LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: MIRAJ DEEP BHANDARI

London Met ID: 22067814

Assignment Submission Date: 2024-01-14

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

Table of Contents:

Table of Figures:

Table of Tables:

1. INTRODUCTION:

In the ever-changing digital landscape, the demand for electronic devices and accessories continues to rise. Recognizing this market opportunity, entrepreneur and electronics enthusiast, Mr. John, starts an ambitious venture to create an online marketplace called "Gadget Emporium." This inclusive e-commerce platform aims to meet the diverse needs of both individual consumers and business organizations, offering them a wide selection of electronic gadgets and accessories. To ensure the success of this new business venture, a strong and comprehensive database system is crucial. This document outlines the main business rules and requirements that will guide the design and implementation of the Gadget Emporium database system.

1.1 Aims:

The primary aims for the implementation of the database for Gadget Emporium include:

- To create a conceptual data model that captures the essential business rules and relationships between various entities in Mr. John's e-commerce system.

- To design a logical database schema based on the conceptual data model, ensuring data integrity, consistency, and ease of retrieval.

- To implement the logical database schema using a suitable database management system (DBMS), configuring necessary tables, columns, constraints, and relationships.

- To develop appropriate database queries and reports to facilitate efficient data retrieval and analysis, supporting Mr. John's business operations and decision making.

1.2 Objectives:

The key objectives for enhancing the database infrastructure of Gadget Emporium are as follows:

- Design a comprehensive product management module to store detailed information about electronic gadgets and accessories, ensuring real-time updates on product names, descriptions, categories, prices, and stock levels.

- Develop a customer management module that categorizes customers (Regular, Staff, VIP), applies appropriate discounts, and captures customer addresses for efficient order delivery.

- Create an efficient order processing module that records detailed order information, including purchased products, quantities, unit prices, and total order amounts, while handling multiple products and orders seamlessly.

- Establish a vendor management module that maintains accurate records of suppliers, associates each product with a single vendor, and enables vendors to supply multiple products.

- Implement a dynamic product availability and inventory management module to track real-time product availability, prevent overselling, and maintain precise stock levels. Integrate diverse payment gateways for secure transactions, offering customers options like cash on delivery, credit/debit card, and e-wallet.

1.3 Current Business Activities and Operations:

Gadget Emporium runs an online shop that specializes in selling electronic devices and accessories. The products cover a variety of categories, including smartphones, laptops, tablets, wearables, gaming consoles, audio equipment, and more. The business takes pride in offering competitive prices, ensuring product availability, and providing excellent customer service. Gadget Emporium's day-to-day activities include managing product stock, processing customer orders, handling payments, and maintaining relationships with vendors and suppliers. With a strong focus on customer satisfaction, the company aims to become a leading player in the online electronics market.

1.3.1 BUSINESS RULES:

Business rules for a database are a set of guidelines or specifications that dictate how data should be handled, stored, processed, and managed within a business or organizational context. These rules help ensure consistency, accuracy, and integrity of data, and they often reflect the business logic and policies that govern the organization's operations (Amghar, Meziane and Flory, 2000).

Gadget Emporium has established its own set of business rules to conduct and manage its company activities and operations which are as follows :

- Each product must have attributes such as product ID, name, description, category, price, and stock level.
- Customers should be categorized as Regular (R), Staff (S), and VIP (V).
- Each customer category is associated with a specific discount rate on product purchases (0%, 5%, and 10% for Regular, Staff, and VIP respectively).

- Customer details should include the customer ID, name, address, and category
- Each order must record details such as order ID, customer ID, date, and total order amount.
- Each product must have inventory details like stock quantity or availability status.
- should Maintain records of vendors or suppliers providing electronic gadgets and accessories.
- The system should support various payment options, including cash on delivery, credit/debit card, and e-wallet.
- An invoice must be generated once the customer checks out their order after confirmation.
- The invoice should include details of the order, customer, and payment details (including applied discounts).

1.4 IDENTIFICATION OF ENTITIES AND ATTRIBUTES:

Based on the assumptions and rules I established for the initial Entity Relationship Diagram, we can identify the following entities and their respective attributes. It's important to note that these entities are not final, and additional ones may arise during the normalization process.

1.4.1 Product

This table contains the required information about the products and the vendors from which the products are supplied, along with the necessary attributes, such as:

product_id(pk), product_name, price, description, stock_quantities, category_id, category_name, vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact, order_id (fk).

Attribute	Data Type	Constraints	Description
product_id	NUMBER	PRIMARY KEY	This field records the unique id of the products.
product_name	VARCHAR2(30)	NOT NULL	This field records the name of the products.
price	VARCHAR2(10,2)	NOT NULL	This field records the unit price of the products.

description	VARCHAR2(255)	NOT NULL	Products Information are store in this description attribute.
stock_quantities	NUMBER	NOT NULL	This field records the numbers of available products which are in the stock.
category_id	NUMBER	NOT NULL	This field records the category ID associated with the product.
category_name	VARCHAR2(30)	NOT NULL	This attribute records the name of the product category.
vendor_id	NUMBER	NOT NULL	This field records the unique id of the vendors who supplies products.
vendor_name	VARCHAR2(30)	NOT NULL	This field records the name of the vendors who supplies products.
vendor_email	VARCHAR2(50)	NOT NULL	This field records the email of the vendors who supplies products.

vendor_address	VARCHAR2(20)	NOT NULL	This field records the address of the vendors.
vendor_contact	VARCHAR2(20)	UNIQUE , NOT NULL	Contact details of the vendors are stored in this vendor_contact attribute.
order_id	NUMBER	FOREIGN KEY, NOT NULL	This field records the order ID for the order made by the customer.

Table 1: Product table before normalization

1.4.2 Order

This table contains the required information about the orders placed by customers, along with the necessary attributes, such as:

order_id(pk), order_date, discount_amount, total_amount, final_amount, payment_method, order_quantities, line_total, customer_id(fk)

Attribute	Data Type	Constraints	Description
order_id	NUMBER	PRIMARY KEY	This field records the unique ID for the order made by the customer.
order_date	DATE	NOT NULL	This attribute records the date on which the customer places an order.
discount_amount	NUMBER(10,2)	NOT NULL	This field records the discount amount given to the customer while purchasing products.
total_amount	NUMBER(10,2)	NOT NULL	This field records the total price of the products without applying the discount.

final_amount	NUMBER(10,2)	NOT NULL	This field records the final price of the products after subtracting the total amount from the discount amount.
payment_method	VARCHAR2(20)	NOT NULL	This field records the method of payment made by the customer during the order.
order_quantities	NUMBER	NOT NULL	This field records the sum of the total product quantity done by the user in one order. For example, if the user orders 3 mobiles and 5 laptops, the order quantity will be 8.
line_total	NUMBER	NOT NULL	The "line_total" attribute records the total cost associated with a specific line item or product in an order. It is calculated by multiplying the product quantity by the unit price.

			For example, the price of coke is 50, and the customer buys 3. The line total would be (product quantity) * (unit price) = 3 * 50 = 150.
customer_id	NUMBER	FOREIGN KEY, NOT NULL	This field records the customer ID from which the order is made.

Table 2: order table before normalization

1.4.3 Customer

This table contains the required information about the customer with the necessary attributes, such as:

customer_id(pk), customer_name, discount, customer_address, customer_phone, customer_category.

Attribute	Data Type	Constraints	Description
customer_id	NUMBER	PRIMARY KEY	This field records the unique id of the customers.
customer_name	VARCHAR2(50)	NOT NULL	This field records the name of the customers.
customer_address	VARCHAR2(20)	NOT NULL	This field records the address of the customers.
customer_category	VARCHAR2(20)	NOT NULL	This field records the customer category either they are regular, staff or VIP.
customer_phone	VARCHAR2(20)	UNIQUE , NOT NULL	Contact details of the customers are store in this customer_phone attribute.
discount	NUMBER(3)	NOT NULL	This field records the discount given to the customer based on their category.

Table 3: customer table before normalization

2. ASSUMPTIONS:

- Each product belongs to only one category, and each category can have one or many products.

- An order can have multiple products and any one type of product might be included in multiple orders placed by various customers.

- Each product is associated with a single vendor, and each vendor can supply one or more products.

- Each order must be associated with one payment option.

- A customer can place no orders or multiple orders, and each order is associated with a single customer

3. INITIAL ERD:

An Entity-Relationship Diagram (ERD) is a visual representation of the data model that depicts entities, their attributes, and the relationships between entities in a database. ERDs are commonly used in database design and serve as a blueprint for designing and understanding the structure of a database (Casanova and Amaral de Sa, 1984).

Entities:

Entities are objects or concepts in the real world that can be identified and described. In a database context, entities are usually represented as tables. Each row in a table represents an instance of the entity, and each column represents an attribute of the entity (Visual Paradigm, 2019).

Attributes:

Attributes are properties or characteristics of entities. They describe the data that can be stored for each instance of an entity. Attributes are typically represented as columns in a table.

Relationships:

Relationships illustrate how entities are related to each other. There are different types of relationships, such as one-to-one, one-to-many, and many-to-many. Relationships are represented by connecting lines between the related entities, and they often have labels to indicate the nature of the relationship.

<u>List of Created Objects that we have created for Gadget Emporium:</u>

1) **Entity:** product

   Attributes:

   - product_id **(primary key)**
   - product_name
   - price
   - description
   - stock_quantities,
   - category_id
   - category_name
   - vendor_id
   - vendor_name
   - vendor_email
   - vendor_address
   - vendor_contact
   - order_id **(foreign key)**

2) **Entity:** order

Attributes:

- order_id **(primary key)**
- order_date
- discount_amount
- total_amount
- final_amount
- payment_method
- order_quantities
- line_total
- customer_id **(foreign key)**

3) **Entity:** customer

Attributes:

- customer_id **(primary key)**
- customer_name
- discount
- customer_address
- customer_phone
- customer_category

Initial ER-diagram is as follows:

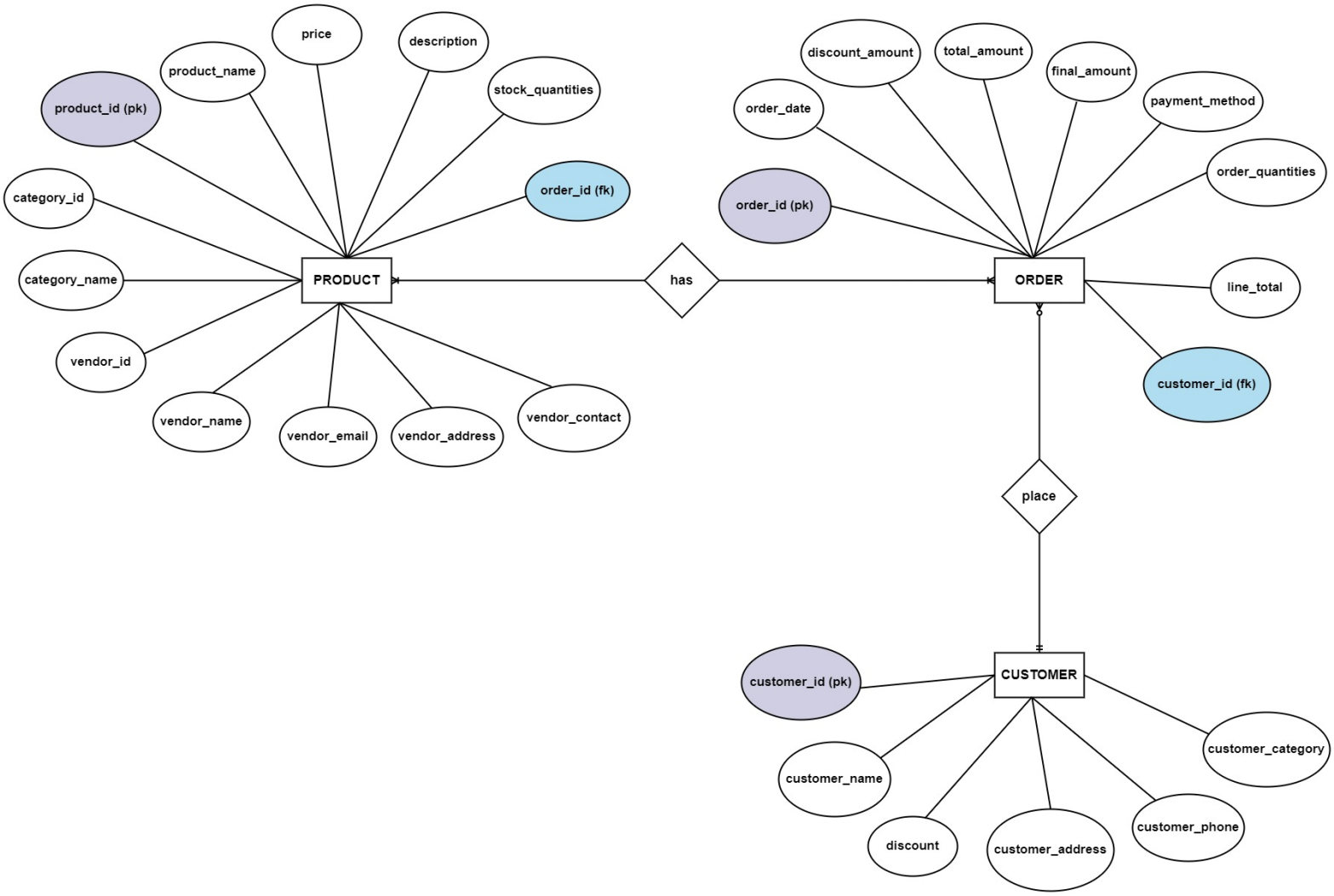

Figure 1: Initial Erd

4. NORMALIZATION:

Normalization involves dividing a table into multiple tables to eliminate insertion, update, and deletion anomalies. The normalization process breaks down larger tables into smaller ones, which are then linked through relationships. The primary goal of SQL normalization is to remove redundant (repetitive) data and ensure accurate data storage (Demba, 2013).

4.1 UNNORMALIZED FORM (UNF)

In the initial step of normalization, known as the Unnormalized Form (UNF), the focus is on compiling a list of attributes, with identification of the unique identifier, typically represented by an underline.

i. The objective is to maintain attributes in their most concise form without sacrificing their meaning.

ii. Repetitive groups should be enclosed within curly braces.

iii. The entity is assigned a specific name.

product(<u>product_id</u>, product_name, price, description, stock_quantities, category_id, category_name, vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact{order_id, order_date, discount_amount, total_amount, final_amount, payment_method, order_quantities, line_total, customer_id, customer_name, discount, customer_address, customer_phone, customer_category})

EXPLAINATION: Here, all the attributes that are in their smallest form are identified. An entity named "product" is created along with its repeating group of attributes in curly braces.

4.2 First Normal Form (1NF)

To achieve the first Normal Form, the repetitive group within curly braces should be extracted, and a unique identifier assigned. The table qualifies for the first Normal Form if it meets the following criteria:

i. Each attribute and column must possess a distinct name.

ii. Only single-valued attributes are allowed.

First step:

Separating the “product” with the key attribute (“product_id”)

- **product**(product_id, product_name, price, description, stock_quantities, category_id, category_name, vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact)

Second step:

Separating the repeating group “order” with the key attribute (“order_id”)

- **order**(product_id*, order_id, order_date, discount_amount, total_amount, final_amount, payment_method, order_quantities, line_total, customer_id, customer_name, discount, customer_address, customer_phone, customer_category)

RESULT:

- **product-1**(product_id, product_name, price, description, stock_quantities, category_id, category_name, vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact)

- **order-1**(product_id*, order_id, order_date, discount_amount, total_amount, final_amount, payment_method, order_quantities, line_total, customer_id, customer_name, discount, customer_address, customer_phone, customer_category)

4.3 Second Normal Form (2NF)

It is essential to examine and eliminate partial dependencies. The crucial point is that when all non-primary key attributes in the First Normal Form are entirely functionally dependent on the primary key, the relation is deemed to have achieved Second Normal Form (2NF).

For product-1:

The entity 'product' does not have a composite primary key, there is no possibility of partial dependency. Thus, 'product-1' is already in 2NF.

- **product**(product_id, product_name, price, description, stock_quantities, category_id, category_name, vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact)

<u>For order-1:</u>

There are two key attributes: **product_id, order_id**. As a result, there is a chance of partial dependency. By employing the formula (2n-1). We can have three types of relational dependencies. They are as follows:

- **product_id** ➡ **X** Doesn't give any attributes [Since, no attributes are dependent on product_id]

- **order_id** ➡ order_date, discount_amount, total_amount, final_amount, payment_method, customer_id, customer_name, discount, customer_address, customer_phone, customer_category (Partial Functional Dependency)

- **product_id, order_id** ➡ order_quantities, line_total (Fully Functional Dependency)

  [By combining both product_id and order_id it can give details order_quantities, line_total]

RESULT:

- **product-2**(product_id, product_name, price, description, stock_quantities, category_id, category_name, vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact)

- **order-2**(order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id, customer_name, discount, customer_address, customer_phone, customer_category)

- **productorder-2**(product_id**,** order_id, order_quantities, line_total)

EXPLAINATION: Here, all partial functional dependencies are examined and removed

4.4 Third Normal Form (3NF)

The transitive dependency has to be removed. Only if the table is on the 2NF and has no partial dependencies can it be considered to be in the 3NF.

For product-2 :

- **product_id** ➡ product_name, price, description, stock_quantities

- **product_id** ➡ category_id ➡ category_name (Transitive Dependency)
  - category_id ➡ category_name

[product_id can provide category_id. Since every category_id is unique so category_id can give category_name]

- **product_id** ➡ vendor_id ➡ vendor_name, vendor_email, vendor_address, vendor_contact (Transitive Dependency)
  - ❖ vendor_id ➡ vendor_name, vendor_email, vendor_address, vendor_contact

[product_id can provide vendor_id. Since every vendor_id is unique so vendor_id can give vendor_name, vendor_email, vendor_address, vendor_contact]

<u>For order-2:</u>

- **order_id** ➡ order_date, discount_amount, total_amount, final_amount, payment_method

- **order_id** ➡ customer_id ➡ customer_name, discount, customer_address, customer_phone, customer_category (Transitive Dependency)
  - ❖ customer_id ➡ customer_name, discount, customer_address, customer_phone, customer_category

[order_id can provide customer_id. Since every customer_id is unique so customer_id can give customer_name, discount, customer_address, customer_phone, customer_category]

<u>For productorder-2:</u>

It have only one non-key attributes. So, it is already in the 3NF

RESULT:

- **product-3**(product_id, product_name, price, description, stock_quantities, category_id *, vendor_id *)

- **category-3**(category_id, category_name)

- **vendor-3**(vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact)

- **order-3**(order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id *)

- **customer-3**(customer_id, customer_name, discount, customer_address, customer_phone, customer_category)

- **productorder-3**(product_id, order_id, line_total, order_quantities)

EXPLAINATION: Here, all transitivity dependencies are removed, and six different tables are created and converted from their unnormal to their normal forms.

5. FINAL ERD

The initial entity-relationship diagram (ERD) included features like repeating groups, repeating data, partial dependency, and transitive dependency. The ERD diagram below is the result of normalization, where the process has successfully eliminated all redundancy.

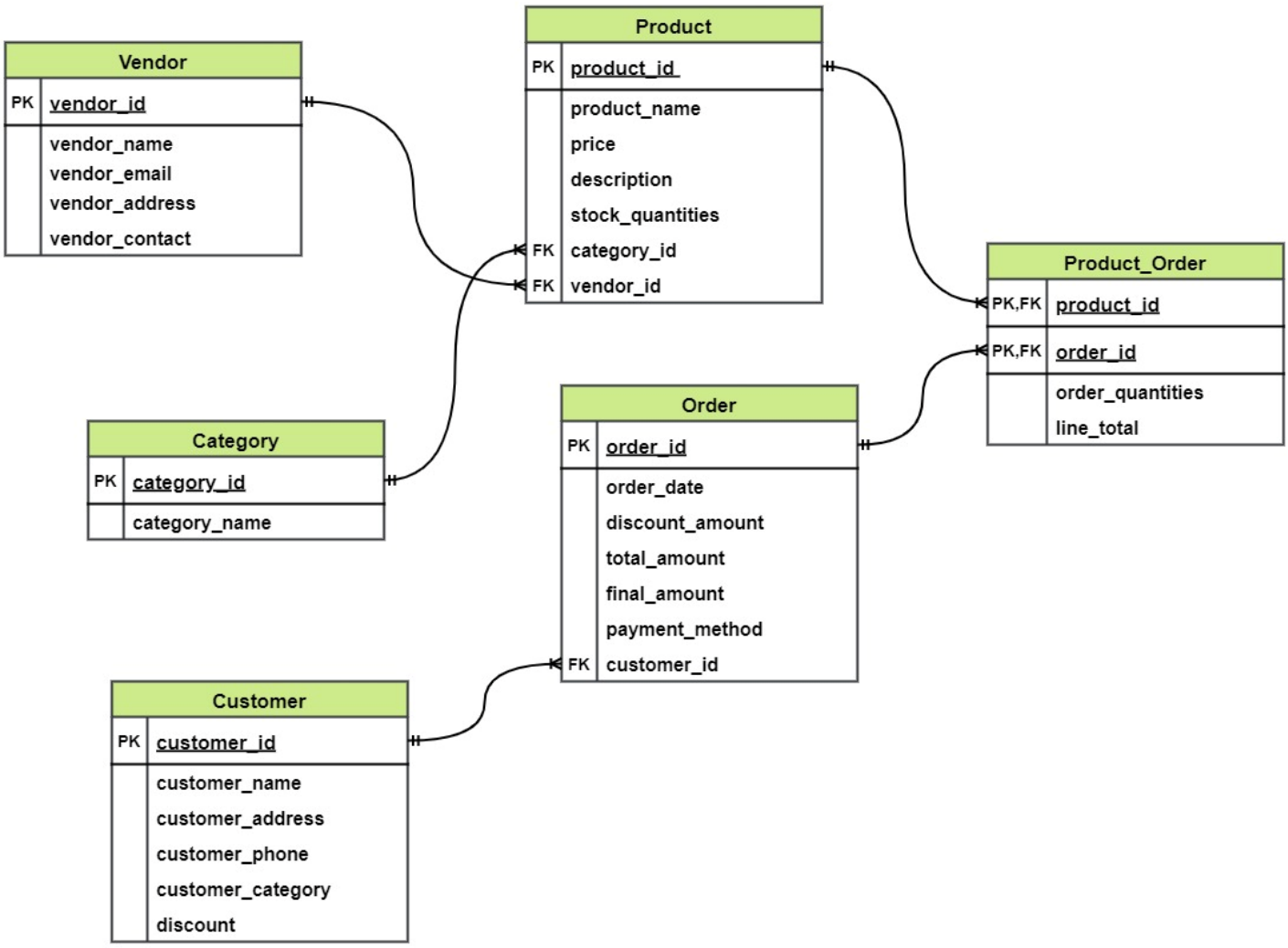

Figure 2: Final ERD

6. IMPLEMENTATIONS

6.1 Creating new user and giving Privileges:

```
SQL*Plus: Release 11.2.0.2.0 Production on Sat Dec 30 10:00:35 2023

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

SQL> conn system/miraj
Connected.
SQL> create user coursework identified by miraj;

User created.

SQL> Grant DBA to coursework;

Grant succeeded.

SQL>
```

Figure 3: Creating new user and giving Privileges

6.2 CREATING ENTITIES AND ESTABLISHING RELATIONS

6.2.1 vendor

vendor table after normalization:

Attribute	Data Type	Constraints	Description
vendor_id	NUMBER	PRIMARY KEY	This field records the unique id of the vendors who supplies products.
vendor_name	VARCHAR2(30)	NOT NULL	This field records the name of the vendors who supplies products.
vendor_email	VARCHAR2(50)	NOT NULL	This field records the email of the vendors who supplies products.
vendor_address	VARCHAR2(20)	NOT NULL	This field records the address of the vendors.

vendor_contact	VARCHAR2(20)	UNIQUE , NOT NULL	Contact details of the vendors are stored in this vendor_contact attribute.

Table 4: vendor table after normalization

CREATION OF VENDOR TABLE:

```
SQL> CREATE TABLE Vendor (
  2      vendor_id NUMBER PRIMARY KEY,
  3      vendor_name VARCHAR2(30) NOT NULL,
  4      vendor_email VARCHAR2(50) NOT NULL,
  5      vendor_address VARCHAR2(20) NOT NULL,
  6      vendor_contact VARCHAR2(20) UNIQUE NOT NULL
  7  );

Table created.
```

Figure 4: Screenshot of creating vendor table

DESCRIPTION OF VENDOR TABLE:

```
SQL> desc vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VENDOR_ID                                 NOT NULL NUMBER
 VENDOR_NAME                               NOT NULL VARCHAR2(30)
 VENDOR_EMAIL                              NOT NULL VARCHAR2(50)
 VENDOR_ADDRESS                            NOT NULL VARCHAR2(20)
 VENDOR_CONTACT                            NOT NULL VARCHAR2(20)

SQL>
```

Figure 5: Screenshot of description of vendor table

INSERTING DATA INTO VENDOR TABLE AND VIEWING THE OVERALL TABLE

```
SQL> INSERT ALL
  2    INTO Vendor (vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact) VALUES
  3      (100, 'ABC TV Nepal', 'ABC@gmail.com', 'Kalanki', '01-22334489')
  4    INTO Vendor (vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact) VALUES
  5      (101, 'LG Kitchen Suppliers', 'LGks@gmail.com', 'Koteshwor', '01-22334490')
  6    INTO Vendor (vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact) VALUES
  7      (102, 'XYZ Accessories Distributers', 'xyzAccessories@gmail.com', 'Boudha', '01-22334491')
  8    INTO Vendor (vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact) VALUES
  9      (103, 'Network House Nepal', 'NetworkHouseNepal@gmail.com', 'Thamel', '01-22334492')
 10    INTO Vendor (vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact) VALUES
 11      (104, 'GadgetGalaxy Nepal', 'gadgetgalaxy@gmail.com', 'Gongabu', '01-22334495')
 12    INTO Vendor (vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact) VALUES
 13      (106, 'Mudita Nepal', 'mudi@gmail.com', 'kavresthali', '01-22534489')
 14    INTO Vendor (vendor_id, vendor_name, vendor_email, vendor_address, vendor_contact) VALUES
 15      (107, 'serpan Suppliers', 'serpan@gmail.com', 'chabil', '01-22904490')
 16    SELECT * FROM dual;
```

Figure 6: Inserting the values into vendor table

```
SQL> select * from vendor;

 VENDOR_ID VENDOR_NAME                    VENDOR_EMAIL                                       VENDOR_ADDRESS       VENDOR_CONTACT
---------- ---------------------------- ------------------------------------------------- -------------------- --------------------
       107 serpan Suppliers               serpan@gmail.com                                  chabil               01-22904490
       100 ABC TV Nepal                   ABC@gmail.com                                     Kalanki              01-22334489
       101 LG Kitchen Suppliers           LGks@gmail.com                                    Koteshwor            01-22334490
       102 XYZ Accessories Distributers   xyzAccessories@gmail.com                          Boudha               01-22334491
       103 Network House Nepal            NetworkHouseNepal@gmail.com                       Thamel               01-22334492
       104 GadgetGalaxy Nepal             gadgetgalaxy@gmail.com                            Gongabu              01-22334495
       106 Mudita Nepal                   mudi@gmail.com                                    kavresthali          01-22534489

7 rows selected.

SQL>
```

Figure 7: Viewing the vendor table

6.2.2 Customer

customer table after normalization:

Attribute	Data Type	Constraints	Description
customer_id	NUMBER	PRIMARY KEY	This field records the unique id of the customers.
customer_name	VARCHAR2(50)	NOT NULL	This field records the name of the customers.
customer_address	VARCHAR2(20)	NOT NULL	This field records the address of the customers.
customer_phone	VARCHAR2(20)	UNIQUE , NOT NULL	Contact details of the customers are store in this customer_phone attribute.
customer_category	VARCHAR2(20)	NOT NULL	This field records the customer category either they are regular, staff or VIP.

discount	NUMBER(3)	NOT NULL	This field records the discount given to the customer based on their category.

Table 5: Customer table after normalization

CREATION OF CUSTOMER TABLE:

```
SQL> CREATE TABLE customer (
  2    customer_id NUMBER PRIMARY KEY,
  3    customer_name VARCHAR2(50) NOT NULL,
  4    discount NUMBER(3) NOT NULL CHECK (discount IN (0, 5, 10)),
  5    customer_address VARCHAR2(20) NOT NULL,
  6    customer_phone VARCHAR2(20) UNIQUE NOT NULL,
  7    customer_category VARCHAR2(20) NOT NULL
  8  );
```

Figure 8: Screenshot of creating customer table

DESCRIPTION OF CUSTOMER TABLE:

```
SQL> desc customer;
 Name                                                                                  Null?    Type
 ----------------------------------------------------------------------------------- -------- ----------------
 CUSTOMER_ID                                                                           NOT NULL NUMBER
 CUSTOMER_NAME                                                                         NOT NULL VARCHAR2(50)
 DISCOUNT                                                                              NOT NULL NUMBER(3)
 CUSTOMER_ADDRESS                                                                      NOT NULL VARCHAR2(20)
 CUSTOMER_PHONE                                                                        NOT NULL VARCHAR2(20)
 CUSTOMER_CATEGORY                                                                     NOT NULL VARCHAR2(20)

SQL>
```

Figure 9:Screenshot of description of customer table

INSERTING DATA INTO CUSTOMER TABLE AND VIEWING THE OVERALL TABLE

```
SQL> INSERT ALL
  2    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (1, 'Sita Basnet', 5, 'Kalanki', '9841245672', 'Regular')
  3    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (2, 'Ramesh Shahi', 10, 'Koteshwor', '9841345678', 'Staff')
  4    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (3, 'Gita Thapa', 0, 'Bajaju', '9841445673', 'VIP')
  5    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (4, 'Bibek Bista', 5, 'Lazimpat', '9841545674', 'Regular')
  6    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (5, 'Anita Rai', 10, 'Boudha', '9841645675', 'VIP')
  7    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (6, 'Kiran Shrestha', 0, 'Patan', '9841745676', 'Staff')
  8    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (7, 'Dilip Lama', 5, 'Jawalakhel', '9841845677', 'Regular')
  9    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (8, 'Meena Tamang', 10, 'Thamel', '9841945678', 'VIP')
 10    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (9, 'Raju Maharjan', 0, 'Gongabu', '9842045679', 'Regular')
 11    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (10, 'Anju Gurung', 5, 'New Baneshwor', '9842145680', 'Staff')
 12    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (11, 'Prakash Magar', 10, 'Kapan', '9842245681', 'VIP')
 13    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (12, 'Sumitra Joshi', 0, 'Durbarmarg', '9842345682', 'Regular')
 14    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (13, 'Sabin Rai', 5, 'Balaju', '9842445683', 'Regular')
 15    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (14, 'Pooja Khadka', 10, 'Dhapasi', '9842545684', 'VIP')
 16    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (15, 'Alok Shrestha', 0, 'Baneshwor', '9842645685', 'Staff')
 17    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (16, 'Sabina Thapa', 5, 'Jhamsikhel', '9842745686', 'Regular')
 18    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (17, 'Bishal Tamrakar', 10, 'Gwarko', '9842845687', 'VIP')
 19    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (18, 'Asmita Karki', 0, 'Sanepa', '9842945688', 'Staff')
 20    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (19, 'Roshan Shrestha', 5, 'Swayambhu', '9843045689', 'Regular')
 21    INTO Customer (customer_id, customer_name, discount, customer_address, customer_phone, customer_category) VALUES (20, 'Nisha Bhattarai', 10, 'Kuleshwor', '9843145690', 'VIP')
 22  SELECT * FROM dual;

20 rows created.

SQL>
```

Figure 10: Inserting the values into customer table

```
SQL> select * from CUSTOMER;

CUSTOMER_ID CUSTOMER_NAME                                      DISCOUNT CUSTOMER_ADDRESS     CUSTOMER_PHONE       CUSTOMER_CATEGORY
----------- -------------------------------------------------- ---------- -------------------- -------------------- --------------------
          1 Sita Basnet                                                 5 Kalanki              9841245672           Regular
          2 Ramesh Shahi                                               10 Koteshwor            9841345678           Staff
          3 Gita Thapa                                                  0 Bajaju               9841445673           VIP
          4 Bibek Bista                                                 5 Lazimpat             9841545674           Regular
          5 Anita Rai                                                  10 Boudha               9841645675           VIP
          6 Kiran Shrestha                                              0 Patan                9841745676           Staff
          7 Dilip Lama                                                  5 Jawalakhel           9841845677           Regular
          8 Meena Tamang                                               10 Thamel               9841945678           VIP
          9 Raju Maharjan                                               0 Gongabu              9842045679           Regular
         10 Anju Gurung                                                 5 New Baneshwor        9842145680           Staff
         11 Prakash Magar                                              10 Kapan                9842245681           VIP
         12 Sumitra Joshi                                               0 Durbarmarg           9842345682           Regular
         13 Sabin Rai                                                   5 Balaju               9842445683           Regular
         14 Pooja Khadka                                               10 Dhapasi              9842545684           VIP
         15 Alok Shrestha                                               0 Baneshwor            9842645685           Staff
         16 Sabina Thapa                                                5 Jhamsikhel           9842745686           Regular
         17 Bishal Tamrakar                                            10 Gwarko               9842845687           VIP
         18 Asmita Karki                                                0 Sanepa               9842945688           Staff
         19 Roshan Shrestha                                             5 Swayambhu            9843045689           Regular
         20 Nisha Bhattarai                                            10 Kuleshwor            9843145690           VIP

20 rows selected.

SQL>
```

Figure 11:Viewing the customer table

6.2.3 product

product table after normalization:

Attribute	Data Type	Constraints	Description
product_id	NUMBER	PRIMARY KEY	This field records the unique id of the products.
product_name	VARCHAR2(30)	NOT NULL	This field records the name of the products.
price	VARCHAR2(10,2)	NOT NULL	This field records the unit price of the products.
description	VARCHAR2(255)	NOT NULL	Products Information are store in this description attribute.
stock_quantities	NUMBER	NOT NULL	This field records the numbers of available products which are in the stock.

category_id	NUMBER	FOREIGN KEY, NOT NULL	This field records the category ID associated with the product.
vendor_id	NUMBER	FOREIGN KEY,NOT NULL	This field records the Vendor ID from which the product is supplied.

Table 6: product table after normalization

CREATION OF PRODUCT TABLE:

```
SQL> CREATE TABLE Product (
  2    product_id NUMBER PRIMARY KEY,
  3    price NUMBER(10, 2) NOT NULL CHECK (price >= 0),
  4    product_name VARCHAR2(30) NOT NULL,
  5    stock_quantities NUMBER NOT NULL CHECK (stock_quantities >= 0),
  6    vendor_id NUMBER,
  7    description VARCHAR2(255) NOT NULL,
  8    category_id NUMBER,
  9    CONSTRAINT fk_vendor FOREIGN KEY (vendor_id) REFERENCES Vendor(vendor_id),
 10    CONSTRAINT fk_category FOREIGN KEY (category_id) REFERENCES Category(category_id)
 11  );
```

Figure 12: Screenshot of creating product table

DESCRIPTION OF PRODUCT TABLE:

```
SQL> desc product
 Name                                                                                  Null?    Type
 ----------------------------------------------------------------------------------- -------- ----------------------
 PRODUCT_ID                                                                            NOT NULL NUMBER
 PRODUCT_NAME                                                                          NOT NULL VARCHAR2(30)
 PRICE                                                                                 NOT NULL NUMBER(10,2)
 DESCRIPTION                                                                           NOT NULL VARCHAR2(255)
 STOCK_QUANTITIES                                                                      NOT NULL NUMBER
 VENDOR_ID                                                                                      NUMBER
 CATEGORY_ID                                                                                    NUMBER

SQL>
```

Figure 13: Screenshot of description of product table

INSERTING DATA INTO PRODUCT TABLE AND VIEWING THE OVERALL TABLE

```
SQL> INSERT ALL
  2  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (1, 'Smart TV Stand', 99.99, 'Stylish stand for your smart TV', 50, 100, 200)
  3  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (2, 'Kitchen Blender', 49.99, 'Powerful blender for your kitchen needs', 30, 101, 201)
  4  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (3, 'HD Soundbar', 149.99, 'Enhance your audio experience with this soundbar', 20, 102, 202)
  5  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (4, 'Ethernet Cable', 9.99, 'High-speed network cable for reliable connections', 100, 103, 203)
  6  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (5, 'Smartphone Charger', 19.99, 'Fast-charging USB-C charger for smartphones', 75, 104, 204)
  7  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (6, 'Smart Watch', 129.99, 'Track your fitness and receive notifications on the go', 50, 100, 205)
  8  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (7, 'Home Decor Set', 79.99, 'Set of stylish home decor items', 40, 101, 206)
  9  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (8, 'Gaming Mouse', 29.99, 'Precision gaming mouse for immersive gaming experience', 60, 102, 202)
 10  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (9, 'Wireless Router', 79.99, 'High-performance wireless router for seamless connectivity', 25, 103, 203)
 11  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (10, 'Fitness Tracker', 49.99, 'Monitor your health and activities with this fitness tracker', 35, 104, 205)
 12  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (11, 'LED Desk Lamp', 34.99, 'Adjustable LED desk lamp for comfortable lighting', 45, 100, 206)
 13  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (12, 'Smart Refrigerator', 899.99, 'Refrigerator with smart features for modern kitchens', 15, 101, 201)
 14  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (13, 'Bluetooth Earphones', 59.99, 'Wireless Bluetooth earphones for on-the-go listening', 30, 102, 202)
 15  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (14, 'Home Security Camera', 129.99, 'Keep your home secure with this smart security camera', 20, 103, 203)
 16  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (15, 'Laptop Backpack', 39.99, 'Durable laptop backpack for everyday use', 50, 104, 206)
 17  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (16, 'Portable Speaker', 69.99, 'Compact portable speaker for on-the-go entertainment', 40, 100, 202)
 18  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (17, 'Smart Thermostat', 89.99, 'Control your home temperature with this smart thermostat', 25, 101, 206)
 19  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (18, 'USB Flash Drive', 14.99, 'Store and transfer data with this USB flash drive', 80, 102, 203)
 20  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (19, 'Digital Camera', 299.99, 'Capture high-quality images with this digital camera', 15, 103, 202)
 21  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (20, 'Smart Light Bulbs', 19.99, 'Adjustable smart light bulbs for customized lighting', 30, 104, 206)
 22  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (21, 'Office Chair', 129.99, 'Comfortable office chair for long hours of work', 35, 100, 206)
 23  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (22, 'Wireless Mouse', 19.99, 'Wireless mouse for convenient computer usage', 60, 101, 202)
 24  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (23, 'Smart Coffee Maker', 79.99, 'Brew your coffee with smart features', 25, 102, 201)
 25  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (24, 'Bluetooth Keyboard', 49.99, 'Wireless keyboard for easy typing', 40, 103, 202)
 26  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (25, 'External Hard Drive', 89.99, 'Store large amounts of data with this external hard drive', 20, 104, 203)
 27  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (26, 'Smart Doorbell', 149.99, 'Monitor your doorstep with this smart doorbell', 15, 100, 203)
 28  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (27, 'Ceramic Cookware Set', 129.99, 'High-quality ceramic cookware set for your kitchen', 30, 101, 201)
 29  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (28, 'Wireless Headphones', 79.99, 'Immersive wireless headphones for music lovers', 25, 102, 205)
 30  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (29, 'Smart Mirror', 199.99, 'Mirror with smart features for your daily routine', 10, 103, 206)
 31  INTO Product (product_id, product_name, price, description, stock_quantities, vendor_id, category_id) VALUES (30, 'Compact Digital Scale', 24.99, 'Compact digital scale for precise measurements', 50, 104, 206)
 32  SELECT * FROM dual;
```

Figure 14:Inserting the values into product table

```
SQL> select * from PRODUCT;

PRODUCT_ID PRODUCT_NAME                    PRICE DESCRIPTION                                                      STOCK_QUANTITIES VENDOR_ID CATEGORY_ID
---------- ------------------------------ ------- ---------------------------------------------------------------- ---------------- --------- -----------
        32 Motion Detector Alarm           49.99 Powerful security alarm for offices                                           30       107         206
         1 Smart TV Stand                  99.99 Stylish stand for your smart TV                                               50       100         200
         2 Kitchen Blender                 49.99 Powerful blender for your kitchen needs                                       30       101         201
         3 HD Soundbar                    149.99 Enhance your audio experience with this soundbar                              20       102         202
         4 Ethernet Cable                   9.99 High-speed network cable for reliable connections                           100       103         203
         5 Smartphone Charger              19.99 Fast-charging USB-C charger for smartphones                                   75       104         204
         6 Smart Watch                    129.99 Track your fitness and receive notifications on the go                        50       100         205
         7 Home Decor Set                  79.99 Set of stylish home decor items                                               40       101         206
         8 Gaming Mouse                    29.99 Precision gaming mouse for immersive gaming experience                        60       102         202
         9 Wireless Router                 79.99 High-performance wireless router for seamless connectivity                    25       103         203
        10 Fitness Tracker                 49.99 Monitor your health and activities with this fitness tracker                  35       104         205
        11 LED Desk Lamp                   34.99 Adjustable LED desk lamp for comfortable lighting                             45       100         206
        12 Smart Refrigerator             899.99 Refrigerator with smart features for modern kitchens                          15       101         201
        13 Bluetooth Earphones             59.99 Wireless Bluetooth earphones for on-the-go listening                          30       102         202
        14 Home Security Camera           129.99 Keep your home secure with this smart security camera                         20       103         203
        15 Laptop Backpack                 39.99 Durable laptop backpack for everyday use                                      50       104         206
        16 Portable Speaker                69.99 Compact portable speaker for on-the-go entertainment                          40       100         202
        17 Smart Thermostat                89.99 Control your home temperature with this smart thermostat                      25       101         206
        18 USB Flash Drive                 14.99 Store and transfer data with this USB flash drive                             80       102         203
        19 Digital Camera                 299.99 Capture high-quality images with this digital camera                          15       103         202
        20 Smart Light Bulbs               19.99 Adjustable smart light bulbs for customized lighting                          30       104         206
        21 Office Chair                   129.99 Comfortable office chair for long hours of work                               35       100         206
        22 Wireless Mouse                  19.99 Wireless mouse for convenient computer usage                                  60       101         202
        23 Smart Coffee Maker              79.99 Brew your coffee with smart features                                          25       102         201
        24 Bluetooth Keyboard              49.99 Wireless keyboard for easy typing                                             40       103         202
        25 External Hard Drive             89.99 Store large amounts of data with this external hard drive                     20       104         203
        26 Smart Doorbell                 149.99 Monitor your doorstep with this smart doorbell                                15       100         203
        27 Ceramic Cookware Set           129.99 High-quality ceramic cookware set for your kitchen                            30       101         201
        28 Wireless Headphones             79.99 Immersive wireless headphones for music lovers                                25       102         205
        29 Smart Mirror                   199.99 Mirror with smart features for your daily routine                             10       103         206
        30 Compact Digital Scale           24.99 Compact digital scale for precise measurements                                50       104         206
        31 Smart Bell                      99.99 Simple protection with smart technology for door                              50       106         206

32 rows selected.

SQL>
```

Figure 15:Viewing the product table

6.2.4 order

order table after normalization:

Attribute	Data Type	Constraints	Description
order_id	NUMBER	PRIMARY KEY	This field records the unique ID for the order made by the customer.
order_date	DATE	NOT NULL	This attribute records the date on which the customer places an order.
discount_amount	NUMBER(10,2)	NOT NULL	This field records the discount amount given to the customer while purchasing products.
total_amount	NUMBER(10,2)	NOT NULL	This field records the total price of the products without applying the discount.

final_amount	NUMBER(10,2)	NOT NULL	This field records the final price of the products after subtracting the total amount from the discount amount.
payment_method	VARCHAR2(20)	NOT NULL	This field records the method of payment made by the customer during the order.
customer_id	NUMBER	FOREIGN KEY,NOT NULL	This field records the customer ID from which the order is made.

Table 7: order table after normalization

CREATION OF ORDER TABLE:

```
SQL> CREATE TABLE OrderTable (
  2    order_id NUMBER PRIMARY KEY,
  3    customer_id NUMBER,
  4    discount_amount NUMBER(10, 2) NOT NULL CHECK (discount_amount >= 0),
  5    total_amount NUMBER(10, 2) NOT NULL CHECK (total_amount >= 0),
  6    order_date DATE NOT NULL,
  7    final_amount NUMBER(10, 2) NOT NULL CHECK (final_amount >= 0),
  8    payment_method VARCHAR2(20) NOT NULL,
  9    CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
 10  );
```

Figure 16: Screenshot of creating order table

DESCRIPTION OF ORDER TABLE:

```
Run SQL Command Line

SQL> desc ordertable;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_ID                                  NOT NULL NUMBER
 ORDER_DATE                                NOT NULL DATE
 DISCOUNT_AMOUNT                           NOT NULL NUMBER(10,2)
 TOTAL_AMOUNT                              NOT NULL NUMBER(10,2)
 FINAL_AMOUNT                              NOT NULL NUMBER(10,2)
 PAYMENT_METHOD                            NOT NULL VARCHAR2(20)
 CUSTOMER_ID                                        NUMBER

SQL>
```

Figure 17:Screenshot of description of order table

INSERTING DATA INTO ORDER TABLE AND VIEWING THE OVERALL TABLE

```
SQL> INSERT ALL
  2  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2000, TO_DATE('2023-01-15', 'YYYY-MM-DD'), 25.99, 519.94, 493.95, 'Esewa', 1)
  3  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2001, TO_DATE('2023-02-20', 'YYYY-MM-DD'), 4.99, 49.97, 44.98, 'Khalti', 2)
  4  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2002, TO_DATE('2023-03-10', 'YYYY-MM-DD'), 0.00, 389.97, 389.97, 'IME Pay', 3)
  5  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2003, TO_DATE('2023-04-05', 'YYYY-MM-DD'), 6.99, 139.97, 132.98, 'Cash', 4)
  6  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2004, TO_DATE('2023-05-01', 'YYYY-MM-DD'), 23.99, 239.97, 215.98, 'Esewa', 5)
  7  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2005, TO_DATE('2023-05-08', 'YYYY-MM-DD'), 0.00, 129.97, 129.97, 'Khalti', 3)
  8  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2006, TO_DATE('2023-05-15', 'YYYY-MM-DD'), 44.99, 899.99, 855, 'IME Pay', 7)
  9  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2007, TO_DATE('2023-05-25', 'YYYY-MM-DD'), 31.99, 319.97, 287.98, 'Cash', 2)
 10  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2008, TO_DATE('2023-05-27', 'YYYY-MM-DD'), 0.00, 119.97, 119.97, 'Esewa', 9)
 11  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2009, TO_DATE('2023-05-28', 'YYYY-MM-DD'), 12.49, 249.97, 237.48, 'Khalti', 4)
 12  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2010, TO_DATE('2023-06-03', 'YYYY-MM-DD'), 4.49, 44.97, 40.48, 'IME Pay', 11)
 13  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2011, TO_DATE('2023-06-08', 'YYYY-MM-DD'), 0.00, 339.97, 339.97, 'Cash', 12)
 14  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2012, TO_DATE('2023-06-10', 'YYYY-MM-DD'), 19.49, 389.97, 370.48, 'Esewa', 13)
 15  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2013, TO_DATE('2023-06-14', 'YYYY-MM-DD'), 17.99, 179.97, 161.98, 'Khalti', 14)
 16  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2014, TO_DATE('2023-07-20', 'YYYY-MM-DD'), 0.00, 149.97, 149.97, 'IME Pay', 9)
 17  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2015, TO_DATE('2023-07-25', 'YYYY-MM-DD'), 0.00, 389.97, 389.97, 'Cash', 9)
 18  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2016, TO_DATE('2023-08-05', 'YYYY-MM-DD'), 0.00, 389.97, 389.97, 'Esewa', 18)
 19  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2017, TO_DATE('2023-08-15', 'YYYY-MM-DD'), 0.00, 479.97, 479.97, 'Khalti', 18)
 20  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2018, TO_DATE('2023-08-22', 'YYYY-MM-DD'), 0.00, 74.97, 74.97, 'IME Pay', 18)
 21  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2019, TO_DATE('2023-08-28', 'YYYY-MM-DD'), 0.00, 204.97, 204.97, 'Cash', 18)
 22  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2020, TO_DATE('2023-08-29', 'YYYY-MM-DD'), 0.00, 239.97, 239.97, 'Esewa', 6)
 23  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2021, TO_DATE('2023-09-14', 'YYYY-MM-DD'), 19.49, 349.97, 330.48, 'Khalti', 13)
 24  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2022, TO_DATE('2023-10-20', 'YYYY-MM-DD'), 23.99, 239.97, 215.98, 'IME Pay', 2)
 25  INTO OrderTable (order_id, order_date, discount_amount, total_amount, final_amount, payment_method, customer_id) VALUES (2023, TO_DATE('2023-11-28', 'YYYY-MM-DD'), 0.00, 89.97, 89.97, 'Cash', 6)
 26  SELECT * FROM dual;

24 rows created.

SQL>
```

Figure 18:Inserting the values into order table

```
SQL> select * from ORDERTABLE;

  ORDER_ID ORDER_DAT DISCOUNT_AMOUNT TOTAL_AMOUNT FINAL_AMOUNT PAYMENT_METHOD       CUSTOMER_ID
---------- --------- --------------- ------------ ------------ -------------------- -----------
      2000 15-JAN-23           25.99       519.94       493.95 Esewa                          1
      2001 20-FEB-23            4.99        49.97        44.98 Khalti                         2
      2002 10-MAR-23               0       389.97       389.97 IME Pay                        3
      2003 05-APR-23            6.99       139.97       132.98 Cash                           4
      2004 01-MAY-23           23.99       239.97       215.98 Esewa                          5
      2005 08-MAY-23               0       129.97       129.97 Khalti                         3
      2006 15-MAY-23           44.99       899.99          855 IME Pay                        7
      2007 25-MAY-23           31.99       319.97       287.98 Cash                           2
      2008 27-MAY-23               0       119.97       119.97 Esewa                          9
      2009 28-MAY-23           12.49       249.97       237.48 Khalti                         4
      2010 03-JUN-23            4.49        44.97        40.48 IME Pay                       11
      2011 08-JUN-23               0       339.97       339.97 Cash                          12
      2012 10-JUN-23           19.49       389.97       370.48 Esewa                         13
      2013 14-JUN-23           17.99       179.97       161.98 Khalti                        14
      2014 20-JUL-23               0       149.97       149.97 IME Pay                        9
      2015 25-JUL-23               0       389.97       389.97 Cash                           9
      2016 05-AUG-23               0       389.97       389.97 Esewa                         18
      2017 15-AUG-23               0       479.97       479.97 Khalti                        18
      2018 22-AUG-23               0        74.97        74.97 IME Pay                       18
      2019 28-AUG-23               0       204.97       204.97 Cash                          18
      2020 29-AUG-23               0       239.97       239.97 Esewa                          6
      2021 14-SEP-23           19.49       349.97       330.48 Khalti                        13
      2022 20-OCT-23           23.99       239.97       215.98 IME Pay                        2
      2023 28-NOV-23               0        89.97        89.97 Cash                           6

24 rows selected.

SQL>
```

Figure 19:Viewing the order table

6.2.5 productorder

productorder table after normalization:

Attribute	Data Type	Constraints	Description
product_id	NUMBER	PRIMARY KEY, FOREIGN KEY	This field records the unique product ID assigned to each product chosen by the customer during the order process.
order_id	NUMBER	PRIMARY KEY, FOREIGN KEY	This attribute records the unique order ID assigned to each order placed by a customer.
line_total	NUMBER	NOT NULL	The "line_total" attribute records the total cost associated with a specific line item or product in an order. It is calculated by multiplying the product quantity by the unit price. For example, the price of

			coke is 50, and the customer buys 3. The line total would be **(product quantity) * (unit price)** = 3 * 50 = 150.
order_quantities	NUMBER	NOT NULL	This field records the sum of the total product quantity done by the user in one order. For example, if the user orders 3 mobiles and 5 laptops, the order quantity will be 8."

Table 8: productorder table after normalization

CREATION OF PRODUCTORDER TABLE

```
SQL> CREATE TABLE ProductOrder (
  2    product_id NUMBER,
  3    order_quantities NUMBER NOT NULL CHECK (order_quantities >= 0),
  4    line_total NUMBER NOT NULL,
  5    order_id NUMBER,
  6    CONSTRAINT fk_product FOREIGN KEY (product_id) REFERENCES Product (product_id),
  7    PRIMARY KEY (product_id, order_id),
  8    CONSTRAINT fk_order FOREIGN KEY (order_id) REFERENCES OrderTable(order_id)
  9  );
```

Figure 20:Screenshot of creating productorder table

DESCRIPTION OF PRODUCTORDER TABLE:

```
SQL> desc productorder;
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------------
 PRODUCT_ID                                NOT NULL NUMBER
 ORDER_ID                                  NOT NULL NUMBER
 ORDER_QUANTITIES                          NOT NULL NUMBER
 LINE_TOTAL                                NOT NULL NUMBER

SQL>
```

Figure 21:Screenshot of description of productorder table

INSERTING DATA INTO PRODUCTORDER TABLE AND VIEWING THE OVERALL TABLE

```
SQL> select * from PRODUCTORDER;

PRODUCT_ID   ORDER_ID ORDER_QUANTITIES LINE_TOTAL
---------- ---------- ---------------- ----------
         1       2000                2      19.98
         2       2000                1      49.99
         3       2000                3     449.97
         4       2001                1       9.99
         5       2001                2      39.98
         6       2002                3     389.97
         7       2003                1      79.99
         8       2003                2      59.98
         9       2004                3      79.99
        10       2005                1      49.99
        11       2005                2      79.98
        12       2006                1     899.99
        13       2007                1      59.99
        14       2007                2     259.98
        15       2008                3     119.97
        16       2009                1      69.99
        17       2009                2     179.98
        18       2010                3      44.97
        19       2011                1     299.99
        20       2011                2      39.98
        21       2012                3     389.97
        22       2013                1      19.99
        23       2013                2     159.98
        24       2014                3     149.97
        25       2015                1      89.99
        26       2015                2     299.98
        27       2016                3     389.97
        28       2017                1      79.99
        29       2017                2     399.98
        30       2018                3      74.97
        30       2019                1      24.99
        17       2019                2     179.98
        23       2020                3     239.97
        10       2021                1      49.99
         3       2021                2     299.98
         9       2022                3     239.97
        13       2023                1      59.99

37 rows selected.

SQL>
```

Figure 22:Viewing the productorder table

6.2.6 category

Category table after normalization:

Attribute	Data Type	Constraints	Description
category_id	NUMBER	PRIMARY KEY	This field records the unique identifier for product categories.
category_name	VARCHAR2(30)	NOT NULL	This attribute records the name of the product category.

Table 9: category table after normalization

CREATION OF CATEGORY TABLE

```
SQL> CREATE TABLE Category (
  2       category_id NUMBER PRIMARY KEY,
  3       category_name VARCHAR2(30) NOT NULL
  4  );

Table created.
```

Figure 23:Screenshot of creating category table

DESCRIPTION OF CATEGORY TABLE:

```
SQL> desc category;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------
 CATEGORY_ID                               NOT NULL NUMBER
 CATEGORY_NAME                             NOT NULL VARCHAR2(30)

SQL>
```

Figure 24: Screenshot of description of category table

INSERTING DATA INTO CATEGORY TABLE AND VIEWING THE OVERALL TABLE

```
SQL> select * from CATEGORY;

CATEGORY_ID CATEGORY_NAME
----------- ------------------------------
        200 TV Accessories
        201 Kitchen Products
        202 Audio and Video Devices
        203 Network Components
        204 Smart Phones
        205 Smart Watches
        206 Home Accessories

7 rows selected.

SQL>
```

Figure 25: Viewing the productorder table

7. Database Querying

7.1 Information query

a) List all the customers that are also staff of the company.

Query- select * from customer where CUSTOMER_CATEGORY= ‘Staff’ ;

```
SQL> select * from customer where CUSTOMER_CATEGORY='Staff';

CUSTOMER_ID CUSTOMER_NAME                                      DISCOUNT CUSTOMER_ADDRESS     CUSTOMER_PHONE       CUSTOMER_CATEGORY
----------- ------------------------------------------------- ---------- -------------------- -------------------- --------------------
          2 Ramesh Shahi                                              10 Koteshwor            9841345678           Staff
          6 Kiran Shrestha                                             0 Patan                9841745676           Staff
         10 Anju Gurung                                                5 New Baneshwor        9842145680           Staff
         15 Alok Shrestha                                              0 Baneshwor            9842645685           Staff
         18 Asmita Karki                                               0 Sanepa               9842945688           Staff

SQL>
```

Figure 26: Listing all customers who are staff

EXPLANATION- The query is used to fetch all the customers that are also staff of the company. The query selects all the rows from the customer table where the customer category is 'Staff'.

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

Query- SELECT o.ORDER_ID, o.ORDER_DATE, o.DISCOUNT_AMOUNT, o.TOTAL_AMOUNT, o.FINAL_AMOUNT, o.PAYMENT_METHOD, o.CUSTOMER_ID, p.PRODUCT_NAME FROM ordertable o JOIN productorder po ON o.ORDER_ID = po.ORDER_ID JOIN product p ON po.PRODUCT_ID = p.PRODUCT_ID

WHERE o.ORDER_DATE BETWEEN TO_DATE('01-05-2023', 'DD-MM-YYYY') AND TO_DATE('28-05-2023', 'DD-MM-YYYY');

```
SQL> SELECT o.ORDER_ID, o.ORDER_DATE, o.DISCOUNT_AMOUNT, o.TOTAL_AMOUNT, o.FINAL_AMOUNT,
  2         o.PAYMENT_METHOD, o.CUSTOMER_ID, p.PRODUCT_NAME
  3  FROM ordertable o
  4  JOIN productorder po ON o.ORDER_ID = po.ORDER_ID
  5  JOIN product p ON po.PRODUCT_ID = p.PRODUCT_ID
  6  WHERE o.ORDER_DATE BETWEEN TO_DATE('01-05-2023', 'DD-MM-YYYY') AND TO_DATE('28-05-2023', 'DD-MM-YYYY');

  ORDER_ID ORDER_DAT DISCOUNT_AMOUNT TOTAL_AMOUNT FINAL_AMOUNT PAYMENT_METHOD       CUSTOMER_ID PRODUCT_NAME
---------- --------- --------------- ------------ ------------ -------------------- ----------- ------------------------------
      2004 01-MAY-23           23.99       239.97       215.98 Esewa                           5 Wireless Router
      2005 08-MAY-23               0       129.97       129.97 Khalti                          3 Fitness Tracker
      2005 08-MAY-23               0       129.97       129.97 Khalti                          3 LED Desk Lamp
      2006 15-MAY-23           44.99       899.99          855 IME Pay                         7 Smart Refrigerator
      2007 25-MAY-23           31.99       319.97       287.98 Cash                            2 Bluetooth Earphones
      2007 25-MAY-23           31.99       319.97       287.98 Cash                            2 Home Security Camera
      2008 27-MAY-23               0       119.97       119.97 Esewa                           9 Laptop Backpack
      2009 28-MAY-23           12.49       249.97       237.48 Khalti                          4 Portable Speaker
      2009 28-MAY-23           12.49       249.97       237.48 Khalti                          4 Smart Thermostat

9 rows selected.

SQL>
```

EXPLANATION- The query shown in the image is used to select all the orders made for any particular product between the dates 01-05-2023 and 28-05-2023.

c) <u>List all the customers with their order details and also the customers who have not ordered any products yet.</u>

<u>To list all the customers who have ordered products</u>

Query- SELECT *FROM customer c INNER JOIN ordertable o ON c.customer_id = o.customer_id;

```
SQL> SELECT *
  2  FROM customer c
  3  INNER JOIN ordertable o ON c.customer_id = o.customer_id;
```

CUSTOMER_ID	CUSTOMER_NAME	DISCOUNT	CUSTOMER_ADDRESS	CUSTOMER_PHONE	CUSTOMER_CATEGORY	ORDER_ID	ORDER_DAT	DISCOUNT_AMOUNT	TOTAL_AMOUNT	FINAL_AMOUNT	PAYMENT_METHOD	CUSTOMER_ID
1	Sita Basnet	5	Kalanki	9841245672	Regular	2000	15-JAN-23	25.99	519.94	493.95	Esewa	1
2	Ramesh Shahi	10	Koteshwor	9841345678	Staff	2001	20-FEB-23	4.99	49.97	44.98	Khalti	2
3	Gita Thapa	0	Bajaju	9841445673	VIP	2002	10-MAR-23	0	389.97	389.97	IME Pay	3
4	Bibek Bista	5	Lazimpat	9841545674	Regular	2003	05-APR-23	6.99	139.97	132.98	Cash	4
5	Anita Rai	10	Boudha	9841645675	VIP	2004	01-MAY-23	23.99	239.97	215.98	Esewa	5
3	Gita Thapa	0	Bajaju	9841445673	VIP	2005	08-MAY-23	0	129.97	129.97	Khalti	3
7	Dilip Lama	5	Jawalakhel	9841845677	Regular	2006	15-MAY-23	44.99	899.99	855	IME Pay	7
2	Ramesh Shahi	10	Koteshwor	9841345678	Staff	2007	25-MAY-23	31.99	319.97	287.98	Cash	2
9	Raju Maharjan	0	Gongabu	9842045679	Regular	2008	27-MAY-23	0	119.97	119.97	Esewa	9
4	Bibek Bista	5	Lazimpat	9841545674	Regular	2009	28-MAY-23	12.49	249.97	237.48	Khalti	4
11	Prakash Magar	10	Kapan	9842245681	VIP	2010	03-JUN-23	4.49	44.97	40.48	IME Pay	11
12	Sumitra Joshi	0	Durbarmarg	9842345682	Regular	2011	08-JUN-23	0	339.97	339.97	Cash	12
13	Sabin Rai	5	Balaju	9842445683	Regular	2012	10-JUN-23	19.49	389.97	370.48	Esewa	13
14	Pooja Khadka	10	Dhapasi	9842545684	VIP	2013	14-JUN-23	17.99	179.97	161.98	Khalti	14
9	Raju Maharjan	0	Gongabu	9842045679	Regular	2014	20-JUL-23	0	149.97	149.97	IME Pay	9
9	Raju Maharjan	0	Gongabu	9842045679	Regular	2015	25-JUL-23	0	389.97	389.97	Cash	9
18	Asmita Karki	0	Sanepa	9842945688	Staff	2016	05-AUG-23	0	389.97	389.97	Esewa	18
18	Asmita Karki	0	Sanepa	9842945688	Staff	2017	15-AUG-23	0	479.97	479.97	Khalti	18
18	Asmita Karki	0	Sanepa	9842945688	Staff	2018	22-AUG-23	0	74.97	74.97	IME Pay	18
18	Asmita Karki	0	Sanepa	9842945688	Staff	2019	28-AUG-23	0	204.97	204.97	Cash	18
6	Kiran Shrestha	0	Patan	9841745676	Staff	2020	29-AUG-23	0	239.97	239.97	Esewa	6
13	Sabin Rai	5	Balaju	9842445683	Regular	2021	14-SEP-23	19.49	349.97	330.48	Khalti	13
2	Ramesh Shahi	10	Koteshwor	9841345678	Staff	2022	20-OCT-23	23.99	239.97	215.98	IME Pay	2
6	Kiran Shrestha	0	Patan	9841745676	Staff	2023	28-NOV-23	0	89.97	89.97	Cash	6

```
24 rows selected.
```

Figure 27: listing all the customers who have ordered products

EXPLANATION- The query shown in the image is used to list all the customers and order details of those who have ordered products.

To list all the customers who the customers who have not ordered any products yet.

Query- SELECT * FROM customer WHERE customer_id NOT IN (SELECT c.customer_id FROM customer c INNER JOIN ordertable o ON c.customer_id = o.customer_id);

```
SQL>  SELECT *
  2   FROM customer
  3   WHERE customer_id NOT IN (SELECT c.customer_id FROM customer c INNER JOIN ordertable o ON c.customer_id = o.customer_id);

CUSTOMER_ID CUSTOMER_NAME                                      DISCOUNT CUSTOMER_ADDRESS     CUSTOMER_PHONE       CUSTOMER_CATEGORY
----------- ------------------------------------------------ ---------- -------------------- -------------------- --------------------
          8 Meena Tamang                                             10 Thamel               9841945678           VIP
         15 Alok Shrestha                                             0 Baneshwor            9842645685           Staff
         19 Roshan Shrestha                                           5 Swayambhu            9843045689           Regular
         10 Anju Gurung                                               5 New Baneshwor        9842145680           Staff
         17 Bishal Tamrakar                                          10 Gwarko               9842845687           VIP
         16 Sabina Thapa                                              5 Jhamsikhel           9842745686           Regular
         20 Nisha Bhattarai                                          10 Kuleshwor            9843145690           VIP

7 rows selected.

SQL>
```

Figure 28:listing all the customers who have not ordered

EXPLANATION- The query shown in the image is used to list all the the customers who have not ordered any products yet.

d) <u>List all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.</u>

Query- SELECT * from product where product_name like '_a%' and STOCK_QUANTITIES >= 50;

```
SQL> select * from product where product_name like '_a%' and STOCK_QUANTITIES >= 50;

PRODUCT_ID PRODUCT_NAME                    PRICE DESCRIPTION                                                          STOCK_QUANTITIES VENDOR_ID CATEGORY_ID
---------- ------------------------------ ------- ------------------------------------------------------------------ ---------------- --------- -----------
         8 Gaming Mouse                    29.99 Precision gaming mouse for immersive gaming experience                           60       102         202
        15 Laptop Backpack                 39.99 Durable laptop backpack for everyday use                                         50       104         206

SQL>
```

EXPLANATION- The query shown in the image is used to list all product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.

e) <u>Find out the customer who has ordered recently.</u>

Query- SELECT CUSTOMER_ID, CUSTOMER_NAME
FROM customer
WHERE CUSTOMER_ID = (
  SELECT CUSTOMER_ID
  FROM (
    SELECT CUSTOMER_ID
    FROM ORDERTABLE
    ORDER BY ORDER_DATE DESC
)
  WHERE ROWNUM <= 1
);

```
SQL> select * from ordertable;

  ORDER_ID ORDER_DAT DISCOUNT_AMOUNT TOTAL_AMOUNT FINAL_AMOUNT PAYMENT_METHOD       CUSTOMER_ID
---------- --------- --------------- ------------ ------------ -------------------- -----------
      2000 15-JAN-23           25.99       519.94       493.95 Esewa                          1
      2001 20-FEB-23            4.99        49.97        44.98 Khalti                         2
      2002 10-MAR-23               0       389.97       389.97 IME Pay                        3
      2003 05-APR-23            6.99       139.97       132.98 Cash                           4
      2004 01-MAY-23           23.99       239.97       215.98 Esewa                          5
      2005 08-MAY-23               0       129.97       129.97 Khalti                         3
      2006 15-MAY-23           44.99       899.99          855 IME Pay                        7
      2007 25-MAY-23           31.99       319.97       287.98 Cash                           2
      2008 27-MAY-23               0       119.97       119.97 Esewa                          9
      2009 28-MAY-23           12.49       249.97       237.48 Khalti                         4
      2010 03-JUN-23            4.49        44.97        40.48 IME Pay                       11
      2011 08-JUN-23               0       339.97       339.97 Cash                          12
      2012 10-JUN-23           19.49       389.97       370.48 Esewa                         13
      2013 14-JUN-23           17.99       179.97       161.98 Khalti                        14
      2014 20-JUL-23               0       149.97       149.97 IME Pay                        9
      2015 25-JUL-23               0       389.97       389.97 Cash                           9
      2016 05-AUG-23               0       389.97       389.97 Esewa                         18
      2017 15-AUG-23               0       479.97       479.97 Khalti                        18
      2018 22-AUG-23               0        74.97        74.97 IME Pay                       18
      2019 28-AUG-23               0       204.97       204.97 Cash                          18
      2020 29-AUG-23               0       239.97       239.97 Esewa                          6
      2021 14-SEP-23           19.49       349.97       330.48 Khalti                        13
      2022 20-OCT-23           23.99       239.97       215.98 IME Pay                        2
      2023 28-NOV-23               0        89.97        89.97 Cash                           6

24 rows selected.

SQL> SELECT CUSTOMER_ID,CUSTOMER_NAME
  2  FROM customer
  3  WHERE CUSTOMER_ID = (
  4      SELECT CUSTOMER_ID
  5      FROM (
  6          SELECT CUSTOMER_ID
  7          FROM ORDERTABLE
  8          ORDER BY ORDER_DATE DESC
  9      )
 10      WHERE ROWNUM <= 1
 11  );

CUSTOMER_ID CUSTOMER_NAME
----------- --------------------------------------------------
          6 Kiran Shrestha
```

Figure 29:displaying the customer who has recently placed an order

EXPLANATION- The query shown in the image is used to display the customer who has recently placed an order. In this case, **Kiran Shrestha**, with **customer ID 6**, has placed an order on the date **2023-November-28.**

7.2 Transaction query

a) Show the total revenue of the company for each month.

Query- SELECT TO_CHAR(ORDER_DATE, 'MON-YYYY') AS MONTH_YEAR,
SUM(FINAL_AMOUNT) AS TOTAL_REVENUE
FROM ordertable
GROUP BY TO_CHAR(ORDER_DATE, 'MON-YYYY')
ORDER BY TO_DATE(TO_CHAR(ORDER_DATE, 'MON-YYYY'), 'MON-YYYY');

```
SQL> SELECT TO_CHAR(ORDER_DATE, 'MON-YYYY') AS MONTH_YEAR,
  2         SUM(FINAL_AMOUNT) AS TOTAL_REVENUE
  3  FROM ordertable
  4  GROUP BY TO_CHAR(ORDER_DATE, 'MON-YYYY')
  5  ORDER BY TO_DATE(TO_CHAR(ORDER_DATE, 'MON-YYYY'), 'MON-YYYY');

MONTH_YEAR        TOTAL_REVENUE
----------------- -------------
JAN-2023                 493.95
FEB-2023                  44.98
MAR-2023                 389.97
APR-2023                 132.98
MAY-2023                1846.38
JUN-2023                 912.91
JUL-2023                 539.94
AUG-2023                1389.85
SEP-2023                 330.48
OCT-2023                 215.98
NOV-2023                  89.97

11 rows selected.

SQL>
```

Figure 30: viewing the total revenue for each month in the order table

EXPLANATION: This query returns the total revenue for each month in the order table. The query first converts the order date to a string in the format 'MON-YYYY' using the TO_CHAR() function. It then groups the rows by the MONTH_YEAR column and sums the FINAL_AMOUNT column to get the total revenue for each month. Finally, the query orders the results by the MONTH_YEAR column in ascending order.

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

Query- SELECT * FROM ordertable WHERE final_amount >= (SELECT AVG(final_amount) FROM ordertable);

```
SQL> SELECT *
  2  FROM ordertable
  3  WHERE FINAL_AMOUNT >= (SELECT AVG(FINAL_AMOUNT) FROM ordertable);

  ORDER_ID ORDER_DAT DISCOUNT_AMOUNT TOTAL_AMOUNT FINAL_AMOUNT PAYMENT_METHOD       CUSTOMER_ID
---------- --------- --------------- ------------ ------------ -------------------- -----------
      2000 15-JAN-23           25.99       519.94       493.95 Esewa                          1
      2002 10-MAR-23               0       389.97       389.97 IME Pay                        3
      2006 15-MAY-23           44.99       899.99          855 IME Pay                        7
      2007 25-MAY-23           31.99       319.97       287.98 Cash                           2
      2011 08-JUN-23               0       339.97       339.97 Cash                          12
      2012 10-JUN-23           19.49       389.97       370.48 Esewa                         13
      2015 25-JUL-23               0       389.97       389.97 Cash                           9
      2016 05-AUG-23               0       389.97       389.97 Esewa                         18
      2017 15-AUG-23               0       479.97       479.97 Khalti                        18
      2021 14-SEP-23           19.49       349.97       330.48 Khalti                        13

10 rows selected.
```

Figure 31: listing those orders that are equal or higher than the average order total value

EXPLANATION:

The query is used to find all the orders where the final amount is greater than or equal to the average final amount of all orders.

c) <u>List the details of vendors who have supplied more than 3 products to the company.</u>

Query-

```
SELECT V.VENDOR_ID, V.VENDOR_NAME, V.VENDOR_EMAIL,
V.VENDOR_CONTACT, V.VENDOR_ADDRESS, COUNT(P.PRODUCT_ID) AS
PRODUCT_COUNT
FROM VENDOR V
INNER JOIN PRODUCT P ON V.VENDOR_ID = P.VENDOR_ID
GROUP BY V.VENDOR_ID, V.VENDOR_NAME, V.VENDOR_EMAIL,
V.VENDOR_CONTACT, V.VENDOR_ADDRESS
HAVING COUNT(P.PRODUCT_ID) >= 3;
```

```
SQL> SELECT V.VENDOR_ID, V.VENDOR_NAME, V.VENDOR_EMAIL, V.VENDOR_CONTACT, V.VENDOR_ADDRESS, COUNT(P.PRODUCT_ID) AS PRODUCT_COUNT
  2  FROM VENDOR V
  3  INNER JOIN PRODUCT P ON V.VENDOR_ID = P.VENDOR_ID
  4  GROUP BY V.VENDOR_ID, V.VENDOR_NAME, V.VENDOR_EMAIL, V.VENDOR_CONTACT, V.VENDOR_ADDRESS
  5  HAVING COUNT(P.PRODUCT_ID) >= 3;

 VENDOR_ID VENDOR_NAME                    VENDOR_EMAIL                                       VENDOR_CONTACT       VENDOR_ADDRESS       PRODUCT_COUNT
---------- ------------------------------ -------------------------------------------------- -------------------- -------------------- -------------
       101 LG Kitchen Suppliers           LGks@gmail.com                                     01-22334490          Koteshwor                        6
       102 XYZ Accessories Distributers   xyzAccessories@gmail.com                            01-22334491          Boudha                           6
       100 ABC TV Nepal                   ABC@gmail.com                                      01-22334489          Kalanki                          6
       103 Network House Nepal            NetworkHouseNepal@gmail.com                        01-22334492          Thamel                           6
       104 GadgetGalaxy Nepal             gadgetgalaxy@gmail.com                             01-22334495          Gongabu                          6

SQL>
```

Figure 32:Listing the details of vendors who have supplied more than 3 products to the company

EXPLANATION:

The query is used to fetch all the vendors who have more than or equal to 3 products. The query joins the VENDOR and PRODUCT tables on the VENDOR_ID column. The COUNT() function is used to count the number of products for each vendor. The HAVING clause is used to filter the results and only return the vendors who have more than or equal to 3 products.

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

Query-

```
SELECT P.Product_id, P.Product_name, P.price, P.DESCRIPTION, P.Stock_quantities,
PO. "Total Order Quantities"
FROM product P
INNER JOIN (
   SELECT product_id, SUM(order_quantities) AS "Total Order Quantities"
   FROM productorder
   GROUP BY product_id
   ORDER BY "Total Order Quantities" DESC
) PO ON P.product_id = PO.product_id
WHERE ROWNUM <= 3
ORDER BY PO. "Total Order Quantities" DESC;
```

```
SQL> SELECT P.Product_id, P.Product_name, P.price, P.DESCRIPTION, P.Stock_quantities, PO. "Total Order Quantities"
  2  FROM product P
  3  INNER JOIN (
  4      SELECT product_id, SUM(order_quantities) AS "Total Order Quantities"
  5      FROM productorder
  6      GROUP BY product_id
  7      ORDER BY "Total Order Quantities" DESC
  8  ) PO ON P.product_id = PO.product_id
  9  WHERE ROWNUM <= 3
 10  ORDER BY PO. "Total Order Quantities" DESC;

Product ID Product Name                     Price Description                                          Stock Quantities Total Order Quantities
---------- ---------------------------- ------- -------------------------------------------------- ---------------- ----------------------
         3 HD Soundbar                    149.99 Enhance your audio experience with this soundbar                 20                      5
         1 Smart TV Stand                  99.99 Stylish stand for your smart TV                                  50                      2
         2 Kitchen Blender                 49.99 Powerful blender for your kitchen needs                          30                      1

SQL>
```

Figure 33: Showing the top 3 product details that have been ordered the most

EXPLANATION:

This query retrieves the top three products with the highest order quantities detailed information including their names, prices, descriptions and stock quantities.

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

Query-

```
SELECT CUSTOMER_ID, CUSTOMER_NAME, "TOTAL_SPENDING ON AUGUST",
"NUMBER_OF_ORDERS"
FROM (
   SELECT C.CUSTOMER_ID, C.CUSTOMER_NAME, SUM(OT.FINAL_AMOUNT) AS
"TOTAL_SPENDING ON AUGUST", COUNT(OT.ORDER_ID) AS
"NUMBER_OF_ORDERS"
   FROM customer C
   JOIN ordertable OT ON C.CUSTOMER_ID = OT.CUSTOMER_ID
   WHERE TO_CHAR(OT.ORDER_DATE, 'MM-YYYY') = '08-2023'
   GROUP BY C.CUSTOMER_ID, C.CUSTOMER_NAME
   ORDER BY "TOTAL_SPENDING ON AUGUST" DESC
) WHERE ROWNUM = 1;
```

```
SQL> SELECT CUSTOMER_ID, CUSTOMER_NAME, "TOTAL_SPENDING ON AUGUST", "NUMBER_OF_ORDERS"
  2  FROM (
  3      SELECT C.CUSTOMER_ID, C.CUSTOMER_NAME, SUM(OT.FINAL_AMOUNT) AS "TOTAL_SPENDING ON AUGUST", COUNT(OT.ORDER_ID) AS "NUMBER_OF_ORDERS"
  4      FROM customer C
  5      JOIN ordertable OT ON C.CUSTOMER_ID = OT.CUSTOMER_ID
  6      WHERE TO_CHAR(OT.ORDER_DATE, 'MM-YYYY') = '08-2023'
  7      GROUP BY C.CUSTOMER_ID, C.CUSTOMER_NAME
  8      ORDER BY "TOTAL_SPENDING ON AUGUST" DESC
  9  ) WHERE ROWNUM = 1;

CUSTOMER_ID CUSTOMER_NAME                                      TOTAL_SPENDING ON AUGUST NUMBER_OF_ORDERS
----------- ------------------------------------------------- ------------------------ ----------------
         18 Asmita Karki                                                        1149.88                4

SQL>
```

Figure 34: listing the customer who has ordered the most in August with his/her total spending on that month.

EXPLANATION:

This query provides information about the customer with the highest number of orders in August, including customer ID and name, along with their total spending for that month. In our case, **Asmita Karki** is the user who has the highest spending of 1149.88 with a total of **4 orders in August**.

8. Critical Evaluation

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

The module on database design and implementation plays a pivotal role in comprehending the principles and practices essential for creating efficient database systems. The coursework facilitated hands-on experience in crafting a database for the hypothetical business scenario, "Gadget Emporium," encompassing various aspects such as conceptual modeling, normalization, and SQL queries. The module's significance extends to real-world applications, recognizing the crucial role of well-designed databases in managing and leveraging valuable data for businesses in today's digital age. The practical application of theoretical concepts in the coursework provided valuable insights into how businesses structure their data to support operations and decision-making.

In addition, the module establishes clear connections with other subjects, especially those in information technology and business management. Database design serves as a foundational element in information systems, and a well-designed database is imperative for the seamless functioning of diverse business processes. Professionals in roles such as data analysis, software development, and system administration benefit from understanding how databases work. This interrelation emphasizes the module's broader impact on interdisciplinary knowledge and its relevance in shaping the skill sets of individuals pursuing various career paths within the tech and business domains.

The Database Systems module offers a comprehensive overview of fundamental concepts, principles, and practices within database management systems. Equipping students with the knowledge and skills needed to design, implement, and administer efficient databases, the module covers diverse aspects such as data modeling, query processing, transaction management, and database security. Beyond the classroom, the

module's applicability extends across different industries and domains, including business intelligence, e-commerce, healthcare, and finance. The acquired insights and techniques from this module serve as a solid foundation for further academic pursuits and practical applications, seamlessly integrating with other subjects within the computer science curriculum, such as data structures and algorithms, operating systems, and software engineering.

8.2 Critical Assessment of coursework

The coursework effectively covered the entire process of designing and normalizing a database for an e-commerce platform, Gadget Emporium. It started with defining business rules and requirements, followed by the creation of a conceptual data model and the subsequent normalization steps. The coursework provided a comprehensive insight into the database design process, showcasing how to handle entities, attributes, relationships, and normalization techniques.

The use of SQL queries for data manipulation and retrieval was well-integrated into the coursework, demonstrating practical applications of the designed database. The implementation of queries for information retrieval and transaction operations showcased the real-world utility of the normalized database structure.

Overall, the coursework successfully achieved its objectives of introducing and applying database design and normalization principles in the context of an e-commerce platform. It provided valuable insights into creating a robust and efficient database system, which is a critical aspect of modern software development and information management.

9. Drop Query , Database Dump file and Spool file creation

9.1 Dropping the Tables:

```
SQL> drop table productorder;

Table dropped.

SQL> drop table ordertable;

Table dropped.

SQL> drop table product;

Table dropped.

SQL> drop table vendor;

Table dropped.

SQL> drop table customer;

Table dropped.

SQL> drop table category;

Table dropped.

SQL>
```

Figure 35: Screenshot of dropping tables

EXPLANATION:

When dropping a table that has a foreign key relationship with another table, it's essential to delete the child table first. This is because the foreign key constraint in the child table establishes a connection to the primary key in the parent table. Removing the child table, along with its foreign key, ensures that the relationship is severed, allowing for the deletion of the parent table without violating referential integrity. In essence, the removal of the child table acts as a prerequisite step to dropping the parent table with foreign key constraints.

9.2 Dump file creation:

```
Select Administrator: Command Prompt
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\miraj\OneDrive\Desktop\dumb files

C:\Users\miraj\OneDrive\Desktop\dumb files>exp COURSEWORK/miraj file=courseworkmiraj.dmp

Export: Release 11.2.0.2.0 - Production on Sat Jan 13 16:33:47 2024

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)

About to export specified users ...
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user COURSEWORK
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user COURSEWORK
About to export COURSEWORK's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export COURSEWORK's tables via Conventional Path ...
. . exporting table                       CATEGORY          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                       CUSTOMER         20 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                     ORDERTABLE         24 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        PRODUCT         32 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                   PRODUCTORDER         37 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                         VENDOR          7 rows exported
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
```

Figure 36: Screenshot of Creating dump file

9.3 SPOOL file creation:

```
SQL> spool C:\Users\miraj\OneDrive\Desktop\spool\query1.sql
SQL> select * from customer where CUSTOMER_CATEGORY= 'Staff' ;

CUSTOMER_ID CUSTOMER_NAME                                      DISCOUNT CUSTOMER_ADDRESS     CUSTOMER_PHONE       CUSTOMER_CATEGORY
----------- -------------------------------------------------- ---------- -------------------- -------------------- --------------------
          2 Ramesh Shahi                                               10 Koteshwor            9841345678           Staff
          6 Kiran Shrestha                                              0 Patan                9841745676           Staff
         10 Anju Gurung                                                 5 New Baneshwor        9842145680           Staff
         15 Alok Shrestha                                               0 Baneshwor            9842645685           Staff
         18 Asmita Karki                                                0 Sanepa               9842945688           Staff

SQL> spool off;
SQL>
```

Figure 37: Screenshot of doing spool

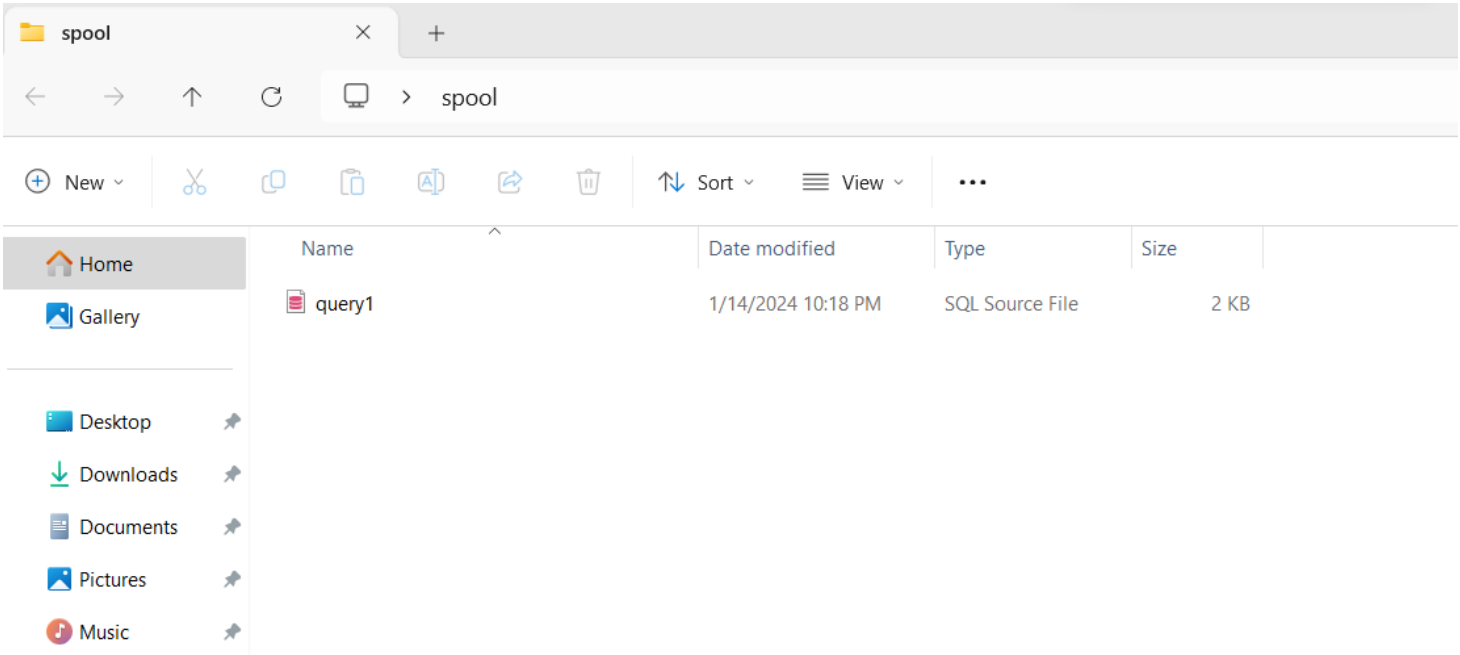

Figure 38: Sucessful Creation of spool file

10. References

Amghar, Y., Meziane, M. and Flory, A. (2000). Using Business Rules within a Design Process of Active Databases. Journal of Database Management, 11(3), pp.3–15. doi:https://doi.org/10.4018/jdm.2000070101.

Casanova, M.A. and Amaral de Sa, J.E. (1984). Mapping Uninterpreted Schemes into Entity-Relationship Diagrams: Two Applications to Conceptual Schema Design. IBM Journal of Research and Development, 28(1), pp.82–94. doi:https://doi.org/10.1147/rd.281.0082.

Demba, M. (2013). Algorithm for Relational Database Normalization Up to 3NF. International Journal of Database Management Systems, 5(3), pp.39–51. doi:https://doi.org/10.5121/ijdms.2013.5303.

Herman, M. (2019). A database design methodology for an integrated database environment. ACM SIGMIS Database, 15(1), pp.20–27. doi:https://doi.org/10.1145/1113500.1113503.

Zhang, Q. (2011). SQL Optimization Based on Oracle Database. Energy Procedia, 11, pp.486–492. doi:https://doi.org/10.1016/j.egypro.2011.10.271.

Worland, P.B. (2004). An efficient algorithm for 3NF determination. Information Sciences, 167(1-4), pp.177–192. doi:https://doi.org/10.1016/j.ins.2003.06.004.

Conceptual modelling of database applications using an extended ER model. (1992). Data & Knowledge Engineering, [online] 9(2), pp.157–204. doi:https://doi.org/10.1016/0169-023X(92)90008-Y.

Visual Paradigm (2019). What is Entity Relationship Diagram (ERD)? [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/.

11. APPENDIX

COMMIT

```
SQL> commit;

Commit complete.

SQL>
```

Figure 39: Screenshot of doing Commit after inseting the values into tables