LONDON METROPOLITAN UNIVERSITY

CC5051NI Databases

50% Individual Coursework

2023-24 Autumn

Student Name: Rajita Maharjan

London Met ID: 22067335

College ID: np01ai4a220052

Assignment Due Date: Monday, January 15, 2024
Assignment Submission Date: Monday, January 15, 2024
Word Count: 2409

I confirm that I understand my coursework needs to be submitted online My Second teacher 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 marks of zero will be awarded.

Table of Contents

Table of Figures

1. Introduction

1.1. Introduction to Gadget Emporium and its business forte

Gadget Emporium is an online marketplace launched by Mr. John, an entrepreneur and electronics enthusiast. The online store focuses in providing a wide range of electronic products and accessories to both customers and business companies. Gadget Emporium's services include managing a huge inventory of items, receiving client orders, receiving payments, and maintaining the relationships with clients. The goal is to build an energetic internet marketplace that helps both individual customers and companies. The system is trying to provide a wide range of electrical devices as well as providing an easy buying experience. This provides a base for the future development of an entire database system which will support all of the e-commerce activity.
Gadget Emporium's commercial strength is its specialty and commitment to providing a wide collection of high-quality electrical gadgets and accessories.
Its forte can be described as follows:

- Gadget Emporium focuses at providing a large collection of high quality of electrical products and accessories.
- Gadget Emporium defines itself through its dedication to outstanding customer service and creativity ensuring customers have access to advanced electronic products.
- Gadget Emporium's abilities in efficient management of supply chains is shown by excellent supplier interactions and immediate stock control.

1.2. Business Rules:

Product Management: Each product is easily identified by a ProductID, and key attributes such as ProductName, Description, Category, Price, and StockLevel should be given while managing products for Gadget Emporium. Products are separated into categories, and each category can have multiple items. Also, each item gets a unique stock number for system verification.

Customer Categories and Discounts: Customers are important to the system with their CustomerID and providing information such as CustomerName, Address, and Category. Discount rates are determined by category. Customers are categorized as Regular (R), Staff (S), or VIP (V), each type has a different discount rate on product purchases. When a customer makes lots of purchases, each purchase is associated with only one customer, this is often referred to as a key relationship.

Order Processing: Orders are important for the Gadget Emporium, each one is properly accepted by an OrderID and bringing important data such as the OrderDate. Each order is associated with a single consumer, increasing the connection between a customer and their purchases. In addition, an employee may work on many purchases at the same time, but each purchase is given to a single employee.

Product Availability and Inventory Management: For Gadget Emporium, product availability and inventory control are essential factors. StockLevel provides real-time tracking to stop

overselling by displaying the current quantity of each product. In addition, a single purchase might include several products, but each item is only connected to a single purchase.

Vendor Management: Vendors, who are required to have a VendorName and are each uniquely identified by a VendorID, are crucial. Vendors provide products, developing a connection between each product and a specific vendor. Other than that, every supplier has a separate supplier ID, and even though a supplier can provide more than one item, only one supplier can provide each item.

Processing of Payments: Having options for paying is important for finishing orders, with choices like cash on delivery, credit/debit card, or e-wallet. To ensure that transactions are completed easily, each order must include a easy method of payment.

Order Detail and Invoice Generation: Order details, which include the product, quantity, and unit price, take all the details about every order. After a customer checks out, an invoice is generated that includes all of the customer, order, and payment information, as well as any necessary discounts.

1.3. Identification of attributes:

1. **Customer:**
    - CustomerID (Primary Key, Not Null, Unique)
    - CustomerName(Not Null)
    - Address(Not Null)
    - PhoneNumber(Not Null)
    - Category (Not Null, Check for ‘Regular’, ‘Staff’, ‘VIP’)

2. **Product:**
    - ProductID (Primary Key, Not Null, Unique)
    - ProductName (Not Null)
    - Description(Not Null)
    - CategoryID (Foreign Key, Not Null)
    - Price(Not Null, Check for positive values)
    - StockLevel(Not Null, Check for negative values)

3. **Category:**
    - CategoryID (Primary Key, Not Null, Unique)
    - CategoryName (Not Null)

4. **Order**:
    - OrderID (Primary Key, Not Null, Unique)
    - CustomerID (Foreign Key(Not Null)
    - OrderDate(Not Null, Default to current date)
    - PaymentMethod (Not Null, Check for 'Cash on Delivery', 'Credit/Debit Card', 'E-Wallet')

5. **OrderDetail**:
    - OrderDetailID (Primary Key, Not Null, Unique)
    - OrderID (Foreign Key, Not Null)
    - ProductID (Foreign Key, Not Null)
    - Quantity(Not Null, Check for positive values)
    - Price(Not Null, Check for positive values)

6. **Employee**:
    - EmployeeID (Primary Key, Not Null, Unique)
    - EmployeeName(Not Null)
    - Address(Not Null)
    - PhoneNumber(Not Null)

7. **Supplier**:
    - SupplierID (Primary Key, Not Null, Unique)
    - SupplierName(Not Null)
    - Address(Not Null)
    - PhoneNumber(Not Null)

8. **Invoice:**
    - InvoiceID (Primary Key, Not Null, Unique)
    - OrderID (Foreign Key, Not Null)
    - TotalAmount (Not Null, Check for non-negative values)
    - Discount (Not Null, Check for non-negative values)
    - FinalAmount (Not Null, Check for non-negative values)

2. Initial ERD

2.1. List of objects and identification and representation of foreign keys

1. **Product**:
    - ProductID (Primary Key, Not Null, Unique)
    - ProductName (Not Null)
    - Description (Not Null)
    - CategoryID (Foreign Key, Not Null)
    - Price (Not Null, Check for positive values)
    - StockLevel (Not Null, Check for non-negative values)

2. **Category**:
    - CategoryID (Primary Key, Not Null, Unique)
    - CategoryName (Not Null)

3. **Customer**:
    - CustomerID (Primary Key, Not Null, Unique)
    - CustomerName (Not Null)
    - Address (Not Null)
    - PhoneNumber (Not Null)
    - CustomerType (NotNull)

4. **Order**:
    - OrderID (Primary Key, Not Null, Unique)
    - CustomerID (Foreign Key, Not Null)
    - OrderDate (Not Null, Default to current date)

5. **OrderDetail**:
    - OrderDetailID (Primary Key, Not Null, Unique)
    - OrderID (Foreign Key, Not Null)
    - ProductID (Foreign Key, Not Null)
    - Quantity (Not Null, Check for positive values)
    - Price (Not Null, Check for positive values)

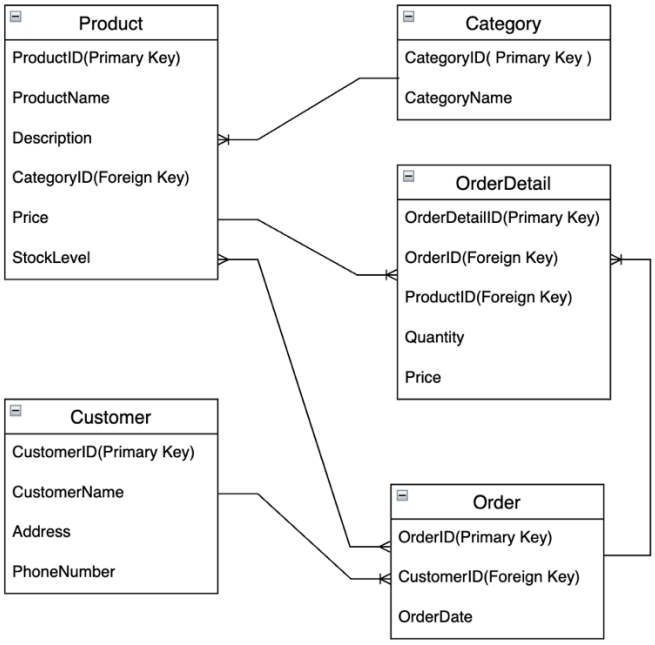

Figure 1 : Initial ERD

3. Normalization

1. Unnormalized Form (UNF):

GadgetEmporium (OrderID, CustomerID, CustomerName, CustomerType, PhoneNumber, Address, {ProductID, ProductName, Description, CategoryID, Price, StockLevel})

2. First Normal Form(1NF):

Order (OrderID, CustomerID)
Customer (CustomerID, CustomerName, CustomerType, Address, PhoneNumber)
OrderDetail (OrderID, {ProductID, Quantity, UnitPrice})
Product (ProductID, ProductName, Description, CategoryID, Price, StockLevel)

3. Second Normal Form(2NF):

Order (OrderID, CustomerID)
Customer (CustomerID, CustomerName, CustomerType, PhoneNumber, Address)
OrderDetail (OrderID, ProductID, Quantity, UnitPrice)
Product (ProductID, ProductName, Description, CategoryID, Price, StockLevel)

4. **Third Normal Form(3NF):**

- **Order** (OrderID, CustomerID)
  - OrderID ⟶ CustomerID

- **Customer** (CustomerID, CustomerName, CustomerType, DiscountRate, Address)
  - CustomerID ⟶ CustomerName, DiscountRate, Address

- **OrderDetail** (OrderID, ProductID, Quantity, UnitPrice)
  - OrderID, ProductID ⟶ Quantity, UnitPrice

- **Product** (ProductID, ProductName, Description, CategoryID, Price, StockLevel)
  - ProductID ⟶ ProductName, Description, CategoryID, Price, StockLevel

5. **Functional Dependencies:**

Order (OrderID, CustomerID)

Functional Dependency: OrderID ⟶ CustomerID

CustomerID is functionally dependent on OrderID. For each OrderID, there is exactly one CustomerID associated with it.

Customer (CustomerID, CustomerName, CustomerType, DiscountRate, Address)

Functional Dependency: CustomerID ⟶ CustomerName, DiscountRate, CustomerType, Address

CustomerName, CustomerType, DiscountRate, and Address are functionally dependent on CustomerID. For each CustomerID, there is exactly one set of CustomerName, DiscountRate, and Address associated with it.

OrderDetail (OrderID, ProductID, Quantity, UnitPrice)

Functional Dependency: OrderID, ProductID ⟶ Quantity, UnitPrice

Quantity and UnitPrice are functionally dependent on the combination of OrderID and ProductID. For each combination of OrderID and ProductID, there is exactly one set of Quantity and UnitPrice associated with it.

Product (ProductID, ProductName, Description, CategoryID, Price, StockLevel)

Functional Dependency: ProductID ⟶ ProductName, Description, CategoryID, Price, StockLevel

ProductName, Description, CategoryID, Price, and StockLevel are functionally dependent on ProductID. For each ProductID, there is exactly one set of ProductName, Description, CategoryID, Price, and StockLevel associated with it.

4. Final ERD

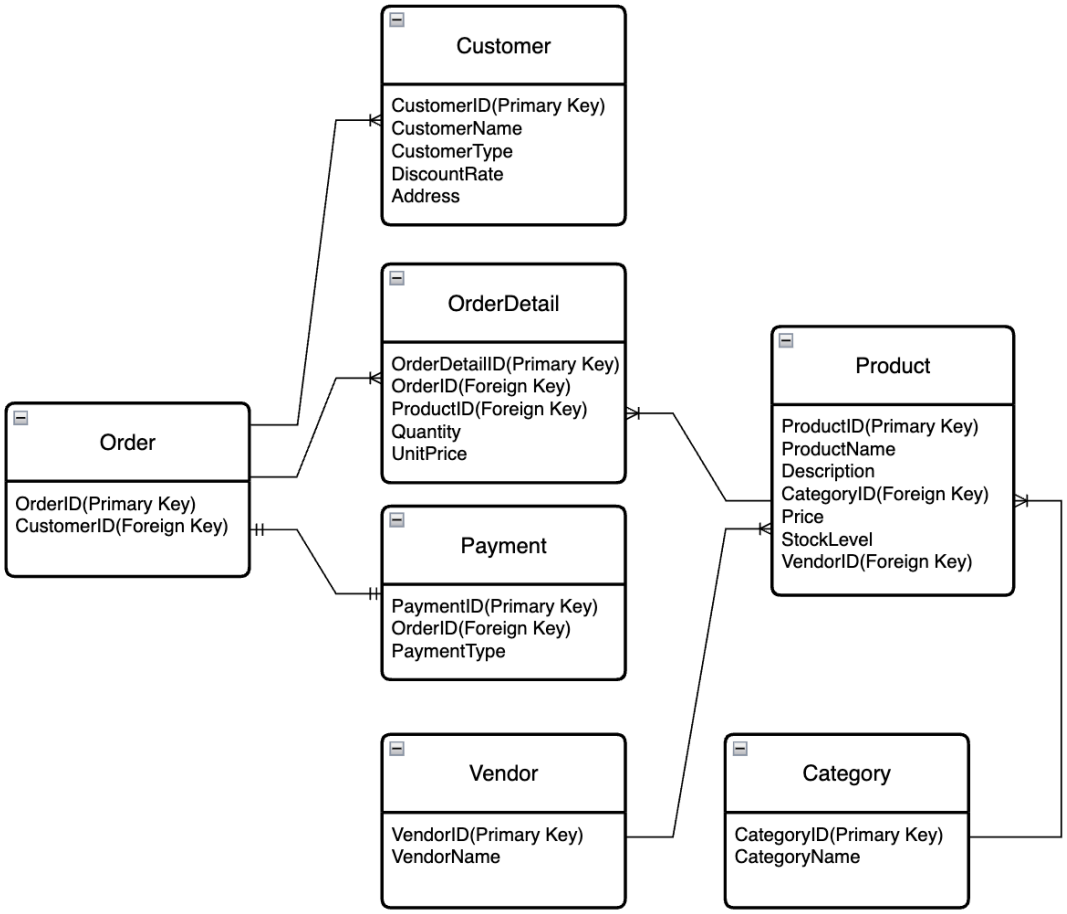

Figure 2: Final ERD

5. Implementation

5.1. Creating Tables

```
SQL*Plus: Release 11.2.0.2.0 Production on Mon Jan 15 03:02:49 2024

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

SQL> connect system/rajita;
Connected.
SQL> CONNECT RAJITA/rajita;
Connected.
SQL> CREATE TABLE Vendor(VendorID INT PRIMARY KEY, VendorName VARCHAR(255) NO
T NULL);

Table created.

SQL>
```

Figure 3: create table vendor

```
SQL> --CREATING TABLE CATEGORY
SQL> CREATE TABLE Category(CategoryID INT PRIMARY KEY,CategoryName VARCHAR(25
5) NOT NULL);

Table created.
```

Figure 4: create table category

```
SQL> --CREATING  TABLE customer
SQL> CREATE TABLE Customer(CustomerID INT PRIMARY KEY, CustomerName VARCHAR(2
55) NOT NULL, Adress VARCHAR(1000), Category CHAR(1) CHECK (Category IN('R',
'S', 'V')));

Table created.
```

Figure 5: create table customer

```
SQL> --CREATING TABLE  OrderTable
SQL> CREATE TABLE OrderTable(OrderID INT PRIMARY KEY, OrderDate DATE NOT NULL
, CustomerID INT NOT NULL, FOREIGN KEY (CustomerID) REFERENCES Customer(Custo
merID));

Table created.

SQL>
```

Figure 6: create table ordertable

```
SQL> --CREATING TABLE Orderdetails
SQL> CREATE TABLE OrderDetail (OrderDetailID INT PRIMARY KEY,OrderID INT NOT
NULL, ProductID INT NOT NULL, Quantity INT NOT NULL, UnitPrice DECIMAL(10, 2)
 NOT NULL, FOREIGN KEY (OrderID) REFERENCES OrderTable(OrderID), FOREIGN KEY
(ProductID) REFERENCES Product(ProductID));

Table created.
```

Figure 7: create table orderdetails

```
SQL> --CREATING TABLE Product
SQL> CREATE TABLE Product(ProductID INT PRIMARY KEY, ProductName VARCHAR(255)
 NOT NULL, Description VARCHAR(1000), CategoryID INT, Price DECIMAL (10, 2) N
OT NULL, StockLevel INT, VendorID INT, FOREIGN KEY (VendorID) REFERENCES Vend
or(VendorID), FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID));

Table created.

SQL>
```

Figure 8: create table product

```
SQL> --CREATING TABLE Invoice
SQL> CREATE TABLE Invoice (InvoiceID INT PRIMARY KEY , OrderID INT NOT NULL,
 CustomerID INT, InvoiceDate DATE NOT NULL, TotalAmount DECIMAL(10, 2) NOT NU
LL, PaymentOption VARCHAR(50), FOREIGN KEY (OrderID) REFERENCES OrderTable(Or
derID), FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID));

Table created.

SQL>
```

Figure 9 : create table invoice

```
SQL> --CREATE TABLE CustomerCategory
SQL> CREATE TABLE CustomerCategory (CategoryCode CHAR(1) PRIMARY KEY, Discoun
tRate DECIMAL(3, 2) NOT NULL);

Table created.
```

Figure 10: create table customerCategory

5.2. Inserting Values into the Tables

```
SQL> --Inserting Values into customer table
SQL>  INSERT INTO Customer(CustomerID, CustomerName, Adress, Category) VALUES (1, 'Rajita M
aharjan', 'Kathmandu', 'R');

1 row created.

SQL>  INSERT INTO Customer(CustomerID, CustomerName, Adress, Category) VALUES (2, 'Manank J
ha', 'Pokhara', 'S');

1 row created.

SQL>  INSERT INTO Customer(CustomerID, CustomerName, Adress, Category) VALUES (3, 'Arshiya
Khanal', 'Kathmandu', 'V');

1 row created.

SQL>  INSERT INTO Customer(CustomerID, CustomerName, Adress, Category) VALUES (4, 'Sarina S
hrestha', 'Bhaktapur', 'R');

1 row created.

SQL>  INSERT INTO Customer(CustomerID, CustomerName, Adress, Category) VALUES (5, 'Barsha B
husal', 'Butwal', 'S');

1 row created.

SQL>  INSERT INTO Customer(CustomerID, CustomerName, Adress, Category) VALUES (6, 'Rajib Ma
harjan', 'Kirtipur', 'V');

1 row created.

SQL>  INSERT INTO Customer(CustomerID, CustomerName, Adress, Category) VALUES (7, 'Suhana T
andukar', 'Kathmandu', 'R');

1 row created.
```

Figure 11: insert values into customer table

```
SQL>
SQL> --insert values into vendor table
SQL>  INSERT INTO Vendor(VendorID, VendorName) VALUES (1, 'Apple Inc.');

1 row created.

SQL>  INSERT INTO Vendor(VendorID, VendorName) VALUES (2, 'Samsung Electronics');

1 row created.

SQL>  INSERT INTO Vendor(VendorID, VendorName) VALUES (3, 'Sony Corporation');

1 row created.

SQL>  INSERT INTO Vendor(VendorID, VendorName) VALUES (4, 'Microsoft Corporation');

1 row created.

SQL>  INSERT INTO Vendor(VendorID, VendorName) VALUES (5, 'LG Electronics');

1 row created.

SQL>  INSERT INTO Vendor(VendorID, VendorName) VALUES (6, 'Dell Technologies');

1 row created.

SQL>  INSERT INTO Vendor(VendorID, VendorName) VALUES (7, 'Hew-Packard Company');

1 row created.
```

Figure 12: insert values into vendor table

```
SQL>  --inserting values into category table
SQL>  INSERT INTO Category(CategoryID, CategoryName) VALUES (1, 'Laptops');

1 row created.

SQL>  INSERT INTO Category(CategoryID, CategoryName) VALUES (2, 'Smartphones');

1 row created.

SQL>  INSERT INTO Category(CategoryID, CategoryName) VALUES (3, 'Tablets');

1 row created.

SQL>  INSERT INTO Category(CategoryID, CategoryName) VALUES (4, 'Monitors');

1 row created.

SQL>  INSERT INTO Category(CategoryID, CategoryName) VALUES (5, 'Earbuds');

1 row created.

SQL>  INSERT INTO Category(CategoryID, CategoryName) VALUES (6, 'Accessories');

1 row created.

SQL>  INSERT INTO Category(CategoryID, CategoryName) VALUES (7, 'Other Electronics');

1 row created.

SQL>
```

Figure 13: insert values into category table

```
SQL>  --inserting values into CustomerCategory table
SQL>  INSERT INTO CustomerCategory(CategoryCode, DiscountRate) VALUES ('R', 0.05);

1 row created.

SQL>  UPDATE CustomerCategory SET DiscountRate = 0 WHERE CategoryCode = 'R';

1 row updated.

SQL>  INSERT INTO CustomerCategory(CategoryCode, DiscountRate) VALUES ('S', 0.05);

1 row created.

SQL>  INSERT INTO CustomerCategory(CategoryCode, DiscountRate) VALUES ('V', 0.10);

1 row created.

SQL>
```

Figure 14: insert values into customer Category table

```
SQL> --Inserting values into Product table
SQL>  INSERT INTO Product(ProductID, ProductName, Description, CategoryID, Price, StockLeve
l, VendorID) VALUES (101, 'MacBook Air', 'Thin and light laptop', 1, 1299.99, 15, 1);

1 row created.

SQL>  INSERT INTO Product(ProductID, ProductName, Description, CategoryID, Price, StockLeve
l, VendorID) VALUES (102, 'Galaxy S21', 'Flagship smartphone', 2, 899.99, 20, 2);

1 row created.

SQL>  INSERT INTO Product(ProductID, ProductName, Description, CategoryID, Price, StockLeve
l, VendorID) VALUES (103, 'Sony Xperia Tablet', 'Premium Android tablet', 3, 699.99, 15, 3)
;

1 row created.

SQL>  INSERT INTO Product(ProductID, ProductName, Description, CategoryID, Price, StockLeve
l, VendorID) VALUES (104, 'Microsoft Surface Pro 7', 'Versatile 2-in-1 laptop', 1, 1199.99,
 10, 4);

1 row created.

SQL>  INSERT INTO Product(ProductID, ProductName, Description, CategoryID, Price, StockLeve
l, VendorID) VALUES (105, 'LG Tone Free Earbuds', 'Wireless earbuds with UVnano', 5, 149.99
, 30, 5);

1 row created.

SQL>  INSERT INTO Product(ProductID, ProductName, Description, CategoryID, Price, StockLeve
l, VendorID) VALUES (106, 'Dell UltraSharp Monitor', '27-inch 4K HDR display', 4, 699.99, 8
, 6);

1 row created.

SQL>  INSERT INTO Product(ProductID, ProductName, Description, CategoryID, Price, StockLeve
l, VendorID) VALUES (107, 'HP Spectre x360', 'Premium convertible laptop', 1, 1399.99, 12,
7);

1 row created.

SQL>
```

Figure 15: insert values into product table

```
SQL>  --Inserting values into OrderTable
SQL>  INSERT INTO OrderTable(OrderID, OrderDate, CustomerID) VALUES (1, TO_DATE('15-01-2023
', 'DD-MM-YYYY'), 1);

1 row created.

SQL>  INSERT INTO OrderTable(OrderID, OrderDate, CustomerID) VALUES (2, TO_DATE('20-02-2023
', 'DD-MM-YYYY'), 2);

1 row created.

SQL>  INSERT INTO OrderTable(OrderID, OrderDate, CustomerID) VALUES (3, TO_DATE('25-03-2023
', 'DD-MM-YYYY'), 3);

1 row created.

SQL>  INSERT INTO OrderTable(OrderID, OrderDate, CustomerID) VALUES (4, TO_DATE('05-04-2023
', 'DD-MM-YYYY'), 4);

1 row created.

SQL>  INSERT INTO OrderTable(OrderID, OrderDate, CustomerID) VALUES (5, TO_DATE('10-05-2023
', 'DD-MM-YYYY'), 5);

1 row created.

SQL>  INSERT INTO OrderTable(OrderID, OrderDate, CustomerID) VALUES (6, TO_DATE('17-06-2023
', 'DD-MM-YYYY'), 6);

1 row created.

SQL>  INSERT INTO OrderTable(OrderID, OrderDate, CustomerID) VALUES (7, TO_DATE('14-07-2023
', 'DD-MM-YYYY'), 7);

1 row created.
```

Figure 16: insert values into order table

```
SQL> --Inserting values into Orderdetail table
SQL>  INSERT INTO OrderDetail(OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUE
S (201, 1, 101, 1, 1299.99);

1 row created.

SQL>  INSERT INTO OrderDetail(OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUE
S (202, 2, 102, 2, 899.99);

1 row created.

SQL>  INSERT INTO OrderDetail(OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUE
S (203, 3, 103, 3, 699.99);

1 row created.

SQL>  INSERT INTO OrderDetail(OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUE
S (204, 4, 104, 1, 1199.99);

1 row created.

SQL>  INSERT INTO OrderDetail(OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUE
S (205, 5, 105, 2, 149.99);

1 row created.

SQL>  INSERT INTO OrderDetail(OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUE
S (206, 6, 106, 3, 699.99);

1 row created.

SQL>  INSERT INTO OrderDetail(OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUE
S (207, 7, 107, 1, 1399.99);

1 row created.

SQL>
```

Figure 17: insert values into orderdetail table

```
SQL>  --Inserting values into Invoice table
SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (301, 1, 1, TO_DATE('18-01-2023', 'DD-MM-YYYY'), 1299.99, 'Credit Card');

1 row created.

SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (302, 2, 2, TO_DATE('23-02-2023', 'DD-MM-YYYY'), 1799.98, 'Cash On Delivery');

1 row created.

SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (303, 3, 3, TO_DATE('28-03-2023', 'DD-MM-YYYY'), 2099.97, 'E-Wallet');

1 row created.

SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (304, 4, 4, TO_DATE('08-04-2023', 'DD-MM-YYYY'), 1199.99, 'Credit Card');

1 row created.

SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (305, 5, 5, TO_DATE('02-05-2023', 'DD-MM-YYYY'), 299.98, 'Cash On Delivery');

1 row created.

SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (306, 6, 6, TO_DATE('20-06-2023', 'DD-MM-YYYY'), 2099.97, 'E-Wallet');

1 row created.

SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (307, 7, 7, TO_DATE(), 1399.99, 'Cash On Delivery');
 INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUES (30
7, 7, 7, TO_DATE(), 1399.99, 'Cash On Delivery')

                     *
ERROR at line 1:
ORA-00938: not enough arguments for function


SQL>  INSERT INTO Invoice(InvoiceID, OrderID, CustomerID, InvoiceDate, TotalAmount, PaymentOption) VALUE
S (307, 7, 7, TO_DATE('23-05-2023', 'DD-MM-YYYY'), 1399.99, 'Cash On Delivery');

1 row created.

SQL>
```

Figure 18: insert value into invoice table

```
SQL>  --updating the invoice table with the discounted price
SQL>  UPDATE Invoice SET TotalAmount = TotalAmount - (TotalAmount * 0.05 ) WHERE InvoiceID IN (302, 305)
;

2 rows updated.

SQL>  UPDATE Invoice SET TotalAmount = TotalAmount - (TotalAmount * 0.10 ) WHERE InvoiceID IN (303, 306)
;

2 rows updated.

SQL>
```

Figure 19: updated invoice table value with discounted price

6. Database Querying

6.1. Information Query

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

```
SQL> SPOOL C:\Users\rajitamaharjan\Downloads\ INFORMATION_QUERY1.txt
SP2-0768: Illegal SPOOL command
Usage: SPOOL { <file> | OFF | OUT }
where <file> is file_name[.ext] [CRE[ATE]|REP[LACE]|APP[END]]
SQL> SPOOL C:\Users\rajitamaharjan\Downloads\INFORMATION_QUERY1.txt
SQL> SELECT CustomerID, CustomerName, Adress FROM Customer WHERE Category = 'S';

CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
         2
Manank Jha
Pokhara

         5
Barsha Bhusal
Butwal

CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
```

Figure 20: List of all customers that are staff of the company

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

```
SQL> SPOOL OFF
SQL> SPOOL C:\Users\rajitamaharjan\Downloads\INFORMATION_QUERY2.txt
SQL> SELECT OrderTable.OrderID, OrderTable.OrderDate, Customer.CustomerName, Product.ProductName, OrderD
etail.Quantity, OrderDetail.UnitPrice FROM OrderTable JOIN OrderDetail ON OrderTable.OrderID = OrderDeta
il.OrderID JOIN Product ON OrderDetail.ProductID = Product.ProductID JOIN Customer ON OrderTable.Custome
rID = Customer.CustomerID WHERE Product.ProductID = 101   AND OrderTable.OrderDate BETWEEN TO_DATE('01-0
5-2023', 'DD-MM-YYYY') AND TO_DATE('28-05-2023', 'DD-MM-YYYY');

no rows selected

SQL> SPOOL OFF
```

Figure 21: orders made for any particular product between the dates 01-05-2023 till 28-05-2023.

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

```
SQL> SELECT C.CustomerID, C.CustomerName, C.Adress, OT.OrderID, OT.OrderDate,
P.ProductName, OD.Quantity, OD.UnitPrice
  2  FROM Customer C
  3  LEFT JOIN OrderTable OT ON C.CustomerID = OT.CustomerID
  4  LEFT JOIN OrderDetail OD ON OT.OrderID = OD.OrderID
  5  LEFT JOIN Product P ON OD.ProductID = P.ProductID;

         1
Rajita Maharjan
Kathmandu
         1 15-JAN-23
MacBook Air
         1    1299.99

         2
Manank Jha
Pokhara
         2 20-FEB-23
Galaxy S21
         2     899.99

         3
Arshiya Khanal
Kathmandu
         3 25-MAR-23
Sony Xperia Tablet
         3     699.99

         4
Sarina Shrestha
Bhaktapur
         4 05-APR-23
Microsoft Surface Pro 7
         1    1199.99

         5
Barsha Bhusal
Butwal
         5 10-MAY-23
LG Tone Free Earbuds
         2     149.99

         6
Rajib Maharjan
Kirtipur
         6 17-JUN-23
Dell UltraSharp Monitor
         3     699.99

         7
Suhana Tandukar
Kathmandu
         7 14-JUL-23
HP Spectre x360
         1    1399.99


7 rows selected.
```

Figure 22: customers with their order details and customers who haven't ordered any products

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

```
SQL> SPOOL OFF
SQL> SPOOL C:\Users\rajitamaharjan\Downloads\INFORMATION_QUERY4.txt
SQL> SELECT * FROM Product WHERE ProductName LIKE '_a%' AND StockQuantity > 50;
SELECT * FROM Product WHERE ProductName LIKE '_a%' AND StockQuantity > 50
                                                       *
ERROR at line 1:
ORA-00904: "STOCKQUANTITY": invalid identifier


SQL> SELECT * FROM Product WHERE ProductName LIKE '_a%' AND Quantity > 50;
SELECT * FROM Product WHERE ProductName LIKE '_a%' AND Quantity > 50
                                                       *
ERROR at line 1:
ORA-00904: "QUANTITY": invalid identifier


SQL> SELECT * FROM Product WHERE ProductName LIKE '_a%' AND StockLevel > 50;

no rows selected

SQL> SPOOL OFF
SQL>
```

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

- Find out the customer who has ordered recently.

```
SQL> SPOOL OFF
SQL> SPOOL C:\Users\rajitamaharjan\Downloads\INFORMATION_QUERY5.txt
SQL> SELECT Customer.CustomerID, Customer.CustomerName, Customer.Adress, MAX(OrderTable.
OrderDate) AS LatestOrderDate FROM Customer LEFT JOIN OrderTable ON Customer.CustomerID
= OrderTable.CustomerID GROUP BY Customer.CustomerID, Customer.CustomerName, Customer.Ad
ress ORDER BY LatestOrderDate DESC;

CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
LATESTORD
---------
         7
Suhana Tandukar
Kathmandu
14-JUL-23


CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
LATESTORD
---------
         6
Rajib Maharjan
Kirtipur
17-JUN-23


CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
LATESTORD
---------
         5
Barsha Bhusal
Butwal
```

Figure 24: customer who has ordered recently

```
         5
Barsha Bhusal
Butwal
10-MAY-23


CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
LATESTORD
---------
         4
Sarina Shrestha
Bhaktapur
05-APR-23


CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
LATESTORD
---------
         3
Arshiya Khanal
Kathmandu
25-MAR-23


CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
LATESTORD
---------
         2
Manank Jha
Pokhara
20-FEB-23


CUSTOMERID
----------
```

Figure 25: customer who has ordered recently 2

```
CUSTOMERID
----------
CUSTOMERNAME
--------------------------------------------------------------------------------
ADRESS
--------------------------------------------------------------------------------
LATESTORD
---------
         1
Rajita Maharjan
Kathmandu
15-JAN-23


7 rows selected.

SQL>
```

Figure 26

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

```
SQL> SELECT TRUNC(OrderTable.OrderDate, 'MM') AS MonthYear, SUM(OrderDetail.QU
ANTITY * OrderDetail.UNITPRICE) AS TotalRevenue FROM OrderTable JOIN OrderDeta
il ON OrderTable.OrderID = OrderDetail.OrderID GROUP BY TRUNC(OrderTable.Order
Date, 'MM') ORDER BY TRUNC(OrderTable.OrderDate, 'MM');
01-JAN-23      1299.99
01-FEB-23      1799.98
01-MAR-23      2099.97
01-APR-23      1199.99
01-MAY-23       299.98
01-JUN-23      2099.97
01-JUL-23      1399.99
SQL> SPOOL OFF
SQL>
```

Figure 27: Total revenue of the company for each month

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

```
SQL> SPOOL C:\Users\rajitamaharjan\Downloads\TRANSACTION_QUERY2.txt
SQL> SELECT * FROM OrderDetail WHERE UNITPRICE >= (SELECT AVG(UNITPRICE) FROM
OrderDetail);
         201          1        101 ##########    1299.99
         204          4        104 ##########    1199.99
         207          7        107 ##########    1399.99
SQL> SPOOL OFF
SQL>
```

Figure 28: order that are equal or higher than the average order total revenue

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

```
SQL> SPOOL C:\Users\rajitamaharjan\Downloads\TRANSACTION_QUERY4.txt
SQL> SELECT * FROM (SELECT p.ProductID, p.ProductName, COUNT(od.OrderID) AS Order_C
ount FROM Product p JOIN OrderDetail od ON p.ProductID = od.ProductID GROUP BY p.Pr
oductID, p.ProductName ORDER BY Order_Count DESC) WHERE ROWNUM <= 3;
       102
Galaxy S21
          1

       107
HP Spectre x360
          1

       104
Microsoft Surface Pro 7
          1

SQL> spool off
SQL>
```


Figure 29: 3 products that have been ordered the most

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

```
SQL> SELECT * FROM (SELECT c.CustomerID, c.CustomerName, COUNT(o.OrderID) AS Orders
Count, SUM(i.TotalAmount) AS TotalSpending FROM Customer c JOIN OrderTable o ON c.C
ustomerID = o.CustomerID JOIN Invoice i ON o.OrderID = i.OrderID WHERE TO_CHAR(o.Or
derDate, 'MM') = '08' GROUP BY c.CustomerID, c.CustomerName ORDER BY OrdersCount DE
SC) WHERE ROWNUM <= 1;
SQL> spool off
SQL>
```


Figure 30: customer who has ordered the most in august

7. Critical Evaluation

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

The database design module was useful in developing practical skills in database management, SQL querying, and normalization procedures. The direct approach of the coursework resulted in a more in-depth understanding of the subject. The real-world application of developing a database for Gadget Emporium enabled that we adjust easily from concepts of theory to practical implementation. The module's content, which included ERD construction, normalization, and SQL querying, showed how useful it was when developing strong databases for a wide range of business requirements.

It was evident how this module related to other topics, especially those that focused on software development and system design. It is essential to comprehend how the database fits into the larger scheme of software systems, and the course of study provided provides helpful details on this. The coursework's dedication to implementation prepared me with abilities useful to many IT disciplines, building an integrated appreciation of information systems.

b. Critical Assessment of coursework

The process of learning was effectively developed by the coursework structure, moving from conceptual basis to practical use. A systematic approach was made possible by breaking down into smaller tasks including the development of ERDs, normalization, and SQL querying. It is especially interesting that the "Gadget Emporium" case study, which represents a real-world scenario, was included. It gave the coursework a more relevant and practical element that helped us comprehend how databases are used in actual applications. This improves remembering and learning while preparing students for tasks that are like what we may encounter in our careers.

All the essential elements of database design and implementation are covered in this complete, well-structured coursework. It gave me a great combination of theoretical understanding and real-world application, enabling them to use what they've learned in practical situations. Overall, the coursework is challenging but beneficial, and it provides students with the skills necessary for achievement in any industry where dealing with databases is a need.

8. References

Business Rules and How They Affect Database Design. (1989 – 2023). From Soutron Global Inc: https://www.soutron.com/blog/general/business-rules-database-design/#:~:text=In%20basic%20terms%2C%20a%20database,a%20State%20of%20Residence%20field.

What Are Business Rules? . (2024). From O'Reilly Media: https://www.oreilly.com/library/view/database-design-for/0201752840/ch11s02.html#:~:text=A%20business%20rule%20is%20a,characteristics%20of%20a%20given%20relationship.