Online Shopping Database Design

Group 3:Yaqi Hu, Canlin Jiang, Chih-Hsuan Su, Xinyi Wu, Yuqiao Feng

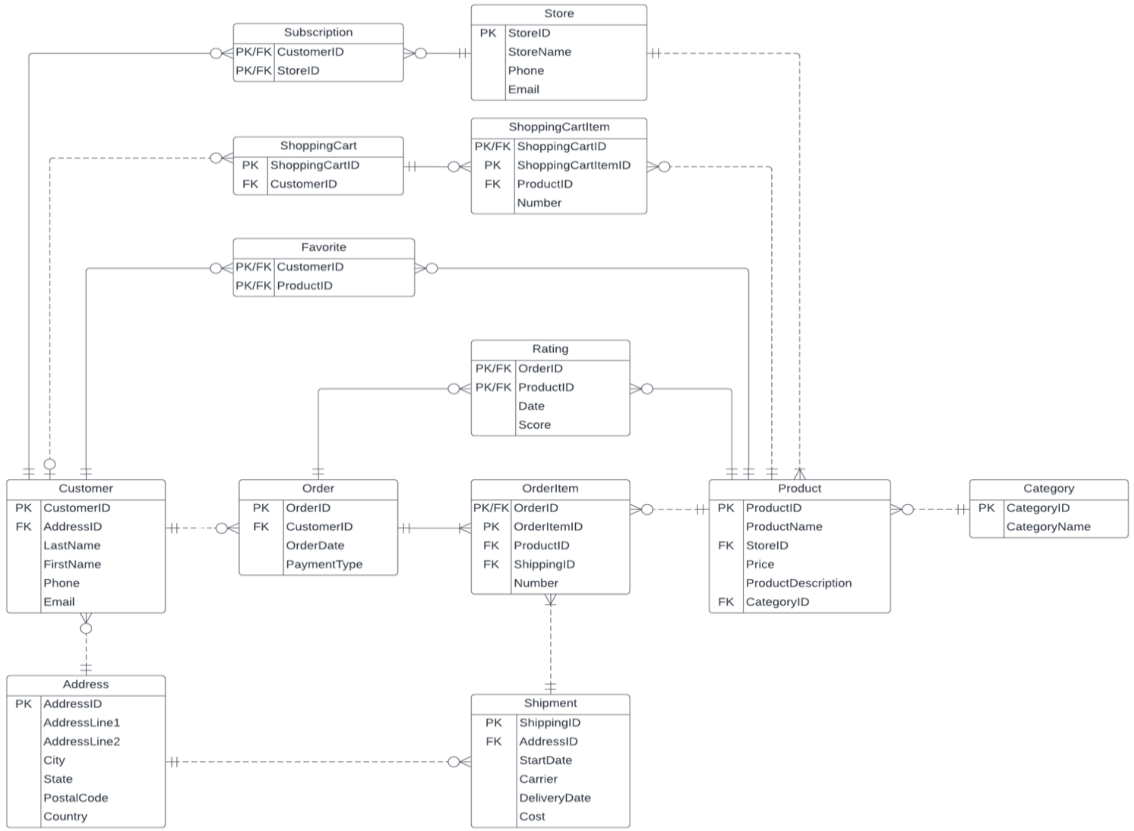
Overview

The purpose of this database is to hold the data used to create and sustain the customer shopping experience. It will be used by users and can view account information from the account information database.

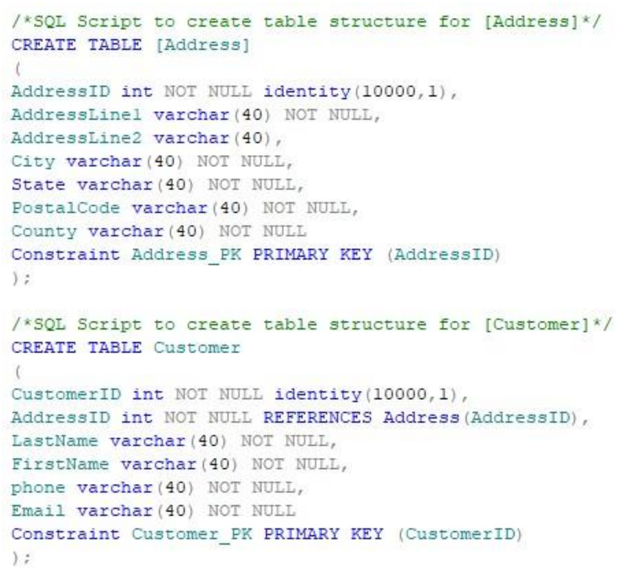
This database will act as a centralized system to manage shopping data (based on products, orders, customers) and provides customer insights about the operations associated with a store such as:

- Shopping carts
- Subscription
- Collection lists

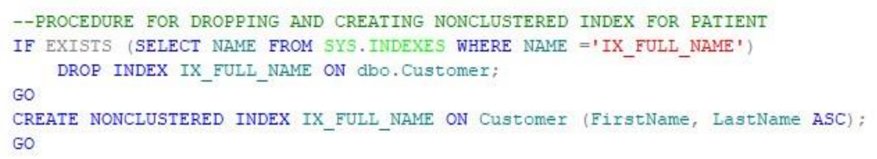
Design
Entity-Relationship Diagram



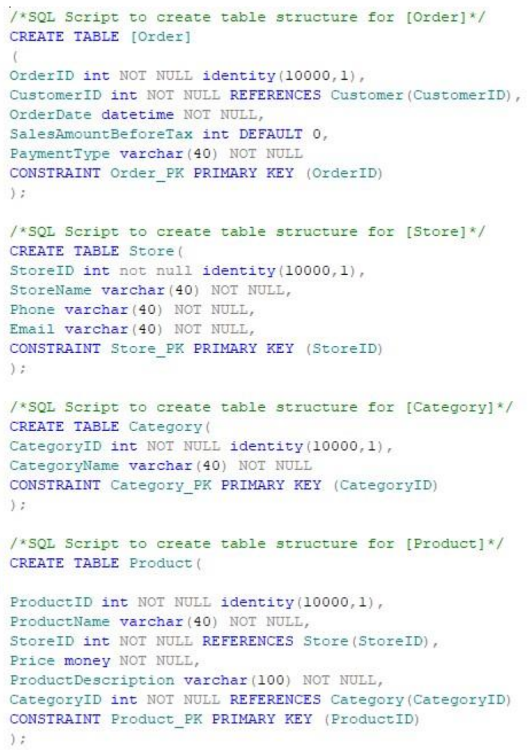
Create Table



```
/*SQL Script to create table structure for [Address]*/
CREATE TABLE [Address]
(
AddressID int NOT NULL identity(10000,1),
AddressLine1 varchar(40) NOT NULL,
AddressLine2 varchar(40),
City varchar(40) NOT NULL,
State varchar(40) NOT NULL,
PostalCode varchar(40) NOT NULL,
County varchar(40) NOT NULL
Constraint Address_PK PRIMARY KEY (AddressID)
);

/*SQL Script to create table structure for [Customer]*/
CREATE TABLE Customer
(
CustomerID int NOT NULL identity(10000,1),
AddressID int NOT NULL REFERENCES Address(AddressID),
LastName varchar(40) NOT NULL,
FirstName varchar(40) NOT NULL,
phone varchar(40) NOT NULL,
Email varchar(40) NOT NULL
Constraint Customer_PK PRIMARY KEY (CustomerID)
);
```



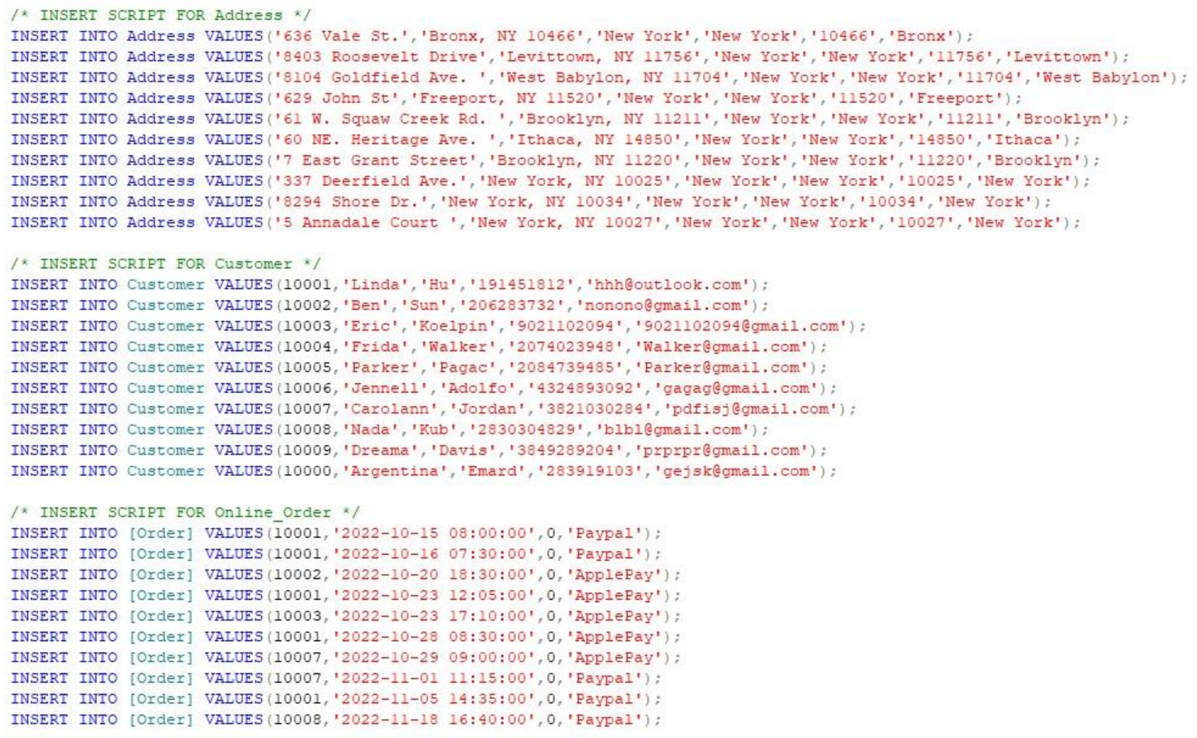
```
 --PROCEDURE FOR DROPPING AND CREATING NONCLUSTERED INDEX FOR PATIENT
 IF EXISTS (SELECT NAME FROM SYS.INDEXES WHERE NAME ='IX_FULL_NAME')
     DROP INDEX IX_FULL_NAME ON dbo.Customer;
 GO
 CREATE NONCLUSTERED INDEX IX_FULL_NAME ON Customer (FirstName, LastName ASC);
 GO
```



```
/*SQL Script to create table structure for [Order]*/
CREATE TABLE [Order]
(
OrderID int NOT NULL identity(10000,1),
CustomerID int NOT NULL REFERENCES Customer(CustomerID),
OrderDate datetime NOT NULL,
SalesAmountBeforeTax int DEFAULT 0,
PaymentType varchar(40) NOT NULL
CONSTRAINT Order_PK PRIMARY KEY (OrderID)
);

/*SQL Script to create table structure for [Store]*/
CREATE TABLE Store(
StoreID int not null identity(10000,1),
StoreName varchar(40) NOT NULL,
Phone varchar(40) NOT NULL,
Email varchar(40) NOT NULL,
CONSTRAINT Store_PK PRIMARY KEY (StoreID)
);

/*SQL Script to create table structure for [Category]*/
CREATE TABLE Category(
CategoryID int NOT NULL identity(10000,1),
CategoryName varchar(40) NOT NULL
CONSTRAINT Category_PK PRIMARY KEY (CategoryID)
);

/*SQL Script to create table structure for [Product]*/
CREATE TABLE Product(

ProductID int NOT NULL identity(10000,1),
ProductName varchar(40) NOT NULL,
StoreID int NOT NULL REFERENCES Store(StoreID),
Price money NOT NULL,
ProductDescription varchar(100) NOT NULL,
CategoryID int NOT NULL REFERENCES Category(CategoryID)
CONSTRAINT Product_PK PRIMARY KEY (ProductID)
);
```

Source: Database Implementation

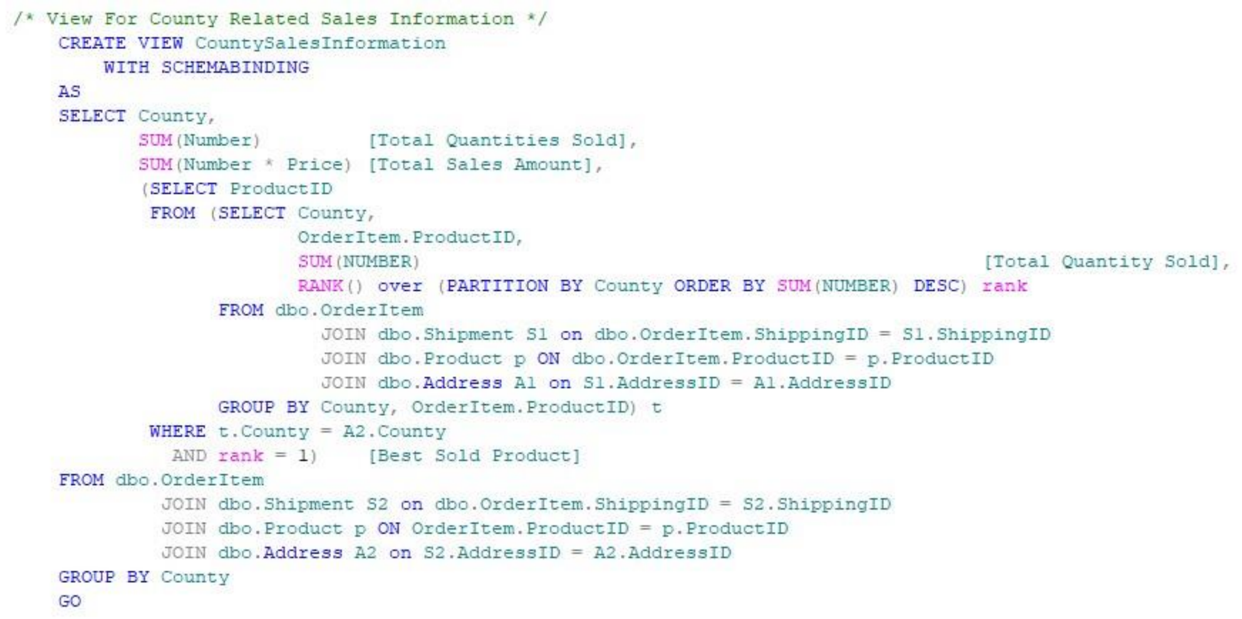
Insert Data By SQL Script



```sql
/* INSERT SCRIPT FOR Address */
INSERT INTO Address VALUES('636 Vale St.','Bronx, NY 10466','New York','New York','10466','Bronx');
INSERT INTO Address VALUES('8403 Roosevelt Drive','Levittown, NY 11756','New York','New York','11756','Levittown');
INSERT INTO Address VALUES('8104 Goldfield Ave. ','West Babylon, NY 11704','New York','New York','11704','West Babylon');
INSERT INTO Address VALUES('629 John St','Freeport, NY 11520','New York','New York','11520','Freeport');
INSERT INTO Address VALUES('61 W. Squaw Creek Rd. ','Brooklyn, NY 11211','New York','New York','11211','Brooklyn');
INSERT INTO Address VALUES('60 NE. Heritage Ave. ','Ithaca, NY 14850','New York','New York','14850','Ithaca');
INSERT INTO Address VALUES('7 East Grant Street','Brooklyn, NY 11220','New York','New York','11220','Brooklyn');
INSERT INTO Address VALUES('337 Deerfield Ave.','New York, NY 10025','New York','New York','10025','New York');
INSERT INTO Address VALUES('8294 Shore Dr.','New York, NY 10034','New York','New York','10034','New York');
INSERT INTO Address VALUES('5 Annadale Court ','New York, NY 10027','New York','New York','10027','New York');

/* INSERT SCRIPT FOR Customer */
INSERT INTO Customer VALUES(10001,'Linda','Hu','191451812','hhh@outlook.com');
INSERT INTO Customer VALUES(10002,'Ben','Sun','206283732','nonono@gmail.com');
INSERT INTO Customer VALUES(10003,'Eric','Koelpin','9021102094','9021102094@gmail.com');
INSERT INTO Customer VALUES(10004,'Frida','Walker','2074023948','Walker@gmail.com');
INSERT INTO Customer VALUES(10005,'Parker','Pagac','2084739485','Parker@gmail.com');
INSERT INTO Customer VALUES(10006,'Jennell','Adolfo','4324893092','gagag@gmail.com');
INSERT INTO Customer VALUES(10007,'Carolann','Jordan','3821030284','pdfisj@gmail.com');
INSERT INTO Customer VALUES(10008,'Nada','Kub','2830304829','blbl@gmail.com');
INSERT INTO Customer VALUES(10009,'Dreama','Davis','3849289204','prprpr@gmail.com');
INSERT INTO Customer VALUES(10000,'Argentina','Emard','283919103','gejsk@gmail.com');

/* INSERT SCRIPT FOR Online_Order */
INSERT INTO [Order] VALUES(10001,'2022-10-15 08:00:00',0,'Paypal');
INSERT INTO [Order] VALUES(10001,'2022-10-16 07:30:00',0,'Paypal');
INSERT INTO [Order] VALUES(10002,'2022-10-20 18:30:00',0,'ApplePay');
INSERT INTO [Order] VALUES(10001,'2022-10-23 12:05:00',0,'ApplePay');
INSERT INTO [Order] VALUES(10003,'2022-10-23 17:10:00',0,'ApplePay');
INSERT INTO [Order] VALUES(10001,'2022-10-28 08:30:00',0,'ApplePay');
INSERT INTO [Order] VALUES(10007,'2022-10-29 09:00:00',0,'ApplePay');
INSERT INTO [Order] VALUES(10007,'2022-11-01 11:15:00',0,'Paypal');
INSERT INTO [Order] VALUES(10001,'2022-11-05 14:35:00',0,'Paypal');
INSERT INTO [Order] VALUES(10008,'2022-11-18 16:40:00',0,'Paypal');
```

Source:Database Implementation

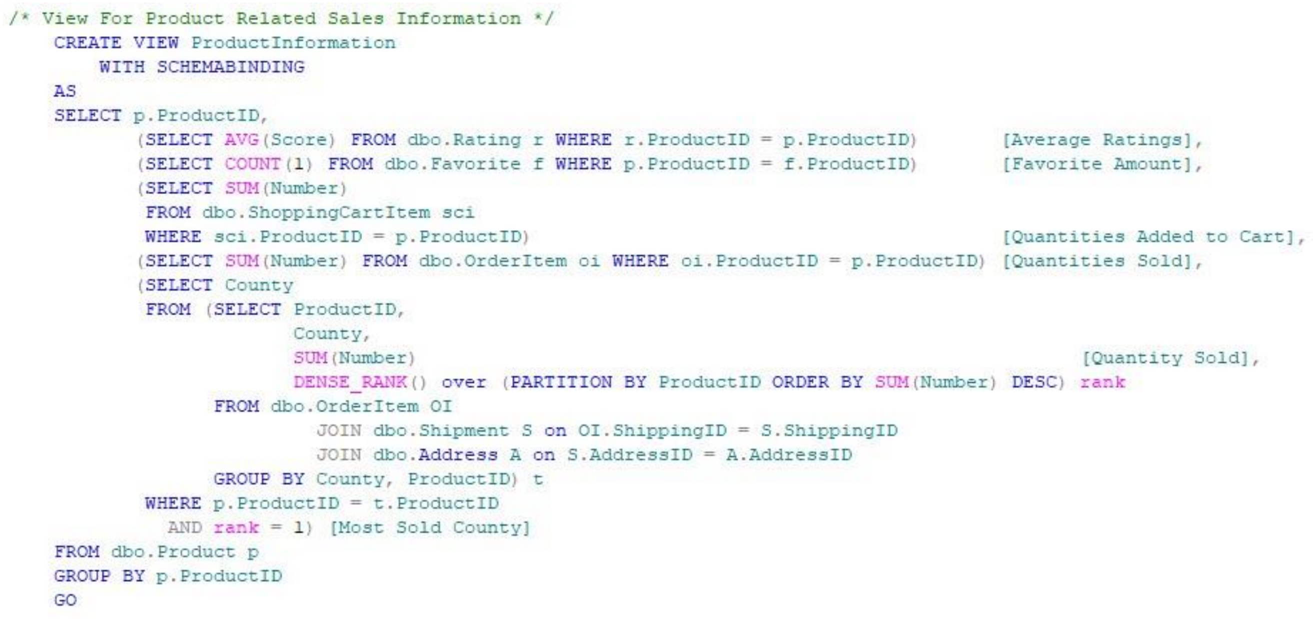
Views For County Related Sales Information



```
/* View For County Related Sales Information */
    CREATE VIEW CountySalesInformation
        WITH SCHEMABINDING
    AS
    SELECT County,
           SUM(Number)         [Total Quantities Sold],
           SUM(Number * Price) [Total Sales Amount],
           (SELECT ProductID
            FROM (SELECT County,
                         OrderItem.ProductID,
                         SUM(NUMBER)                                                [Total Quantity Sold],
                         RANK() over (PARTITION BY County ORDER BY SUM(NUMBER) DESC) rank
                  FROM dbo.OrderItem
                           JOIN dbo.Shipment S1 on dbo.OrderItem.ShippingID = S1.ShippingID
                           JOIN dbo.Product p ON dbo.OrderItem.ProductID = p.ProductID
                           JOIN dbo.Address A1 on S1.AddressID = A1.AddressID
                  GROUP BY County, OrderItem.ProductID) t
            WHERE t.County = A2.County
              AND rank = 1)    [Best Sold Product]
    FROM dbo.OrderItem
             JOIN dbo.Shipment S2 on dbo.OrderItem.ShippingID = S2.ShippingID
             JOIN dbo.Product p ON OrderItem.ProductID = p.ProductID
             JOIN dbo.Address A2 on S2.AddressID = A2.AddressID
    GROUP BY County
    GO
```

Source:Database Implementation

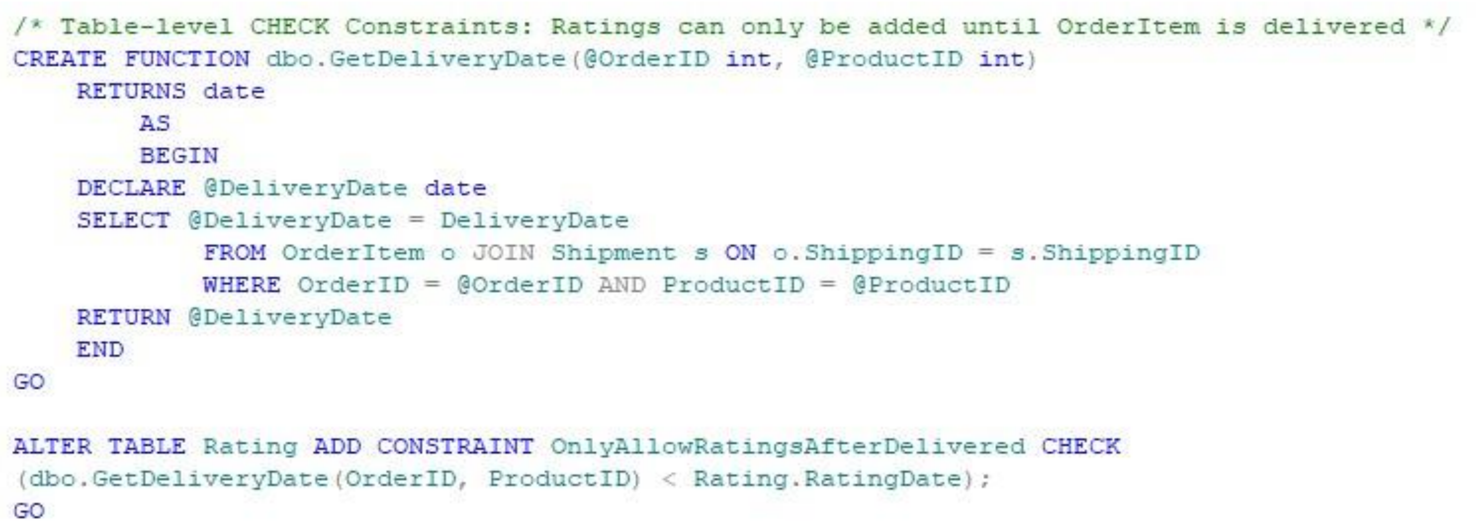
View For Product Related Sales Information



```sql
/* View For Product Related Sales Information */
    CREATE VIEW ProductInformation
        WITH SCHEMABINDING
    AS
    SELECT p.ProductID,
           (SELECT AVG(Score) FROM dbo.Rating r WHERE r.ProductID = p.ProductID)       [Average Ratings],
           (SELECT COUNT(1) FROM dbo.Favorite f WHERE p.ProductID = f.ProductID)        [Favorite Amount],
           (SELECT SUM(Number)
            FROM dbo.ShoppingCartItem sci
            WHERE sci.ProductID = p.ProductID)                                          [Quantities Added to Cart],
           (SELECT SUM(Number) FROM dbo.OrderItem oi WHERE oi.ProductID = p.ProductID) [Quantities Sold],
           (SELECT County
            FROM (SELECT ProductID,
                         County,
                         SUM(Number)                                                          [Quantity Sold],
                         DENSE_RANK() over (PARTITION BY ProductID ORDER BY SUM(Number) DESC) rank
                  FROM dbo.OrderItem OI
                           JOIN dbo.Shipment S on OI.ShippingID = S.ShippingID
                           JOIN dbo.Address A on S.AddressID = A.AddressID
                  GROUP BY County, ProductID) t
            WHERE p.ProductID = t.ProductID
              AND rank = 1) [Most Sold County]
    FROM dbo.Product p
    GROUP BY p.ProductID
    GO
```

Source: Database Implementation

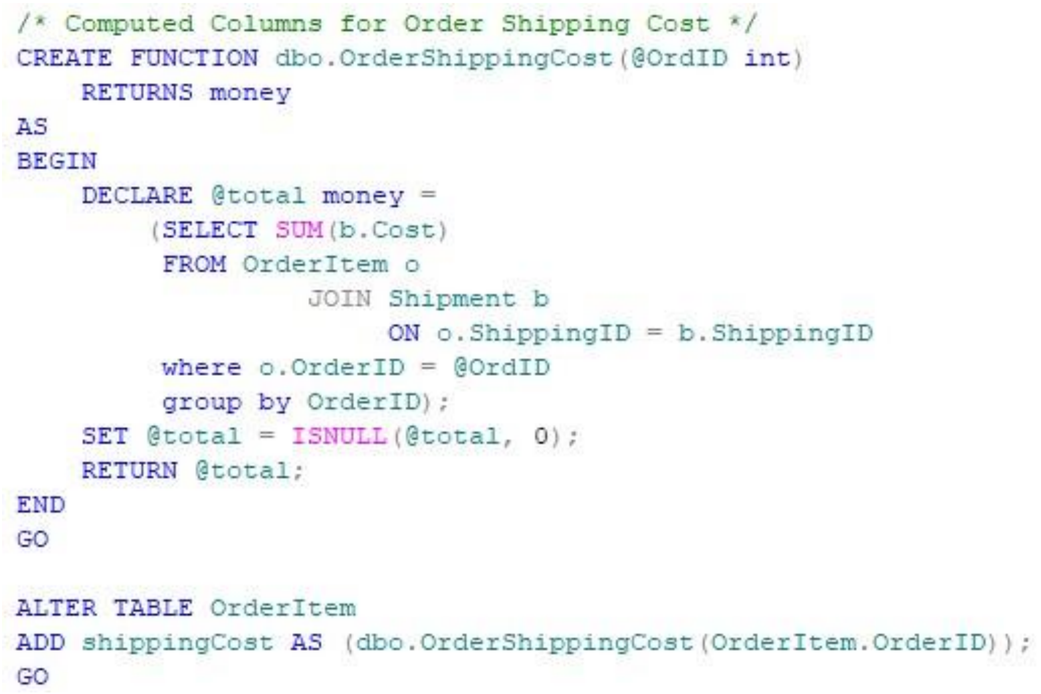
Table-level CHECK Constraints



```
/* Table-level CHECK Constraints: Ratings can only be added until OrderItem is delivered */
CREATE FUNCTION dbo.GetDeliveryDate(@OrderID int, @ProductID int)
    RETURNS date
        AS
        BEGIN
    DECLARE @DeliveryDate date
    SELECT @DeliveryDate = DeliveryDate
            FROM OrderItem o JOIN Shipment s ON o.ShippingID = s.ShippingID
            WHERE OrderID = @OrderID AND ProductID = @ProductID
    RETURN @DeliveryDate
    END
GO

ALTER TABLE Rating ADD CONSTRAINT OnlyAllowRatingsAfterDelivered CHECK
(dbo.GetDeliveryDate(OrderID, ProductID) < Rating.RatingDate);
GO
```

Source: Database Implementation

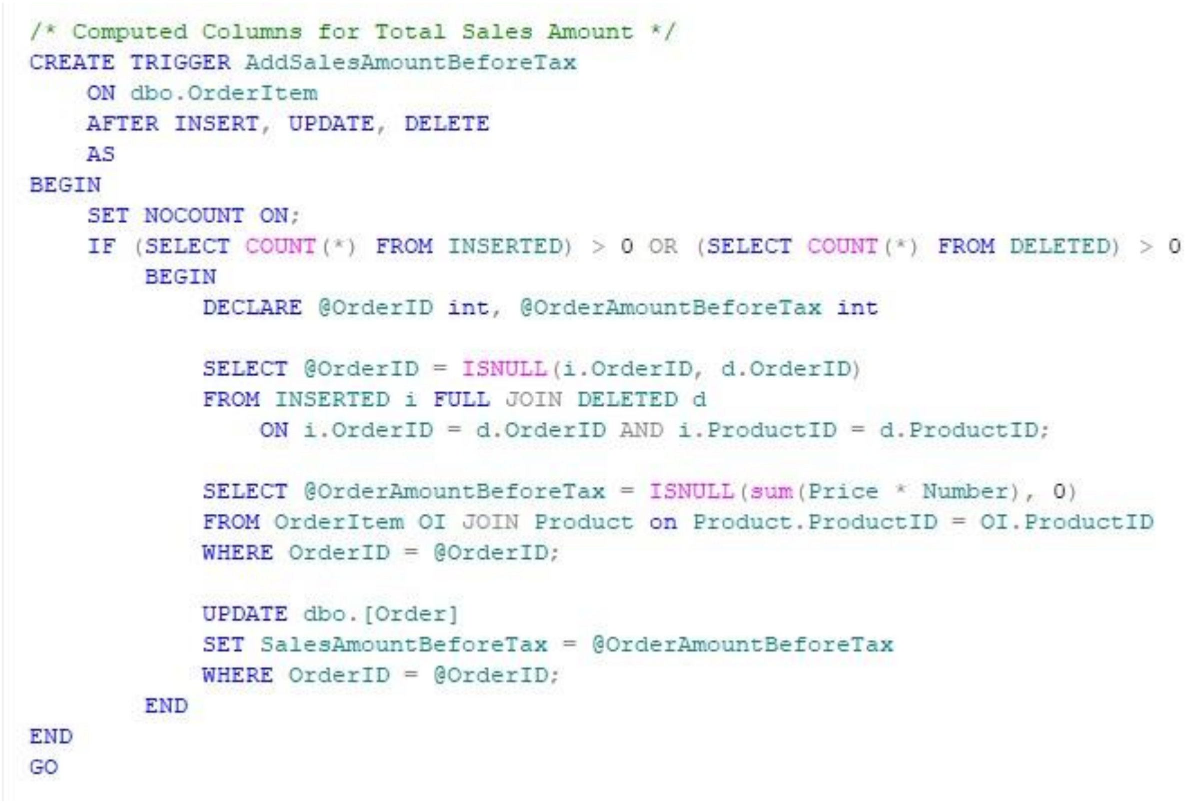
Computed Columns for Order Shipping Cost



```
/* Computed Columns for Order Shipping Cost */
CREATE FUNCTION dbo.OrderShippingCost(@OrdID int)
    RETURNS money
AS
BEGIN
    DECLARE @total money =
        (SELECT SUM(b.Cost)
         FROM OrderItem o
                  JOIN Shipment b
                       ON o.ShippingID = b.ShippingID
         where o.OrderID = @OrdID
         group by OrderID);
    SET @total = ISNULL(@total, 0);
    RETURN @total;
END
GO

ALTER TABLE OrderItem
ADD shippingCost AS (dbo.OrderShippingCost(OrderItem.OrderID));
GO
```

Source: Database Implementation

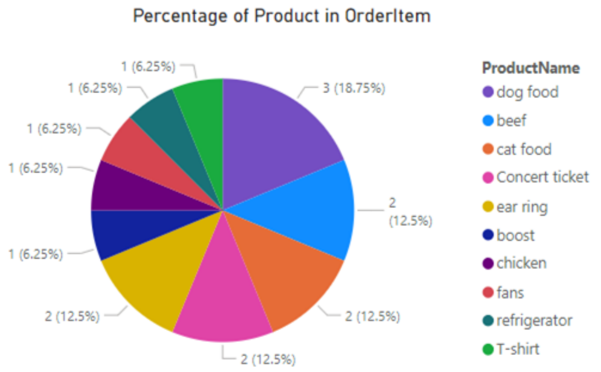
Computed Columns for Total Sales Amount

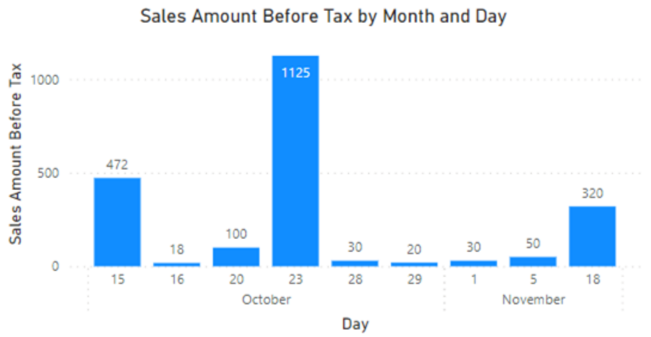


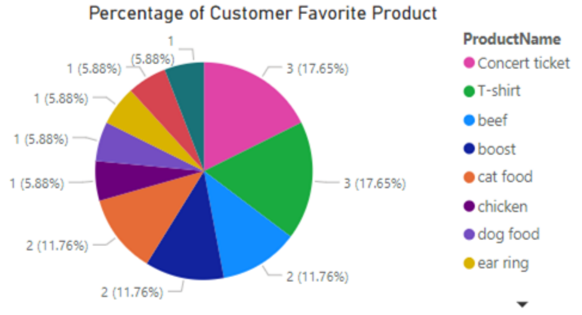
```
/* Computed Columns for Total Sales Amount */
CREATE TRIGGER AddSalesAmountBeforeTax
    ON dbo.OrderItem
    AFTER INSERT, UPDATE, DELETE
    AS
BEGIN
    SET NOCOUNT ON;
    IF (SELECT COUNT(*) FROM INSERTED) > 0 OR (SELECT COUNT(*) FROM DELETED) > 0
        BEGIN
            DECLARE @OrderID int, @OrderAmountBeforeTax int

            SELECT @OrderID = ISNULL(i.OrderID, d.OrderID)
            FROM INSERTED i FULL JOIN DELETED d
                ON i.OrderID = d.OrderID AND i.ProductID = d.ProductID;

            SELECT @OrderAmountBeforeTax = ISNULL(sum(Price * Number), 0)
            FROM OrderItem OI JOIN Product on Product.ProductID = OI.ProductID
            WHERE OrderID = @OrderID;

            UPDATE dbo.[Order]
            SET SalesAmountBeforeTax = @OrderAmountBeforeTax
            WHERE OrderID = @OrderID;
        END
END
GO
```

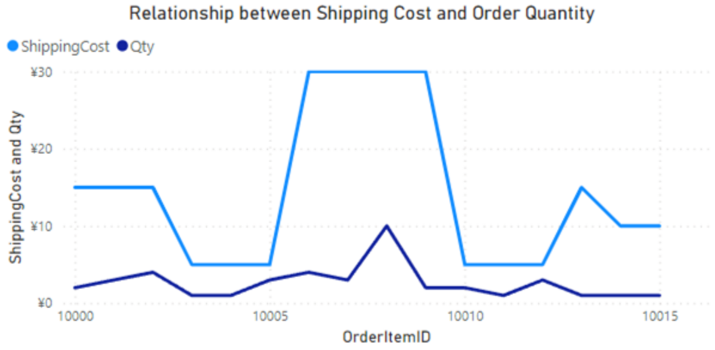
Source: Database Implementation

Visualization by Power BI









Source: Power BI

Self-Assessment
database market attractiveness/ability to win

MARKET ATTRACTIVENESS

Criterion	Comment	Assess-ment
Comprehensive	• Existing 13 tables reflecting all components involving in shopping.	◕
Business Rules	• Our database can meet all basic online shopping business rules.	◕
User Friendly	• Currently easy for salesman to know more about their customers.	◕
Reflect Customers' Preference	• available to analyse customers' data for salesman	◕
Customer Friendly	• Customers can feel free to make ratings	◕

ABILITY TO WIN

Criterion	Comment	Assess-ment
Big Volumes Of data	• This database can serve as market leader for online shopping industry	●
Cost-Saving Design.	• Help storekeepers find competitive shipping cost	●
Customer Data Analysis	• Large customer base • Strong regional presence	●
	• Strong product related analysis	◕

Low				High
○	◔	◑	◕	●

Our High-level Database Design
Summary

EXAMPLE AUTOMOTIVE SHOPPING INDUSTRY

CREATE TABLE

- In this phase , we are creating table structures for all the major components related to online shopping management.

INSERT DATA

- Inserting data into address,customer,order,store,category,product,shipment,orderitem, subscription,shoppingcart,shoppingcartitem,favorite,rating.

MAINTAIN RECORD

- After ordering an item, a record will be created for maintaining history of Customers in Store.

RATINGS

- Use table –level check constraints to make sure Ratings can only be added until order Item is delivered

SALES AMOUNT

- Use a Trigger to computed columns for total sales amount

Thank you for listening!