LONDON
METROPOLITAN
UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI - Databases

50% Individual Coursework

2023-24 Spring

Student Name: Aayushree Dahal

London Met ID: 22067884

College ID: NP01CP4A220265

Assignment Due Date: Sunday, July 28, 2024

Assignment Submission Date: Sunday, July 28, 2024

Word Count: 2869

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher 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 marks of zero will be awarded.

Table of Contents

Table of Figures

Tables of Tables

1. Introduction

The Gadget Emporium is an online store focused on selling electronics and accessories led by Mr. John. The purpose of establishing the “Gadget Emporium” online platform is to provide both private consumers and business organizations with a diverse section of electronics devices. The following product names, descriptions, categories, prices and stock levels of electrical devices and accessories should all be maintained by the system. The platform leads to tracking the system which divides customers into Regular (R), Staff (S), and VIP (V). The system links with several kinds of payment gateways to supply encrypted and simple order transactions. The online store needs to have a flexible and efficient digital infrastructure that meets the immediate needs of the online store while providing a solid foundation for future growth and expansion. The proposed design is required to manage all customers, products and orders records.

1.1 Aims and Objectives

The main aims and objectives of this project are mentioned below:

- It creates and implements a strong database system to support the new e-commerce plan.
- It serves as the foundation of Gadget Emporium facilitating the efficient and well-structured functioning of the online marketplace.
- It manages regular operations effectively and allows for future growth and improvements for Gadget Emporium.
- It tracks real-time product availability to prevent overselling and maintain accurate stock levels.
- Its goal is to satisfy both individual customers and commercial customers by supplying a wide range of electronic products.

1.2 Business Activities and Operations

The following business activities and operations of Gadget Emporium are mentioned below:

1. Engaged with multiple payment gateways to ensure secure and seamless transactions of each purchase.
2. Tracks product supply to prevent overcharging and maintain specific stock levels.
3. Created the records of vendors or suppliers providing electronic gadget and accessories.
4. Different discount rates on product purchases are made accessible for various customer categories.
5. Improve the customer counter by providing choices like discounts, promotions, and customer feedback.
6. Protecting private customer information using effective safety regulations including hashed and encrypted passwords.

1.3 Business Rules

The following Business Rules of Gadget Emporium are mentioned below:

1. Each product needs to have only one category.
2. Each category can have one or many products.
3. Each customer must be categorized as Regular (R), Staff (S), and VIP (V).
4. Each category refers to a different discount rate on product purchases, such as 0%, 5%, and 10% respectively.
5. Each customer can check out and obtain one or more electronics gadgets online.
6. Each order can have multiple products.
7. Each product can be multiple orders placed by numerous customers.
8. Each product must supply details like stock quality or availability status.
9. Each vendor can supply one or more products.
10. Each product should be associated with a single vendor.
11. Each order detail must have one payment option.
12. Each invoice must have one payment option.
13. Each invoice needs to be issued once the customer checks out their order that follows confirmation which contains the details and information of order, customer, and payment details.

2. Entities and Attributes

Entities are a fundamental concept in the design and management of databases. An entity is separated from additional products by its independent existence. A quality database provides information about the relationships between its entities. Every entity is distinguished by a primary key, which acts as a unique identifier, and an attribute is implemented to define the qualities of each entity. To entirely understand their importance in the context of data management, it is necessary to go more deeply into each of their primary attributes (Taylor, December 15, 2023). The following entities and attributes for storing information in database are mentioned below:

2.1 Customers

Attributes	**Data Type**	**Constraints**
customerId	VARCHAR (255)	PRIMARY KEY (PK), Unique
customerAddress	VARCHAR (255)	NOT NULL
customerName	VARCHAR (255)	NOT NULL
customerCategory	VARCHAR (255)	NOT NULL
discountRate	DECIMAL (5,2)	NOT NULL

Table 1: Entities and Attributes of Customers.

2.2 Order

Attributes	Data Type	Constraints
orderId	VARCHAR (255)	PRIMARY KEY (PK), Unique
paymentDetails	VARCHAR (255)	NOT NULL
orderDate	DATE	NOT NULL
totalAmount	INT	NOT NULL
disAmount	INT	NOT NULL
billId	VARCHAR (255)	NOT NULL
billDate	DATE	NOT NULL
quantity	INT	NOT NULL
totalPrice	INT	NOT NULL

Table 2: Entities and Attributes of Order.

2.3 Product

Attributes	Data Type	Constraints
productId	VARCHAR (255)	PRIMARY KEY (PK), Unique
productName	VARCHAR (255)	NOT NULL
productPrice	DECIMAL (10,2)	NOT NULL
productStock	INT	NOT NULL
productDescription	VARCHAR (255)	NOT NULL
productcategoryId	VARCHAR (255)	NOT NULL
productcategoryName	VARCHAR (255)	NOT NULL
vendorName	VARCHAR (255)	NOT NULL
vendorId	VARCHAR (255)	NOT NULL

Table 3: Entities and Attributes of Product.

2.4 Initial ERD

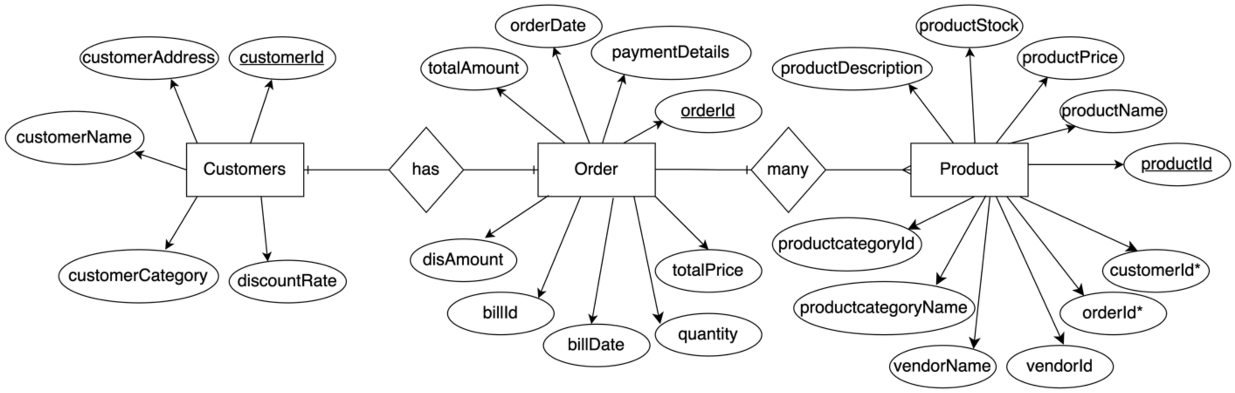


Figure 1: Initial ERD.

3. Normalization

Normalization is the process of minimizing redundancy from a relation or set of relations. This method requires numerous stages to organize the data in tabular form and remove unnecessary data from relational databases. It arranges database columns and sections to guarantee that database integrity constraints can be carried out their demand effectively. It permits simple retrieval, simplifies data maintenance, and reduces the need to restructure data (Chris, December 21, 2022).

The following section includes the four-normalization types that are normally used in relational databases:

3.1 UNF

UNF is an un-normalized form normalization procedure that allows users to generate a structured frame that accurately represents an organizational database. UNF intends to begin with a brief collection of qualities and then specify which detailed sections might contain data that is repeated (DBS211, 2024).

Applying the above table of UNF produces the following results:

Customers(customerId, customerAddress, customerName, customerCategory, discountRate, {orderId, paymentDetails, orderDate, totalAmount, disAmount, billId, billDate, quantity, totalPrice, {productId, productName, productPrice, productStock, productDescription, productcategoryId, productcategoryName, vendorName, vendorId}})

3.2 1NF

1NF, known as the first normal form, requires that all attributes contain atomic values of single, indivisible data and there are no repeating groups within the table. Additionally, each attribute must have a unique name. The 1NF ensures that each attribute of the data maintains a single value rather than numerous values by removing duplicate columns of repetitive data (Anand, July 16, 2024).

Unnormalized form (UNF) to first normal form (1NF):

Eliminating duplicate columns of repetitive data:

Customers-1(customerId, customerAddress, customerName, customerCategory, discountRate)

Orders-1(orderId, customerId*, totalAmount, disAmount, orderDate, paymentDetails, billId, billDate, quantity, total_price)

Products-1(productId, customerId*, orderId*, productDescription, productName, productStock, productPrice, productcategoryId, productcategoryName, vendorName, vendorId)

3.3 2NF

2NF, known as the second normal form, requires that a relation is in the first normal form and that every non-key attribute is fully functionally dependent on the primary key, that mean there are no part-key dependencies. In the second normal formal, the identification of functional dependence is required. It creates relationships between these new tables and their predecessors using foreign keys (RoseIndia, 2024).

For order table,

orderId, customerId → quantity, totalPrice

customerId → no attributes

orderId → orderId, orderDate, disAmount, totalAmount, paymentDetails, billId, billDate)

Orders-2(orderId, orderDate, totalAmount, disAmount, paymentDetails, billId, billDate)

OrderCustomer-2**(**customerId*, orderId*, quantity, totalPrice)

For product table,

productId, customerId, orderId → quantity, totalPrice

customerId → no attributes

productId → productName, productDescription, productPrice, productStock, productcategoryId, productcategoryName, vendorId, vendorName)

orderId → no attributes

Products-2(productId, productName, productDescription, productPrice, productStock, productcategoryId, productcategoryName, vendorId, vendorName)

ProductsCustomersOrders-2(customerId*, orderId*, productId*, quantity, totalPrice)

For customer table,

There are no partial dependencies in customerTable.

Separating the table after removing partial dependencies:

Customers-2(customerId, customerName, customerAddress, customerCategory, discountRate)

Orders-2(orderId, orderDate, totalAmount, disAmount, paymentDetails, billId, billDate)

OrderCustomer-2(customerId*, orderId*)

Products-2(productId, productName, productDescription, productPrice, productStock, productcategoryId, productcategoryName, vendorId, vendorName)

ProductsCustomersOrders-2(customerId*, orderId*, productId*, quantity, totalPrice)

3.4 3NF

3NF, known as the third normal form, is designed to eliminate transitive dependencies. These dependencies can lead to data inconsistencies and redundancies which helps minimize data duplication, making the database more efficient and easier to maintain. It simplifies data manipulation because managing, updating, and querying data is made simpler by an organized table in 3NF (Sileshi, June 2, 2024).

For Customers table,

customerId → customerCategory → discountRate

For Orders table,

orderId → invoiceId → billDate, paymentDetails, disAmount

For Products table,

productId → productcategoryId → productcategoryName

productId → vendorId → vendorName

There are no transitive dependencies in the ProductsCustomersOrders table.

Removing transitive dependencies,

Customers-3(customerId, customerName, customerAddress, customerCategory*)

Cus_category-3(customerCategory, discountRate)

Orders-3(orderId, orderDate, totalAmount, billId*)

Bill-3**(**billId, billDate, paymentDetails, disAmount)

Products-3(productId, productName, productDescription, productPrice, productStock, productcategoryId*, vendorId*)

ProductCategory-3(productcategoryId, productcategoryName)

Vendor-3(vendorId, vendorName)

ProductsCustomersOrders-3(customerId*, orderId*, productId*, quantity, totalPrice)

4. Final ERD

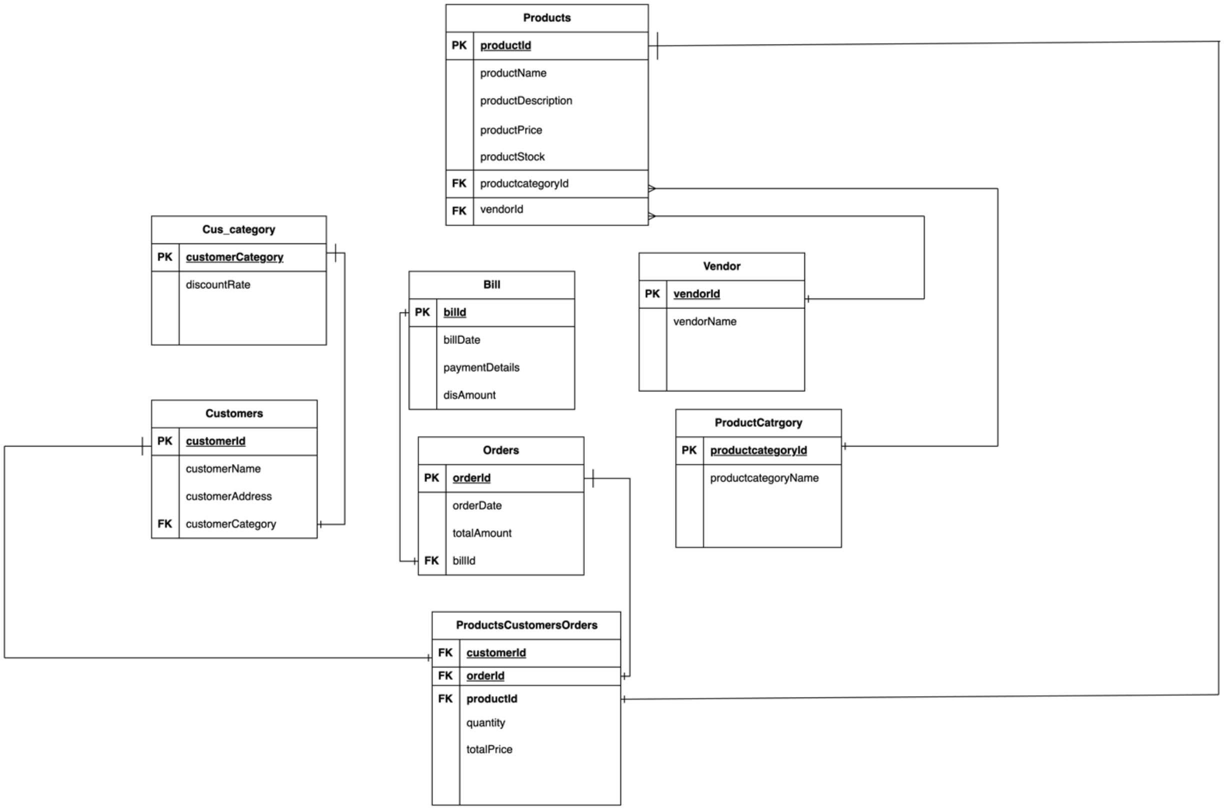


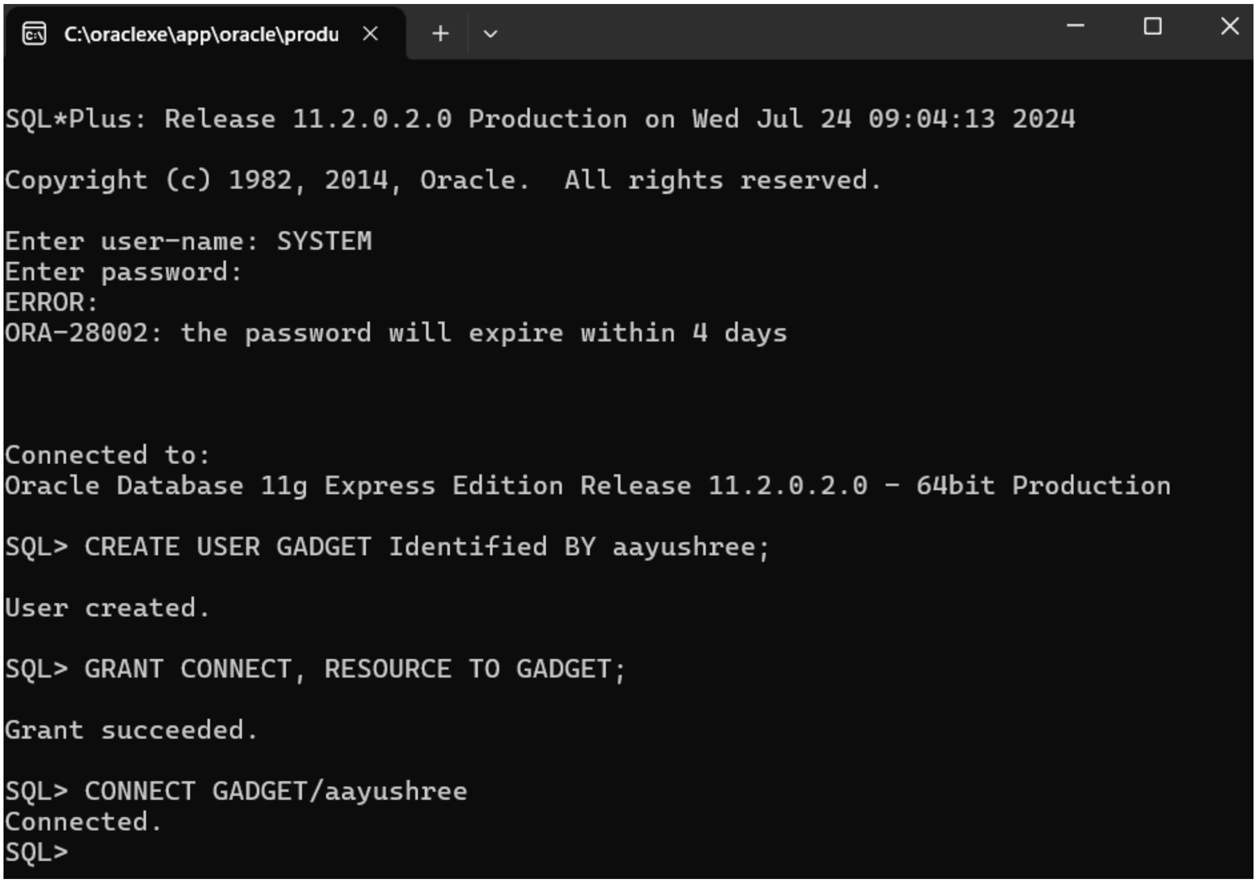
Figure 2: Final ERD.

5. Database Implementation

Following the normalization process, implementing the normalization in Oracle database involves creating tables, inserting values and establishing relationships using foreign keys (FK). It outlines the steps taken to implement the normalized tables in Oracle database.

5.1 Creating and Inserting values in the Table

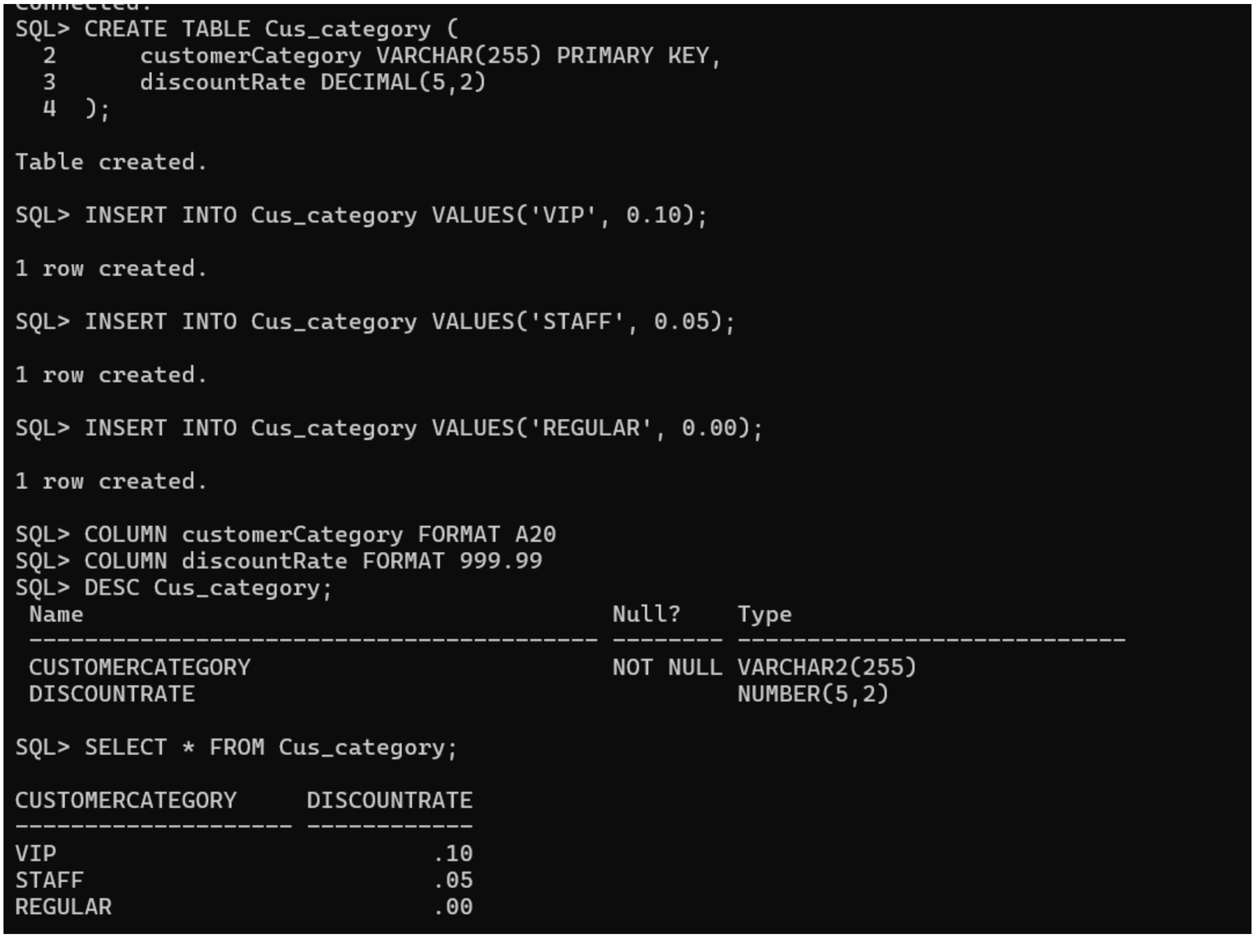
- **Creating a User and Give permission**.



```
SQL*Plus: Release 11.2.0.2.0 Production on Wed Jul 24 09:04:13 2024

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

Enter user-name: SYSTEM
Enter password:
ERROR:
ORA-28002: the password will expire within 4 days



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

SQL> CREATE USER GADGET Identified BY aayushree;

User created.

SQL> GRANT CONNECT, RESOURCE TO GADGET;

Grant succeeded.

SQL> CONNECT GADGET/aayushree
Connected.
SQL>
```

Figure 3: Creating a User and Give permission.

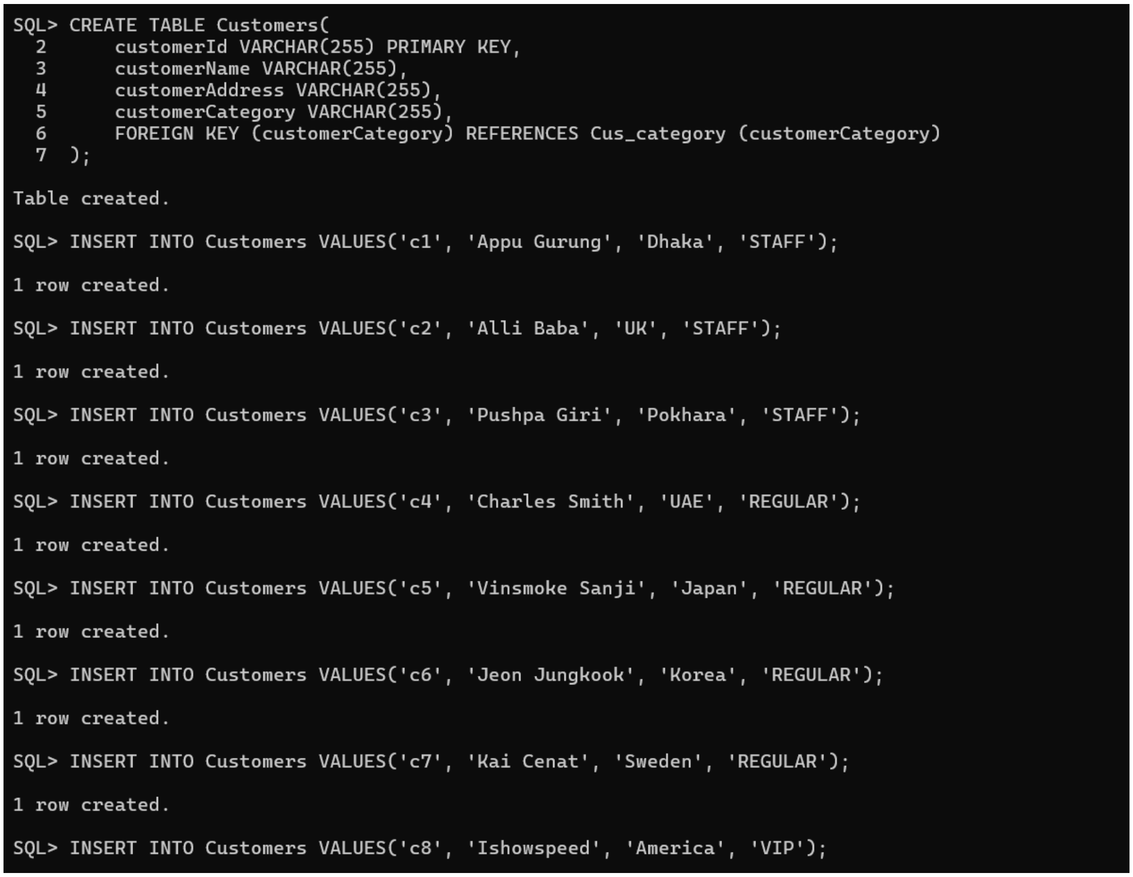
- **For Cus_category Table**.



```
Connected.
SQL> CREATE TABLE Cus_category (
  2      customerCategory VARCHAR(255) PRIMARY KEY,
  3      discountRate DECIMAL(5,2)
  4  );

Table created.

SQL> INSERT INTO Cus_category VALUES('VIP', 0.10);

1 row created.

SQL> INSERT INTO Cus_category VALUES('STAFF', 0.05);

1 row created.

SQL> INSERT INTO Cus_category VALUES('REGULAR', 0.00);

1 row created.

SQL> COLUMN customerCategory FORMAT A20
SQL> COLUMN discountRate FORMAT 999.99
SQL> DESC Cus_category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMERCATEGORY                          NOT NULL VARCHAR2(255)
 DISCOUNTRATE                                       NUMBER(5,2)

SQL> SELECT * FROM Cus_category;

CUSTOMERCATEGORY     DISCOUNTRATE
-------------------- ------------
VIP                           .10
STAFF                         .05
REGULAR                       .00
```

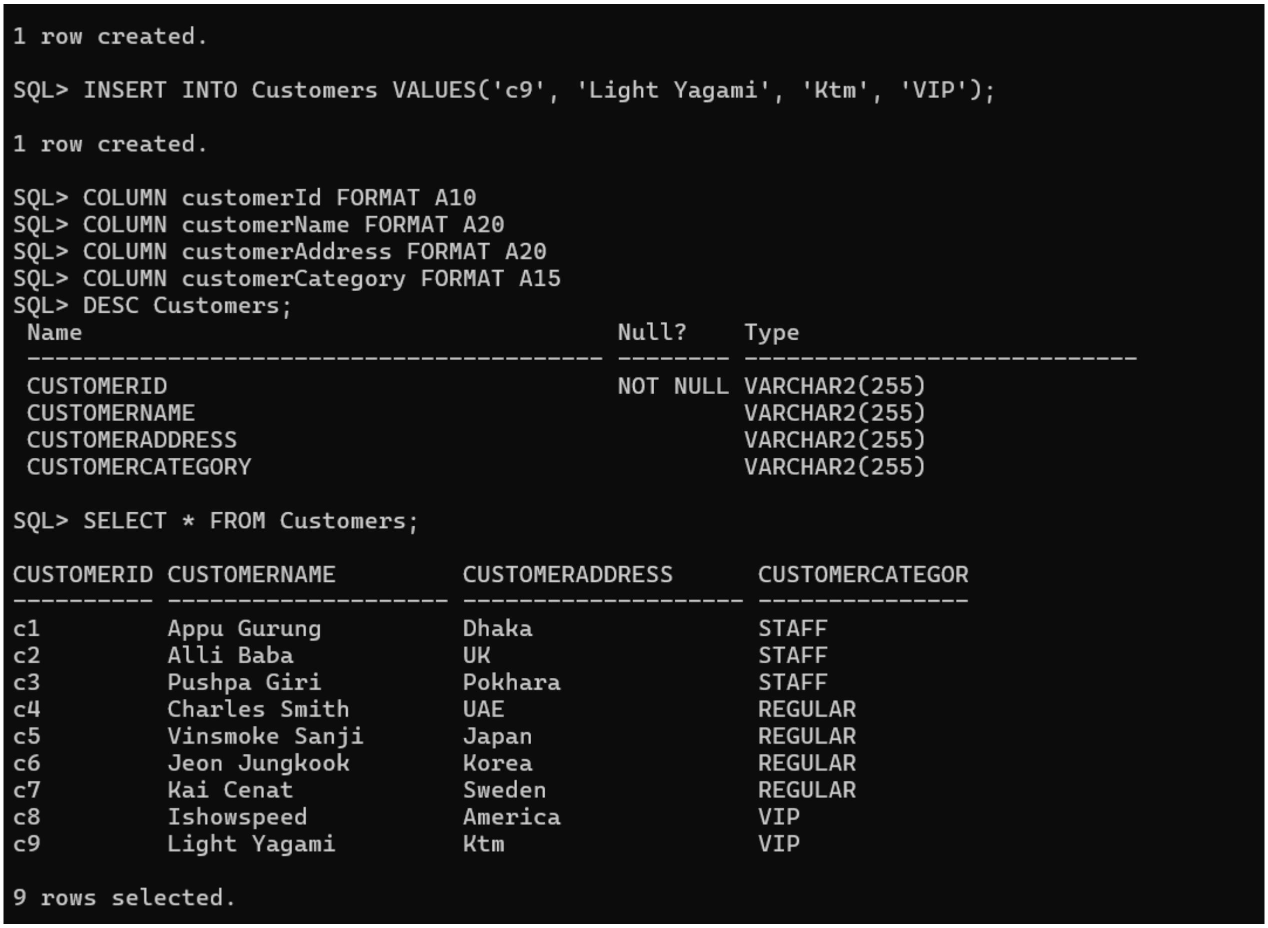
Figure 4: Creating and Inserting display values in Cus_category Table.

- **For Customers Table.**



```
SQL> CREATE TABLE Customers(
         customerId VARCHAR(255) PRIMARY KEY,
         customerName VARCHAR(255),
         customerAddress VARCHAR(255),
         customerCategory VARCHAR(255),
         FOREIGN KEY (customerCategory) REFERENCES Cus_category (customerCategory)
     );

Table created.

SQL> INSERT INTO Customers VALUES('c1', 'Appu Gurung', 'Dhaka', 'STAFF');

1 row created.

SQL> INSERT INTO Customers VALUES('c2', 'Alli Baba', 'UK', 'STAFF');

1 row created.

SQL> INSERT INTO Customers VALUES('c3', 'Pushpa Giri', 'Pokhara', 'STAFF');

1 row created.

SQL> INSERT INTO Customers VALUES('c4', 'Charles Smith', 'UAE', 'REGULAR');

1 row created.

SQL> INSERT INTO Customers VALUES('c5', 'Vinsmoke Sanji', 'Japan', 'REGULAR');

1 row created.

SQL> INSERT INTO Customers VALUES('c6', 'Jeon Jungkook', 'Korea', 'REGULAR');

1 row created.

SQL> INSERT INTO Customers VALUES('c7', 'Kai Cenat', 'Sweden', 'REGULAR');

1 row created.

SQL> INSERT INTO Customers VALUES('c8', 'Ishowspeed', 'America', 'VIP');
```

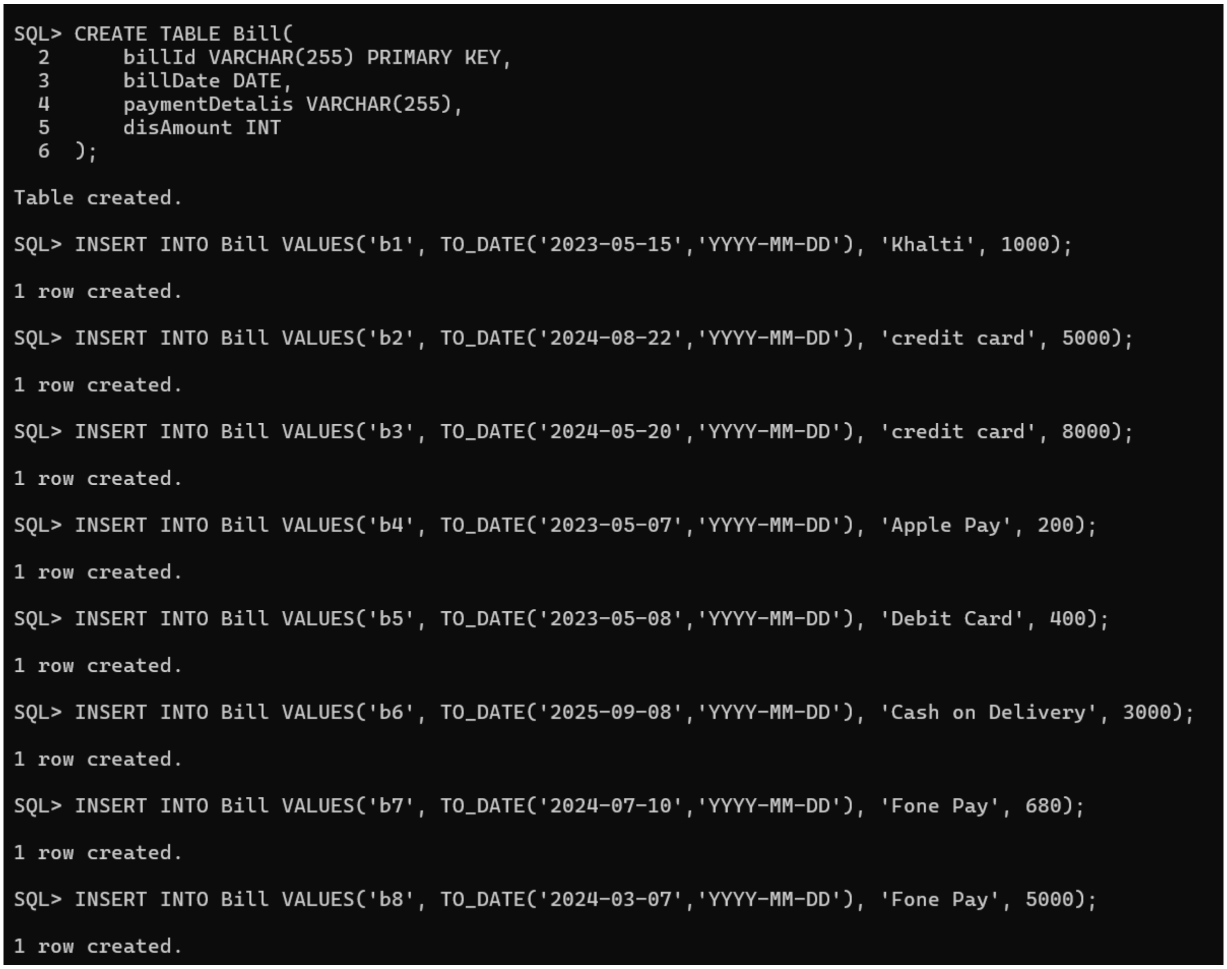
Figure 5: Creating and Inserting values in Customers Table.



```
1 row created.

SQL> INSERT INTO Customers VALUES('c9', 'Light Yagami', 'Ktm', 'VIP');

1 row created.

SQL> COLUMN customerId FORMAT A10
SQL> COLUMN customerName FORMAT A20
SQL> COLUMN customerAddress FORMAT A20
SQL> COLUMN customerCategory FORMAT A15
SQL> DESC Customers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMERID                                NOT NULL VARCHAR2(255)
 CUSTOMERNAME                                       VARCHAR2(255)
 CUSTOMERADDRESS                                    VARCHAR2(255)
 CUSTOMERCATEGORY                                   VARCHAR2(255)

SQL> SELECT * FROM Customers;

CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS      CUSTOMERCATEGOR
---------- -------------------- -------------------- ---------------
c1         Appu Gurung          Dhaka                STAFF
c2         Alli Baba            UK                   STAFF
c3         Pushpa Giri          Pokhara              STAFF
c4         Charles Smith        UAE                  REGULAR
c5         Vinsmoke Sanji       Japan                REGULAR
c6         Jeon Jungkook        Korea                REGULAR
c7         Kai Cenat            Sweden               REGULAR
c8         Ishowspeed           America              VIP
c9         Light Yagami         Ktm                  VIP

9 rows selected.
```

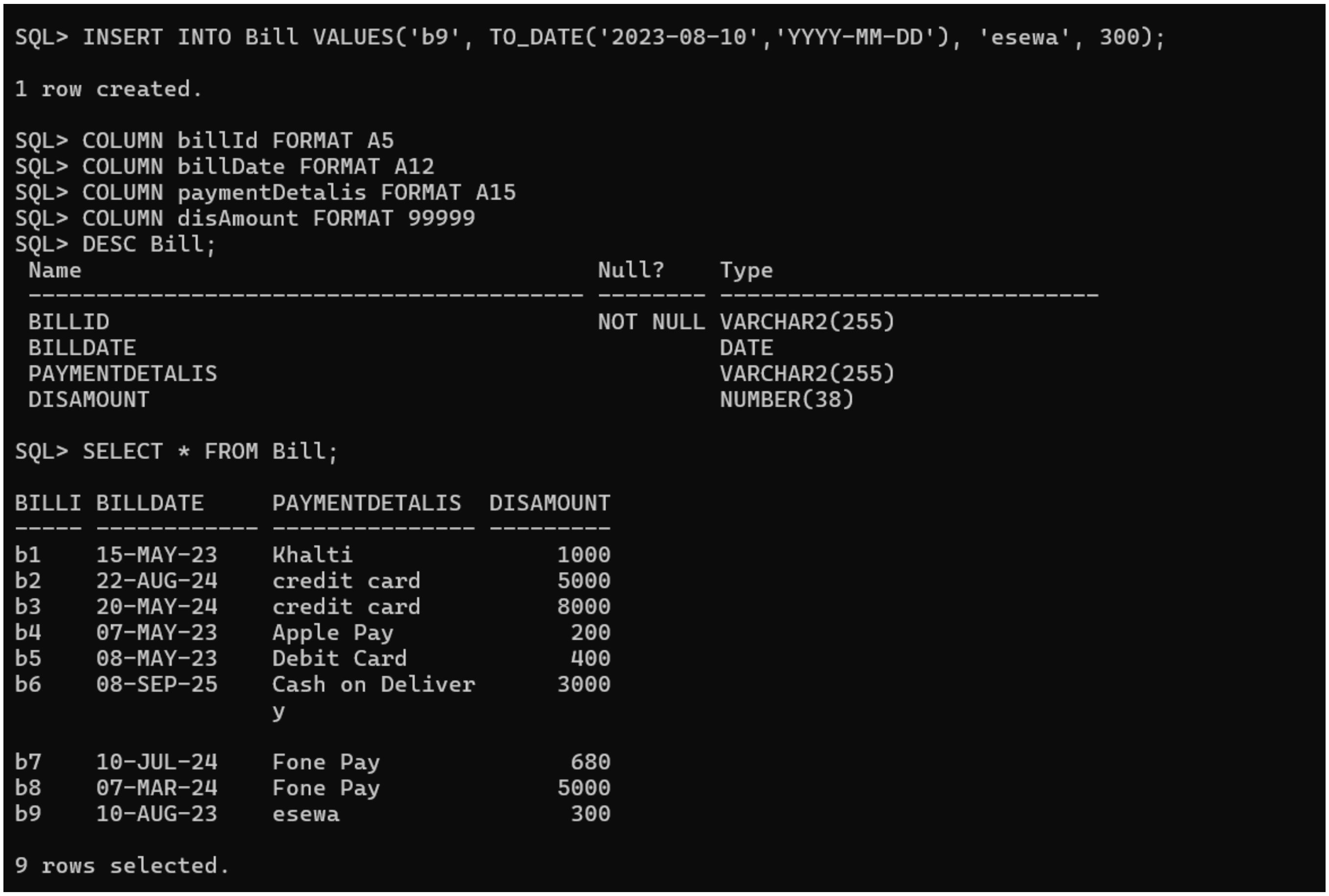
Figure 6: Displaying values of Customers Table.

- **For Bill Table.**



```
SQL> CREATE TABLE Bill(
         billId VARCHAR(255) PRIMARY KEY,
         billDate DATE,
         paymentDetalis VARCHAR(255),
         disAmount INT
     );

Table created.

SQL> INSERT INTO Bill VALUES('b1', TO_DATE('2023-05-15','YYYY-MM-DD'), 'Khalti', 1000);

1 row created.

SQL> INSERT INTO Bill VALUES('b2', TO_DATE('2024-08-22','YYYY-MM-DD'), 'credit card', 5000);

1 row created.

SQL> INSERT INTO Bill VALUES('b3', TO_DATE('2024-05-20','YYYY-MM-DD'), 'credit card', 8000);

1 row created.

SQL> INSERT INTO Bill VALUES('b4', TO_DATE('2023-05-07','YYYY-MM-DD'), 'Apple Pay', 200);

1 row created.

SQL> INSERT INTO Bill VALUES('b5', TO_DATE('2023-05-08','YYYY-MM-DD'), 'Debit Card', 400);

1 row created.

SQL> INSERT INTO Bill VALUES('b6', TO_DATE('2025-09-08','YYYY-MM-DD'), 'Cash on Delivery', 3000);

1 row created.

SQL> INSERT INTO Bill VALUES('b7', TO_DATE('2024-07-10','YYYY-MM-DD'), 'Fone Pay', 680);

1 row created.

SQL> INSERT INTO Bill VALUES('b8', TO_DATE('2024-03-07','YYYY-MM-DD'), 'Fone Pay', 5000);

1 row created.
```

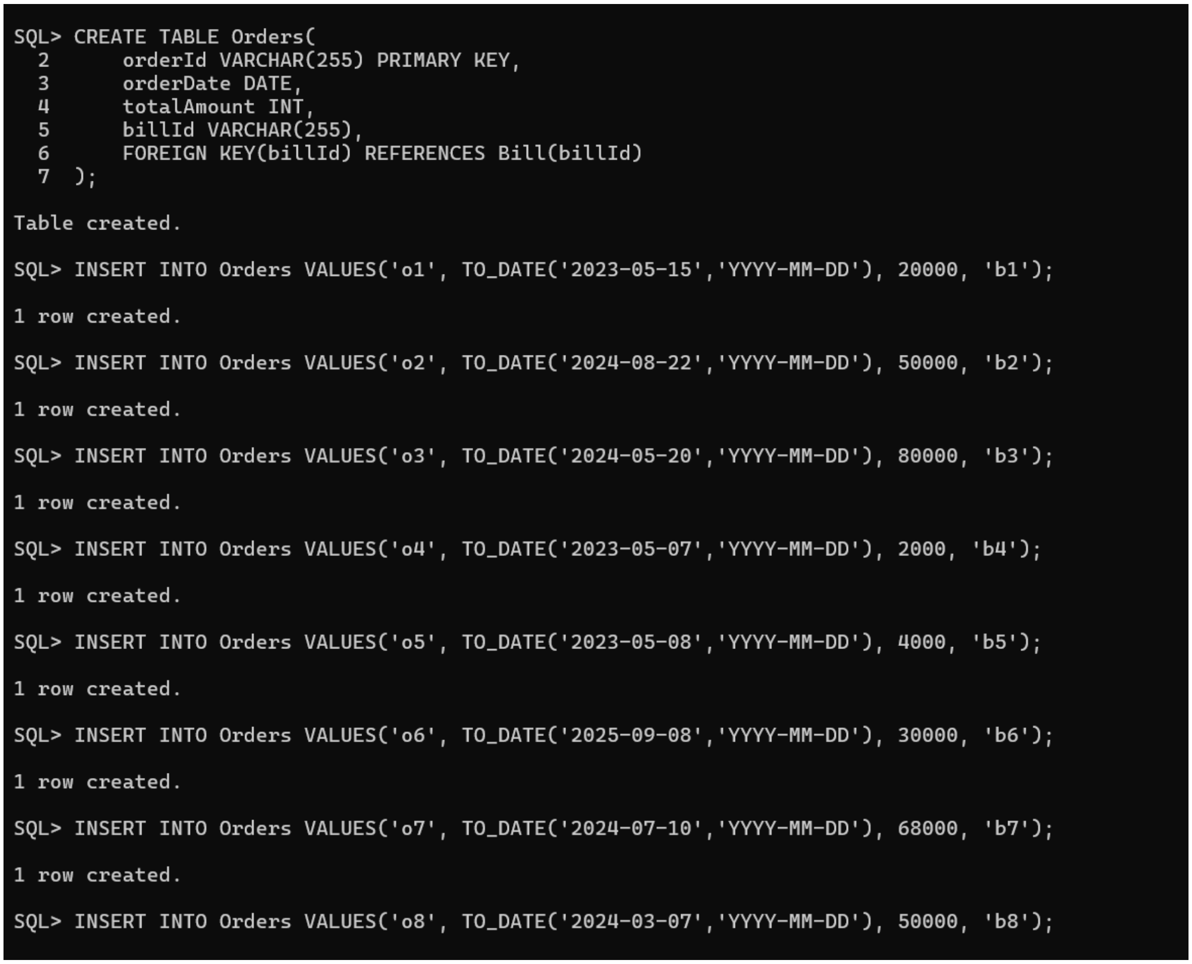
Figure 7: Creating and Inserting values in Bill Table.



```
SQL> INSERT INTO Bill VALUES('b9', TO_DATE('2023-08-10','YYYY-MM-DD'), 'esewa', 300);

1 row created.

SQL> COLUMN billId FORMAT A5
SQL> COLUMN billDate FORMAT A12
SQL> COLUMN paymentDetalis FORMAT A15
SQL> COLUMN disAmount FORMAT 99999
SQL> DESC Bill;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BILLID                                    NOT NULL VARCHAR2(255)
 BILLDATE                                           DATE
 PAYMENTDETALIS                                     VARCHAR2(255)
 DISAMOUNT                                          NUMBER(38)

SQL> SELECT * FROM Bill;

BILLI BILLDATE     PAYMENTDETALIS  DISAMOUNT
----- ------------ --------------- ---------
b1    15-MAY-23    Khalti               1000
b2    22-AUG-24    credit card          5000
b3    20-MAY-24    credit card          8000
b4    07-MAY-23    Apple Pay             200
b5    08-MAY-23    Debit Card            400
b6    08-SEP-25    Cash on Deliver      3000
                   y

b7    10-JUL-24    Fone Pay              680
b8    07-MAR-24    Fone Pay             5000
b9    10-AUG-23    esewa                 300

9 rows selected.
```

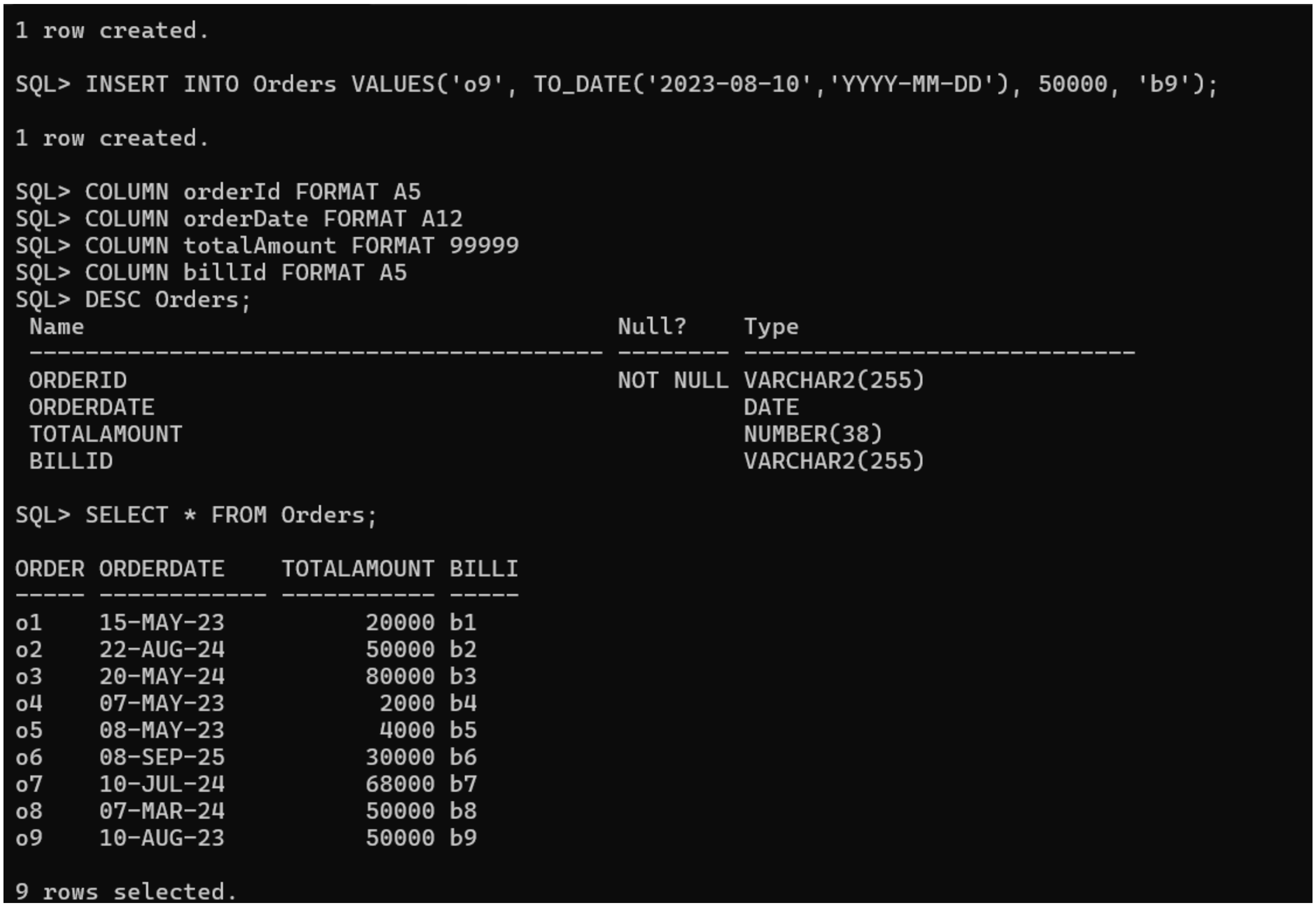
Figure 8: Displaying values of Bill Table.

- **For Orders Table.**



```
SQL> CREATE TABLE Orders(
        orderId VARCHAR(255) PRIMARY KEY,
        orderDate DATE,
        totalAmount INT,
        billId VARCHAR(255),
        FOREIGN KEY(billId) REFERENCES Bill(billId)
     );

Table created.

SQL> INSERT INTO Orders VALUES('o1', TO_DATE('2023-05-15','YYYY-MM-DD'), 20000, 'b1');

1 row created.

SQL> INSERT INTO Orders VALUES('o2', TO_DATE('2024-08-22','YYYY-MM-DD'), 50000, 'b2');

1 row created.

SQL> INSERT INTO Orders VALUES('o3', TO_DATE('2024-05-20','YYYY-MM-DD'), 80000, 'b3');

1 row created.

SQL> INSERT INTO Orders VALUES('o4', TO_DATE('2023-05-07','YYYY-MM-DD'), 2000, 'b4');

1 row created.

SQL> INSERT INTO Orders VALUES('o5', TO_DATE('2023-05-08','YYYY-MM-DD'), 4000, 'b5');

1 row created.

SQL> INSERT INTO Orders VALUES('o6', TO_DATE('2025-09-08','YYYY-MM-DD'), 30000, 'b6');

1 row created.

SQL> INSERT INTO Orders VALUES('o7', TO_DATE('2024-07-10','YYYY-MM-DD'), 68000, 'b7');

1 row created.

SQL> INSERT INTO Orders VALUES('o8', TO_DATE('2024-03-07','YYYY-MM-DD'), 50000, 'b8');
```

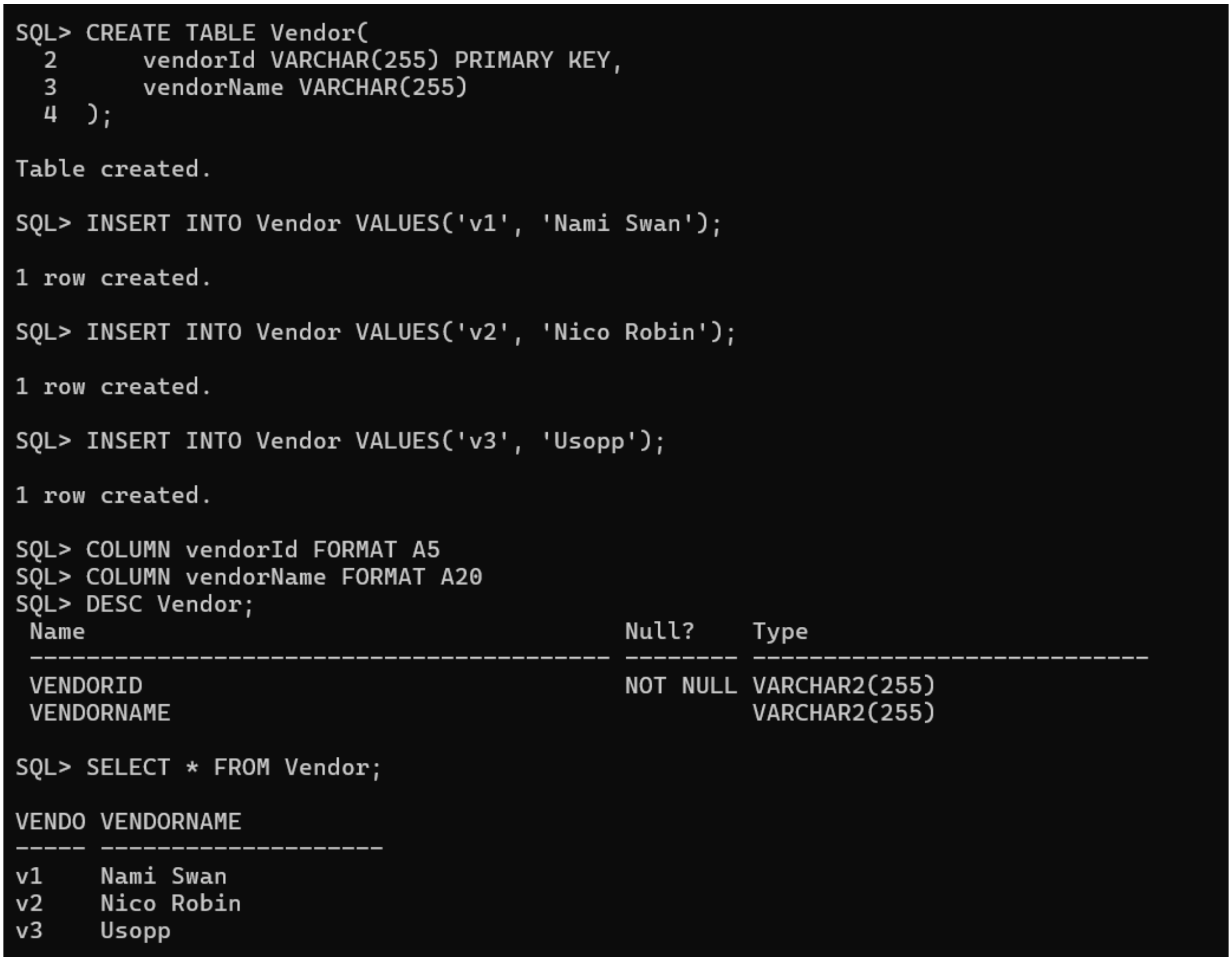
Figure 9: Creating and Inserting values in Orders Table.



```
1 row created.

SQL> INSERT INTO Orders VALUES('o9', TO_DATE('2023-08-10','YYYY-MM-DD'), 50000, 'b9');

1 row created.

SQL> COLUMN orderId FORMAT A5
SQL> COLUMN orderDate FORMAT A12
SQL> COLUMN totalAmount FORMAT 99999
SQL> COLUMN billId FORMAT A5
SQL> DESC Orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDERID                                   NOT NULL VARCHAR2(255)
 ORDERDATE                                          DATE
 TOTALAMOUNT                                        NUMBER(38)
 BILLID                                             VARCHAR2(255)

SQL> SELECT * FROM Orders;

ORDER ORDERDATE    TOTALAMOUNT BILLI
----- ------------ ----------- -----
o1    15-MAY-23          20000 b1
o2    22-AUG-24          50000 b2
o3    20-MAY-24          80000 b3
o4    07-MAY-23           2000 b4
o5    08-MAY-23           4000 b5
o6    08-SEP-25          30000 b6
o7    10-JUL-24          68000 b7
o8    07-MAR-24          50000 b8
o9    10-AUG-23          50000 b9

9 rows selected.
```

Figure 10: Displaying values of Orders Table.

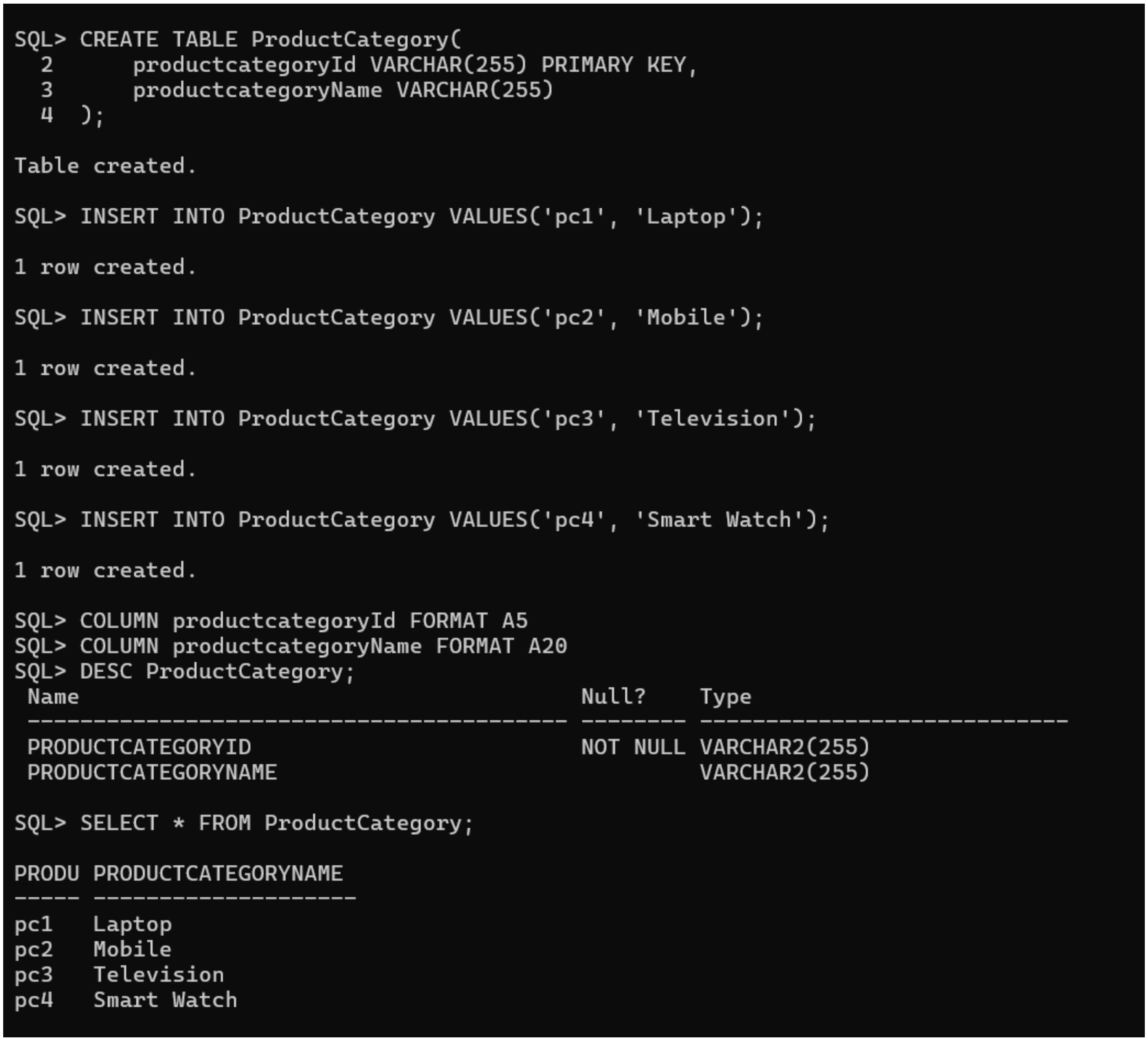
- **For Vendor Table**



```
SQL> CREATE TABLE Vendor(
  2      vendorId VARCHAR(255) PRIMARY KEY,
  3      vendorName VARCHAR(255)
  4  );

Table created.

SQL> INSERT INTO Vendor VALUES('v1', 'Nami Swan');

1 row created.

SQL> INSERT INTO Vendor VALUES('v2', 'Nico Robin');

1 row created.

SQL> INSERT INTO Vendor VALUES('v3', 'Usopp');

1 row created.

SQL> COLUMN vendorId FORMAT A5
SQL> COLUMN vendorName FORMAT A20
SQL> DESC Vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VENDORID                                  NOT NULL VARCHAR2(255)
 VENDORNAME                                         VARCHAR2(255)

SQL> SELECT * FROM Vendor;

VENDO VENDORNAME
----- --------------------
v1    Nami Swan
v2    Nico Robin
v3    Usopp
```

Figure 11: Creating and Inserting display values in Vendor Table.

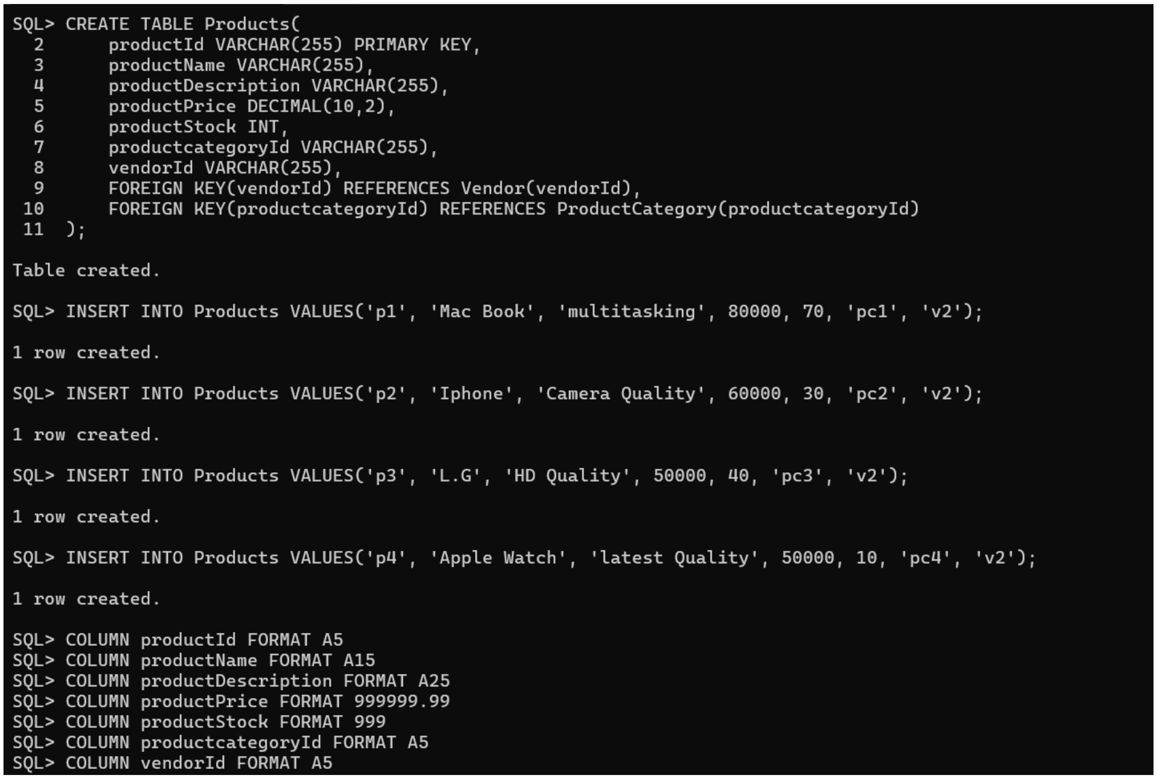
- **For ProductCategory Table**



```
SQL> CREATE TABLE ProductCategory(
  2      productcategoryId VARCHAR(255) PRIMARY KEY,
  3      productcategoryName VARCHAR(255)
  4  );

Table created.

SQL> INSERT INTO ProductCategory VALUES('pc1', 'Laptop');

1 row created.

SQL> INSERT INTO ProductCategory VALUES('pc2', 'Mobile');

1 row created.

SQL> INSERT INTO ProductCategory VALUES('pc3', 'Television');

1 row created.

SQL> INSERT INTO ProductCategory VALUES('pc4', 'Smart Watch');

1 row created.

SQL> COLUMN productcategoryId FORMAT A5
SQL> COLUMN productcategoryName FORMAT A20
SQL> DESC ProductCategory;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCTCATEGORYID                         NOT NULL VARCHAR2(255)
 PRODUCTCATEGORYNAME                                VARCHAR2(255)

SQL> SELECT * FROM ProductCategory;

PRODU PRODUCTCATEGORYNAME
----- --------------------
pc1   Laptop
pc2   Mobile
pc3   Television
pc4   Smart Watch
```

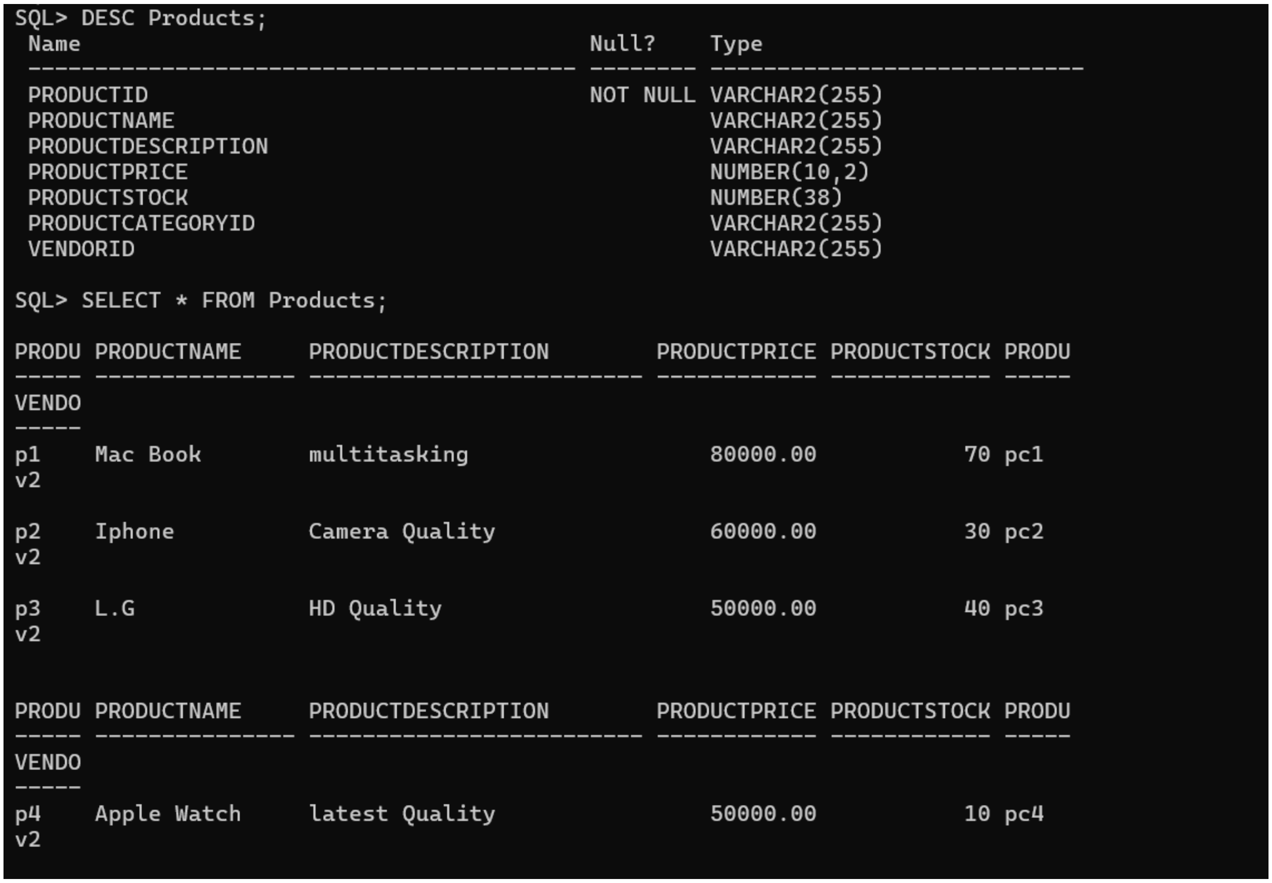
Figure 12: Creating and Inserting display values in ProductCategory Table.

- **For Products Table**



```
SQL> CREATE TABLE Products(
         productId VARCHAR(255) PRIMARY KEY,
         productName VARCHAR(255),
         productDescription VARCHAR(255),
         productPrice DECIMAL(10,2),
         productStock INT,
         productcategoryId VARCHAR(255),
         vendorId VARCHAR(255),
         FOREIGN KEY(vendorId) REFERENCES Vendor(vendorId),
         FOREIGN KEY(productcategoryId) REFERENCES ProductCategory(productcategoryId)
     );

Table created.

SQL> INSERT INTO Products VALUES('p1', 'Mac Book', 'multitasking', 80000, 70, 'pc1', 'v2');

1 row created.

SQL> INSERT INTO Products VALUES('p2', 'Iphone', 'Camera Quality', 60000, 30, 'pc2', 'v2');

1 row created.

SQL> INSERT INTO Products VALUES('p3', 'L.G', 'HD Quality', 50000, 40, 'pc3', 'v2');

1 row created.

SQL> INSERT INTO Products VALUES('p4', 'Apple Watch', 'latest Quality', 50000, 10, 'pc4', 'v2');

1 row created.

SQL> COLUMN productId FORMAT A5
SQL> COLUMN productName FORMAT A15
SQL> COLUMN productDescription FORMAT A25
SQL> COLUMN productPrice FORMAT 999999.99
SQL> COLUMN productStock FORMAT 999
SQL> COLUMN productcategoryId FORMAT A5
SQL> COLUMN vendorId FORMAT A5
```

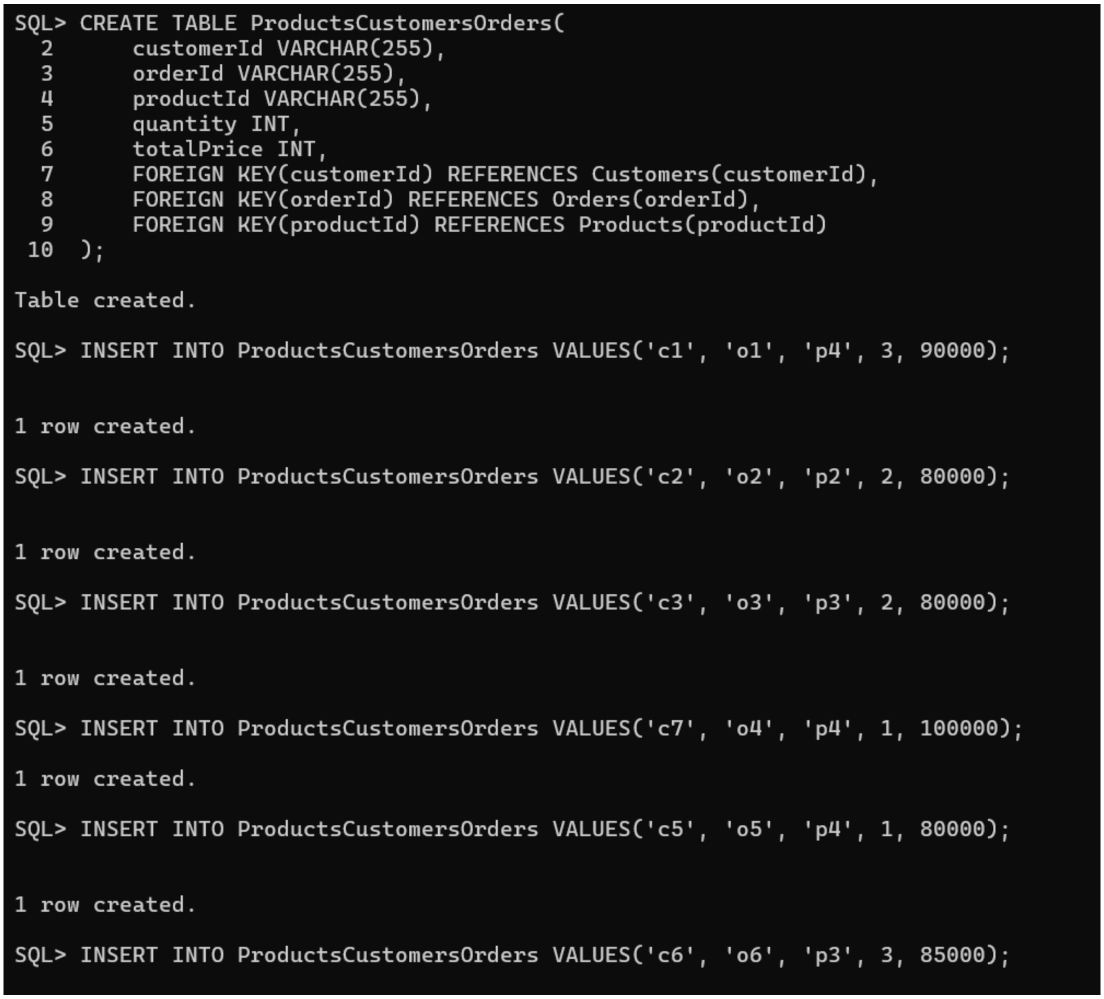
Figure 13: Creating and Inserting values in Products Table.



```
SQL> DESC Products;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCTID                                 NOT NULL VARCHAR2(255)
 PRODUCTNAME                                        VARCHAR2(255)
 PRODUCTDESCRIPTION                                 VARCHAR2(255)
 PRODUCTPRICE                                       NUMBER(10,2)
 PRODUCTSTOCK                                       NUMBER(38)
 PRODUCTCATEGORYID                                  VARCHAR2(255)
 VENDORID                                           VARCHAR2(255)

SQL> SELECT * FROM Products;

PRODU PRODUCTNAME     PRODUCTDESCRIPTION        PRODUCTPRICE PRODUCTSTOCK PRODU
----- --------------- ------------------------- ------------ ------------ -----
VENDO
-----
p1    Mac Book        multitasking                  80000.00           70 pc1
v2

p2    Iphone          Camera Quality                60000.00           30 pc2
v2

p3    L.G             HD Quality                    50000.00           40 pc3
v2


PRODU PRODUCTNAME     PRODUCTDESCRIPTION        PRODUCTPRICE PRODUCTSTOCK PRODU
----- --------------- ------------------------- ------------ ------------ -----
VENDO
-----
p4    Apple Watch     latest Quality                50000.00           10 pc4
v2
```

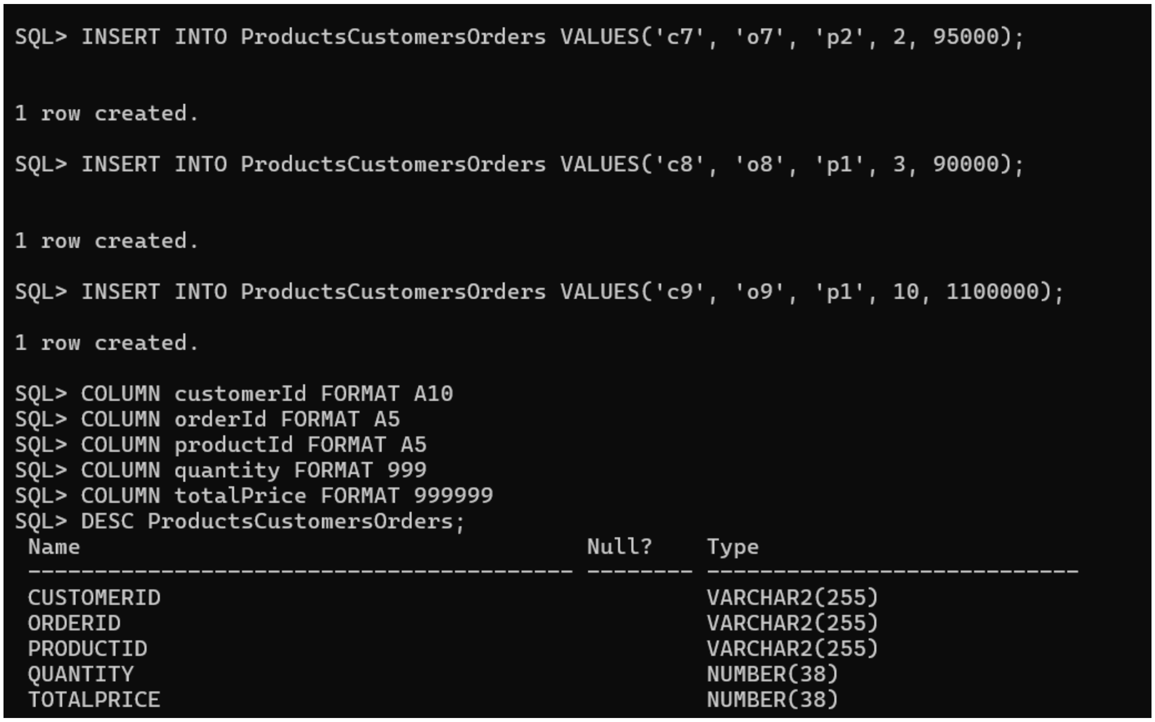
Figure 14: Displaying values of Products Table.

- **For ProductsCustomersOrders Table**



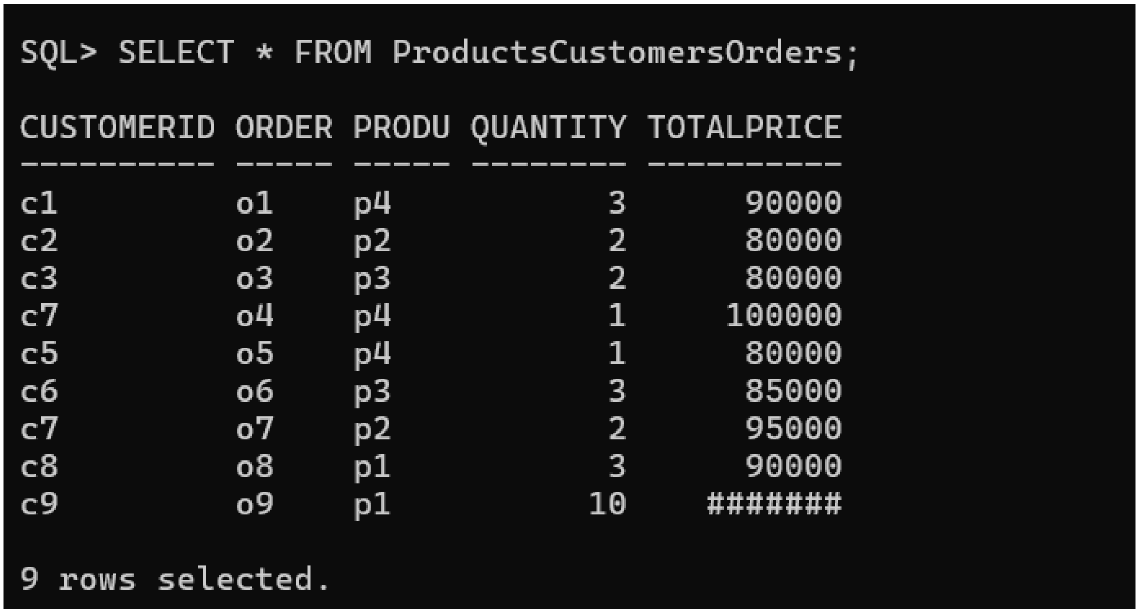
```
SQL> CREATE TABLE ProductsCustomersOrders(
         customerId VARCHAR(255),
         orderId VARCHAR(255),
         productId VARCHAR(255),
         quantity INT,
         totalPrice INT,
         FOREIGN KEY(customerId) REFERENCES Customers(customerId),
         FOREIGN KEY(orderId) REFERENCES Orders(orderId),
         FOREIGN KEY(productId) REFERENCES Products(productId)
     );

Table created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c1', 'o1', 'p4', 3, 90000);


1 row created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c2', 'o2', 'p2', 2, 80000);


1 row created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c3', 'o3', 'p3', 2, 80000);


1 row created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c7', 'o4', 'p4', 1, 100000);

1 row created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c5', 'o5', 'p4', 1, 80000);


1 row created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c6', 'o6', 'p3', 3, 85000);
```

Figure 15: Creating and Inserting values in ProductsCustomersOrders Table.



```
SQL> INSERT INTO ProductsCustomersOrders VALUES('c7', 'o7', 'p2', 2, 95000);


1 row created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c8', 'o8', 'p1', 3, 90000);


1 row created.

SQL> INSERT INTO ProductsCustomersOrders VALUES('c9', 'o9', 'p1', 10, 1100000);

1 row created.

SQL> COLUMN customerId FORMAT A10
SQL> COLUMN orderId FORMAT A5
SQL> COLUMN productId FORMAT A5
SQL> COLUMN quantity FORMAT 999
SQL> COLUMN totalPrice FORMAT 999999
SQL> DESC ProductsCustomersOrders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMERID                                         VARCHAR2(255)
 ORDERID                                            VARCHAR2(255)
 PRODUCTID                                          VARCHAR2(255)
 QUANTITY                                           NUMBER(38)
 TOTALPRICE                                         NUMBER(38)
```

Figure 16: Displaying values of ProductsCustomersOrders Table.



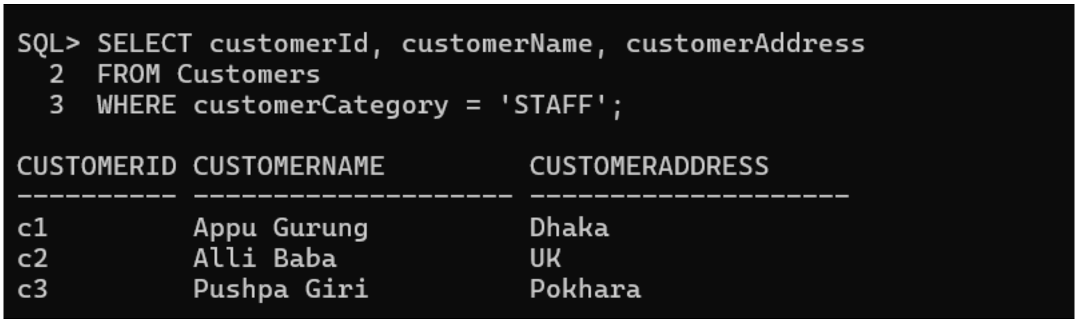
```
SQL> SELECT * FROM ProductsCustomersOrders;

CUSTOMERID ORDER PRODU QUANTITY TOTALPRICE
---------- ----- ----- -------- ----------
c1         o1    p4           3      90000
c2         o2    p2           2      80000
c3         o3    p3           2      80000
c7         o4    p4           1     100000
c5         o5    p4           1      80000
c6         o6    p3           3      85000
c7         o7    p2           2      95000
c8         o8    p1           3      90000
c9         o9    p1          10    #######

9 rows selected.
```

Figure 17: Displaying inserting values of ProductsCustomersOrders Table.

6. Database Querying

6.1 Information query

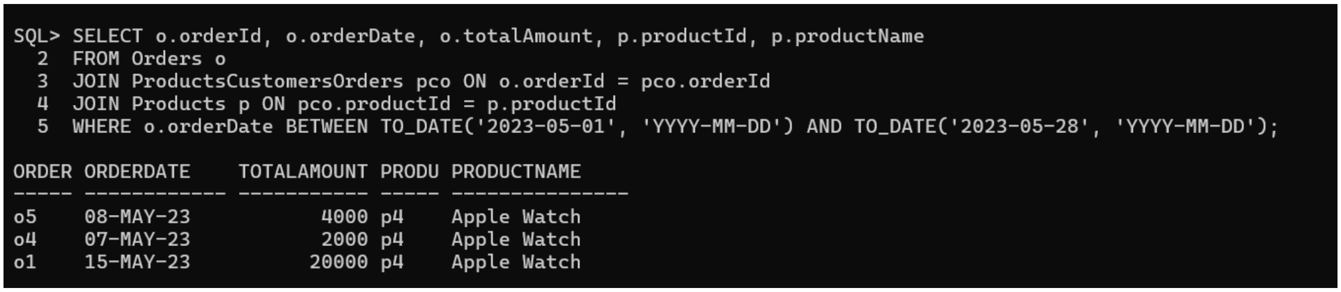
1. List all the customers that are also staff of the company.



```
SQL> SELECT customerId, customerName, customerAddress
     FROM Customers
     WHERE customerCategory = 'STAFF';

CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS
---------- -------------------- --------------------
c1         Appu Gurung          Dhaka
c2         Alli Baba            UK
c3         Pushpa Giri          Pokhara
```

Figure 18: Customers that are also staff of the company.

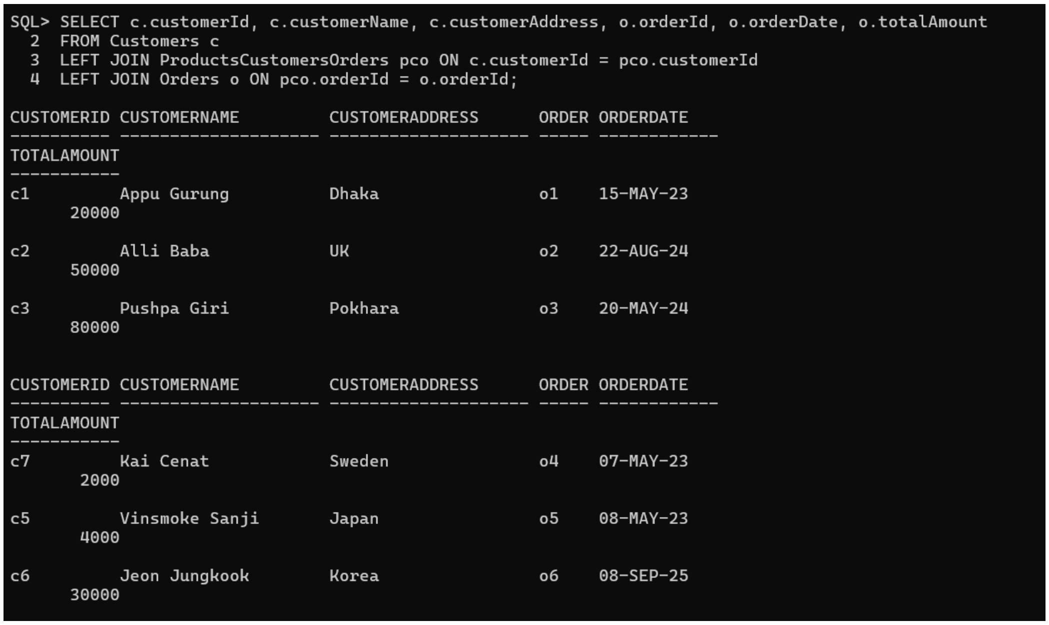
2. List all the orders made for any particular product between the dates 01-05-2023 till 28-05-2023.



```
SQL> SELECT o.orderId, o.orderDate, o.totalAmount, p.productId, p.productName
     FROM Orders o
     JOIN ProductsCustomersOrders pco ON o.orderId = pco.orderId
     JOIN Products p ON pco.productId = p.productId
     WHERE o.orderDate BETWEEN TO_DATE('2023-05-01', 'YYYY-MM-DD') AND TO_DATE('2023-05-28', 'YYYY-MM-DD');

ORDER ORDERDATE    TOTALAMOUNT PRODU PRODUCTNAME
----- ------------ ----------- ----- ---------------
o5    08-MAY-23           4000 p4    Apple Watch
o4    07-MAY-23           2000 p4    Apple Watch
o1    15-MAY-23          20000 p4    Apple Watch
```

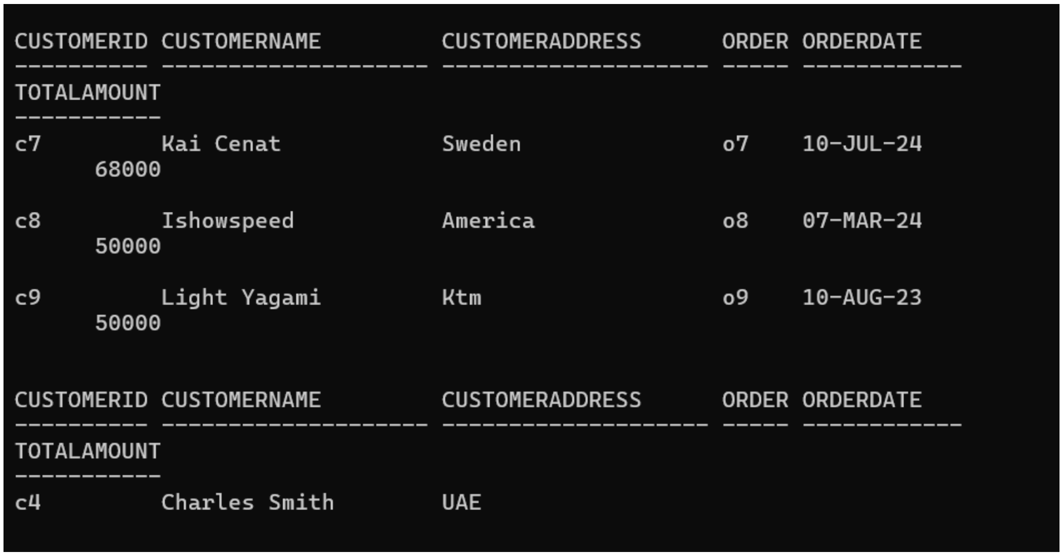
Figure 19: List all the orders particular product.

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



```
SQL> SELECT c.customerId, c.customerName, c.customerAddress, o.orderId, o.orderDate, o.totalAmount
     FROM Customers c
     LEFT JOIN ProductsCustomersOrders pco ON c.customerId = pco.customerId
     LEFT JOIN Orders o ON pco.orderId = o.orderId;

CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS      ORDER ORDERDATE
---------- -------------------- -------------------- ----- ------------
TOTALAMOUNT
-----------
c1         Appu Gurung          Dhaka                o1    15-MAY-23
      20000

c2         Alli Baba            UK                   o2    22-AUG-24
      50000

c3         Pushpa Giri          Pokhara              o3    20-MAY-24
      80000


CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS      ORDER ORDERDATE
---------- -------------------- -------------------- ----- ------------
TOTALAMOUNT
-----------
c7         Kai Cenat            Sweden               o4    07-MAY-23
       2000

c5         Vinsmoke Sanji       Japan                o5    08-MAY-23
       4000

c6         Jeon Jungkook        Korea                o6    08-SEP-25
      30000
```

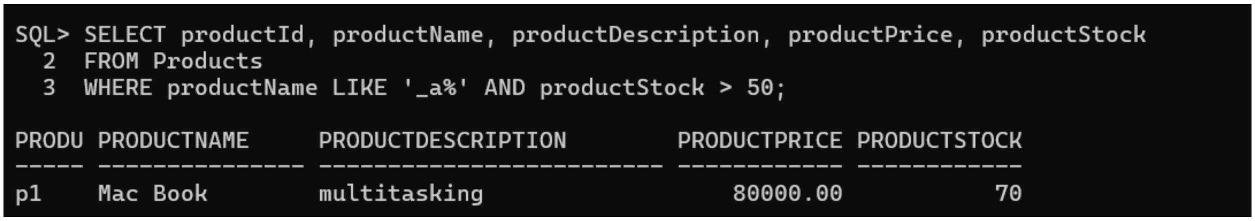
Figure 20: List all the customers with their order details and also the customers who have not ordered any products yet.



```
CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS      ORDER ORDERDATE
---------- -------------------- -------------------- ----- ------------
TOTALAMOUNT
-----------
c7         Kai Cenat            Sweden               o7    10-JUL-24
      68000

c8         Ishowspeed           America              o8    07-MAR-24
      50000

c9         Light Yagami         Ktm                  o9    10-AUG-23
      50000


CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS      ORDER ORDERDATE
---------- -------------------- -------------------- ----- ------------
TOTALAMOUNT
-----------
c4         Charles Smith        UAE
```

Figure 21: Displaying the values

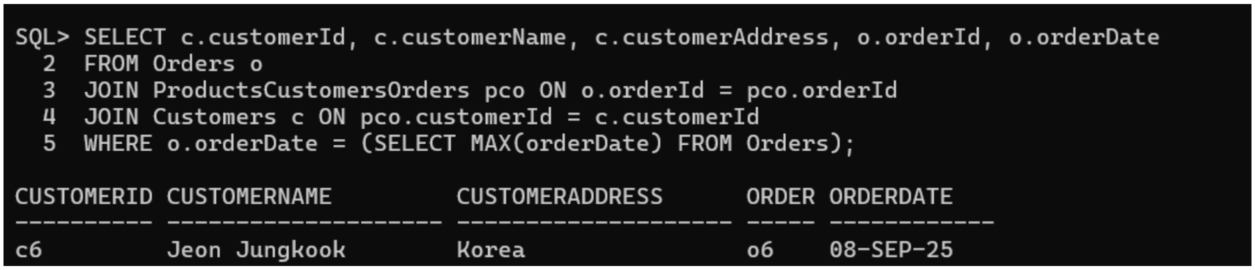
4. List all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.



```
SQL> SELECT productId, productName, productDescription, productPrice, productStock
     FROM Products
     WHERE productName LIKE '_a%' AND productStock > 50;

PRODU PRODUCTNAME     PRODUCTDESCRIPTION        PRODUCTPRICE PRODUCTSTOCK
----- --------------- ------------------------- ------------ ------------
p1    Mac Book        multitasking                  80000.00           70
```

Figure 22: List all product details that have the second letter ‘a’ in their product name

5. Find out the customer who has ordered recently.

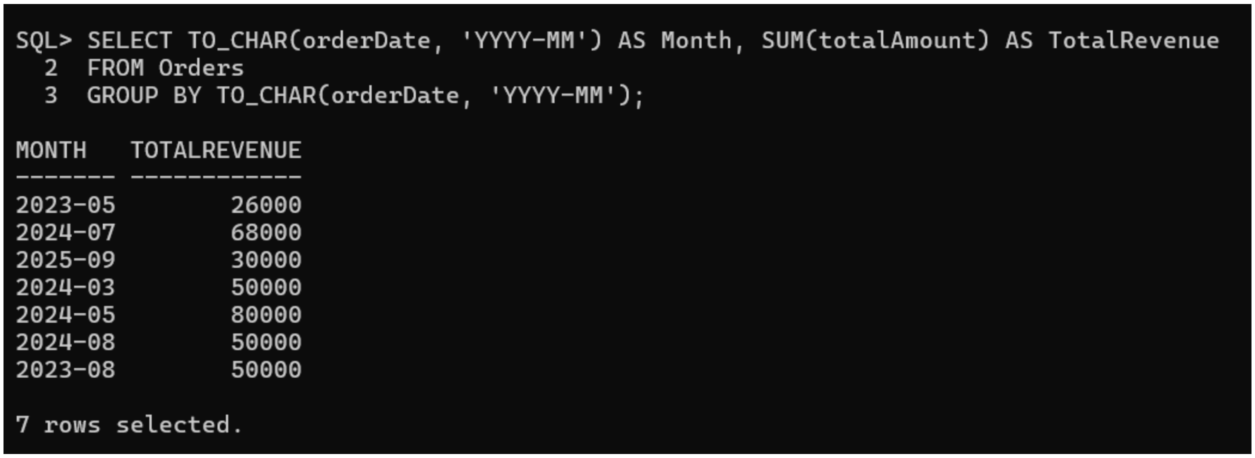


```
SQL> SELECT c.customerId, c.customerName, c.customerAddress, o.orderId, o.orderDate
     FROM Orders o
     JOIN ProductsCustomersOrders pco ON o.orderId = pco.orderId
     JOIN Customers c ON pco.customerId = c.customerId
     WHERE o.orderDate = (SELECT MAX(orderDate) FROM Orders);

CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS      ORDER ORDERDATE
---------- -------------------- -------------------- ----- ------------
c6         Jeon Jungkook        Korea                o6    08-SEP-25
```

Figure 23: Finding out the customer who has ordered recently.

6.2 Transaction Query

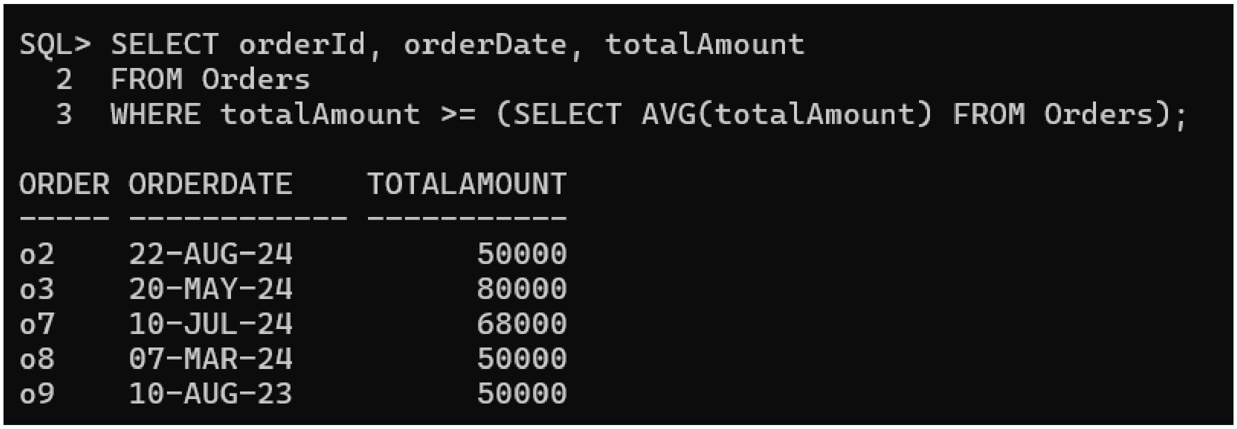
1. Show the total revenue of the company for each month.



```
SQL> SELECT TO_CHAR(orderDate, 'YYYY-MM') AS Month, SUM(totalAmount) AS TotalRevenue
     FROM Orders
     GROUP BY TO_CHAR(orderDate, 'YYYY-MM');

MONTH   TOTALREVENUE
------- ------------
2023-05        26000
2024-07        68000
2025-09        30000
2024-03        50000
2024-05        80000
2024-08        50000
2023-08        50000

7 rows selected.
```

Figure 24: Showing the total revenue of the company for each month

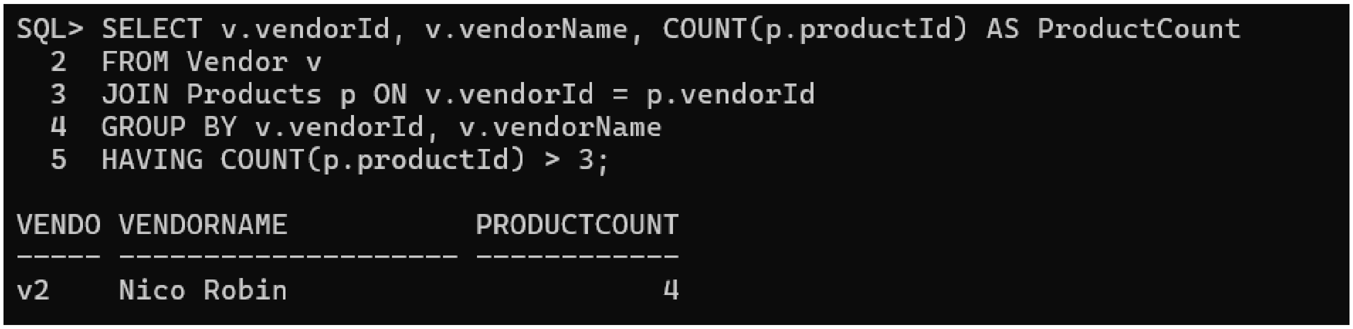
2. Find those orders that are equal or higher than the average order total value.



```
SQL> SELECT orderId, orderDate, totalAmount
     FROM Orders
     WHERE totalAmount >= (SELECT AVG(totalAmount) FROM Orders);

ORDER ORDERDATE    TOTALAMOUNT
----- ------------ -----------
o2    22-AUG-24          50000
o3    20-MAY-24          80000
o7    10-JUL-24          68000
o8    07-MAR-24          50000
o9    10-AUG-23          50000
```

Figure 25: Finding those orders that are equal or higher than the average order total value.

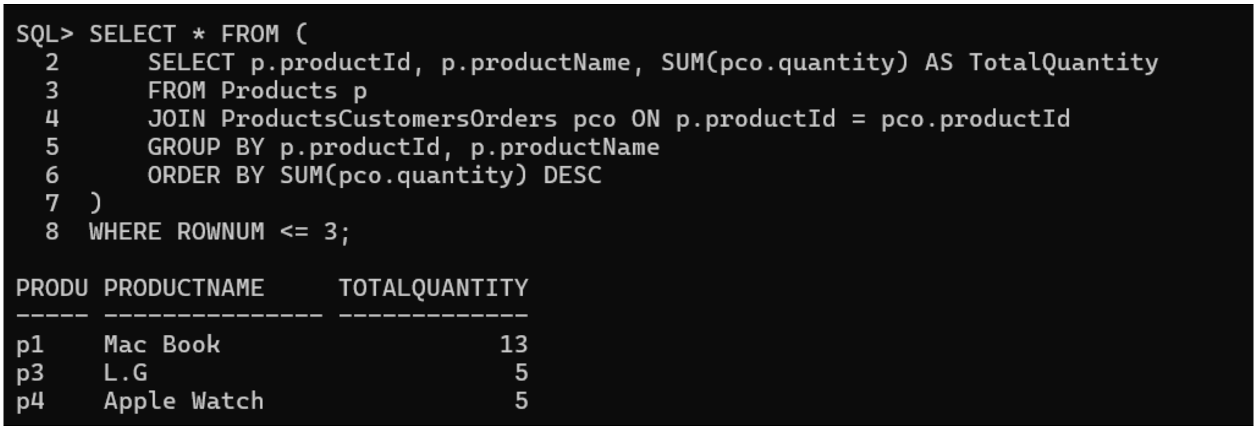
3. List the details of vendors who have supplied more than 3 products to the company.



```
SQL> SELECT v.vendorId, v.vendorName, COUNT(p.productId) AS ProductCount
     FROM Vendor v
     JOIN Products p ON v.vendorId = p.vendorId
     GROUP BY v.vendorId, v.vendorName
     HAVING COUNT(p.productId) > 3;

VENDO VENDORNAME           PRODUCTCOUNT
----- -------------------- ------------
v2    Nico Robin                      4
```

Figure 26: Listing the details of vendors who have supplied more than 3 products to the company.

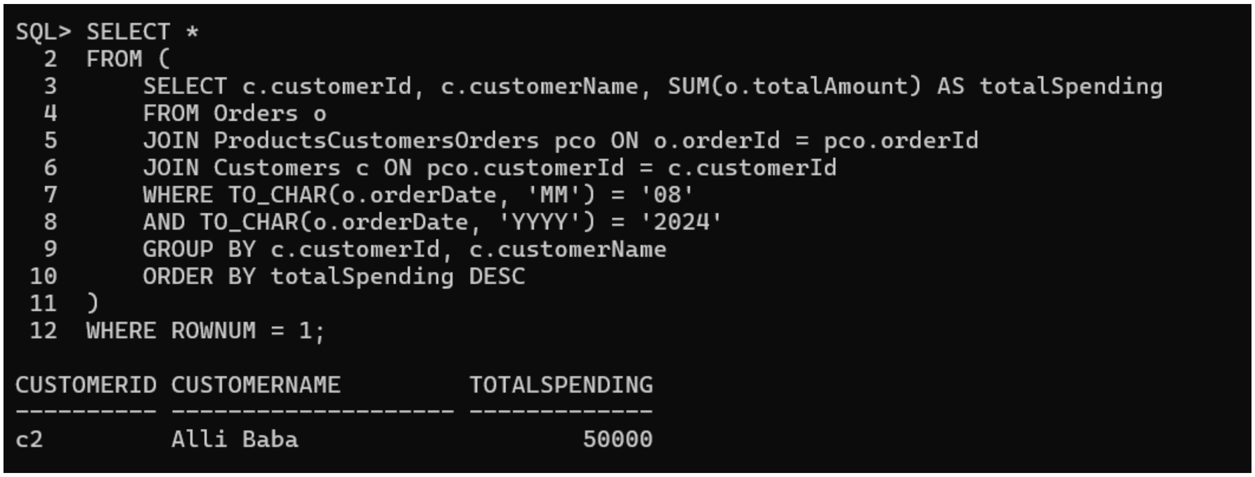
4. Show the top 3 product details that have been ordered the most.



```
SQL> SELECT * FROM (
         SELECT p.productId, p.productName, SUM(pco.quantity) AS TotalQuantity
         FROM Products p
         JOIN ProductsCustomersOrders pco ON p.productId = pco.productId
         GROUP BY p.productId, p.productName
         ORDER BY SUM(pco.quantity) DESC
     )
     WHERE ROWNUM <= 3;

PRODU PRODUCTNAME     TOTALQUANTITY
----- --------------- -------------
p1    Mac Book                   13
p3    L.G                         5
p4    Apple Watch                 5
```

Figure 27: Showing the top 3 product details that have been ordered the most.

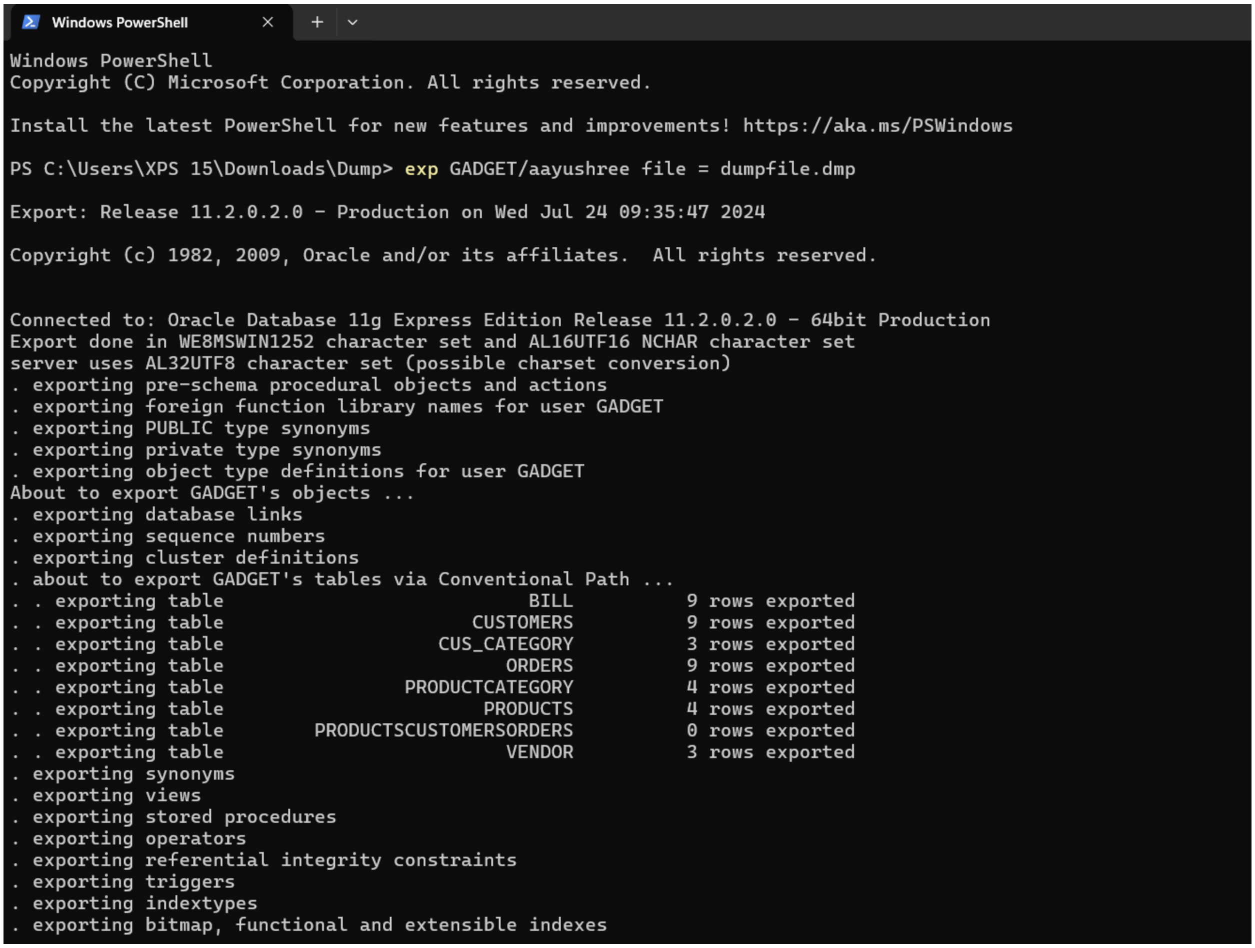
5. Find out the customer who has ordered the most in August with his/her total spending on that month.



```
SQL> SELECT *
     FROM (
         SELECT c.customerId, c.customerName, SUM(o.totalAmount) AS totalSpending
         FROM Orders o
         JOIN ProductsCustomersOrders pco ON o.orderId = pco.orderId
         JOIN Customers c ON pco.customerId = c.customerId
         WHERE TO_CHAR(o.orderDate, 'MM') = '08'
         AND TO_CHAR(o.orderDate, 'YYYY') = '2024'
         GROUP BY c.customerId, c.customerName
         ORDER BY totalSpending DESC
     )
     WHERE ROWNUM = 1;

CUSTOMERID CUSTOMERNAME         TOTALSPENDING
---------- -------------------- -------------
c2         Alli Baba                    50000
```

Figure 28: Find out the customer who has ordered the most in August with his/her total spending on that month.

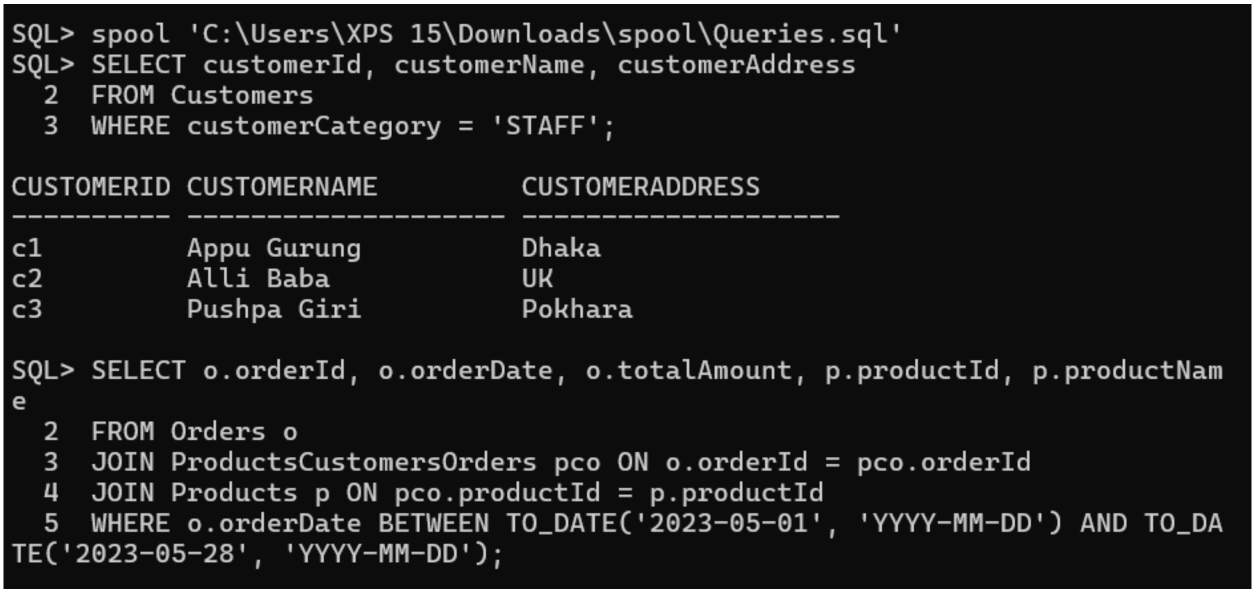
7. Backup dump file of the database



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\XPS 15\Downloads\Dump> exp GADGET/aayushree file = dumpfile.dmp

Export: Release 11.2.0.2.0 - Production on Wed Jul 24 09:35:47 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 GADGET
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user GADGET
About to export GADGET's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export GADGET's tables via Conventional Path ...
. . exporting table                           BILL          9 rows exported
. . exporting table                      CUSTOMERS          9 rows exported
. . exporting table                   CUS_CATEGORY          3 rows exported
. . exporting table                         ORDERS          9 rows exported
. . exporting table                PRODUCTCATEGORY          4 rows exported
. . exporting table                       PRODUCTS          4 rows exported
. . exporting table        PRODUCTSCUSTOMERSORDERS          0 rows exported
. . exporting table                         VENDOR          3 rows exported
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
```

Figure 29: Backup dump file of the database.

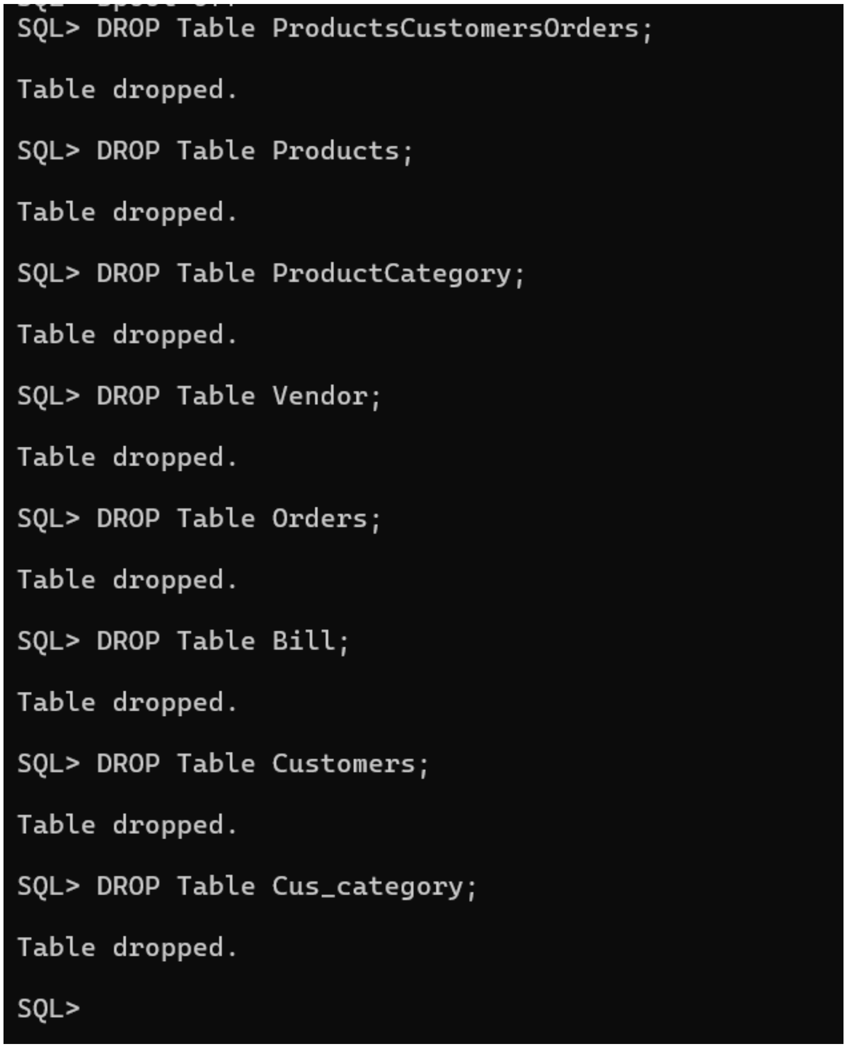
8. Spool file Creation



```
SQL> spool 'C:\Users\XPS 15\Downloads\spool\Queries.sql'
SQL> SELECT customerId, customerName, customerAddress
  2  FROM Customers
  3  WHERE customerCategory = 'STAFF';

CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS
---------- -------------------- --------------------
c1         Appu Gurung          Dhaka
c2         Alli Baba            UK
c3         Pushpa Giri          Pokhara

SQL> SELECT o.orderId, o.orderDate, o.totalAmount, p.productId, p.productNam
e
  2  FROM Orders o
  3  JOIN ProductsCustomersOrders pco ON o.orderId = pco.orderId
  4  JOIN Products p ON pco.productId = p.productId
  5  WHERE o.orderDate BETWEEN TO_DATE('2023-05-01', 'YYYY-MM-DD') AND TO_DA
TE('2023-05-28', 'YYYY-MM-DD');
```

Figure 30: Spool file Creation.

9. Dropping Tables



```
SQL> DROP Table ProductsCustomersOrders;

Table dropped.

SQL> DROP Table Products;

Table dropped.

SQL> DROP Table ProductCategory;

Table dropped.

SQL> DROP Table Vendor;

Table dropped.

SQL> DROP Table Orders;

Table dropped.

SQL> DROP Table Bill;

Table dropped.

SQL> DROP Table Customers;

Table dropped.

SQL> DROP Table Cus_category;

Table dropped.

SQL>
```

Figure 31: Dropping Table.

10. Critical Evaluation

A key component of computer science and information systems coursework is the database design module. It teaches students the fundamental knowledge and abilities expected to design, operate, and improve databases.

The coursework contains a wide variety of items, such as database design concepts, methods for data modeling, database implementation, and database administration. It connects theoretical knowledge with implementation, covering table construction, data storage, and Oracle SQL queries in detail.

The database module delivered skills in SQL development, which allow effective database connectivity and the fulfillment of third normal form, along with a variety of online tools that make it easier to create diagrams connected to databases. These qualities are requiring for developing and overseeing effective, structured databases. The case study delivers skills on creating proper bills, handling supplies, and tracking supplies instantly. The illustrated case addresses the statistical separation between theory and practice by offering students the tools essential for managing all aspects presented by modern e-commerce environments.

11. Conclusion

In conclusion, the “Gadget Emporium” database structure has effectively created an effective basis for reliability of data and performance. It maintained the accuracy and dependability of the data by placing primary keys (PK), foreign keys (FK), and other constraints in place and using normalization processes.

These components make certain the design remains dependable and efficient in addition to secure to handle adjustments and growth in the corporate environment. Through a strong basis for both present procedures and forthcoming changes, this strategic tactic sets up the database to support long-term achievement.

To promote the accomplishments of the “Gadget Emporium” in the rapidly changing environment of digital commerce, an extensively reported and engaging database design was created through meticulous resolving issues and partnership with customers. This systematic and innovative technique assures that the “Gadget Emporium” will continue operating effortlessly giving Mr. John’s e-commerce ambition in the combative electronics market a strong base.

References

Chris, K., December 21, 2022. *freeCodeCamp.* [Online]
Available at: https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/
[Accessed 15 07 2024].

Taylor, E., December 15, 2023. *The knowledge Academy.* [Online]
Available at: https://www.theknowledgeacademy.com/blog/entity-in-dbms/
[Accessed 15 07 2024].

Anand, G., July 16, 2024. *Naukri.* [Online]
Available at: https://www.naukri.com/code360/library/first-normal-form-in-dbms
[Accessed 15 06 2024].

RoseIndia, 2024. *RoseIndia.* [Online]
Available at: https://www.roseindia.net/sql/databaseconcepts/second-normal-form.shtml
[Accessed 15 07 2024].

Sileshi, D., June 2, 2024. *Baeldung.* [Online]
Available at: https://www.baeldung.com/cs/3nf-vs-bcnf-database-normalization
[Accessed 15 07 2024].

DBS211, 2024. *DBS211.* [Online]
Available at: http://dbs211.ca/courses/dbs211/Week10/index.html
[Accessed 15 07 2024].