# CC5051NI Databases

## 50% Individual Coursework

**Autumn 2023**

**Student Name: Rohit Raut**

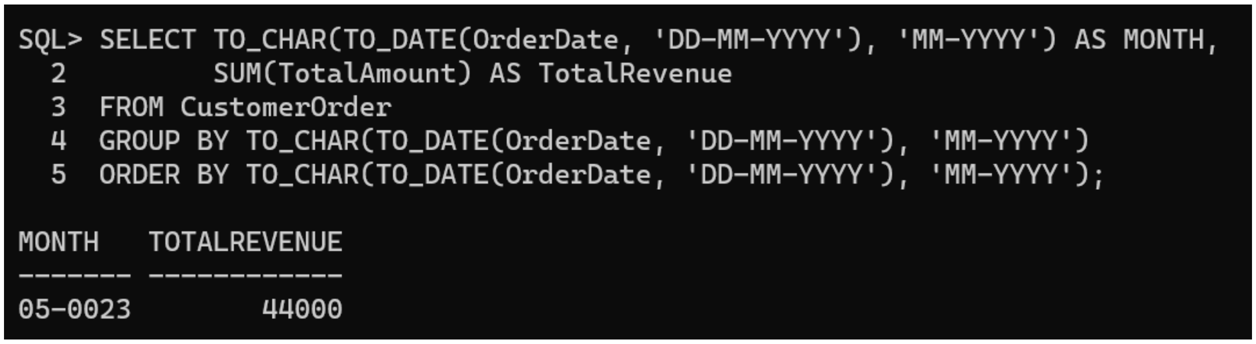
**London Met ID: 22068148**

**Assignment Submission Date: 01/15/2024**

**Word Count:**

## Table of Contents

## Table of Figure



```
SQL> SELECT TO_CHAR(TO_DATE(OrderDate, 'DD-MM-YYYY'), 'MM-YYYY') AS MONTH,
  2         SUM(TotalAmount) AS TotalRevenue
  3  FROM CustomerOrder
  4  GROUP BY TO_CHAR(TO_DATE(OrderDate, 'DD-MM-YYYY'), 'MM-YYYY')
  5  ORDER BY TO_CHAR(TO_DATE(OrderDate, 'DD-MM-YYYY'), 'MM-YYYY');

MONTH   TOTALREVENUE
------- ------------
05-0023        44000
```

# 1. Introduction

Mr. Smith's passion to his online invention rapidly captivates one in the beautiful world of Gadget Emporium, a digital wonderland painstakingly constructed by the effervescent Mr. Smith. Mr. Smith sees this as more than simply an e-commerce venture; it is a living, breathing organism fueled by his infectious enthusiasm. His enthusiasm overflows as he joyfully recounts the one-of-a-kind partnerships he has struck with tech heavyweights Apple and Samsung, portraying them not as distant corporate entities but as cherished friends in the noble pursuit of creativity.

Mr. Smith's enthusiasm for Gadget Emporium extends beyond standard sales presentations, transforming each technological marvel into a compelling tale just begging to be told. In his perspective, this digital utopia is more than just a website. It's a thriving environment where connections are fostered and the latest technology marvels smoothly fit into the fabric of one's life.

As we progress through our coursework, the case study reveals four critical entities at the heart of Gadget Emporium's operations: Customers, Products, Orders, and Vendors. Recognizing the requirement for efficiency and flexibility, the emphasis switches to database organization, which includes removing data duplication and minimizing partial and transitive dependencies. This painstaking procedure, known as normalization, strives to effortlessly split and partition entities and characteristics into discrete groups, decreasing anomalies and maintaining a smooth operating flow for Gadget Emporium, an e-commerce paradise. The objective is to design a database structure that not only improves but also running of the e-commerce store data in proper manner.

## 1.1. Aim and objective

The Gadget Emporium Database Project is focused on transforming corporate operations through the implementation of a large database system. Through a user-friendly interface, it attempts to improve order administration, vendor billing, and customer relations. The primary focus is on personalizing client contacts, boosting product offerings, and optimizing internal procedures to provide customers, vendors, and the greater business community with an improved experience.

Objective:

1. Sell high-quality items at a discount to buyers based on their responsibilities.
2. Assign responsibilities to users and offer discounts.
3. Save essential client information for their protection and privacy.

4. Provide clients with delivery services when they purchase things.
5. Increase the adoption of high-quality items with timely guarantees and warranties.

## 1.2. CURRENT BUSINESS ACTIVITIES AND OPERATION

Gadget Emporium is a technology wonderland, offering a wide range of products from high-speed computers to smart home devices. It establishes itself as the go-to location for tech lovers, perfectly blending a contemporary style with cutting-edge technology. The firm, known as Nepal's leading digital e-shop, has intentionally adopted its own business principles to meet both consumer and organizational needs:

1. The customer is divided into three groups: regulars, staff, and VIPs. Regular clients receive no discount, employees receive a 5% discount, and VIPs receive a larger 10% discount.
2. A real-time monitoring system enables correct stock level management, eliminating over sales, and ensuring product availability.
3. To ease payments, several payment methods such as cash, credit/debit cards, and e-wallets are easily linked.
4. Every product within our stock is defined by its name, description, category, price, and the current stock quantity.
5. Each transaction is linked to a specific payment method.
6. Customer addresses are securely stored in our systems, streamlining the order delivery process.
7. Our products are systematically organized into categories, each encompassing numerous items.
8. Upon order confirmation, an invoice is promptly generated, incorporating details about the order, customer information, and specifications regarding the chosen payment method

## 1.3. Business rule

Gadget Emporium stands as a technological haven, offering a diverse range of products from high-speed computers to innovative smart home devices. It serves as the perfect destination for tech enthusiasts seeking the latest and most stylish items, blending a contemporary design with cutting-edge technology. Renowned as a leading digital e-commerce platform in Nepal, the company has established its unique business principles tailored to both customer needs and organizational goals.

1. The customer base is divided into three categories: regular, staff, and VIP. Regular clients receive no discounts, employees receive a 5% discount, and VIPs receive a 10% discount.
2. Continuous real-time monitoring of product availability to avoid overselling and maintain exact stock level control.
3. Integration of several payment methods, such as cash, credit/debit cards, and e-wallets, to enable safe and easy purchases.

4. Each product has a name, description, category, price, and quantity available in stock.
5. Every order that is placed is connected with a certain payment method.
6. Customer addresses are securely saved for faster order fulfillment.
7. Products are grouped methodically, with each category hosting many goods.
8. Following order confirmation, an invoice is created that includes detailed information about the order, client information, and payment details.

### 1.4. Identification of Entities and attribute

## 2. Assumption

1. Each customer can make multiple product at the same time.
2. In single order multiple product can be purchase similarly, multiple product can be order in multiple order.
3. In each order made by customer, the order must have one payment option.
4. Customer is free to choose payment type such as cash on delivery, Credit/Debit payment, E-wallet s according to their choice.
5. A single vendor can supply multiply product but of only one category Type.
6. The order Id and Customer ID gives the payment for an individual order.
7. The order id and product id gives order quantity to know how many products has been purchase by customer at a particular order.
8. Appropriate discount is given to customer according to their category.
9. A single customer should be of single category. For example customer id 1 can be only VIP not both VIP and STAFF.
10. Name, Address cannot be declared as Primary key which is Bad practice in database as well.

## 3. Initial ERD

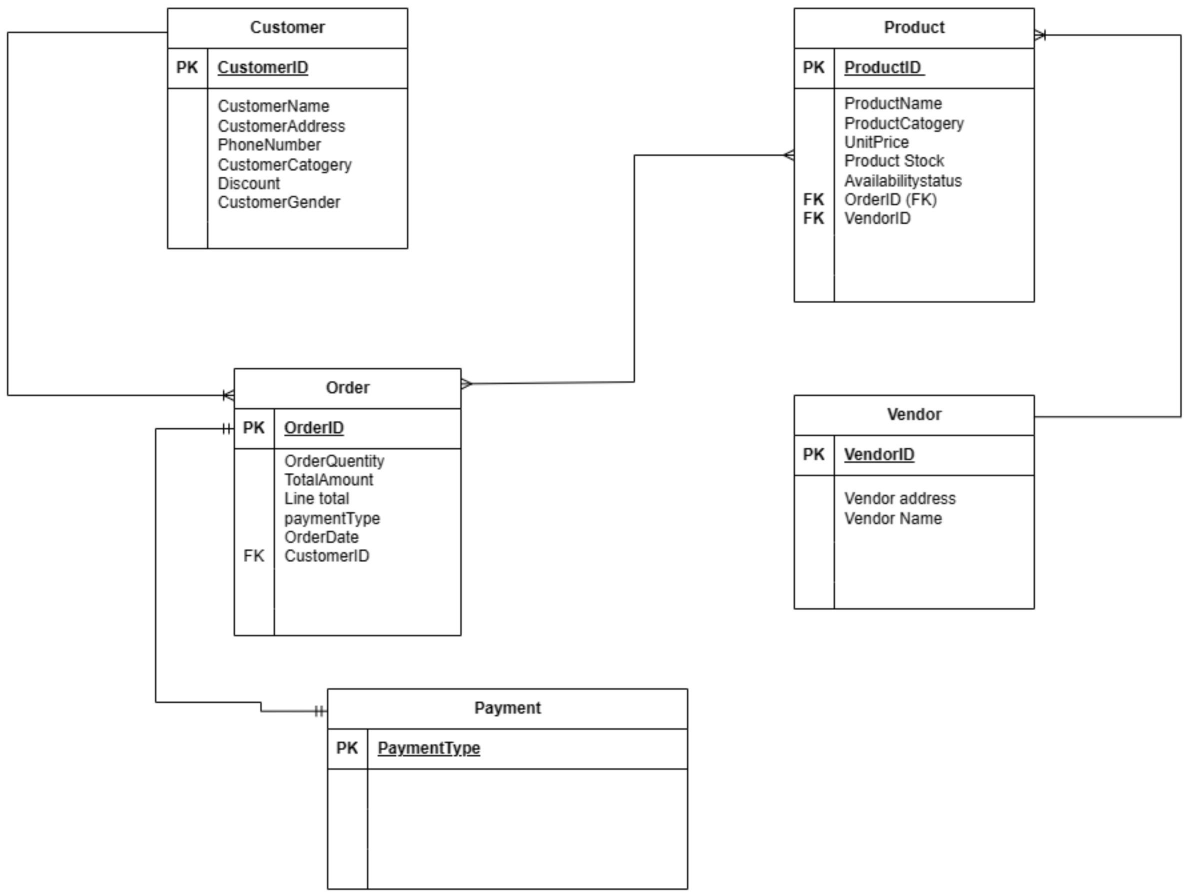


Figure 1: Initial ERD

In this initial ERD, a single customer can make multiple order at same time. And in multiple order multiple product can be purchase. Similarly, multiple product can be supply by single vendor. Lastly, a single order have single payment type.

## 4. Normalization

Normalization is the process of process or step to organize the raw data into database. It basically used to minimize the redundancy of the data for a relation or the set of relations. It help to reduce or eliminate undesirable characteristics such as Insertion, Update and Deletion Anomalies. The main reason to do Normalization is to remove anomalies. The normalization work through a series of stage called Normal from but in this particular case scenario only 1NF, 2NF and 3NF is done. (StudyTonight, 2022)

### UNF

Customer (customer id (PK), name, address, phone number, Customer Gender, category, Customer discount {Order id, order date, total amount, line total, payment option, invoice id, invoice date {product id, product category, product name, order quantity, product stock, Unit price , Availability status, vendor id, vendor address, vendor name})

### 1NF

Customer 1 = ( Customer id(PK), Customer name, Customer address, customer number, Gender id, customer Gender, Category id, Customer category, Customer discount)

Order-1 = (Order id (PK), customer id (FK), order quantity, order date, total amount, Line total, payment id, payment, invoice id, invoice date)

Product-1 = (customer id(FK),Customer id(FK), Product id(PK), product category, product name, product stock, unit price, availability status, vendor id, vendor address, vendor name)

## Assumption for 2NF

### Checking partial dependencies for customer 1 table

Customer id = Customer id(PK), Customer name, Customer address, customer number, customer Gender, Category id, Customer category, Customer discount

For order table

Number of table made (2^n-1) = 3

Order id = Order id (PK), customer id (FK), order date, total amount, Lime total, invoice id, invoice date

Customer id = X

Order id + customer id = payment option, invoice id, invoice date

(In my assumption, one order include one payment and the payment is done by customer)

**For product 1**

Number of table (2^n-1) = 7

Customer id = X

Order id = X

Product id =( customer id(FK), order id(FK), Product id(PK), product category, product name, product stock, unit price, availability status, vendor id, vendor address, vendor name, order quantity)

Customer id + order id = X

Customer id + product id = X

Order id + product id = order quantity

(In an order the product are purchase so to know how much quantity of each product is purchase by the customer order id and product id together gives order quantity)

Order id + product id + customer id = X

## Final table of 2NF are as follow

Customer 2 = ( Customer id(PK), Customer name, Customer address, customer number, Gender id, customer Gender, Category id, Customer category, Customer discount )

Order 2 = (Order id (PK), order date, total amount, Lime total, invoice id, invoice date)

Payment 2 = (Order id (FK), Customer id (FK), payment option)

Product 2 = (Product id (PK), product category, product name, product stock, unit price, availability status, vendor id, vendor address, vendor name)

Quantity order = (order id (FK), product id (FK), order quantity)

Customer id + order id = X

Customer id + product id = X

Order id + product id + customer id = X

(All this empty tables are bridging entity made by composite key, which does not give any values. The normalization is done to reduce data redundancy, so here same foreign key are repeating several times due to this I decided to remove all this table.)

## Assumption for 3NF

Removing transitive dependency in 3NF

**Assumption of customer**

Customer 2 = (Customer id (PK), Customer name, Customer address, customer number, Customer category, Category id, , Customer discount, Gender id, gender)

Customer id → customer name→ customer address X

Customer id → customer address → customer number X

Customer id → contact number →category id X

Customer id → category→ discount

Customer category →(category, discount)

**Assumption of order**

Order 2 = (Order id (PK), order quantity, order date, total amount, Lime total, invoice id, invoice date, invoice time, delivery id, delivery charge)

Order id → order quantity → order date X

Order id → order date →total amount X

Order id →total amount →Line total

(Because lime total is the (quantity order * unit price) and total amount is the amount including discount of the customer according to their category so total amount did not give line total)

Order id → lime total →invoice id X

Order id →invoice id →invoice date

Invoice = (invoice id (PK), invoice date)

**For payment table**

Order id (FK) + Customer id (FK) → →payment option

- Payment (payment id (PK), payment option)

**For product table**

Product 2 = (Product id (PK), product category, product name, product stock, unit price, availability status, vendor id, vendor address, vendor name)

Product id → product category →name X

Product id→ name→ product stock

(Cannot create name as a foreign key because it cannot be PK, so product id directly gives name of product)

Product id →product stock → unit price X

Product id →unit price → available status X

Product id → available status → vendor id X

Product id →vendor id → vendor address

Product id →vendor address →vendor name X

(Address can't be primary key so in my assumption vendor)

Vendor = (vendor id (PK), vendor address, vendor name)

**FOR Total Quantity order table**

- Order id (FK) + product id (FK) = order quantity,

(This key have only one non key value so this table there is no transitive dependency.)

**FINAL TABLE OF 3NF**

Customer 3 = (Customer id (PK), Customer name, Customer address, customer number, Customer category (FK), customer gender)

Category Discount = (category (PK), discount)

Order 3 = (Order id (PK), order quantity, order date, total amount, invoice id (PK),

Invoice 3 = (invoice id (PK), invoice date)

Payment 3= (Order id (FK), Customer id (FK), payment Type)

Product 3= (Product id (PK), product category, product name, product stock, unit price, availability status, vendor id (FK))

Vendor 3 = (vendor id (PK), vendor address, Vendor name)

Order quantity 3= (Order id (FK), product id (FK), quantity order)

## 5. Final ERD

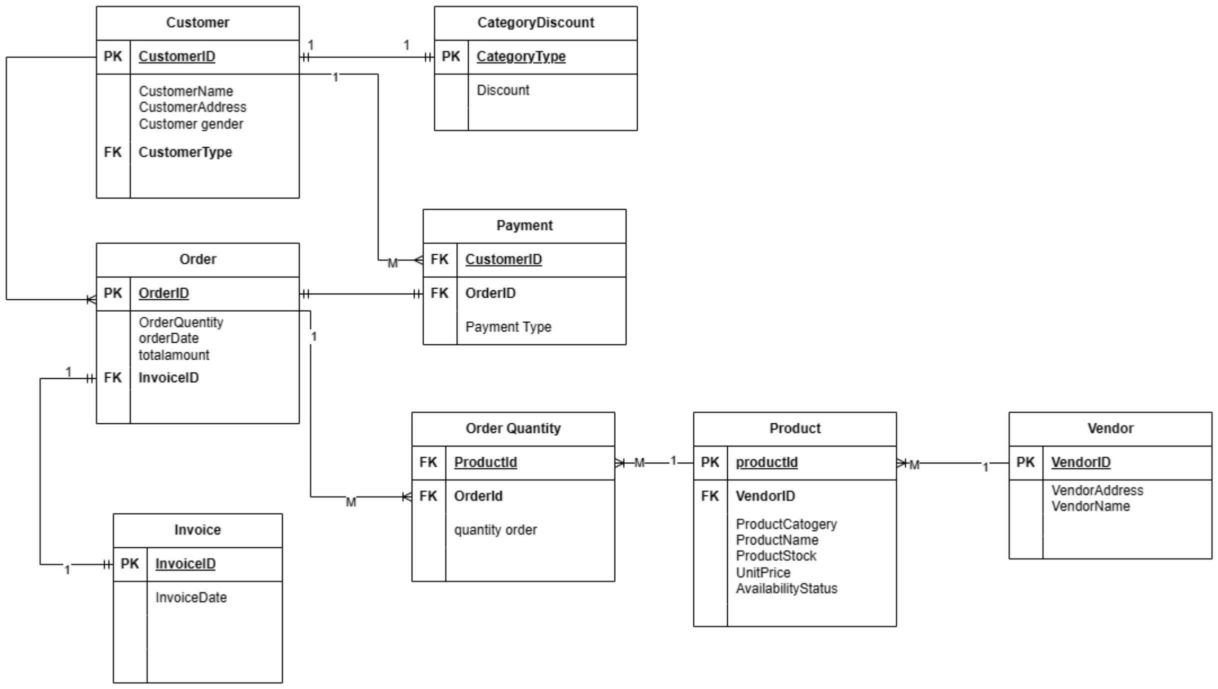


Figure 2: Final ERD

## 6. Implementation

### 6.1. Creating and establishing relations.

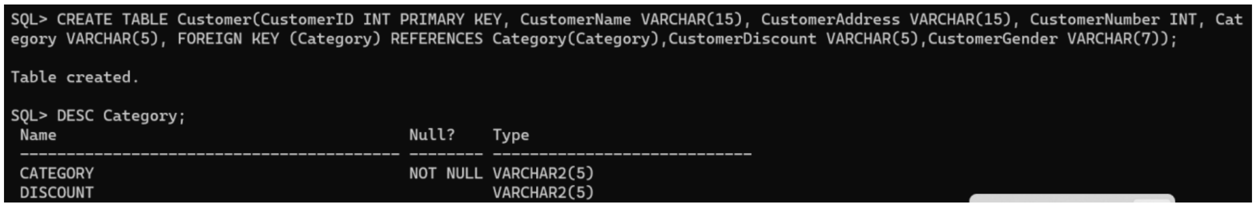
### 6.2. Category

The Category of customer after Normalization look like following table:

| Attribute | Data Type | Description |
|---|---|---|
| Category | VARCHAR(5)<br><br>PRIMARY KEY | Store category of customer. |
| Discount | VARCHAR(15) | This field store the discount rate for customer. |

Table 1: Attributes of Category Table.

**Creation and description of category**



```
SQL> CREATE TABLE Customer(CustomerID INT PRIMARY KEY, CustomerName VARCHAR(15), CustomerAddress VARCHAR(15), CustomerNumber INT, Cat
egory VARCHAR(5), FOREIGN KEY (Category) REFERENCES Category(Category),CustomerDiscount VARCHAR(5),CustomerGender VARCHAR(7));

Table created.

SQL> DESC Category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CATEGORY                                  NOT NULL VARCHAR2(5)
 DISCOUNT                                           VARCHAR2(5)
```

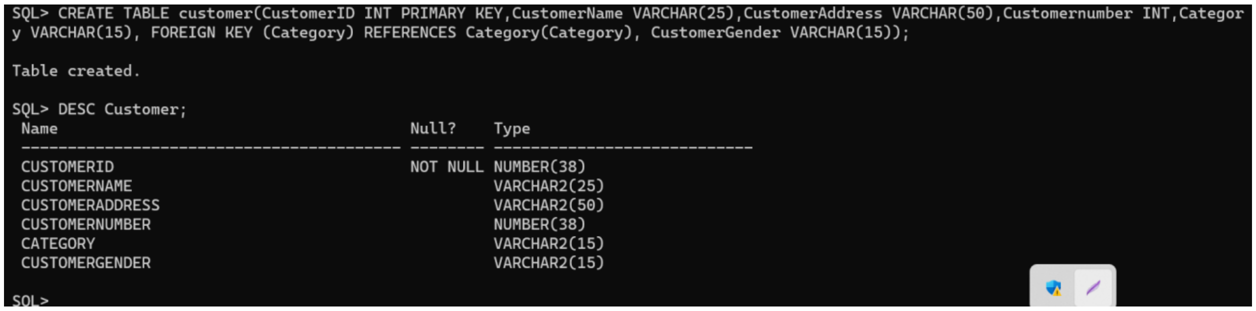
*Figure 3: Creating table for customer and describing it,*

## 6.3. Customer

The customer table look like this after normalization:

| Attribute | Data Type | Description |
|---|---|---|
| CustomerID | INT<br><br>PRIMARY KEY | This field store customer Id |
| CustomerName | VARCHAR(25) | This field store the first name and last name of the customer. |
| CustomerAddress | VARCHAR(50) | This field store address of customer for delivery purpose. |
| CustomerNumber | INT | The field store customer contact number. |
| Category | VARCHAR(15)<br><br>FOREIGN KEY | This field store category for customer. |
| CustomerGender | VARCHAR(15) | The field store the gender of  customer. |

### Creation and description of Customer



```
SQL> CREATE TABLE customer(CustomerID INT PRIMARY KEY,CustomerName VARCHAR(25),CustomerAddress VARCHAR(50),Customernumber INT,Categor
y VARCHAR(15), FOREIGN KEY (Category) REFERENCES Category(Category), CustomerGender VARCHAR(15));

Table created.

SQL> DESC Customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMERID                                NOT NULL NUMBER(38)
 CUSTOMERNAME                                       VARCHAR2(25)
 CUSTOMERADDRESS                                    VARCHAR2(50)
 CUSTOMERNUMBER                                     NUMBER(38)
 CATEGORY                                           VARCHAR2(15)
 CUSTOMERGENDER                                     VARCHAR2(15)

SQL>
```

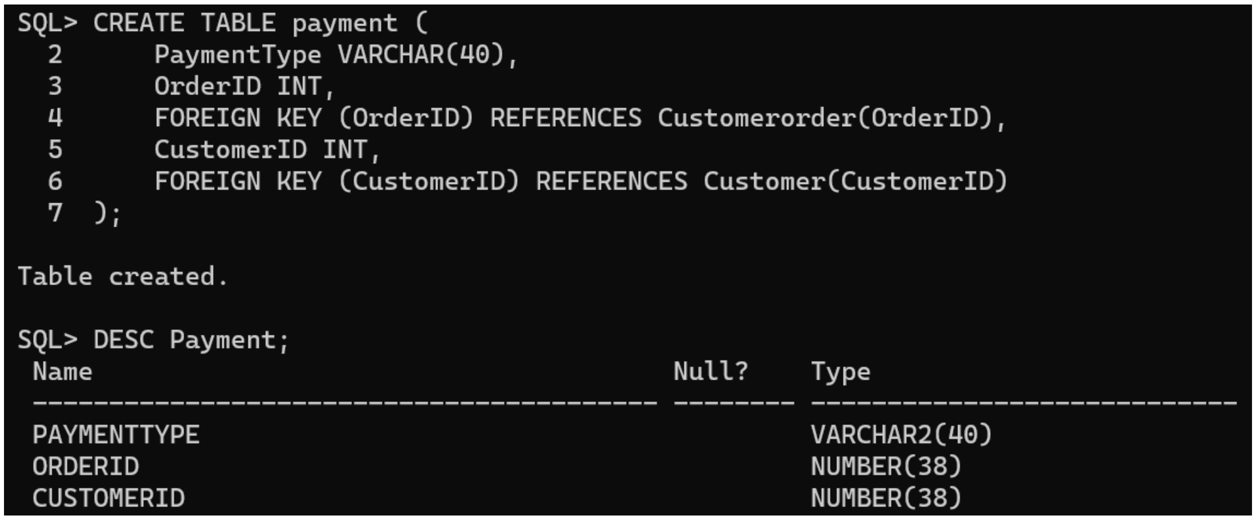
Figure 4: Creating table and describing table.

## 6.4. Payment

The Payment table will be like this after normalization.

| Attribute | Data Type | Description |
|---|---|---|
| OrderID | INT<br><br>FOREIGN KEY | This field is foreign key which extract the value of order table. |
| CustomerID | INT<br><br>FOREIGN KEY | This field id foreign key which reference customer Id in customer table. |
| PaymentType | VARCHAR(40) | This field store the type of payment option for customer. |

### Creation and description of Payment



```
SQL> CREATE TABLE payment (
  2      PaymentType VARCHAR(40),
  3      OrderID INT,
  4      FOREIGN KEY (OrderID) REFERENCES Customerorder(OrderID),
  5      CustomerID INT,
  6      FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
  7  );

Table created.

SQL> DESC Payment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PAYMENTTYPE                                        VARCHAR2(40)
 ORDERID                                            NUMBER(38)
 CUSTOMERID                                         NUMBER(38)
```

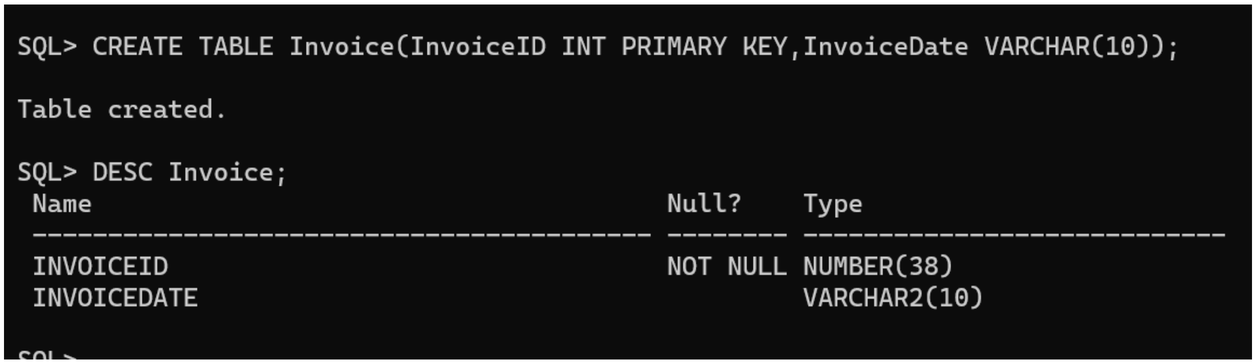
*Figure 5: Creating Payment table and discribing it.*

### 5.5. Invoice Details

Invoice 3 = (invoice id (PK), invoice date)

| Attribute | Data Type | Description |
|---|---|---|
| InvoiceID | INT<br><br>PRIMARY KEY | This field is unique identifier which store invoice id of each order. |
| InvoiceDate | VARCHAR(10) | |

**Creation and description of Customer**



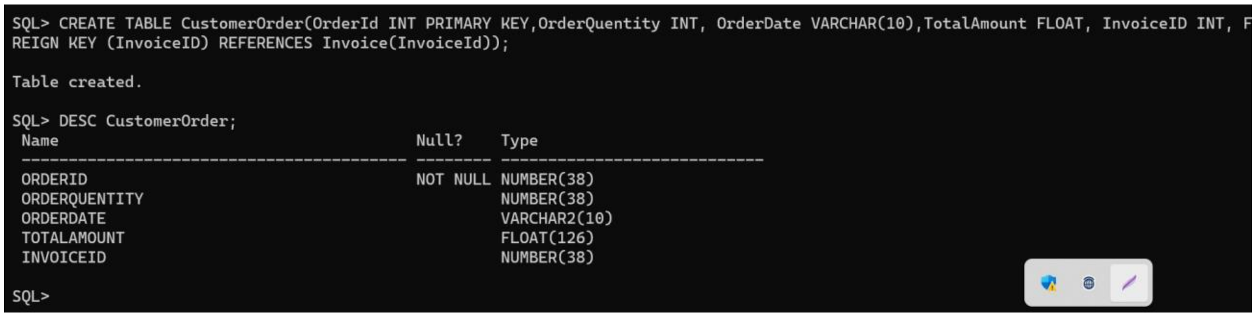
```
SQL> CREATE TABLE Invoice(InvoiceID INT PRIMARY KEY,InvoiceDate VARCHAR(10));

Table created.

SQL> DESC Invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INVOICEID                                 NOT NULL NUMBER(38)
 INVOICEDATE                                        VARCHAR2(10)

SQL>
```

*Figure 6: Creating Invoice Id and describing it.*

### 5.6. Order

| Attribute | Data Type | Description |
|---|---|---|
| OrderID | INT<br><br>PRIMARY KEY | |
| OrderQurntity | INT | |
| OrderDate | VARCHAR(10) | |
| TotalAmount | FLOAT | |
| InvoiceID | INT<br><br>FOREIGN KEY | |

### Creation and description of Customer



```
SQL> CREATE TABLE CustomerOrder(OrderId INT PRIMARY KEY,OrderQuentity INT, OrderDate VARCHAR(10),TotalAmount FLOAT, InvoiceID INT, F
REIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceId));

Table created.

SQL> DESC CustomerOrder;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDERID                                   NOT NULL NUMBER(38)
 ORDERQUENTITY                                      NUMBER(38)
 ORDERDATE                                          VARCHAR2(10)
 TOTALAMOUNT                                        FLOAT(126)
 INVOICEID                                          NUMBER(38)

SQL>
```

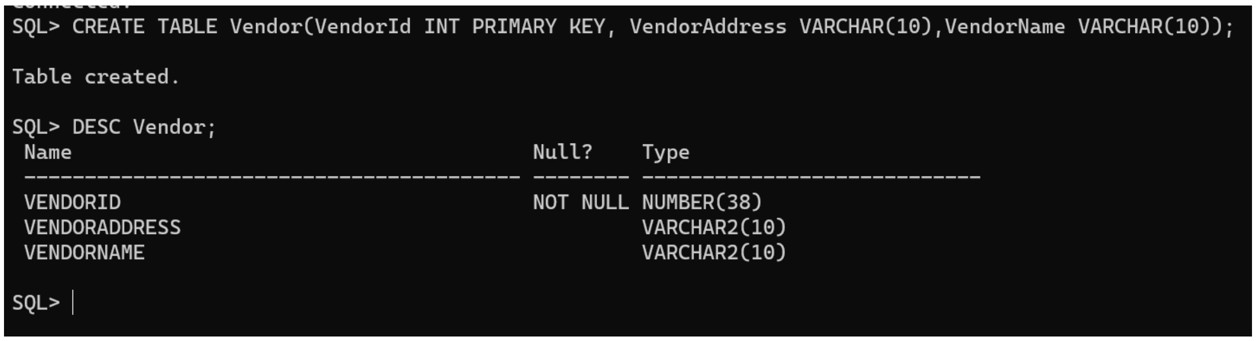
*Figure 7:Creating table Customer Order And Describing that.*

## 5.7. Vendor

Vendor 3 = (vendor id (PK), vendor address, Vendor name)

| Attribute | Data Type | Description |
|---|---|---|
| VendorID | INT<br><br>PRIMARY KEY | |
| VendorAddress | VARCHAR(50) | |
| VendorName | VARCHAR(50) | |

### Creation and description of Vendor



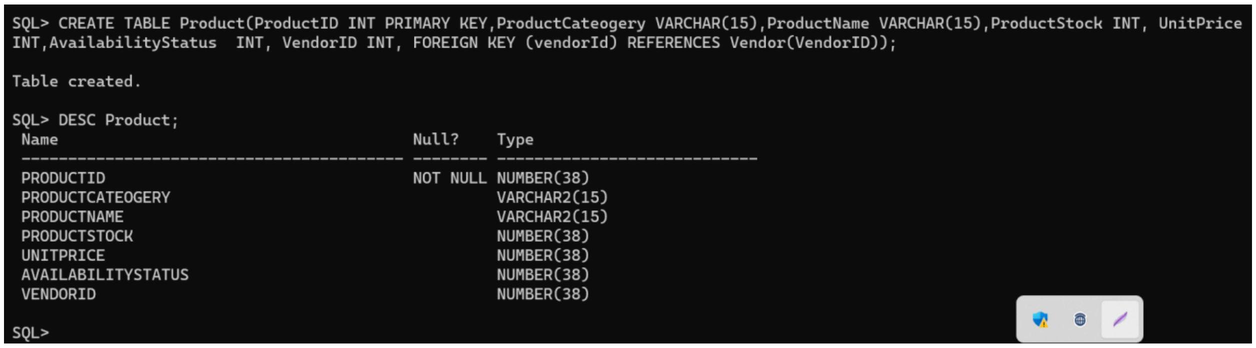
```
SQL> CREATE TABLE Vendor(VendorId INT PRIMARY KEY, VendorAddress VARCHAR(10),VendorName VARCHAR(10));

Table created.

SQL> DESC Vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VENDORID                                  NOT NULL NUMBER(38)
 VENDORADDRESS                                      VARCHAR2(10)
 VENDORNAME                                         VARCHAR2(10)

SQL> |
```

*Figure 8:Creating vendor table and Describing that.*

## 5.7. Product

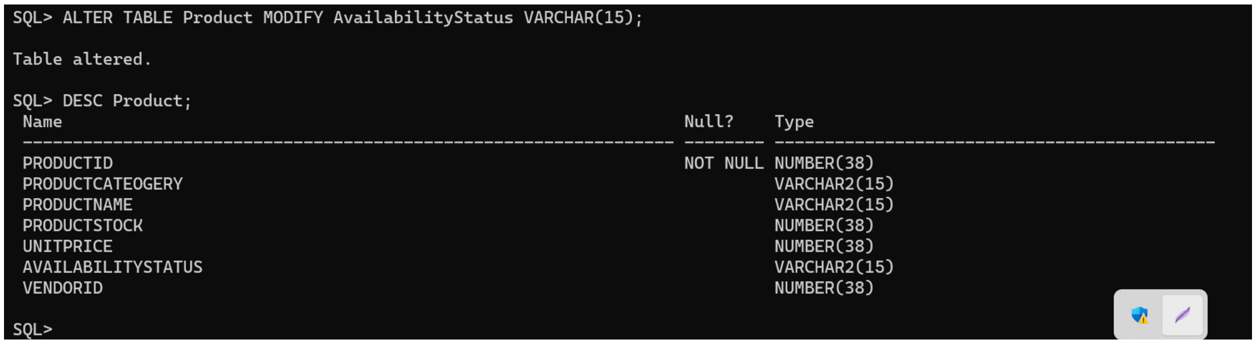
| Attribute | Data Type | Description |
|---|---|---|
| ProductID | INT<br><br>PRIMARY KEY | |
| Productcategory | VARCHAR(15) | |
| ProductName | VARCHAR (15) | |
| ProductStock | INT | |
| UnitPrice | INT | |
| AvailabilityStatus | VARCHAR(15) | |
| VendorID | INT | |

### Creation and description of Vendor



```
SQL> CREATE TABLE Product(ProductID INT PRIMARY KEY,ProductCateogery VARCHAR(15),ProductName VARCHAR(15),ProductStock INT, UnitPrice
INT,AvailabilityStatus  INT, VendorID INT, FOREIGN KEY (vendorId) REFERENCES Vendor(VendorID));

Table created.

SQL> DESC Product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCTID                                 NOT NULL NUMBER(38)
 PRODUCTCATEOGERY                                   VARCHAR2(15)
 PRODUCTNAME                                        VARCHAR2(15)
 PRODUCTSTOCK                                       NUMBER(38)
 UNITPRICE                                          NUMBER(38)
 AVAILABILITYSTATUS                                 NUMBER(38)
 VENDORID                                           NUMBER(38)

SQL>
```

*Figure 9: Creating product table and Describing it.*



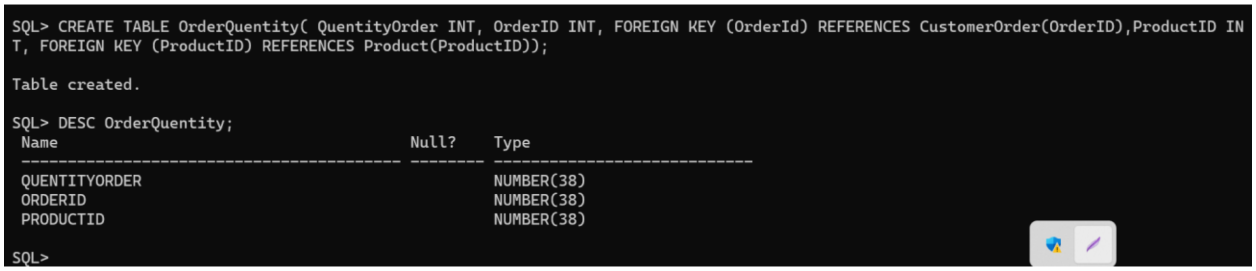
```
SQL> ALTER TABLE Product MODIFY AvailabilityStatus VARCHAR(15);

Table altered.

SQL> DESC Product;
 Name                                                              Null?    Type
 ----------------------------------------------------------------- -------- --------------------------------------------
 PRODUCTID                                                         NOT NULL NUMBER(38)
 PRODUCTCATEOGERY                                                           VARCHAR2(15)
 PRODUCTNAME                                                                VARCHAR2(15)
 PRODUCTSTOCK                                                               NUMBER(38)
 UNITPRICE                                                                  NUMBER(38)
 AVAILABILITYSTATUS                                                         VARCHAR2(15)
 VENDORID                                                                   NUMBER(38)

SQL>
```

*Figure 10: Describing product table and Alter it to modify*

## 5.8. Order Quantity

| Attribute | Data Type | Description |
|---|---|---|
| OrderID | INT<br><br>FOREIGN KEY | |
| ProdutID | INT<br><br>FOREIGN KEY | |
| QueantityOrder | INT | |

### Creation and description of Vendor

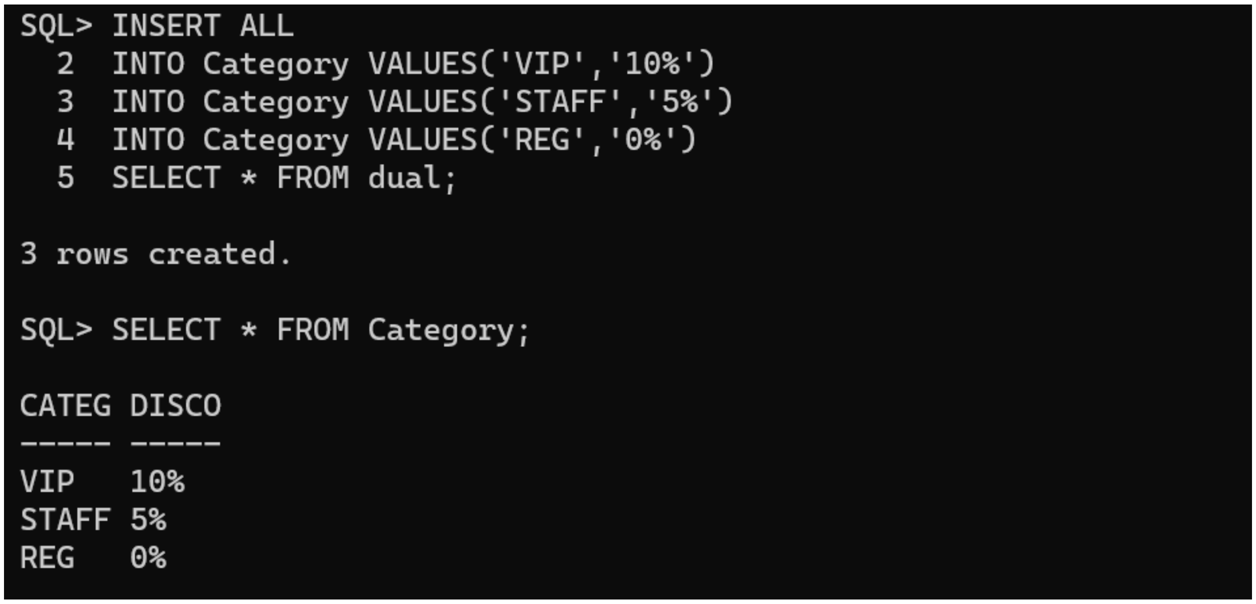


```
SQL> CREATE TABLE OrderQuentity( QuentityOrder INT, OrderID INT, FOREIGN KEY (OrderId) REFERENCES CustomerOrder(OrderID),ProductID IN
T, FOREIGN KEY (ProductID) REFERENCES Product(ProductID));

Table created.

SQL> DESC OrderQuentity;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 QUENTITYORDER                                      NUMBER(38)
 ORDERID                                            NUMBER(38)
 PRODUCTID                                          NUMBER(38)

SQL>
```

*Figure 11: Creating table vendor table and describing it.*

### 5.9. Inserting values into Tables.

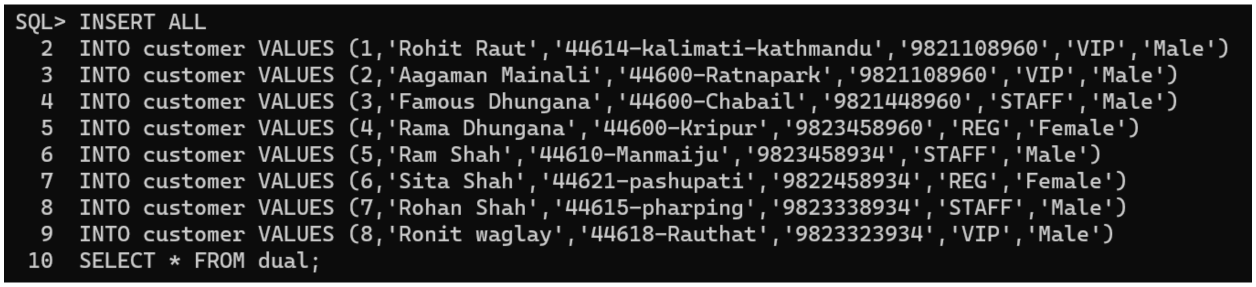
#### 5.9.1. Category



```
SQL> INSERT ALL
  2  INTO Category VALUES('VIP','10%')
  3  INTO Category VALUES('STAFF','5%')
  4  INTO Category VALUES('REG','0%')
  5  SELECT * FROM dual;

3 rows created.

SQL> SELECT * FROM Category;

CATEG DISCO
----- -----
VIP   10%
STAFF 5%
REG   0%
```

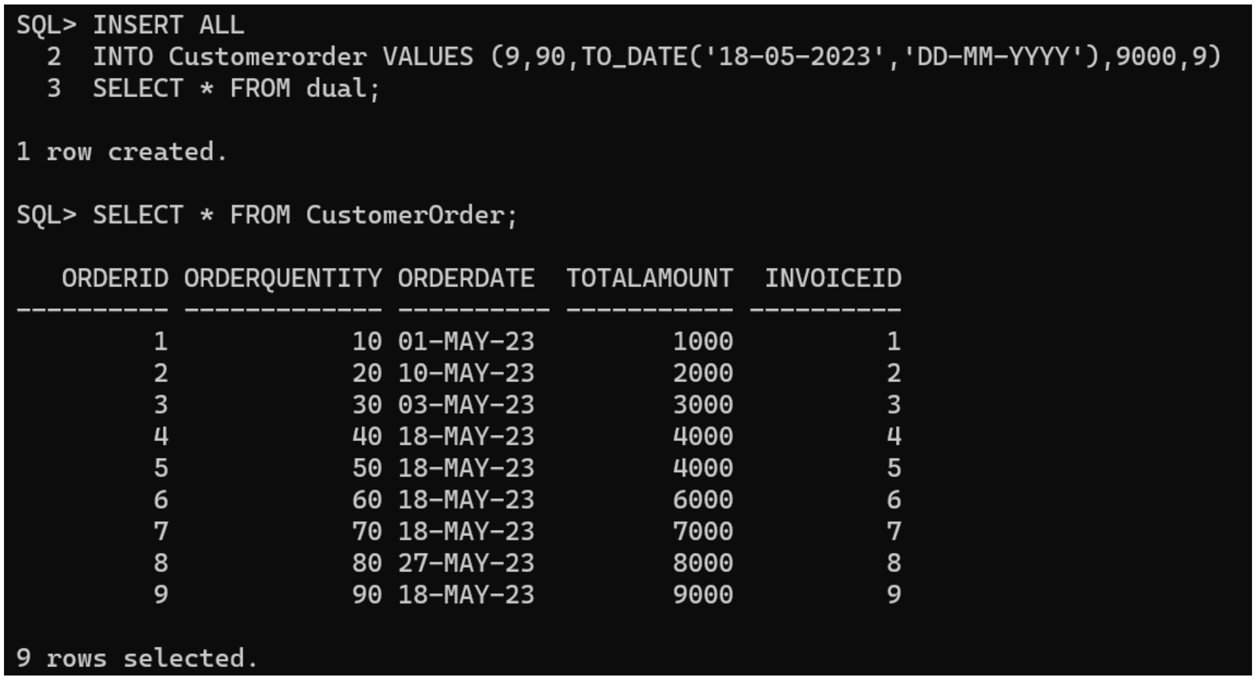
*Figure 12:Inserting values into category And showing table*

### 5.9.2. Customer



```
SQL> INSERT ALL
  INTO customer VALUES (1,'Rohit Raut','44614-kalimati-kathmandu','9821108960','VIP','Male')
  INTO customer VALUES (2,'Aagaman Mainali','44600-Ratnapark','9821108960','VIP','Male')
  INTO customer VALUES (3,'Famous Dhungana','44600-Chabail','9821448960','STAFF','Male')
  INTO customer VALUES (4,'Rama Dhungana','44600-Kripur','9823458960','REG','Female')
  INTO customer VALUES (5,'Ram Shah','44610-Manmaiju','9823458934','STAFF','Male')
  INTO customer VALUES (6,'Sita Shah','44621-pashupati','9822458934','REG','Female')
  INTO customer VALUES (7,'Rohan Shah','44615-pharping','9823338934','STAFF','Male')
  INTO customer VALUES (8,'Ronit waglay','44618-Rauthat','9823323934','VIP','Male')
  SELECT * FROM dual;
```

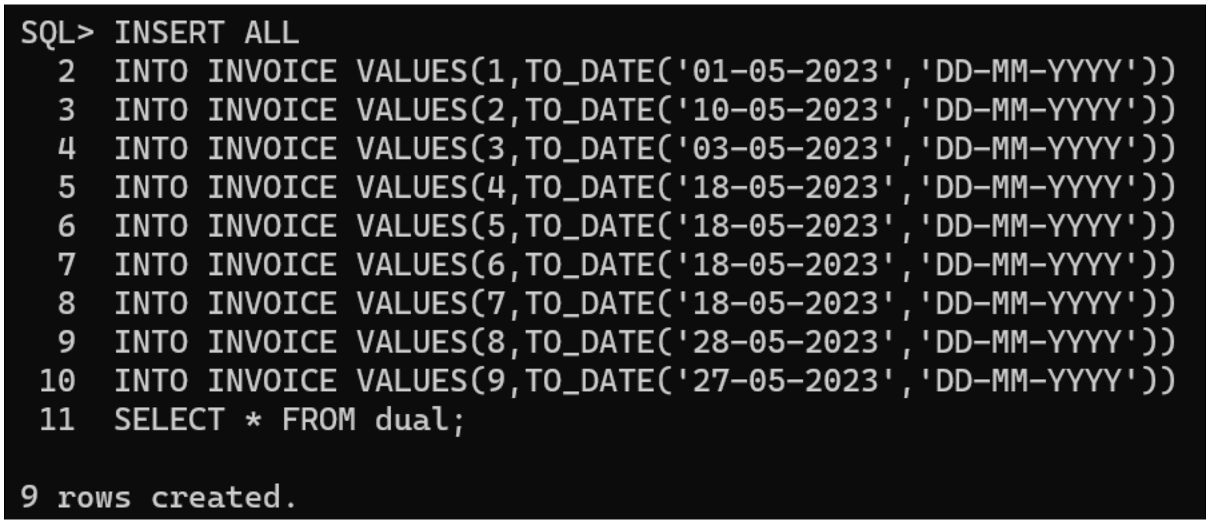
*Figure 13: Inserting values into customer.*



```
SQL> INSERT ALL
  INTO Customerorder VALUES (9,90,TO_DATE('18-05-2023','DD-MM-YYYY'),9000,9)
  SELECT * FROM dual;

1 row created.

SQL> SELECT * FROM CustomerOrder;

   ORDERID ORDERQUANTITY ORDERDATE  TOTALAMOUNT  INVOICEID
---------- ------------- ---------- ----------- ----------
         1            10 01-MAY-23         1000          1
         2            20 10-MAY-23         2000          2
         3            30 03-MAY-23         3000          3
         4            40 18-MAY-23         4000          4
         5            50 18-MAY-23         4000          5
         6            60 18-MAY-23         6000          6
         7            70 18-MAY-23         7000          7
         8            80 27-MAY-23         8000          8
         9            90 18-MAY-23         9000          9

9 rows selected.
```

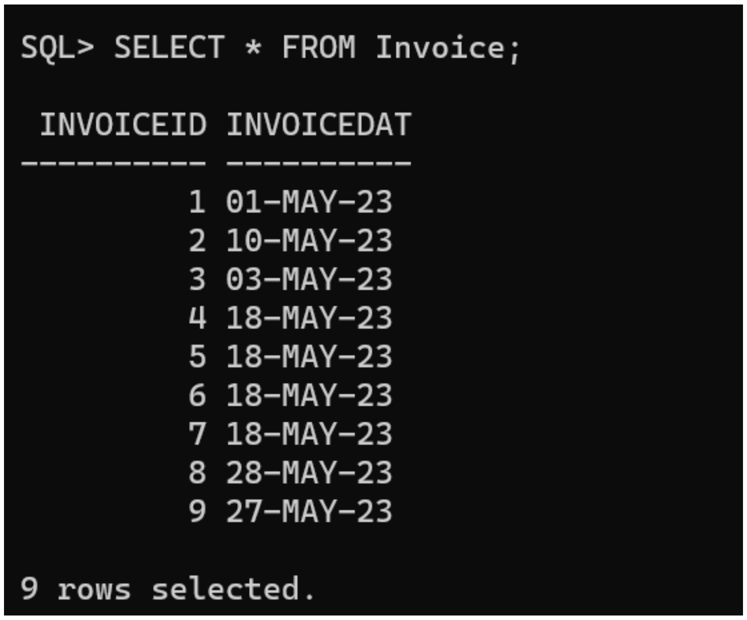
*Figure 14: Inserting values into customer Order table and displaying values.*

### 5.9.3. Invoice



```
SQL> INSERT ALL
INTO INVOICE VALUES(1,TO_DATE('01-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(2,TO_DATE('10-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(3,TO_DATE('03-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(4,TO_DATE('18-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(5,TO_DATE('18-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(6,TO_DATE('18-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(7,TO_DATE('18-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(8,TO_DATE('28-05-2023','DD-MM-YYYY'))
INTO INVOICE VALUES(9,TO_DATE('27-05-2023','DD-MM-YYYY'))
SELECT * FROM dual;

9 rows created.
```

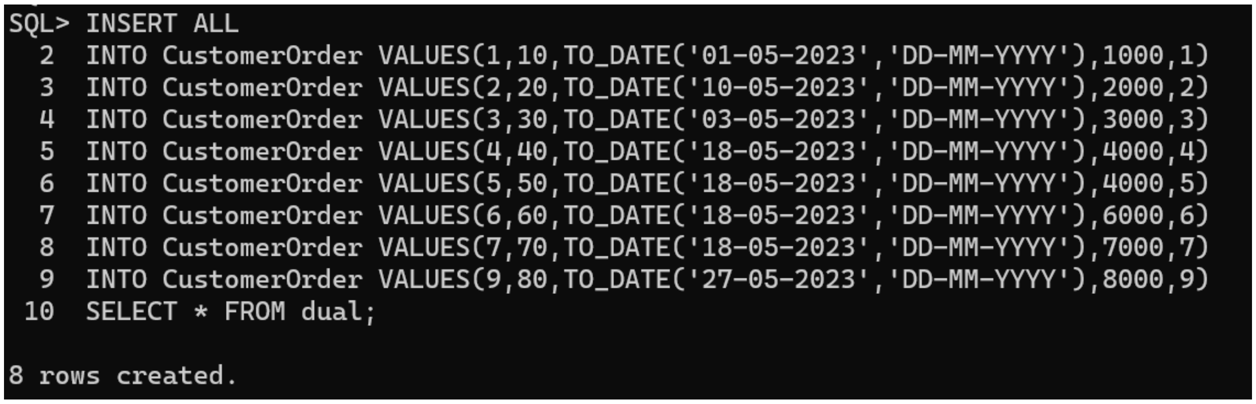
*Figure 15: Inserting Values into Invoice and desplaying values.*



```
SQL> SELECT * FROM Invoice;

 INVOICEID INVOICEDAT
---------- ----------
         1 01-MAY-23
         2 10-MAY-23
         3 03-MAY-23
         4 18-MAY-23
         5 18-MAY-23
         6 18-MAY-23
         7 18-MAY-23
         8 28-MAY-23
         9 27-MAY-23

9 rows selected.
```

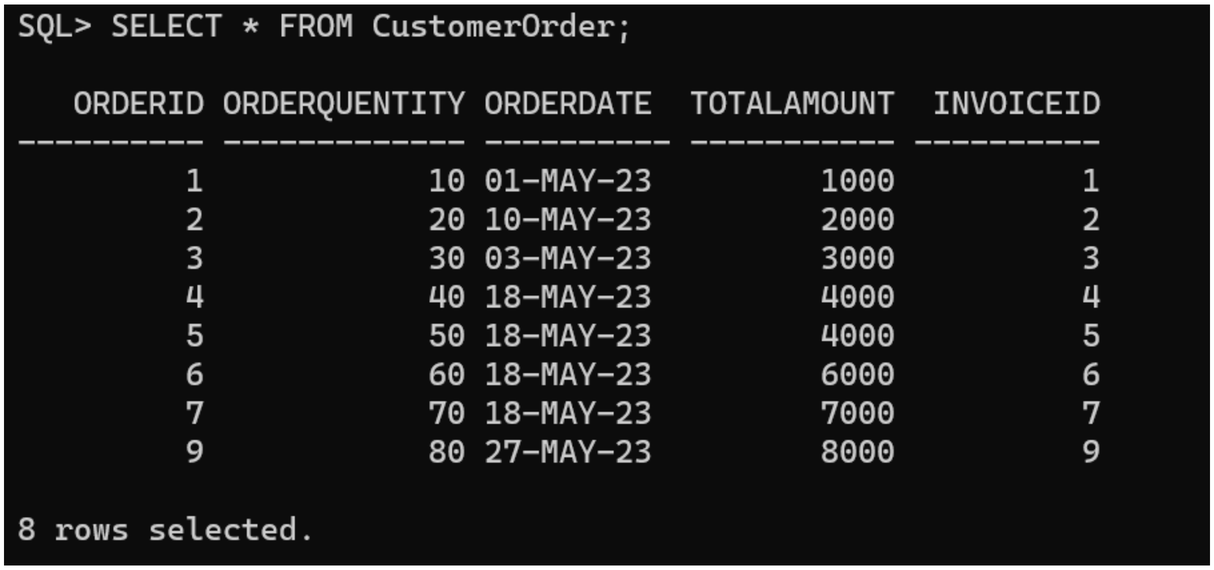
*Figure 16: Displaying invoice values.*

### 5.9.3. Order



```
SQL> INSERT ALL
  2  INTO CustomerOrder VALUES(1,10,TO_DATE('01-05-2023','DD-MM-YYYY'),1000,1)
  3  INTO CustomerOrder VALUES(2,20,TO_DATE('10-05-2023','DD-MM-YYYY'),2000,2)
  4  INTO CustomerOrder VALUES(3,30,TO_DATE('03-05-2023','DD-MM-YYYY'),3000,3)
  5  INTO CustomerOrder VALUES(4,40,TO_DATE('18-05-2023','DD-MM-YYYY'),4000,4)
  6  INTO CustomerOrder VALUES(5,50,TO_DATE('18-05-2023','DD-MM-YYYY'),4000,5)
  7  INTO CustomerOrder VALUES(6,60,TO_DATE('18-05-2023','DD-MM-YYYY'),6000,6)
  8  INTO CustomerOrder VALUES(7,70,TO_DATE('18-05-2023','DD-MM-YYYY'),7000,7)
  9  INTO CustomerOrder VALUES(9,80,TO_DATE('27-05-2023','DD-MM-YYYY'),8000,9)
 10  SELECT * FROM dual;

8 rows created.
```

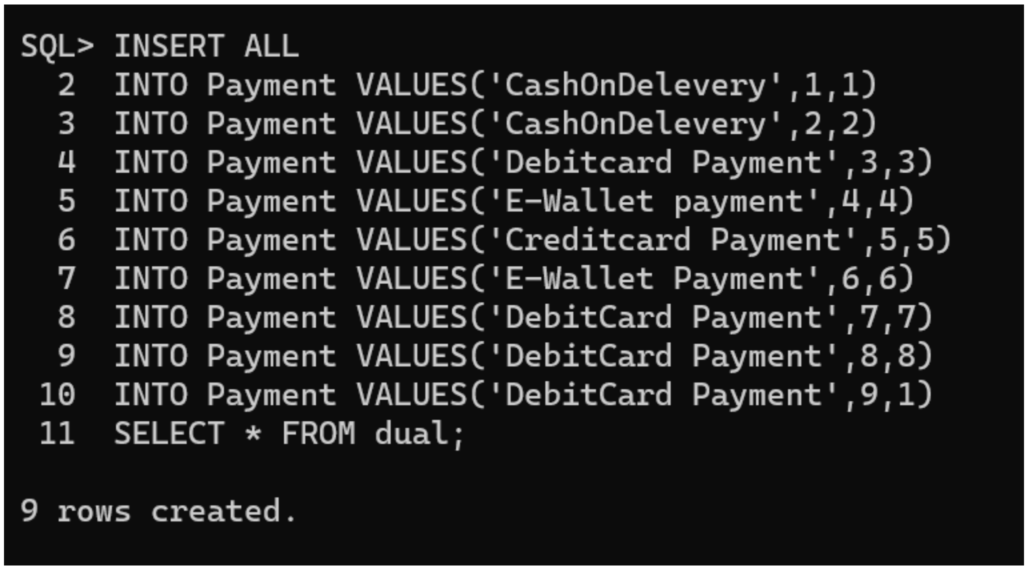
*Figure 17: Inserting values into CustomerOrder*



```
SQL> SELECT * FROM CustomerOrder;

   ORDERID ORDERQUENTITY ORDERDATE  TOTALAMOUNT  INVOICEID
---------- ------------- ---------- ----------- ----------
         1            10 01-MAY-23         1000          1
         2            20 10-MAY-23         2000          2
         3            30 03-MAY-23         3000          3
         4            40 18-MAY-23         4000          4
         5            50 18-MAY-23         4000          5
         6            60 18-MAY-23         6000          6
         7            70 18-MAY-23         7000          7
         9            80 27-MAY-23         8000          9

8 rows selected.
```

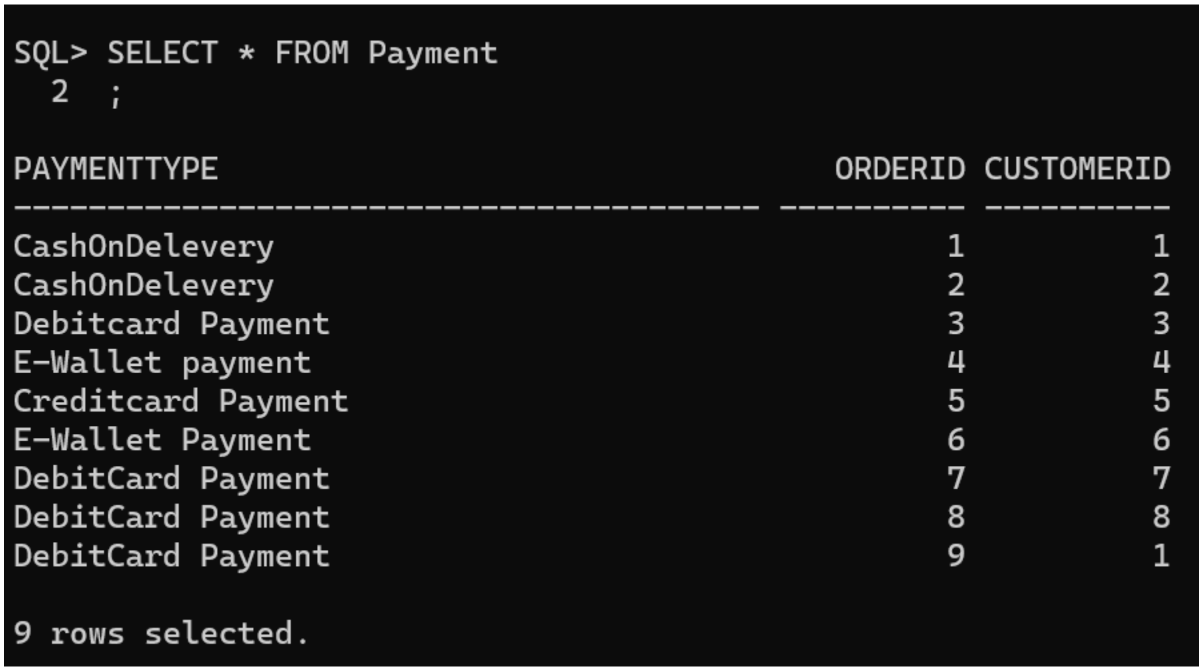
*Figure 18: Displaying  values of customerOrder.*

### 5.9.4. Payment



```
SQL> INSERT ALL
  2  INTO Payment VALUES('CashOnDelevery',1,1)
  3  INTO Payment VALUES('CashOnDelevery',2,2)
  4  INTO Payment VALUES('Debitcard Payment',3,3)
  5  INTO Payment VALUES('E-Wallet payment',4,4)
  6  INTO Payment VALUES('Creditcard Payment',5,5)
  7  INTO Payment VALUES('E-Wallet Payment',6,6)
  8  INTO Payment VALUES('DebitCard Payment',7,7)
  9  INTO Payment VALUES('DebitCard Payment',8,8)
 10  INTO Payment VALUES('DebitCard Payment',9,1)
 11  SELECT * FROM dual;

9 rows created.
```

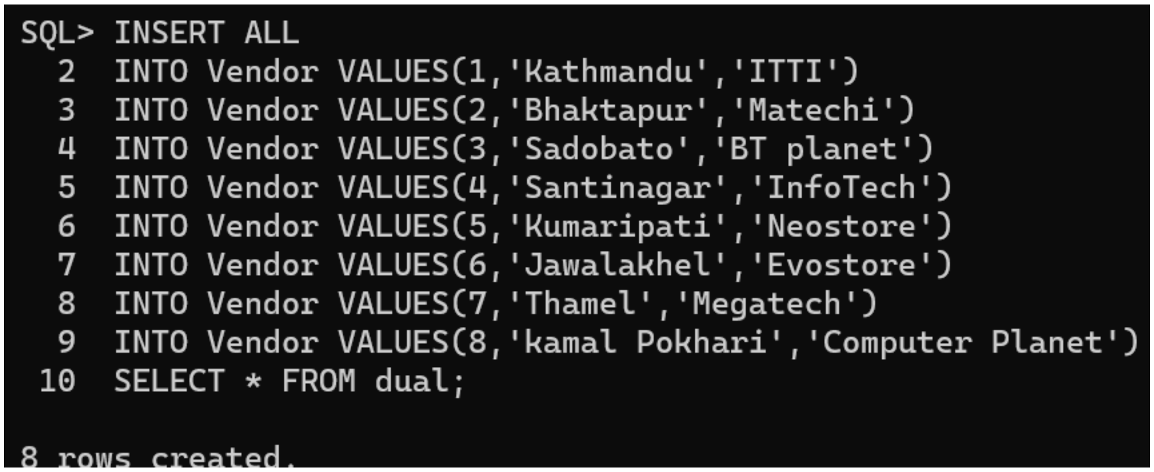
*Figure 19: Inserting values into Payment table*



```
SQL> SELECT * FROM Payment
  2  ;

PAYMENTTYPE                                 ORDERID CUSTOMERID
---------------------------------------- ---------- ----------
CashOnDelevery                                    1          1
CashOnDelevery                                    2          2
Debitcard Payment                                 3          3
E-Wallet payment                                  4          4
Creditcard Payment                                5          5
E-Wallet Payment                                  6          6
DebitCard Payment                                 7          7
DebitCard Payment                                 8          8
DebitCard Payment                                 9          1

9 rows selected.
```

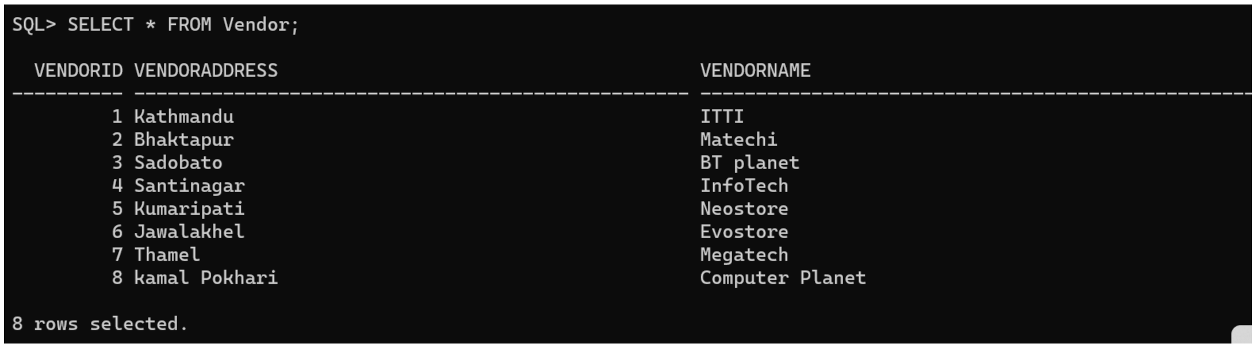
*Figure 20: Displaying values of Payment table.*

## 5.9.5. Vendor



```
SQL> INSERT ALL
  2  INTO Vendor VALUES(1,'Kathmandu','ITTI')
  3  INTO Vendor VALUES(2,'Bhaktapur','Matechi')
  4  INTO Vendor VALUES(3,'Sadobato','BT planet')
  5  INTO Vendor VALUES(4,'Santinagar','InfoTech')
  6  INTO Vendor VALUES(5,'Kumaripati','Neostore')
  7  INTO Vendor VALUES(6,'Jawalakhel','Evostore')
  8  INTO Vendor VALUES(7,'Thamel','Megatech')
  9  INTO Vendor VALUES(8,'kamal Pokhari','Computer Planet')
 10  SELECT * FROM dual;

8 rows created.
```

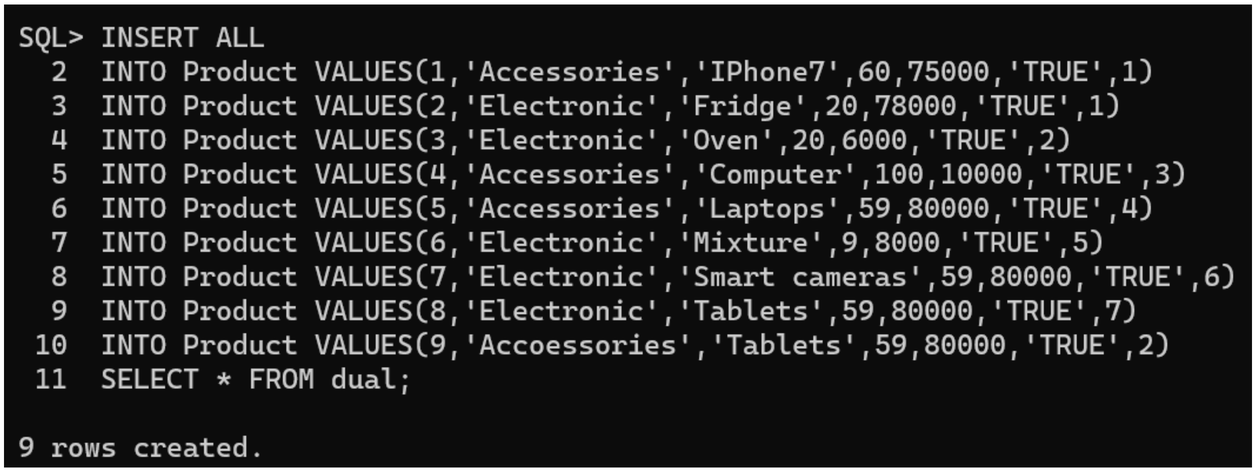
*Figure 21: Inserting values into Vendor table.*



```
SQL> SELECT * FROM Vendor;

  VENDORID VENDORADDRESS                                      VENDORNAME
---------- -------------------------------------------------- --------------------------------------------------
         1 Kathmandu                                          ITTI
         2 Bhaktapur                                          Matechi
         3 Sadobato                                           BT planet
         4 Santinagar                                         InfoTech
         5 Kumaripati                                         Neostore
         6 Jawalakhel                                         Evostore
         7 Thamel                                             Megatech
         8 kamal Pokhari                                      Computer Planet

8 rows selected.
```

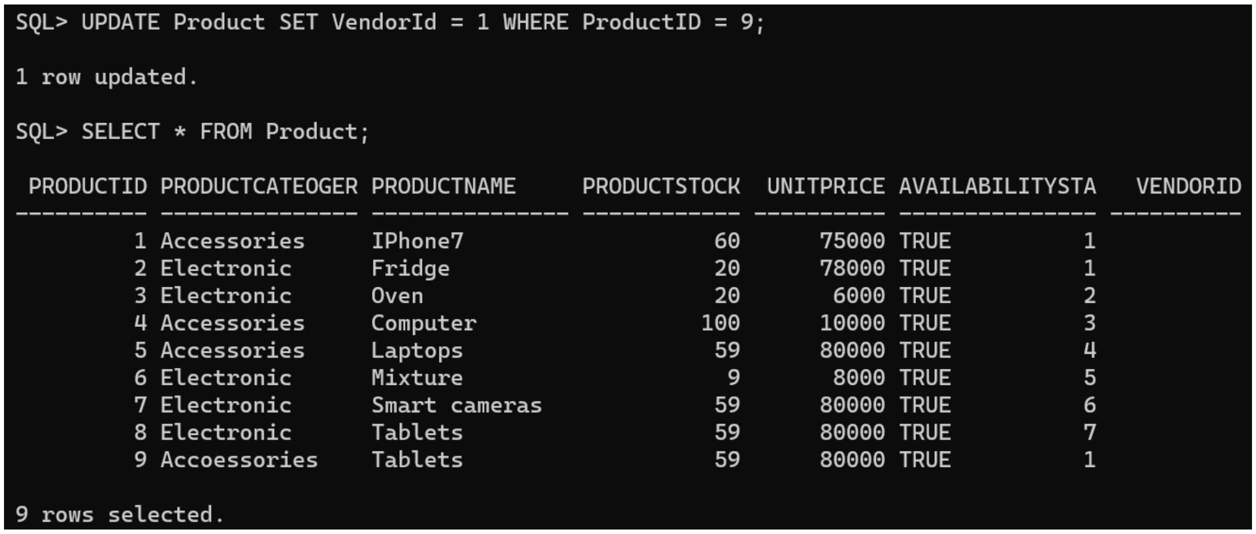
*Figure 22: Displaying values of vendor table.*

### 5.9.6. Product



```
SQL> INSERT ALL
INTO Product VALUES(1,'Accessories','IPhone7',60,75000,'TRUE',1)
INTO Product VALUES(2,'Electronic','Fridge',20,78000,'TRUE',1)
INTO Product VALUES(3,'Electronic','Oven',20,6000,'TRUE',2)
INTO Product VALUES(4,'Accessories','Computer',100,10000,'TRUE',3)
INTO Product VALUES(5,'Accessories','Laptops',59,80000,'TRUE',4)
INTO Product VALUES(6,'Electronic','Mixture',9,8000,'TRUE',5)
INTO Product VALUES(7,'Electronic','Smart cameras',59,80000,'TRUE',6)
INTO Product VALUES(8,'Electronic','Tablets',59,80000,'TRUE',7)
INTO Product VALUES(9,'Accoessories','Tablets',59,80000,'TRUE',2)
SELECT * FROM dual;

9 rows created.
```

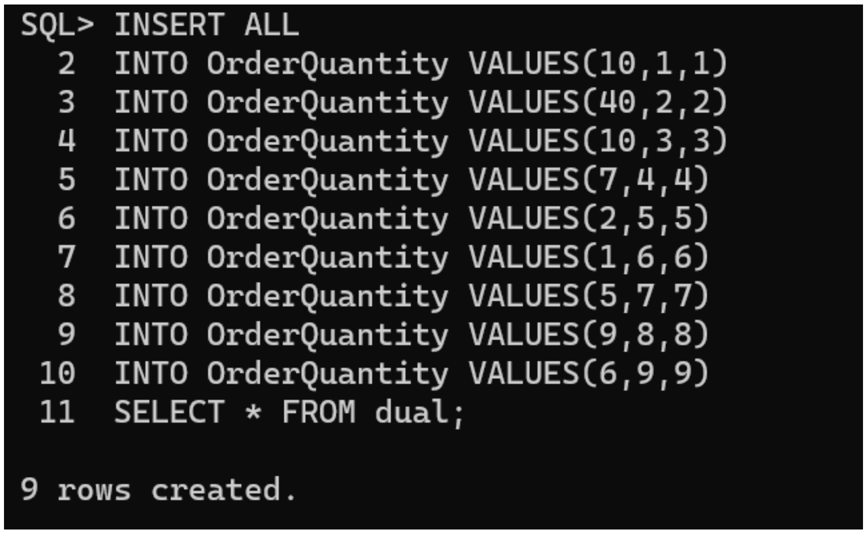
*Figure 23: Inserting values into product table.*



```
SQL> UPDATE Product SET VendorId = 1 WHERE ProductID = 9;

1 row updated.

SQL> SELECT * FROM Product;

 PRODUCTID PRODUCTCATEOGER PRODUCTNAME     PRODUCTSTOCK  UNITPRICE AVAILABILITYSTA   VENDORID
---------- --------------- --------------- ------------ ---------- --------------- ----------
         1 Accessories     IPhone7                   60      75000 TRUE                     1
         2 Electronic      Fridge                    20      78000 TRUE                     1
         3 Electronic      Oven                      20       6000 TRUE                     2
         4 Accessories     Computer                 100      10000 TRUE                     3
         5 Accessories     Laptops                   59      80000 TRUE                     4
         6 Electronic      Mixture                    9       8000 TRUE                     5
         7 Electronic      Smart cameras             59      80000 TRUE                     6
         8 Electronic      Tablets                   59      80000 TRUE                     7
         9 Accoessories    Tablets                   59      80000 TRUE                     1

9 rows selected.
```

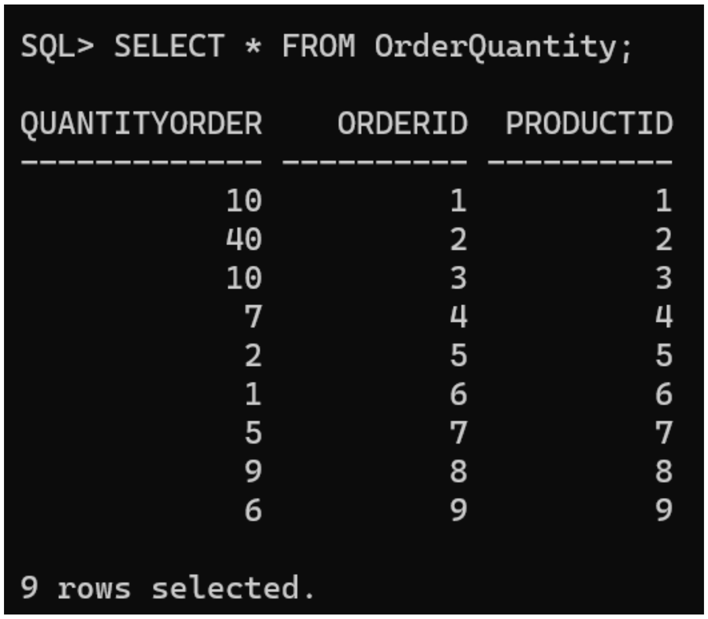
*Figure 24: Displaying values into Vendor table.*

### 5.9.7. Order Quantity



```
SQL> INSERT ALL
  2  INTO OrderQuantity VALUES(10,1,1)
  3  INTO OrderQuantity VALUES(40,2,2)
  4  INTO OrderQuantity VALUES(10,3,3)
  5  INTO OrderQuantity VALUES(7,4,4)
  6  INTO OrderQuantity VALUES(2,5,5)
  7  INTO OrderQuantity VALUES(1,6,6)
  8  INTO OrderQuantity VALUES(5,7,7)
  9  INTO OrderQuantity VALUES(9,8,8)
 10  INTO OrderQuantity VALUES(6,9,9)
 11  SELECT * FROM dual;

9 rows created.
```

*Figure 25: Inserting values into OrderQuentity table.*



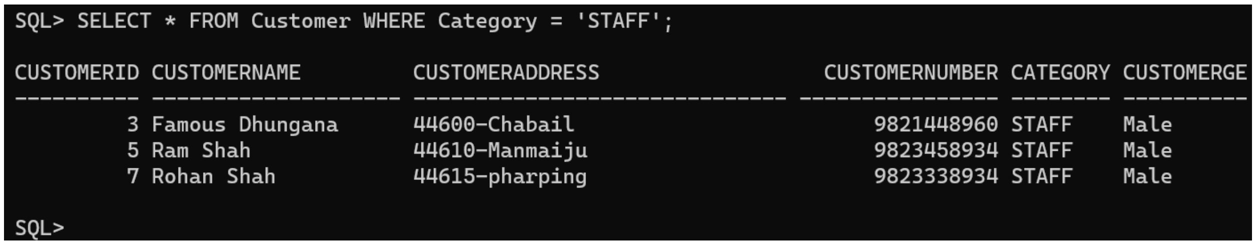
```
SQL> SELECT * FROM OrderQuantity;

QUANTITYORDER    ORDERID  PRODUCTID
------------- ---------- ----------
           10          1          1
           40          2          2
           10          3          3
            7          4          4
            2          5          5
            1          6          6
            5          7          7
            9          8          8
            6          9          9

9 rows selected.
```

*Figure 26: Displaying values of OrderQuentty table.*

## 7. Database Querying

### 7.1. Information query

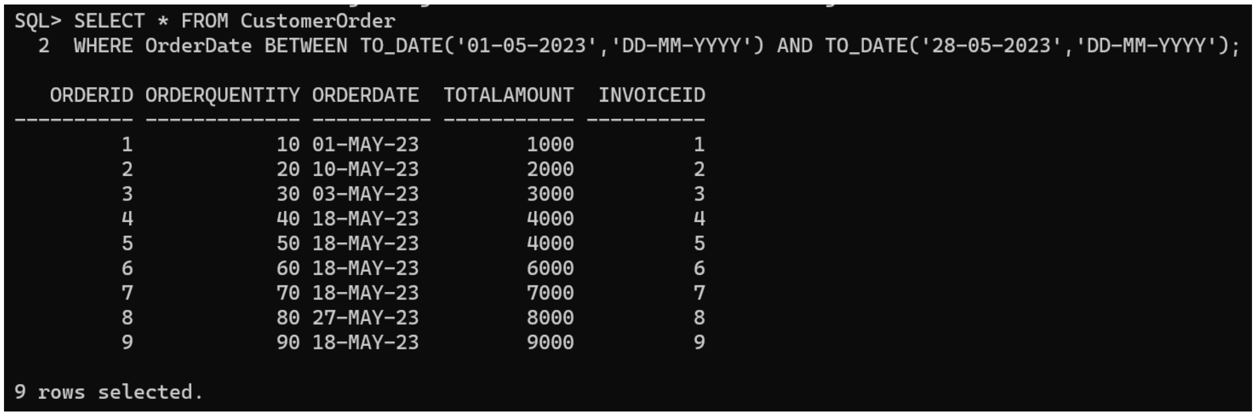
- List all the customers that are also staff of the company.



```
SQL> SELECT * FROM Customer WHERE Category = 'STAFF';

CUSTOMERID CUSTOMERNAME         CUSTOMERADDRESS                CUSTOMERNUMBER CATEGORY CUSTOMERGE
---------- -------------------- ------------------------------ -------------- -------- ----------
         3 Famous Dhungana      44600-Chabail                      9821448960 STAFF    Male
         5 Ram Shah             44610-Manmaiju                     9823458934 STAFF    Male
         7 Rohan Shah           44615-pharping                     9823338934 STAFF    Male

SQL>
```

Figure 27: Answer of Information query 1.

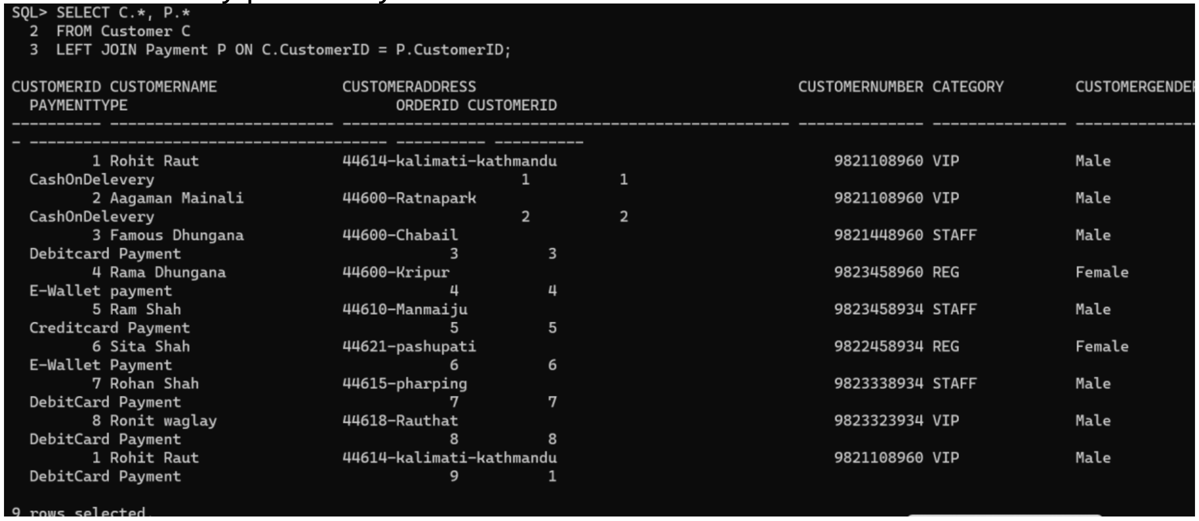
- List all the orders made for any particular product between the dates 01-05-2023 till 28- 05-2023.



```
SQL> SELECT * FROM CustomerOrder
  2  WHERE OrderDate BETWEEN TO_DATE('01-05-2023','DD-MM-YYYY') AND TO_DATE('28-05-2023','DD-MM-YYYY');

   ORDERID ORDERQUENTITY ORDERDATE  TOTALAMOUNT  INVOICEID
---------- ------------- ---------- ----------- ----------
         1            10 01-MAY-23         1000          1
         2            20 10-MAY-23         2000          2
         3            30 03-MAY-23         3000          3
         4            40 18-MAY-23         4000          4
         5            50 18-MAY-23         4000          5
         6            60 18-MAY-23         6000          6
         7            70 18-MAY-23         7000          7
         8            80 27-MAY-23         8000          8
         9            90 18-MAY-23         9000          9

9 rows selected.
```

Figure 28:Answer of Information query 2

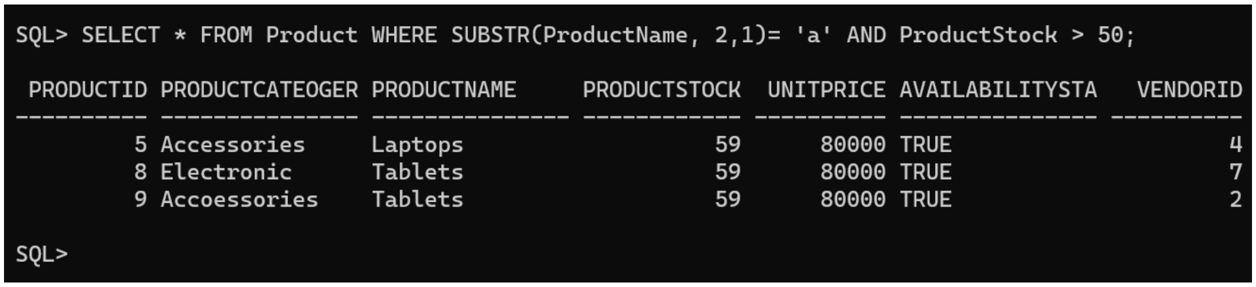
- List all the customers with their order details and also the customers who have not ordered any products yet.



```
SQL> SELECT C.*, P.*
  2  FROM Customer C
  3  LEFT JOIN Payment P ON C.CustomerID = P.CustomerID;

CUSTOMERID CUSTOMERNAME              CUSTOMERADDRESS                                    CUSTOMERNUMBER CATEGORY       CUSTOMERGENDE
  PAYMENTTYPE                              ORDERID CUSTOMERID
---------- ------------------------- -------------------------------------------------- -------------- -------------- -------------
- ---------------------------------------- ---------- ----------
         1 Rohit Raut                44614-kalimati-kathmandu                               9821108960 VIP            Male
  CashOnDelevery                                    1          1
         2 Aagaman Mainali           44600-Ratnapark                                        9821108960 VIP            Male
  CashOnDelevery                                    2          2
         3 Famous Dhungana           44600-Chabail                                          9821448960 STAFF          Male
  Debitcard Payment                        3          3
         4 Rama Dhungana             44600-Kripur                                           9823458960 REG            Female
  E-Wallet payment                         4          4
         5 Ram Shah                  44610-Manmaiju                                         9823458934 STAFF          Male
  Creditcard Payment                       5          5
         6 Sita Shah                 44621-pashupati                                        9822458934 REG            Female
  E-Wallet Payment                         6          6
         7 Rohan Shah                44615-pharping                                         9823338934 STAFF          Male
  DebitCard Payment                        7          7
         8 Ronit waglay              44618-Rauthat                                          9823323934 VIP            Male
  DebitCard Payment                        8          8
         1 Rohit Raut                44614-kalimati-kathmandu                               9821108960 VIP            Male
  DebitCard Payment                        9          1

9 rows selected.
```

Figure 29: Answer of Information query

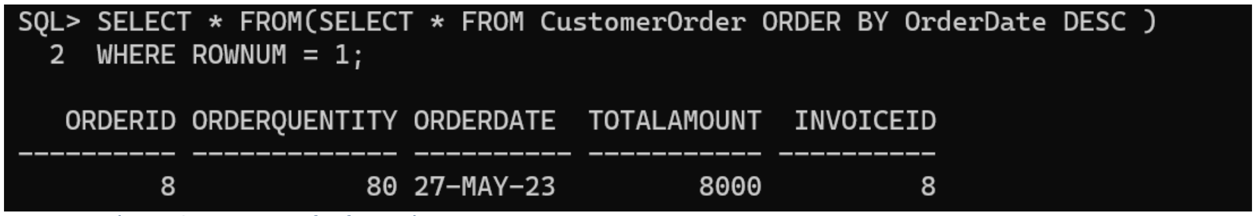
- List all the customers with their order details and also the customers who have not ordered any products yet.



```
SQL> SELECT * FROM Product WHERE SUBSTR(ProductName, 2,1)= 'a' AND ProductStock > 50;

 PRODUCTID PRODUCTCATEOGER PRODUCTNAME     PRODUCTSTOCK  UNITPRICE AVAILABILITYSTA   VENDORID
---------- --------------- --------------- ------------ ---------- --------------- ----------
         5 Accessories     Laptops                   59      80000 TRUE                     4
         8 Electronic      Tablets                   59      80000 TRUE                     7
         9 Accoessories    Tablets                   59      80000 TRUE                     2

SQL>
```

*Figure 30: Answer of Information query 4*

- Find out the customer who has ordered recently.

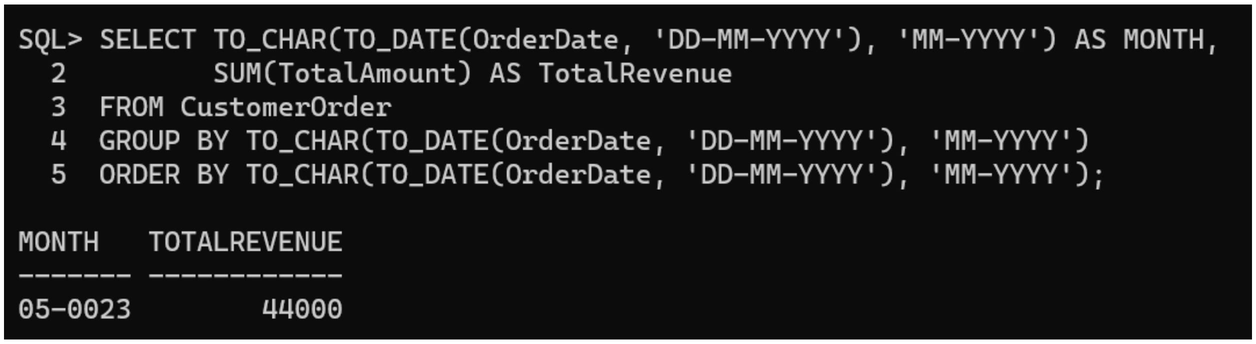


```
SQL> SELECT * FROM(SELECT * FROM CustomerOrder ORDER BY OrderDate DESC )
  2  WHERE ROWNUM = 1;

   ORDERID ORDERQUENTITY ORDERDATE  TOTALAMOUNT  INVOICEID
---------- ------------- ---------- ----------- ----------
         8            80 27-MAY-23         8000          8
```

*Figure 31:Answer of Information query 5.*

## 7.2. Transaction query

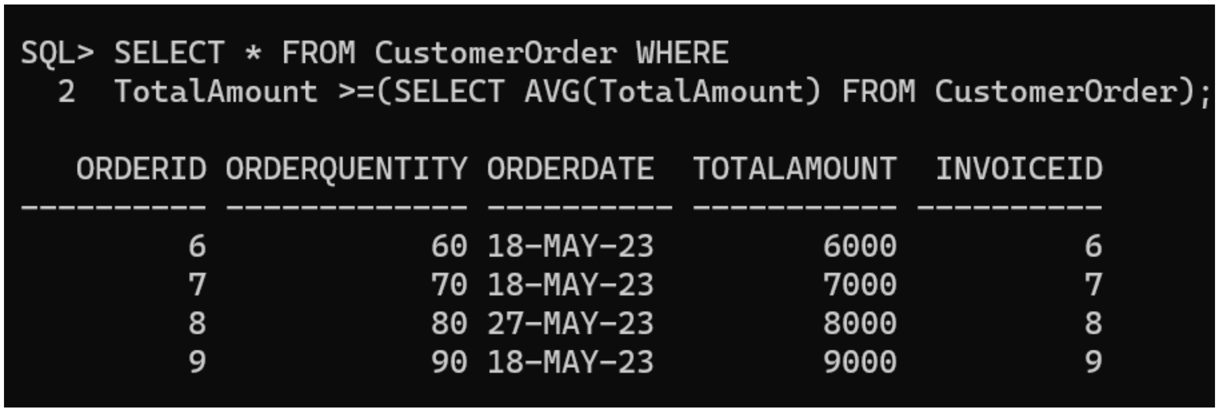
- Show the total revenue of the company for each month.



```
SQL> SELECT TO_CHAR(TO_DATE(OrderDate, 'DD-MM-YYYY'), 'MM-YYYY') AS MONTH,
  2         SUM(TotalAmount) AS TotalRevenue
  3  FROM CustomerOrder
  4  GROUP BY TO_CHAR(TO_DATE(OrderDate, 'DD-MM-YYYY'), 'MM-YYYY')
  5  ORDER BY TO_CHAR(TO_DATE(OrderDate, 'DD-MM-YYYY'), 'MM-YYYY');

MONTH   TOTALREVENUE
------- ------------
05-0023        44000
```

*Figure 32: Answer of transaction query 1*

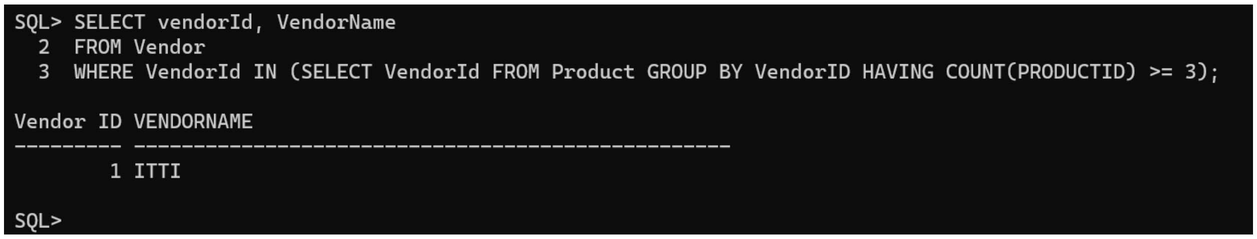
- Find those orders that are equal or higher than the average order total value.



```
SQL> SELECT * FROM CustomerOrder WHERE
  2  TotalAmount >=(SELECT AVG(TotalAmount) FROM CustomerOrder);

   ORDERID ORDERQUENTITY ORDERDATE   TOTALAMOUNT  INVOICEID
---------- ------------- ---------- ----------- ----------
         6            60 18-MAY-23         6000          6
         7            70 18-MAY-23         7000          7
         8            80 27-MAY-23         8000          8
         9            90 18-MAY-23         9000          9
```

Figure 33:Answer of transaction query 2

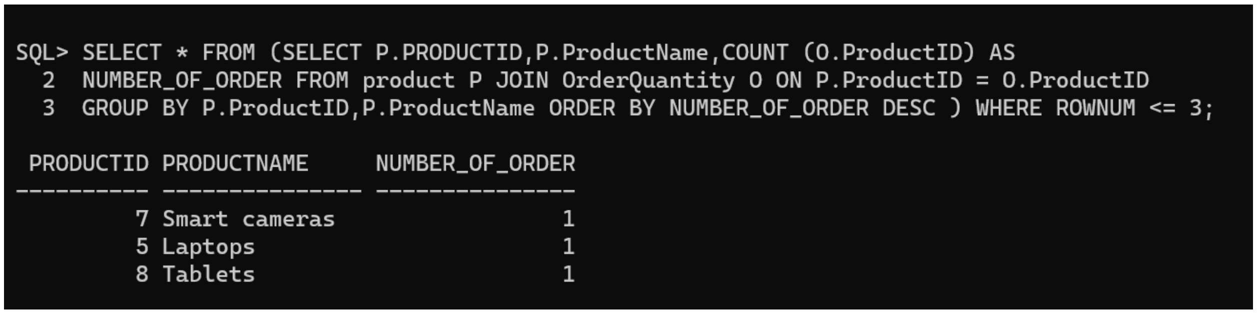
- List the details of vendors who have supplied more than 3 products to the company.



```
SQL> SELECT vendorId, VendorName
  2  FROM Vendor
  3  WHERE VendorId IN (SELECT VendorId FROM Product GROUP BY VendorID HAVING COUNT(PRODUCTID) >= 3);

Vendor ID VENDORNAME
--------- --------------------------------------------------
        1 ITTI

SQL>
```

Figure 34: Answer of transaction query 3

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



```
SQL> SELECT * FROM (SELECT P.PRODUCTID,P.ProductName,COUNT (O.ProductID) AS
  2  NUMBER_OF_ORDER FROM product P JOIN OrderQuantity O ON P.ProductID = O.ProductID
  3  GROUP BY P.ProductID,P.ProductName ORDER BY NUMBER_OF_ORDER DESC ) WHERE ROWNUM <= 3;

 PRODUCTID PRODUCTNAME     NUMBER_OF_ORDER
---------- --------------- ---------------
         7 Smart cameras                 1
         5 Laptops                       1
         8 Tablets                       1
```

Figure 35: Answer of transaction query 4

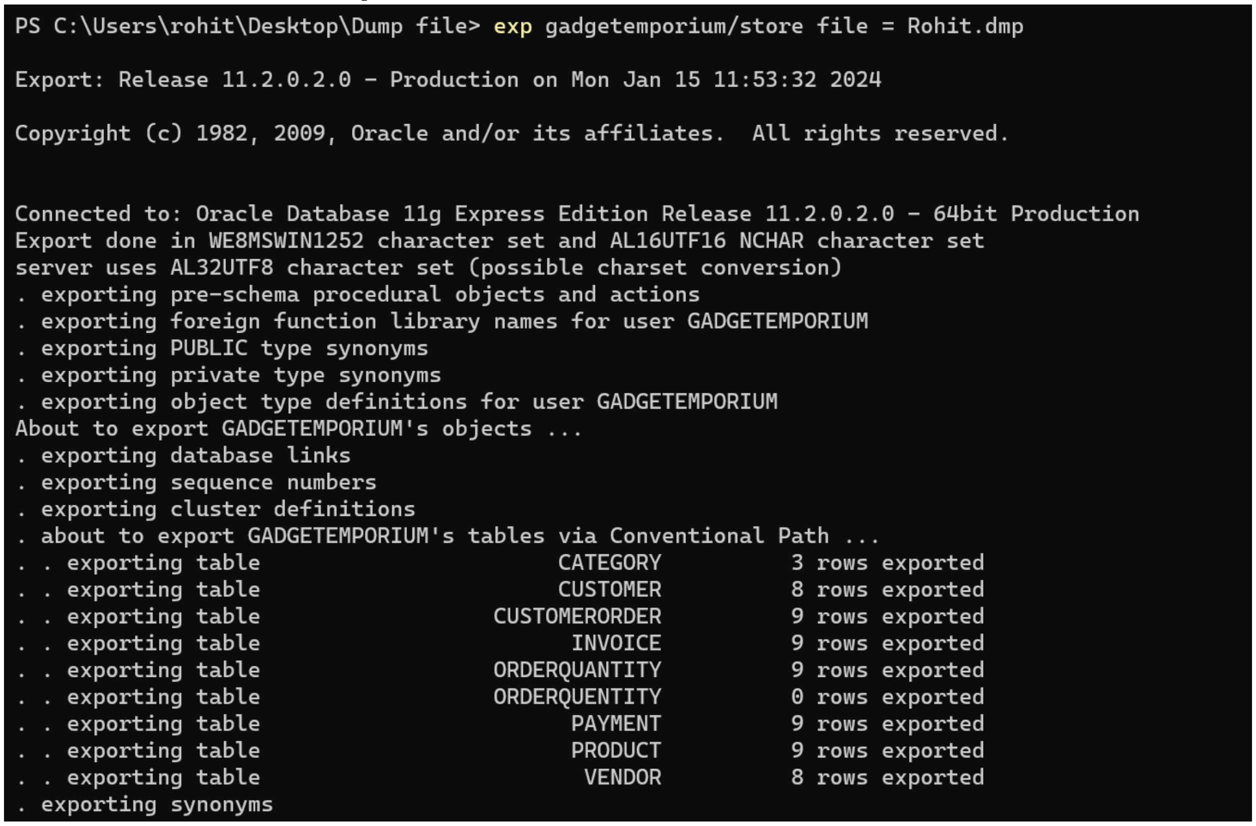
- Find out the customer who has ordered the most in August with his/her total spending on that month.

## 8. Critical Evaluation

The user reflects on a database course in the critical assessment part. The review discusses the value of databases in organizing and discovering data, with an emphasis on well-designed databases. It discusses the growth of cloud-based databases, emphasizing privacy and data control problems. The critical evaluation of coursework explains the difficulties encountered, the variety of answers provided, and the progress made in comprehending the subject.

In layman's words, the chat logs disclose discussions on various topics such as Java programming, coursework issues, and database assessments. Users exchange files, express thanks, and debate various elements of databases, underlining their importance in information organization. The critical examination comments on the significance of well-designed databases, as well as the problems and rewards they provide, highlighting the need of balancing technological use.

## 9. Database Dump File Creation



```
PS C:\Users\rohit\Desktop\Dump file> exp gadgetemporium/store file = Rohit.dmp

Export: Release 11.2.0.2.0 - Production on Mon Jan 15 11:53:32 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 GADGETEMPORIUM
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user GADGETEMPORIUM
About to export GADGETEMPORIUM's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export GADGETEMPORIUM's tables via Conventional Path ...
. . exporting table                       CATEGORY          3 rows exported
. . exporting table                       CUSTOMER          8 rows exported
. . exporting table                  CUSTOMERORDER          9 rows exported
. . exporting table                        INVOICE          9 rows exported
. . exporting table                  ORDERQUANTITY          9 rows exported
. . exporting table                  ORDERQUENTITY          0 rows exported
. . exporting table                        PAYMENT          9 rows exported
. . exporting table                        PRODUCT          9 rows exported
. . exporting table                         VENDOR          8 rows exported
. exporting synonyms
```

*Figure 36:Dump file creation*

## 10. Spool File creation

## 11. Drop Query

## 12. Conclusion

Completing this difficult database task has been a memorable trip for us. We ran into complicated normalization and SQL+ challenges that forced us to think critically and imaginatively. It's intriguing to see the various techniques each classmate took to answering the questions, highlighting the many opinions within our group. Conquering these hurdles gave us a sense of success while also expanding our awareness of the complexities inherent in managing databases in real-world circumstances.

The collaborative perseverance demonstrated in tackling these difficult tasks not only increased our knowledge but also built a sense of brotherhood among us. Working together on these challenging challenges created shared experiences that will last as we continue in our academic and professional lives. Despite the difficulties of the work, the sense of accomplishment.

Reflecting on the path as we complete this difficult database task gives us a sense of accomplishment. The difficulties we experienced were more than just impediments; they presented possibilities for intellectual growth, teamwork, and mutual learning. The obstacles of the task increased our group dynamic, forming relationships that go beyond databases and SQL queries. Despite the difficulties, our combined effort and the satisfying sense of comprehension and triumph have left an unforgettable impression on our academic experience. Moving forward, this shared experience becomes a treasured part of our academic story, providing as a reminder that success frequently results from working together to overcome problems.