



CC5051NI Databases
50% Individual Coursework

Autumn 2023

Student Name: Yasna Dongol

London Met ID: 22068112

Assignment Submission Date: 13th January, 2024 Saturday

Word Count: 4578

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

1. Introduction.....	1
1.1. Current Business Activities/Operation	1
1.2. Business Rule.....	3
1.3. Identification of Entities and Attributes	5
2. Initial ERD	6
3. Normalization	11
3.1. UNF (Unnormalized Form).....	12
3.2. 1NF(First Normal Form)	13
3.3. 2NF(Second Normal Form).....	14
3.4. 3NF (Third Normal Form).....	16
4. Final ERD	19
5. Implementation.....	21
6. Instruction query.....	29
7. Transaction query	32
8. Critical Evaluation	35
9. Drop Query.....	36
10. Database Dump file Creation	36
11. References	37

List of Figures

• Figure 1 Screenshot of Initial ERD of the Gadget Emporium	6
• Figure 2 Screenshot of the final ERD	21
• Figure 3 Screenshot of creating category table using sql query	21
• Figure 4 Screenshot of creating Vendor table using sql query	21
• Figure 5 Screenshot of creating Product table using sql query	21
• Figure 6 Screenshot of creating Product Availability table using sql query	22
• Figure 7 Screenshot of creating Customer table using sql query	22
• Figure 8 Screenshot of creating Payment Option table using sql query	22
• Figure 9 Screenshot of creating Orders table using sql query	22
• Figure 10 Screenshot of creating Invoice table using sql query	23
• Figure 11 Screenshot of creating OrderProduct table using sql query	23
• Figure 12 Screenshot of inserting values in the category table	24
• Figure 13 Screenshot of retrieving data from the category table	24
• Figure 14 Screenshot of inserting and retrieving data from the vendor table	24
• Figure 15 Screenshot of inserting data into product table	25
• Figure 16 Screenshot of retrieving data from the product table	25
• Figure 17 Screenshot of inserting data into the ProductAvailability table	25
• Figure 18 Screenshot of retrieving data from Product Availability table	25
• Figure 19 Screenshot of inserting data into customer table	26
• Figure 20 Screenshot of retrieving data from customer table	26
• Figure 21 Screenshot of inserting data into Payment Option table	26
• Figure 22 Screenshot of retrieving data from Payment Option table	26
• Figure 23 Screenshot of inserting and retrieving data from Orders table	27
• Figure 24 Screenshot of retrieving data from Orders table	27
• Figure 25 Screenshot of inserting data into Invoice table	27
• Figure 26 Screenshot of retrieving data from Invoice table	28
• Figure 27 Screenshot of inserting data into OrderProduct table	28
• Figure 28 Screenshot of retrieving data from OrderProduct table	28
• Figure 29 Screenshot of retrieving data from customer who are staff of a company	29
• Figure 30 Screenshot of retrieving orderdate from Orders between 01-05-2023 to 05-28-2023	29
• Figure 31 Screenshot of retrieving data from customer with order details	30
• Figure 32 Screenshot of retrieving product details that have the second letter 'a'	30
• Figure 33 Screenshot of retrieving data of customer that has ordered recently	31
• Figure 34 Screenshot of calculating total revenue of each month	32
• Figure 35 Screenshot of retrieving data from order which has equal or higher than the average order total value	32
• Figure 36 Screenshot of retrieving data from vendor who has supplied more than 3 products	33
• Figure 37 Screenshot of showing the product details that have been ordered most	33
• Figure 38 Screenshot of finding the customer who has ordered most in the August	34
• Figure 39 Screenshot of dropping all the tables	36
• Figure 40 Screenshot of Dump file creation	36

List of Tables

- Table 1 Relational Table of Customer Entity 7
- Table 2 Relational Table of Category Entity 7
- Table 3 Relational Table of Vendor Entity..... 8
- Table 4 Relational Table of Product Entity..... 8
- Table 5 Relational Table of ProductAvailability Entity 8
- Table 6 Relational Table of PaymentOption Entity 9
- Table 7 Relational Table of Orders Entity 9
- Table 8 Relational Table of Invoice Entity..... 9
- Table 9 Relational Table of OrderProduct Entity 10

Introduction

In the fast-paced world of technology, **“Gadget Emporium”** has been established as a leading light for both private and public consumers looking for a comprehensive online marketplace for electronic goods. It is considered as prominent destination for purchase of wide range of electronic goods in the market. It was founded by a tech enthusiast named Michael Smith who has contributed in it's growth and development since past 8 years. The founder of the online electronic store developed a platform that makes the buying of the goods convenient for the consumers.

The goal of Gadget Emporium is to extend the collection of electronic equipment, excellent customer support, creating an environment where consumers can share and exchange ideas in their tech journeys. This online marketplace has been serving its customers for a long time and is skilled at adding, changing, and removing products and categories. It also registers customers for appropriate discounts based on their categories, records order details and tracks product availability in real-time, also incorporates with payment gateways to ensure safe transactions and valid invoices at the point of purchase.

The online marketplace Gadget Emporium assures to commit to database design and the implementation of strong business rules where the system guarantees smooth operations in several important areas.

Current Business Activities/Operation

In Mr.John e-commerce endeavor “Gadget Emporium” online business store it allows customers searching for electronics gadgets to buy electronics or accessories with a smooth and simple online shopping experience. Customers/buyers can have a variety of products in which all of the products are categorized according to their specific category so that it makes the customer a smooth buying experience saving their time. Once the customer has decided to which category of product they want to buy from Mr.John online store,they can move forward on order placing by providing their details including the total

number of each selected item they wanted to buy and specify the payment option according to customer's preference. "Gadget Emporium" online marketplace system is designed in such a way that it records the orders of each product customer has ordered. For storing order details it includes important information such as name of the product, quantities of selected item and unit price of the selected item and their total amount of the entire order of the gadgets and accessories. On every order processing system the customers are assigned with some discount rate(0%,5%,10%) according to their category(Regular,Staff,VIP) on their product purchase. This system involves real-time inventory management to maintain control over its stock levels. By real-time tracking system it figures out whether the stock precisely reflects the current status or not so as to prevent it from overselling. It directly keeps the links with the vendors to obtain the high quality product for the online business. Each and every product comes from the specific vendor so each product is linked with the vendors. There are multiple ways for customers to pay for the product when they purchase gadget accessories through the system. They select the best payment method, such as cash on delivery,e-wallets, or credit/debit cards according to the customer's capacity to pay. Having a variety of payment options available to our system customer's aids in their smooth and safe transaction experiences. The database system allows "Gadget Emporium" to operate flexibly and successfully in the competitive online electronics market by effectively managing products, orders, vendors and payments.

Business Rule

❖ Product Management:

- Product management is a business function that mainly focuses on the development of ongoing success of a product.
- Each product comes into a single category but there may be several products in a single category. So, this product and category has a Many-to-One relationship between them.

❖ Customer Categories and Discount:

- Customers are categorized as Regular (R),Staff (S), and VIP (V) based on their category; they are provided with some different discount rate as 0%, 5% and 10% on each and every product purchase.
- One customer can order many products but each order is connected with one customer.So, this Customer and order have One-to-Many relationship between them

❖ Order Processing:

- One customer can order many products but each order is connected with one customer.So, this Customer and order have One-to-Many relationship between them.
- Each and every order in a system should have a unique verification and for every order a customer is linked with that order.

❖ Vendor Management:

- Each product available in a system should be associated with one and only vendor but a vendor can provide multiple products(electronic gadgets). So, Product and Vendor have a Many-to-One relationship among them.

❖ Product Availability and Inventory Management:

- Product availability tracks the product stock quantity whether it is in stock or out of stock. It has a direct connection with product after order of each product the stock quantity is maintained.
- There is a single stock quantity availability record for every product in the system. Each stock quantity availability record is connected to a single product. So, product and product availability has a One-to-One relationship between them.

❖ Payment Processing:

- For every order processing different payment processes have been assigned in a system for secure and smooth transactions.
- For every order of an electronic gadget they can only have one payment option for payment and each payment option is linked with one order. So, payment and order have One-to-One relationship between them.

❖ Invoice:

- Once the order processing is finished an invoice is generated which is sent to the customer .
- Each payment option for order can be included in a single invoice but each invoice and payment option should be connected with each other. So, payment option and invoice have One-to-One relationship between them.

Identification of Entities and Attributes

The identification of Entities and attributes are the most important step in the process of creating database structure. Entities are the objects in the real world that we want to store the data in a database. Each entity correlates to a table in relational data whereas attributes are the properties of an entity. Attributes describe the data we want to store about each entity.

Entities that are present in this ER diagram are: Product, Customer, Vendor, Category, Orders, OrderProduct, Invoice, PaymentOption etc. Attributes that are present in the ER diagram defining the properties of entities are: ProductName, ProductID, Description, Unitprice, CustomerID, Fullname, Category, VendorID, VendorName etc.

The relationships are created between the entities to maintain the data integrity by defining rules. Relationships also support the normalization process. It helps to retrieve data from multiple tables. If the relationships are well defined it makes the database schema more maintainable and extensible. (silberschatz)

Initial ERD

Initial ERD is a graphical representation of entity sets, attributes and relationships among the entities. Nodes in the graph represent the elements of each of these kinds, we use special geometric shapes to represent the entity, attribute and relationship. (Watt)

- Entity sets are represented by rectangles.
- Attributes are represented by ovals.
- Relationships are represented by diamonds.
- Each entity is connected to its attributes and also in a relationship to its entity from edges.

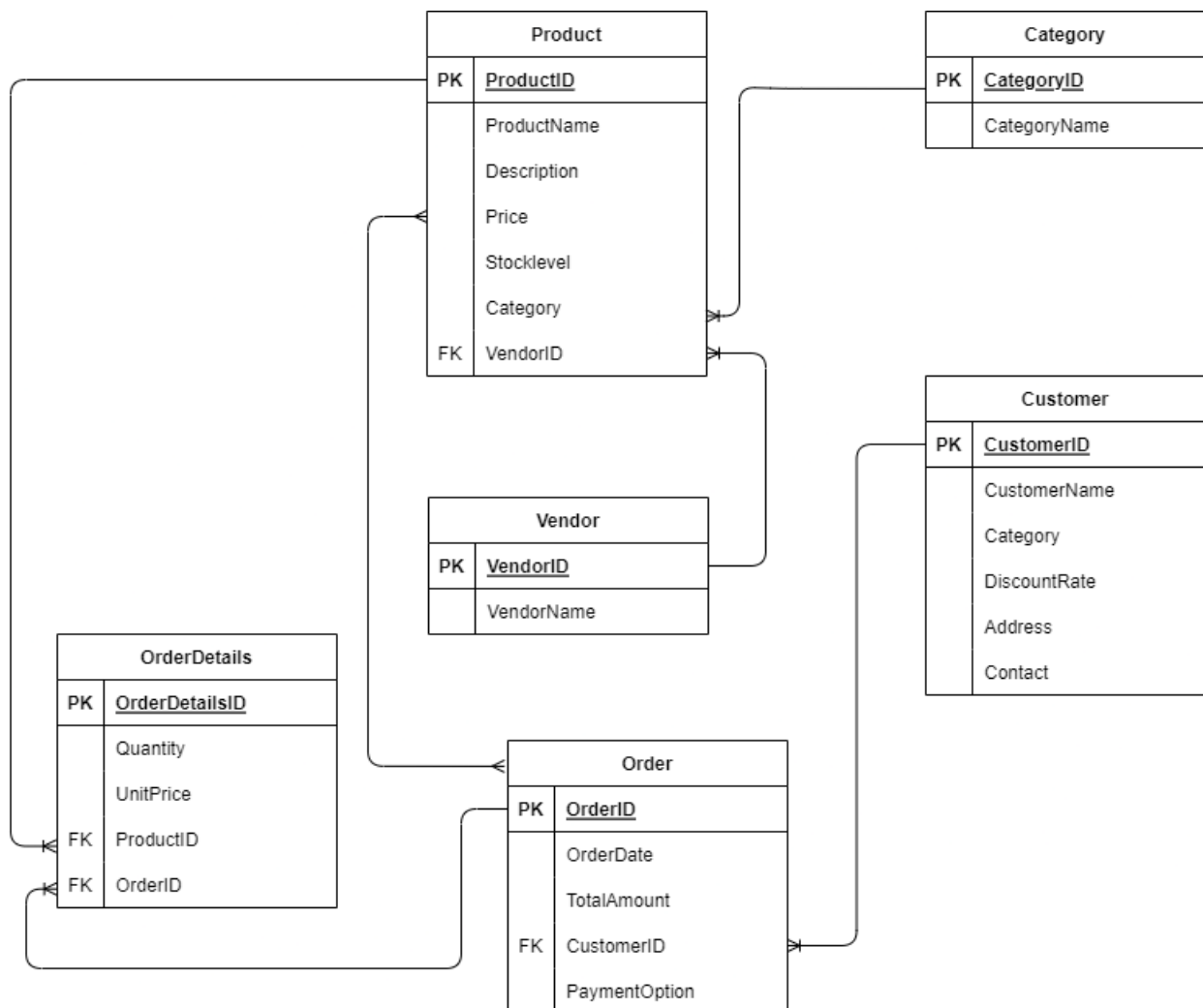


Figure 1 Screenshot of Initial ERD of the Gadget Emporium

In this above Initial ERD we can clearly see that there are six entities and each entity contains its attributes and having relationships among them. Each entity are connected with another entity through edges having relationship like:-

1. One-to-One (1-to-1)
2. One-to-Many (1-to-M)
3. Many-to-Many (M-to-M)

Following are the relational tables of created objects and their attributes with primary/foreign key identification along with its respective Datatype:

➤ Customer

Table 1 Relational Table of Customer Entity

Entity	Attributes	Data Type	Constraints
Customer	CustomerID	VARCHAR(20)	PK NOT NULL
	Fullname	VARCHAR(255)	NOT NULL
	Address	VARCHAR(255)	NOT NULL
	DiscountRate	VARCHAR(15)	NOT NULL
	Category	VARCHAR(20)	NOT NULL

➤ Category

Table 2 Relational Table of Category Entity

Entity	Attributes	Data Type	Constraints
Category	CategoryID	VARCHAR(10)	PK NOT NULL
	Categoryname	VARCHAR(255)	NOT NULL

➤ Vendor

Table 3 Relational Table of Vendor Entity

Entity	Attributes	Data Type	Constraints
Vendor	VendorID	VARCHAR(10)	PK NOT NULL
	Vendorname	VARCHAR(255)	NOT NULL
	VendorAddress	VARCHAR(255)	NOT NULL
	Contactno	VARCHAR(20)	NOT NULL

➤ Product

Table 4 Relational Table of Product Entity

Entity	Attributes	Data Type	Constraints
Product	ProductID	VARCHAR(20)	PK NOT NULL
	ProductName	VARCHAR(255)	NOT NULL
	Description	VARCHAR(255)	NOT NULL
	Unitprice	DECIMAL(10,2)	NOT NULL
	Stocklevel	INT	NOT NULL
	CategoryID	VARCHAR(20)	FK
	VendorID	VARCHAR(20)	FK

➤ ProductAvailability

Table 5 Relational Table of ProductAvailability Entity

Entity	Attributes	Data Type	Constraints
ProductAvailability	ProductID	VARCHAR(20)	PK,FK NOT NULL
	ProductName	VARCHAR(255)	NOT NULL
	StockQuantity	INT	NOT NULL
	AvailabilityStatus	VARCHAR(255)	NOT NULL

➤ PaymentOption

Table 6 Relational Table of PaymentOption Entity

Entity	Attributes	Data Type	Constraints
PaymentOption	PaymentOption	VARCHAR(20)	PK NOT NULL
	OptionName	VARCHAR(255)	NOT NULL

➤ Orders

Table 7 Relational Table of Orders Entity

Entity	Attributes	Data Type	Constraints
Orders	OrderID	VARCHAR(5)	PK NOT NULL
	OrderDate	DATE	NOT NULL
	TotalAmount	DECIMAL(10,2)	NOT NULL
	CustomerID	VARCHAR(5)	FK NOT NULL
	PaymentOptionID	VARCHAR(5)	FK NOT NULL

➤ Invoice

Table 8 Relational Table of Invoice Entity

Entity	Attributes	Data Type	Constraints
Invoice	InvoiceID	VARCHAR(20)	PK NOT NULL
	OrderID	VARCHAR(20)	FK NOT NULL
	PaymentOptionID	VARCHAR(20)	FK NOT NULL
	CustomerID	VARCHAR(20)	FK NOT NULL
	TotalAmount	DECIMAL(10,2)	NOT NULL
	DiscountAmount	DECIMAL(10,2)	NOT NULL
	FinalAmount	DECIMAL(10,2)	NOT NULL
	InvoiceDate	DATE	NOT NULL

➤ OrderProduct

Table 9 Relational Table of OrderProduct Entity

Entity	Attributes	Data Type	Constraints
OrderProduct	OrderID	VARCHAR(20)	PK,FK NOT NULL
	ProductID	VARCHAR(20)	PK, FK NOT NULL
	Quantity	INT	NOT NULL
	UnitPrice	DECIMAL(10,2)	NOT NULL

Normalization

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. It can be considered as a “filtering” or “purification” process to make the design have successively better quality.

Hence normalization of data is a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of:

- Minimizing redundancy and
- Minimizing the insertion, deletion and update anomalies.

We assume that a set of functional dependencies is given for each relation, and that each relation has a designated primary key. Each relation is then evaluated for adequacy and decomposed further as needed to achieve higher normal forms, using the normalization theory. (JavaTpoint, 2011-2021)

Advantages of Normalization:

- It is used to reduce the null values.
- It is used to eliminate or reduce redundancy in database tables.
- It organizes tables in a systematic manner by decomposing tables.

Normal forms:

The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized. Certain rules in the database management system design have been developed to better organize tables and minimize anomalies. The stage at which the table is organized is known as its normal form or a stage of normalization.

There are different types of normal forms which are as follows:

1. First Normal Forms(1NF)
2. Second Normal Forms(2NF)
3. Third Normal Forms(3NF)

- A relation is said to be in a particular normal form if it satisfies the set of this particular normal form's constraints.
- In practical terms, the third normal form is sufficient to design a reliable database system.

UNF (Unnormalized Form)

In database Normalization, UNF stands for Unnormalized form or also called as non-first normal form. UNF is a database data model which does not meet the relation model defined by the database normalization. In the relation model, unnormalized relation can be overview as the starting point of a process of normalization. In this UNF form model suffer problem like data redundancy which lacks the efficiency of database normalization. (DBS211, n.d.)

Rules:

From the context of the information provided by the e-commerce system the Unnormalized form might look like a set of tables, getting every information without implementing normalization or redundancy rules into consideration. All necessary attributes may be included in each table without needing to be divided into different tables.

(ProductID, Name, Description, UnitPrice, StockLevel, OrdersID, CategoryID, VendorID, Quantity, PaymentOptionID, StockQuantity, AvailabilityStatus, InvoiceID {VendorName, CategoryName, VendorAddress, DiscountRate, OrderDate, TotalAmount, CustomerName, CustomerAddress, Category, DiscountRate, OptionName})

In the above UNF it shows the repeating attributes and non repeating attributes with respect to their particular tables. This attributes describes the properties of the entity. The repeating element are kept inside the curly bracket {} whereas non repeating are kept as it is. Due to the repetition of the attributes it doesn't follow the normalization rules and can lead to data redundancy as data is repeated. Therefore, to achieve

normalization and move to 1NF we should remove these repeating attributes and create a separate table for them.

1NF(First Normal Form)

In database normalization, 1NF stands for First Normal form. A relation is in 1NF form if it doesn't contain any composite or multi-valued attribute. If it contains composite or multi-valued attribute it violates 1NF rule. (geeksforgeeks, 2023)

Rules:

- A relation is in first normal form if every attribute in that relation is a single valued attribute.
- The domain of an attribute must include only atomic (simple, indivisible) values. (Normalizing the above Unnormalized form into 1NF form by removing multivalued attributes by ensuring that each attribute contains atomic values. (JavaTpoint, 2023)

Product -1(ProductID, Name, Description, Price, StockLevel, CategoryID*, VendorID*)

Customer -1(CustomerID, Name, Address, Category, DiscountRate)

Order -1(OrderID, OrderDate, TotalAmount, CategoryID*, PaymentOptionID*)

OrderProduct -1(OrderID*, ProductID*, Quantity, Unitprice)

Vendor -1(VendorID, VendorName, VendorAddress)

Category -1(CategoryID, Name)

PaymentOption -1(PaymentOptionID, OptionName)

ProductAvailability -1(ProductID*, stockQuantity, AvailabilityStatus)

Invoice -1(InvoiceID, OrderID*, PaymentOptionID*, OrderDate, TotalAmount)

The above UNF is normalized into 1NF form by eliminating the repeating group by breaking down the original table into a smaller table, more atomic table which has its single text.

2NF(Second Normal Form)

2NF stands for Second Normal Form. To be in the second normal form, a relation must be in 1NF form and the relation must not contain any partial dependency (non-prime attribute functionally dependent on any candidate key).is dependent on any proper subset of any candidate key of the table.

Partial Dependency: If the proper subset of candidate key determines non-prime attribute, it is called partial dependency. (byju's.com, 2023)

Rules:

- The table should be in First Normal Form.
- There should be no partial dependency.

If the table contains some attributes which are partially dependent on the primary key of that table then it is not in 2NF.

Since the above tables are in first normal form now to obtain second normal form we need to ensure that those attributes that are not the part of candidate key are fully functionally dependent on the entire primary key.

Here, in the above table all the tables are in the 2NF form. Since, non-prime attributes don't depend upon a proper subset of the candidate key, which means that there is no partial dependency.

1. In Product table:

Product -2(ProductID, Name, Description, Price, StockLevel, CategoryID*, VendorID*).

It is in 2NF form, since primary key → ProductID

Name, Description, Price, StockLevel are the attributes that are functionally dependent on its primary key(ProductID) and 'CategoryID' and 'VendorID' column are foreign key referencing another table, which are properly defined on the reference table (category and vendor).

2. In Vendor Table:

Vendor -2(VendorID, VendorName)

Primary key → VendorID

VendorName is the attribute that is functionally dependent on its primary key.

3. In Category Table:

Category -2(CategoryID, CategoryName)

Primary key → CategoryID

CategoryName is the attribute that is functionally dependent on its primary key.

4. In Customer Table:

Customer -2(CustomerID, Name, Address, Category, DiscountRate)

Primary key → CustomerID

The above customer table is in 2NF form since its attributes are functionally dependent on its primary key.

5. In Orders Table:

Orders -2(OrderID, OrderDate, TotalAmount, CategoryID*, PaymentOptionID*)

Primary key → OrderID

The above customer table is in 2NF form since its attributes are functionally dependent on its primary key and 'CategoryID' and 'PaymentOptionID' columns are foreign key referencing another table, which are properly defined on the reference table (category and PaymentOption).

6. In Invoice Table:

Invoice -2(InvoiceID, OrderDate, TotalAmount)

Primary key → InvoiceID

The above customer table is in 2NF form since its attributes are functionally dependent on its primary key.

7. In PaymentOption Table:

PaymentOption -2(PaymentOptionID, OptionName)

Primary key → PaymentOption

OptionName is the attribute that is functionally dependent on its primary key.

8. In OrderProduct Table:

OrderProduct -2(OrderID*,ProductID*, Quantity, Unitprice)

Primary key → (OrderID,ProductID)

The above OrderProduct table is in 2NF form since its attributes are functionally dependent on its primary key and 'OrderID' and 'ProductID' columns are foreign key referencing another table, which are properly defined on the reference table (Order and Product).

9. In ProductAvailability Table:

ProductAvailability -2(ProductID*, stockQuantity, AvailabilityStatus)

Primary key → ProductID

The above ProductAvailability table is in 2NF form since its attributes are functionally dependent on its primary key and 'ProductID' columns is foreign key referencing another table, which are properly defined on the reference table Product.

3NF (Third Normal Form)

Third Normal Form is a database schema design based approach for relational databases that uses the principle of normalizing to remove transitive partial dependency. This means a non-prime attribute (an attribute that is not part of the candidate's key) is dependent on another non-prime attribute. This is what the third normal form (3NF) eliminates. (Chris, 2022)

Transitive dependency:

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. $X \rightarrow Y$ is a transitive dependency if the following three functional dependencies holds true:

- If $X \rightarrow Y$
- And $Y \rightarrow Z$
- Then $X \rightarrow Z$

Rules:

1. Table must be in 2NF form.
2. It should not have transitive dependency.

Since the above tables are in Second normal form now to obtain third normal form we need to eliminate the transitive dependencies.

Here, in the above table all the tables are in the 3NF form. Since, transitive dependencies are not present.

- In Product table:

Product -3(ProductID, Name, Description, Price, StockLevel, CategoryID*, VendorID*).

There are no transitive dependencies present and ProductID → primary key where all the other attributes are dependent on it.

- In Vendor Table:

Vendor -3(VendorID, VendorName)

The above table is in 3NF form and VendorID → primary key where its attribute VendorName is directly dependent on it.

- In Category Table:

Category -3(CategoryID, CategoryName)

Primary key → CategoryID

Here, The above table is in 3NF form and its attribute is directly dependent on its primary key.

- In Customer Table:

Customer -3(CustomerID, Name, Address, Category, DiscountRate)

Primary key → CustomerID

No transitive dependencies are present and all attributes are directly dependent on the primary key.

- In Orders Table:

Orders-3(OrderID, OrderDate, TotalAmount, CategoryID*, PaymentOptionID*)

Primary key → OrderID

No transitive dependencies are present and all attributes are directly dependent on the primary key.

- In Invoice Table:

Invoice-3(InvoiceID, OrderID*, PaymentOptionID*, OrderDate, TotalAmount)

Primary key → InvoiceID

In this Invoice table OrderID and PaymentOptionID is added to remove transitive dependency on Order and PaymentOption attributes.

- In PaymentOption Table:

PaymentOption -3(PaymentOptionID, OptionName)

Primary key → PaymentOption

No transitive dependencies are present and all attributes are directly dependent on the primary key.

- In OrderProduct Table:

OrderProduct -3(OrderID*, ProductID*, Quantity, Unitprice)

Primary key → (OrderID, ProductID)

No transitive dependencies are present and all attributes are directly dependent on the primary key.

- In ProductAvailability Table:

ProductAvailability -3(ProductID*, stockQuantity, AvailabilityStatus)

Primary key → ProductID

No transitive dependencies are present and all attributes are directly dependent on the primary key.

Final ERD

Final Entity-Relationship Diagram is a visual representation of a database model of a system or database, which provides a full view of the entities, attributes, relationships and constraints by defining how data stores in the database and how it interacts with the system. ERD should be developed first before designing any database so it is also called as the blueprint of the database design. Final ERD is most generated after the normalization ensuring that there are no multivalued attributes, partial dependency and transitive dependency.

In this Mr.John "Gadget Emporium" database system we have created an entity-relationship diagram after the normalization by ensuring that there are no multivalued attributes, partial dependency and transitive dependency. For his database system we have created nine entities and a property describing its entities called as an attribute and having relationships among the entities so that it helps to understand the base of the data or information that is going to store on a database and also helps for the systematic interaction on the system.

In this database model the product table is created so that it manages the category of product according to its categories. So it has many to one relation with the category. This category table helps customers to find what type of product they want to buy according to the basis of their customer. After selecting the category of the product the order table helps us to place the order. Order and customer are linked together and have one to many relationships between them. Each product is associated with the vendor table which ensures that each product comes from the specific vendors. This vendor has many-to-one relationships among them. The product availability table helps to manage inventory so that there is no any overselling of the product and helps to stock the product if the quantities are out of stock. Product availability table has a one-to-one relationship among them. After all the order placement is completely finished different payment options are provided to pay for the products by this ProductOption table which helps for the smooth and secure transaction. (Navathe)

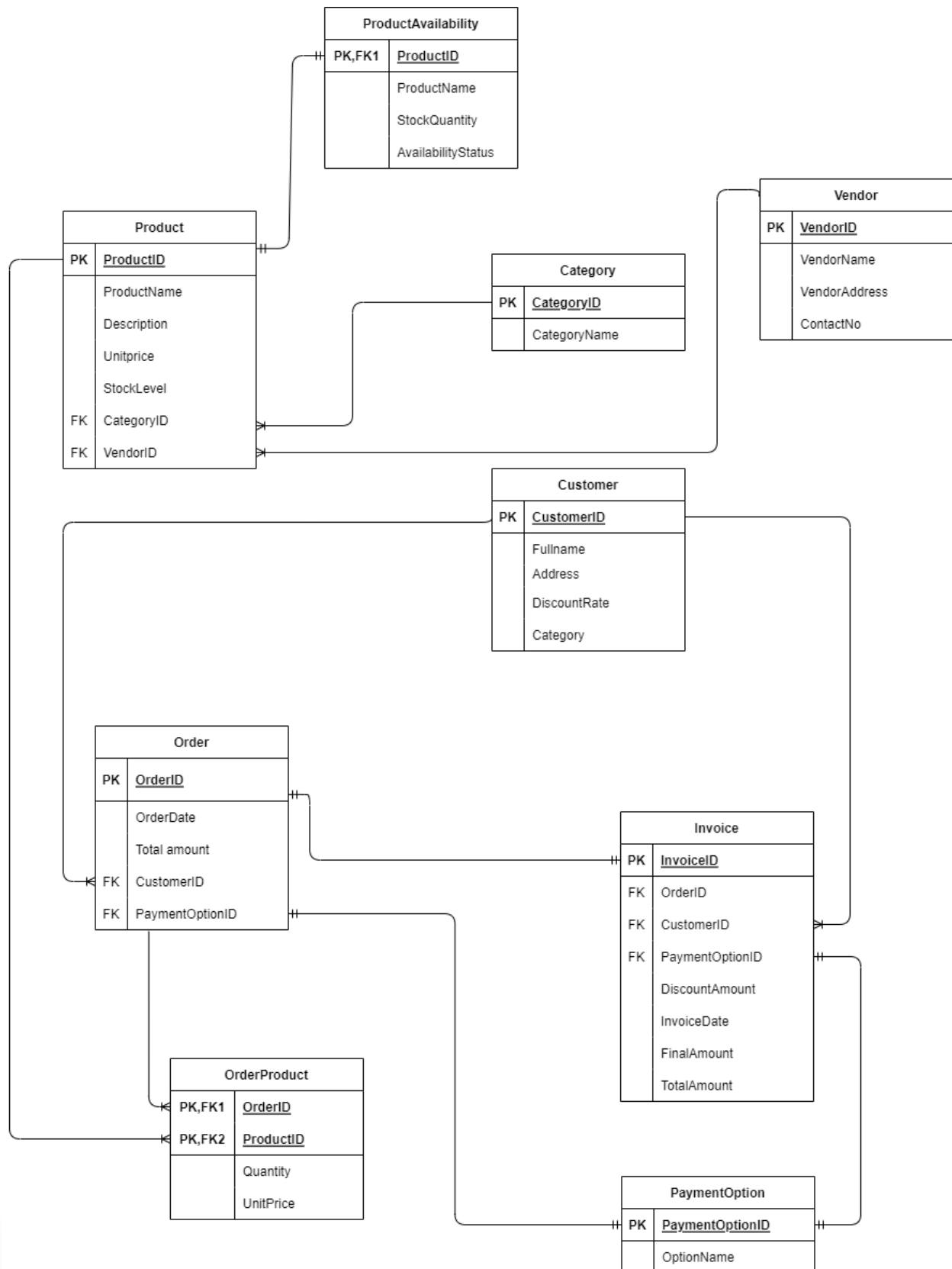


Figure 2 Screenshot of the final ERD

Implementation

- Creating tables of the entities present on the database system using sql query on Oracle Sql Plus.

Creating Category table:

```
SQL> create table Category(CategoryID varchar(10), Categoryname varchar(255),primary key (CategoryID));
Table created.

SQL> desc category;
  Name                                Null?    Type
-----
CATEGORYID                          NOT NULL VARCHAR2(10)
CATEGORYNAME                          VARCHAR2(255)

SQL>
```

Figure 3 Screenshot of creating category table using sql query

Creating Vendor table:

```
SQL> create table Vendor(VendorID varchar(10), Vendorname varchar(255),Vendoraddress varchar(255), Contactno varchar(20), primary key (VendorID));
Table created.

SQL> desc vendor;
  Name                                Null?    Type
-----
VENDORID                          NOT NULL VARCHAR2(10)
VENDORNAME                          VARCHAR2(255)
VENDORADDRESS                       VARCHAR2(255)
CONTACTNO                           VARCHAR2(20)

SQL>
```

Figure 4 Screenshot of creating Vendor table using sql query

Creating Product table:

```
SQL> create table Product(ProductID varchar(20), ProductName varchar(255), Description varchar(255), Unitprice decimal(10, 2), Stocklevel int, CategoryID varchar(20), VendorID varchar(20), primary key (ProductID), foreign key (CategoryID) references Category(CategoryID), foreign key (VendorID) references Vendor(VendorID));
Table created.

SQL> desc product;
  Name                                Null?    Type
-----
PRODUCTID                          NOT NULL VARCHAR2(20)
PRODUCTNAME                          VARCHAR2(255)
DESCRIPTION                          VARCHAR2(255)
UNITPRICE                           NUMBER(10,2)
STOCKLEVEL                           NUMBER(38)
CATEGORYID                          VARCHAR2(20)
VENDORID                             VARCHAR2(20)

SQL>
```

Figure 5 Screenshot of creating Product table using sql query

Create Product Availability table:

```
SQL> create table ProductAvailability(ProductID varchar(20), ProductName varchar(255), StockQuantity int, AvailabilityStatus varchar(255), primary key (ProductID), foreign key (ProductID) references Product(ProductID));
```

Table created.

```
SQL> desc productavailability;
```

Name	Null?	Type
PRODUCTID	NOT NULL	VARCHAR2(20)
PRODUCTNAME		VARCHAR2(255)
STOCKQUANTITY		NUMBER(38)
AVAILABILITYSTATUS		VARCHAR2(255)

Activate Windows
Go to Settings to activate Windows.

Figure 6 Screenshot of creating Product Availability table using sql query

Create Customer Table:

```
SQL> create table Customer(CustomerID varchar(20), Fullname varchar(255), Address varchar(255), DiscountRate varchar(15), Category varchar(20), primary key (CustomerID));
```

Table created.

```
SQL> desc Customer;
```

Name	Null?	Type
CUSTOMERID	NOT NULL	VARCHAR2(20)
FULLNAME		VARCHAR2(255)
ADDRESS		VARCHAR2(255)
DISCOUNTRATE		VARCHAR2(15)
CATEGORY		VARCHAR2(20)

Activate Windows
Go to Settings to activate Windows.

Figure 7 Screenshot of creating Customer table using sql query

Create PaymentOption Table:

```
SQL> create table PaymentOption(PaymentOptionID varchar(20), OptionName varchar(255), primary key (PaymentOptionID));
```

```
SQL> desc paymentoption;
```

Name	Null?	Type
PAYMENTOPTIONID	NOT NULL	VARCHAR2(20)
OPTIONNAME		VARCHAR2(255)

SQL> _

Figure 8 Screenshot of creating Payment Option table using sql query

Create Orders Table:

```
SQL> create table Orders(OrderID varchar(5), OrderDate date, TotalAmount decimal(10,2), CustomerID varchar(5), PaymentOptionID varchar(5), primary key (OrderID), foreign key (CustomerID) references Customer(CustomerID), foreign key (PaymentOptionID) references PaymentOption(PaymentOptionID));
```

Table created.

```
SQL> desc Orders;
```

Name	Null?	Type
ORDERID	NOT NULL	VARCHAR2(5)
ORDERDATE		DATE
TOTALAMOUNT		NUMBER(10,2)
CUSTOMERID		VARCHAR2(5)
PAYMENTOPTIONID		VARCHAR2(5)

SQL> _

Figure 9 Screenshot of creating Orders table using sql query

Create Invoice Table:

```
SQL> create table Invoice(InvoiceID varchar(20), OrderID varchar(20), PaymentOptionID varchar(20), CustomerID varchar(20), TotalAmount decimal(10, 2), DiscountAmount decimal(10, 2), FinalAmount decimal(10, 2), InvoiceDate date, primary key (InvoiceID), foreign key (OrderID) references Orders(OrderID), foreign key (PaymentOptionID) references PaymentOption(PaymentOptionID), foreign key (CustomerID) references Customer(CustomerID));
```

```
SQL> desc invoice;
Name                               Null?    Type
-----
INVOICEID                          NOT NULL VARCHAR2(20)
ORDERID                            VARCHAR2(20)
PAYMENTOPTIONID                    VARCHAR2(20)
CUSTOMERID                         VARCHAR2(20)
TOTALAMOUNT                        NUMBER(10,2)
DISCOUNTAMOUNT                   NUMBER(10,2)
FINALAMOUNT                        NUMBER(10,2)
INVOICEDATE                        DATE
```

SQL>

Figure 10 Screenshot of creating Invoice table using sql query

Create OrderProduct Table:

```
SQL> create table OrderProduct(OrderID varchar(20), ProductID varchar(20), Quantity int, UnitPrice decimal(10, 2), primary key (OrderID , ProductID), foreign key (OrderID) references Orders(OrderID), foreign key (ProductID) references Product(ProductID));
```

Table created.

```
SQL> desc OrderProduct;
Name                               Null?    Type
-----
ORDERID                            NOT NULL VARCHAR2(20)
PRODUCTID                          NOT NULL VARCHAR2(20)
QUANTITY                           NUMBER(38)
UNITPRICE                           NUMBER(10,2)
```

SQL> _

Figure 11 Screenshot of creating OrderProduct table using sql query

- Inserting/Retrieving the data using the sql query on Oracle Sql Plus

Inserting value into Category table:

```
SQL> insert all
  2 into Category(CategoryID,CategoryName) values('1', 'Television')
  3 into Category(CategoryID,CategoryName) values('2', 'Speakers')
  4 into Category(CategoryID,CategoryName) values('3', 'Accessories')
  5 into Category(CategoryID,CategoryName) values('4', 'Electronics')
  6 into Category(CategoryID,CategoryName) values('5', 'Gaming Accessories')
  7 into Category(CategoryID,CategoryName) values('6', 'Mobile Phones')
  8 into Category(CategoryID,CategoryName) values('7', 'Airpods')
  9 select * from dual;

7 rows created.
```

Figure 12 Screenshot of inserting values in the category table

```
SQL> select * from category;

CATEGORYID CATEGORYNAME
-----
1          Television
2          Speakers
3          Accessories
4          Electronics
5          Gaming Accessories
6          Mobile Phones
7          Airpods

7 rows selected.
```

Figure 13 Screenshot of retrieving data from the category table

Inserting into vendor table:

```
SQL> insert all
  2 into Vendor(VendorID,VendorName,VendorAddress,Contactno) values('V01', 'Samsung', 'Dallas', 1234567890)
  3 into Vendor(VendorID,VendorName,VendorAddress,Contactno) values('V02', 'Soundmaster', 'Sydney', 5555679876)
  4 into Vendor(VendorID,VendorName,VendorAddress,Contactno) values('V03', 'LG', 'Toronto', 0015991256)
  5 into Vendor(VendorID,VendorName,VendorAddress,Contactno) values('V04', 'Agilis electronics', 'Dallas', 7869870912)
  6 into Vendor(VendorID,VendorName,VendorAddress,Contactno) values('V05', 'Gamer Electronics', 'Portland',2354519876)
  7 into Vendor(VendorID,VendorName,VendorAddress,Contactno) values('V06', 'ansay Mobiles', 'Austin',3461209575)
  8 into Vendor(VendorID,VendorName,VendorAddress,Contactno) values('V07', 'Boat', 'Adelaide',6716349823)
  9 select * from dual;

7 rows created.
```

Figure 14 Screenshot of inserting and retrieving data from the vendor table

Insert into product table:

```
SQL> into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P01', 'Samsung Neo QLED', 'Discover the latest 8k samsung neo QLED Technol
ogy', '1197.99', '60', '1', 'V01')
SP2-0734: unknown command beginning "into Produ..." - rest of line ignored.
SQL> insert all
  2 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P01', 'Samsung Neo QLED', 'Discover the latest 8k samsung neo QLED Technol
ogy', '1197.99', '60', '1', 'V01')
  3 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P02', 'Samsung S23', 'Experience the latest galaxy device', '350.99', '54'
, '6', 'V01')
  4 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values ('P03', 'JBL Speaker', 'Good quality sound', '250.56', '40','2', 'V02')
  5 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values ('P04', 'iphone 15', 'best quality phone', '1120', '70','6', 'V04')
  6 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P05', 'laptop', 'Best quality laptop', '1265.22', '55', '3', 'V04')
  7 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P06', 'Airpod', 'Background noise cancelation', '350', '99', '7', 'V07')
  8 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P07', 'GTX 3060ti', 'Smooth gaming', '1253.56', '8', '5', 'V05')
  9 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P08', 'POCO X4', 'Enjoy the new technology', '125.26', '3', '6', 'V06')
 10 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P09', 'PSS', 'Advance gaming', '786', '20', '5', 'V04')
 11 into Product(ProductID,ProductName,Description,Unitprice,Stocklevel, CategoryID,VendorID) values('P10', 'VR box', 'Enjoy the virtual reality', '560', '75', '5', 'V04')
 12 select * from dual;

10 rows created.
```

Figure 15 Screenshot of inserting data into product table

```
SQL> select * from Product;
```

PRODUCTID	PRODUCTNAME	DESCRIPTION	UNITPRICE	STOCKLEVEL	CATEGORYID	VENDORID
P01	Samsung Neo QLED	Discover the latest 8k samsung neo QLED Technology	1197.99	60	1	V01
P02	Samsung S23	Experience the latest galaxy device	350.99	54	6	V01
P03	JBL Speaker	Good quality sound	250.56	40	2	V02
P04	iphone 15	best quality phone	1120.00	70	6	V04
P05	laptop	Best quality laptop	1265.22	55	3	V04
P06	Airpod	Background noise cancelation	350.00	99	7	V07
P07	GTX 3060ti	Smooth gaming	1253.56	8	5	V05
P08	POCO X4	Enjoy the new technology	125.26	3	6	V06
P09	PSS	Advance gaming	786.00	20	5	V04
P10	VR box	Enjoy the virtual reality	560.00	75	5	V04

```
10 rows selected.
SQL>
```

Figure 16 Screenshot of retrieving data from the product table

Insert into Product Availability table:

```
SQL> insert all
  2 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P01', 'Samsung Neo QLED', '57', 'Instock')
  3 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P02', 'Samsung S23', '51', 'Instock')
  4 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P03', 'JBL Speaker', '25', 'Instock')
  5 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P04', 'iphone 15', '60', 'Instock')
  6 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P05', 'laptop', '0', 'Outofstock')
  7 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P06', 'Airpod', '82', 'Instock')
  8 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P07', 'GTX 3060ti', '0', 'Outofstock')
  9 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P08', 'POCO X4', '0', 'Outofstock')
 10 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P09', 'PSS', '0', 'Outofstock')
 11 into ProductAvailability(ProductID,ProductName,StockQuantity,AvailabilityStatus) values('P10', 'VR box', '70', 'Instock')
 12 select * from dual;

10 rows created.
```

Figure 17 Screenshot of inserting data into the ProductAvailability table

```
SQL> select * from Productavailability;
```

PRODUCTID	PRODUCTNAME	STOCKQUANTITY	AVAILABILITYSTATUS
P01	Samsung Neo QLED	57	Instock
P02	Samsung S23	51	Instock
P03	JBL Speaker	25	Instock
P04	iphone 15	60	Instock
P05	laptop	0	Outofstock
P06	Airpod	82	Instock
P07	GTX 3060ti	0	Outofstock
P08	POCO X4	0	Outofstock
P09	PSS	0	Outofstock
P10	VR box	70	Instock

```
10 rows selected.
```

Figure 18 Screenshot of retrieving data from Product Availability table

Insert into customer table:

```

SQL> insert all
2  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C01', 'Eyan Owen', '579 RainbowDrive','0%','Regular')
3  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C02', 'Johnson Nirvik', '1932 Cherry Camp Road', '0%', 'Regular')
4  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C03', 'Endrick Paul', '2663 Stonky Lane', '0%', 'Regular')
5  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C04', 'Johnson Linda', '3641 Freed Drive', '5%', 'Staff')
6  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C05', 'Reece James', '39 Northgate street', '5%', 'Staff')
7  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C06', 'Lauren James', '11 Hudson St', '10%', 'VIP')
8  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C07', 'Mason Mount', '98 Consett Rd', '10%', 'VIP')
9  into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C08', 'Malung Sarr', '34 Dossiter Street', '5%', 'Staff')
10 into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C09', 'Koku Harold', '10 Duff Street', '0%', 'Regular')
11 into Customer(CustomerID,Fullname,Address,DiscountRate,category) values('C10', 'Joe Ryan', '27 CorioStreet', '5%', 'Staff')
12 select * from dual;

10 rows created.

```

Figure 19 Screenshot of inserting data into customer table

```

SQL> select * from customer;

CUSTOMERID  FULLNAME      ADDRESS                DISCOUNTRA  CATEGORY
-----
C01         Eyan Owen     579 RainbowDrive      0%          Regular
C02         Johnson Nirvik 1932 Cherry Camp Road 0%          Regular
C03         Endrick Paul   2663 Stonky Lane      0%          Regular
C04         Johnson Linda  3641 Freed Drive      5%          Staff
C05         Reece James    39 Northgate street   5%          Staff
C06         Lauren James   11 Hudson St          10%         VIP
C07         Mason Mount    98 Consett Rd         10%         VIP
C08         Malung Sarr    34 Dossiter Street    5%          Staff
C09         Koku Harold    10 Duff Street        0%          Regular
C10         Joe Ryan       27 CorioStreet        5%          Staff

10 rows selected.

```

Figure 20 Screenshot of retrieving data from customer table

Insert into Payment Option table:

```

SQL> insert all
2  into PaymentOption(PaymentOptionID,OptionName) values('701', 'Credit/Debit card')
3  into PaymentOption(PaymentOptionID,OptionName) values('702', 'e-wallet')
4  into PaymentOption(PaymentOptionID,OptionName) values('703', 'Cash-on-Delivery')
5  select * from dual;

3 rows created.

```

Figure 21 Screenshot of inserting data into Payment Option table

```

SQL> select * from paymentoption;

PAYMENTOPTIONID  OPTIONNAME
-----
701              Credit/Debit card
702              e-wallet
703              Cash-on-Delivery

```

Figure 22 Screenshot of retrieving data from Payment Option table

Insert into Orders table:

```

SQL> insert all
2  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('01', to_date('2023-05-02', 'YYYY-MM-DD'), '1800', 'C04', '701')
3  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('02', to_date('2023-05-05', 'YYYY-MM-DD'), '350', 'C01', '703')
4  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('03', to_date('2023-05-15', 'YYYY-MM-DD'), '911.26', 'C10', '702')
5  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('04', to_date('2023-05-23', 'YYYY-MM-DD'), '1720.99', 'C02', '701')
6  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('05', to_date('2023-06-01', 'YYYY-MM-DD'), '1253.56', 'C03', '702')
7  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('06', to_date('2023-06-23', 'YYYY-MM-DD'), '3078.56', 'C06', '702')
8  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('07', to_date('2023-06-23', 'YYYY-MM-DD'), '1250.55', 'C06', '702')
9  Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('08', to_date('2023-07-06', 'YYYY-MM-DD'), '1500', 'C07', '701')
10 Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('09', to_date('2023-08-05', 'YYYY-MM-DD'), '450.22', 'C04', '701')
11 Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('010', to_date('2023-08-07', 'YYYY-MM-DD'), '345.65', 'C03', '703')
12 Into Orders(OrderID,OrderDate,TotalAmount,CustomerID,PaymentOptionID) values('011', to_date('2023-08-07', 'YYYY-MM-DD'), '2345.77', 'C01', '702')
13 select * from dual;

11 rows created.

```

Figure 23 Screenshot of inserting and retrieving data from Orders table

```

SQL> select * from Orders;

ORDERID    ORDERDATE    TOTALAMOUNT  CUSTOMERID  PAYMENTOPT
-----
01         02-MAY-23      1800.00     C04         701
02         05-MAY-23      350.00     C01         703
03         15-MAY-23      911.26     C10         702
04         23-MAY-23     1720.99     C02         701
05         01-JUN-23     1253.56     C03         702
06         23-JUN-23     3078.56     C06         702
07         23-JUN-23     1250.55     C06         702
08         06-JUL-23     1500.00     C07         701
09         05-AUG-23      450.22     C04         701
010        07-AUG-23      345.65     C03         703
011        07-AUG-23     2345.77     C01         702

11 rows selected.

```

Figure 24 Screenshot of retrieving data from Orders table

Insert into invoice table:

```

SQL> insert all
2  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I1', '01', '701', 'C04', '1800', '90.00', '1710.00', t
o_date('2023-05-03', 'YYYY-MM-DD'))
3  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I2', '02', '703', 'C01', '350', '0.00', '350.00', to_d
ate('2023-05-15', 'YYYY-MM-DD'))
4  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I3', '03', '702', 'C10', '911.26', '45.56', '865.697',
to_date('2023-05-23', 'YYYY-MM-DD'))
5  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I4', '04', '701', 'C02', '1720.99', '0.00', '1720.99',
to_date('2023-05-27', 'YYYY-MM-DD'))
6  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I5', '05', '702', 'C03', '1253.56', '0.00', '1253.56',
to_date('2023-06-01', 'YYYY-MM-DD'))
7  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I6', '06', '702', 'C06', '3078.56', '307.85', '2770.70',
to_date('2023-05-23', 'YYYY-MM-DD'))
8  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I7', '07', '702', 'C06', '1250.55', '125.05', '1125.49',
to_date('2023-06-14', 'YYYY-MM-DD'))
9  into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I8', '08', '701', 'C07', '1500.00', '150.00', '1350.00',
to_date('2023-06-30', 'YYYY-MM-DD'))
10 into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I9', '09', '701', 'C04', '450.22', '22.51', '427.70',
to_date('2023-07-05', 'YYYY-MM-DD'))
11 into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I10', '010', '703', 'C03', '345.65', '0.00', '345.65',
to_date('2023-05-02', 'YYYY-MM-DD'))
12 into Invoice(InvoiceID,OrderID,PaymentOptionID,CustomerID,TotalAmount,DiscountAmount,FinalAmount,InvoiceDate) values('I11', '011', '702', 'C01', '2345.77', '0.00', '2345.77',
to_date('2023-07-06', 'YYYY-MM-DD'))
13 select * from dual;

11 rows created.

```

Figure 25 Screenshot of inserting data into Invoice table

```
SQL> select * from invoice;
```

INVOICEID	ORDERID	PAYMENTOPTIONID	CUSTOMERID	TOTALAMOUNT	DISCOUNTAMOUNT	FINALAMOUNT	INVOICEDATE
I1	01	701	C04	1800	90	1710.00	03-MAY-23
I2	02	703	C01	350	0	350.00	15-MAY-23
I3	03	702	C10	911.26	45.56	865.70	23-MAY-23
I4	04	701	C02	1720.99	0	1720.99	27-MAY-23
I5	05	702	C03	1253.56	0	1253.56	01-JUN-23
I6	06	702	C06	3078.56	307.85	2770.70	23-MAY-23
I7	07	702	C06	1250.55	125.05	1125.49	14-JUN-23
I8	08	701	C07	1500	150	1350.00	30-JUN-23
I9	09	701	C04	450.22	22.51	427.70	05-JUL-23
I10	010	703	C03	345.65	0	345.65	02-MAY-23
I11	011	702	C01	2345.77	0	2345.77	06-JUL-23

11 rows selected.

Figure 26 Screenshot of retrieving data from Invoice table

Insert OrderProduct table:

```
SQL> insert all
2  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('02', 'P04', '1', '1120.00')
3  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('01', 'P09', '2', '786.00')
4  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('03', 'P02', '5', '350.99')
5  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('04', 'P07', '4', '1253.56')
6  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('05', 'P10', '2', '560.00')
7  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('05', 'P08', '2', '125.26')
8  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('06', 'P03', '1', '250.56')
9  into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('07', 'P05', '1', '1265.22')
10 into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('07', 'P01', '1', '1197.99')
11 into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('08', 'P06', '4', '350.00')
12 into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('09', 'P06', '5', '350.00')
13 into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('010', 'P01', '2', '1197.99')
14 into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('010', 'P03', '1', '250')
15 into OrderProduct(OrderID,ProductID,Quantity,UnitPrice) values('011', 'P09', '1', '786.00')
16 select * from dual;
```

14 rows created.

Figure 27 Screenshot of inserting data into OrderProduct table

```
SQL> select * from OrderProduct;
```

ORDERID	PRODUCTID	QUANTITY	UNITPRICE
02	P04	1	1120.00
01	P09	2	786.00
03	P02	5	350.99
04	P07	4	1253.56
05	P10	2	560.00
05	P08	2	125.26
06	P03	1	250.56
07	P05	1	1265.22
07	P01	1	1197.99
08	P06	4	350.00
09	P06	5	350.00
010	P01	2	1197.99
010	P03	1	250.00
011	P09	1	786.00

14 rows selected.

Figure 28 Screenshot of retrieving data from OrderProduct table

Instruction query

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

- select * from Customer where Category = 'Staff';

```
SQL> select * from Customer where category = 'Staff';
```

CUSTOMERID	FULLNAME	ADDRESS	DISCOUNTRATE	CATEGORY
C04	Johnson Linda	3641 Freed Drive	5%	Staff
C05	Reece James	39 Northgate street	5%	Staff
C08	Malung Sarr	34 Dossiter Street	5%	Staff
C10	Joe Ryan	27 Corio Street	5%	Staff

```
SQL> _
```

Figure 29 Screenshot of retrieving data from customer who are staff of a company

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

- select * from Orders where OrderDate >= to_date('2023-05-01', 'YYYY-MM-DD') and OrderDate <= to_date('2023-05-28', 'YYYY-MM-DD');

```
SQL> select * from Orders where orderdate >= to_date('2023-05-01', 'YYYY-MM-DD') and orderdate <= to_date('2023-05-28', 'YYYY-MM-DD');
```

ORDERID	ORDERDATE	TOTALAMOUNT	CUSTOMERID	PAYMENTOPTIONID
01	02-MAY-23	1800	C04	701
02	05-MAY-23	350	C01	703
03	15-MAY-23	911.26	C10	702
04	23-MAY-23	1720.99	C02	701

```
SQL>
```

Figure 30 Screenshot of retrieving orderdate from Orders between 01-05-2023 to 05-28-2023

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

- select Orders.OrderID, Orders.OrderDate, Orders.TotalAmount, Customer.CustomerID, Customer.FullName from Customer left join Orders on Customer.CustomerId = Orders.CustomerID;

```
SQL> select Orders.OrderID, Orders.OrderDate, Orders.TotalAmount, Customer.CustomerID, Customer.FullName from Customer left join Orders on Customer.CustomerID = Orders.CustomerID;
```

ORDERID	ORDERDATE	TOTALAMOUNT	CUSTOMERID	FULLNAME
01	02-MAY-23	1800	C04	Johnson Linda
02	05-MAY-23	350	C01	Eyan Owen
03	15-MAY-23	911.26	C10	Joe Ryan
04	23-MAY-23	1720.99	C02	Johnson Nirvik
05	01-JUN-23	1253.56	C03	Endrick Paul
06	23-JUN-23	3078.56	C06	Lauren James
07	23-JUN-23	1250.55	C06	Lauren James
08	06-JUL-23	1500	C07	Mason Mount
09	05-AUG-23	450.22	C04	Johnson Linda
010	07-AUG-23	345.65	C03	Endrick Paul
011	07-AUG-23	2345.77	C01	Eyan Owen
			C08	Malung Sarr
			C05	Reece James
			C09	Koju Harold

14 rows selected.

Figure 31 Screenshot of retrieving data from customer with order details

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

- select * from productavailability where substr(productname, 2, 1) = 'a' and stockquantity > 50;

```
SQL> select * from productavailability where substr(productname, 2, 1) = 'a' and stockquantity > 50;
```

PRODUCTID	PRODUCTNAME	STOCKQUANTITY	AVAILABILITYSTATUS
P01	Samsung Neo QLED	57	Instock
P02	Samsung S23	51	Instock

SQL> _

Figure 32 Screenshot of retrieving product details that have the second letter 'a'

5. Find out the customer who has ordered recently.

- select Customer.CustomerID, Customer.Fullname from Customer join Orders on Customer.CustomerID = Orders.CustomerID where Orders.OrderDate = (select max(OrderDate) from Orders);

```
SQL> select Customer.CustomerId, Customer.Fullname from Customer join Orders on Customer.CustomerID = Orders.CustomerID where Orders.OrderDate = (select max(OrderDate) from Orders);
```

CUSTOMERID	FULLNAME
C01	Eyan Owen
C03	Endrick Paul

```
SQL> █
```

Figure 33 Screenshot of retrieving data of customer that has ordered recently

Transaction query

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

- select to_char(OrderDate, 'YYYY-MM') as month, sum(TotalAmount) as TotalRevenue from Orders group by to_char(OrderDate, 'YYYY-MM') order by to_char(OrderDate, 'YYYY-MM');

```
SQL> select to_char(OrderDate, 'YYYY-MM') as month, sum(TotalAmount) as TotalRevenue from Orders group by to_char(OrderDate, 'YYYY-MM')
2 order by to_char(OrderDate, 'YYYY-MM');
```

MONTH	TOTALREVENUE
2023-05	4782.25
2023-06	5582.67
2023-07	1500
2023-08	3141.64

```
SQL>
```

Figure 34 Screenshot of calculating total revenue of each month

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

- select * from Orders where TotalAmount >= (select avg(TotalAmount) from Orders);

```
SQL> select * from Orders where TotalAmount >= (select avg(TotalAmount) from Orders);
```

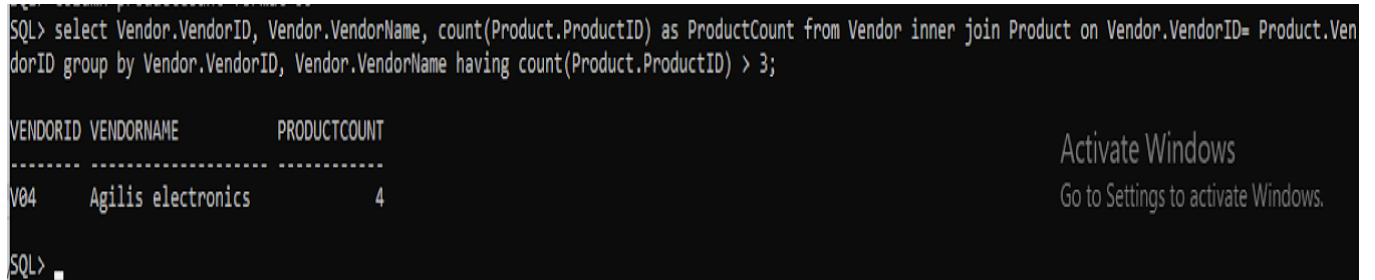
ORDERID	ORDERDATE	TOTALAMOUNT	CUSTOMERID	PAYMENTOPTIONID
01	02-MAY-23	1800	C04	701
04	23-MAY-23	1720.99	C02	701
06	23-JUN-23	3078.56	C06	702
08	06-JUL-23	1500	C07	701
011	07-AUG-23	2345.77	C01	702

```
SQL>
```

Figure 35 Screenshot of retrieving data from order which has equal or higher than the average order total value

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

- select Vendor.VendorID, Vendor.VendorName, count(Product.ProductID) as ProductCount from Vendor inner join Product on Vendor.VendorID = Product.VendorID group by Vendor.VendorID, Vendor.VendorName having count(Product.ProductID) > 3;



```
SQL> select Vendor.VendorID, Vendor.VendorName, count(Product.ProductID) as ProductCount from Vendor inner join Product on Vendor.VendorID= Product.VendorID group by Vendor.VendorID, Vendor.VendorName having count(Product.ProductID) > 3;
```

VENDORID	VENDORNAME	PRODUCTCOUNT
V04	Agilis electronics	4

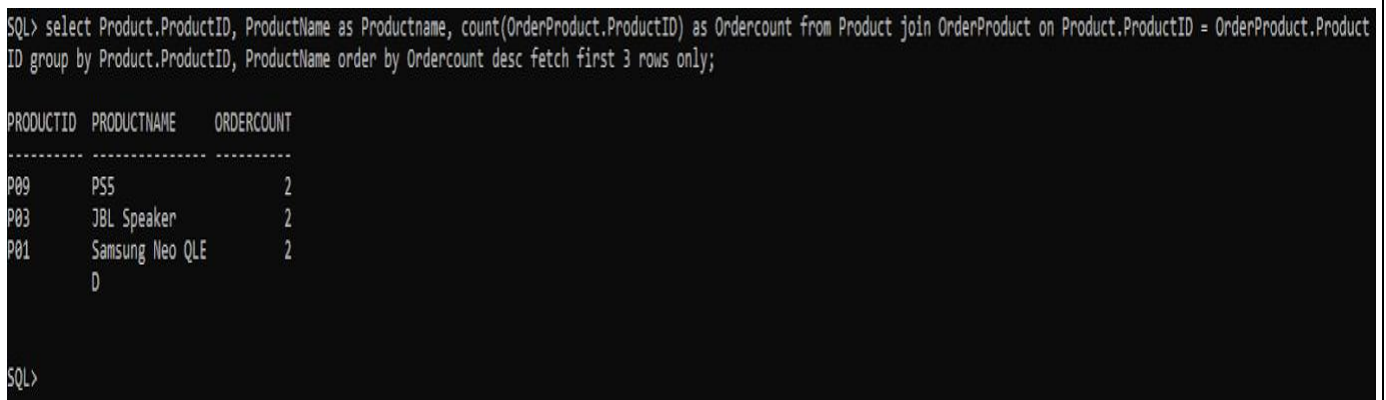
SQL> █

Activate Windows
Go to Settings to activate Windows.

Figure 36 Screenshot of retrieving data from vendor who has supplied more than 3 products

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

- select Product.ProductID, ProductName as Productname, count(OrderProduct.ProductID) as Ordercount from Product join OrderProduct on Product.ProductID = OrderProduct.ProductID group by Product.ProductID, Productname order by Ordercount desc fetch first 3 rows only;



```
SQL> select Product.ProductID, ProductName as Productname, count(OrderProduct.ProductID) as Ordercount from Product join OrderProduct on Product.ProductID = OrderProduct.ProductID group by Product.ProductID, ProductName order by Ordercount desc fetch first 3 rows only;
```

PRODUCTID	PRODUCTNAME	ORDERCOUNT
P09	PS5	2
P03	JBL Speaker	2
P01	Samsung Neo QLED	2

SQL>

Figure 37 Screenshot of showing the product details that have been ordered most

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

- select Customer.CustomerID, Fullname as Customername, sum(Product.Unitprice * OrderProduct.Quantity) as TotalSpending from Customer join Orders on Customer.CustomerID = Orders.CustomerID join OrderProduct on Orders.OrderID = OrderProduct.OrderID join PProduct on OrderProduct.ProductID = Product.ProductID where extract(month from Orders.OrderDate) = 8 group by Customer.CustomerID, Fullname order by TotalSpending desc fetch first 1 row only;

```
SQL> select Customer.CustomerID, Fullname as CustomerName, sum(Product.Unitprice * OrderProduct.Quantity) as TotalSpending from Customer join Orders on Customer.CustomerID = Orders.CustomerID join OrderProduct on Orders.OrderID = OrderProduct.OrderID join Product on OrderProduct.ProductID = Product.ProductID where extract(month from Orders.OrderDate) = 8 group by Customer.CustomerID, Fullname order by TotalSpending desc fetch first 1 row only;
```

CUSTOMERID	CUSTOMERNAME	TOTALSPENDING
C03	Endrick Paul	2646.54

SQL> .

Activate Windows
Go to Settings to activate Windows.

Figure 38 Screenshot of finding the customer who has ordered most in the August

Critical Evaluation

Database module is applicable to other modules like software engineering, AWS (Amazon Web Services), and networking, the database module in the course has grown to be an essential learning experience. Understanding how to incorporate databases into a software application is crucial, as illustrated by the database module's foundation. Software engineers acquired knowledge about how they ensure data consistency in concurrent systems involving database transactions through their research of databases. to ensure data consistency in concurrent systems with database transactions. The DFD Diagrams created for the software engineering curriculum using draw.io was also guided by Database module. We can make use of AWS services like Amazon RDS (Relational Database Service) and database security with AWS due to the information the database has given us. It has also given us ideas about encryption and access control.

The Database coursework is a complete balance between theoretical understanding and practical application as we are equipped with knowledge and skills necessary for the real-world scenario of database management. The coursework designed requires us to design and implement databases that address the need of specific business. The concept of normalization , ERD development strategies has proved to be an important application for a real-world Database System. This coursework also provides exposure to widely used systems such as MYSQL Plus, Oracle.

As a result, the coursework has proven to be very beneficial with high knowledge of advanced database principles for real-world application, which include basic forms of database normalization, query optimization, and handling of multiple issues.

Drop Query

```
SQL> drop table OrderProduct;
Table dropped.

SQL> drop table invoice;
Table dropped.

SQL> drop table Orders;
Table dropped.

SQL> drop table PaymentOption;
Table dropped.

SQL> drop table Customer;
Table dropped.

SQL> drop table ProductAvailability;
Table dropped.

SQL> drop table Product;
Table dropped.

SQL> drop table Vendor;
Table dropped.

SQL> drop table category;
Table dropped.

SQL>
```

Figure 39 Screenshot of dropping all the tables

Database Dump file Creation

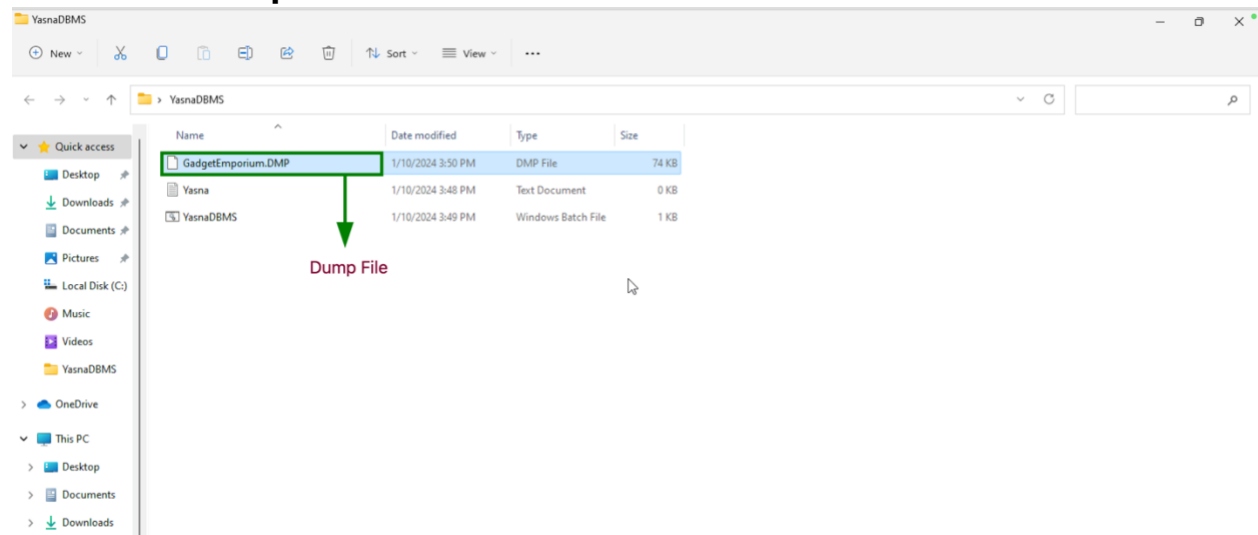


Figure 40 Screenshot of Dump file creation

References

- *geeksforgeeks*. (2023). Retrieved from <https://www.geeksforgeeks.org/first-normal-form-1nf/>
- *JavaTpoint*. (2023). Retrieved from [javatpoint.com/dbms-first-normal-form](https://www.javatpoint.com/dbms-first-normal-form)
- *byju's.com*. (2023). Retrieved from <https://byjus.com/gate/second-normal-form-in-dbms/#:~:text=A%20relation%20is%20said%20to,of%20a%20relation's%20candi date%20key>
- Chris, K. (2022, December 21). *freeCodeCamp*. Retrieved from <https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/>
- Watt, A. (n.d.). *BC Campus*. Retrieved from https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/?fbclid=IwAR3atnugNx117blibaV68HCky8q7yqvULsHChxVgoe-ogIbC_N0-fHAgbzE
- silberschatz, A. (n.d.). *Database system concepts*.
- Navathe, E. (n.d.). *Fundamentals of Database Systems*.
- *JavaTpoint*. (2011-2021). Retrieved from <https://www.javatpoint.com/dbms-normalization>
- (n.d.). Retrieved from DBS211: <http://dbs211.ca/courses/dbs211/Week10/index.html>