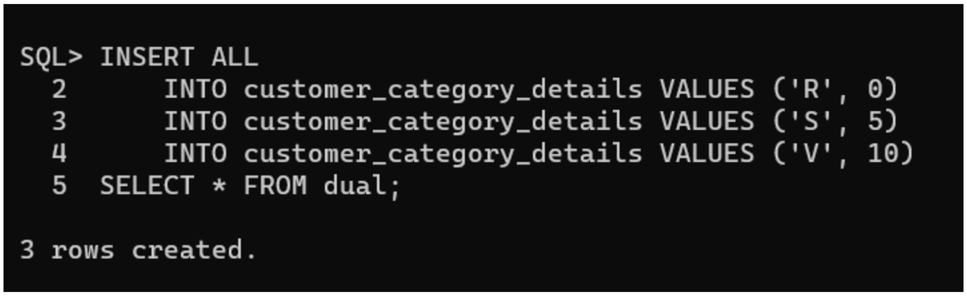
14. Displaying the values inserted in vendor table.



```
SQL> SELECT * FROM vendor;

 VENDOR_ID VENDOR_NAME                 VENDOR_ADDRESS
---------- ------------------------- --------------------
         1 Apple                     California, US
         2 Samsung                   South Korea
         3 Oneplus                   China
         4 Mi                        Beijing, China
         5 Dell                      Texas, US
         6 Asus                      Taiwan
         7 Sony                      Tokyo
         8 Razer                     Singapore
         9 Stream                    Washington, US

9 rows selected.

SQL>
```

Figure 16: Screenshot of displaying values of vendor table

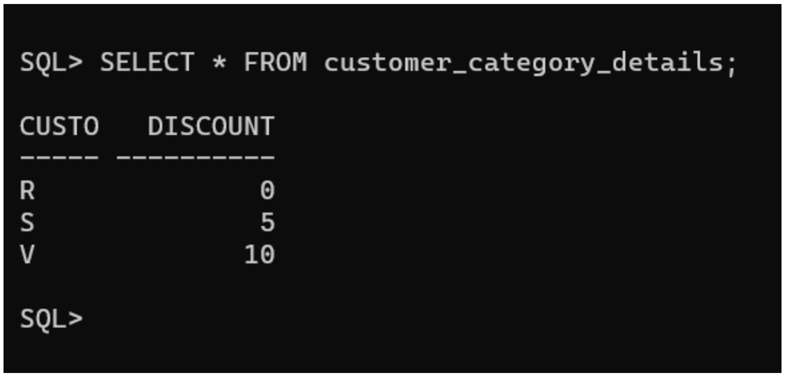
15. Inserting the values in customer_category_details table.



```
SQL> INSERT ALL
         INTO customer_category_details VALUES ('R', 0)
         INTO customer_category_details VALUES ('S', 5)
         INTO customer_category_details VALUES ('V', 10)
     SELECT * FROM dual;

3 rows created.
```

Figure 17: Screenshot of inserting values in customer_category_details table

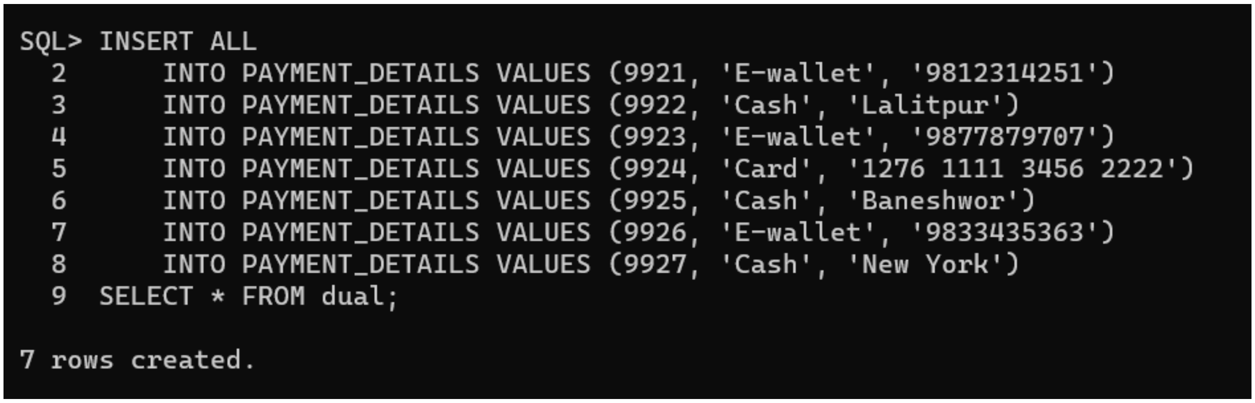
16. Displaying the inserted values in customer_category_details table.



```
SQL> SELECT * FROM customer_category_details;

CUSTO   DISCOUNT
----- ----------
R              0
S              5
V             10

SQL>
```

Figure 18: Screenshot of displaying values of customer_category_details table

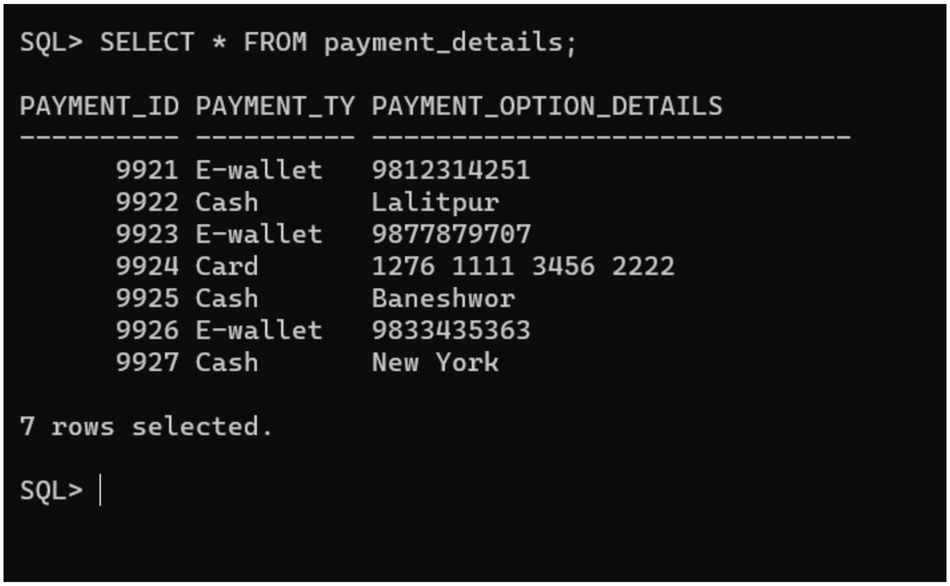
17. Inserting the values in payment_details table.



```
SQL> INSERT ALL
         INTO PAYMENT_DETAILS VALUES (9921, 'E-wallet', '9812314251')
         INTO PAYMENT_DETAILS VALUES (9922, 'Cash', 'Lalitpur')
         INTO PAYMENT_DETAILS VALUES (9923, 'E-wallet', '9877879707')
         INTO PAYMENT_DETAILS VALUES (9924, 'Card', '1276 1111 3456 2222')
         INTO PAYMENT_DETAILS VALUES (9925, 'Cash', 'Baneshwor')
         INTO PAYMENT_DETAILS VALUES (9926, 'E-wallet', '9833435363')
         INTO PAYMENT_DETAILS VALUES (9927, 'Cash', 'New York')
     SELECT * FROM dual;

7 rows created.
```

Figure 19: Screenshot of inserting values in payment_details table

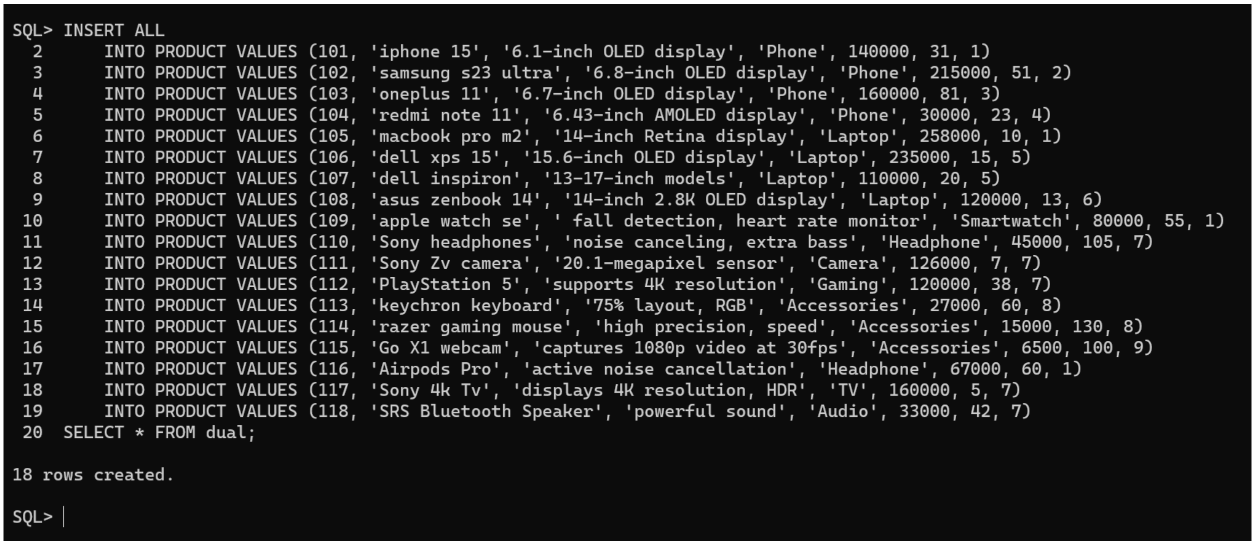
18. Displaying the values inserted in payment_details table.



```
SQL> SELECT * FROM payment_details;

PAYMENT_ID PAYMENT_TY PAYMENT_OPTION_DETAILS
---------- ---------- ------------------------------
      9921 E-wallet   9812314251
      9922 Cash       Lalitpur
      9923 E-wallet   9877879707
      9924 Card       1276 1111 3456 2222
      9925 Cash       Baneshwor
      9926 E-wallet   9833435363
      9927 Cash       New York

7 rows selected.

SQL>
```

Figure 20: Screenshot of displaying values of payment_details table

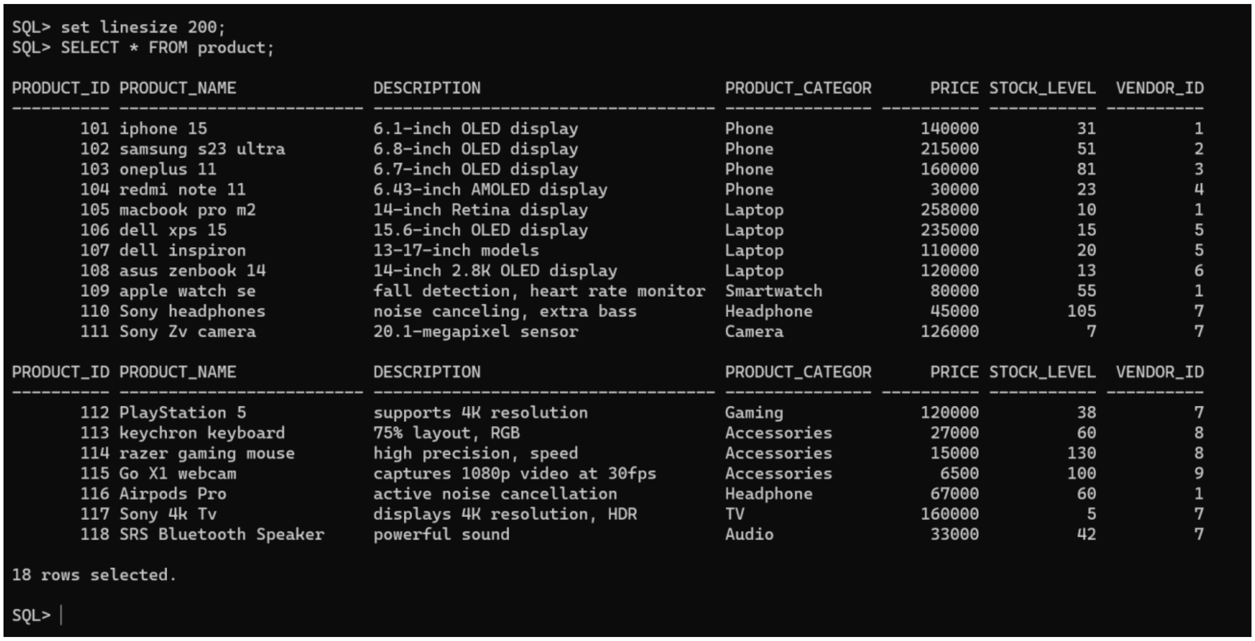
19. Inserting the values in product table.



```
SQL> INSERT ALL
        INTO PRODUCT VALUES (101, 'iphone 15', '6.1-inch OLED display', 'Phone', 140000, 31, 1)
        INTO PRODUCT VALUES (102, 'samsung s23 ultra', '6.8-inch OLED display', 'Phone', 215000, 51, 2)
        INTO PRODUCT VALUES (103, 'oneplus 11', '6.7-inch OLED display', 'Phone', 160000, 81, 3)
        INTO PRODUCT VALUES (104, 'redmi note 11', '6.43-inch AMOLED display', 'Phone', 30000, 23, 4)
        INTO PRODUCT VALUES (105, 'macbook pro m2', '14-inch Retina display', 'Laptop', 258000, 10, 1)
        INTO PRODUCT VALUES (106, 'dell xps 15', '15.6-inch OLED display', 'Laptop', 235000, 15, 5)
        INTO PRODUCT VALUES (107, 'dell inspiron', '13-17-inch models', 'Laptop', 110000, 20, 5)
        INTO PRODUCT VALUES (108, 'asus zenbook 14', '14-inch 2.8K OLED display', 'Laptop', 120000, 13, 6)
        INTO PRODUCT VALUES (109, 'apple watch se', ' fall detection, heart rate monitor', 'Smartwatch', 80000, 55, 1)
        INTO PRODUCT VALUES (110, 'Sony headphones', 'noise canceling, extra bass', 'Headphone', 45000, 105, 7)
        INTO PRODUCT VALUES (111, 'Sony Zv camera', '20.1-megapixel sensor', 'Camera', 126000, 7, 7)
        INTO PRODUCT VALUES (112, 'PlayStation 5', 'supports 4K resolution', 'Gaming', 120000, 38, 7)
        INTO PRODUCT VALUES (113, 'keychron keyboard', '75% layout, RGB', 'Accessories', 27000, 60, 8)
        INTO PRODUCT VALUES (114, 'razer gaming mouse', 'high precision, speed', 'Accessories', 15000, 130, 8)
        INTO PRODUCT VALUES (115, 'Go X1 webcam', 'captures 1080p video at 30fps', 'Accessories', 6500, 100, 9)
        INTO PRODUCT VALUES (116, 'Airpods Pro', 'active noise cancellation', 'Headphone', 67000, 60, 1)
        INTO PRODUCT VALUES (117, 'Sony 4k Tv', 'displays 4K resolution, HDR', 'TV', 160000, 5, 7)
        INTO PRODUCT VALUES (118, 'SRS Bluetooth Speaker', 'powerful sound', 'Audio', 33000, 42, 7)
    SELECT * FROM dual;

18 rows created.

SQL>
```

Figure 21: : Screenshot of inserting values in product table

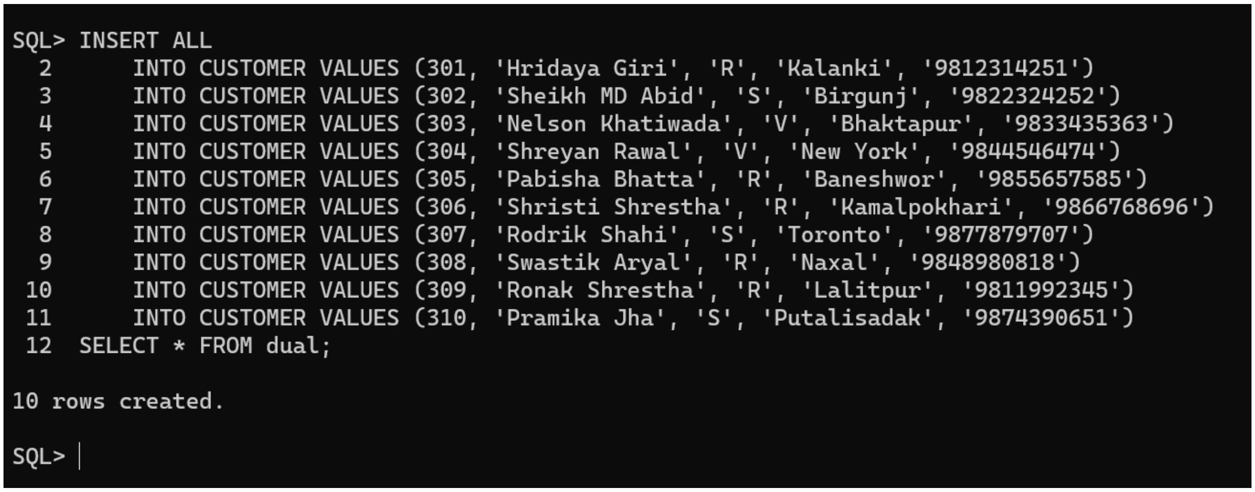
20. Displaying the values inserted in product table.



```
SQL> set linesize 200;
SQL> SELECT * FROM product;

PRODUCT_ID PRODUCT_NAME              DESCRIPTION                         PRODUCT_CATEGOR      PRICE STOCK_LEVEL  VENDOR_ID
---------- ------------------------- ----------------------------------- --------------- ---------- ----------- ----------
       101 iphone 15                 6.1-inch OLED display               Phone               140000          31          1
       102 samsung s23 ultra         6.8-inch OLED display               Phone               215000          51          2
       103 oneplus 11                6.7-inch OLED display               Phone               160000          81          3
       104 redmi note 11             6.43-inch AMOLED display            Phone                30000          23          4
       105 macbook pro m2            14-inch Retina display              Laptop              258000          10          1
       106 dell xps 15               15.6-inch OLED display              Laptop              235000          15          5
       107 dell inspiron             13-17-inch models                   Laptop              110000          20          5
       108 asus zenbook 14           14-inch 2.8K OLED display           Laptop              120000          13          6
       109 apple watch se            fall detection, heart rate monitor  Smartwatch           80000          55          1
       110 Sony headphones           noise canceling, extra bass         Headphone            45000         105          7
       111 Sony Zv camera            20.1-megapixel sensor               Camera              126000           7          7

PRODUCT_ID PRODUCT_NAME              DESCRIPTION                         PRODUCT_CATEGOR      PRICE STOCK_LEVEL  VENDOR_ID
---------- ------------------------- ----------------------------------- --------------- ---------- ----------- ----------
       112 PlayStation 5             supports 4K resolution              Gaming              120000          38          7
       113 keychron keyboard         75% layout, RGB                     Accessories          27000          60          8
       114 razer gaming mouse        high precision, speed               Accessories          15000         130          8
       115 Go X1 webcam              captures 1080p video at 30fps       Accessories           6500         100          9
       116 Airpods Pro               active noise cancellation           Headphone            67000          60          1
       117 Sony 4k Tv                displays 4K resolution, HDR         TV                  160000           5          7
       118 SRS Bluetooth Speaker     powerful sound                      Audio                33000          42          7

18 rows selected.

SQL>
```

Figure 22: Screenshot of displaying values of product table

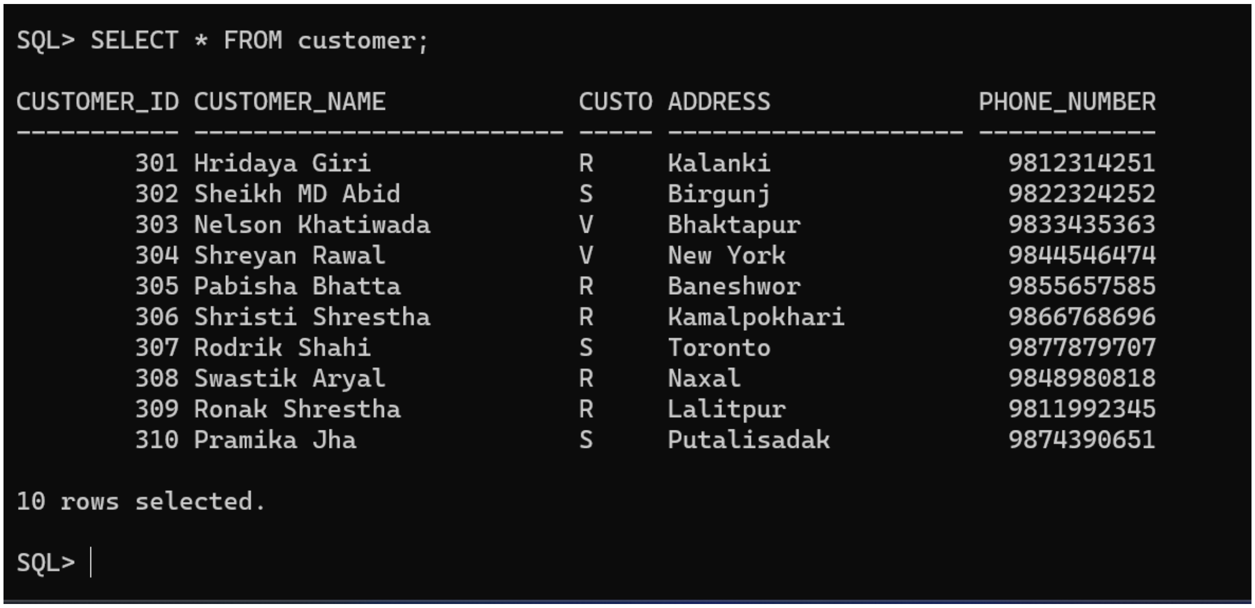
21. Inserting the values in customer table.



```
SQL> INSERT ALL
         INTO CUSTOMER VALUES (301, 'Hridaya Giri', 'R', 'Kalanki', '9812314251')
         INTO CUSTOMER VALUES (302, 'Sheikh MD Abid', 'S', 'Birgunj', '9822324252')
         INTO CUSTOMER VALUES (303, 'Nelson Khatiwada', 'V', 'Bhaktapur', '9833435363')
         INTO CUSTOMER VALUES (304, 'Shreyan Rawal', 'V', 'New York', '9844546474')
         INTO CUSTOMER VALUES (305, 'Pabisha Bhatta', 'R', 'Baneshwor', '9855657585')
         INTO CUSTOMER VALUES (306, 'Shristi Shrestha', 'R', 'Kamalpokhari', '9866768696')
         INTO CUSTOMER VALUES (307, 'Rodrik Shahi', 'S', 'Toronto', '9877879707')
         INTO CUSTOMER VALUES (308, 'Swastik Aryal', 'R', 'Naxal', '9848980818')
         INTO CUSTOMER VALUES (309, 'Ronak Shrestha', 'R', 'Lalitpur', '9811992345')
         INTO CUSTOMER VALUES (310, 'Pramika Jha', 'S', 'Putalisadak', '9874390651')
     SELECT * FROM dual;

10 rows created.

SQL>
```

Figure 23: Screenshot of inserting values in customer table

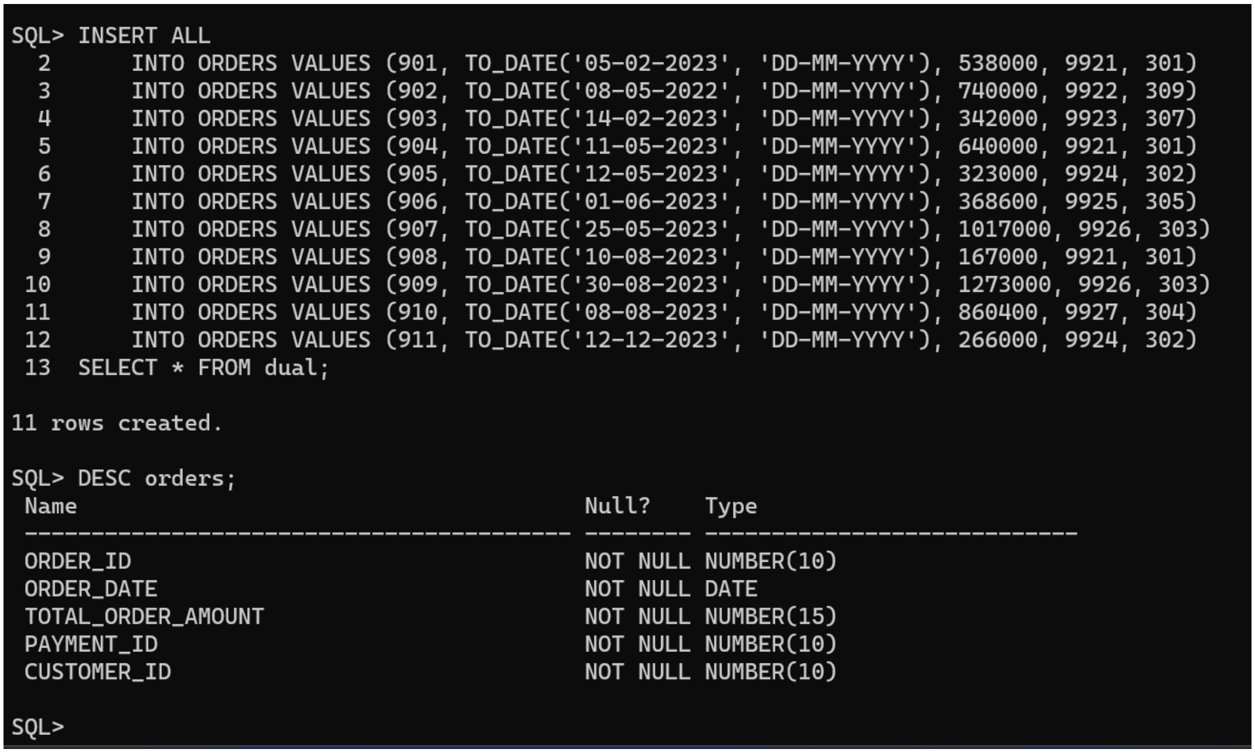
22. Displaying the values inserted in customer table.



```
SQL> SELECT * FROM customer;

CUSTOMER_ID CUSTOMER_NAME             CUSTO ADDRESS              PHONE_NUMBER
----------- ------------------------- ----- -------------------- ------------
        301 Hridaya Giri              R     Kalanki                9812314251
        302 Sheikh MD Abid            S     Birgunj                9822324252
        303 Nelson Khatiwada          V     Bhaktapur              9833435363
        304 Shreyan Rawal             V     New York               9844546474
        305 Pabisha Bhatta            R     Baneshwor              9855657585
        306 Shristi Shrestha          R     Kamalpokhari           9866768696
        307 Rodrik Shahi              S     Toronto                9877879707
        308 Swastik Aryal             R     Naxal                  9848980818
        309 Ronak Shrestha            R     Lalitpur               9811992345
        310 Pramika Jha               S     Putalisadak            9874390651

10 rows selected.

SQL> |
```

Figure 24: Screenshot of displaying values of customer table

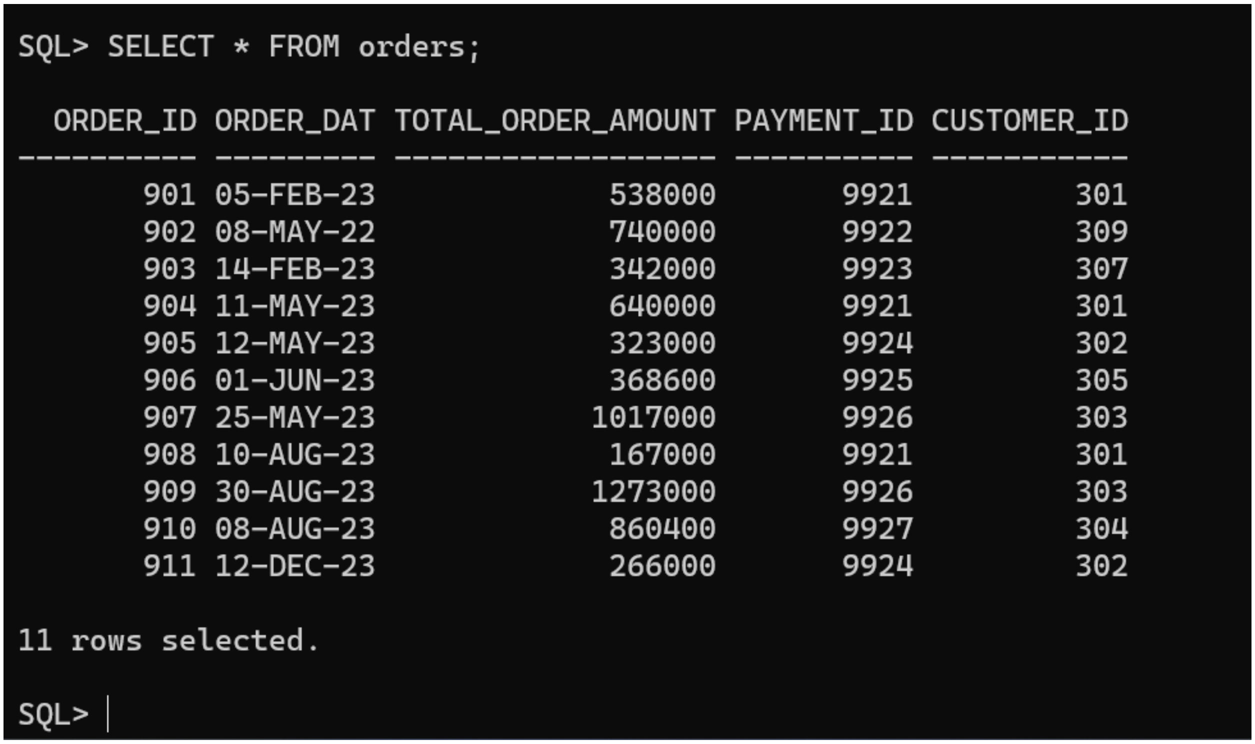
23. Inserting the values in orders table.



```
SQL> INSERT ALL
         INTO ORDERS VALUES (901, TO_DATE('05-02-2023', 'DD-MM-YYYY'), 538000, 9921, 301)
         INTO ORDERS VALUES (902, TO_DATE('08-05-2022', 'DD-MM-YYYY'), 740000, 9922, 309)
         INTO ORDERS VALUES (903, TO_DATE('14-02-2023', 'DD-MM-YYYY'), 342000, 9923, 307)
         INTO ORDERS VALUES (904, TO_DATE('11-05-2023', 'DD-MM-YYYY'), 640000, 9921, 301)
         INTO ORDERS VALUES (905, TO_DATE('12-05-2023', 'DD-MM-YYYY'), 323000, 9924, 302)
         INTO ORDERS VALUES (906, TO_DATE('01-06-2023', 'DD-MM-YYYY'), 368600, 9925, 305)
         INTO ORDERS VALUES (907, TO_DATE('25-05-2023', 'DD-MM-YYYY'), 1017000, 9926, 303)
         INTO ORDERS VALUES (908, TO_DATE('10-08-2023', 'DD-MM-YYYY'), 167000, 9921, 301)
         INTO ORDERS VALUES (909, TO_DATE('30-08-2023', 'DD-MM-YYYY'), 1273000, 9926, 303)
         INTO ORDERS VALUES (910, TO_DATE('08-08-2023', 'DD-MM-YYYY'), 860400, 9927, 304)
         INTO ORDERS VALUES (911, TO_DATE('12-12-2023', 'DD-MM-YYYY'), 266000, 9924, 302)
     SELECT * FROM dual;

11 rows created.

SQL> DESC orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL NUMBER(10)
 ORDER_DATE                                NOT NULL DATE
 TOTAL_ORDER_AMOUNT                        NOT NULL NUMBER(15)
 PAYMENT_ID                                NOT NULL NUMBER(10)
 CUSTOMER_ID                               NOT NULL NUMBER(10)

SQL>
```

Figure 25: Screenshot of inserting values in orders table

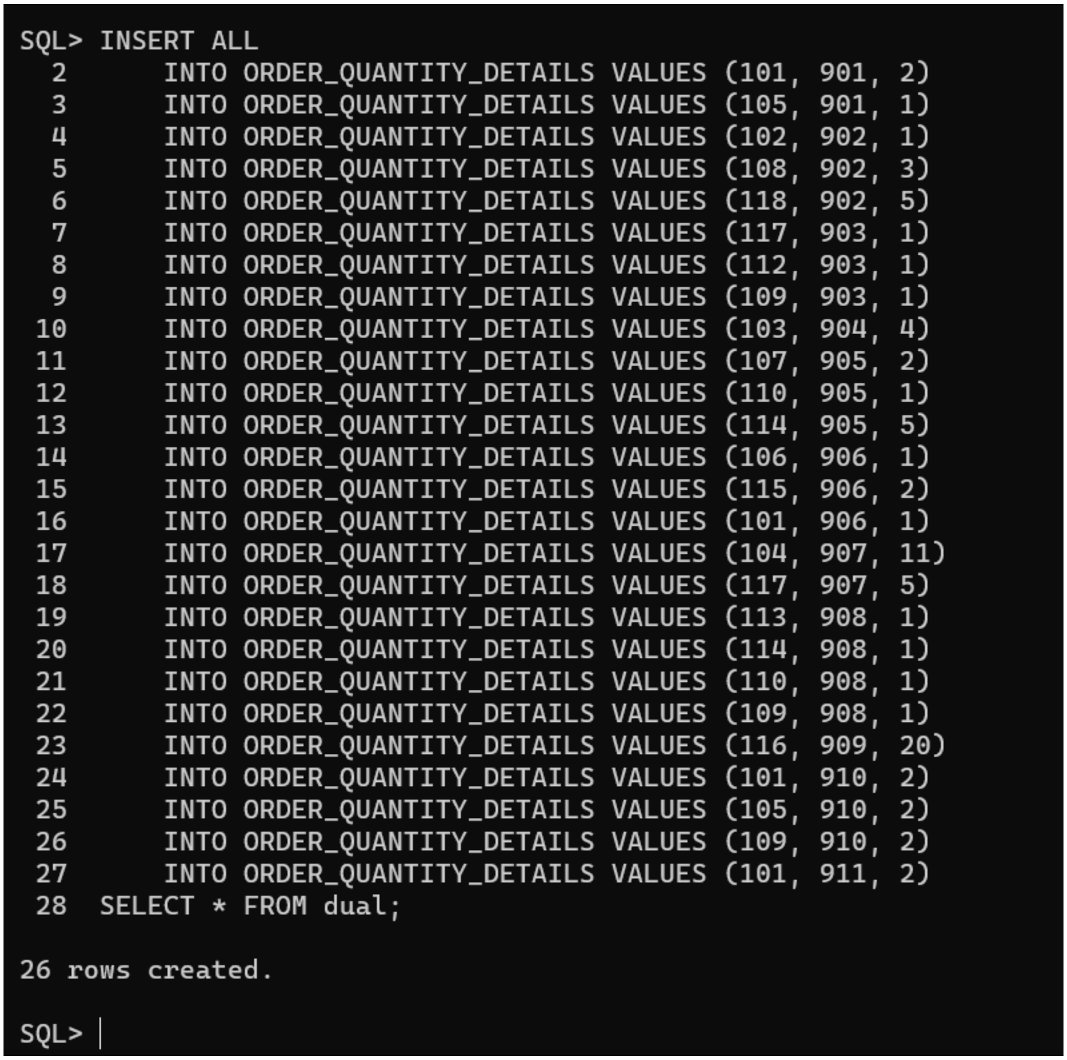
24. Displaying the values inserted in orders table.



```
SQL> SELECT * FROM orders;

  ORDER_ID ORDER_DAT TOTAL_ORDER_AMOUNT PAYMENT_ID CUSTOMER_ID
---------- --------- ------------------ ---------- -----------
       901 05-FEB-23             538000       9921         301
       902 08-MAY-22             740000       9922         309
       903 14-FEB-23             342000       9923         307
       904 11-MAY-23             640000       9921         301
       905 12-MAY-23             323000       9924         302
       906 01-JUN-23             368600       9925         305
       907 25-MAY-23            1017000       9926         303
       908 10-AUG-23             167000       9921         301
       909 30-AUG-23            1273000       9926         303
       910 08-AUG-23             860400       9927         304
       911 12-DEC-23             266000       9924         302

11 rows selected.

SQL> |
```

Figure 26: Screenshot of displaying values of orders table

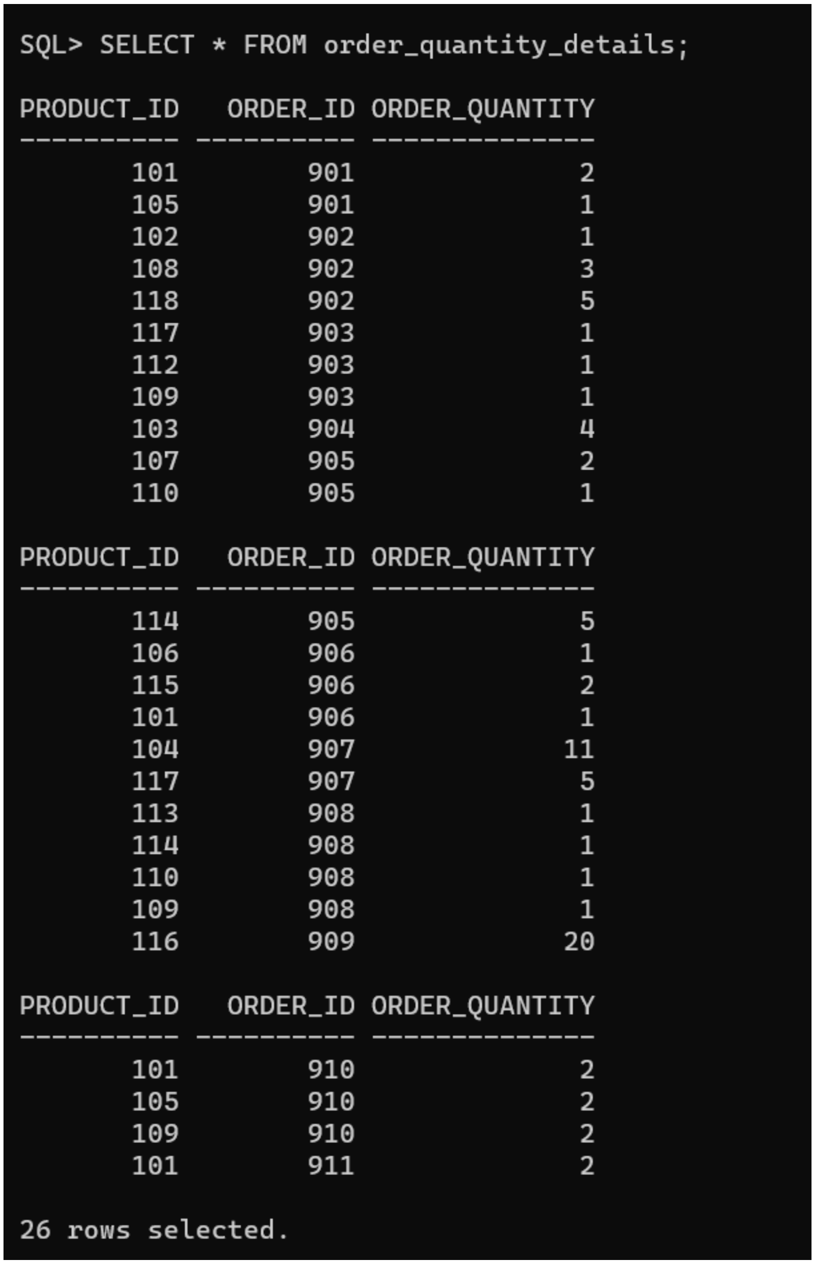
25. Inserting the values in order_quantity_details table.



```
SQL> INSERT ALL
        INTO ORDER_QUANTITY_DETAILS VALUES (101, 901, 2)
        INTO ORDER_QUANTITY_DETAILS VALUES (105, 901, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (102, 902, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (108, 902, 3)
        INTO ORDER_QUANTITY_DETAILS VALUES (118, 902, 5)
        INTO ORDER_QUANTITY_DETAILS VALUES (117, 903, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (112, 903, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (109, 903, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (103, 904, 4)
        INTO ORDER_QUANTITY_DETAILS VALUES (107, 905, 2)
        INTO ORDER_QUANTITY_DETAILS VALUES (110, 905, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (114, 905, 5)
        INTO ORDER_QUANTITY_DETAILS VALUES (106, 906, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (115, 906, 2)
        INTO ORDER_QUANTITY_DETAILS VALUES (101, 906, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (104, 907, 11)
        INTO ORDER_QUANTITY_DETAILS VALUES (117, 907, 5)
        INTO ORDER_QUANTITY_DETAILS VALUES (113, 908, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (114, 908, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (110, 908, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (109, 908, 1)
        INTO ORDER_QUANTITY_DETAILS VALUES (116, 909, 20)
        INTO ORDER_QUANTITY_DETAILS VALUES (101, 910, 2)
        INTO ORDER_QUANTITY_DETAILS VALUES (105, 910, 2)
        INTO ORDER_QUANTITY_DETAILS VALUES (109, 910, 2)
        INTO ORDER_QUANTITY_DETAILS VALUES (101, 911, 2)
    SELECT * FROM dual;

26 rows created.

SQL> |
```

Figure 27: Screenshot of inserting values in order_quantity_details table

26. Displaying the values in order_quantity_details table.



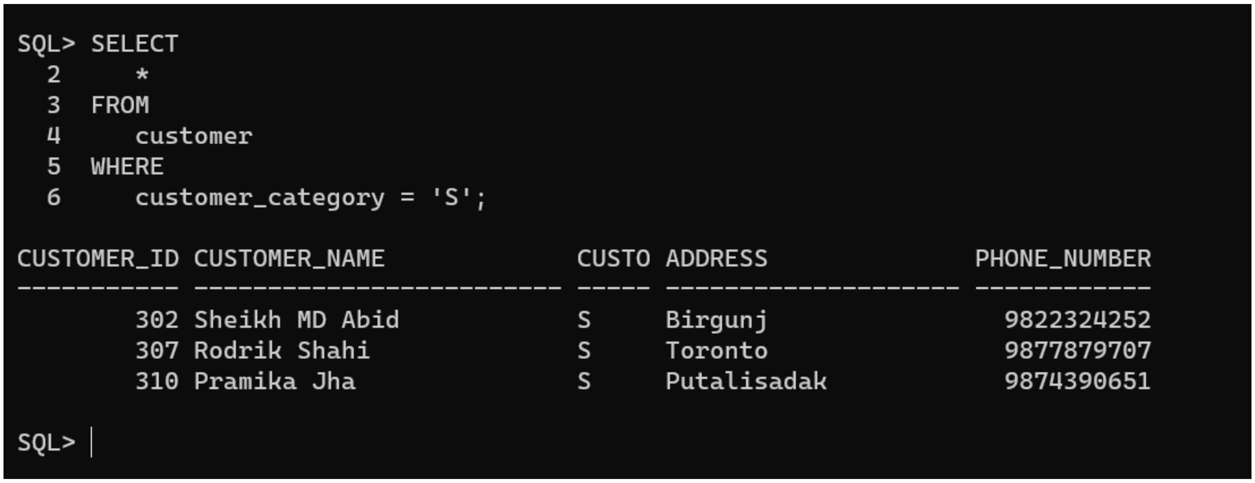
```
SQL> SELECT * FROM order_quantity_details;

PRODUCT_ID   ORDER_ID ORDER_QUANTITY
---------- ---------- --------------
       101        901              2
       105        901              1
       102        902              1
       108        902              3
       118        902              5
       117        903              1
       112        903              1
       109        903              1
       103        904              4
       107        905              2
       110        905              1

PRODUCT_ID   ORDER_ID ORDER_QUANTITY
---------- ---------- --------------
       114        905              5
       106        906              1
       115        906              2
       101        906              1
       104        907             11
       117        907              5
       113        908              1
       114        908              1
       110        908              1
       109        908              1
       116        909             20

PRODUCT_ID   ORDER_ID ORDER_QUANTITY
---------- ---------- --------------
       101        910              2
       105        910              2
       109        910              2
       101        911              2

26 rows selected.
```

Figure 28: Screenshot of displaying values of order_quantity_details table

6. Database Querying

6.1. Information query

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



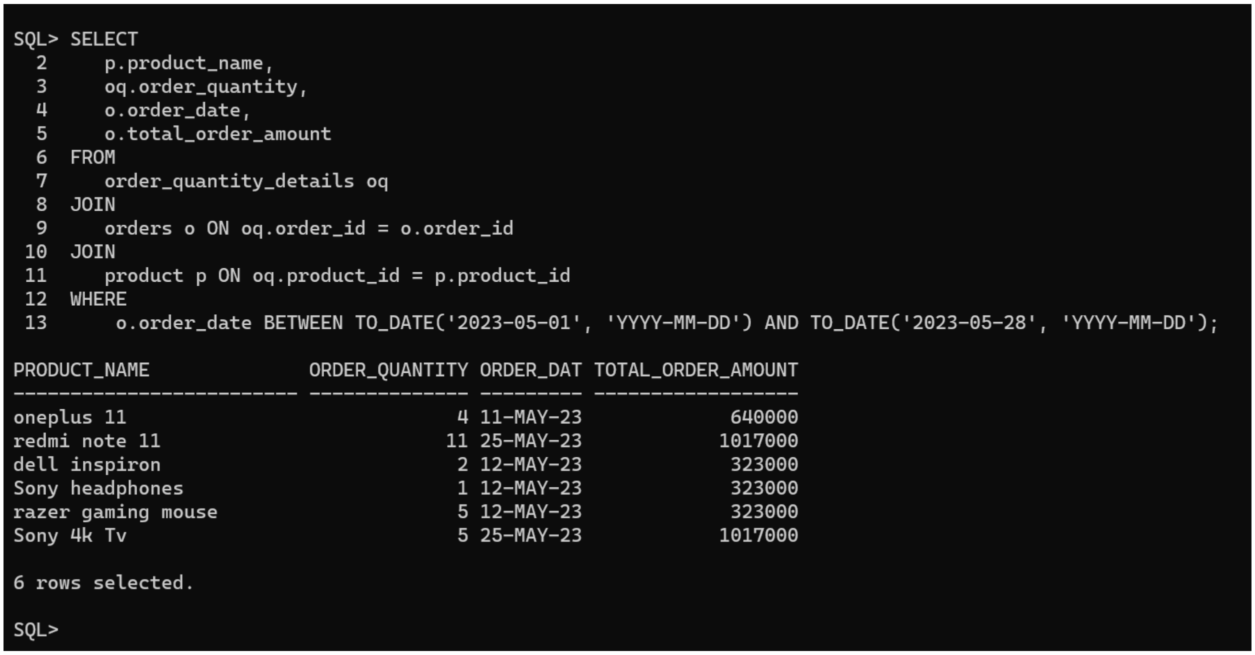
```
SQL> SELECT
        *
     FROM
        customer
     WHERE
        customer_category = 'S';

CUSTOMER_ID CUSTOMER_NAME             CUSTO ADDRESS              PHONE_NUMBER
----------- ------------------------- ----- -------------------- ------------
        302 Sheikh MD Abid            S     Birgunj                9822324252
        307 Rodrik Shahi              S     Toronto                9877879707
        310 Pramika Jha               S     Putalisadak            9874390651

SQL> |
```

Figure 29: Screenshot of information query 1

Purpose: The purpose of this query is to show details of all the customers that are the staff of the company.

Explanation: The above querry displays all the colums(Id, name, category, address and phone number) from customer . “The where customer_category = ‘S’ ” part in the query represents the condition where only customer with category s are displayed.

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



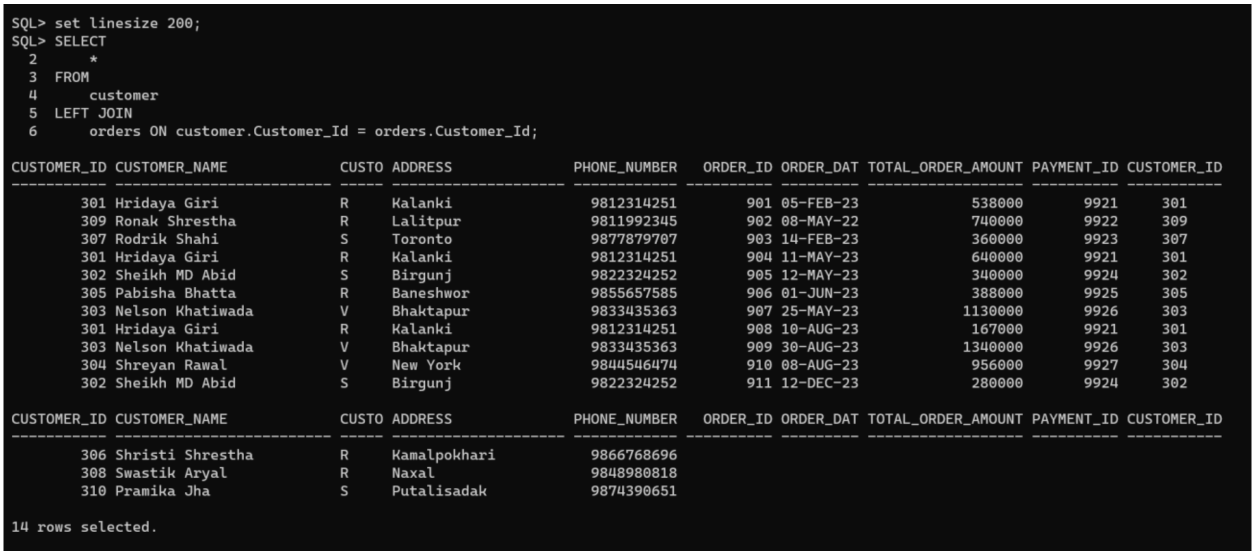
```
SQL> SELECT
        p.product_name,
        oq.order_quantity,
        o.order_date,
        o.total_order_amount
     FROM
        order_quantity_details oq
     JOIN
        orders o ON oq.order_id = o.order_id
     JOIN
        product p ON oq.product_id = p.product_id
     WHERE
         o.order_date BETWEEN TO_DATE('2023-05-01', 'YYYY-MM-DD') AND TO_DATE('2023-05-28', 'YYYY-MM-DD');

PRODUCT_NAME              ORDER_QUANTITY ORDER_DAT TOTAL_ORDER_AMOUNT
------------------------- -------------- --------- ------------------
oneplus 11                             4 11-MAY-23             640000
redmi note 11                         11 25-MAY-23            1017000
dell inspiron                          2 12-MAY-23             323000
Sony headphones                        1 12-MAY-23             323000
razer gaming mouse                     5 12-MAY-23             323000
Sony 4k Tv                             5 25-MAY-23            1017000

6 rows selected.

SQL>
```

Figure 30: Screenshot of information query 2

Purpose: The purpose of this query is to list all the products with their order details made between a particular month.

Explanation: The above query displays product name, quantity, order date, and total order amount of all the products purchased between the 1st May 2023 to 28th May 2023. Three tables order_quantity_details, order and product are joined and condition of date is placed in the query to display the result.

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



```
SQL> set linesize 200;
SQL> SELECT
         *
     FROM
         customer
     LEFT JOIN
         orders ON customer.Customer_Id = orders.Customer_Id;

CUSTOMER_ID CUSTOMER_NAME             CUSTO ADDRESS              PHONE_NUMBER   ORDER_ID ORDER_DAT TOTAL_ORDER_AMOUNT PAYMENT_ID CUSTOMER_ID
----------- ------------------------- ----- -------------------- ------------ ---------- --------- ------------------ ---------- -----------
        301 Hridaya Giri              R     Kalanki                9812314251        901 05-FEB-23             538000       9921     301
        309 Ronak Shrestha            R     Lalitpur               9811992345        902 08-MAY-22             740000       9922     309
        307 Rodrik Shahi              S     Toronto                9877879707        903 14-FEB-23             360000       9923     307
        301 Hridaya Giri              R     Kalanki                9812314251        904 11-MAY-23             640000       9921     301
        302 Sheikh MD Abid            S     Birgunj                9822324252        905 12-MAY-23             340000       9924     302
        305 Pabisha Bhatta            R     Baneshwor              9855657585        906 01-JUN-23             388000       9925     305
        303 Nelson Khatiwada          V     Bhaktapur              9833435363        907 25-MAY-23            1130000       9926     303
        301 Hridaya Giri              R     Kalanki                9812314251        908 10-AUG-23             167000       9921     301
        303 Nelson Khatiwada          V     Bhaktapur              9833435363        909 30-AUG-23            1340000       9926     303
        304 Shreyan Rawal             V     New York               9844546474        910 08-AUG-23             956000       9927     304
        302 Sheikh MD Abid            S     Birgunj                9822324252        911 12-DEC-23             280000       9924     302

CUSTOMER_ID CUSTOMER_NAME             CUSTO ADDRESS              PHONE_NUMBER   ORDER_ID ORDER_DAT TOTAL_ORDER_AMOUNT PAYMENT_ID CUSTOMER_ID
----------- ------------------------- ----- -------------------- ------------ ---------- --------- ------------------ ---------- -----------
        306 Shristi Shrestha          R     Kamalpokhari           9866768696
        308 Swastik Aryal             R     Naxal                  9848980818
        310 Pramika Jha               S     Putalisadak            9874390651

14 rows selected.
```

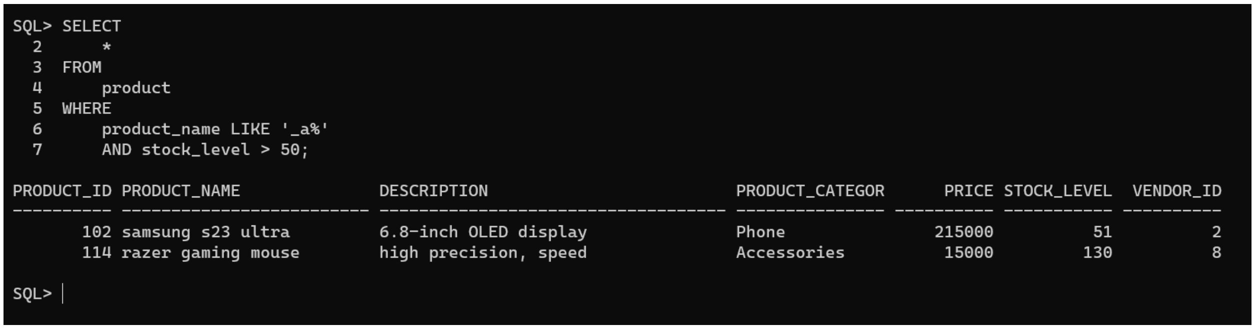
Figure 31: Screenshot of information query 3

Purpose: The purpose of this query is to list all the customer with their order details whether they have made any order or not.

Explanation: The above query displays all the customer details including the order details of all the customers. Order details of those customers who have not made any order is left empty.

For this customer and order table are joined and all the columns are displayed.

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



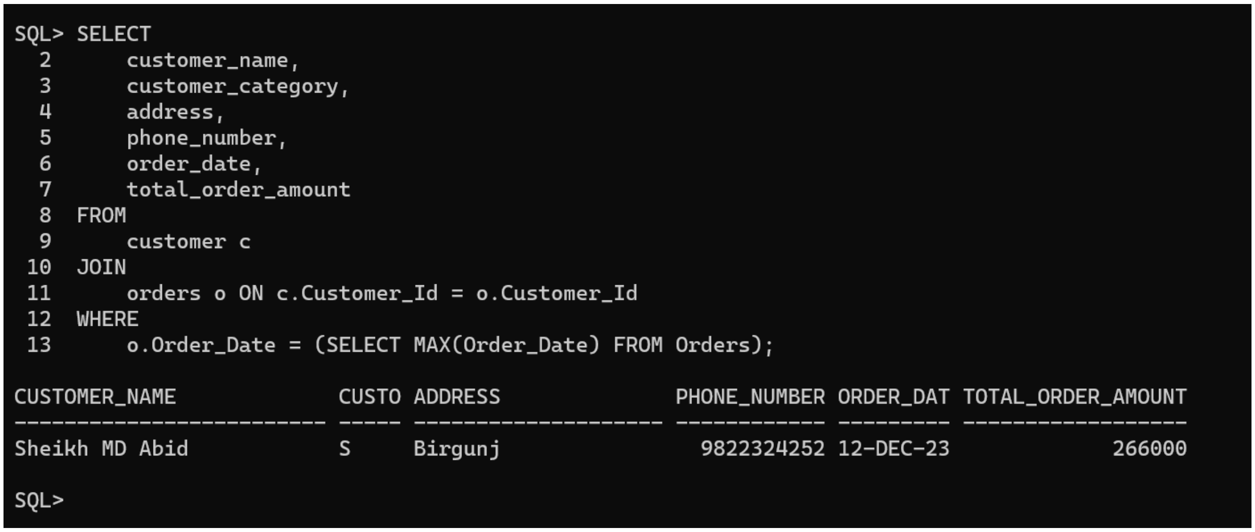
```
SQL> SELECT
         *
     FROM
         product
     WHERE
         product_name LIKE '_a%'
         AND stock_level > 50;

PRODUCT_ID PRODUCT_NAME              DESCRIPTION                          PRODUCT_CATEGOR      PRICE STOCK_LEVEL  VENDOR_ID
---------- ------------------------- ------------------------------------ --------------- ---------- ----------- ----------
       102 samsung s23 ultra         6.8-inch OLED display                Phone               215000          51          2
       114 razer gaming mouse        high precision, speed                Accessories          15000         130          8

SQL> |
```

Figure 32: Screenshot of information query 4

Purpose: The purpose of this query is to list all the details of product having ‘a’ as a second letter in their name and having quantity more than 50 in the stock.

Explanation: The above query displays all the product details like Id, name, description, category, price, stock level and vendor id. In “_a%” _ represents any single character, the a must appear in second then “%” represents any number of any characters. This pattern helps to check for names with second letter ‘a’. Similarly stock > 50 condition is checked and the result is displayed.

5. Find out the customer who has ordered recently.



```
SQL> SELECT
         customer_name,
         customer_category,
         address,
         phone_number,
         order_date,
         total_order_amount
     FROM
         customer c
     JOIN
         orders o ON c.Customer_Id = o.Customer_Id
     WHERE
         o.Order_Date = (SELECT MAX(Order_Date) FROM Orders);

CUSTOMER_NAME             CUSTO ADDRESS              PHONE_NUMBER ORDER_DAT TOTAL_ORDER_AMOUNT
------------------------- ----- -------------------- ------------ --------- ------------------
Sheikh MD Abid            S     Birgunj                9822324252 12-DEC-23             266000

SQL>
```

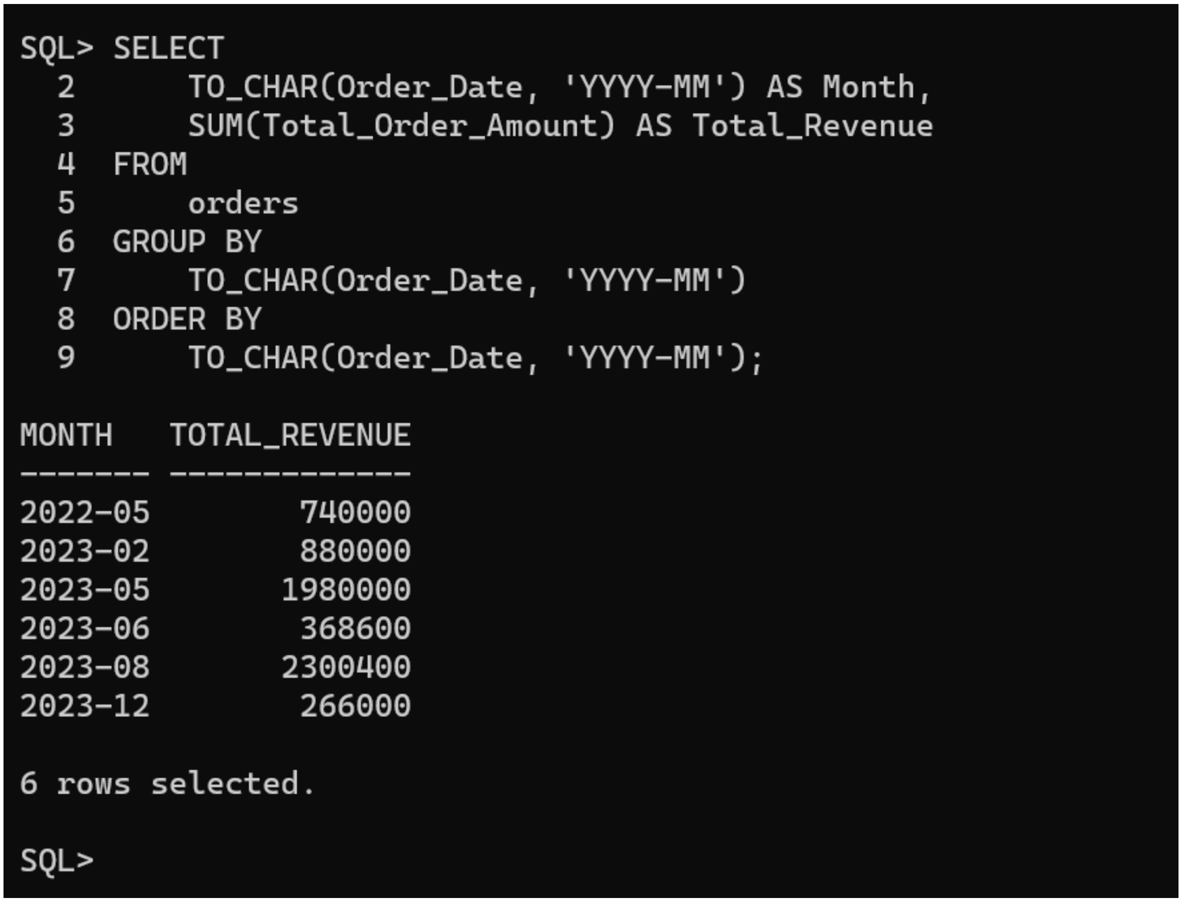
Figure 33: Screenshot of information query 5

Purpose: The purpose of this query is to find the last customer of purchase. His purchase data is closest to the current date among all.

Explanation: The above query displays customer’s name, category, address, ph. no, date of order and total amount paid. For this, two tables i.e orders and customers are joined. Then the order date is matched to maximum order date from orders table and the results are displayed.

6.2. Transaction query

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



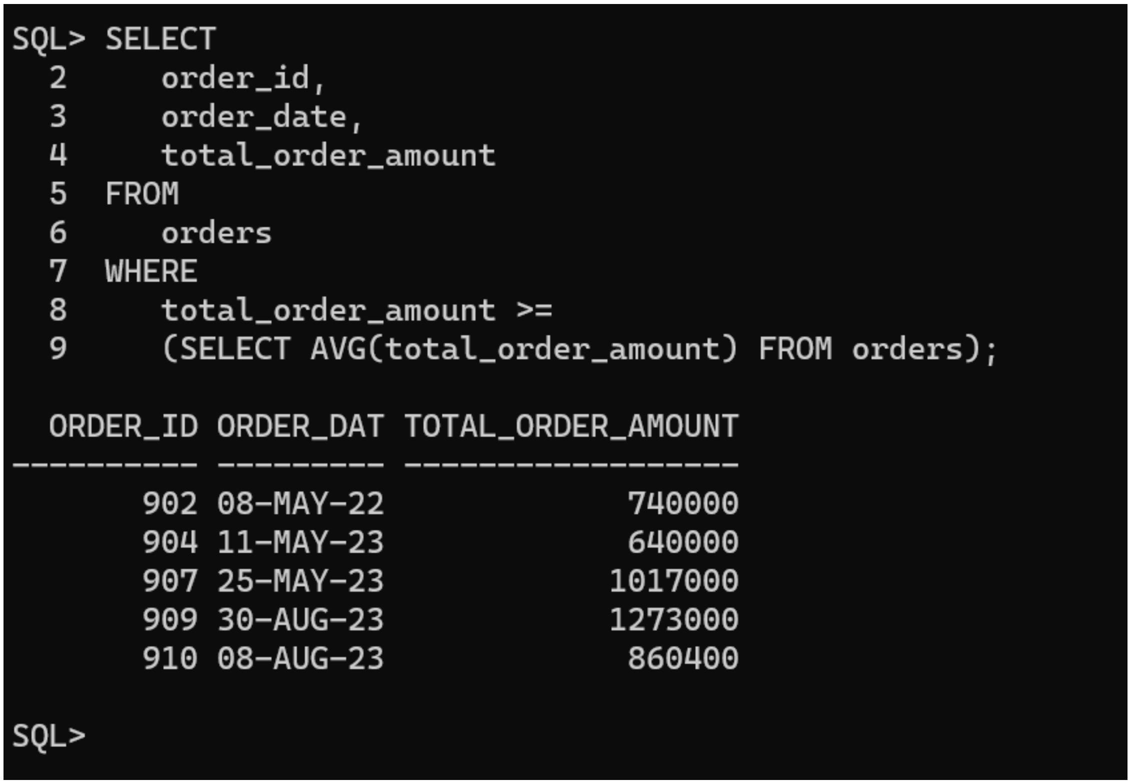
```
SQL> SELECT
         TO_CHAR(Order_Date, 'YYYY-MM') AS Month,
         SUM(Total_Order_Amount) AS Total_Revenue
     FROM
         orders
     GROUP BY
         TO_CHAR(Order_Date, 'YYYY-MM')
     ORDER BY
         TO_CHAR(Order_Date, 'YYYY-MM');

MONTH   TOTAL_REVENUE
------- -------------
2022-05        740000
2023-02        880000
2023-05       1980000
2023-06        368600
2023-08       2300400
2023-12        266000

6 rows selected.

SQL>
```

Figure 34: Screenshot of transaction query 1

Purpose: The purpose of this query is to show the total income generated by the company from selling products each month.

Explanation: This query displays month and corresponding total revenue. For this, order date is grouped according to their months and the total order amount is added. Finally, the month and revenue are displayed. Months with no revenue are not displayed.

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



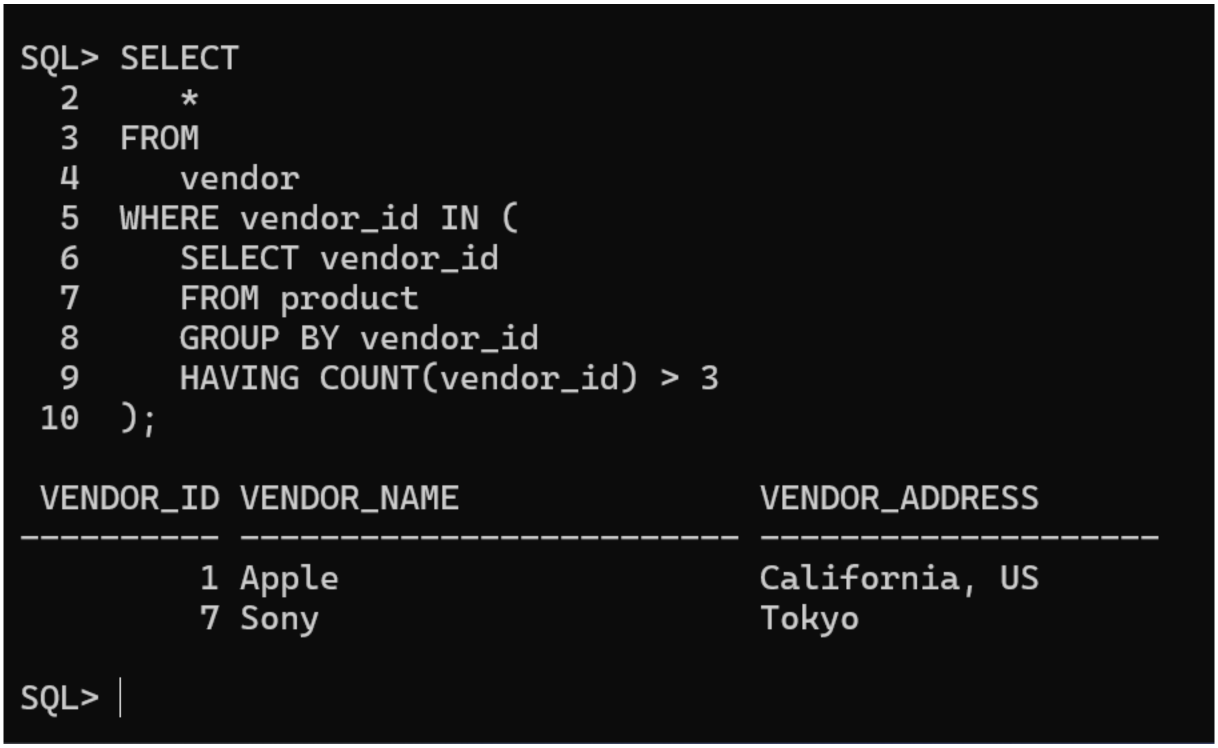
```
SQL> SELECT
        order_id,
        order_date,
        total_order_amount
     FROM
        orders
     WHERE
        total_order_amount >=
        (SELECT AVG(total_order_amount) FROM orders);

  ORDER_ID ORDER_DAT TOTAL_ORDER_AMOUNT
---------- --------- ------------------
       902 08-MAY-22             740000
       904 11-MAY-23             640000
       907 25-MAY-23            1017000
       909 30-AUG-23            1273000
       910 08-AUG-23             860400

SQL>
```

Figure 35: Screenshot of transaction query 2

Purpose: The purpose of this query is to fine the above average orders in case of total order amount.

Explanation: This query displays order id, order date and total order amount of the above average others. For this, total order amount column from orders is simply checked if it is grater then the average.

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



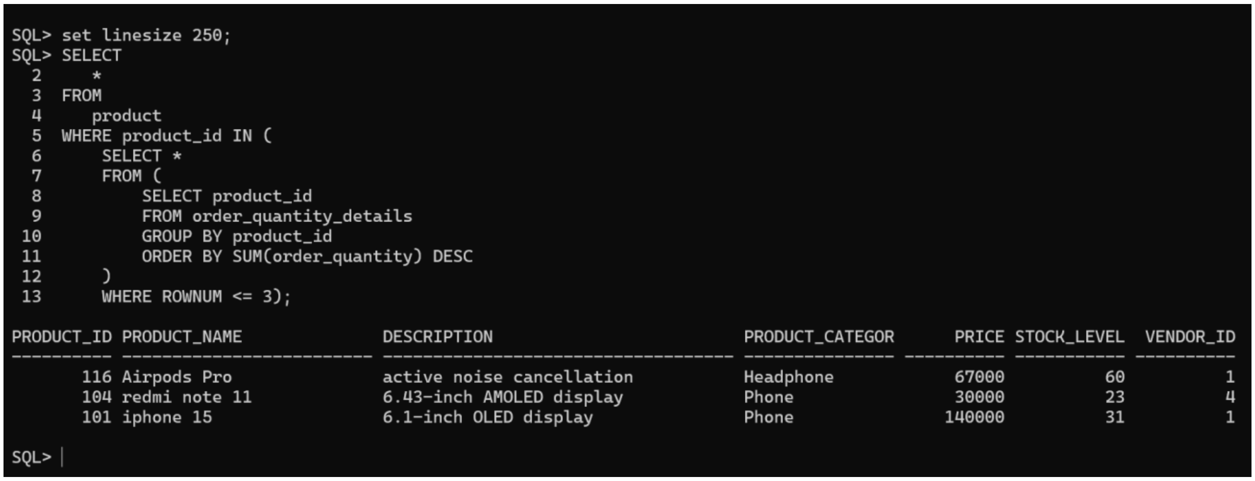
```
SQL> SELECT
  2     *
  3  FROM
  4     vendor
  5  WHERE vendor_id IN (
  6     SELECT vendor_id
  7     FROM product
  8     GROUP BY vendor_id
  9     HAVING COUNT(vendor_id) > 3
 10  );

 VENDOR_ID VENDOR_NAME               VENDOR_ADDRESS
---------- ------------------------- --------------------
         1 Apple                     California, US
         7 Sony                      Tokyo

SQL> 
```

Figure 36: Screenshot of transaction query 3

Purpose: The purpose of this query is to display the details of vendors that supplied more than 3 products

Explanation: This query displays the vendor's id, name and address. At first, vendor id is selected from product table grouping vendor id and checking if it occurs more than 3 times. Then the vendor id from vendor table is checked if it lies in the above condition table. This uses nested select statement.

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



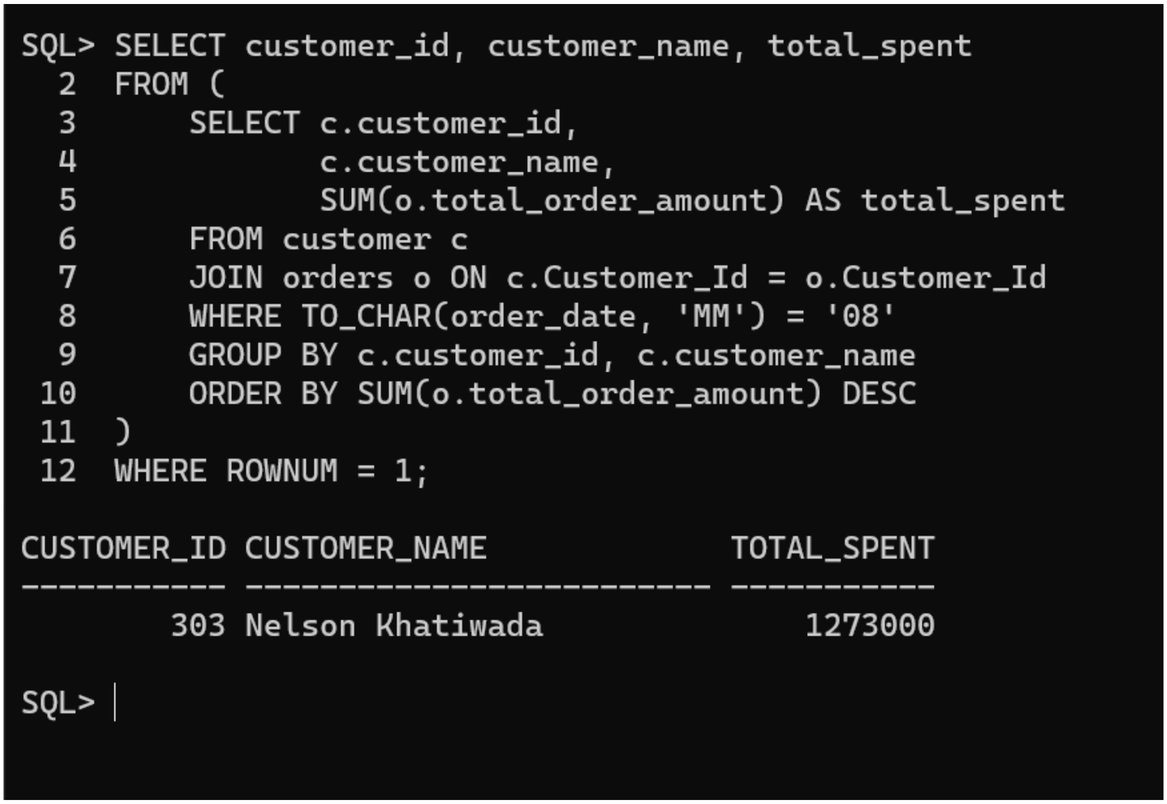
```
SQL> set linesize 250;
SQL> SELECT
        *
     FROM
        product
     WHERE product_id IN (
         SELECT *
         FROM (
             SELECT product_id
             FROM order_quantity_details
             GROUP BY product_id
             ORDER BY SUM(order_quantity) DESC
         )
         WHERE ROWNUM <= 3);

PRODUCT_ID PRODUCT_NAME              DESCRIPTION                        PRODUCT_CATEGOR      PRICE STOCK_LEVEL  VENDOR_ID
---------- ------------------------- ---------------------------------- --------------- ---------- ----------- ----------
       116 Airpods Pro               active noise cancellation          Headphone            67000          60          1
       104 redmi note 11             6.43-inch AMOLED display           Phone                30000          23          4
       101 iphone 15                 6.1-inch OLED display              Phone               140000          31          1

SQL> |
```

Figure 37: Screenshot of transaction query 4

Purpose: The purpose of this query is to display the top 3 product details that have been ordered the most.

Explanation: First linesize is set to 250 for proper display. Then product id of product with most orders is extracted from product_quantity_details table. Then that product id is matched with product id of product table to display all the details of the product. This query also uses nested select statements.

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, total_spent
     FROM (
         SELECT c.customer_id,
                c.customer_name,
                SUM(o.total_order_amount) AS total_spent
         FROM customer c
         JOIN orders o ON c.Customer_Id = o.Customer_Id
         WHERE TO_CHAR(order_date, 'MM') = '08'
         GROUP BY c.customer_id, c.customer_name
         ORDER BY SUM(o.total_order_amount) DESC
     )
     WHERE ROWNUM = 1;

CUSTOMER_ID CUSTOMER_NAME             TOTAL_SPENT
----------- ------------------------- -----------
        303 Nelson Khatiwada              1273000

SQL>
```

Figure 38: Screenshot of transaction query 5

Purpose: The purpose of this query is to display the customer who has spent the most in the month of August including his/her total spending

Explanation: The above query displays customer id, name and total spent of most spent customer in August. For this, customer and orders table are joined, and the condition of month is checked. Grouping by customer id and name is done where the total order amount is added (sum) to calculate the total spent. It is ordered in descending order to bring the most spent customer at the top. Finally, all the rows are selected and "ROWNUM = 1" gives the first row which is the top customer.

6.3 Creation of Spool file

Spool file is created for all the above (10) queries. The syntax for spool file is :

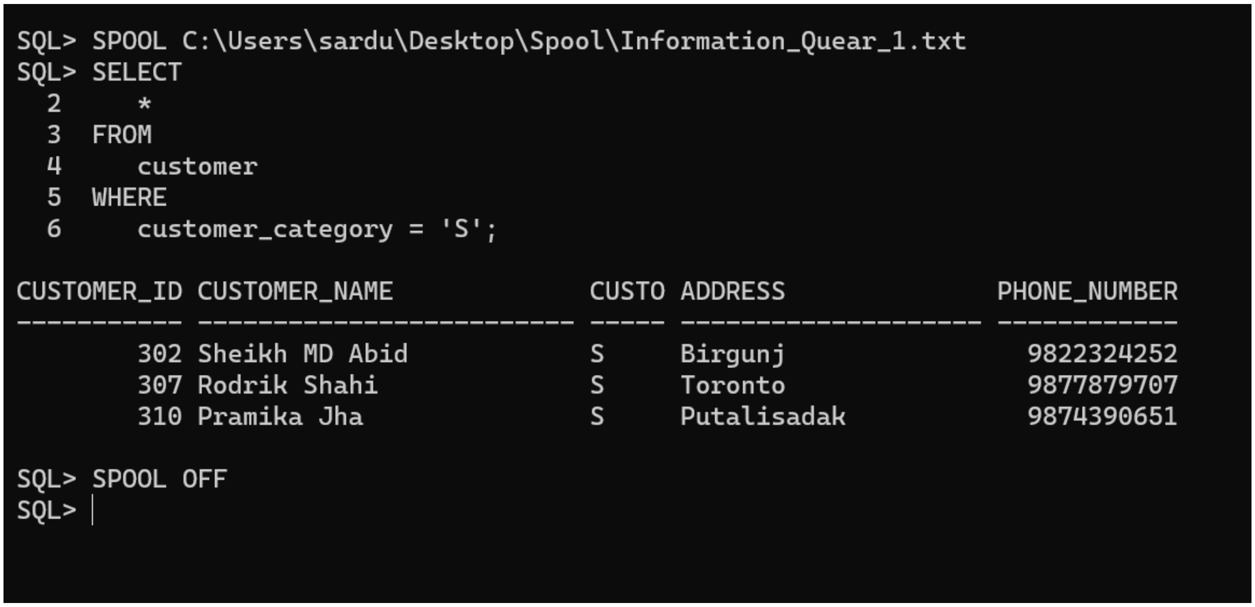
SPOOL path\name.txt

Query

SPOOL OFF

Where path is the path of folder to create spool file, name is the name of the spool file to be created and Query is the query to be displayed in the file.

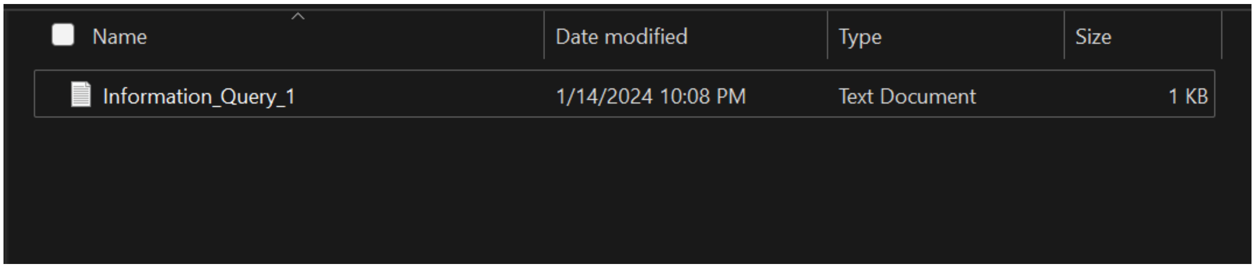
Creating a spool file for information query 1:



```
SQL> SPOOL C:\Users\sardu\Desktop\Spool\Information_Quear_1.txt
SQL> SELECT
         *
     FROM
         customer
     WHERE
         customer_category = 'S';

CUSTOMER_ID CUSTOMER_NAME             CUSTO ADDRESS              PHONE_NUMBER
----------- ------------------------- ----- -------------------- ------------
        302 Sheikh MD Abid            S     Birgunj                9822324252
        307 Rodrik Shahi              S     Toronto                9877879707
        310 Pramika Jha               S     Putalisadak            9874390651

SQL> SPOOL OFF
SQL>
```

Figure 39: Screenshot of creating spool file for information query 1

Spool file is created:



Name	Date modified	Type	Size
Information_Query_1	1/14/2024 10:08 PM	Text Document	1 KB

Figure 40: Screenshot of spool file created in the given path

7. Critical Evaluation

The 'Database' module is one of the most important modules for me personally as a Data Analyst Enthusiast. We had the chance to learn about one of the most powerful databases, 'Oracle SQL,' and its Database Management System (DBMS). It taught us about creating a database, querying to retrieve information, and manipulating it. Many problems may occur when we create a database, so we learned about normalization. We were taught to make ERDs to show relations between entities, and finally, it ended with relational algebra.

It is said that "Data is the future, and the future is now." In any company, lots of data is present and must be stored somewhere for proper retrieval. This data can also help in making business decisions. So, the things taught in this module are almost used for any software application like in the backend of a website or an app, data analysis, training the model using data in machine learning, Blockchain, Internet of Things, and even Artificial Intelligence. This 'Database' module is closely related to another module, 'Data Structures and Specialist Programming,' where our task was to create desktop applications. In the next semester, we will be taught to connect the created application to a database.

This coursework was very important as it taught us the concept of a database through a real-life scenario of “Gadget Emporium”. We had to write the business rules, identify entities and attributes, create ERDs, normalize, and finally query and drop the database. We got to learn about databases from creation to deletion. Lots of hard work and understanding of the database concept were required for this task. Hence, it was fruitful to sharpen our understanding of databases.

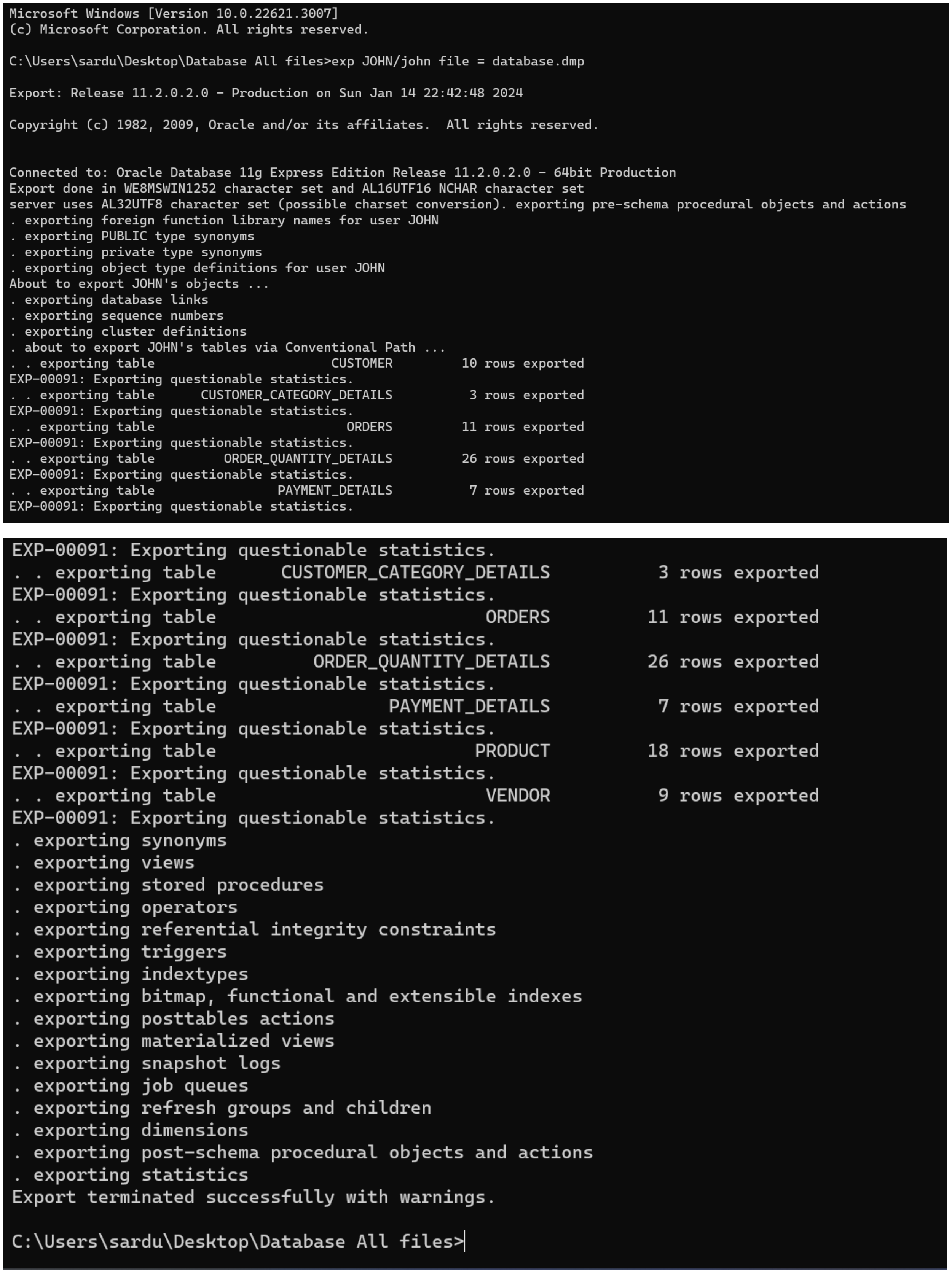
8. Drop Query and Database Dump file creation

8.1 Dump file creation

A file that contains database structure and content is a dump file. It can be created in many ways. We created a dump file for our database using “exp username/password file = database.dmp” in the command prompt. Here, username/password is the name when creating a user and its password, and “database.dmp” is then name of our dump file.

Some snapshots of creating the dump file :

1. A dump file created after locating the path to create the dump file in the command prompt.

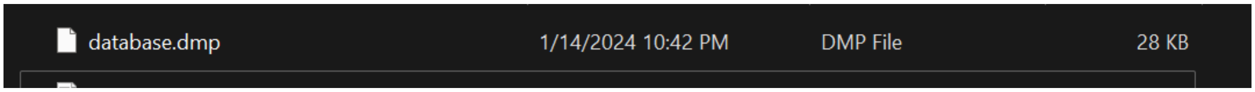


```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sardu\Desktop\Database All files>exp JOHN/john file = database.dmp

Export: Release 11.2.0.2.0 - Production on Sun Jan 14 22:42:48 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 JOHN
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user JOHN
About to export JOHN's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export JOHN's tables via Conventional Path ...
. . exporting table                       CUSTOMER         10 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table      CUSTOMER_CATEGORY_DETAILS          3 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                         ORDERS         11 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table         ORDER_QUANTITY_DETAILS         26 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                PAYMENT_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
```

```
EXP-00091: Exporting questionable statistics.
. . exporting table      CUSTOMER_CATEGORY_DETAILS          3 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                         ORDERS         11 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table         ORDER_QUANTITY_DETAILS         26 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                PAYMENT_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        PRODUCT         18 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                         VENDOR          9 rows exported
EXP-00091: Exporting questionable statistics.
. 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
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.

C:\Users\sardu\Desktop\Database All files>
```

Figure 41: Screenshot of creating dump file

2. Dump filed named “database.dmp” created in the mentioned path.

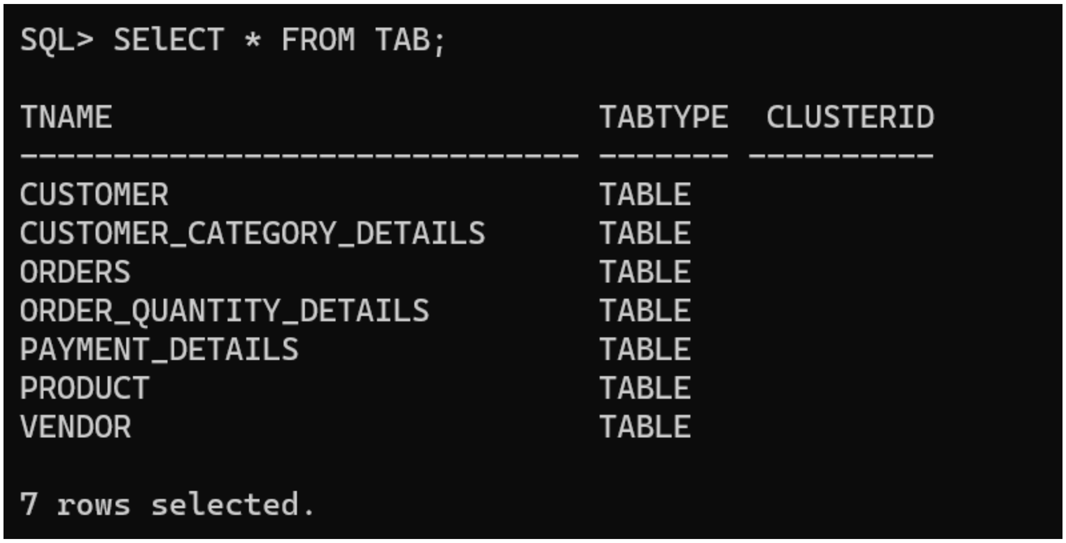


```
database.dmp                1/14/2024 10:42 PM        DMP File            28 KB
```

Figure 42: Screenshot of dumpfile created in the path

8.2 Drop Query

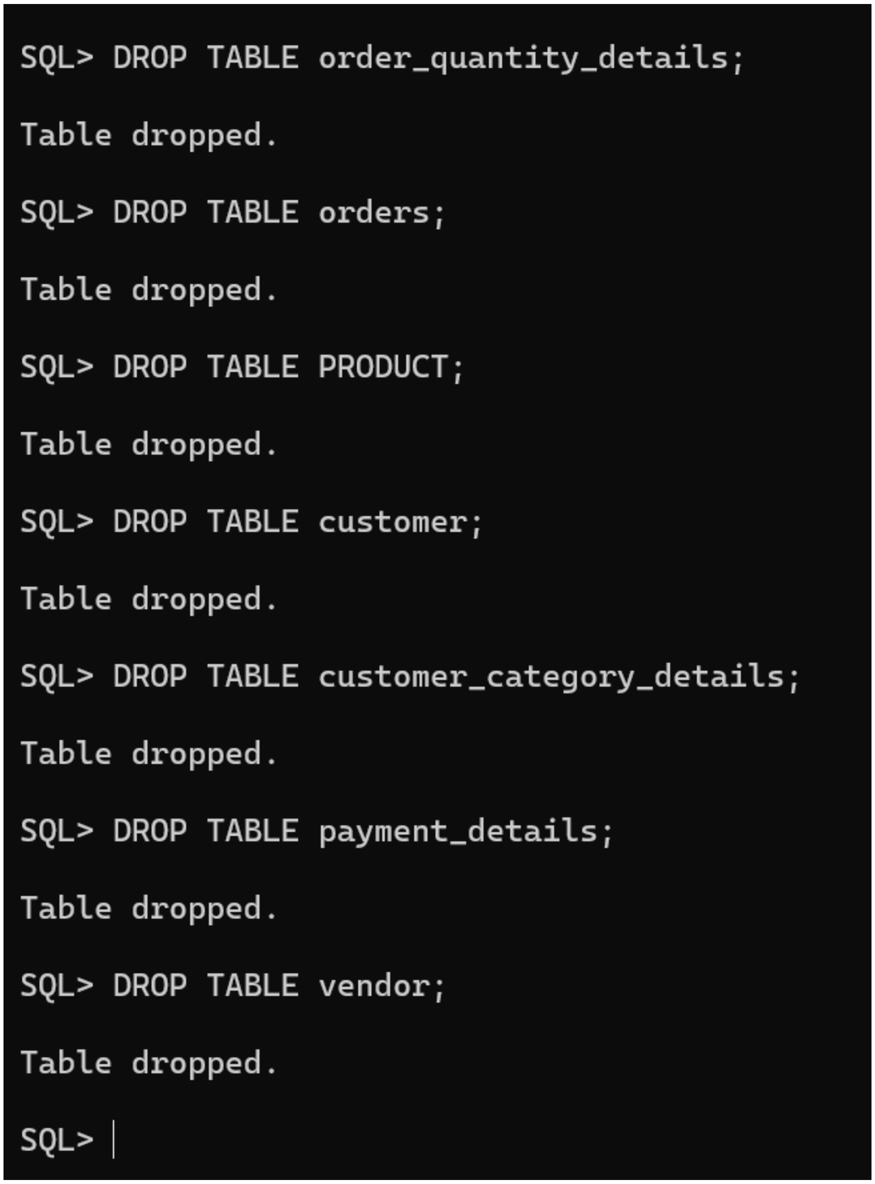
The tables before dropping:



```
SQL> SElECT * FROM TAB;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
CUSTOMER                       TABLE
CUSTOMER_CATEGORY_DETAILS      TABLE
ORDERS                         TABLE
ORDER_QUANTITY_DETAILS         TABLE
PAYMENT_DETAILS                TABLE
PRODUCT                        TABLE
VENDOR                         TABLE

7 rows selected.
```

Figure 43: Screenshot of tables before dropping

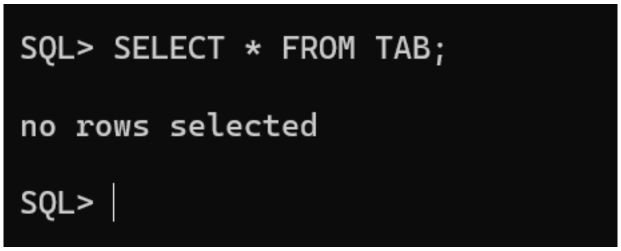
The tables are dropped in the order of child to parent:



```
SQL> DROP TABLE order_quantity_details;

Table dropped.

SQL> DROP TABLE orders;

Table dropped.

SQL> DROP TABLE PRODUCT;

Table dropped.

SQL> DROP TABLE customer;

Table dropped.

SQL> DROP TABLE customer_category_details;

Table dropped.

SQL> DROP TABLE payment_details;

Table dropped.

SQL> DROP TABLE vendor;

Table dropped.

SQL>
```

Figure 44: Screenshot of dropping the tables

The tables after dropping:



```
SQL> SELECT * FROM TAB;

no rows selected

SQL>
```

Figure 45: Screenshot to tables after dropping

9. Conclusion

In this report, the creation of a database from scratch has been conducted, and it was successfully executed. All ten queries were run successfully, and all identified problems were solved. All of the project's objectives, as outlined in the coursework, were met.

This coursework has been a valuable learning experience, aiding me in acquiring knowledge, skills, and experience. I became much more familiar with Oracle DBMS, SQL, MS Word, Draw.io, and various other tools. The real-world scenario presented in this problem deepened my understanding of how a database is created and its practical applications. Additionally, this project enhanced my research and report writing skills. All these skills and knowledge will be useful in my future endeavors.

I encountered many challenges during my coursework journey. Firstly, normalization was a significant challenge for me. Managing the attributes according to the question and organizing the database according to business rules posed challenges. Similarly, illustrating the relationships in the Final Entity-Relationship Diagram (ERD) and managing data to fill in the database so that it aligns with the queries' output required a lot of work.

Overall, this project has been a challenging but rewarding experience. I am proud that I was able to complete this work and would like to thank everyone, especially our lecturer and tutor, who helped me make this project a success. It was a great learning experience, and I am grateful to have been a part of it.