UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
SEMESTER 1/20232024

SECD2523-10 (Database)

Project : Phase 3

< VogueVerse.com >

Group : Burger (Group 9)

1. Liow Zhi Heng	A22EC0073
2. Denies Wong Ke Ying	A22EC0047
3. Lim Xiao Xuan	A22EC0071
4. Pang Zhi Xuan	A22SC0466

INSTRUCTOR: DR ROZILAWATI BINTI DOLLAH @ MD. ZAIN

Table of Contents

1.0	Introduction		3
2.0	Overview of project		3
3.0	Database Conceptual Design		4
	3.1	Updated business rule	4
	3.2	Conceptual ERD	5
4.0	DB logical design		6
	4.1	Logical ERD	6
	4.2	Updated Data Dictionary	6
	4.3	Normalization	13
5.0	Relational DB Schemas (after normalization)		17
6.0	SQL Statements (DDL & DML)		20
7.0	Summary		31

1.0 Introduction

The global COVID-19 pandemic has transformed consumer behavior, forcing businesses, including traditional physical shops, to rapidly adapt to online platforms. Online shopping has become an important part of daily life in Malaysia, providing consumers with a safe and convenient way to access necessary goods during movement restrictions. This transition has resulted in a growth in online commercial business as sellers handle the pandemic's challenges.

Even though the immediate threat of the pandemic has passed, the trend of online shopping continues to thrive. E-commerce has grown into a desired and popular option for consumers. Online transactions' simplicity, accessibility, and cost-effectiveness have not only maintained the growth that was established during the pandemic but also established e-commerce as an essential part of current consumer culture. Every consumer, regardless of age, is involved in this brand-new consumer culture.

As a result of these ongoing changes, businesses such as GENIE Fashion Style have embraced internet retail, particularly in the field of fashion. However, the transition to online operations has shown difficulties in managing orders, tracking shipments, and analyzing business data. Ms Joanne, the founder of GENIE Fashion Style, faces difficulties using existing platforms, which is causing bottlenecks in stock management, order tracking, and data analysis.

This proposal plans for the development of a new system to solve GENIE Fashion Style's current problems. This system attempts to analyze order data and payment status, enhance package tracking, and provide an organized stock management solution. It will not only address existing challenges but also contribute to a better user experience and a more convenient and effective business monitoring experience for the seller.

2.0 Overview of project

After finishing phase 2, we proceeded to phase 3 to implement the database logical design and also SQL statement for both DDL and DML for VogueVerse.com. We will change the conceptual ERD that we did in phase 2 to logical ERD and make a Relational database schema (normalized table). Normalization will also be done in this phase and a data dictionary for the normalized logical design will be done also for VogueVerse.com. relation. Lastly, the database for the website will be created using SQL statements.

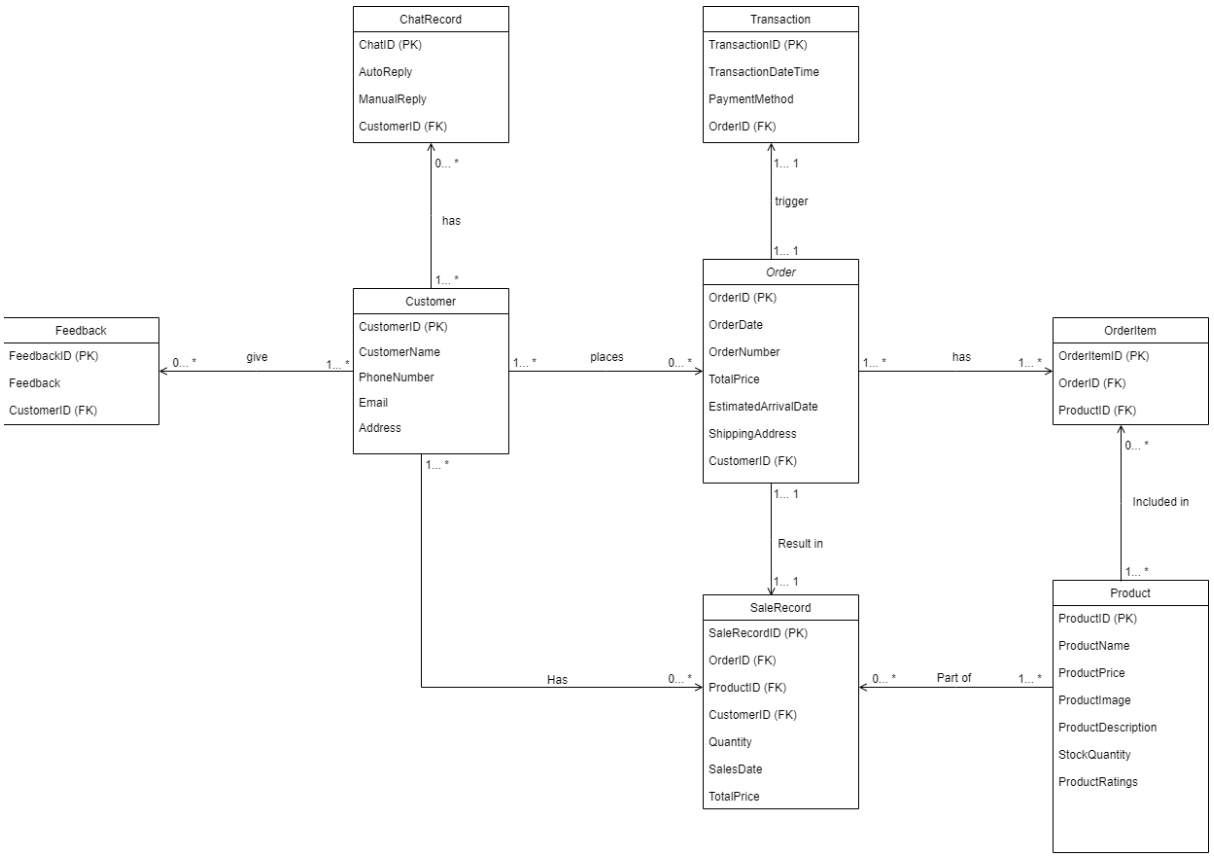
3.0 Database Conceptual Design

3.1 Updated business rule

1. The system will work 23 hours for 7 days.
2. The payment system will stop service every day at 12 am until 1 am because of the maintenance of the bank system.
3. Users will be required to sign up on the platform by entering their name, email, phone number and address.
4. The system will establish a systematic process for collecting and analyzing customer feedback, using insights to refine products, services, and the overall customer experience.
5. The system will automatically update the stock levels in real-time upon the arrival of new products or the fulfillment of customer orders.
6. When stock levels drop below a set point, the system will automatically notify users so she can make timely decisions about reordering.
7. The system will enable easy updating of product images and videos to showcase the latest visual content, improving the overall presentation of products.
8. By separating personal and business-related communications, the system will make it easier to communicate in a systematic way and guarantee that important information is accessible and clear.
9. Upon customer order placement, the system will automatically update relevant records, including order history and payment status of the product.
10. The system will immediately send automated order confirmation notifications to customers upon successful placement of an order.
11. The system offers flexible payment scheduling options, allowing customers to choose convenient payment intervals or installment plans for certain products.
12. The post office will implement an automated system to update shipping information with the post office once an order has been processed and dispatched.

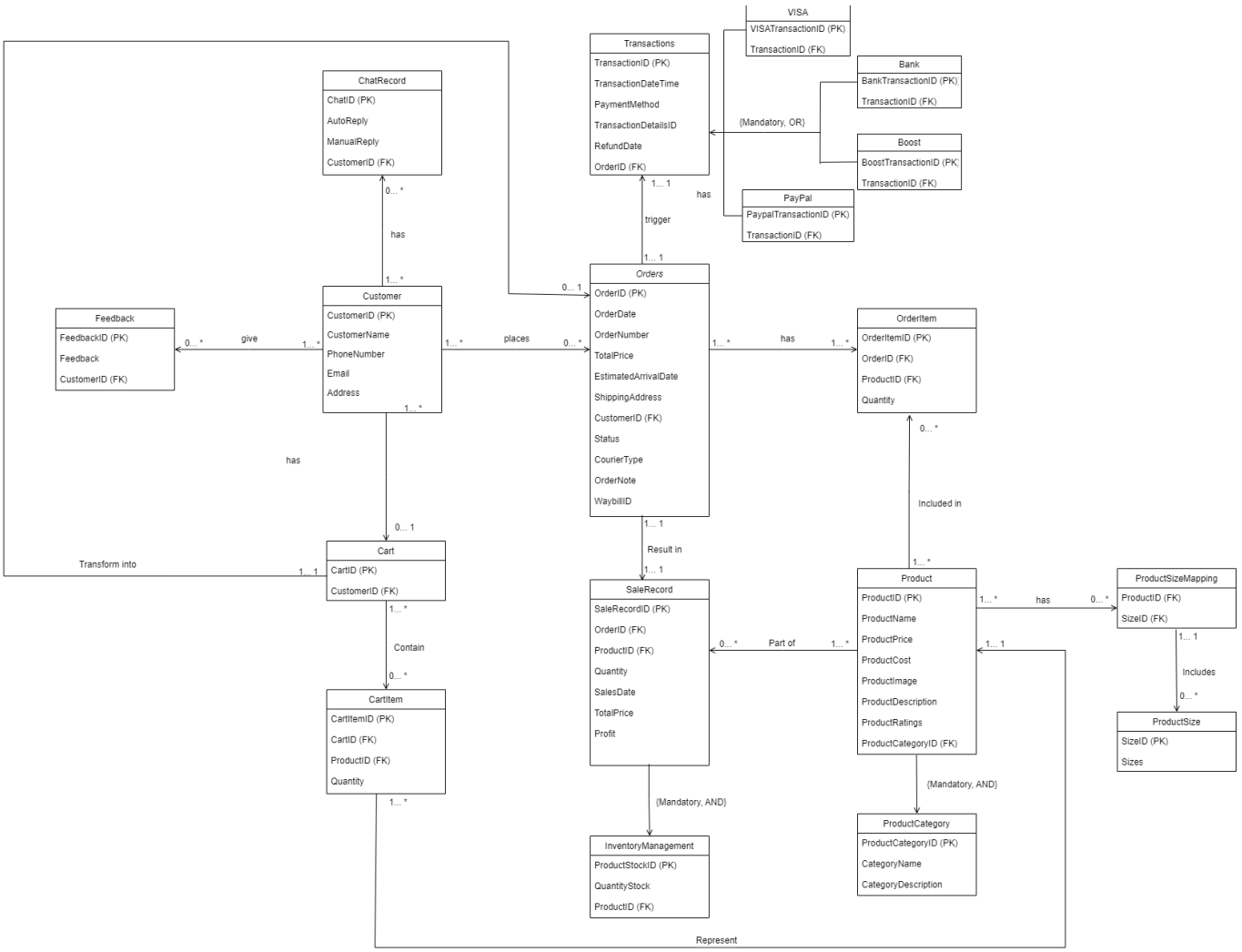
13. The system will provide an automated parcel tracking feature, enabling customers to monitor the accurate arrival time of their orders.
14. The system facilitates real-time parcel tracking updates by the post office to the platform and ensures accurate and timely information for users.
15. The system will automate profit and cost calculations, minimizing manual efforts and reducing the risk of errors in financial operations.
16. The system will enable the generation of sales reports for specified periods, allowing users to easily access and analyze sales performance.

3.2 Conceptual ERD



4.0 DB logical design

4.1 Logical ERD



4.2 Updated Data Dictionary

4.2.1 Description of entity

Entity	**Description**	**Occurence**
Customer	Hold customer data	A customer registers on the system
Cart	Hold cart data	A customer add item to the cart
CartItem	Hold data for each item in the cart	A specific item is added to a cart

Orders	Hold order data	A new order placed by customer
Product	Hold product data	A new product is added to the inventory
ProductCategory	Holds data for different categories of products	A new category of products is introduced
ProductSizeMapping	Maps products to their respective sizes	A new size is added for a product
ProductSize	Holds data for different sizes of products	A new size is introduced
Transactions	Holds transaction data	A customer completes a payment for an order
VISA	Holds transaction data	A customer choose VISA method to complete a payment for an order
Bank	Holds transaction data	A customer choose Bank method to complete a payment for an order
Boost	Holds transaction data	A customer choose Boost method to complete a payment for an order
PayPal	Holds transaction data	A customer choose PayPal method to complete a payment for an order
OrderItem	Holds data for each item in an order	A specific item is added to an order
Feedback	Holds feedback data provided by customers	A customer leaves feedback about their experience
ChatRecord	Holds data for each chat interaction between the customer and the system/seller	A new chat interaction occurs between the customer and the system
SaleRecord	Holds data for each sale	Record a new sale
InventoryManagement	Holds stock data	Record stock data

4.2.2 Relationships between entity

Entity	Multiplicity	Relationship	Multiplicity	Entity
Customer	1…*	Places	0…*	Orders
Customer	1…*	Gives	0…*	Feedback
Customer	1…*	Has	0…*	ChatRecord
Customer	1…*	Has	0…1	Cart
Cart	1…*	Contains	0…*	CartItem
Cart	1…1	Transform into	0…1	Orders
CartItem	1…*	Represents	1…1	Product
Orders	1…1	Triggers	1…1	Transactions
Orders	1…*	Has	1…*	OrderItem
Orders	1…1	Result in	1…1	SaleRecord
Product	1…*	Included in	0…*	OrderItem
Product	1…*	Part of	0…*	SaleRecord
Product	1…*	Belongs to	1…1	ProductCategory
Product	1…*	Has	0…*	ProductSizeMapping
ProductSizeMapping	1…1	Included	0…*	ProductSize

4.2.3 Attributes of entity

Entity	Attribute	Description	Data Type	Constraint
Customer	CustomerID	Customer’s ID	VARCHAR2(15)	PRIMARY KEY
	CustomerName	Customer’s name	VARCHAR2(30)	NOT NULL
	PhoneNumber	Customer’s phone number	VARCHAR2(12)	NOT NULL

	Email	Customer's email	VARCHAR2(30)	NOT NULL
	Address	Customer's address	VARCHAR2(50)	NOT NULL
Cart	CartID	Cart's ID	VARCHAR2(15)	PRIMARY KEY
	CustomerID	Customer's ID	VARCHAR2(15)	FOREIGN KEY reference Customer
CartItem	CartItemID	Cart Item's ID	VARCHAR2(15)	PRIMARY KEY
	CartID	Cart's ID	VARCHAR2(15)	FOREIGN KEY reference Cart
	ProductID	Product's ID	VARCHAR2(15)	FOREIGN KEY reference Product
	Quantity	Quantity of the product in the cart	NUMBER(10)	NOT NULL
Orders	OrderID	Order's ID	VARCHAR2(15)	PRIMARY KEY
	OrderDate	Order's date	DATE	NOT NULL
	OrderNumber	Order's number	VARCHAR2(15)	NOT NULL
	TotalPrice	Order's total price	DECIMAL (8,2)	NOT NULL
	EstimatedArrivalDate	Estimated Arrival Date of Order	DATE	NOT NULL
	ShippingAddress	Shipping Address for Order	VARCHAR2(50)	NOT NULL
	CustomerID	Customer's ID	VARCHAR2(15)	FOREIGN KEY reference

				Customer
	Status	Order status as ‘Completed’, ‘Cancelled’ or ‘Processing’	VARCHAR2(15)	NOT NULL
	CourierType	Brand of courier that use	VARCHAR2(15)	NOT NULL
	OrderNote	Order’s note for seller	VARCHAR2(30)	NOT NULL
	WaybillID	Waybill for percel	VARCHAR2(15)	NOT NULL
OrderItem	OrderItemID	Order Item’s ID	VARCHAR2(15)	PRIMARY KEY
	OrderID	Order’s ID	VARCHAR2(15)	FOREIGN KEY reference Orders
	ProductID	Product’s ID	VARCHAR2(15)	FOREIGN KEY reference Product
	Quantity	Quantity of the product in the order	NUMBER(10)	NOT NULL
Product	ProductID	Product’s ID	VARCHAR2(15)	PRIMARY KEY
	ProductName	Product’s name	VARCHAR2(30)	NOT NULL
	ProductPrice	Product’s unit price for sell	DECIMAL (8,2)	NOT NULL
	ProductCost	Product’s production cost	DECIMAL (8,2)	NOT NULL
	ProductImage	Product’s image	VARCHAR2(255)	NOT NULL
	ProductDescription	Product’s description	VARCHAR2(100)	NOT NULL
	ProductRatings	Product’s rating	DECIMAL(3, 2)	NOT NULL

	ProductCategoryID	Product Category's ID	VARCHAR2(15)	FOREIGN KEY reference ProductCategory
ProductCategory	ProductCategoryID	Product Category's ID	VARCHAR2(15)	PRIMARY KEY
	CategoryName	Category's Name	VARCHAR2(30)	NOT NULL
	CategoryDescription	Category's Description	VARCHAR2(100)	NOT NULL
ProductSize	SizeID	Size's ID	VARCHAR2(15)	PRIMARY KEY
	Sizes	Product size	VARCHAR2(10)	NOT NULL
ProductSizeMapping	ProductID	Product's ID	VARCHAR2(15)	FOREIGN KEY reference Product
	SizeID	Size's ID	VARCHAR2(15)	FOREIGN KEY reference ProductSize
Transactions	TransactionID	Transacrion's ID	VARCHAR2(15)	PRIMARY KEY
	TransactionDateTime	Transaction's time	TIMESTAMP	NOT NULL
	PaymentMethod	payment method	VARCHAR2(15)	NOT NULL
	TransactionDetailsID	Transaction details' ID	VARCHAR2(15)	NOT NULL
	RefundDate	Refund date when order is cancelled	DATE	-
	OrderID	Order's ID	VARCHAR2(15)	FOREIGN KEY reference Order
VISA	VISATransacti	VISATransactio	VARCHAR2(15)	PRIMARY

	onID	n’s ID		KEY
	TransactionID	Transacrion’s ID	VARCHAR2(15)	FOREIGN KEY reference Transactions
Bank	BankTransactionID	BankTransaction’s ID	VARCHAR2(15)	PRIMARY KEY
	TransactionID	Transacrion’s ID	VARCHAR2(15)	FOREIGN KEY reference Transactions
Boost	BoostTransactionID	BoostTransaction’s ID	VARCHAR2(15)	PRIMARY KEY
	TransactionID	Transacrion’s ID	VARCHAR2(15)	FOREIGN KEY reference Transactions
PayPal	PayPalTransactionID	PayPalTransaction’s ID	VARCHAR2(15)	PRIMARY KEY
	TransactionID	Transacrion’s ID	VARCHAR2(15)	FOREIGN KEY reference Transactions
SaleRecord	SaleRecordID	Sale record’s ID	VARCHAR2(15)	PRIMARY KEY
	Quantity	Sale’s quantity	NUMBER (10)	NOT NULL
	SalesDate	Sale’s date	DATE	NOT NULL
	TotalPrice	Sale’s total price	DECIMAL (10,2)	NOT NULL
	Profit	Profit earned	DECIMAL (10,2)	NOT NULL
	OrderID	Order’s ID	VARCHAR2(15)	FOREIGN KEY reference Orders
	ProductID	Product’s ID	VARCHAR2(15)	FOREIGN KEY reference

				Product
InventoyManagement	ProductStockID	ProductStock's ID	VARCHAR2(15)	PRIMARY KEY
	QuantityStock	Stock's quantity	NUMBER (10)	NOT NULL
	ProductID	Product's ID	VARCHAR2(15)	FOREIGN KEY reference Product
Feedback	FeedbackID	Feedback's ID	VARCHAR2(15)	PRIMARY KEY
	Feedback	Feedback from customer	CLOB	NOT NULL
	CustomerID	Customer's ID	VARCHAR2(15)	FOREIGN KEY reference Customer
ChatRecord	ChatID	Chat's ID	VARCHAR2(15)	PRIMARY KEY
	AutoReply	Automatic reply from system	CLOB	-
	ManualReply	Manual reply from seller	CLOB	-
	CustomerID	Customer's ID	VARCHAR2(15)	FOREIGN KEY reference Customer

4.3 Normalization

1. Customer (CustomerID, CustomerName, PhoneNumber, Email, Address)
    - **fd1:**
      CustomerID → CustomerName, PhoneNumber, Email, Address
    - **1NF & 2NF & 3NF & BCNF:**
      Customer (CustomerID, CustomerName, PhoneNumber, Email, Address)

2. Cart (CartID, CustomerID)
    - **fd1:**
      CartID → CustomerID
    - **1NF & 2NF & 3NF & BCNF:**
      Cart (<u>CartID</u>, CustomerID)

3. CartItem (CartItemID, CartID, ProductID, Quantity)
    - **fd1:**
      CartItemID → CartID, ProductID, Quantity
    - **1NF & 2NF & 3NF & BCNF:**
      CartItem (<u>CartItemID</u>, CartID, ProductID, Quantity)

4. Orders (OrderID, OrderDate, OrderNumber, TotalPrice, EstimatedArrivalDate, ShippingAddress, CustomerID, Status, CourierType, OrderNote, WaybillID)
    - **fd1:**
      OrderID → OrderDate, OrderNumber, TotalPrice, EstimatedArrivalDate, ShippingAddress, CustomerID, Status, CourierType, OrderNote, WaybillID
    - **1NF & 2NF & 3NF & BCNF:**
      Order (<u>OrderID</u>, OrderDate, OrderNumber, TotalPrice, EstimatedArrivalDate, ShippingAddress, CustomerID, Status, CourierType, OrderNote, WaybillID)

5. OrderItem (OrderItemID, OrderID, ProductID, Quantity)
    - **fd1:**
      OrderItemID → OrderID, ProductID, Quantity
    - **1NF & 2NF & 3NF & BCNF:**
      OrderItem (<u>OrderItemID</u>, OrderID, ProductID, Quantity)

6. Product (ProductID, ProductName, ProductPrice, ProductCost, ProductImage, ProductDescription, ProductRatings, ProductCategoryID)
    - **fd1:**
      ProductID → ProductName, ProductPrice, ProductCost, ProductImage, ProductDescription, ProductRatings, ProductCategoryID
    - **1NF & 2NF & 3NF & BCNF:**
      Product (<u>ProductID</u>, ProductName, ProductPrice, ProductCost, ProductImage, ProductDescription, ProductRatings, ProductCategoryID)

7. ProductCategory (ProductCategoryID, CategoryName, CategoryDescription)
    - **fd1:**
      ProductCategoryID → CategoryName, CategoryDescription
    - **1NF & 2NF & 3NF & BCNF:**
      ProductCategory (<u>ProductCategoryID</u>, CategoryName, CategoryDescription)

8. ProductSize (SizeID, Sizes)
    - **fd1:**

SizeID → Sizes

- **1NF & 2NF & 3NF & BCNF:**
  ProductSize (SizeID , Sizes)

9. ProductSizeMapping (ProductID, SizeID)
   - **fd1:**
     ProductID, SizeID → ProductID, SizeID
   - **1NF & 2NF & 3NF & BCNF:**
     ProductSizeMapping (ProductID, SizeID)

10. Transactions (TransactionID, TransactionDateTime, PaymentMethod,TransactionDetailsID, RefundDate OrderID)
    - **fd1:**
      TransactionID → TransactionDateTime, PaymentMethod, TransactionDetailsID, RefundDate OrderID
    - **1NF & 2NF & 3NF & BCNF:**
      Transaction (TransactionID, TransactionDateTime, PaymentMethod, TransactionDetailsID, RefundDate, OrderID)

11. VISATransaction (VISATransactionID, TransactionID)
    - **fd1:**
      VISATransactionID → TransactionID
    - **1NF & 2NF & 3NF & BCNF:**
      VISATransaction (VISATransactionID, TransactionID)

12. BankTransaction (BankTransactionID, TransactionID)
    - **fd1:**
      BankTransactionID → TransactionID
    - **1NF & 2NF & 3NF & BCNF:**
      BankTransaction (BankTransactionID, TransactionID)

13. BoostTransaction (BoostTransactionID, TransactionID)
    - **fd1:**
      BoostTransactionID → TransactionID
    - **1NF & 2NF & 3NF & BCNF:**
      BoostTransaction (BoostTransactionID, TransactionID)

14. PayPalTransaction (PayPalTransactionID, TransactionID)
    - **fd1:**
      PayPalTransactionID → TransactionID
    - **1NF & 2NF & 3NF & BCNF:**
      PayPalTransaction (PayPalTransactionID, TransactionID)

15. SaleRecord (SaleRecordID, Quantity, SalesDate, TotalPrice, Profit, OrderID, ProductID)
    - **fd1:**
      SaleRecordID → Quantity, SalesDate, TotalPrice, Profit, OrderID, ProductID
    - **1NF & 2NF & 3NF & BCNF:**
      SaleRecord (SaleRecordID, Quantity, SalesDate, TotalPrice, Profit, OrderID, ProductID)

16. InventoryManagement (ProductStockID, QuantityStock, ProductID)
    - **fd1:**
      ProductStockID → QuantityStock, ProductID
    - **1NF & 2NF & 3NF & BCNF:**
      InventoryManagement (ProductStockID, QuantityStock, ProductID)

17. Feedback (FeedbackID, Feedback, CustomerID)
    - **fd1:**
      FeedbackID → Feedback, CustomerID
    - **1NF & 2NF & 3NF & BCNF:**
      Feedback (FeedbackID, Feedback, CustomerID)

18. ChatRecord (ChatID, AutoReply, ManualReply, CustomerID)
    - **fd1:**
      ChatID → AutoReply, ManualReply, CustomerID
    - **1NF & 2NF & 3NF & BCNF:**
      ChatRecord (ChatID, AutoReply, ManualReply, CustomerID)

5.0 Relational DB Schemas (after normalization)

The relational database schema for the VogueVerse database consists of these sets of relation schemas.

- Customer (CustomerID, CustomerName, PhoneNumber, Email, Address)
- Cart (CartID, CustomerID)
- CartItem (CartItemID, CartID, ProductID, Quantity)
- Order (OrderID, OrderDate, OrderNumber, TotalPrice, EstimatedArrivalDate, ShippingAddress, CustomerID, Status, CourierType, OrderNote, WaybillID)
- OrderItem (OrderItemID, OrderID, ProductID, Quantity)
- Product (ProductID, ProductName, ProductPrice, ProductCost, ProductImage, ProductDescription, ProductRatings, ProductCategoryID)
- ProductCategory (ProductCategoryID, CategoryName, CategoryDescription)
- ProductSize (SizeID , Sizes)
- ProductSizeMapping (ProductID, SizeID)
- Transaction (TransactionID, TransactionDateTime, PaymentMethod, TransactionDetailsID, RefundDate, OrderID)
- VISATransaction (VISATransactionID, TransactionID)
- BankTransaction (BankTransactionID, TransactionID)
- BoostTransaction (BoostTransactionID, TransactionID)
- PayPalTransaction (PayPalTransactionID, TransactionID)
- SaleRecord (SaleRecordID, Quantity, SalesDate, TotalPrice, Profit, OrderID, ProductID)
- InventoryManagement (ProductStockID, QuantityStock, ProductID)
- Feedback (FeedbackID, Feedback, CustomerID)
- ChatRecord (ChatID, AutoReply, ManualReply, CustomerID)

Customer

CustomerID	CustomerName	PhoneNumber	Email	Address

Cart

CartID	CustomerID

CartItem

CartItemID	CartID	ProductID	Quantity

Orders

OrderID	OrderDate	OrderNumber	TotalPrice	EstimatedArrivalDate	
ShippingAddress	CustomerID	Status	CourierType	OrderNote	WaybillID

OrderItem

OrderItemID	OrderID	ProductID	Quantity

Product

ProductID	ProductName	ProductPrice	ProductCost
ProductImage	ProductDescription	ProductRatings	ProductCategoryID

ProductCategory

ProductCategoryID	CategoryName	CategoryDescription

ProductSize

SizeID	Sizes

ProductSizeMapping

ProductID	SizeID

Transactions

TransactionID	TransactionDateTime	PaymentMethod
TransactionDetailsID	RefundDate	OrderID

VISATransaction

VISATransactionID	TransactionID

BankTransaction

BankTransactionID	TransactionID

BoostTransaction

BoostTransactionID	TransactionID

PayPalTransaction

PayPalTransactionID	TransactionID

SaleRecord

SaleRecordID	Quantity	SalesDate	TotalPrice
Profit	OrderID	ProductID	

InventoryManagement

ProductStockID	QuantityStock	ProductID

Feedback

FeedbackID	Feedback	CustomerID

ChatRecord

ChatID	AutoReply	ManualReply	CustomerID

6.0 SQL Statements (DDL & DML)

DDL

```
--create table for each entity
CREATE TABLE Customer (
    CustomerID VARCHAR2(15) PRIMARY KEY,
    CustomerName VARCHAR2(30) NOT NULL,
    PhoneNumber VARCHAR2(12) NOT NULL,
    Email VARCHAR2(30) NOT NULL,
    Address VARCHAR2(50) NOT NULL
);

CREATE TABLE ProductCategory (
    ProductCategoryID VARCHAR2(15) PRIMARY KEY,
    CategoryName VARCHAR2(30) NOT NULL,
    CategoryDescription VARCHAR2(100) NOT NULL
);

CREATE TABLE Product (
    ProductID VARCHAR2(15) PRIMARY KEY,
    ProductName VARCHAR2(30) NOT NULL,
    ProductPrice DECIMAL(8,2) NOT NULL,
    ProductImage VARCHAR2(255) NOT NULL,
    ProductDescription VARCHAR2(100) NOT NULL,
    ProductRatings DECIMAL(3,2) NOT NULL,
    ProductCategoryID VARCHAR2(15),
    FOREIGN KEY (ProductCategoryID) REFERENCES
ProductCategory(ProductCategoryID)
);

CREATE TABLE ProductSize (
    SizeID VARCHAR2(15) PRIMARY KEY,
    Sizes VARCHAR2(10) NOT NULL
);

CREATE TABLE ProductSizeMapping (
    ProductID VARCHAR2(15),
    SizeID VARCHAR2(15),
    PRIMARY KEY (ProductID, SizeID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (SizeID) REFERENCES ProductSize(SizeID)
);

CREATE TABLE Cart (
    CartID VARCHAR2(15) PRIMARY KEY,
    CustomerID VARCHAR2(15),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
```

```
);

CREATE TABLE CartItem (
    CartItemID VARCHAR2(15) PRIMARY KEY,
    CartID VARCHAR2(15),
    ProductID VARCHAR2(15),
    Quantity NUMBER(10) NOT NULL,
    FOREIGN KEY (CartID) REFERENCES Cart(CartID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE Orders (
    OrderID VARCHAR2(15) PRIMARY KEY,
    OrderDate DATE NOT NULL,
    OrderNumber VARCHAR2(15) NOT NULL,
    TotalPrice DECIMAL(8,2) NOT NULL,
    EstimatedArrivalDate DATE NOT NULL,
    ShippingAddress VARCHAR2(50) NOT NULL,
    CustomerID VARCHAR2(15),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);

CREATE TABLE OrderItem (
    OrderItemID VARCHAR2(15) PRIMARY KEY,
    OrderID VARCHAR2(15),
    ProductID VARCHAR2(15),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE Transactions (
    TransactionID VARCHAR2(15) PRIMARY KEY,
    TransactionDateTime TIMESTAMP NOT NULL,
    PaymentMethod VARCHAR2(15) NOT NULL,
    TransactionDetailsID VARCHAR2(15) NOT NULL,
    OrderID VARCHAR2(15),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);

CREATE TABLE VISA (
    VISATransactionID VARCHAR2(15) PRIMARY KEY,
    TransactionID VARCHAR2(15),
    FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID)
);

CREATE TABLE Bank (
    BankTransactionID VARCHAR2(15) PRIMARY KEY,
```

```
    TransactionID VARCHAR2(15),
    FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID)
);

CREATE TABLE Boost (
    BoostTransactionID VARCHAR2(15) PRIMARY KEY,
    TransactionID VARCHAR2(15),
    FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID)
);

CREATE TABLE PayPal (
    PayPalTransactionID VARCHAR2(15) PRIMARY KEY,
    TransactionID VARCHAR2(15),
    FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID)
);

CREATE TABLE SaleRecord (
    SaleRecordID VARCHAR2(15) PRIMARY KEY,
    Quantity NUMBER(10) NOT NULL,
    SalesDate DATE NOT NULL,
    TotalPrice DECIMAL(10,2) NOT NULL,
    OrderID VARCHAR2(15),
    ProductID VARCHAR2(15),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE InventoryManagement (
    ProductStockID VARCHAR2(15),
    QuantityStock NUMBER(10) NOT NULL,
    ProductID VARCHAR2(15)
);

CREATE TABLE Feedback (
    FeedbackID VARCHAR2(15) PRIMARY KEY,
    Feedback CLOB NOT NULL,
    CustomerID VARCHAR2(15),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);

CREATE TABLE ChatRecord (
    ChatID VARCHAR2(15) PRIMARY KEY,
    AutoReply CLOB,
    ManualReply CLOB,
    CustomerID VARCHAR2(15),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
```

```
ALTER TABLE InventoryManagement
ADD CONSTRAINT pk_InventoryManagement PRIMARY KEY (ProductStockID);

ALTER TABLE InventoryManagement
ADD CONSTRAINT fk_InventoryManagement_Product FOREIGN KEY (ProductID)
REFERENCES Product(ProductID);

ALTER TABLE Orders
ADD Status VARCHAR2(15) NOT NULL;

ALTER TABLE Orders
ADD CourierType VARCHAR2(15) NOT NULL;

ALTER TABLE Orders
ADD OrderNote VARCHAR2(30) NOT NULL;

ALTER TABLE Orders
ADD WaybillID VARCHAR2(15) NOT NULL;

ALTER TABLE OrderItem
ADD Quantity NUMBER(10) NOT NULL;

ALTER TABLE Transactions
ADD (RefundDate DATE);

ALTER TABLE Product
ADD ProductCost DECIMAL (8,2) NOT NULL;

ALTER TABLE SaleRecord
ADD Profit DECIMAL (10,2) NOT NULL;
```

DML

```
--Customer

INSERT INTO Customer (CustomerID, CustomerName, PhoneNumber, Email,
Address)
VALUES ('CUST001', 'Joyce', '0127075670', 'joyce@gmail.com', '60, Pesona
16');

INSERT INTO Customer (CustomerID, CustomerName, PhoneNumber, Email,
Address)
```

```
VALUES ('CUST002', 'Chloe', '01111311384', 'chloe@gmail.com', '25, Cantik
9');

INSERT INTO Customer (CustomerID, CustomerName, PhoneNumber, Email,
Address)
VALUES ('CUST003', 'Alice', '0167990855', 'alicsmith@gmail.com', '20, Lawa
16');

INSERT INTO Customer (CustomerID, CustomerName, PhoneNumber, Email,
Address)
VALUES ('CUST004', 'Ennis', '01128720516', 'ennis@gamil.com', '32, Menawan
8');

INSERT INTO Customer (CustomerID, CustomerName, PhoneNumber, Email,
Address)
VALUES ('CUST005', 'Winnie', '01111413231', 'winnie@gmail.com', '18, Ayu
25');

--ProductCategory

INSERT INTO ProductCategory (ProductCategoryID, CategoryName,
CategoryDescription)
VALUES ('CAT01', 'Dresses', 'All types of dresses for women like casual,
formal, party wear, etc.');

INSERT INTO ProductCategory (ProductCategoryID, CategoryName,
CategoryDescription)
VALUES ('CAT02', 'Tops & Blouses', 'T-shirts, tank tops, blouses, and more
for women.');

INSERT INTO ProductCategory (ProductCategoryID, CategoryName,
CategoryDescription)
VALUES ('CAT03', 'Bottoms', 'Pants, skirts, shorts, and other types of
bottom wear for women.');

INSERT INTO ProductCategory (ProductCategoryID, CategoryName,
CategoryDescription)
VALUES ('CAT04', 'Outerwear', 'Items like jackets, sweaters, and coats for
women.');

INSERT INTO ProductCategory (ProductCategoryID, CategoryName,
CategoryDescription)
VALUES ('CAT05', 'Accessories', 'Items like jewelry, handbags, scarves,
hats, and more for women.');
```

```
--Product
INSERT INTO Product (ProductID, ProductName, ProductPrice, ProductCost,
ProductImage, ProductDescription, ProductRatings, ProductCategoryID)
VALUES ('PROD01', 'Summer Floral Dress', 50.00, 30.00,
'http://yourwebsite.com/images/summer_floral_dress.jpg', 'A beautiful
floral dress perfect for summer.', 4.8, 'CAT01');

INSERT INTO Product (ProductID, ProductName, ProductPrice, ProductCost,
ProductImage, ProductDescription, ProductRatings, ProductCategoryID)
VALUES ('PROD02', 'Classic White Blouse', 40.00, 25.00,
'http://yourwebsite.com/images/classic_white_blouse.jpg', 'A classic white
blouse that pairs well with any outfit.', 4.7, 'CAT02');

INSERT INTO Product (ProductID, ProductName, ProductPrice, ProductCost,
ProductImage, ProductDescription, ProductRatings, ProductCategoryID)
VALUES ('PROD03', 'Black Pencil Skirt', 60.00, 35.00,
'http://yourwebsite.com/images/black_pencil_skirt.jpg', 'A versatile black
pencil skirt for the office or a night out.', 4.5, 'CAT03');

INSERT INTO Product (ProductID, ProductName, ProductPrice, ProductCost,
ProductImage, ProductDescription, ProductRatings, ProductCategoryID)
VALUES ('PROD04', 'Denim Jacket', 80.00, 50.00,
'http://yourwebsite.com/images/denim_jacket.jpg', 'A stylish denim jacket
for cooler weather.', 4.8, 'CAT04');

INSERT INTO Product (ProductID, ProductName, ProductPrice, ProductCost,
ProductImage, ProductDescription, ProductRatings, ProductCategoryID)
VALUES ('PROD05', 'Canvas Tote Bag', 35.00, 20.00,
'http://yourwebsite.com/images/canvas_tote_bag.jpg', 'A versatile and
durable canvas tote bag, perfect for everyday use.', 4.9, 'CAT05');

--ProductSize
INSERT INTO ProductSize (SizeID, Sizes) VALUES ('SIZE01', 'XS');
INSERT INTO ProductSize (SizeID, Sizes) VALUES ('SIZE02', 'S');
INSERT INTO ProductSize (SizeID, Sizes) VALUES ('SIZE03', 'M');
INSERT INTO ProductSize (SizeID, Sizes) VALUES ('SIZE04', 'L');
INSERT INTO ProductSize (SizeID, Sizes) VALUES ('SIZE05', 'XL');


--ProductSizeMapping
INSERT INTO ProductSizeMapping (ProductID, SizeID) VALUES ('PROD01',
'SIZE01');
INSERT INTO ProductSizeMapping (ProductID, SizeID) VALUES ('PROD01',
'SIZE02');
INSERT INTO ProductSizeMapping (ProductID, SizeID) VALUES ('PROD02',
'SIZE03');
```

```
INSERT INTO ProductSizeMapping (ProductID, SizeID) VALUES ('PROD03',
'SIZE04');
INSERT INTO ProductSizeMapping (ProductID, SizeID) VALUES ('PROD04',
'SIZE05');

--initial stock quantity of the product
--InventoryManagement
INSERT INTO InventoryManagement (ProductStockID, QuantityStock, ProductID)
VALUES ('PS001', 100, 'PROD01');
INSERT INTO InventoryManagement (ProductStockID, QuantityStock, ProductID)
VALUES ('PS002', 100, 'PROD02');
INSERT INTO InventoryManagement (ProductStockID, QuantityStock, ProductID)
VALUES ('PS003', 100, 'PROD03');
INSERT INTO InventoryManagement (ProductStockID, QuantityStock, ProductID)
VALUES ('PS004', 100, 'PROD04');
INSERT INTO InventoryManagement (ProductStockID, QuantityStock, ProductID)
VALUES ('PS005', 100, 'PROD05');

--Cart
INSERT INTO Cart (CartID, CustomerID) VALUES ('CART01', 'CUST001');
INSERT INTO Cart (CartID, CustomerID) VALUES ('CART02', 'CUST002');
INSERT INTO Cart (CartID, CustomerID) VALUES ('CART03', 'CUST003');
INSERT INTO Cart (CartID, CustomerID) VALUES ('CART04', 'CUST004');
INSERT INTO Cart (CartID, CustomerID) VALUES ('CART05', 'CUST005');

--CartItem
INSERT INTO CartItem (CartItemID, CartID, ProductID, Quantity) VALUES
('CARTITEM01', 'CART01', 'PROD01', 2);
INSERT INTO CartItem (CartItemID, CartID, ProductID, Quantity) VALUES
('CARTITEM02', 'CART01', 'PROD02', 1);
INSERT INTO CartItem (CartItemID, CartID, ProductID, Quantity) VALUES
('CARTITEM03', 'CART02', 'PROD02', 2);
INSERT INTO CartItem (CartItemID, CartID, ProductID, Quantity) VALUES
('CARTITEM04', 'CART03', 'PROD03', 3);
INSERT INTO CartItem (CartItemID, CartID, ProductID, Quantity) VALUES
('CARTITEM05', 'CART04', 'PROD04', 1);
INSERT INTO CartItem (CartItemID, CartID, ProductID, Quantity) VALUES
('CARTITEM06', 'CART05', 'PROD05', 2);


—-----------------------------------------------------------------------

--Customers 1 places an order
--Orders
INSERT INTO Orders (OrderID, OrderDate, OrderNumber, TotalPrice,
EstimatedArrivalDate, ShippingAddress, CustomerID, Status, CourierType,
OrderNote, WaybillID)
```

```
VALUES ('ORD001', TO_DATE('2024-01-11', 'YYYY-MM-DD'), 'ON001',
    (SELECT SUM(p.ProductPrice * ci.Quantity) FROM CartItem ci JOIN
Product p ON ci.ProductID = p.ProductID WHERE ci.CartID = 'CART01'),
    TO_DATE('2024-01-18', 'YYYY-MM-DD'), '60, Pesona 16', 'CUST001',
'Processing', 'J&T', 'Handle with care', 'WB001');

--OrderItem
INSERT INTO OrderItem (OrderItemID, OrderID, ProductID, Quantity)
SELECT 'OI' || ROWNUM || 'ORD001', 'ORD001', ProductID, Quantity
FROM CartItem
WHERE CartID = 'CART01';

--Transactions
INSERT INTO Transactions (TransactionID, TransactionDateTime,
PaymentMethod, TransactionDetailsID, OrderID)
VALUES ('TRANS001', CURRENT_TIMESTAMP, 'VISA', 'TD001', 'ORD001');

INSERT INTO VISA (VISATransactionID, TransactionID)
VALUES ('VISA001', 'TRANS001');

--SaleRecord
INSERT INTO SaleRecord (SaleRecordID, Quantity, SalesDate, TotalPrice,
OrderID, ProductID, Profit)
SELECT 'SR' || ROWNUM || 'ORD001', oi.Quantity, o.OrderDate,
p.ProductPrice * oi.Quantity, o.OrderID, oi.ProductID, (p.ProductPrice -
p.ProductCost) * oi.Quantity
FROM Orders o
JOIN OrderItem oi ON o.OrderID = oi.OrderID
JOIN Product p ON oi.ProductID = p.ProductID
WHERE o.OrderID = 'ORD001';

--Update the stock quantity
--InventoryManagement
UPDATE InventoryManagement im
SET im.QuantityStock = im.QuantityStock -
    (SELECT SUM(oi.Quantity)
     FROM OrderItem oi
     WHERE oi.ProductID = im.ProductID AND oi.OrderID = 'ORD001')
WHERE EXISTS
    (SELECT 1
     FROM OrderItem oi
     WHERE oi.ProductID = im.ProductID AND oi.OrderID = 'ORD001');

--if customer 1 want to cancel order
--Customer ask for cancellation of order
INSERT INTO ChatRecord (ChatID, AutoReply, ManualReply, CustomerID)
```

```
VALUES ('CHAT001', 'Your request to cancel the order has been received.',
'Hello, I see that you wish to cancel your order. I am currently
processing your request. I will keep you updated on the status. Thank you
for your patience.', 'CUST001');

--Check if the order is still 'Processing'
SELECT Status FROM Orders WHERE OrderID = 'ORD001';

--If the status is 'Processing', cancel the order
UPDATE Orders
SET Status = 'Cancelled'
WHERE OrderID = 'ORD001' AND Status = 'Processing';

--Refund to the customer
UPDATE Transactions
SET RefundDate = CURRENT_DATE
WHERE TransactionID = 'TRANS001';

-- Update the inventory
UPDATE InventoryManagement im
SET im.QuantityStock = im.QuantityStock +
    (SELECT SUM(oi.Quantity)
     FROM OrderItem oi
     WHERE oi.ProductID = im.ProductID AND oi.OrderID = 'ORD001')
WHERE EXISTS
    (SELECT 1
     FROM OrderItem oi
     WHERE oi.ProductID = im.ProductID AND oi.OrderID = 'ORD001');

--Delete the sale record for the cancelled order
DELETE FROM SaleRecord WHERE OrderID = 'ORD001';

--Delete cart item since customer already cancel the order
DELETE FROM CartItem WHERE CartID = 'CART01';
```

```
--Customer 2 has a completed order and gives feedback
--Orders
```

```
INSERT INTO Orders (OrderID, OrderDate, OrderNumber, TotalPrice,
EstimatedArrivalDate, ShippingAddress, CustomerID, Status, CourierType,
OrderNote, WaybillID)
VALUES ('ORD002', TO_DATE('2024-01-12', 'YYYY-MM-DD'), 'ON002',
    (SELECT SUM(p.ProductPrice * ci.Quantity) FROM CartItem ci JOIN
Product p ON ci.ProductID = p.ProductID WHERE ci.CartID = 'CART02'),
    TO_DATE('2024-01-19', 'YYYY-MM-DD'), '25, Cantik 9', 'CUST002',
'Processing', 'J&T', 'Careful packaging please', 'WB002');

--OrderItem
INSERT INTO OrderItem (OrderItemID, OrderID, ProductID, Quantity)
SELECT 'OI' || ROWNUM || 'ORD002', 'ORD002', ProductID, Quantity
FROM CartItem
WHERE CartID = 'CART02';

--Transactions
INSERT INTO Transactions (TransactionID, TransactionDateTime,
PaymentMethod, TransactionDetailsID, OrderID)
VALUES ('TRANS002', CURRENT_TIMESTAMP, 'VISA', 'TD002', 'ORD002');

INSERT INTO VISA (VISATransactionID, TransactionID)
VALUES ('VISA002', 'TRANS002');

--SaleRecord
INSERT INTO SaleRecord (SaleRecordID, Quantity, SalesDate, TotalPrice,
OrderID, ProductID, Profit)
SELECT 'SR' || ROWNUM || 'ORD002', oi.Quantity, o.OrderDate,
p.ProductPrice * oi.Quantity, o.OrderID, oi.ProductID, (p.ProductPrice -
p.ProductCost) * oi.Quantity
FROM Orders o
JOIN OrderItem oi ON o.OrderID = oi.OrderID
JOIN Product p ON oi.ProductID = p.ProductID
WHERE o.OrderID = 'ORD002';

--Update the stock quantity
--InventoryManagement
UPDATE InventoryManagement im
SET im.QuantityStock = im.QuantityStock -
    (SELECT SUM(oi.Quantity)
     FROM OrderItem oi
     WHERE oi.ProductID = im.ProductID AND oi.OrderID = 'ORD002')
WHERE EXISTS
    (SELECT 1
     FROM OrderItem oi
     WHERE oi.ProductID = im.ProductID AND oi.OrderID = 'ORD002');

--Delete cart item since customer already placed the order
```

```
DELETE FROM CartItem WHERE CartID = 'CART02';

--After receiving the item, update the order status and customer gives
feedback
--Orders
UPDATE Orders
SET Status = 'Completed'
WHERE OrderID = 'ORD002';

--Feedback
INSERT INTO Feedback (FeedbackID, Feedback, CustomerID)
VALUES ('FB002', 'Great service and fast delivery!', 'CUST002');
```

7.0 Summary

The conceptual entity relationship diagram (ERD) from phase 3 of the GENIE Fashion System was successfully translated into a logical ERD during this project phase. In order to keep up with the updated business rules, this required determining and establishing functional dependencies between relationships. Then, using the logical ERD as a guide, we created relation schemas and normalized the data from first normal form (1NF) to Boyce-Codd normal form (BCNF). In order to ensure the effectiveness of website access and manipulation while maintaining data integrity for future operations, this normalization process tried to reduce data redundancy. We also updated the data dictionary, which provides accurate information about attributes, entities, and relationships, to reflect modifications brought about by the normalization. The final phase in this process was to use SQL statements in the Oracle Apex platform to create tables. The final objective is to provide Ms. Joanne with a system that is organized, effective, and easy to use so she can manage her business operations with simplicity.

In conclusion, the efforts in this phase are focused on improving the GENIE Fashion System's usability and functionality. We hope to provide Mrs. Joanne with a strong foundation for her business management needs by putting logical design principles, normalization techniques, and data dictionary updates into practice. This will ensure the system's efficiency and flexibility in response to changing needs.

8.0 Link Demonstrate Video

https://youtu.be/DislJ6hIUcM