LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: sumit pandit

London Met ID: 22067569

Assignment Submission Date: 15 January 2024

Word Count:

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

INTRODUCTION TO BUSINESS

Identification of Entities and Attributes

Entities	Attributes
Customer	**Customer_Id(PK)**, Customer_Name, Customer_Contact,Customer_Category_ID, Customer_Category,Discount_Rate
Order	**Order_Id(PK)**,Delivery_Status,Order_Date,Total_Amount, Purchased_Quantity, Purchased_Quantity_Price, Invoice_ID, Payment_Option,Payment_Status, Discount_Amount, Net_Amount,
Product	**Product_Id(PK)**,Product_Name, Product_Category, Description, Unit_Price, Stock_Level, Availibility_Status, Vendor_Id, Vendor_Name, Vendor_Contact

Initial ERD

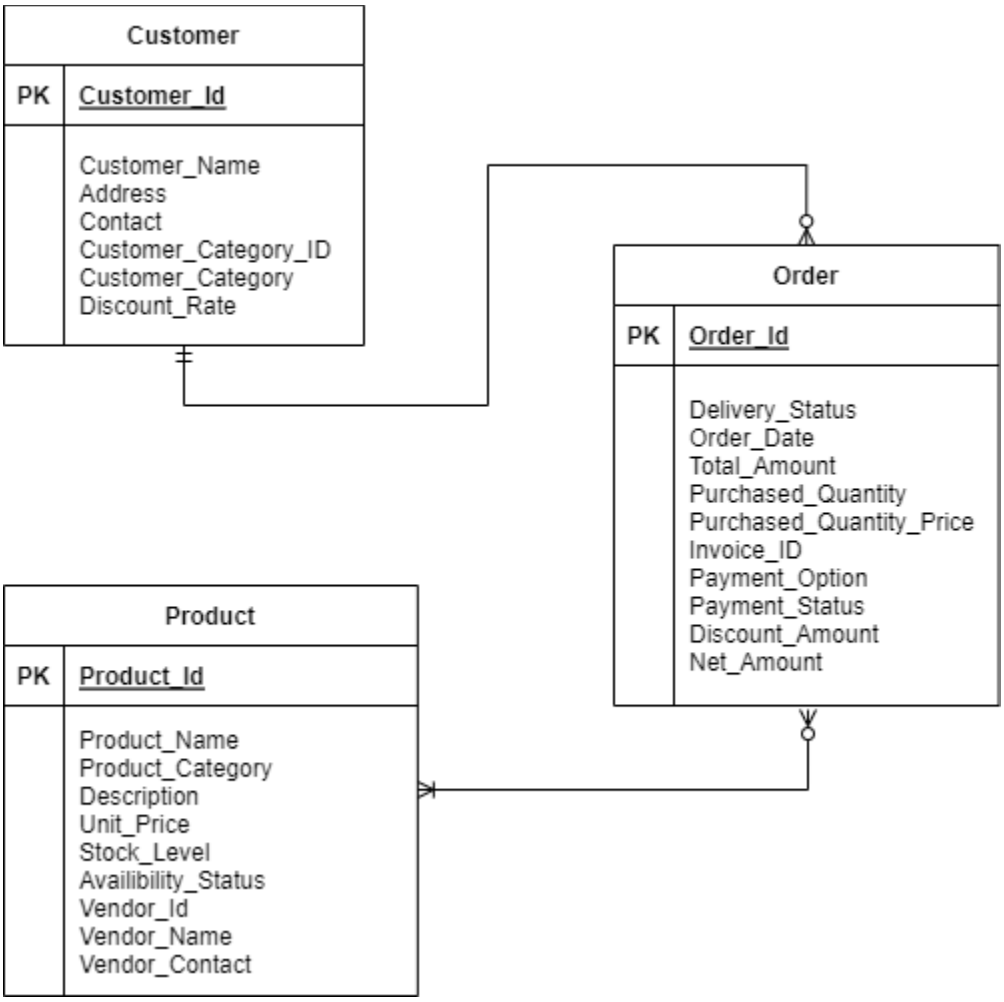

Figure 1: Initial Entity Relationship Diagram.

The above figure represents the Initial EDR of the whole Database. This is before normalization, so the entities and attributes present in the figure are not complete and will go through multiple changes during the normalization process. The following are the relation between these entities.

- A customer can make multiple orders or can also choose to not make any orders at all, and a single order can only belong to one customer.
- An order can have multiple or single product in it and a single product can belong to multiple order or none of the orders.

Normalization

UNF

Customer(Customer_Id, Customer_Name, Address, Contact, Customer_Category_Id, Customer_Category, Discount_Rate { Order_Id, Order_Date, Total_Amount, Invoice_ID, Payment_Option, Payment_Status, Discount_Amount, Net_Amount, Delivery_Status {Product_Id, Product_Name, Product_Category, Purchased_Quantity, Purchased_Quantity_Price, Description, Unit_Price, Stock_Level, Availability_Status, Vendor_Id, Vendor_Name, Vendor_Contact)

1NF

After removing all the repeating groups here is the final First normal form of the entities in our database.

Customer-1 (Customer_Id, Customer_Name, Address, Contact, Customer_Category_Id, Customer_Category, Discount_Rate)

Order-1 (Order_Id, Customer_Id, Order_Date, Total_Amount, Invoice_Id, Payment_Option, Payment_Status, Discount_Amount, Net_Amount, Delivery_Status)

Product-1 (Product_Id, Order_Id, Customer_Id, Product_Name, Product_Category, Purchased_Quantity, Purchased_Quantity_Price, Description,Unit_Price, Stock_Level, Availability_Status, Vendor_Id, Vendor_Name, Vendor_Contact)

2NF

From Customer-1 entity:

Customer_Id -> Customer_Name, Address, Contact,
Customer_Category_Id, Customer_Category, Discount_Rate

From Order entity:

Order_Id, Customer_Id -> X

Order_Id -> Order_Date, Total_Amount, Invoice_Id, Payment_Option,
Payment_Status, Discount_Amount, Net_Amount, Delivery_Status

From Product entity:

Product_Id, Order_Id, Customer_Id -> Purchased_Quantity,
Purchased_Quantity_Price

Product_ID -> Product_Name, Product_Category, Description,Unit_Price,
Stock_Level, Availability_Status, Vendor_Id, Vendor_Name,
Vendor_Contact

Final 2NF form:

After removing all the functional dependencies here is the final Second normal form of the entities in our database.

Customer-2 (Customer_Id, Customer_Name, Address, Contact,
Customer_Category_Id, Customer_Category, Discount_Rate)

Order-2 (Order_Id, Customer_Id)

Order_Details-2 (Order_Id, Order_Date, Total_Amount, Invoice_ID,
Payment_Option, Payment_Status,
Discount_Amount, Net_Amount, Delivery_Status)

Purchased_Product-2 (Product_Id, Order_Id, Customer_Id,
Purchased_Quantity,Purchased_Quantity_Price)

Product_Details-2 (Product_Id, Product_Name,Product_Category, Description,Unit_Price,Stock_Level,Availability_Status, Vendor_Id, Vendor_Name, Vendor_Contact)

3NF

Customer-2 (Customer_Id, Customer_Name, Address, Contact, Customer_Category_Id, Customer_Category, Discount_Rate)

Customer_Id -> Customer_Name, Address, Contact, Customer_Category_Id, Customer_Category, Discount_Rate

Customer_Category_Id -> Customer_Category, Discount_Rate

Order_Details-2 (Order_Id, Order_Date, Total_Amount, Invoice_ID, Payment_Option, Payment_Status, Discount_Amount, Net_Amount, Delivery_Status)

Order_Id -> Order_Date, Total_Amount, Invoice_ID, Payment_Option, Payment_Status, Discount_Amount, Net_Amount, Delivery_Status

Invoice_ID -> Payment_Option, Payment_Status, Discount_Amount, Net_Amount

Product_Details-2 (Product_Id, Product_Name,Product_Category, Description,Unit_Price,Stock_Level,Availability_Status, Vendor_Id, Vendor_Name, Vendor_Contact)

Product_Id -> Product_Name,Product_Category, Description,Unit_Price,Stock_Level,Availability_Status, Vendor_Id, Vendor_Name, Vendor_Contact

Vendor_Id -> Vendor_Name, Vendor_Contact

FINAL 3NF

After removing all the transative dependencies here is the final third normal form of the entities in our database.

Customer-3 (Customer_Id, Customer_Name, Address, Contact, Customer_Category_Id*)

Customer_Category-3 (Customer_Category_Id, Customer_Category, Discount_rate)

Order-3 (Order_Id, Customer_Id)

Order_Details-3 (Order_Id, Order_Date, Total_Amount, Delivery_Status, Invoice_ID*)

Invoice-3 (Invoice_ID, Payment_Option, Payment_Status, Discount_Amount, Net_Amount)

Purchase_Product-3 (Product_Id, Order_Id, Customer_Id, Purchased_Quantity,Purchased_Quantity_Price)

Purchase_Details-3 (Prodcut_Id, Product_Name, Product_Category, Description, Unit_Price, Stock_Level, Availability_Status, Vendor_Id*)

Vendor-3 (Vendor_Id, Vendor_Name, Vendor_Contact)

Final ERD

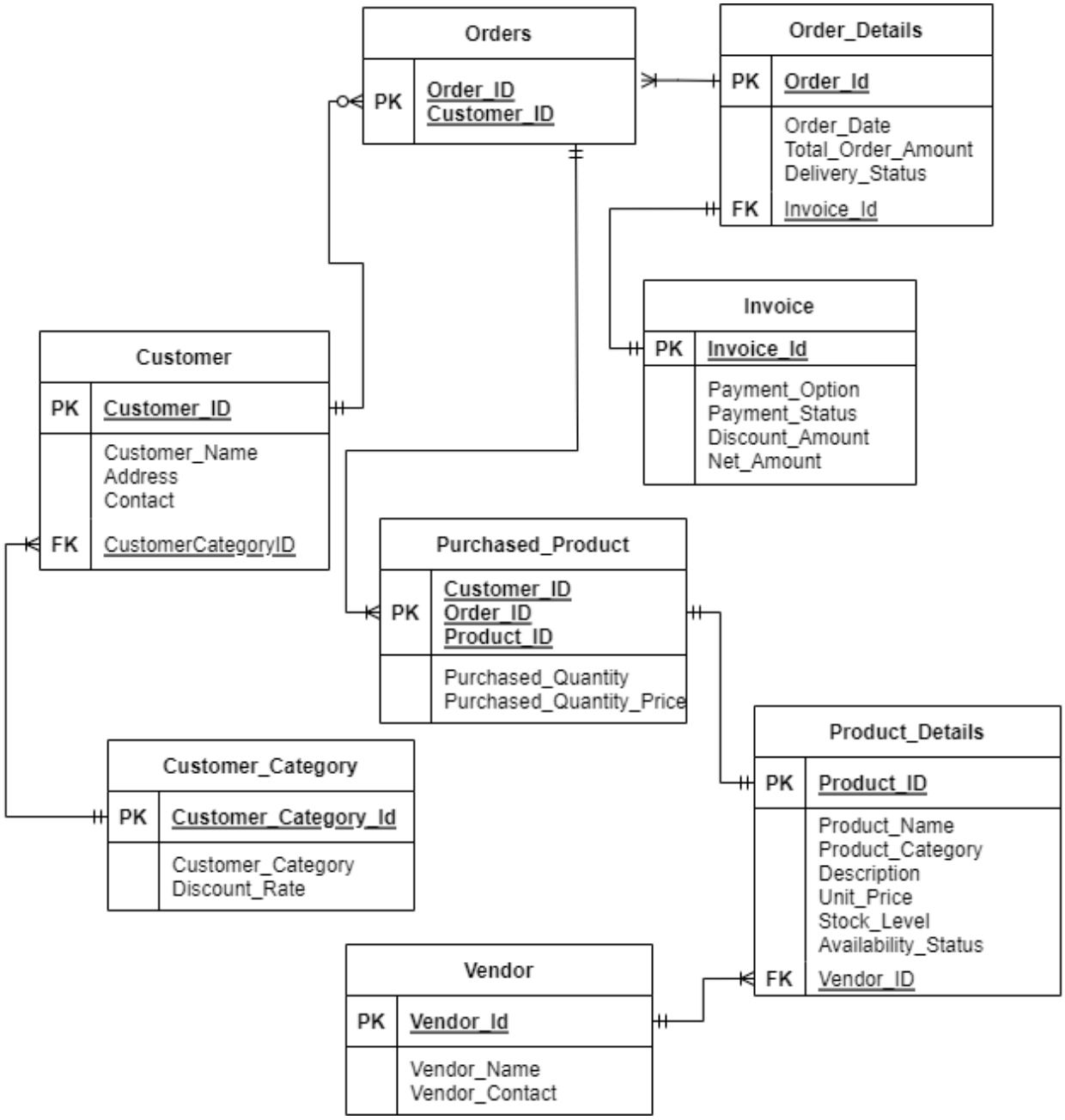

Figure 2: Final Entity Relationship Diagram.

The above diagram represents the Final Entity Relationship of the whole database after the normalization process. Here we can see that the three entities from before have been separated into more entities and properly related to each other by the constraints like primary and secondary keys. This EDR has the following relations between their entities:

- A Customer can belong to only one category at a time, but a category can have multiple or single customers at a time.
- A single customer can order multiple or zero time, but a single order can relate to only one customer.
- One order has only one unique order details, but the exact details can be same in multiple orders.
- One order creates only one invoice, and one invoice belongs to only one order.
- A single order can have single or multiple products in it, but a single purchased product can belong to only one order.
- One product can have a single description and a single product description belongs to only one product.
- One product can have only one vendor, but a vendor can provide one or more products.

Implementation

Creating Entities and Relations.

Creating Tables in SQL.

Customer Category Table in SQL.

First, the customer category table is created using the following code.

```
SQL> CREATE TABLE Customer_Category (
  2  Customer_Category_Id INT PRIMARY KEY NOT NULL,
  3  Customer_Category VARCHAR2(15) NOT NULL,
  4  Discount_Rate DECIMAL NOT NULL);

Table created.

SQL>
```

Figure 3: Creating Customer Category Table in SQL.

Now, to check if the table has been created successfully, I used the ‘DESC’ Command to describe the Customer Category table.

```
SQL> DESC Customer_Category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 CUSTOMER_CATEGORY_ID                      NOT NULL NUMBER(38)
 CUSTOMER_CATEGORY                         NOT NULL VARCHAR2(15)
 DISCOUNT_RATE                             NOT NULL NUMBER(38)

SQL>
```

Figure 4: Customer Category Table Properties.

Customer Table in SQL.

I used the following command to create the Customer table in Sql Plus.

```
SQL> CREATE TABLE Customer (
  2  Customer_Id INT PRIMARY KEY NOT NULL,
  3  Customer_Name VARCHAR2(30) NOT NULL,
  4  Address VARCHAR(25) NOT NULL,
  5  Contact INT NOT NULL,
  6  Customer_Category_Id INT NOT NULL,
  7  FOREIGN KEY (Customer_Category_Id) REFERENCES Customer_Category(Custome
r_Category_Id));

Table created.

SQL>
```

Figure 5: Creating Customer Table in SQL.

Using ‘DESC’ command to describe the created table.

```
SQL> DESC Customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
----
 CUSTOMER_ID                               NOT NULL NUMBER(38)
 CUSTOMER_NAME                             NOT NULL VARCHAR2(30)
 ADDRESS                                   NOT NULL VARCHAR2(25)
 CONTACT                                   NOT NULL NUMBER(38)
 CUSTOMER_CATEGORY_ID                      NOT NULL NUMBER(38)

SQL>
```

Figure 6: Customer Table Properties.

Orders Table in SQL.

I used the following command to create the Orders table in Sql Plus.

```
SQL> CREATE TABLE Orders (
  2  Order_Id INT NOT NULL,
  3  Customer_Id INT NOT NULL,
  4  PRIMARY KEY (Order_Id, Customer_Id));

Table created.
```

Figure 7: Creating Order Table in SQL.

Using ‘DESC’ command to describe the created table.

```
SQL> DESC Orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 ORDER_ID                                  NOT NULL NUMBER(38)
 CUSTOMER_ID                               NOT NULL NUMBER(38)

SQL>
```

Figure 8: Orders Table Properties.

Invoice Table in SQL.

I used the following command to create the Invoice table in Sql Plus.

```
SQL> CREATE TABLE Invoice (
  2  Invoice_Id INT PRIMARY KEY NOT NULL,
  3  Payment_Option VARCHAR(20) NOT NULL,
  4  Payment_Status VARCHAR(20) NOT NULL,
  5  Discount_Amount DECIMAL(10, 2) NOT NULL,
  6  Net_Amount DECIMAL(10, 2) NOT NULL
  7  );

Table created.

SQL>
```

Figure 9: Creating Invoice Table in SQL.

Using ‘DESC’ command to describe the created table.

```
SQL> DESC Invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 INVOICE_ID                                NOT NULL NUMBER(38)
 PAYMENT_OPTION                            NOT NULL VARCHAR2(20)
 PAYMENT_STATUS                            NOT NULL VARCHAR2(20)
 DISCOUNT_AMOUNT                           NOT NULL NUMBER(10,2)
 NET_AMOUNT                                NOT NULL NUMBER(10,2)

SQL>
```

Figure 10: Invoice Table Properties.

Order Details Table in SQL.

I used the following command to create the Order table in Sql Plus.

```
SQL> CREATE TABLE Order_Details (
  2  Order_Id INT PRIMARY KEY NOT NULL,
  3  Order_Date DATE NOT NULL,
  4  Total_Amount DECIMAL(10, 2) NOT NULL,
  5  Invoice_Id INT NOT NULL,
  6  Delivery_Status VARCHAR(20) NOT NULL,
  7  FOREIGN KEY (Invoice_Id) REFERENCES Invoice(Invoice_Id)
  8  );

Table created.

SQL>
```

Figure 11: Creating Order Details Table in SQL.

Using ‘DESC’ command to describe the created table.

```
SQL> DESC Order_Details;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 ORDER_ID                                  NOT NULL NUMBER(38)
 ORDER_DATE                                NOT NULL DATE
 TOTAL_AMOUNT                              NOT NULL NUMBER(10,2)
 INVOICE_ID                                NOT NULL NUMBER(38)
 DELIVERY_STATUS                           NOT NULL VARCHAR2(20)

SQL>
```

Figure 12: Order Details Table Properties.

Purchased Product Table in SQL.

I used the following command to create the Purchase Product table in Sql Plus.

```
SQL> CREATE TABLE Purchased_Product (
  2  Customer_Id INT NOT NULL ,
  3  Order_Id INT NOT NULL,
  4  Product_Id INT NOT NULL,
  5  Purchase_Quantity INT NOT NULL,
  6  Purchase_Quantity_Price DECIMAL(10, 2) NOT NULL,
  7  PRIMARY KEY (Customer_Id, Order_Id, Product_Id));

Table created.

SQL>
```

Figure 13: Creating Purchased Product Table in SQL.

Using ‘DESC’ command to describe the created table.

```
SQL> DESC Purchased_Product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 CUSTOMER_ID                               NOT NULL NUMBER(38)
 ORDER_ID                                  NOT NULL NUMBER(38)
 PRODUCT_ID                                NOT NULL NUMBER(38)
 PURCHASE_QUANTITY                         NOT NULL NUMBER(38)
 PURCHASE_QUANTITY_PRICE                   NOT NULL NUMBER(10,2)

SQL>
```

Figure 14: Purchased Product Table Properties.

Vendor Table in SQL.

I used the following command to create the Vendor table in Sql Plus.

```
SQL> CREATE TABLE Vendor (
  2  Vendor_Id INT PRIMARY KEY NOT NULL,
  3  Vendor_Name VARCHAR(20) NOT NULL,
  4  Vendor_Contact VARCHAR(20) NOT NULL
  5  );

Table created.

SQL>
```

Figure 15: Creating Vendor Table in SQL.

Using ‘DESC’ command to describe the created table.

```
SQL> DESC Vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 VENDOR_ID                                 NOT NULL NUMBER(38)
 VENDOR_NAME                               NOT NULL VARCHAR2(20)
 VENDOR_CONTACT                            NOT NULL VARCHAR2(20)

SQL>
```

Figure 16: Vendor Table Properties.

Product Details Table in SQL.

I used the following command to create the Product Details table in Sql Plus.

```
SQL> CREATE TABLE Product_Details (
  2  Product_ID INT PRIMARY KEY NOT NULL,
  3  Product_Name VARCHAR(25) NOT NULL,
  4  Product_Category VARCHAR(20) NOT NULL,
  5  Description VARCHAR(30) NOT NULL,
  6  Unit_Price DECIMAL(10, 2) NOT NULL,
  7  Stock_level INT NOT NULL,
  8  Availablity_Status VARCHAR(20) NOT NULL,
  9  Vendor_Id INT NOT NULL,
 10  FOREIGN KEY (Vendor_Id) REFERENCES Vendor(Vendor_Id)
 11  );

Table created.

SQL>
```

Figure 17: Creating Product Details Table in SQL.

Using ‘DESC’ command to describe the created table.

```
SQL> DESC Product_Details;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
----
 PRODUCT_ID                                NOT NULL NUMBER(38)
 PRODUCT_NAME                              NOT NULL VARCHAR2(25)
 PRODUCT_CATEGORY                          NOT NULL VARCHAR2(20)
 DESCRIPTION                               NOT NULL VARCHAR2(30)
 UNIT_PRICE                                NOT NULL NUMBER(10,2)
 STOCK_LEVEL                               NOT NULL NUMBER(38)
 AVAILABLITY_STATUS                        NOT NULL VARCHAR2(20)
 VENDOR_ID                                 NOT NULL NUMBER(38)

SQL>
```

Figure 18: Product Details Table Properties.

Inserting Data into Tables.

Inserting into Customer Category Table.

Inserting the following data into Customer Category table.

```
SQL> INSERT ALL INTO Customer_Category VALUES (1, 'Standard', 0)
  2  INTO Customer_Category VALUES (2, 'Staff', 5)
  3  INTO Customer_Category VALUES (3, 'VIP', 10)
  4  SELECT * FROM dual;

3 rows created.

SQL>
```

Figure 19: Inserting Data into Customer Category Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Customer_Category;

CUSTOMER_CATEGORY_ID CUSTOMER_CATEGO DISCOUNT_RATE
-------------------- -------------- -------------
                   1 Standard                   0
                   2 Staff                      5
                   3 VIP                       10
```

Figure 20: Checking Data in Customer Category Table.

Inserting into Customer Table.

Inserting the following data into Customer table.

```
SQL> INSERT ALL
  2  INTO Customer VALUES (1, 'Ram Kumar', 'Kathmandu', '9815435', 1)
  3  INTO Customer VALUES (2, 'Shyam Khadka', 'Pokhara', '9845612', 3)
  4  INTO Customer VALUES (3, 'Sishir Parajuli', 'Chitwan', '9846731', 2)
  5  INTO Customer VALUES (4, 'Prasiddha KC', 'Dhading', '9846531', 1)
  6  INTO Customer VALUES (5, 'Bikash Rokya', 'Hetuda', '9874651', 1)
  7  INTO Customer VALUES (6, 'Ravi Sapkota', 'Lalitpur', '9831645', 3)
  8  INTO Customer VALUES (7, 'Saurav BC', 'Chitwan', '9825641', 2)
  9  INTO Customer VALUES (8, 'Akshya Kumar', 'Nepalgunj', '9835164', 1)
 10  INTO Customer VALUES (9, 'Nikhil Upredi', 'Morang', '9845631', 3)
 11  INTO Customer VALUES (10, 'Aayush Khadka', 'Jhapa', '9895214', 1)
 12  SELECT * FROM dual;

10 rows created.
```

Figure 21: Inserting Data into Customer Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Customer;

CUSTOMER_ID CUSTOMER_NAME                ADDRESS                   CONTACT CUSTOMER_CATEGORY_ID
----------- ---------------------------- ------------------------- ---------- -------------------
          1 Ram Kumar                    Kathmandu                 9815435                     1
          2 Shyam Khadka                 Pokhara                   9845612                     3
          3 Sishir Parajuli              Chitwan                   9846731                     2
          4 Prasiddha KC                 Dhading                   9846531                     1
          5 Bikash Rokya                 Hetuda                    9874651                     1
          6 Ravi Sapkota                 Lalitpur                  9831645                     3
          7 Saurav BC                    Chitwan                   9825641                     2
          8 Akshya Kumar                 Nepalgunj                 9835164                     1
          9 Nikhil Upredi                Morang                    9845631                     3
         10 Aayush Khadka                Jhapa                     9895214                     1

10 rows selected.

SQL>
```

Figure 22: Checking Data in Customer Table.

Inserting into Orders Table.

Inserting the following data into Orders table.

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

7 rows created.

SQL>
```

Figure 23: Inserting Data into Orders Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Orders;

  ORDER_ID CUSTOMER_ID
---------- -----------
         1           2
         2           5
         3           7
         4           1
         5           3
         6           1
         7           2

7 rows selected.

SQL>
```

Figure 24: Checking Data in Orders Table.

Inserting into Invoice Table.

Inserting the following data into Invoice table.

```
SQL> INSERT ALL
  2  INTO Invoice VALUES (1, 'COD', 'Paid', 600, 5400)
  3  INTO Invoice VALUES (2, 'Online', 'Paid', 0, 2400)
  4  INTO Invoice VALUES (3, 'Online', 'Paid', 75, 1425)
  5  INTO Invoice VALUES (4, 'COD', 'Paid', 0, 3200)
  6  INTO Invoice VALUES (5, 'Online', 'Paid', 200, 3800)
  7  INTO Invoice VALUES (6, 'Online', 'Paid', 0, 1500)
  8  INTO Invoice VALUES (7, 'COD', 'Paid', 90, 810)
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 25: Inserting Data into Invoice Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Invoice;

INVOICE_ID PAYMENT_OPTION       PAYMENT_STATUS       DISCOUNT_AMOUNT NET_AMOUNT
---------- -------------------- -------------------- --------------- ----------
         1 COD                  Paid                             600       5400
         2 Online               Paid                               0       2400
         3 Online               Paid                              75       1425
         4 COD                  Paid                               0       3200
         5 Online               Paid                             200       3800
         6 Online               Paid                               0       1500
         7 COD                  Paid                              90        810

7 rows selected.

SQL>
```

Figure 26: Checking Data in Invoice Table.

Inserting into Order Details Table.

Inserting the following data into Order Details table.

```
SQL> INSERT ALL
  2  INTO Order_Details VALUES (1, TO_DATE('1/5/2023', 'DD/MM/YYYY'), 6000, 1, 'Delivered')
  3  INTO Order_Details VALUES (2, TO_DATE('12/5/2023', 'DD/MM/YYYY'), 2400, 2, 'Delivered')
  4  INTO Order_Details VALUES (3, TO_DATE('20/05/2023', 'DD/MM/YYYY'), 1500, 3, 'Delivered')
  5  INTO Order_Details VALUES (4, TO_DATE('30/5/2023', 'DD/MM/YYYY'), 3200, 4, 'Delivered')
  6  INTO Order_Details VALUES (5, TO_DATE('15/6/2023', 'DD/MM/YYYY'), 4000, 5, 'Delivered')
  7  INTO Order_Details VALUES (6, TO_DATE('22/06/2023', 'DD/MM/YYYY'), 1500, 6, 'Delivered')
  8  INTO Order_Details VALUES (7, TO_DATE('3/8/2023', 'DD/MM/YYYY'), 900, 7, 'Delivered')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 27: Inserting Data into Order Details Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Order_Details;

  ORDER_ID ORDER_DAT TOTAL_AMOUNT INVOICE_ID DELIVERY_STATUS
---------- --------- ------------ ---------- ---------------
         1 01-MAY-23         6000          1 Delivered
         2 12-MAY-23         2400          2 Delivered
         3 20-MAY-23         1500          3 Delivered
         4 30-MAY-23         3200          4 Delivered
         5 15-JUN-23         4000          5 Delivered
         6 22-JUN-23         1500          6 Delivered
         7 03-AUG-23          900          7 Delivered

7 rows selected.

SQL>
```

Figure 28: Checking Data in Order Details Table.

Inserting into Purchased Product Table.

Inserting the following data into Purchased Product table.

```
SQL> INSERT ALL
  2  INTO Purchased_Product VALUES (2, 1, 1, 3, 6000)
  3  INTO Purchased_Product VALUES (5, 2, 5, 2, 1600)
  4  INTO Purchased_Product VALUES (5, 2, 4, 1, 800)
  5  INTO Purchased_Product VALUES (7, 3, 6, 3, 1500)
  6  INTO Purchased_Product VALUES (1, 4, 4, 4, 3200)
  7  INTO Purchased_Product VALUES (3, 5, 1, 2, 4000)
  8  INTO Purchased_Product VALUES (1, 6, 2, 1, 1500)
  9  INTO Purchased_Product VALUES (2, 7, 3, 3, 900)
 10  SELECT * FROM dual;

8 rows created.

SQL>
```

Figure 29: Inserting Data into Purchased Product Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Purchased_Product;

CUSTOMER_ID   ORDER_ID PRODUCT_ID PURCHASE_QUANTITY PURCHASE_QUANTITY_PRICE
----------- ---------- ---------- ----------------- ----------------------
          2          1          1                 3                   6000
          5          2          5                 2                   1600
          5          2          4                 1                    800
          7          3          6                 3                   1500
          1          4          4                 4                   3200
          3          5          1                 2                   4000
          1          6          2                 1                   1500
          2          7          3                 3                    900

8 rows selected.

SQL>
```

Figure 30: Checking Data in Purchased Product Table.

Inserting into Vendor Table.

Inserting the following data into Vendor table.

```
SQL> INSERT ALL
  2  INTO Vendor VALUES (1, 'Apple', 'Apple@gmail.com')
  3  INTO Vendor VALUES (2, 'Samsung', 'Samsung@gmail.com')
  4  INTO Vendor VALUES (3, 'Sony', 'Sony@gmail.com')
  5  INTO Vendor VALUES (4, 'Microsoft', 'Microsoft@gmail.com')
  6  INTO Vendor VALUES (5, 'Dell', 'Dell@gmail.com')
  7  INTO Vendor VALUES (6, 'Canon', 'Canon@gmail.com')
  8  INTO Vendor VALUES (7, 'Acer', 'Acer@gmail.com')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 31: Inserting Data into Vendor Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Vendor;

 VENDOR_ID VENDOR_NAME          VENDOR_CONTACT
---------- -------------------- --------------------
         1 Apple                Apple@gmail.com
         2 Samsung              Samsung@gmail.com
         3 Sony                 Sony@gmail.com
         4 Microsoft            Microsoft@gmail.com
         5 Dell                 Dell@gmail.com
         6 Canon                Canon@gmail.com
         7 Acer                 Acer@gmail.com

7 rows selected.

SQL>
```

Figure 32: Checking Data in Vendor Table.

Inserting into Product Details Table.

Inserting the following data into Product Details table.

```
SQL> INSERT ALL
  2  INTO Product_Details VALUES (1, 'GalaxyS23', 'Phone', 'Samsung Galaxy S23', 2000, 55, 'Available', 2)
  3  INTO Product_Details VALUES (2, 'Iphone15', 'Phone', 'Apple Iphone15', 1500, 12, 'Available', 1)
  4  INTO Product_Details VALUES (3, 'Airpods', 'Headphones', 'Apple Airpods', 300, 58, 'Available', 1)
  5  INTO Product_Details VALUES (4, 'Ipad5', 'Tablet', 'Apple Ipad 5', 800, 33, 'Available', 1)
  6  INTO Product_Details VALUES (5, 'Inspron15', 'Laptop', 'Dell Inspron 15', 800, 44, 'Available', 5)
  7  INTO Product_Details VALUES (6, 'PS5', 'Gaming', 'Sony Playstation 5', 500, 15, 'Available', 3)
  8  INTO Product_Details VALUES (7, 'Surface15', 'Laptop', 'Microsoft Surface 15', 1200, 22, 'Available', 4)
  9  INTO Product_Details VALUES (8, 'M50', 'Camera', 'Canon M50', 900, 56, 'Available', 6)
 10  INTO Product_Details VALUES (9, 'Nitro5', 'Laptop', 'Acer Nitro 5 (2023)', 1500, 42, 'Available', 7)
 11  INTO Product_Details VALUES (10, 'AirTag', 'Others', 'Apple AirTag', 200, 111, 'Available', 1)
 12  SELECT * FROM dual;

10 rows created.

SQL>
```

Figure 33: Inserting Data into Product Details Table.

Using (Select * FROM table_name) command to see if the entered data is stored properly.

```
SQL> SELECT * FROM Product_Details;

PRODUCT_ID PRODUCT_NAME              PRODUCT_CATEGORY     DESCRIPTION                    UNIT_PRICE STOCK_LEVEL AVAILABLITY_STATUS    VENDOR_ID
---------- ------------------------- -------------------- ------------------------------ ---------- ----------- -------------------- ----------
         1 GalaxyS23                 Phone                Samsung Galaxy S23                   2000          55 Available                     2
         2 Iphone15                  Phone                Apple Iphone15                       1500          12 Available                     1
         3 Airpods                   Headphones           Apple Airpods                         300          58 Available                     1
         4 Ipad5                     Tablet               Apple Ipad 5                          800          33 Available                     1
         5 Inspron15                 Laptop               Dell Inspron 15                       800          44 Available                     5
         6 PS5                       Gaming               Sony Playstation 5                    500          15 Available                     3
         7 Surface15                 Laptop               Microsoft Surface 15                 1200          22 Available                     4
         8 M50                       Camera               Canon M50                             900          56 Available                     6
         9 Nitro5                    Laptop               Acer Nitro 5 (2023)                  1500          42 Available                     7
        10 AirTag                    Others               Apple AirTag                          200         111 Available                     1

10 rows selected.

SQL>
```

Figure 34: Checking Data in Product Details Table.

Data Querying

Information Querying

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

ANS: Here, I have entered the following command to find out the customer who are also staffs.

```
SQL> SELECT * FROM Customer
  2  WHERE Customer_Category_ID = (SELECT Customer_Category_ID FROM Customer_Category WHERE Customer_Category = 'Staff');

CUSTOMER_ID CUSTOMER_NAME                  ADDRESS                        CONTACT CUSTOMER_CATEGORY_ID
----------- ------------------------------ ------------------------------ ---------- --------------------
          3 Sishir Parajuli                Chitwan                        9846731                     2
          7 Saurav BC                      Chitwan                        9825641                     2

SQL>
```

Figure 35: Information Query no 1 Ans.

We can see that Sishir Parajuli and Saurav BC are the customer who are also staffs.

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

ANS: Here, I have entered the following command to find out the orders made for PS5 between the dates 01-05-2023 till 28- 05-2023.

```
SQL> SELECT * FROM Order_Details
  2  JOIN Purchased_Product ON Order_Details.Order_Id = Purchased_Product.Order_Id
  3  JOIN Product_Details ON Purchased_Product.Product_Id = Product_Details.Product_ID
  4  WHERE Order_Details.Order_Date BETWEEN TO_DATE('2023-05-01', 'YYYY-MM-DD') AND TO_DATE('2023-05-28', 'YYYY-MM-DD')
  5  AND Product_Details.Product_Name = 'PS5';

  ORDER_ID ORDER_DAT TOTAL_AMOUNT INVOICE_ID DELIVERY_STATUS      CUSTOMER_ID       ORDER_ID PRODUCT_ID PURCHASE_QUANTITY PURCHASE_QUANTITY_PRICE PRODUCT_ID PRODUCT_NAME               PRODUCT_CATEGORY     DESCRIPTION
      UNIT_PRICE STOCK_LEVEL AVAILABLITY_STATUS    VENDOR_ID
---------- --------- ------------ ---------- ------------------ ----------- ---------- ---------- ------------------ ---------------------- ---------- ------------------------- -------------------- --------------------
------ ---------- -------------------- ----------
         3 20-MAY-23         1500          3 Delivered
7              3         6              3                1500         6 PS5                       Gaming             Sony Playstation 5       500         15 Available                 3

SQL>
```

Figure 36: Information Query no 2 Ans.

We can see that order no 7 was made for PS5 between the dates.

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

ANS: Here, I have entered the following command to list all the customers with their order details and also the customers who have not ordered any products yet.

```
SQL> SELECT * FROM Customer
  2  LEFT JOIN Orders ON Customer.Customer_Id = Orders.Customer_Id
  3  LEFT JOIN Order_Details ON Orders.Order_Id = Order_Details.Order_Id
  4  ORDER BY Customer.Customer_Id;

CUSTOMER_ID CUSTOMER_NAME                  ADDRESS                    CONTACT CUSTOMER_CATEGORY_ID   ORDER_ID CUSTOMER_ID   ORDER_ID ORDER_DAT TOTAL_AMOUNT INVOICE_ID DELIVERY_STATUS
----------- ------------------------------ -------------------------- ---------- -------------------- ---------- ----------- ---------- --------- ------------ ---------- ------------------------------
          1 Ram Kumar                      Kathmandu                  9815435                    1          6           1          6 22-JUN-23         1500          6 Delivered
          1 Ram Kumar                      Kathmandu                  9815435                    1          4           1          4 30-MAY-23         3200          4 Delivered
          2 Shyam Khadka                   Pokhara                    9845612                    3          7           2          7 03-AUG-23          900          7 Delivered
          2 Shyam Khadka                   Pokhara                    9845612                    3          1           2          1 01-MAY-23         6000          1 Delivered
          3 Sishir Parajuli                Chitwan                    9846731                    2          5           3          5 15-JUN-23         4000          5 Delivered
          4 Prasiddha KC                   Dhading                    9846531                    1
          5 Bikash Rokya                   Hetuda                     9874651                    1          2           5          2 12-MAY-23         2400          2 Delivered
          6 Ravi Sapkota                   Lalitpur                   9831645                    3
          7 Saurav BC                      Chitwan                    9825641                    2          3           7          3 20-MAY-23         1500          3 Delivered
          8 Akshya Kumar                   Nepalgunj                  9835164                    1
          9 Nikhil Upredi                  Morang                     9845631                    3
         10 Aayush Khadka                  Jhapa                      9895214                    1

12 rows selected.

SQL>
```

Figure 37: Information Query no 3 Ans.

We can see that Customer with the ID no 4, 8, 9 and 10 haven't made any purchase and the rest has purchased at least one product from the shop.

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

ANS: Here, I have entered the following command to list all product details that have the second letter 'a' in their product name and have a stock quantity more than 50.

```
SQL> SELECT * FROM Product_Details
  2  WHERE SUBSTR(Product_Name, 2, 1) = 'a' AND Stock_Level > 50;

PRODUCT_ID PRODUCT_NAME               PRODUCT_CATEGORY     DESCRIPTION                    UNIT_PRICE STOCK_LEVEL AVAILABLITY_STATUS    VENDOR_ID
---------- ------------------------- -------------------- ------------------------------ ---------- ----------- -------------------- ----------
         1 GalaxyS23                 Phone                Samsung Galaxy S23                   2000          55 Available                     2

SQL>
```

Figure 38: Information Query no 4 Ans.

We can see that GalaxyS23 is the only product that has the second letter 'a' in its name and has a stock quantity more than 50.

5. Find out the customer who has ordered recently.

ANS: Here, I have entered the following command to find out the customer who has ordered recently.

```
SQL> SELECT * FROM Customer
  2  JOIN Orders ON Customer.Customer_Id = Orders.Customer_Id
  3   JOIN Order_Details ON Orders.Order_ID = Order_Details.Order_Id
  4   WHERE Order_Details.Order_Date >= (SELECT MAX(Order_Date) FROM Order_Details);

CUSTOMER_ID CUSTOMER_NAME                  ADDRESS                     CONTACT CUSTOMER_CATEGORY_ID   ORDER_ID CUSTOMER_ID   ORDER_ID ORDER_DAT TOTAL_AMOUNT INVOICE_ID DELIVERY_STATUS
----------- ------------------------------ ------------------------- ---------- -------------------- ---------- ----------- ---------- ---------- ------------ ---------- ------------------
          2 Shyam Khadka                   Pokhara                   9845612                    3          7           2          7 03-AUG-23          900          7 Delivered

SQL>
```

Figure 39: Information Query no 5 Ans.

We can see that Shyam Khadka is the customer who has ordered recently on 03-Aug-2023.

Transactional Querying

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

ANS: Here, I have entered the following command to show the total revenue of the company for each month.

```
SQL> SELECT TO_CHAR(Order_Details.Order_Date, 'MM-YYYY') AS Month,
  2  SUM(Invoice.Net_Amount) AS Total_Monthly_Revenue
  3  FROM Order_Details
  4  JOIN Invoice ON Order_Details.Invoice_Id = Invoice.Invoice_Id
  5  GROUP BY TO_CHAR(Order_Details.Order_Date, 'MM-YYYY')
  6  ORDER BY TO_CHAR(Order_Details.Order_Date, 'MM-YYYY');

MONTH   TOTAL_MONTHLY_REVENUE
------- --------------------
05-2023                12425
06-2023                 5300
08-2023                  810

SQL>
```

Figure 40: Transactional Query no 1 Ans.

We can see the total month for each month in the figure above.

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

   ANS: Here, I have entered the following command to find those orders that are equal or higher than the average order total value.

```
SQL> SELECT *FROM Order_Details
  2  WHERE Total_Amount >= (SELECT AVG(Total_Amount) FROM Order_Details);

  ORDER_ID ORDER_DAT TOTAL_AMOUNT INVOICE_ID DELIVERY_STATUS
---------- --------- ------------ ---------- --------------------
         1 01-MAY-23         6000          1 Delivered
         4 30-MAY-23         3200          4 Delivered
         5 15-JUN-23         4000          5 Delivered

SQL>
```

Figure 41 Transactional Query no 2 Ans.

We can see that the order with IDs 1, 4 and 5 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.**

   ANS: Here, I have entered the following command to List the details of vendors who have supplied more than 3 products to the company.

```
SQL> SELECT Vendor.*, COUNT(Product_Details.Product_Id) AS Total_Supplied_Products FROM Vendor
  2  JOIN Product_Details ON Vendor.Vendor_Id = Product_Details.Vendor_Id
  3  GROUP BY Vendor.Vendor_Id, Vendor.Vendor_Name, Vendor.Vendor_Contact
  4  HAVING COUNT(Product_Details.Product_Id) > 3;

 VENDOR_ID VENDOR_NAME          VENDOR_CONTACT       TOTAL_SUPPLIED_PRODUCTS
---------- -------------------- -------------------- -----------------------
         1 Apple                Apple@gmail.com                            4

SQL>
```

Figure 42 Transactional Query no 3 Ans.

We can see that Apple is the only vendor that supplies more than 3 products to our shop.

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

ANS: Here, I have entered the following command to show the top 3 product details that have been ordered the most.

```
SQL> SELECT *FROM (
  2  SELECT
  3  Product_Details.Product_Id,
  4  Product_Details.Product_Name,
  5  Product_Details.Product_Category,
  6  Product_Details.Description,
  7  Product_Details.Unit_Price,
  8  Product_Details.Stock_Level,
  9  Product_Details.Availablity_Status,
 10  Product_Details.Vendor_Id,
 11  COUNT(Purchased_Product.Order_Id) AS OrderCount
 12  FROM
 13  Purchased_Product
 14  JOIN
 15  Product_Details ON Purchased_Product.Product_Id = Product_Details.Product_Id
 16  GROUP BY
 17  Product_Details.Product_Id,Product_Details.Product_Name,
 18  Product_Details.Product_Category,Product_Details.Description,
 19  Product_Details.Unit_Price,Product_Details.Stock_Level,
 20  Product_Details.Availablity_Status, Product_Details.Vendor_Id
 21  ORDER BY
 22  OrderCount DESC)
 23  WHERE ROWNUM <= 3;

PRODUCT_ID PRODUCT_NAME              PRODUCT_CATEGORY     DESCRIPTION                    UNIT_PRICE STOCK_LEVEL AVAILABLITY_STATUS    VENDOR_ID ORDERCOUNT
---------- ------------------------- -------------------- ------------------------------ ---------- ----------- -------------------- ---------- ----------
         1 GalaxyS23                 Phone                Samsung Galaxy S23                   2000          55 Available                     2          2
         4 Ipad5                     Tablet               Apple Ipad 5                          800          33 Available                     1          2
         2 Iphone15                  Phone                Apple Iphone15                       1500          12 Available                     1          1

SQL>
```

Figure 43 Transactional Query no 4 Ans.

We can see that the product with product ID 1,4 and 2 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.

ANS: Here, I have entered the following command to find out the customer who has ordered the most in August with his/her total spending on that month.

```
SQL> SELECT * FROM (
  2  SELECT
  3  Customer.Customer_Id,
  4  Customer.Customer_Name,
  5  SUM(Invoice.Net_Amount) AS Total_Spending
  6  FROM Customer JOIN
  7  Purchased_Product ON Customer.Customer_Id = Purchased_Product.Customer_Id
  8  JOIN
  9  Order_Details ON Purchased_Product.Order_Id = Order_Details.Order_Id
 10  JOIN
 11  Invoice ON Order_Details.Invoice_Id = Invoice.Invoice_Id
 12  WHERE
 13  TO_CHAR(Order_Details.Order_Date, 'MM') = '08'
 14  GROUP BY
 15  Customer.Customer_Id, Customer.Customer_Name
 16  ORDER BY
 17  Total_Spending DESC)
 18  WHERE ROWNUM <= 1;

CUSTOMER_ID CUSTOMER_NAME                 TOTAL_SPENDING
----------- ------------------------------ --------------
          2 Shyam Khadka                              810

SQL>
```

Figure 44 Transactional Query no 5 Ans.

We can see that Shyam Khadka is the customer who has ordered the most in August with his spending of 810.