UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING, UTMJB

SEMESTER 1, SESSION 2023/2024

PROJECT: PHASE 3

SECD2523: DATABASE

SECTION 10

LECTURER'S NAME:

PN. ROZILAWATI BINTI DOLLAH @ MD ZAIN

GROUP NAME:

DATADUDES

GROUP MEMBERS:

1. MEOR AFIQ BIN MEOR SHAMSUL BAHRI B23CS0043
2. MUHAMMAD AINUL WAFIQ BIN MOHD FAIZ A23CS5046
3. NURAINN SOFEA BINTI NIZAMIL FAIRUZ A23EC5050
4. IZNURIN FATIHAH BINTI MD FAIZAL B23CS0041

1. Introduction

Our online shopping system's database is logically designed as a counterpart to popular platforms like Shopee, ensuring an effective and well-organized structure for managing system data. It entails specifying the entities (products, orders, customers), their properties (names, ID), and the relationships that exist between them. The design seeks to improve data integrity and get rid of duplication through normalisation. To guarantee accurate and safe data management, additional procedures such as indexing, integrity limitations, and security controls are put in place. This logical design contributes to the overall functionality and performance of the online shopping system by acting as the basis for the physical implementation of the database later on.

2. Overview of project

An online shopping system typically involves the development of a comprehensive database to support various functionalities of the platform. Here's an overview of the project that might be part of this project for an online shopping system. In this phase, we will describe about database conceptual design, DB logical design , relational DB schema(after normalization) and SQL statement. In database conceptual design we will updated business rule and make conceptual ERD. For the DB logical design, we will create logical ERD, update data dictionary and normalization. After make some normalization we will have relational DB schema. Lastly, in this phase we will create SQL statement.This overview of the project provides a general idea of the essential components involved in managing the database for such a online shopping system.

3. Database conceptual design

3.1 Updated business rule

- An admin can have multiple roles such as super admin, product manager, but each role is associated with only one admin.
- An admin can manage multiple sellers, but each seller is managed by only one admin.
- An admin can manage multiple customers, but each customer is managed by only one admin.
- A seller can manage multiple products, but each product is managed by only one seller.
- A seller can process multiple orders, but each order is processed by only one seller.
- Each customer must have a unique account, and each account is associated with only one customer.
- A customer can place multiple orders, but each order is placed by only one customer.
- Each customer manages only their own account.
- Multiple products can be included in multiple orders, and each order can contain multiple products.
- An order can consist of various products, and each product can be included in multiple orders.
- Each order is associated with one and only one payment, while each payment corresponds to one and only one order.
- Each order is associated with one and only one receipt, while each receipt corresponds to one and only one order.
- A customer can have multiple receipts, while each receipt is associated with only one customer.

3.2 Conceptual ERD

The ERD for our online shopping system provides a visual representation of the essential components that make up our e-commerce platform. In this diagram, we can see the relationships and interactions among various entities, such as customers, admins, sellers, products, orders, payments, categories, carts, and reviews. These entities work collaboratively to ensure the functionality and success of our online shopping system.

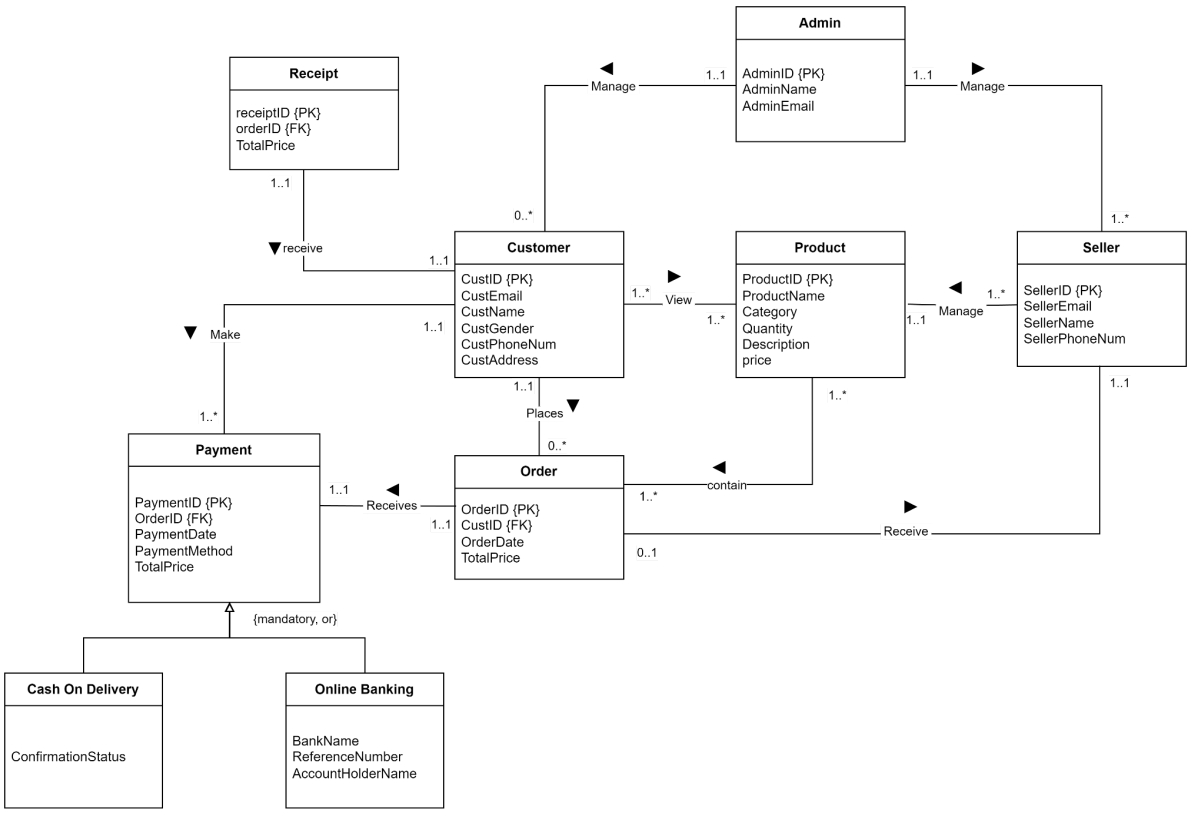


Figure 1.1: Conceptual ERD

4. DB logical design

4.1. Logical ERD

Figure below show the relationship between "Online Banking" and "Cash on Delivery" as subclasses and their superclass "Payment" is characterized as a "many-to-one" relationship. This means that multiple instances of the specialized subclasses ("Online Banking" and "Cash on Delivery") can be associated with a single instance of the superclass ("Payment"). The "many-to-one" relationship signifies that several occurrences of online banking transactions or cash-on-delivery payments can be linked to a common overarching payment record.

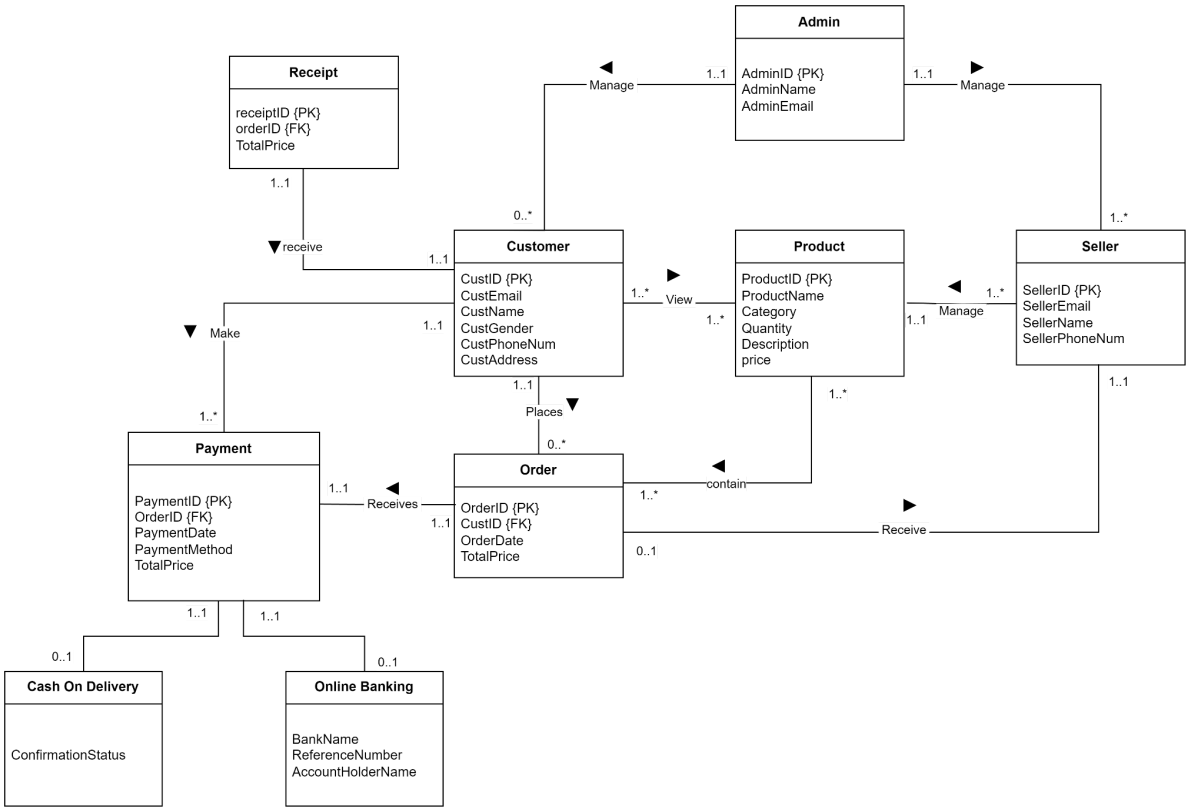


Figure 1.2: Logical ERD

1) Strong entity

- ADMIN (**adminID**, adminName, adminEmail)
- SELLER (**sellerID**, sellerEmail, sellerName, sellerPhoneNum)
- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)
- PRODUCT (**productID**, productName, category, quantity, decription, price)
- ORDER (**OrderID**, custD, orderDate, totalPrice)
- PAYMENT (**paymentID**, orderID, paymentDate, paymentMethod, totalPrice)
- RECEIPT (**receiptID**, orderID, totalPrice)

2) Many-to-many (1:*) binary relationship

I. Relationship places between customer and order.

Parent → customer

Child → order

- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)
- ORDER (**OrderID**, cusrID, orderDate, totalPrice)

II. Relationship makes between customer and payment.

Parent → customer

Child → payment

- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)
- PAYMENT (**paymentID**, orderID, paymentDate, paymentMethod, totalPrice)

III. Relationship manages between admin and customer.

Parent → admin

Child → customer

- ADMIN (**adminID**, adminName, adminEmail)
- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)

IV. Relationship manages between admin and seller.

Parent → admin

Child → seller

- ADMIN (**adminID**, adminName, adminEmail)
- SELLER (**sellerID**, sellerEmail, sellerName, sellerPhoneNum)

V. Relationship manages between seller and product.

Parent → seller

Child → product

- SELLER (**sellerID**, sellerEmail, sellerName, sellerPhoneNum)
- PRODUCT (**productID**, productName, category, quantity, decription, price)

3) One-to-one (1:1) binary relationship

Relationship manages between admin and seller.

- ADMIN (**adminID**, adminName, adminEmail)
- SELLER (**sellerID**, sellerEmail, sellerName, sellerPhoneNum)

Relationship manages between admin and customer.

- ADMIN (**adminID**, adminName, adminEmail)
- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)

Relationship receives between seller and order.

- SELLER (**sellerID**, sellerEmail, sellerName, sellerPhoneNum)
- ORDER (**OrderID**, cusrID, orderDate, totalPrice)

Relationship receives between order and payment.

- ORDER (**OrderID**, custID, orderDate, totalPrice)
- PAYMENT (**paymentID**, orderID, paymentDate, paymentMethod, totalPrice)

Relationship makes between customer and payment.

- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)
- PAYMENT (**paymentID**, orderID, paymentDate, paymentMethod, totalPrice)

Relationship between receives between customer and receipt.

- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)
- RECEIPT (**receiptID**, orderID, totalPrice)

4) Many-to-many (*:*) binary relationship

Relationship contains between orders and product.

- PRODUCT (**productID**, productName, category, quantity, decription, price)
- ORDER (**OrderID**, custID, orderDate, totalPrice)
- OrderProduct(OrderID, productID)

Relationship view between customer and product.

- CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)
- PRODUCT (productID, productName, category, quantity, decription, price)
- ViewProduct (custID, productID)

5) Superclass/ subclass relationship
    - CASHONDELIVERY (**PaymentID**, confirmationStatus, orderID, PaymentDate, totalPrice)
    - ONLINEBANKING (**PaymentID**, BankName, ReferenceNumber, AccountHolderName, orderID, PaymentDate, totalPrice)

Finalize

1. ADMIN (**adminID**, adminName, adminEmail)
2. SELLER **(sellerID**, sellerEmail, sellerName, sellerPhoneNum)
3. CUSTOMER (**custID**, custEmail, custName custGender, custPhoneNum, custAddress)
4. PRODUCT (**productID**, productName, category, quantity, decription, price)
5. ORDER (**OrderID,** custD, orderDate, totalPrice)
6. PAYMENT (**paymentID,** orderID, paymentDate, paymentMethod, totalPrice)
7. CASHONDELIVERY (**PaymentID**, confirmationStatus, orderID, PaymentDate, totalPrice)
8. ONLINEBANKING (**PaymentID,** BankName, ReferenceNumber, AccountHolderName, orderID, PaymentDate, totalPrice)
9. RECEIPT (**receiptID**, orderID, totalPrice)

4.2. Updated data dictionary

Entity name	Attributes	Data type	Nulls	Multi-valued	Description
Admin	AdminID	VARCHAR2 (5)	NO	NO	Unique identifier for administrator
	AdminName	VARCHAR2 (30)	NO	NO	Name of the Admin
	Email	VARCHAR2 (30)	NO	NO	Admin email address
Seller	SellerID	VARCHAR2 (5)	NO	NO	Unique identifier for seller
	SellerName	VARCHAR2 (30)	NO	NO	Name of the seller
	SellerEmail	VARCHAR2 (30)	NO	NO	Seller email address
	SellerPhoneNum	NUMBER(2 0)	NO	NO	Seller phone number
Customer	CustID	VARCHAR2 (5)	NO	NO	Unique identifier for customer
	CustName	VARCHAR2 (30)	NO	NO	Name of the customer
	CustEmail	VARCHAR2 (30)	NO	NO	Customer email address
	CustGender	VARCHAR2 (10)	NO	NO	Gender of customer
	CustPhoneNum	VARCHAR2 (30)	NO	NO	Customer phone number
	CustAddress	VARCHAR2 (50)	NO	NO	Customer Address

Product	ProductID	VARCHAR2 (5)	NO	NO	Unique identifier for product
	ProductName	VARCHAR2 (30)	NO	NO	Name of the product
	Quantity	NUMBER (5)	NO	NO	Quantity of the product
	Category	VARCHAR2 (30)	NO	NO	Category of product
	Description	VARCHAR2 (50)	NO	NO	Description of the product
	Prices	NUMBER(5, 2)	NO	NO	Price of the product
Order	OrderID	VARCHAR2 (5)	NO	NO	Unique identifier for order
	CustID	VARCHAR2 (5)	NO	NO	Unique identifier for customer (FK)
	OrderDate	DATE	NO	NO	Date of the order
	ProductID	VARCHAR2 (5)	NO	NO	Unique identifier for product
	Quantity	NUMBER (5	NO	NO	Quantity of the product
	TotalPrice	NUMBER(5, 2)	NO	NO	Total price of the order
Payment	PaymentID	VARCHAR2 (5)	NO	NO	Unique identifier for payment
	OrderID	VARCHAR2 (5)	NO	NO	Unique identifier for order (FK)
	PaymentDate	DATE	NO	NO	Date of the payment

	PaymentMethod	VARCHAR2 (30)	NO	NO	Method used for payment
	TotalPrice	NUMBER(5, 2)	NO	NO	Total price of the product
Receipt	ReceiptID	VARCHAR2 (5)	NO	NO	Unique identifier for receipt
	OrderID	VARCHAR2 (5)	NO	NO	Unique identifier for order (FK)
	TotalPrice	NUMBER(5, 2)	NO	NO	Total price of the product
Cash On Delivery	PaymentID	VARCHAR2 (5)	NO	NO	Unique identifier for payment (FK)
	ConfirmationStatus	VARCHAR2 (15)	NO	NO	Confirmation status of payment
	OrderID	VARCHAR2 (5)	NO	NO	Unique identifier for order(FK)
OnlineBanking	BankName	VARCHAR2 (30	NO	NO	The name of the bank used
	ReferenceNumber	NUMBER (15)	NO	NO	Uniquely identifies each customer transaction
	AccountHolderName	VARCHAR2 (50)	NO	NO	Name of the account holder

4.3. Normalization

1. **Admin**
   - Attributes: AdminID, AdminName, Email
   - Functional Dependencies:
     - AdminID → AdminName, Email
     - AdminName → AdminID, Email
     - Email → AdminID, AdminName
   - Normalization:
     1. 1NF (First Normal Form):
     - All attributes are atomic (no repeating groups or arrays).
     2. 2NF (Second Normal Form):
     - AdminID is a candidate key, and there are no partial dependencies.
     3. 3NF (Third Normal Form):
     - There are no transitive dependencies.

2. **Seller**
   - Attributes: SellerID, SellerName, SellerEmail, SellerPhoneNum
   - Functional Dependencies:
     - SellerID → SellerName, SellerEmail, SellerPhoneNum
     - SellerEmail → SellerID, SellerName, SellerPhoneNum
   - Normalization:
     1. 1NF (First Normal Form):
     - All attributes are atomic (no repeating groups or arrays).
     2. 2NF (Second Normal Form):
     - SellerID is a candidate key, and there are no partial dependencies.

3. 3NF (Third Normal Form):
   - There are no transitive dependencies.

3. **Customer**

- Attributes: CustID, CustName, CustEmail, CustPhoneNum, CustAddress
- Functional Dependencies:
  - CustID → CustName, CustEmail, CustPhoneNum, CustAddress
  - CustEmail → CustID, CustName, CustPhoneNum, CustAddress

- Normalization:

  1. 1NF (First Normal Form):
     - All attributes are atomic (no repeating groups or arrays).

  2. 2NF (Second Normal Form):
     - CustID is a candidate key, and there are no partial dependencies.

  3. 3NF (Third Normal Form):
     - There are no transitive dependencies.

4. **Product**

- Attributes:ProductID, ProductName, Quantity, Category, Description, Prices
- Functional Dependencies:
  - ProductID → ProductName, Quantity, Category, Description, Prices

- Normalization:

  1. 1NF (First Normal Form):
     - All attributes are atomic (no repeating groups or arrays).

  2. 2NF (Second Normal Form):

- ProductID is a candidate key, and there are no partial dependencies.

3. 3NF (Third Normal Form):
   - There are no transitive dependencies

5. Order

- Attributes: OrderID, CustID, OrderDate, TotalPrice
- Functional Dependencies:
  - OrderID → CustID, OrderDate, TotalPrice
  - CustID → OrderID, OrderDate, TotalPrice

- Normalization:
  1. 1NF (First Normal Form):
     - All attributes are atomic (no repeating groups or arrays).

  2. 2NF (Second Normal Form):
     - OrderID is a candidate key, and there are no partial dependencies.

  3. 3NF (Third Normal Form):
     - There are no transitive dependencies

6. Payment

- Attributes: PaymentID, OrderID, PaymentDate, PaymentMethod, TotalPrice
- Functional Dependencies:
  - PaymentID → OrderID, PaymentDate, PaymentMethod, TotalPrice
  - OrderID → PaymentID, PaymentDate, PaymentMethod, TotalPrice

- Normalization:
  1. 1NF (First Normal Form):
     - All attributes are atomic (no repeating groups or arrays).

2. 2NF (Second Normal Form):
   - PaymentID is a candidate key, and there are no partial dependencies.

3. 3NF (Third Normal Form):
   - There are no transitive dependencies

7. Receipt

- Attributes: ReceiptID, OrderID, TotalPrice
- Functional Dependencies:
  - ReceiptID → OrderID, TotalPrice
  - OrderID → ReceiptID, TotalPrice
- Normalization:

1. 1NF (First Normal Form):
   - All attributes are atomic (no repeating groups or arrays).

2. 2NF (Second Normal Form):
   - ReceiptID is a candidate key, and there are no partial dependencies.

3. 3NF (Third Normal Form):
   - There are no transitive dependencies.

8. Cash On Delivery (Subclass of Payment)

- Attributes:PaymentID, ConfirmationStatus, OrderID
- Functional Dependencies:
  - PaymentID → ConfirmationStatus, OrderID
  - OrderID → PaymentID, ConfirmationStatus
- Normalization:

1. 1NF (First Normal Form):
   - All attributes are atomic (no repeating groups or arrays).

2. 2NF (Second Normal Form):
   - PaymentID is a candidate key, and there are no partial dependencies.

3. 3NF (Third Normal Form):
   - There are no transitive dependencies.

9. Online Banking (Subclass of Payment)

- Attributes:BankName, ReferenceNumber, AccountHolderName
- Functional Dependencies:
  - ReferenceNumber → BankName, AccountHolderName
- Normalization:

1. 1NF (First Normal Form):
   - All attributes are atomic (no repeating groups or arrays).

2. 2NF (Second Normal Form):
   - ReferenceNumber is a candidate key, and there are no partial dependencies.

3. 3NF (Third Normal Form):
   - There are no transitive dependencies.

.

5. **Relational DB Schema (After normalization)**

The relational database schema for online shopping database is a set of relation schemas, namely,

Admin	(AdminID, AdminName, AdminEmail)
Seller	(SellerID, SellerName, SellerEmail, SellerPhoneNum)
Customer	(CustID, CustName, CustEmail, CustPhoneNum, CustAddress)
Product	(ProductID, ProductName, Quantity,Category, Description, Prices)
Order	(OrderID, CustID, OrderDate, TotalPrice)
Payment	(PaymentID, OrderID, PaymentDate, PaymentMethod, TotalPrice)
Receipt	(ReceiptID, OrderID, TotalPrice)
CashOnDelivery	(PaymentID, ComfirmationStatus, OrderID)

****Remark: Underline Word is Primary Key.**

Admin

AdminID	AdminName	AdminEmail

Seller

SellerID	SellerName	SellerEmail	SellerPhoneNum

Customer

CustID	CustName	CustEmail	CustPhoneNum	CustAddress

Product

ProductID	ProductName	Quantity	Category	Description	Prices

Order

OrderID	CustID	OrderDate	TotalPrice

Payment

PaymentID	OrderID	PaymentDate	PaymentMethod	TotalPrice

Receipt

ReceiptID	OrderID	TotalPrice

CashOnDelivery

PaymentID	ConfirmationStatus	OrderID

6. SQL statements

Sql is a domain-specific language used to manage and manipulate relational databases. SQL statements can be broadly categorized into two main types which are DDL (Data Definition Language) and DML (Data Manipulation Language).

i. Data Definition Language (DDL)

DDL (Data Definition Language) includes SQL statements for defining and managing database structure. Simple DDL commands, such as CREATE for creating tables, ALTER for modifying structure, and DROP for deleting objects

```
-- Create Admin table
CREATE TABLE Admin (
   adminID VARCHAR2(5) CONSTRAINT Admin_PK PRIMARY KEY,
   adminName VARCHAR2(30),
   adminEmail VARCHAR2(30)
);

-- Create Seller table
CREATE TABLE Seller (
   sellerID VARCHAR2(5) CONSTRAINT Seller_PK PRIMARY KEY,
   sellerEmail VARCHAR2(30),
   sellerName VARCHAR2(30),
   sellerPhoneNum NUMBER(20)
);

-- Create Customer table
CREATE TABLE Customer (
   custID VARCHAR2(5) CONSTRAINT Customer_PK PRIMARY KEY,
   custEmail VARCHAR2(30),
   custName VARCHAR2(30),
   custGender VARCHAR2(10),
```

```
   custPhoneNum VARCHAR2(30),
   custAddress VARCHAR2(50)
);

-- Create Product table
CREATE TABLE Product (
   productID VARCHAR2(5) CONSTRAINT Product_PK PRIMARY KEY,
   productName VARCHAR2(30),
   category VARCHAR2(30),
   quantity NUMBER(5),
   description VARCHAR2(50),
   price NUMBER(5,2)
);

-- Create Orders table
CREATE TABLE Orders (
   orderID VARCHAR2(5) CONSTRAINT Orders_PK PRIMARY KEY,
   custID VARCHAR2(5),
   productID VARCHAR2(5),
   quantity NUMBER(5),
   orderDate DATE,
   totalPrice NUMBER(5,2),
   CONSTRAINT Orders_FK1 FOREIGN KEY (custID) REFERENCES
Customer(custID),
   CONSTRAINT Orders_FK2 FOREIGN KEY (productID) REFERENCES
Product(productID)
);

-- Create Payment table
CREATE TABLE Payment (
   paymentID VARCHAR2(5) CONSTRAINT Payment_PK PRIMARY KEY,
   orderID VARCHAR2(5),
   paymentMethod VARCHAR2(30),
  paymentDate DATE,
```

```
   totalPrice NUMBER(5,2),
   CONSTRAINT Payment_FK FOREIGN KEY (orderID) REFERENCES
Orders(orderID)
);
```

-- Create CashOnDelivery table

```
CREATE TABLE CashOnDelivery (
   paymentID VARCHAR2(5) CONSTRAINT CashOnDelivery_PK PRIMARY
KEY,
   confirmationStatus VARCHAR2(15),
   orderID VARCHAR2(5),
   CONSTRAINT CashOnDelivery_FK FOREIGN KEY (orderID) REFERENCES
Orders(orderID)
);
```

-- Create OnlineBanking table

```
CREATE TABLE OnlineBanking (
   bankName VARCHAR2(30),
   referenceNumber NUMBER(15),
   accountHolderName VARCHAR2(50),
   paymentID VARCHAR2(5),
   CONSTRAINT OnlineBanking_PK PRIMARY KEY (paymentID),
   CONSTRAINT OnlineBanking_FK FOREIGN KEY (paymentID) REFERENCES
Payment(paymentID)
);
```

-- Create Receipt table

```
CREATE TABLE Receipt (
   receiptID VARCHAR2(5) CONSTRAINT Receipt_PK PRIMARY KEY,
   orderID VARCHAR2(5),
   totalPrice NUMBER(5,2),
   CONSTRAINT Receipt_FK FOREIGN KEY (orderID) REFERENCES
Orders(orderID)
);
```

ii. Data Manipulation Language (DML)

DML encompasses SQL statements used for interacting with and manipulating data within a database. Common DML commands include SELECT for retrieving data, INSERT for adding new records, UPDATE for modifying existing records, and DELETE for removing records. These commands enable the manipulation and control of the actual data stored in the database, ensuring efficient data handling and retrieval.

- **Admin table**

  - **Insert data into admin table**

```
INSERT INTO ADMIN
VALUES ('ADM01', 'Ahmad Abdullah', 'ahmad.abdullah@email.com');
```

  - **Select to retrieve data from admin table**

```
SELECT *
FROM ADMIN;
```

ADMINID	ADMINNAME	ADMINEMAIL
ADM01	Ahmad Abdullah	ahmad.abdullah@email.com

- **Customer table**

  - **Insert data into Customer table**

```
INSERT INTO CUSTOMER
VALUES ('C001', 'Amirah Hassan', 'amirah.hassan@email.com', 'Female',
'0127894567', 'Jalan Bunga Raya, Kuala Lumpur');
```

```
INSERT INTO CUSTOMER
VALUES ('C002', 'mohd.amir@email.com', 'Mohd Amir', 'Male', '0135678901', 'Jalan Tun Razak, Johor Bahru');

INSERT INTO CUSTOMER
VALUES ('C003', 'nurul.izzah@email.com', 'Nurul Izzah', 'Female', '0142136587', 'Jalan Kedah, Penang');

INSERT INTO CUSTOMER
VALUES ('C004', 'farhan.yusof@email.com’, 'Farhan Yusof', 'Male', '0153469875', 'Jalan Sultan Ismail, Kuala Lumpur');

INSERT INTO CUSTOMER
VALUES ('C005', 'zainalIsa@email.com’, 'zainal Isa', 'Male', '0182398741', 'Jalan Laksamana, Melaka');

INSERT INTO CUSTOMER
VALUES ('C006', 'Muhammad Haziq', 'muhammad.haziq@email.com', 'Male', '0112233444', 'Jalan Tok Guru, Kelantan');

INSERT INTO CUSTOMER
VALUES ('C007', 'ain.said@email.com', 'Ain Said', 'Female', '0196874532', 'Jalan Utama, Pulau Pinang');

INSERT INTO CUSTOMER
VALUES ('C008', 'azizul.hakim@email.com', 'Azizul Hakim', 'Male', '0187654321', 'Jalan Bukit Bintang, Kuala Lumpur');
```

- **Select to retrieve data from Customer table**

```
select *
from customer;
```

CUSTID	CUSTEMAIL	CUSTNAME	CUSTGENDER	CUSTPHONENUM	CUSTADDRESS
C001	amirah.hassan@email.com	Amirah Hassan	Female	0127894567	Jalan Bunga Raya, Kuala Lumpur
C002	mohd.amir@email.com	Mohd Amir	Male	0135678901	Jalan Tun Razak, Johor Bahru
C003	nurul.izzah@email.com	Nurul Izzah	Female	0142136587	Jalan Kedah, Penang
C004	farhan.yusof@email.com	Farhan Yusof	Male	0153469875	Jalan Sultan Ismail, Kuala Lumpur
C005	zainalIsa@email.com	zainal Isa	Male	0182398741	Jalan Laksamana, Melaka
C006	haziq00@email.com	Muhammad Haziq	Male	0112233444	Jalan Tok Guru, Kelantan
C007	ain.said@email.com	Ain Said	Female	0196874532	Jalan Utama, Pulau Pinang
C008	azizul.hakim@email.com	Azizul Hakim	Male	0187654321	Jalan Bukit Bintang, Kuala Lumpur

- **Seller table**

  - **Insert data into admin table**

INSERT INTO SELLER
VALUES ('S001', 'nor.azizah@email.com', 'Nor Azizah', 0123456789);

INSERT INTO SELLER
VALUES ('S002', 'muhammad.rahman@email.com', 'Muhammad Rahman', 0122334455);

INSERT INTO SELLER
VALUES ('S003', 'siti.aishah@email.com','Siti Aishah', 0111223344);

INSERT INTO SELLER
VALUES ('S004' , 'ahmad.ali@email.com','Ahmad Ali', 0199887766);

INSERT INTO SELLER

VALUES ('S005', 'nurul.huda@email.com', 'Nurul Huda', 0101010101);

- **Select to retrieve data from admin table**

```
SELECT *
FROM Seller;
```

SELLERID	SELLEREMAIL	SELLERNAME	SELLERPHONENUM
S001	nor.azizah@email.com	Nor Azizah	123456789
S002	muhammad.rahman@email.com	Muhammad Rahman	122334455
S003	siti.aishah@email.com	Siti Aishah	111223344
S004	ahmad.ali@email.com	Ahmad Ali	199887766
S005	nurul.huda@email.com	Nurul Huda	101010101

- **Update data from seller table**

The statement is updating the sellerPhoneNum for the seller with the name 'Nurul Huda' to the new value '0135983850'.

```
UPDATE SELLER
SET sellerPhoneNum = '0135983850'
WHERE sellerName = 'Nurul Huda';
```

1 row(s) updated.

SELLERID	SELLEREMAIL	SELLERNAME	SELLERPHONENUM
S001	nor.azizah@email.com	Nor Azizah	123456789
S002	muhammad.rahman@email.com	Muhammad Rahman	122334455
S003	siti.aishah@email.com	Siti Aishah	111223344
S004	ahmad.ali@email.com	Ahmad Ali	199887766
S005	nurul.huda@email.com	Nurul Huda	135983850

- **Delete data from seller table**

  This statement deletes the seller whose name is 'Muhammad Rahman' from the SELLER table.

```sql
DELETE FROM SELLER
WHERE sellerName = 'Muhammad Rahman';
```

1 row(s) deleted.

SELLERID	SELLEREMAIL	SELLERNAME	SELLERPHONENUM
S001	nor.azizah@email.com	Nor Azizah	123456789
S003	siti.aishah@email.com	Siti Aishah	111223344
S004	ahmad.ali@email.com	Ahmad Ali	199887766
S005	nurul.huda@email.com	Nurul Huda	101010101

- **Sorting data using order by sellerName**

  This query selects all columns from the SELLER table and orders the results based on the sellerName column in ascending order.

```sql
SELECT *
FROM SELLER
ORDER BY sellerName;
```

SELLERID	SELLEREMAIL	SELLERNAME	SELLERPHONENUM
S004	ahmad.ali@email.com	Ahmad Ali	199887766
S001	nor.azizah@email.com	Nor Azizah	123456789
S005	nurul.huda@email.com	Nurul Huda	101010101
S003	siti.aishah@email.com	Siti Aishah	111223344

- **Product table**

  - **Insert data into Product table**

```
INSERT INTO PRODUCT
VALUES ('P001', 'Stylish Denim Jacket', 'Fashion', 50, 'Trendy denim jacket for all occasions', 99.99);

INSERT INTO PRODUCT
VALUES ('P002', 'Luxurious Facial Cream', 'Beauty', 100, 'Premium facial cream for radiant skin', 49.99);

INSERT INTO PRODUCT
VALUES ('P003', 'High-Performance Running Shoes', 'Sport', 75, 'Comfortable shoes for running enthusiasts', 79.99);

INSERT INTO PRODUCT
VALUES ('P004', 'Smart LED TV', 'Electronic', 30, 'Ultra HD smart TV with advanced features', 699.99);

INSERT INTO PRODUCT
VALUES ('P005', 'Modern Coffee Table', 'Furniture', 20, 'Elegant coffee table for your living room', 149.99);

INSERT INTO PRODUCT
VALUES ('P006', 'Chic Summer Dress', 'Fashion', 60, 'Beautiful dress perfect for summer days', 69.99);

INSERT INTO PRODUCT
VALUES ('P007', 'Wireless Headphones', 'Electronic', 40, 'Immersive audio experience on the go', 129.99);

INSERT INTO PRODUCT
```

VALUES ('P008', 'Elegant Dining Table Set', 'Furniture', 15, 'Complete dining set for a stylish home', 299.99);

- **Select to retrieve data from admin table**

```sql
SELECT *
FROM PRODUCT;
```

PRODUCTID	PRODUCTNAME	CATEGORY	QUANTITY	DESCRIPTION	PRICE
P001	Stylish Denim Jacket	Fashion	50	Trendy denim jacket for all occasions	99.99
P002	Luxurious Facial Cream	Beauty	100	Premium facial cream for radiant skin	49.99
P003	High-Performance Running Shoes	Sport	75	Comfortable shoes for running enthusiasts	79.99
P004	Smart LED TV	Electronic	30	Ultra HD smart TV with advanced features	699.99
P005	Modern Coffee Table	Furniture	20	Elegant coffee table for your living room	149.99
P006	Chic Summer Dress	Fashion	60	Beautiful dress perfect for summer days	69.99
P007	Wireless Headphones	Electronic	40	Immersive audio experience on the go	129.99
P008	Elegant Dining Table Set	Furniture	15	Complete dining set for a stylish home	299.99

- **Update product table**

This UPDATE statement will set the productName to 'Denim jacket' and the quantity to 60 for the product with productID 'P001'

```sql
UPDATE Product
SET productname = 'Denim jacket', quantity = 60
WHERE productID = 'P001';
```

Resize Code Editor

PRODUCTID	PRODUCTNAME	CATEGORY	QUANTITY	DESCRIPTION	PRICE
P001	Denim jacket	Fashion	60	Trendy denim jacket for all occasions	99.99
P002	Luxurious Facial Cream	Beauty	100	Premium facial cream for radiant skin	49.99
P003	High-Performance Running Shoes	Sport	75	Comfortable shoes for running enthusiasts	79.99
P004	Smart LED TV	Electronic	30	Ultra HD smart TV with advanced features	699.99
P005	Modern Coffee Table	Furniture	20	Elegant coffee table for your living room	149.99
P006	Chic Summer Dress	Fashion	60	Beautiful dress perfect for summer days	69.99
P007	Wireless Headphones	Electronic	40	Immersive audio experience on the go	129.99
P008	Elegant Dining Table Set	Furniture	15	Complete dining set for a stylish home	299.99

- **Delete data from product table**

  This DELETE statement will remove the product with productID 'P001' from the Product table.

```
SELECT *
FROM product;
```

PRODUCTID	PRODUCTNAME	CATEGORY	QUANTITY	DESCRIPTION	PRICE
P001	Denim jacket	Fashion	60	Trendy denim jacket for all occasions	99.99
P002	Luxurious Facial Cream	Beauty	100	Premium facial cream for radiant skin	49.99
P003	High-Performance Running Shoes	Sport	75	Comfortable shoes for running enthusiasts	79.99
P004	Smart LED TV	Electronic	30	Ultra HD smart TV with advanced features	699.99
P005	Modern Coffee Table	Furniture	20	Elegant coffee table for your living room	149.99
P006	Chic Summer Dress	Fashion	60	Beautiful dress perfect for summer days	69.99
P007	Wireless Headphones	Electronic	40	Immersive audio experience on the go	129.99

- **Orders table**

  - **Insert data into Orders table**

INSERT INTO Orders (orderID, custID, productID, quantity, orderDate, totalPrice) VALUES ('ORD01', 'C001', 'P001', 2, TO_DATE('2024-01-15', 'YYYY-MM-DD'), 199.98);

INSERT INTO Orders (orderID, custID, productID, quantity, orderDate, totalPrice) VALUES ('ORD02', 'C002', 'P002', 1, TO_DATE('2024-01-17', 'YYYY-MM-DD'), 49.99);

INSERT INTO Orders (orderID, custID, productID, quantity, orderDate, totalPrice) VALUES ('ORD03', 'C003', 'P003', 3, TO_DATE('2024-01-20', 'YYYY-MM-DD'), 239.97);

INSERT INTO Orders (orderID, custID, productID, quantity, orderDate, totalPrice) VALUES ('ORD04', 'C004', 'P006', 1, TO_DATE('2024-01-22', 'YYYY-MM-DD'), 69.99);

INSERT INTO Orders (orderID, custID, productID, quantity, orderDate, totalPrice) VALUES ('ORD05', 'C005', 'P005', 1, TO_DATE('2024-01-25', 'YYYY-MM-DD'), 149.99);

- **Select to retrieve data from Orders table**

```
select *
from orders;
```

1 row(s) updated.

ORDERID	CUSTID	PRODUCTID	QUANTITY	ORDERDATE	TOTALPRICE
ORD01	C001	P001	2	15-JAN-24	199.98
ORD02	C002	P002	1	17-JAN-24	49.99
ORD03	C003	P003	3	20-JAN-24	239.97
ORD04	C004	P006	1	22-JAN-24	69.99
ORD05	C005	P005	1	25-JAN-24	149.99

- **Sorting data from orders table using order by totalprice**

This query will retrieve the specified columns from the Orders table and arrange the results in ascending order based on the totalPrice.

```
SELECT orderid, orderDate, totalPrice
FROM orders
ORDER BY totalPrice;
```

ORDERID	ORDERDATE	TOTALPRICE
ORD02	17-JAN-24	49.99
ORD04	22-JAN-24	69.99
ORD05	25-JAN-24	149.99
ORD01	15-JAN-24	199.98
ORD03	20-JAN-24	239.97

- **Payment table**

  - **Insert data into Payment table**

```
INSERT INTO Payment (paymentID, orderID, paymentMethod, paymentDate, totalPrice)
VALUES ('PAY01', 'ORD01', 'Online Banking', TO_DATE('2024-01-16', 'YYYY-MM-DD'), 199.98);

INSERT INTO Payment (paymentID, orderID, paymentMethod, paymentDate, totalPrice)
VALUES ('PAY02', 'ORD02', 'Cash on Delivery', TO_DATE('2024-01-18', 'YYYY-MM-DD'), 49.99);

INSERT INTO Payment (paymentID, orderID, paymentMethod, paymentDate, totalPrice)
VALUES ('PAY03', 'ORD03', 'Online Banking', TO_DATE('2024-01-21', 'YYYY-MM-DD'), 239.97);

INSERT INTO Payment (paymentID, orderID, paymentMethod, paymentDate, totalPrice)
VALUES ('PAY04', 'ORD04', 'Online Banking', TO_DATE('2024-01-23', 'YYYY-MM-DD'), 69.99);

INSERT INTO Payment (paymentID, orderID, paymentMethod, paymentDate, totalPrice)
VALUES ('PAY05', 'ORD05', 'Cash on Delivery', TO_DATE('2024-01-26', 'YYYY-MM-DD'), 149.99);
```

-

- **Select to retrieve data from Payment table**

```sql
select *
from payment;
```

1 row(s) updated.

PAYMENTID	ORDERID	PAYMENTMETHOD	PAYMENTDATE	TOTALPRICE
PAY01	ORD01	Online Banking	16-JAN-24	199.98
PAY02	ORD02	Cash on Delivery	18-JAN-24	49.99
PAY03	ORD03	Online Banking	21-JAN-24	239.97
PAY04	ORD04	Online Banking	23-JAN-24	69.99
PAY05	ORD05	Cash on Delivery	26-JAN-24	149.99

- **Join customer, orders, product and payment table.**

This query performs inner joins between the Customer, Orders, Product, and Payment tables based on their relationships. It selects the customer name (custName), product name (productName), payment method (paymentMethod), and total price (totalPrice) based on the specified conditions.

```sql
SELECT
    C.custName AS CustomerName,
    P.productName AS ProductName,
    Pa.paymentMethod AS PaymentMethod,
    Pa.totalPrice AS TotalPrice
FROM Customer C
JOIN Orders O ON C.custID = O.custID
JOIN Product P ON O.productID = P.productID
JOIN Payment Pa ON O.orderID = Pa.orderID;
```

CUSTOMERNAME	PRODUCTNAME	PAYMENTMETHOD	TOTALPRICE
Amirah Hassan	Denim jacket	Online Banking	199.98
Mohd Amir	Luxurious Facial Cream	Cash on Delivery	49.99
Nurul Izzah	High-Performance Running Shoes	Online Banking	239.97
Farhan Yusof	Chic Summer Dress	Online Banking	69.99
zainal Isa	Modern Coffee Table	Cash on Delivery	149.99

- **CashOnDelivery table**

  - **Insert data into CashOnDelivery table**

```
INSERT INTO CashOnDelivery (paymentID, confirmationStatus, orderID)
SELECT paymentID, 'Confirmed', orderID
FROM Payment
WHERE paymentMethod = 'Cash on Delivery';
```

  - **Select to retrieve data from CashOnDelivery table**

```
SELECT * FROM
CashOnDelivery;
```

5 rows selected.

PAYMENTID	CONFIRMATIONSTATUS	ORDERID
PAY02	Confirmed	ORD02
PAY05	Confirmed	ORD05

  - **Join Payment table and cashOnDelivery table**

This SQL query selects columns from both the CashOnDelivery and Payment tables, joining them on the paymentID column

```
SELECT
    CashOnDelivery.paymentID,
    CashOnDelivery.orderID,
    CashOnDelivery.confirmationStatus,
    payment.paymentDate,
    Payment.paymentMethod,
    Payment.totalPrice
FROM
    CashOnDelivery
JOIN
    Payment ON CashOnDelivery.paymentID = Payment.paymentID;
```

PAYMENTID	ORDERID	CONFIRMATIONSTATUS	PAYMENTDATE	PAYMENTMETHOD	TOTALPRICE
PAY02	ORD02	Confirmed	18-JAN-24	Cash on Delivery	49.99
PAY05	ORD05	Confirmed	26-JAN-24	Cash on Delivery	149.99

- **OnlineBanking table**

  - **Insert data into OnlineBanking table**

    INSERT INTO OnlineBanking (bankName, referenceNumber, accountHolderName, paymentID)
    VALUES ('Maybank', 1234567890145, 'Amirah Hassan', 'PAY01');

    INSERT INTO OnlineBanking (bankName, referenceNumber, accountHolderName, paymentID)
    VALUES ('CIMB', 9876543210345, 'Nurul Izzah', 'PAY03');

    INSERT INTO OnlineBanking (bankName, referenceNumber, accountHolderName, paymentID)
    VALUES ('HSBC', 5678901230678, 'Farhan Yusof', 'PAY04');

  - **Select to retrieve data from OnlineBanking table**

```
SELECT *
FROM onlinebanking;
```

BANKNAME	REFERENCENUMBER	ACCOUNTHOLDERNAME	PAYMENTID
Maybank	1234567890145	Amirah Hassan	PAY01
CIMB	9876543210345	Nurul Izzah	PAY03
HSBC	5678901230678	Farhan Yusof	PAY04

  -

- **Updata data from OnlineBanking table**

This UPDATE statement will change the bankName to 'Bank Islam' and the referenceNumber to 6334567788094 for the row where the paymentID is 'PAY004' in the OnlineBanking table.

```sql
UPDATE OnlineBanking
SET bankName = 'Bank Islam', referenceNumber = 6334567788094
WHERE paymentID = 'PAY04';
```

1 row(s) updated.

BANKNAME	REFERENCENUMBER	ACCOUNTHOLDERNAME	PAYMENTID
Maybank	1234567890145	Amirah Hassan	PAY01
CIMB	9876543210345	Nurul Izzah	PAY03
Bank Islam	6334567788094	Farhan Yusof	PAY04

- **Join Payment table and onlineBanking using inner join**

This SQL query selects columns from both the OnlineBanking and Payment tables and joins them using the INNER JOIN condition on the paymentID column.

```sql
SELECT
    OnlineBanking.bankName,
    OnlineBanking.referenceNumber,
    OnlineBanking.accountHolderName,
    Payment.paymentID AS PaymentID,
    Payment.orderID AS PaymentOrderID,
    Payment.paymentMethod,
    Payment.totalPrice
FROM OnlineBanking
INNER JOIN Payment ON OnlineBanking.paymentID = Payment.paymentID;
```

BANKNAME	REFERENCENUMBER	ACCOUNTHOLDERNAME	PAYMENTID	PAYMENTORDERID	PAYMENTMETHOD	TOTALPRICE
Maybank	1234567890145	Amirah Hassan	PAY01	ORD01	Online Banking	199.98
CIMB	9876543210345	Nurul Izzah	PAY03	ORD03	Online Banking	239.97
Bank Islam	6334567788094	Farhan Yusof	PAY04	ORD04	Online Banking	69.99

- **Receipt table**

  - **Insert data into receipt table**

INSERT INTO Receipt (receiptID, orderID, totalPrice)
VALUES ('REC01', 'ORD01', 199.98);

INSERT INTO Receipt (receiptID, orderID, totalPrice)
VALUES ('REC02', 'ORD02', 49.99);

INSERT INTO Receipt (receiptID, orderID, totalPrice)
VALUES ('REC03', 'ORD03', 299.97);

INSERT INTO Receipt (receiptID, orderID, totalPrice)
VALUES ('REC04', 'ORD04', 149.98);

INSERT INTO Receipt (receiptID, orderID, totalPrice)
VALUES ('REC05', 'ORD05', 79.99);

  - **Select to retrieve data from receipt table**

```
SELECT * FROM
RECEIPT;
```

RECEIPTID	ORDERID	TOTALPRICE
REC01	ORD01	199.98
REC02	ORD02	49.99
REC03	ORD03	299.97
REC04	ORD04	149.98
REC05	ORD05	79.99

7. Summary

In this phase, an online shopping system is a crucial component that stores and manages various types of data related to products, customers, orders and payments. it can give benefit to customers to buy online products easily. In conclusion, a well-designed database is the backbone of an efficient online shopping system. By carefully considering entities database conceptual design ERD, logical design ERD, normalization, relational DB schema and SQL statements.

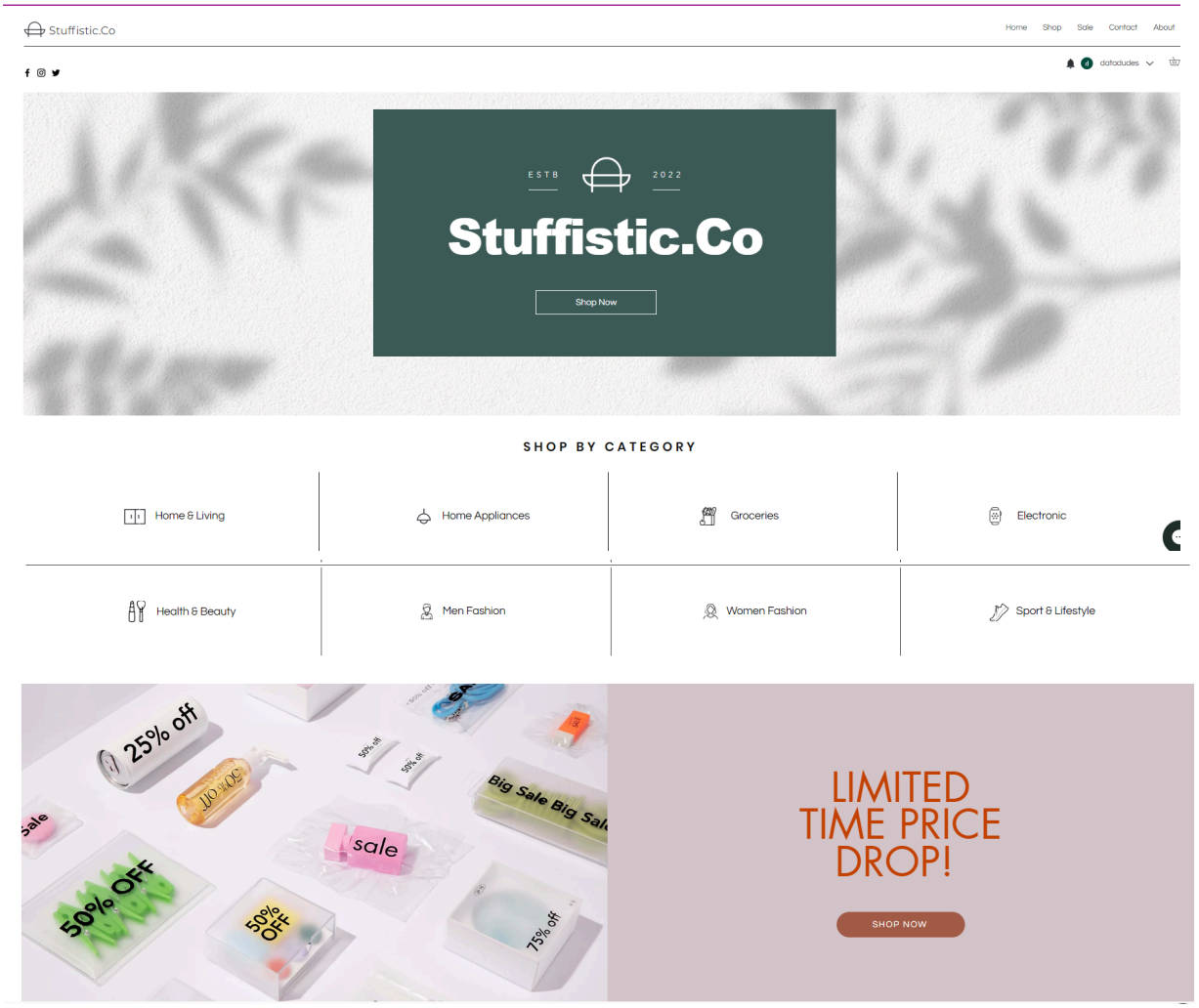
8. Link video presentation

https://youtu.be/YSzvm2PpsEc

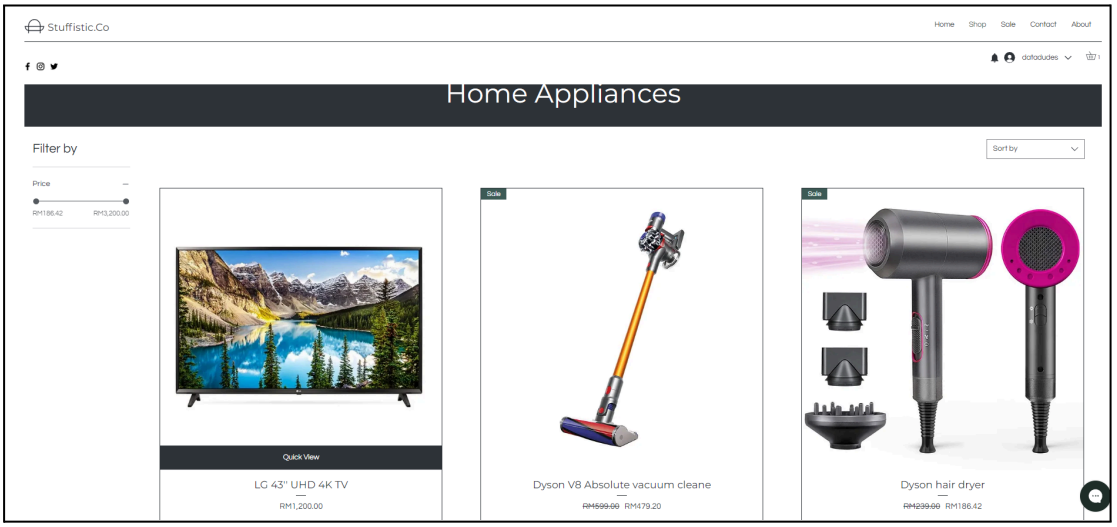
Appendix

UI for our system

Based on the figure below, this is the homepage for our system.



If a user wants to view products by category, this is the example, when selecting "Home and Appliances," it will display the following.



The figure below shows the payment section in which user can choose whether they want cash or online banking to pay their orders.