CC5051NI Databases

50% Individual Coursework

Autumn 2023

Student Name: Rashi Maharjan

London Met ID: 22067683

Assignment Submission Date: 15 January 2024

Word Count: 3908

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

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1 About Gadget Emporium

Gadget Emporium is an e-commerce website that offers a curated selection of the most recent gadgets, smart devices, and inventive accessories of practically any sort, as well as a variety of other things. The items are available in a variety of situations, from brand new or factory sealed to used and everything in between (Gadget Emporium, 2017).

Mr. John who is an entrepreneur and electronics enthusiast plans to launch an online marketplace "Gadget Emporium" that specializes in providing and selling electronic devices and accessories to both private consumers and business organizations with a large selection of electronic devices. The proposed database system would be able to keep track of all customers, products, orders, and vendors.

1.2 Aims and Objectives

The primary goal of this project is to develop a database for Gadget Emporium to guarantee key business data is structured and easily available to relevant persons when required. The database is designed to provide important insights for the business by allowing queries to be conducted on the available data.

The objectives of this project are listed below:

a. To be able to create entities and attributes as well as their relationship types.
b. To identify and include keys and constraints.
c. To do Normalization of the Relationships (3NF) with explanation and reasonings.
d. To have detailed knowledge of making ERD before and after normalization.
e. To be able to develop database to keep record and store business records.

1.3 Current Business Activities and Operations

Gadget Emporium is an online marketplace which engage in the activity of providing a large selection of electronic items for both individual private consumers and large business organizations. It mainly specializes in selling varieties of gadgets, devices, and accessories. This ecommerce business needs a strong database system to support and maintain business records to be able keep track of all customers, products, and orders. But it also needs to follow certain business rules to run without any hassle. Some rules are mentioned below:

1.3.1 Business Rules

The following are some guidelines and policies that the business has to go by in order for operations to proceed smoothly and quickly:

a. Each product must be of only one category and each category can have one or many products.
b. Customers can purchase one or more products.
c. An order can have multiple products and any one type of product might be included in multiple orders placed by various customers.
d. Each product should be associated with a single vendor and each vendor can supply one or more products.
e. The inventory details and availability of products should be tracked.
f. Each order detail must have one payment processing via multiple gateways.
g. An invoice must be generated after order confirmation stating order, customer, and payment details with discount.

1.3.2 Assumptions

The assumptions are listed below:

1. A different discount rate on product purchases will be provided according to customer category.
2. Customers should be classified as Regular (R), Staff (S), or VIP (V) and eligible for different discount rate on goods purchases.
3. The total price of the product will vary according to the quantity of the product purchased.
4. The Net Amount will be evaluated after deducting Discount Amount through various discount rate.
5. Customer's address should be stored for delivery process.
6. Various payment gateways can be facilitated such as cash on delivery, credit/debit card or e-wallet.
7. An invoice is issued once the customer checks out their order after confirmation.

1.3.3 Identification of Entities and Attributes

The following are the entities with their attributes:

Entity	Attributes
Product	Product_ID, Product_Name, Product_Description, Product_Category, Product_Quantity, Unit_Cost, Line_Total, Product_Stock, Availability, Vendor_ID, Vendor_Name, Vendor_Address, Vendor_Contact
Customer	Customer_ID, Customer_Name, Customer_Contact, Customer_Address, Category, Discount_Rate
Order	Order_ID, Order_Date, Invoice_ID, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option

Table 1 List of Entities and Attributes

2. Initial ERD

2.1 Entities and Attributes

An entity is a distinct single object in the real world that can be uniquely identified, such as a person, a product, or an organization. An attribute is a characteristic of an entity that describes properties of the entity, such as a person's date of birth or the product's price. An entity can have one or more attributes in the context of database architecture (IBM, 2022).

The following entities and their characteristics can be identified as a starting point based on the business rules:

2.1.1 Product

Attributes	Data Type	Constraints	Description
Product_ID	Integer	Primary key	This attribute stores product's unique ID
Product_Name	Varchar(20)	Not Null	This attribute stores product's name
Product_Description	Varchar(30)	Null	This attribute stores product's description
Product_Category	Varchar(15)	Not Null	This attribute stores category under which the product falls on.
Product_Quantity	Integer	Not Null	This attribute stores quantity of the ordered product.
Unit_Cost	Integer	Not Null	This attribute stores unit cost price of the product.
Line_Total	Integer	Not Null	The total amount of product according to the quantity is stored in this field.

Product_Stock	Integer	Null	This attribute stores total number of stocks of the product.
Availability	Varchar(15)	Null	This attribute stores availability status of the product.
Vendor_ID	Integer	Not Null	This attribute stores vendor's unique ID.
Vendor_Name	Varchar(20)	Not Null	This attribute stores vendor's name.
Vendor_Address	Varchar(10)	Not Null	This attribute stores vendor's address.
Vendor_Contact	Integer	Not Null, Unique key	This attribute stores vendor's phone number.

Table 2 Product table before normalization

2.1.2 Customer

Attributes	Data Type	Constraints	Description
Customer_ID	Integer	Primary key	This attribute stores customer's unique ID.
Customer_Name	Varchar(20)	Not Null	This attribute stores customer's full name.
Customer_Address	Varchar(10)	Not Null	This attribute stores customer's delivery address.
Customer_Contact	Integer	Not Null, Unique	This attribute stores customer's contact number.
Discount_Rate	Integer	Null	This attribute stores the discount that customer

			received based on their category.
Category	Varchar(10)	Not Null	This attribute stores category of the customer that they belong to.

Table 3 Customer table before normalization

2.1.3 Order

Attributes	Data Type	Constraints	Description
Order_ID	Integer	Primary key	This attribute stores order's unique ID.
Order_Date	Date	Null	This attribute stores date when the order was placed.
Total	Integer	Null	This attribute stores total amount of the order.
Discount_Amount	Integer	Null	This attribute stores discount amount of the order.
Net_Amount	Integer	Null	This attribute stores total amount after deducting discount amount.
Invoice_ID	Integer	Not Null	This attribute stores invoice's unique ID.
Invoice_Date	Date	Null	This attribute stores date when the invoice was issued.
Payment_Option	Varchar(15)	Not Null	This attribute stores method in which payment was carried out.

Table 4 Order table before normalization

2.2 Entity Relationship Diagram

An entity-relationship diagram (ERD) is a form of flowchart that shows the relationships between "entities" in a system, which can be people, concepts, or objects. It is frequently used for relational database design and debugging in the domains of software, business information systems, study, and research. The primary entities inside the system scope and their relationships with one another are represented visually by a variety of symbols and connectors found in an ERD (Lucidchart , 2023).

The initial ERD is as follows:

In the following figure, there is one mandatory to many optional relations between Customer and Order as one customer may or may not have one or many orders. Order and Product have one mandatory to many mandatory relation as an order should at least consist one or many products.

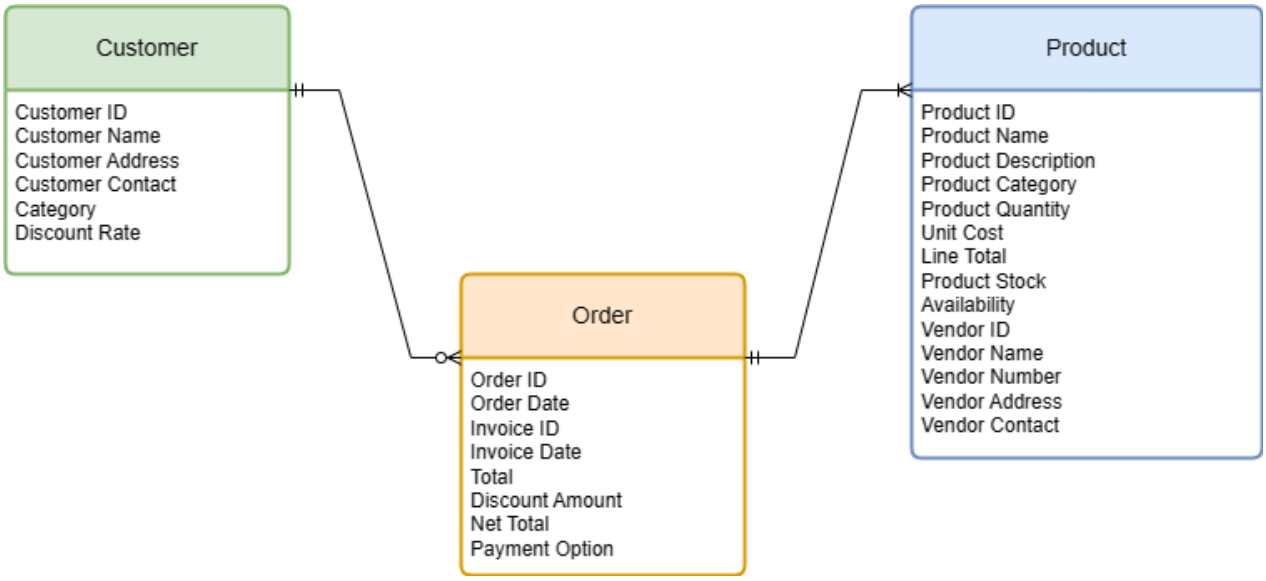

Figure 1 Initial ERD

3. Normalization

Normalization is the process of breaking down a complex relation into simple form. It reduces data redundancies and helps to eliminate the data anomalies that result from redundancies (Database Star, 2022). It simplifies the structure of the table and improves the performance of the data system.

3.1 UNF (Unnormalized Form)

Unnormalized Form (UNF) refers to a table that has not been normalized. It is a fundamental concept in database architecture and is a preparation stage that allows to produce a structured frame indicative of organizational data, such as a form or document (Geeksforgeeks, 2020).

All the attributes from the initial ERD are listed in UNF, and repeated groups of are included in a single relation in between curly brackets, i.e., ‘{}'. This entity is also given a suitably unique identifier, or primary key.

Applying UNF:

CUSTOMER (**Customer_ID**, Customer_Name, Customer_Contact, Customer_Address, Discount_Rate, Category, **{** Order_ID, Order_Date, Invoice_ID, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option, **{** Product_ID, Product_Name, Product_Description, Product_Category, Product_Quantity, Unit_Cost, Line_Total, Product_Stock, Availability, Vendor_ID, Vendor_Name, Vendor_Address, Vendor_Contact **}}**)

3.2 1NF (First Normal Form)

First Normal Form (1NF) is a relational database feature that assures that data is formatted in a way that maintains data integrity and eliminates redundancy (Database Star, 2022). A rule for 1NF is listed below:

- Relation without Repeating Group is already in 1NF.
- Remove all Repeating Group forming new relations (Shrestha, 2023).

Applying 1NF:

CUSTOMER-1 (**Customer_ID**, Customer_Name, Customer_Contact, Customer_Address, Discount_Rate, Category)

ORDER_DETAILS-1 (**Customer_ID*, Order_ID**, Order_Date, Invoice_ID, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option)

PRODUCT_ORDER-1 (**Customer_ID*, Order_ID*, Product_ID**, Product_Name, Product_Description, Product_Category, Product_Quantity, Unit_Cost, Line_Total, Product_Stock, Availability, Vendor_ID, Vendor_Name, Vendor_Address, Vendor_Contact)

3.3 2NF (Second Normal Form)

Second Normal Form (2NF) is a relational database feature that maintains data integrity and eliminates redundancy (Database Star, 2022). The rule for 2NF is listed below:

- Find and Separate Partial Functional Dependency.
- Need to check for Partial Functional Dependency in all Relation with Composite Key.
- Relation without Composite Key is already in 2NF.

Functional dependency is the relationship between attributes of the table dependent on each other i.e., using all attributes of Composite Determinant to identify its object uniquely. Partial Functional Dependency uses use only subset of Attributes of a Composite Determinant to identify object uniquely (Shrestha, 2023).

For CUSTOMER-1 Entity:

Customer_ID -> Customer_Name, Customer_Number, Customer_Address, Discount_Rate, Category

For ORDER_DETAILS-1 Entity- under assumption: an invoice is issued after the customer checks out their order.

Customer_ID, Order_ID -> x

Order_ID -> Order_Date, Invoice_ID, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option

Customer_ID -> x

For PRODUCT_ORDER-1 Entity- under assumption: a product is associated with a vendor.

<u>Customer_ID,</u> <u>Order_ID,</u> <u>Product_ID</u> -> Product_Quantity, Line_Total

<u>Order_ID,</u> <u>Product_ID</u> -> x

Customer_ID, Order_ID -> x

Customer_ID,Product_ID ->

Product_ID -> Product_Name, Product_Description, Product_Category, Unit_Cost, Product_Stock, Availability, Vendor_ID, Vendor_Name, Vendor_Address, Vendor_Contact
Order_ID -> x
Customer_ID -> x

Initial 2NF:

ORDER_DETAILS-2 (**Customer_ID, Order_ID**)
ORDERS-2 (**Order_ID**, Order_Date, Invoice_ID, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option)
PRODUCT_ORDER-2 (**Customer_ID, Order_ID, Product_ID**, Product_Quantity, Line_Total)
PRODUCT-2 (**Product_ID**, Product_Name, Product_Description, Product_Category, Unit_Cost, Product_Stock, Availability, Vendor_ID, Vendor_Name, Vendor_Address, Vendor_Contact)

Final 2NF:

CUSTOMER-2 (**Customer_ID**, Customer_Name, Customer_Number, Customer_Address, Discount_Rate, Category)
ORDER_DETAILS-2 (**Customer_ID*, Order_ID***)
ORDERS-2 (**Order_ID**, Order_Date, Invoice_ID, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option)
PRODUCT_ORDER-2 (**Customer_ID*, Order_ID*, Product_ID***, Product_Quantity, Line_Total)
PRODUCT-2 (**Product_ID**, Product_Name, Product_Description, Product_Category, Unit_Cost, Product_Stock, Availability, Vendor_ID, Vendor_Name, Vendor_Address, Vendor_Contact)

3.4 3NF (Third Normal Form)

Third Normal Form (3NF) is a relational database schema design method that seeks to decrease data duplication, avoid data anomalies, maintain referential integrity, and simplify data maintenance (Database Star, 2022). The rule for 3NF is listed below:

- Find and Separate Transitive Functional Dependency.
- Need to check for Transitive Functional Dependency in all Relation with more than one Non-Key Attribute.
- Relation with only one Non-Key Attribute is already in 3NF.

Non-Key Attribute is an attribute that does not participate in the primary key. Transitive Dependency is an indirect relationship which causes functional dependency. It exists when there is an intermediate dependency (Shrestha, 2023).

For ORDERS-2 Entity:

Order_ID -> Order_Date, Invoice_ID

Invoice_ID -> Invoice_Date, Total, Discount_Amount, Net_Amount

Order_ID -> Invoice_ID -> Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option

For PRODUCT-2 Entity:

Product_ID, -> Product_Name, Product_Description, Product_Category, Unit_Cost, Product_Stock, Availability, Vendor_ID

Vendor_ID -> Vendor_Name, Vendor_Address, Vendor_Contact

Product_ID -> Vendor_ID -> Vendor_Name, Vendor_Address, Vendor_Contact

Initial 3NF:

ORDERS-3 (**Order_ID**, Order_Date, **Invoice_ID**)
INVOICE-3 (**Invoice_ID**, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option)
PRODUCT-3 (**Product_ID**, Product_Name, Product_Description, Product_Category, Unit_Cost, Product_Stock, Availability, **Vendor_ID**)
VENDOR-3 (**Vendor_ID**, Vendor_Name, Vendor_Address, Vendor_Contact)

Final 3NF:

CUSTOMER-3 (**Customer_ID**, Customer_Name, Customer_Number, Customer_Address, Discount_Rate, Category)
ORDER_DETAILS-3 (**Customer_ID*, Order_ID***)
ORDERS-3 (**Order_ID**, Order_Date, **Invoice_ID***)
INVOICE-3 (**Invoice_ID**, Invoice_Date, Total, Discount_Amount, Net_Amount, Payment_Option)
PRODUCT_ORDER-3 (**Customer_ID*, Order_ID*, Product_ID***, Product_Quantity, Line_Total)
PRODUCT-3 (**Product_ID**, Product_Name, Product_Description, Product_Category, Unit_Cost, Product_Stock, Availability, **Vendor_ID***)
VENDOR-3 (**Vendor_ID**, Vendor_Name, Vendor_Address, Vendor_Contact)

4. Final ERD

The final ERD has been generated after performing normalization which consist of seven entities, and they are: Customer, Order_Details, Order, Invoice, Product_Order, Product and Vendor. Initially, there were three entities i.e., Customer, Order and Product. The entities in initial ERD consist of data redundancy and dependencies like partial dependency and transitive dependency which have been reduced with the help of normalization.

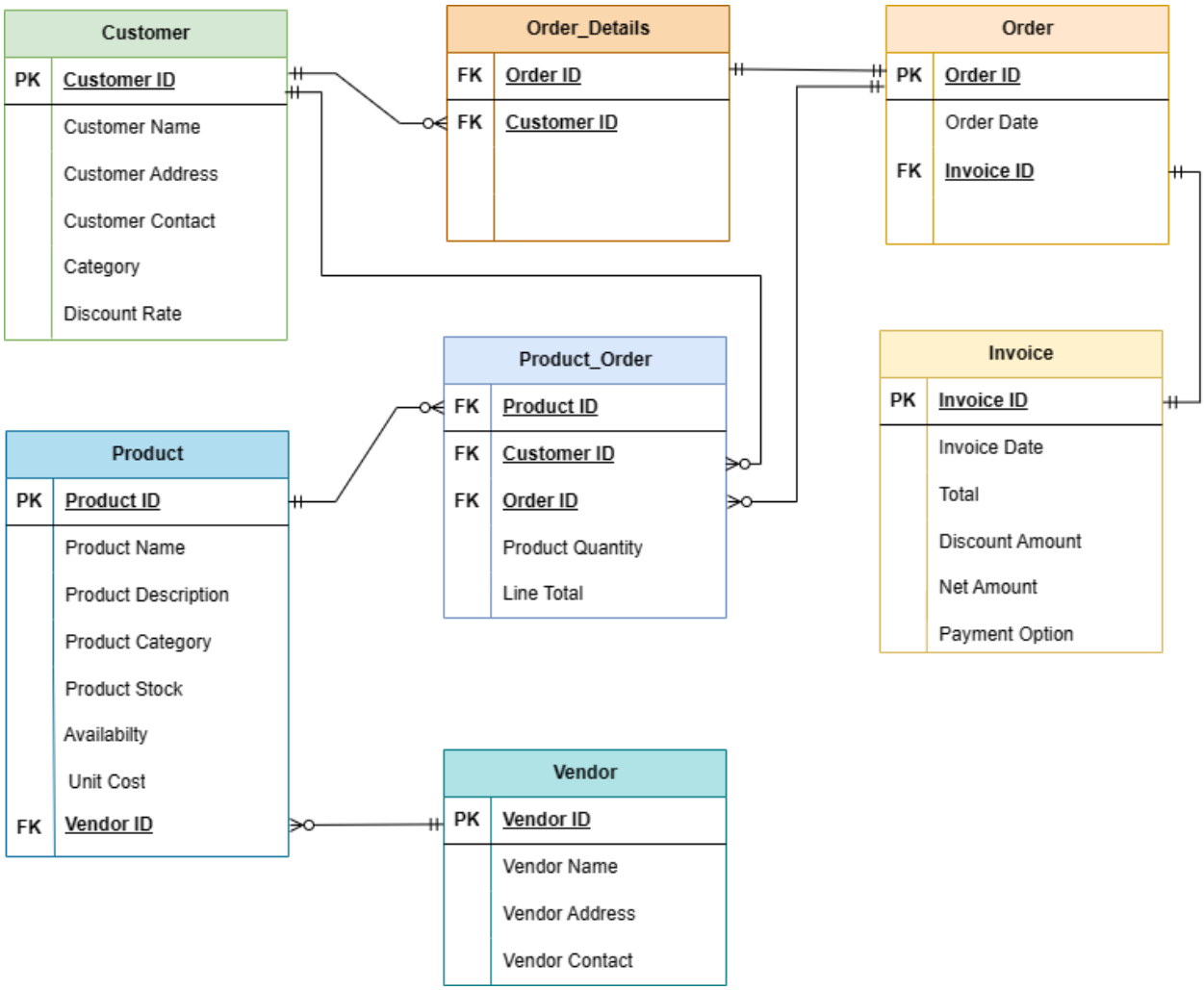

Figure 2 Final ERD

Relation between entities:

Entities-Relation	Relationship
Customer-Order_Details	Customer and Order_Details have relationship of one mandatory to many optional because a customer can have one or many order details stored of order made but an order detail must have a customer.
Customer-Product_Order	Customer and Product_Order have relationship of one mandatory to many optional because a customer can have one or many order made of products, but a product order must have a customer.
Order-Order_Details	Order and Order_Details have relationship of one-to-one mandatory because an order must have an order detail stored of order made.
Order-Product_Order	Order and Product_Order have relationship of one mandatory to many optional because an order can have one or many products orders made but a product order must have an order.
Product- Product_Order	Product and Product_Order have relationship of one mandatory to many optional because a product can be in multiple products orders, but a product order must have a product in it.
Vendor-Product	Vendor and Product have relationship of one mandatory to many optional because a vendor may supply one or many products supplied but a product must have a vendor.
Invoice-Order	Invoice and Order have relationship of one-to-one mandatory because a invoice must be generated in every order.

Table 5 Relation between entities.

5. Implementation

5.1 Creating a New User

The user Rashi_CW has been created to store the data and create the database for the Gadget Emporium. The privilege is granted to the user to execute the required SQL statements and perform database actions.

```
SQL*Plus: Release 11.2.0.2.0 Production on Wed Jan 10 18:09:20 2024

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

SQL> conn System/totoro
Connected.
SQL> CREATE USER Rashi_CW IDENTIFIED BY rashi;

User created.

SQL> GRANT CONNECT, RESOURCE TO Rashi_CW;

Grant succeeded.

SQL> conn Rashi_CW/rashi
Connected.
SQL> |
```

Figure 3 Creating a New User

5.2 Creating Entities with their attributes

5.2.1 Creating table Customer

Attribute	Data Type	Constraint	Description
Customer_ID	Integer	Primary key	This attribute stores customer's unique ID.
Customer_Name	Varchar(20)	Not Null	This attribute stores customer's full name.
Customer_Address	Varchar(10)	Not Null	This attribute stores customer's delivery address.
Customer_Contact	Integer	Not Null, Unique	This attribute stores customer's contact number.
Discount_Rate	Integer	Null	This attribute stores discount that customer received based on their category.

Table 6 Customer table after normalization.

```
SQL> CREATE TABLE Customer
  2  (Customer_ID int Primary key,
  3  Customer_Name varchar(20) not null,
  4  Customer_Address varchar(10) not null,
  5  Customer_Contact int not null Unique,
  6  Discount_Rate int,
  7  Category varchar(10) not null);

Table created.

SQL> DESCRIBE Customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CUSTOMER_ID                               NOT NULL NUMBER(38)
 CUSTOMER_NAME                             NOT NULL VARCHAR2(20)
 CUSTOMER_ADDRESS                          NOT NULL VARCHAR2(10)
 CUSTOMER_CONTACT                          NOT NULL NUMBER(38)
 DISCOUNT_RATE                                      NUMBER(38)
 CATEGORY                                  NOT NULL VARCHAR2(10)
```

Figure 4 Creating Customer table.

5.2.2 Creating table Invoice

Attribute	Data Type	Constraint	Description
Invoice_ID	Integer	Not Null	This attribute stores invoice's unique ID.
Invoice_Date	Date	Null	This attribute stores date when the invoice was issued.
Total	Integer	Null	This attribute stores total amount of the order.
Discount_Amount	Integer	Null	This attribute stores discount amount of the order.
Net_Amount	Integer	Null	This attribute stores total amount after deducting discount amount.
Payment_Option	Varchar(15)	Not Null	This attribute stores method in which payment was carried out.

Table 7 Invoice table after normalization.

```
SQL> CREATE TABLE Invoice
  2  (Invoice_ID int Primary key,
  3  Invoice_Date date,
  4  Total int,
  5  Discount_Amount int,
  6  Net_Amount int,
  7  Payment_Option varchar(15));

Table created.

SQL> DESCRIBE Invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 INVOICE_ID                                NOT NULL NUMBER(38)
 INVOICE_DATE                                       DATE
 TOTAL                                              NUMBER(38)
 DISCOUNT_AMOUNT                                    NUMBER(38)
 NET_AMOUNT                                         NUMBER(38)
 PAYMENT_OPTION                                     VARCHAR2(15)
```

Figure 5 Creating Invoice table.

5.2.3 Creating table Vendor

Attribute	Data Type	Constraint	Description
Vendor_ID	Integer	Not Null	This attribute stores vendor's unique ID.
Vendor_Name	Varchar(20)	Not Null	This attribute stores vendor's name.
Vendor_Address	Varchar(10)	Not Null	This attribute stores vendor's address.
Vendor_Contact	Integer	Not Null, Unique key	This attribute stores vendor's phone number.

Table 8 Vendor table after normalization.

```
SQL> CREATE TABLE Vendor
  2  (Vendor_ID int Primary Key,
  3  Vendor_Name varchar(20) not null,
  4  Vendor_Address varchar(10) not null,
  5  Vendor_Contact int not null Unique);

Table created.

SQL> DESCRIBE Vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 VENDOR_ID                                 NOT NULL NUMBER(38)
 VENDOR_NAME                               NOT NULL VARCHAR2(20)
 VENDOR_ADDRESS                            NOT NULL VARCHAR2(10)
 VENDOR_CONTACT                            NOT NULL NUMBER(38)
```

Figure 6 Creating Vendor table.

5.2.4 Creating table Product

Attribute	Data Type	Constraint	Description
Product_ID	Integer	Primary key	This attribute stores product's unique ID.
Product_Name	Varchar(20)	Not Null	This attribute stores product's name.
Product_Description	Varchar(30)	Null	This attribute stores product's description.
Product_Category	Varchar(15)	Not Null	This attribute stores category under which the product falls on.
Unit_Cost	Integer	Not Null	This attribute stores the unit cost price of the product.
Product_Stock	Integer	Null	This attribute stores the total number of stocks of the product.
Availability	Varchar(15)	Null	This attribute stores availability status of the product.
Vendor_ID	Integer	Foreign key, Not Null	This attribute stores vendor's unique ID.

Table 9 Product table after normalization.

```
SQL> CREATE TABLE Product
  2  (Product_ID int Primary key,
  3  Product_Name varchar(20) not null,
  4  Product_Description varchar(30),
  5  Product_Category varchar(15) not null,
  6  Unit_Cost int not null,
  7  Product_Stock int,
  8  Availability varchar(15),
  9  Vendor_ID int not null,
 10  Foreign key(Vendor_ID) REFERENCES Vendor(Vendor_ID));

Table created.

SQL> DESCRIBE Product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 PRODUCT_ID                                NOT NULL NUMBER(38)
 PRODUCT_NAME                              NOT NULL VARCHAR2(20)
 PRODUCT_DESCRIPTION                                VARCHAR2(30)
 PRODUCT_CATEGORY                          NOT NULL VARCHAR2(15)
 UNIT_COST                                 NOT NULL NUMBER(38)
 PRODUCT_STOCK                                      NUMBER(38)
 AVAILABILITY                                       VARCHAR2(15)
 VENDOR_ID                                 NOT NULL NUMBER(38)
```

Figure 7 Creating Product table.

5.2.5 Creating table Orders

Attribute	Data Type	Constraint	Description
Order_ID	Integer	Primary key	This attribute stores order's unique ID.
Order_Date	Date	Null	This attribute stores date when the order was placed.
Invoice_ID	Integer	Foreign key, Not Null	This attribute stores invoice's unique ID.

Table 10 Orders table after normalization.

```
SQL> CREATE TABLE Orders
  2  (Order_ID int Primary Key,
  3  Order_Date date,
  4  Invoice_ID int not null,
  5  Foreign Key(Invoice_ID) REFERENCES Invoice(Invoice_ID));

Table created.

SQL> DESCRIBE Orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 ORDER_ID                                  NOT NULL NUMBER(38)
 ORDER_DATE                                         DATE
 INVOICE_ID                                NOT NULL NUMBER(38)
```

Figure 8 Creating Orders table.

5.2.6 Creating table Order_Details

Attribute	Data Type	Constraint	Description
Order_ID	Integer	Foreign key, Not Null	This attribute stores order's unique ID.
Customer_ID	Integer	Foreign key, Not Null key	This attribute stores customer's unique ID.

Table 11 Order_Details table after normalization.

```
SQL> CREATE TABLE Order_Details
  2  (Order_ID int not null,
  3  Customer_ID int not null,
  4  Foreign Key(Order_ID) REFERENCES Orders(Order_ID),
  5  Foreign Key(Customer_ID) REFERENCES Customer(Customer_ID));

Table created.

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

Figure 9 Creating Order_Details table.

5.2.7 Creating table Product_Order

Attribute	Data Type	Constraint	Description
Customer_ID	Integer	Foreign key, Not Null key	This attribute stores customer's unique ID.
Order_ID	Integer	Foreign key, Not Null	This attribute stores order's unique ID.
Product_ID	Integer	Foreign key, Not Null key	This attribute stores product's unique ID.
Product_Quantity	Integer	Not Null	This attribute stores quantity of the product ordered.
Line_Total	Integer	Not Null	This attribute stores the total amount of product according to the quantity.

Table 12 Product_Order table after normalization.

```
SQL> CREATE TABLE Product_Order
  2  (Customer_ID int not null,
  3  Order_ID int not null,
  4  Product_ID int not null,
  5  Product_Quantity int,
  6  Line_Total int not null,
  7  Foreign Key (Customer_ID) REFERENCES Customer (Customer_ID),
  8  Foreign Key (Order_ID) REFERENCES Orders (Order_ID),
  9  Foreign Key (Product_ID) REFERENCES Product (Product_ID));

Table created.

SQL> DESCRIBE Product_Order;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 CUSTOMER_ID                               NOT NULL NUMBER(38)
 ORDER_ID                                  NOT NULL NUMBER(38)
 PRODUCT_ID                                NOT NULL NUMBER(38)
 PRODUCT_QUANTITY                                   NUMBER(38)
 LINE_TOTAL                                NOT NULL NUMBER(38)
```

Figure 10 Creating Product_Order table.

5.2.8 Display tables

```
SQL> SELECT * FROM tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
CUSTOMER                       TABLE
INVOICE                        TABLE
ORDERS                         TABLE
ORDER_DETAILS                  TABLE
PRODUCT                        TABLE
PRODUCT_ORDER                  TABLE
VENDOR                         TABLE

7 rows selected.
```

Figure 11 Displaying created tables.

5.3 Inserting Values and Checking the value insertion

5.3.1 Value for Customer table

```
SQL> INSERT ALL
  2  INTO Customer values (1011, 'Rima Suwal', 'Teku', 9813000111, 0, 'Regular')
  3  INTO Customer values (1122, 'Sezan Rai', 'Baluwatar', 9841123443, 0, 'Regular')
  4  INTO Customer values (1150, 'Avanish Karna', 'Baneshwor', 9841432660, 5, 'Staff')
  5  INTO Customer values (1244, 'Eva Lama', 'Ason', 9841777788, 10, 'VIP')
  6  INTO Customer values (1460, 'Daya Shrestha', 'Kuleshwor', 9851909090, 10, 'VIP')
  7  INTO Customer values (1593, 'Subin Malla', 'Bafal', 9851334455, 5, 'Staff')
  8  INTO Customer values (1678, 'Ram Maharjan', 'Sitapaila', 9813212121, 10, 'VIP')
  9  INTO Customer values (1746, 'Neha Karki', 'Kamaladi', 9814335678, 0, 'Regular')
 10  INTO Customer values (1867, 'Bina Thakur', 'Kohity', 9841098765, 0, 'Regular')
 11  INTO Customer values (1987, 'Aayush Nepali', 'Babarmahal', 9841888000, 5, 'Staff')
 12  SELECT * FROM dual;

10 rows created.

SQL> SELECT * FROM Customer;

CUSTOMER_ID CUSTOMER_NAME        CUSTOMER_A CUSTOMER_CONTACT DISCOUNT_RATE CATEGORY
----------- -------------------- ---------- ---------------- ------------- ----------
       1011 Rima Suwal           Teku            9813000111             0 Regular
       1122 Sezan Rai            Baluwatar       9841123443             0 Regular
       1150 Avanish Karna        Baneshwor       9841432660             5 Staff
       1244 Eva Lama             Ason            9841777788            10 VIP
       1460 Daya Shrestha        Kuleshwor       9851909090            10 VIP
       1593 Subin Malla          Bafal           9851334455             5 Staff
       1678 Ram Maharjan         Sitapaila       9813212121            10 VIP
       1746 Neha Karki           Kamaladi        9814335678             0 Regular
       1867 Bina Thakur          Kohity          9841098765             0 Regular
       1987 Aayush Nepali        Babarmahal      9841888000             5 Staff

10 rows selected.
```

Figure 12 Inserting value in Customer table.

5.3.2 Value for Invoice table

```
SQL> INSERT ALL
  2  INTO Invoice values (11023, '03-MAY-2023', 4200, 0, 4200, 'Delivery')
  3  INTO Invoice values (13890, '08-MAY-2023', 87000, 4350, 82650, 'E-Wallet')
  4  INTO Invoice values (17869, '20-MAY-2023', 9800, 980, 8820, 'Delivery')
  5  INTO Invoice values (20312, '09-JUN-2023', 3500, 0, 3500, 'Debit')
  6  INTO Invoice values (25619, '18-JUL-2023', 66000, 3300, 62700, 'Debit')
  7  INTO Invoice values (28934, '13-AUG-2023', 175000, 17500, 157500, 'Credit')
  8  INTO Invoice values (33006, '23-AUG-2023', 52500, 0, 52500, 'E-Wallet')
  9  INTO Invoice values (33033, '30-AUG-2023', 12500, 625, 11875, 'E-Wallet')
 10  SELECT * FROM dual;

8 rows created.

SQL> SELECT * FROM Invoice;

INVOICE_ID INVOICE_D      TOTAL DISCOUNT_AMOUNT NET_AMOUNT PAYMENT_OPTION
---------- -------- ---------- --------------- ---------- --------------
     11023 03-MAY-23       4200               0       4200 Delivery
     13890 08-MAY-23      87000            4350      82650 E-Wallet
     17869 20-MAY-23       9800             980       8820 Delivery
     20312 09-JUN-23       3500               0       3500 Debit
     25619 18-JUL-23      66000            3300      62700 Debit
     28934 13-AUG-23     175000           17500     157500 Credit
     33006 23-AUG-23      52500               0      52500 E-Wallet
     33033 30-AUG-23      12500             625      11875 E-Wallet

8 rows selected.
```

Figure 13 Inserting value in Invoice table.

5.3.3 Value for Vendor table

```
SQL> INSERT ALL
  2  INTO Vendor values (105, 'Shyam Mali', 'Balaju', 9841231542)
  3  INTO Vendor values (124, 'Hari Shakya', 'Tinkune', 9841987007)
  4  INTO Vendor values (139, 'Naresh Suwal', 'New Road', 9851676767)
  5  INTO Vendor values (155, 'Birendra Adhikari', 'Sanepa', 9813221221)
  6  INTO Vendor values (198, 'Prakash Ojha', 'Chhauni', 9841551055)
  7  INTO Vendor values (117, 'Sailesh Nepal', 'Baneshwor', 9813444440)
  8  INTO Vendor values (182, 'Laxman Pradhan', 'Thapagaun', 9813335590)
  9  SELECT * FROM dual;

7 rows created.

SQL> SELECT * FROM Vendor;

 VENDOR_ID VENDOR_NAME          VENDOR_ADD VENDOR_CONTACT
---------- -------------------- ---------- --------------
       105 Shyam Mali           Balaju         9841231542
       124 Hari Shakya          Tinkune        9841987007
       139 Naresh Suwal         New Road       9851676767
       155 Birendra Adhikari    Sanepa         9813221221
       198 Prakash Ojha         Chhauni        9841551055
       117 Sailesh Nepal        Baneshwor      9813444440
       182 Laxman Pradhan       Thapagaun      9813335590

7 rows selected.
```

Figure 14 Inserting value in Vendor table.

5.3.4 Value for Product table

```
SQL> INSERT ALL
  2  INTO Product values (304, 'Iphone XR', 'Black 64GB', 'Mobile', 66000, 30, 'Available', 139)
  3  INTO Product values (332, 'Samsung Galaxy A24', 'White 256GB', 'Mobile', 50000, 56, 'Available', 117)
  4  INTO Product values (367, 'Vaio FE14', 'Black 14.1 inch', 'Laptop', 75000, 52, 'Available', 117)
  5  INTO Product values (321, 'Panasonic', 'Black  32 inch', 'TV', 87000, 59, 'Available', 124)
  6  INTO Product values (310, 'Ultima Atom 520 Pro', 'Black', 'Earbuds', 2500, 45, 'Available', 155)
  7  INTO Product values (359, 'T800 Ultra', 'Orange 44mm', 'Smart Watch', 1750, 51, 'Available', 105)
  8  INTO Product values (365, 'Apple Watch SE', 'Silver 40mm', 'Smart Watch', 6500, 36, 'Out of Stock', 139)
  9  INTO Product values (378, 'Asus Vivobook', 'Navy 14 inch', 'Laptop', 112500, 17, 'Out of Stock', 182)
 10  INTO Product values (385, 'Iphone 15 Pro Max', 'Blue 1TB', 'Mobile', 218000, 57, 'Available', 139)
 11  INTO Product values (399, 'Airpods Pro', 'White', 'Earbuds', 2100, 55, 'Available', 139)
 12  SELECT * FROM dual;

10 rows created.

SQL> SELECT * FROM Product;

PRODUCT_ID PRODUCT_NAME         PRODUCT_DESCRIPTION           PRODUCT_CATEGOR  UNIT_COST PRODUCT_STOCK AVAILABILITY     VENDOR_ID
---------- -------------------- ----------------------------- --------------- ---------- ------------- --------------- ----------
       304 Iphone XR            Black 64GB                    Mobile               66000            30 Available              139
       332 Samsung Galaxy A24   White 256GB                   Mobile               50000            56 Available              117
       367 Vaio FE14            Black 14.1 inch               Laptop               75000            52 Available              117
       321 Panasonic            Black  32 inch                TV                   87000            59 Available              124
       310 Ultima Atom 520 Pro  Black                         Earbuds               2500            45 Available              155
       359 T800 Ultra           Orange 44mm                   Smart Watch           1750            51 Available              105
       365 Apple Watch SE       Silver 40mm                   Smart Watch           6500            36 Out of Stock           139
       378 Asus Vivobook        Navy 14 inch                  Laptop              112500            17 Out of Stock           182
       385 Iphone 15 Pro Max    Blue 1TB                      Mobile              218000            57 Available              139
       399 Airpods Pro          White                         Earbuds               2100            55 Available              139

10 rows selected.
```

Figure 15 Inserting value in Product table.

5.3.5 Value for Orders table

```
SQL> INSERT ALL
  2  INTO Orders values (2003, '01-MAY-2023', 11023)
  3  INTO Orders values (2060, '08-MAY-2023', 13890)
  4  INTO Orders values (2133, '20-MAY-2023', 17869)
  5  INTO Orders values (2290, '04-JUN-2023', 20312)
  6  INTO Orders values (2401, '06-JUL-2023', 25619)
  7  INTO Orders values (2679, '10-AUG-2023', 28934)
  8  INTO Orders values (2808, '20-AUG-2023', 33006)
  9  INTO Orders values (2999, '30-AUG-2023', 33033)
 10  SELECT * FROM dual;

8 rows created.

SQL> SELECT * FROM Orders;

  ORDER_ID ORDER_DAT INVOICE_ID
---------- --------- ----------
      2003 01-MAY-23      11023
      2060 08-MAY-23      13890
      2133 20-MAY-23      17869
      2290 04-JUN-23      20312
      2401 06-JUL-23      25619
      2679 10-AUG-23      28934
      2808 20-AUG-23      33006
      2999 30-AUG-23      33033

8 rows selected.
```

Figure 16 Inserting value in Orders table.

5.3.6 Value for Order_Details table

```
SQL> INSERT ALL
  2  INTO Order_Details values (2003, 1122)
  3  INTO Order_Details values (2060, 1150)
  4  INTO Order_Details values (2133, 1460)
  5  INTO Order_Details values (2290, 1122)
  6  INTO Order_Details values (2401, 1593)
  7  INTO Order_Details values (2679, 1678)
  8  INTO Order_Details values (2808, 1867)
  9  INTO Order_Details values (2999, 1987)
 10  SELECT * FROM dual;

8 rows created.

SQL> SELECT * FROM Order_Details;

  ORDER_ID CUSTOMER_ID
---------- -----------
      2003        1122
      2060        1150
      2133        1460
      2290        1122
      2401        1593
      2679        1678
      2808        1867
      2999        1987

8 rows selected.
```

Figure 17 Inserting value in Order_Details table.

5.3.7 Value for Product_Order table

```
SQL> INSERT ALL
  2  INTO Product_Order values (1122, 2003, 399, 2, 4200)
  3  INTO Product_Order values (1150, 2060, 321, 1, 87000)
  4  INTO Product_Order values (1460, 2133, 399, 3, 6300)
  5  INTO Product_Order values (1460, 2133, 359, 2, 3500)
  6  INTO Product_Order values (1122, 2290, 359, 2, 3500)
  7  INTO Product_Order values (1593, 2401, 304, 1, 66000)
  8  INTO Product_Order values (1678, 2679, 332, 2, 100000)
  9  INTO Product_Order values (1678, 2679, 367, 1, 75000)
 10  INTO Product_Order values (1867, 2808, 332, 1, 50000)
 11  INTO Product_Order values (1867, 2808, 310, 1, 2500)
 12  INTO Product_Order values (1987, 2999, 310, 5, 12500)
 13  SELECT * FROM dual;

11 rows created.

SQL> SELECT * FROM Product_Order;

CUSTOMER_ID   ORDER_ID PRODUCT_ID PRODUCT_QUANTITY LINE_TOTAL
----------- --------- ---------- ---------------- ----------
       1122      2003        399                2       4200
       1150      2060        321                1      87000
       1460      2133        399                3       6300
       1460      2133        359                2       3500
       1122      2290        359                2       3500
       1593      2401        304                1      66000
       1678      2679        332                2     100000
       1678      2679        367                1      75000
       1867      2808        332                1      50000
       1867      2808        310                1       2500
       1987      2999        310                5      12500

11 rows selected.
```

Figure 18 Inserting value in Product_Order table.

6. Database Querying

6.1 Information Query

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

Query: SELECT * FROM *Customer*

WHERE *Category* = 'Staff';

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

CUSTOMER_ID CUSTOMER_NAME        CUSTOMER_A CUSTOMER_CONTACT DISCOUNT_RATE CATEGORY
----------- -------------------- ---------- ---------------- ------------- --------
       1150 Avanish Karna        Baneshwor        9841432660             5 Staff
       1593 Subin Malla          Bafal            9851334455             5 Staff
       1987 Aayush Nepali        Babarmahal       9841888000             5 Staff
```

Figure 19 All the customers that are also staff.

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

Query: SELECT *Order_ID, Order_Date* FROM *Orders*

WHERE *Order_Date* BETWEEN '01-MAY-2023' AND '28-MAY-2023';

```
SQL> SELECT Order_ID, Order_Date FROM Orders
  2  WHERE Order_Date BETWEEN '01-MAY-2023' AND '28-MAY-2023';

  ORDER_ID ORDER_DAT
---------- ---------
      2003 01-MAY-23
      2060 08-MAY-23
      2133 20-MAY-23
```

Figure 20 All the orders made for any product between the dates 01-05-2023 till 28-05-2023.

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

Query: SELECT *Orders.Order_ID, Orders.Order_Date,*

Order_Details.Customer_ID, Customer.Customer_Name

FROM *Order_Details*

JOIN *Orders*

ON *Order_Details.Order_ID = Orders.Order_ID*

JOIN *Customer*

ON *Order_Details.Customer_ID = Customer.Customer_ID;*

```
SQL> SELECT Orders.Order_ID, Orders.Order_Date,
  2  Order_Details.Customer_ID, Customer.Customer_Name
  3  FROM Order_Details
  4  JOIN Orders
  5  ON Order_Details.Order_ID = Orders.Order_ID
  6  JOIN Customer
  7  ON Order_Details.Customer_ID = Customer.Customer_ID;

  ORDER_ID ORDER_DAT CUSTOMER_ID CUSTOMER_NAME
---------- --------- ----------- --------------------
      2290 04-JUN-23        1122 Sezan Rai
      2003 01-MAY-23        1122 Sezan Rai
      2060 08-MAY-23        1150 Avanish Karna
      2133 20-MAY-23        1460 Daya Shrestha
      2401 06-JUL-23        1593 Subin Malla
      2679 10-AUG-23        1678 Ram Maharjan
      2808 20-AUG-23        1867 Bina Thakur
      2999 30-AUG-23        1987 Aayush Nepali

8 rows selected.
```

Figure 21 All the customers with their order details.

Query: SELECT *Customer.Customer_ID, Customer.Customer_Name*

FROM *Customer*

LEFT JOIN *Order_Details ON Customer.Customer_ID = Order_Details.Customer_ID*

WHERE *Order_Details.Customer_ID* IS NULL;

```
SQL> SELECT Customer.Customer_ID, Customer.Customer_Name
  2  FROM Customer
  3  LEFT JOIN Order_Details ON Customer.Customer_ID = Order_Details.Customer_ID
  4  WHERE Order_Details.Customer_ID IS NULL;

CUSTOMER_ID CUSTOMER_NAME
----------- --------------------
       1746 Neha Karki
       1244 Eva Lama
       1011 Rima Suwal
```

Figure 22 Customers who have not ordered any products yet.

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

Query: SELECT * FROM *Product*

WHERE Product_Name LIKE '_a%'

AND *Product_Stock* > 50;

```
SQL> SELECT * FROM Product
  2  WHERE Product_Name LIKE '_a%'
  3  AND Product_Stock > 50;

PRODUCT_ID PRODUCT_NAME         PRODUCT_DESCRIPTION           PRODUCT_CATEGOR  UNIT_COST PRODUCT_STOCK AVAILABILITY     VENDOR_ID
---------- -------------------- ----------------------------- --------------- ---------- ------------- --------------- ---------
       332 Samsung Galaxy A24   White 256GB                   Mobile               50000            56 Available             117
       367 Vaio FE14            Black 14.1 inch               Laptop               75000            52 Available             117
       321 Panasonic            Black  32 inch                TV                   87000            59 Available             124
```

Figure 23 All product details that have the second letter ‘a’ in their product name and have a stock quantity more than 50.

6.1.5 Find out the customer who has ordered recently.

Query: SELECT *Order_Details.Order_ID, Order_Details.Customer_ID, Customer.Customer_Name, Orders.Order_Date*

FROM *Order_Details*

JOIN *Orders*

ON *Order_Details.Order_ID = Orders.Order_ID*

JOIN *Customer*

ON *Order_Details.Customer_ID = Customer.Customer_ID*

WHERE *Orders.Order_Date* = (SELECT MAX (*Order_Date*) FROM *Orders*);

```
SQL> SELECT Order_Details.Order_ID, Order_Details.Customer_ID,
  2  Customer.Customer_Name, Orders.Order_Date
  3  FROM Order_Details
  4  JOIN Orders
  5  ON Order_Details.Order_ID = Orders.Order_ID
  6  JOIN Customer
  7  ON Order_Details.Customer_ID = Customer.Customer_ID
  8  WHERE Orders.Order_Date = (SELECT MAX (Order_Date) FROM Orders);

  ORDER_ID CUSTOMER_ID CUSTOMER_NAME        ORDER_DAT
---------- ----------- -------------------- ---------
      2999        1987 Aayush Nepali        30-AUG-23
```

Figure 24 Customer who has ordered recently.

6.2 Transaction Query

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

Query: SELECT TO_CHAR(*Invoice_Date*, 'Month') AS month,

SUM(*Total*) AS Total_Revenue

FROM *Invoice*

GROUP BY TO_CHAR(*Invoice_Date*, 'Month')

ORDER BY month;

```
SQL> SELECT TO_CHAR(Invoice_Date, 'Month') AS month,
  2  SUM(Total) AS Total_Revenue
  3  FROM Invoice
  4  GROUP BY TO_CHAR(Invoice_Date, 'Month')
  5  ORDER BY month;

MONTH                                TOTAL_REVENUE
------------------------------------ -------------
August                                       240000
July                                          66000
June                                           3500
May                                          101000
```

Figure 25 Total revenue of the company for each month.

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

Query: SELECT * FROM *Invoice*

WHERE *Total* >= (SELECT AVG(*Total*) FROM *Invoice)*;

```
SQL> SELECT * FROM Invoice
  2  WHERE Total >= (SELECT AVG(Total) FROM Invoice);

INVOICE_ID INVOICE_D      TOTAL DISCOUNT_AMOUNT NET_AMOUNT PAYMENT_OPTION
---------- --------- ---------- --------------- ---------- ---------------
     13890 08-MAY-23      87000            4350      82650 E-Wallet
     25619 18-JUL-23      66000            3300      62700 Debit
     28934 13-AUG-23     175000           17500     157500 Credit
     33006 23-AUG-23      52500               0      52500 E-Wallet
```

Figure 26 Orders that are equal or higher than the average order total value.

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

Query: SELECT *Vendor.Vendor_ID, Vendor.Vendor_Name,*

Vendor.Vendor_Address, Vendor.Vendor_Contact,

COUNT*(Product.Product_ID*) AS Total_Supply

FROM *Vendor*

JOIN *Product*

ON *Vendor.Vendor_ID = Product.Vendor_ID*

GROUP BY *Vendor.Vendor_ID, Vendor.Vendor_Name,*
Vendor.Vendor_Address, Vendor.Vendor_Contact

HAVING COUNT(*Product.Product_ID)* > 3;

```
SQL> SELECT Vendor.Vendor_ID, Vendor.Vendor_Name,
  2  Vendor.Vendor_Address, Vendor.Vendor_Contact,
  3  COUNT(Product.Product_ID) AS Total_Supply
  4  FROM Vendor
  5  JOIN Product
  6  ON Vendor.Vendor_ID = Product.Vendor_ID
  7  GROUP BY Vendor.Vendor_ID, Vendor.Vendor_Name, Vendor.Vendor_Address, Vendor.Vendor_Contact
  8  HAVING COUNT(Product.Product_ID) > 3;

 VENDOR_ID VENDOR_NAME          VENDOR_ADD VENDOR_CONTACT TOTAL_SUPPLY
---------- -------------------- ---------- -------------- ------------
       139 Naresh Suwal         New Road       9851676767            4
```

Figure 27 Details of vendors who have supplied more than 3 products to the company.

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

Query: SELECT *Product_ID, Product_Name*, *Product_Category*, Repetition

FROM (

SELECT *Product.Product_ID, Product.Product_Name, Product.Product_Category,*

COUNT(*) AS Repetition

FROM *Product_Order*

JOIN *Product*

ON *Product_Order.Product_ID = Product.Product_ID*

GROUP BY *Product.Product_ID, Product.Product_Name, Product.Product_Category*

ORDER BY Repetition DESC)

WHERE ROWNUM <= 3;

```
SQL> SELECT Product_ID, Product_Name, Product_Category, Repetition
  2  FROM (
  3  SELECT Product.Product_ID, Product.Product_Name, Product.Product_Category,
  4  COUNT(*) AS Repetition
  5  FROM Product_Order
  6  JOIN Product
  7  ON Product_Order.Product_ID = Product.Product_ID
  8  GROUP BY Product.Product_ID, Product.Product_Name, Product.Product_Category
  9  ORDER BY Repetition DESC )
 10  WHERE ROWNUM <= 3;

PRODUCT_ID PRODUCT_NAME         PRODUCT_CATEGOR REPETITION
---------- -------------------- --------------- ----------
       310 Ultima Atom 520 Pro  Earbuds                  2
       399 Airpods Pro          Earbuds                  2
       359 T800 Ultra           Smart Watch              2
```

Figure 28 Top 3 product details that have been ordered the most.

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

Query:

```
SELECT * FROM

(SELECT Customer.Customer_ID, Customer.Customer_Name, Customer.Category,

COUNT(Product_Order.Order_ID) AS Number_of_Orders,

SUM(Invoice.Total) AS Total_Expense

FROM Customer

JOIN Product_Order

ON Customer.Customer_ID = Product_Order.Customer_ID

JOIN Orders

ON Product_Order.Order_ID = Orders.Order_ID

JOIN Invoice

ON Orders.Invoice_ID = Invoice.Invoice_ID

WHERE TO_CHAR(Orders.Order_Date, 'YYYY-MM') = '2023-08'

GROUP BY Customer.Customer_ID, Customer.Customer_Name, Customer.Category

ORDER BY Number_of_Orders DESC)

WHERE ROWNUM = 1;
```

```
SQL> SELECT * FROM
  2  (SELECT Customer.Customer_ID, Customer.Customer_Name, Customer.Category,
  3  COUNT(Product_Order.Order_ID) AS Number_of_Orders,
  4  SUM(Invoice.Total) AS Total_Expense
  5  FROM Customer
  6  JOIN Product_Order
  7  ON Customer.Customer_ID = Product_Order.Customer_ID
  8  JOIN Orders
  9  ON Product_Order.Order_ID = Orders.Order_ID
 10  JOIN Invoice
 11  ON Orders.Invoice_ID = Invoice.Invoice_ID
 12  WHERE TO_CHAR(Orders.Order_Date, 'YYYY-MM') = '2023-08'
 13  GROUP BY Customer.Customer_ID, Customer.Customer_Name, Customer.Category
 14  ORDER BY Number_of_Orders DESC)
 15  WHERE ROWNUM = 1;

CUSTOMER_ID CUSTOMER_NAME        CATEGORY   NUMBER_OF_ORDERS TOTAL_EXPENSE
----------- -------------------- ---------- ---------------- -------------
       1867 Bina Thakur          Regular                  2        105000
```

Figure 29 Customer who has ordered the most in August with his/her total spending on that month.

7. Critical Evaluation

7.1 Critical Evaluation of Module

Database is a collection of information that is organized in a way so that it can be easily accessed, managed, and updated. It entails evaluating the module's efficacy, efficiency, and usefulness in a real-world scenario. This module is required for selecting, evaluating, and implementing a commercial database management system (DBMS).

The programming principles and languages like SQL is used to access and modify a database. Databases are frequently used in combination with web development because they store and retrieve data that is displayed on websites. The designing, implementing, and maintaining an organization's database systems oversee database administrators. So, this module is closely linked with programming, web development, and software engineering.

The module helps to grow the understanding about database management system. It broadens the critical abilities required as a developer, such as designing, implementing, and managing a database system. The extensive usage of databases in various businesses and areas makes it a necessary skill for a student to acquire.

7.2 Critical Assessment of Coursework

The coursework is based on a conceptual database model which consists of various techniques and methods of designing, analyzing, implementing, and maintaining a database system. It involves the creation of objects as well as the discovery of entities and their attributes. It also includes the development of relationship types.

This assessment provided us with wide understanding of how to identify constraints such as null, not null, unique etc. and keys like Primary key, Foreign key, Unique key etc. and include them in the table. We learned about the process and different levels of normalization of the relationships. We were also able to learn about the difference between initial ERD which is made before normalization and final ERD which is made after normalization.

The database queries were carried out in Oracle SQL PLUS to enter, edit, store, retrieve, and run SQL commands which gave us insights about data manipulation in database system. We were able to create the database system for an online marketplace for electronic devices and accessories that stores information about customers, products, vendors, orders and payment details.

It was challenging to complete this assessment in the allotted time. We encountered several errors and there was shortage of time. However, it also assisted us in improving our time management skills and critical thinking as well as problem solving abilities.

8. Drop Query and Database Dump file creation.

The dump file was created after completion of designing and implementation of database system. The implementation of the system was done by creating tables and establishing relationship between them using primary keys and foreign keys.

```
Command Prompt

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

C:\Users\Rashi> cd C:\22067683_Rashi_Maharjan

C:\22067683_Rashi_Maharjan> exp Rashi_CW/rashi file = 22067683_Rashi_Maharjan.dmp

Export: Release 11.2.0.2.0 - Production on Sat Jan 13 23:07:24 2024

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


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user RASHI_CW
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user RASHI_CW
About to export RASHI_CW's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export RASHI_CW's tables via Conventional Path ...
. . exporting table                          CUSTOMER         10 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                           INVOICE          8 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                            ORDERS          8 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                     ORDER_DETAILS          8 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                           PRODUCT         10 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                     PRODUCT_ORDER         11 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                            VENDOR          7 rows exported
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.

C:\22067683_Rashi_Maharjan>
```

Figure 30 Creating a dump file.

Drop query:

```
SQL> DROP TABLE Product_Order;

Table dropped.

SQL> DROP TABLE Order_Details;

Table dropped.

SQL> DROP TABLE Orders;

Table dropped.

SQL> DROP TABLE Product;

Table dropped.

SQL> DROP TABLE Invoice;

Table dropped.

SQL> DROP TABLE Vendor;

Table dropped.

SQL> DROP TABLE Customer;

Table dropped.

SQL> SELECT * FROM TAB;

no rows selected
```

Figure 31 Dropping all tables.

9. References

Database Star, 2022. *Database Normalization.* [Online]
Available at: https://www.databasestar.com/database-normalization/
[Accessed 19 December 2023].

Gadget Emporium, 2017. *About Us.* [Online]
Available at: https://gadgetemporium.pk/about-us/
[Accessed 7 December 2023].

Geeksforgeeks, 2020. *Types of Normal Forms in DBMS.* [Online]
Available at: https://www.geeksforgeeks.org/types-of-normal-forms-in-dbms/
[Accessed 19 December 2023].

IBM, 2022. *Key concepts: Entity, attribute, and entity type.* [Online]
Available at: https://www.ibm.com/docs/en/imdm/12.0?topic=concepts-key-entity-attribute-entity-type
[Accessed 12 December 2023].

Lucidchart , 2023. *What is an Entity Relationship Diagram (ERD)?.* [Online]
Available at: https://www.lucidchart.com/pages/er-diagrams
[Accessed 18 December 2023].

Shrestha, P. L., 2023. *Worshop Slides.* Kathmandu: s.n.