CSE 581
Introduction to Database Management System

Project 2
Online-Shopping Database Design

Leo Wang

Dr. Ehat Ercanli

December 5, 2019

Contents

Abstract

In the online shopping world, it's all about promoting the products to target customers and fulfill the order in efficient way fast. The database is the center-piece of the operation and the flexibility is the key feature of the system. Besides the desired features, the database has to ensure the data accuracy and maintenance.

This database is designed to provide the flexibility required for the business. It's designed after analysis of the following major online shopping business operations : (1) The order-handling function which takes an order, return or exchange requests and create correct work orders. (2) The Warehouse function who maintain the inventory data of various warehouse, make shipment and receive returns according to the work orders. (3) Accounting function collect closed orders and achieve all data from the database to offline database. (4) Marketing function make production definition, manage customers' shopping wishlists and reviews. The financial transaction is not included in this project and would be a neighboring schema or separate database. The project provides the database backend processing and the shopping features will be done separately in frontend GUI program.

The database is created using SQL Server on Windows PC. it meets the 3NF principles and balanced the design between the normalization and denormalization. Database features, integrity, security, database role & permission control are also implemented. The design is presented in this report with detail explanation and test cases for various scenarios.

1. Introduction

This project is to design the database used for a new online shopping company. The new company is completing with the established giant Amazon. Since most of us are Amazon customers and we know how Amazon provide the on-line shopping to customers. This is an ideal case for the project about a database creation. The following are the key items to address in this project:

(1) Flexibility of business to be enable by this database:

Based on the project suggested informations

(1) Each customer has multiple billing and shipping addresses.
(2) Although there is only one shipping address used per order, the items could be shipped from different warehouses based on the availability and shipping cost.
(3) Suppliers have multiple products in various warehouses.

(2) The database has to enable different work functions to work in their areas:

(1) The order-handling function which takes an order, return or exchange requests and create correct work orders.
(2) The Warehouse function who maintain the inventory data of various warehouse, make shipment and receive returns according to the work orders.
(3) Accounting function collect closed orders and achieve all data from the database to offline database.
(4) Marketing function make production definition, manage customers' shopping wishlists and reviews.

(3) The database to be integrated with other functions:

(1) The financial transaction is not included in this project and would be a neighboring schema or separate database.
(2) The project provides the database backend processing and the shopping features will be done separately in frontend GUI program.
(3) Server IT team can use add member to the the database roles and permission established in this project

The following is the summary of what has been done in this project:

(1) The business requirement is analyzed, database elements are listed and real world entities are mapped into tables.

(2) Different transaction scenarios are analyzed :

- Handling of order, return or exchange
- Maintaining inventory data, making shipment, and receiving returns.
- Collect and archive closed orders from the live to offline database
- Marketing function to define products and promote sales, to manage customers' shopping wishlists and reviews.
- IT to create server login and user for different peoples and add the membership to different database roles

(3) Seventeen tables created and populated with sample data. Relationships between tables are defined. Reference from foreign keys to primary keys were identified, constrained in column attribute or table level.

(4) The E/R diagram is finished with iterations of normalizations and denormalizations.

(5) Some reports are generated to demonstrate the features of this database using views, stored procedures, functions, constrains, triggers

(6) Five database roles are defined : OrderEntryRole, WarehouseRole, AccountingRole, MarketingRole, ITRole. The database permissions assigned to each roles based on their work function.

2. Design Considerations and Choices

The following are the different considerations and the solutions used to address these considerations:

(1) Each customer has multiple billing and shipping addresses.

This feature is very desired for customers to choose which credit card to use and which address to receive the products. To implement that, customer information is splitter into three tables : Customers, BillingAddress and ShippingAddress tables.

(2) Although there is only one shipping address used per order, the items could be shipped from different warehouses based on the availability and shipping cost.

To implement this flexibility, shipping address information is stored in Order table and each order line items ship from their warehouse locations. The database provides the Zip code information to make decision of which warehouse to ship each specific order line items.

(3) Suppliers have multiple products in various warehouses.

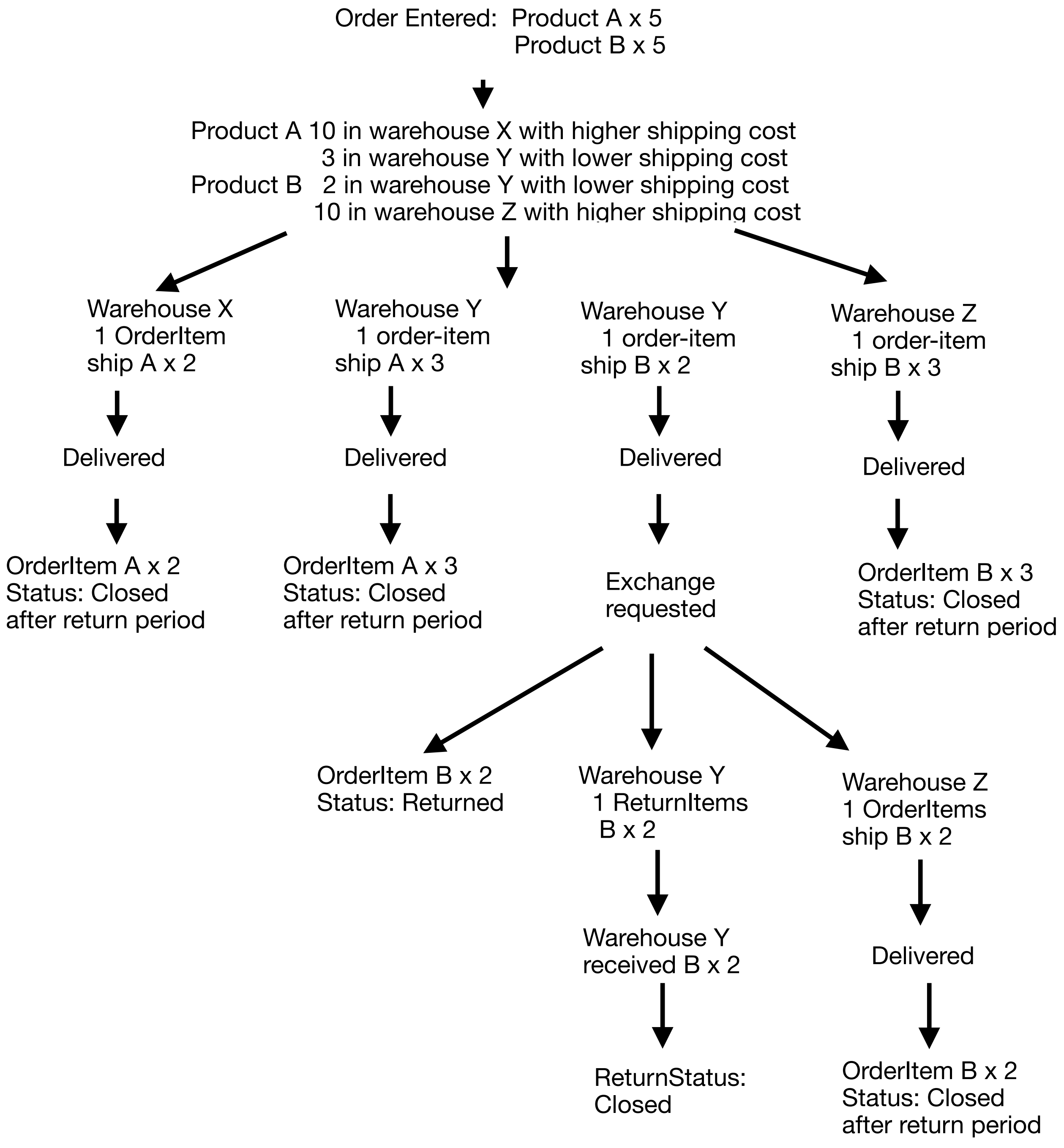
To implement this feature, the Suppliers and Warehouses table has one to many relationship with Inventories table.

(4) Each order contains multiple line items of different quantities. Each one can be accepted by customer, rejected and returned it, or reject and request for replacement (exchange).

Since the solution for consideration 'multiple items in one order', the collection of data for each order should archived together.

To address the consideration (4) and (5), each order will be handled in one "order" and multiple 'order items' and multiple 'return items' . The details is explained in the following diagram. The primary key OrderID in 'Orders' table and foreign key 'OrderID' multiple rows in 'OrderItems' and 'ReturnItems' are used to identify (JOIN) the order record together.

In order to offer the flexibility of this database can provide, the typical transition flow is described using following diagram



To explain the diagram diagram above, The following is details of the exampled case. In this example, a customer placed one order and the database created 7 rows in 3 tables and has following items:

Orders :

Row for A x 5 B x 2

OrderItems :

Row A x 2 warehouse X (Order Created -> .. -> Closed)
Row A x 3 warehouse Y (Order Created -> .. -> Closed)
Row A x 2 warehouse Y (Order Created -> .. -> Returned)
Row A x 3 warehouse Z. (Order Created -> .. -> Closed)
Row A x 2 warehouse Z (Order Created -> .. -> Closed)

ReturnItems :

Row A x 2 warehouse Y (Return Created -> .. -> Closed)

Normally closed case:

Order Created > Delivered > Closed

If one item is returned:

Order Created > Delivered > Returned
Return Created > Return Received > Closed

If one item is exchanged:

Order Created > > Delivered > Returned
Return Created > Return Received > Closed
Order Created > Delivered > Closed

The inventory data in the InventoryItems table is also updated accordingly:

InStockQuantity (X, A)	3 -> 0
InStockQuantity (Y, A)	10 -> 8
InStockQuantity (Y, B)	2 -> 0 -> 2
InStockQuantity (Z, B)	10 -> 7 -> 10

(5) The database has to link to different functions of the company to finish the task. The access permission of the database for different groups are different.

To address this security control requirement, five 'Application Roles' are defined. The database permissions assigned to each roles based on their work function.

(1) OrderEntryRole : for sales associates to create (INSERT), access (SELECT), and update (UPDATE) orders. The main permission focus is the INSERT, UPDATE to 'order' rows

(2) WarehouseRole : for warehouse associates to maintain inventory data, make shipments, and receive returns. The main permission focus is the UPDATE to 'order' rows and 'inventory' rows

(3) AccountingRole : for accounting associates to archive closed orders. The main permission focus is the DELETE to 'order' rows

(4) MarketingRole : for marketing team who defines products, manage product review and customer wishlist. The main permission focus is the INSERT, UPDATE to 'product category, product, review, wishlist' related tables

(5) IT Role : This Role has most of permission and expected to serve the database update.

3. E/R Diagram

The E/R diagram is on the next page. There are seventeen tables created. The list of tables and their relationships are shown as following:

Order related tables:

WishLists : Customer’s wishlist and shopping cart
Orders :
Shippers :
OrderStatus :
OrderItems :
ReturnStatus :
ReturnItems :

Customers —one to many —> WishLists
Products —one to many —> WishLists

ShippingAddress —one to many —> Orders
BillingAddress —one to many —> Orders
Customers —one to many —> Orders

OrderStatus —one to many —> OrderItems
Orders —one to many —> OrderItems
Products —one to many —> OrderItems
Warehouses —one to many —> OrderItems
Shippers —one to many —> OrderItems

ReturnStatus. —one to many —> ReturnItems
Orders —one to many —> ReturnItems
Products —one to many —> ReturnItems
Warehouses —one to many —> ReturnItems
Shippers —one to many —> ReturnItems

Customer related tables :

Customers :
BillingAddress :
ShippingAddress :

Customers —one to many —> BillingAddress
Customers —one to many —> ShippingAddress

Product related tables :

Categories :
Products :
ReviewScores :
Reviews :

Categories —one to many —> Products
Suppliers —one to many —> Products

Orders —one to many —> Reviews
Products —one to many —> Reviews
ReviewScores —one to many —> Reviews

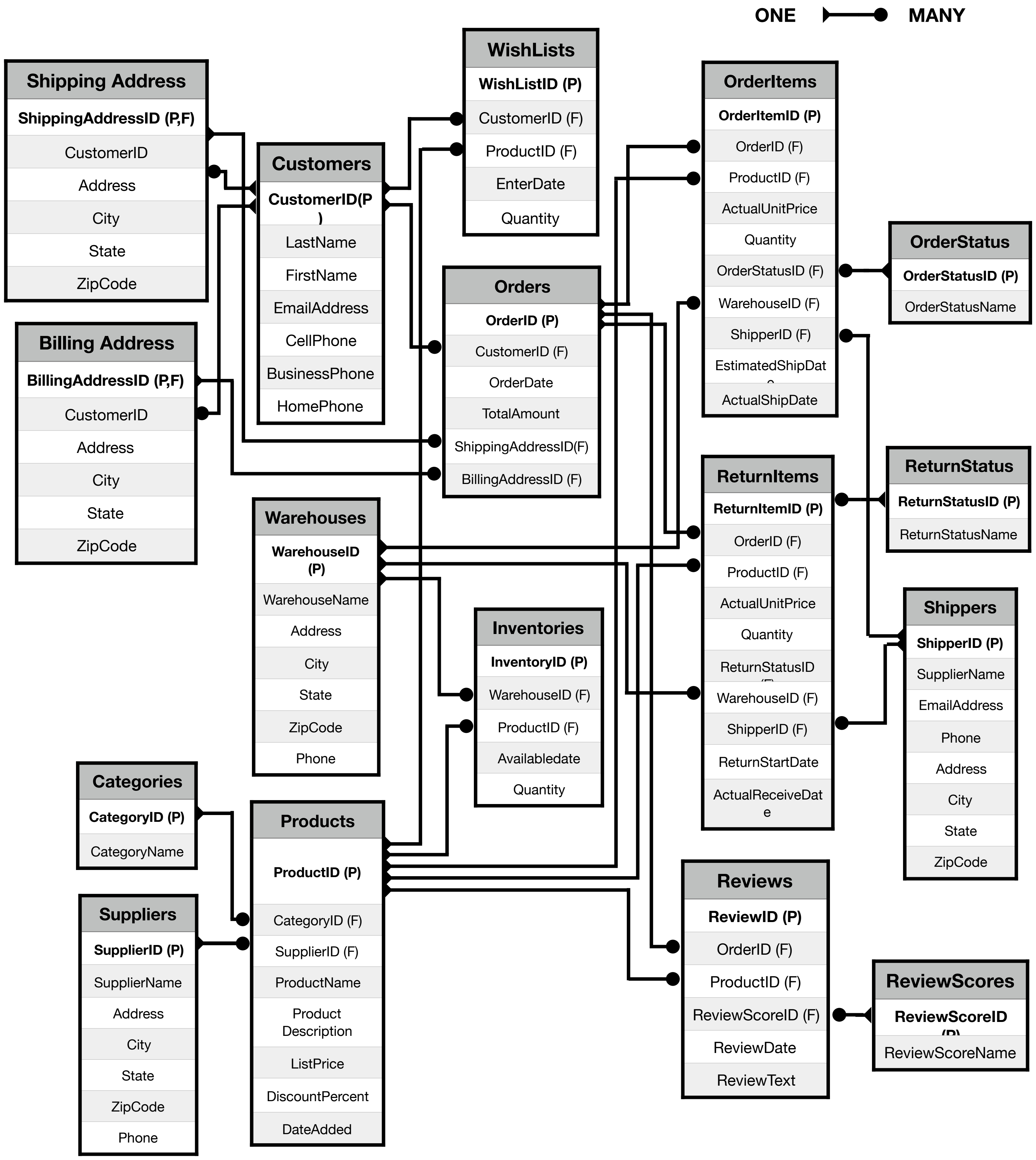
Inventory related tables :

Suppliers :
Warehouse :
InventoryItems :

Products —one to many —> InventoryItems
Products —one to many —> InventoryItems

Warehouses —one to many —> InventoryItems
Products —one to many —> InventoryItems

The relationship can be viewed in E/R diagram in next page.


ONE
MANY
Shipping Address
ShippingAddressID (P,F)
CustomerID
Address
City
State
ZipCode
Billing Address
BillingAddressID (P,F)
CustomerID
Address
City
State
ZipCode
Customers
CustomerID(P
)
LastName
FirstName
EmailAddress
CellPhone
BusinessPhone
HomePhone
WishLists
WishListID (P)
CustomerID (F)
ProductID (F)
EnterDate
Quantity
Orders
OrderID (P)
CustomerID (F)
OrderDate
TotalAmount
ShippingAddressID(F)
BillingAddressID (F)
OrderItems
OrderItemID (P)
OrderID (F)
ProductID (F)
ActualUnitPrice
Quantity
OrderStatusID (F)
WarehouseID (F)
ShipperID (F)
EstimatedShipDat
e
ActualShipDate
OrderStatus
OrderStatusID (P)
OrderStatusName
ReturnStatus
ReturnStatusID (P)
ReturnStatusName
Warehouses
WarehouseID
(P)
WarehouseName
Address
City
State
ZipCode
Phone
Inventories
InventoryID (P)
WarehouseID (F)
ProductID (F)
Availabledate
Quantity
ReturnItems
ReturnItemID (P)
OrderID (F)
ProductID (F)
ActualUnitPrice
Quantity
ReturnStatusID
(F)
WarehouseID (F)
ShipperID (F)
ReturnStartDate
ActualReceiveDat
e
Shippers
ShipperID (P)
SupplierName
EmailAddress
Phone
Address
City
State
ZipCode
Categories
CategoryID (P)
CategoryName
Suppliers
SupplierID (P)
SupplierName
Address
City
State
ZipCode
Phone
Products
ProductID (P)
CategoryID (F)
SupplierID (F)
ProductName
Product
Description
ListPrice
DiscountPercent
DateAdded
Reviews
ReviewID (P)
OrderID (F)
ProductID (F)
ReviewScoreID (F)
ReviewDate
ReviewText
ReviewScores
ReviewScoreID
(P)
ReviewScoreName

The following is the description of the database design:

(1) Each customer has a row in Customer table and multiple rows in BillingAddress and ShippingAddress tables with one-to-many relationships.

(2) Each row in Order table represents an order entry. Each order has multiple rows in OrderItems table represent the multiple shipments. If return occurs, one or more rows in ReturnItems will be added. If exchange occurs, one or more rows in OrderItems to be added to represent the shipments of exchange items.

(3) Each category of products has a row in Categories table. Each product-item to be sold as a click item has a row in Products table. They are in one-to-many relationship.

(4) Each inventory item has a row in Inventories table. Product table and Warehouse table have one-to-many relationships with Inventories table. So, one product item has multiple rows in Inventories table to represent it's available in multiple warehouse. One warehouse has multiple rows in Inventories table to represent multiple products available in each warehouse.

(5) Each wishlist item has a row in WishLists table. Customer table and Product table have one-to-many relationships with WishList table. The WishList table also serve as shopping cart to make order.

(6) The foreign key table REFERENCE to primary key table can use the ON UPDATE or ON DELETE to handle the integrity issue. Excess usage of the constraints could result unlimited loops. The usage of ON DELETE and ON UPDATE on order-related tables will be determined at frontend program development stage. This database implement the ON UPDATE and ON DELETE on very stable tables which do not expect update. For example,

```
ALTER TABLE Products WITH NOCHECK
ADD CONSTRAINT FK_Products_Categories
        FOREIGN KEY(CategoryID) REFERENCES Categories (CategoryID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
```

(7) Normalize your design into 3rd Normal Form.

(1) First (1NF) : The value stored in each cell must be scalar value
All cells must have scalar value only. Many descriptive cells

are using pre-defined ID in integer to use the pre-defined descriptions such as

ReviewScoreID	for	ReviewScoreName
CategoryID	for	CategoryName
OrderStatusID	for	OrderStatusName
ReturnStatusID	for	ReturnStatusName

For all columns, the review concluded that they pass 1NF.

(2) Second(2NF): Every non-key column must depends on the entire primary key
For all the non-key columns of all tables, I have reviewed and make sure they are depends on the whole primary keys. Therefore this database is design into 2NF

(3) Third (3NF) : Every non-key column must depend ONLY on the primary key

All non-key columns are checked and confirmed that they depend on entire primary key and depends ONLY on primary key. Therefore this database is design into 3NF.

(8) The database role is assigned based on the focus of the duty. The source code of the database permission and role creation is listed in this report.

(1) OrderEntryRole : for sales associates to create (INSERT), access (SELECT), and update (UPDATE) orders. The main permission focus is the INSERT, UPDATE to 'order' rows

(2) WarehouseRole : for warehouse associates to maintain inventory data, make shipments, and receive returns. The main permission focus is the UPDATE to 'order' rows and 'inventory' rows

(3) AccountingRole : for accounting associates to archive closed orders. The main permission focus is the DELETE to 'order' rows

(4) MarketingRole : for marketing team who defines products, manage product review and customer wishlist. The main permission focus is the INSERT, UPDATE to 'product category, product, review, wishlist' related tables

(5) IT Role : This Role has most of permission and expected to serve the database update.

(9) Application Role : The database will be integrated with frontend processing program. The Application Role is defined.

```
CREATE APPLICATION ROLE MyOnlineApplicationRole
        WITH PASSWORD = 'passwordword',
        DEFAULT_SCHEMA = dbo;

GRANT SELECT, REFERENCES ON Products TO MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON WishLists TO
        MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON Reviews
        TO MyOnlineApplicationRole;
GRANT SELECT, UPDATE, REFERENCES ON Customers
        TO MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON BillingAddress
        TO MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON ShippingAddress
        TO MyOnlineApplicationRole;
```

(10) The server login and database user creation is considered system admin work to be determined by the server administrators.

```
-- Server login and database user to 'ADD MEMBER' by IT
—CREATE LOGIN LOGIN001 FROM WINDOWS WITH DEFAULT_DATABASE = MyOnline;
—CREATE USER MyOnlineEmployee001 WITH DEFAULT_SCHEMA = dbo;
—ALTER ROLE OrderEntry ADD MEMBER MyOnlinEmployee001;
```

4. Implement Source Code and Design Result

There are several source code file created in this project. The source code listed first, then the execution results will be included.

(A) Create the database and table

(B) Script to fill data into tables for initial testing and restore the Foreign Keys after the data fill.

(C) Create the permission and database role

(D) Views, Functions, Stored procedures and reports

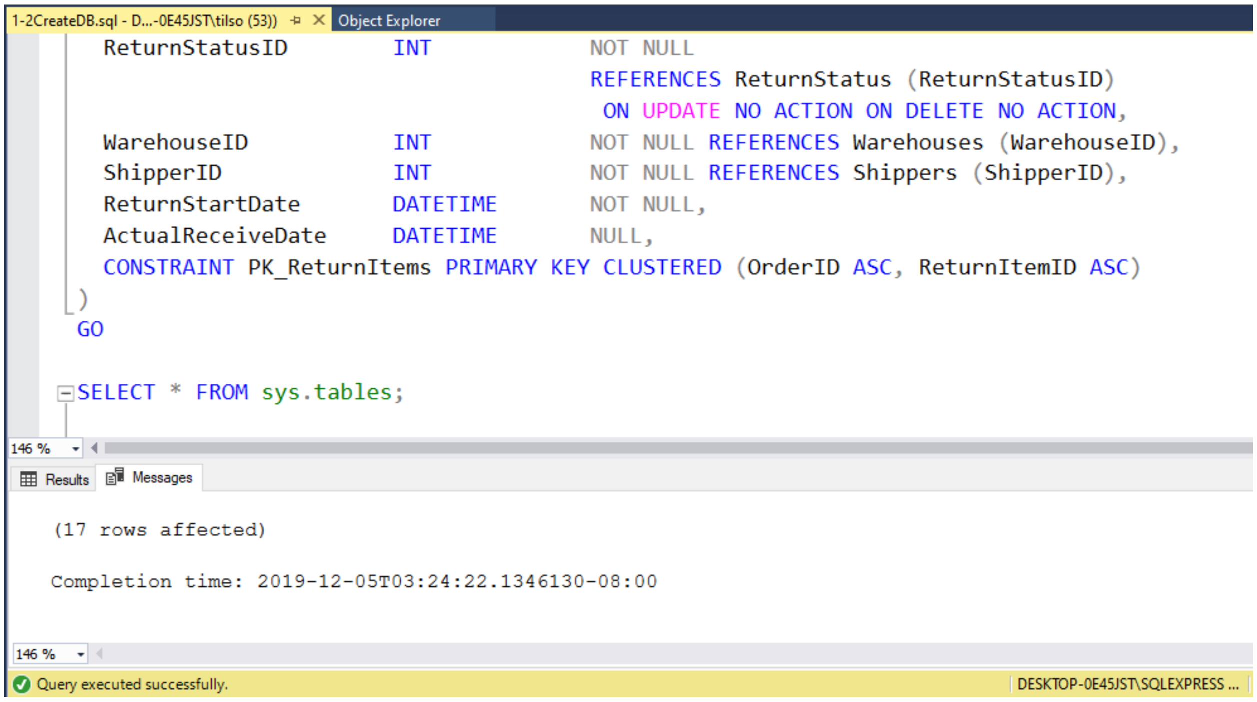
(E) Index Creation for performance Improvement

(A) Create the database and table.

```
-- This is the first step of the database creation: The creation of DATABASE and TABLE objects.
-- 17 tables are created in the order of PRIMARY KEY TABLE first and followed by
-- FOREIGN KEY TABLE.
--
--  PRIMARY KEY TABLE:     ReviewScores OrderStatus         ReturnStatus
--                         Warehouse          Shippers            Suppliers
--                         Catagories         Customers
-- Stable FOREIGN KEY TABLE:
--                         BillingAddress     ShoppingAddress     Products
--  Most active FOREIGN KEY TABLE:
--                         InventoryItems     WishLists     Reviews
--                         Orders             OrderItems    ReturnItems

USE master;
GO

-- create the database
IF DB_ID('MyOnline') IS NOT NULL
        DROP DATABASE MyOnline;
GO

CREATE DATABASE MyOnline;
GO

USE MyOnline;

-- create tables for the database

CREATE TABLE Customers (
  CustomerID          INT                   PRIMARY KEY   IDENTITY,
  FirstName           VARCHAR(20)           NOT NULL,
  LastName            VARCHAR(20)           NOT NULL,
  EmailAddress        VARCHAR(100)          NULL,
  CellPhone           VARCHAR(20)           NULL,
  BusinessPhone       VARCHAR(20)           NULL,
  HomePhone           VARCHAR(20)           NULL
);

CREATE TABLE BillingAddress (
  BillingAddressID    INT             PRIMARY KEY   IDENTITY,
  CustomerID          INT             NOT NULL REFERENCES Customers (CustomerID),
  Address             VARCHAR(60) NOT NULL,
  City                VARCHAR(20) NOT NULL,
  State               VARCHAR(2)   NOT NULL,
  ZipCode             VARCHAR(10) NOT NULL
);

CREATE TABLE ShippingAddress (
  ShippingAddressID  INT          PRIMARY KEY   IDENTITY,
  CustomerID          INT          NOT NULL REFERENCES Customers (CustomerID),
  Address             VARCHAR(60)    NOT NULL,
  City             VARCHAR(20)    NOT NULL,
  State            VARCHAR(2)     NOT NULL,
  ZipCode          VARCHAR(10)    NOT NULL
);

CREATE TABLE Categories (
  CategoryID       INT                      PRIMARY KEY   IDENTITY,
  CategoryName  VARCHAR(50)         NOT NULL
);

CREATE TABLE Suppliers (
  SupplierID             INT                 PRIMARY KEY   IDENTITY,
  SupplierName         VARCHAR(50)    NOT NULL,
  EmailAddress          VARCHAR(100)   NULL,
  Phone                    VARCHAR(20)     NULL,
  Address                 VARCHAR(60)     NOT NULL,
  City                      VARCHAR(40)     NOT NULL,
  State                     VARCHAR(2)       NOT NULL,
  ZipCode                 VARCHAR(10)      NOT NULL
);

CREATE TABLE Products (
  ProductID              INT                PRIMARY KEY   IDENTITY,
  CategoryID            INT                NOT NULL
                                                 REFERENCES Categories (CategoryID)
                                                 ON UPDATE NO ACTION ON DELETE NO ACTION,
  SupplierID             INT                NOT NULL REFERENCES Suppliers (SupplierID),
  ProductName          VARCHAR(40)     NOT NULL,
  ProductDescription  VARCHAR(100)   NOT NULL,
  ListPrice               MONEY          NOT NULL,
  DiscountPercent      MONEY          NOT NULL  DEFAULT 0.00,
  DateAdded             SmallDateTime      NULL
);

CREATE TABLE Warehouses (
  WarehouseID          INT                         PRIMARY KEY   IDENTITY,
  IsThirdParty          INT                         NOT NULL,
  WarehouseName      VARCHAR(50)            NOT NULL,
  EmailAddress          VARCHAR(100)           NULL,
  Phone                    VARCHAR(20)            NULL,
  Address                 VARCHAR(60)            NOT NULL,
  City                      VARCHAR(40)            NOT NULL,
  State                     VARCHAR(2)              NOT NULL,
  ZipCode                 VARCHAR(10)            NOT NULL
);

CREATE TABLE InventoryItems (
  InventoryItemID       INT                PRIMARY KEY  IDENTITY,
  ProductID              INT                NOT NULL REFERENCES Products (ProductID),
  WarehouseID          INT                NOT NULL REFERENCES Warehouses (WarehouseID),
  AvailableDate         SmallDateTime       NOT NULL,
  Quantity                INT                NOT NULL   CHECK (Quantity >=0)
);
```

```
CREATE TABLE WishLists (
  WishListID            INT           PRIMARY KEY  IDENTITY,
  CustomerID            INT           NOT NULL REFERENCES Customers (CustomerID),
  ProductID             INT           NOT NULL REFERENCES Products (ProductID),
  EnterDate             SmallDateTime      NOT NULL,
  Quantity              INT           NOT NULL   CHECK (Quantity >=0)
);

CREATE TABLE Orders (
  OrderID               INT               PRIMARY KEY  IDENTITY,
  CustomerID            INT               NOT NULL REFERENCES Customers (CustomerID),
  OrderDate             SmallDateTime     NOT NULL,
  TotalAmount           MONEY            NOT NULL,
  ShippingAddressID  INT               NOT NULL
                                          REFERENCES  ShippingAddress (ShippingAddressID),
  BillingAddressID      INT               NOT NULL
                                          REFERENCES  BillingAddress (BillingAddressID)
);

CREATE TABLE ReviewScores (
  ReviewScoreID                  INT                         PRIMARY KEY   IDENTITY,
  ReviewScoreName              VARCHAR(20)             NOT NULL
);

CREATE TABLE Reviews (
  ReviewID              INT               PRIMARY KEY  IDENTITY,
  ProductID             INT               NOT NULL REFERENCES Products (ProductID),
  OrderID               INT               NOT NULL REFERENCES Customers (CustomerID),
  ReviewDate            SmallDateTime       NOT NULL,
  ReviewScoreID         INT               NOT NULL
                                          REFERENCES ReviewScores (ReviewScoreID)
                                           ON UPDATE NO ACTION ON DELETE NO ACTION,
  ReviewText   VARCHAR(500)           NULL
);

CREATE TABLE OrderStatus (
  OrderStatusID                  INT                         PRIMARY KEY   IDENTITY,
  OrderStatusName              VARCHAR(20)             NOT NULL
);

CREATE TABLE Shippers (
  ShipperID                      INT              PRIMARY KEY   IDENTITY,
  ShipperName           VARCHAR(50)          NOT NULL,
  EmailAddress          VARCHAR(100)         NULL,
  Phone                 VARCHAR(20)          NULL,
  Address               VARCHAR(60)          NOT NULL,
  City                  VARCHAR(40)          NOT NULL,
  State                 VARCHAR(2)           NOT NULL,
  ZipCode               VARCHAR(10)          NOT NULL
);

CREATE TABLE OrderItems (
  OrderItemID           INT           IDENTITY,
  OrderID               INT           NOT NULL REFERENCES Orders (OrderID),
  ProductID           INT             NOT NULL REFERENCES Products (ProductID),
  ActualUnitPrice     MONEY          NOT NULL  CHECK (ActualUnitPrice >=0),
  Quantity            INT             NOT NULL  CHECK (Quantity >=0),
  OrderStatusID       INT             NOT NULL
                                      REFERENCES OrderStatus (OrderStatusID)
                                       ON UPDATE NO ACTION ON DELETE NO ACTION,
  WarehouseID         INT             NOT NULL REFERENCES Warehouses (WarehouseID),
  ShipperID           INT             NOT NULL REFERENCES Shippers (ShipperID),
  EstimatedShipDate  SmallDateTime       NOT NULL,
  ActualShipDate      SmallDateTime      NULL,
  CONSTRAINT PK_OrderItems PRIMARY KEY CLUSTERED (OrderID ASC, OrderItemID ASC)
);

CREATE TABLE ReturnStatus (
  ReturnStatusID              INT                     PRIMARY KEY   IDENTITY,
  ReturnStatusName            VARCHAR(20)             NOT NULL
);

CREATE TABLE ReturnItems (
  ReturnItemID                INT           IDENTITY,
  OrderID                     INT           NOT NULL REFERENCES Orders (OrderID),
  ProductID                   INT           NOT NULL REFERENCES Products (ProductID) ON
UPDATE CASCADE,
  ActualUnitPrice             MONEY    NOT NULL  CHECK (ActualUnitPrice >=0),
  Quantity                    INT           NOT NULL CHECK (Quantity >=0),
  ReturnStatusID              INT           NOT NULL
                                          REFERENCES ReturnStatus (ReturnStatusID)
                                          ON UPDATE NO ACTION ON DELETE NO ACTION,
  WarehouseID                 INT          NOT NULL REFERENCES Warehouses (WarehouseID),
  ShipperID                   INT           NOT NULL REFERENCES Shippers (ShipperID),
  ReturnStartDate             SmallDateTime       NOT NULL,
  ActualReceiveDate           SmallDateTime       NULL,
  CONSTRAINT PK_ReturnItems PRIMARY KEY CLUSTERED (OrderID ASC, ReturnItemID
ASC)
)
GO

SELECT * FROM sys.tables;
```

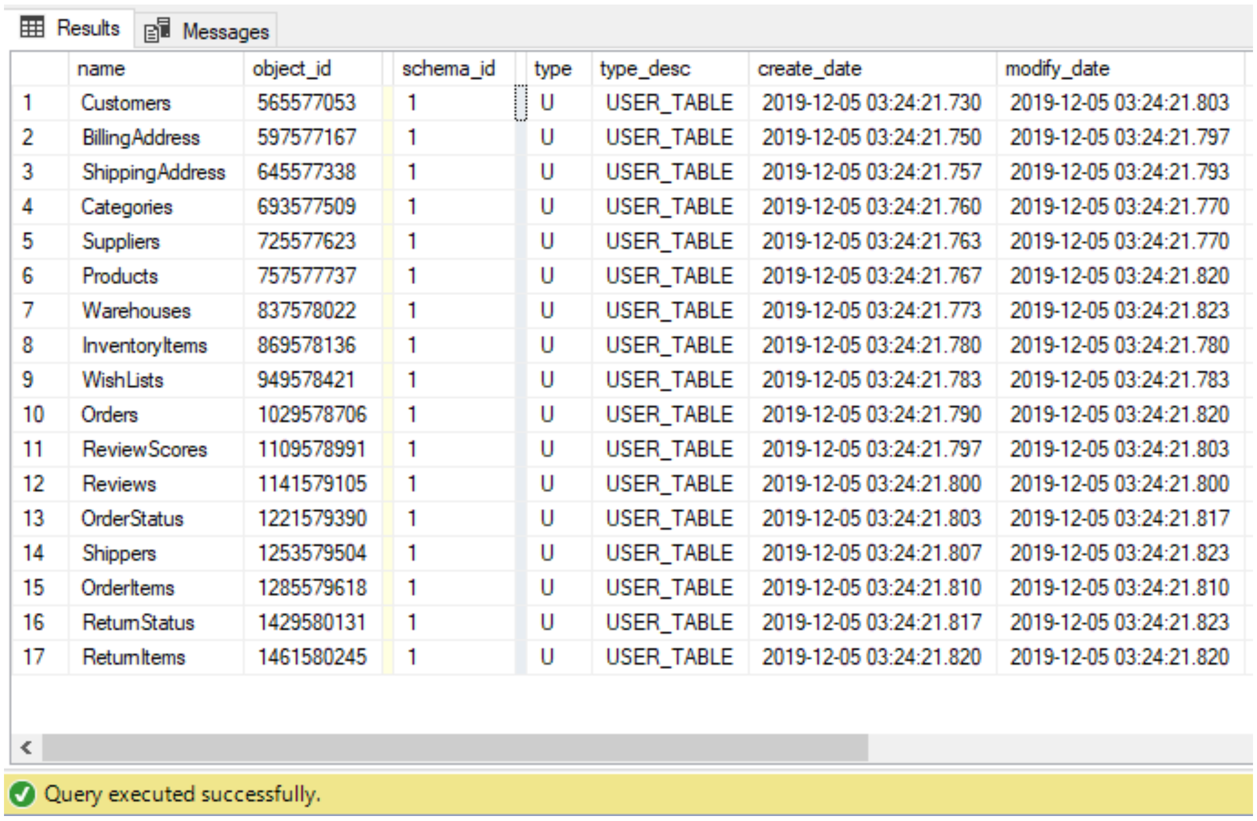
The script to create the database and tables are executed successfully. Seven tables created.


1-2CreateDB.sql - D...-0E45JST\tilso (53))
Object Explorer
146 %
Results
Messages
DESKTOP-0E45JST\SQLEXPRESS ...

```
    ReturnStatusID          INT             NOT NULL
                                            REFERENCES ReturnStatus (ReturnStatusID)
                                             ON UPDATE NO ACTION ON DELETE NO ACTION,
    WarehouseID             INT             NOT NULL REFERENCES Warehouses (WarehouseID),
    ShipperID               INT             NOT NULL REFERENCES Shippers (ShipperID),
    ReturnStartDate         DATETIME        NOT NULL,
    ActualReceiveDate       DATETIME        NULL,
    CONSTRAINT PK_ReturnItems PRIMARY KEY CLUSTERED (OrderID ASC, ReturnItemID ASC)
  )
  GO

  SELECT * FROM sys.tables;
```

```
(17 rows affected)

Completion time: 2019-12-05T03:24:22.1346130-08:00
```

Query executed successfully.



	name	object_id	schema_id	type	type_desc	create_date	modify_date
1	Customers	565577053	1	U	USER_TABLE	2019-12-05 03:24:21.730	2019-12-05 03:24:21.803
2	BillingAddress	597577167	1	U	USER_TABLE	2019-12-05 03:24:21.750	2019-12-05 03:24:21.797
3	ShippingAddress	645577338	1	U	USER_TABLE	2019-12-05 03:24:21.757	2019-12-05 03:24:21.793
4	Categories	693577509	1	U	USER_TABLE	2019-12-05 03:24:21.760	2019-12-05 03:24:21.770
5	Suppliers	725577623	1	U	USER_TABLE	2019-12-05 03:24:21.763	2019-12-05 03:24:21.770
6	Products	757577737	1	U	USER_TABLE	2019-12-05 03:24:21.767	2019-12-05 03:24:21.820
7	Warehouses	837578022	1	U	USER_TABLE	2019-12-05 03:24:21.773	2019-12-05 03:24:21.823
8	InventoryItems	869578136	1	U	USER_TABLE	2019-12-05 03:24:21.780	2019-12-05 03:24:21.780
9	WishLists	949578421	1	U	USER_TABLE	2019-12-05 03:24:21.783	2019-12-05 03:24:21.783
10	Orders	1029578706	1	U	USER_TABLE	2019-12-05 03:24:21.790	2019-12-05 03:24:21.820
11	ReviewScores	1109578991	1	U	USER_TABLE	2019-12-05 03:24:21.797	2019-12-05 03:24:21.803
12	Reviews	1141579105	1	U	USER_TABLE	2019-12-05 03:24:21.800	2019-12-05 03:24:21.800
13	OrderStatus	1221579390	1	U	USER_TABLE	2019-12-05 03:24:21.803	2019-12-05 03:24:21.817
14	Shippers	1253579504	1	U	USER_TABLE	2019-12-05 03:24:21.807	2019-12-05 03:24:21.823
15	OrderItems	1285579618	1	U	USER_TABLE	2019-12-05 03:24:21.810	2019-12-05 03:24:21.810
16	ReturnStatus	1429580131	1	U	USER_TABLE	2019-12-05 03:24:21.817	2019-12-05 03:24:21.823
17	ReturnItems	1461580245	1	U	USER_TABLE	2019-12-05 03:24:21.820	2019-12-05 03:24:21.820

Query executed successfully.

(B) Script to fill data into tables for initial testing

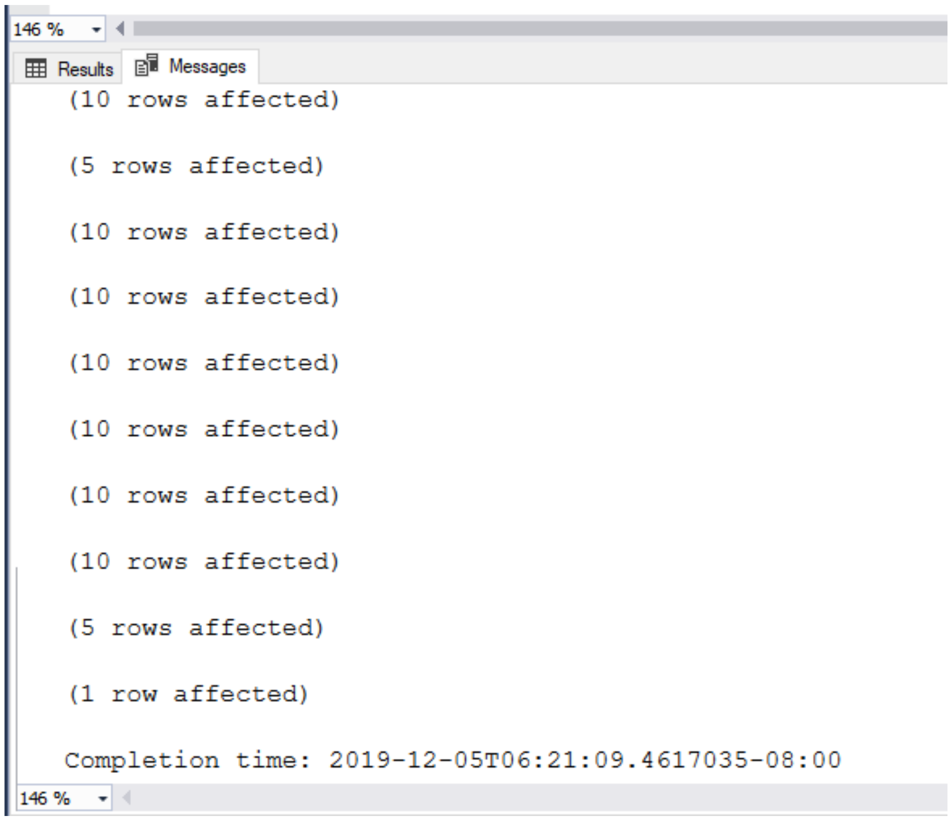
In order to fill populate data into the tables, the Foreign key constraints has to be removed. The following is the script to remove the foreign key constraints, fill-in the data, and restore the foreign key constraints.

(B.1) This is the script to populate data into the tables. In order to make the report more readable, I removed some raw data portion of the code. The fill results are shown following the source code.

```
— This is the script to populate data into tables
        USE MyOnline;

        SET IDENTITY_INSERT ReviewScores ON
        INSERT ReviewScores (ReviewScoreID, ReviewScoreName)
        VALUES
        (1,'One Stars'),
        (2,'Two Stars'),
        (3,'Three Stars'),
        (4,'Four Stars'),
        (5,'Five Stars')
        SET IDENTITY_INSERT ReviewScores OFF
        SELECT * FROM ReviewScores;

        SET IDENTITY_INSERT OrderStatus ON
        INSERT OrderStatus (OrderStatusID, OrderStatusName)
        VALUES
        (1,'Order Created'),
        (2,'Ready to Ship'),
        (3,'Order shipped'),
        (4,'On the way'),
        (5,'Final Delivery'),
        (6,'Delivered'),
        (7,'Closed'),
        (8,'Returned')
        SET IDENTITY_INSERT OrderStatus OFF

        SET IDENTITY_INSERT ReturnStatus ON
        INSERT ReturnStatus (ReturnStatusID, ReturnStatusName)
        VALUES
        (1,'Return Created'),
        (2,'Return Received'),
        (3,'Return Closed')
        SET IDENTITY_INSERT ReturnStatus OFF
        SET IDENTITY_INSERT Warehouses ON
        INSERT Warehouses (WarehouseID, IsThirdParty, WarehouseName,
                        EmailAddress, Phone, Address, City, State, ZipCode)
        VALUES
        (10, 1, ‘Charlotte Warehouse',NULL,'(704) 555-3500','2709 Water Ridge
        Parkway, Ste 500', 'Charlotte', 'NC', '28217')
        SET IDENTITY_INSERT Warehouses OFF

        SET IDENTITY_INSERT Shippers ON
        INSERT Shippers (ShipperID, ShipperName, EmailAddress, Phone, Address,
        City, State, ZipCode)
        VALUES
        (1, 'US Postal Service', 'HQ@ups.com','(800) 555-1205','PO Box 7005',
        'Madison', 'WI', '53707'),
        (2, 'Federal Express Corporation', 'Corporate@fedex.com','(800) 555-4091','P.O.
        Box 1140', 'Memphis', 'TN', '38101'),
        (3, 'US Postal Services','Postmaster@usps.gov','(559) 555-7785','1900 E
        Street', 'Fresno', 'CA', '93706')
        SET IDENTITY_INSERT Shippers OFF

        SET IDENTITY_INSERT Suppliers ON
        INSERT Suppliers (SupplierID, SupplierName, EmailAddress, Phone, Address,
        City, State, ZipCode)
        VALUES
        (10, 'Baker & Taylor Books',NULL,'(704) 555-3500','2709 Water Ridge Parkway,
        Ste 500', 'Charlotte', 'NC', '28217')
        SET IDENTITY_INSERT Suppliers OFF

        SET IDENTITY_INSERT Categories ON
        INSERT Categories (CategoryID, CategoryName)
        VALUES
        (1,'Science Books'),
        (2,'Social Books'),
        (3,'Local Interests Books'),
        (4,'Travel Books'),
        (5,'Cook Books')
        SET IDENTITY_INSERT Categories OFF

        SET IDENTITY_INSERT Customers ON
        INSERT Customers (CustomerID,FirstName, LastName, EmailAddress,
        CellPhone, BusinessPhone, HomePhone)
        VALUES
        (10,'Steve','Job','steve@via.com','(408) 555-3770','(704) 111-2222’,'(467)
         555-7777')
        SET IDENTITY_INSERT Customers OFF
        SET IDENTITY_INSERT BillingAddress ON
        INSERT BillingAddress (BillingAddressID,CustomerID,
        Address,City,State,ZipCode)
        VALUES
        (10,5,'2709 Sweet Street, Ste 500', 'Charlotte', 'NC', '28217')
        SET IDENTITY_INSERT BillingAddress OFF

        SET IDENTITY_INSERT ShippingAddress ON
        INSERT ShippingAddress
        (ShippingAddressID,CustomerID,Address,City,State,ZipCode)
        VALUES
        (10,5,'2709 Sweet Street, Ste 500', 'Charlotte', 'NC', '28217')
        SET IDENTITY_INSERT ShippingAddress OFF

        SET IDENTITY_INSERT Reviews ON;
        INSERT INTO Reviews (ReviewID, ProductID, OrderID, ReviewDate,
        ReviewScoreID, ReviewText)
        VALUES
        (10, 3, 4,'2016-07-30 13:58:35', 4,'This book was purchased as a gift for my
        son.')
        SET IDENTITY_INSERT Reviews OFF;

        SET IDENTITY_INSERT Products ON;
        INSERT INTO Products (ProductID, CategoryID, SupplierID, ProductName,
        ProductDescription, ListPrice, DiscountPercent, DateAdded)
        (10, 5,  3,'Salt Fat Acid Heat', ' Mastering the Elements of Good Cooking',
        70, 00, NULL)
        SET IDENTITY_INSERT Products OFF;

        SET IDENTITY_INSERT InventoryItems ON
        INSERT InventoryItems (InventoryItemID, ProductID, WarehouseID,
        AvailableDate, Quantity)
        VALUES
        (10, 5, 2, '2019-11-05 16:33:13', 10)
        SET IDENTITY_INSERT InventoryItems OFF

        SET IDENTITY_INSERT WishLists ON
        INSERT WishLists (WishListID, CustomerID, ProductID, EnterDate, Quantity)
        VALUES
        (10, 2, 4, '2019-12-06 19:33:13', 1)
        SET IDENTITY_INSERT WishLists OFF

        SET IDENTITY_INSERT Orders ON
        INSERT Orders (OrderID, CustomerID, OrderDate, TotalAmount,
        ShippingAddressID, BillingAddressID)
        VALUES
```

```
( 2, 1,'2019-12-06 11:33:13',  350, 2, 1)
SET IDENTITY_INSERT Orders OFF
SELECT * FROM Orders;

SET IDENTITY_INSERT OrderItems ON
INSERT OrderItems (OrderItemID, OrderID, ProductID, ActualUnitPrice, Quantity,
OrderStatusID, WarehouseID, ShipperID, EstimatedShipDate, ActualShipDate )
VALUES
( 5, 2, 1, 40, 3, 1, 3, 1, '2019-12-08 10:00:00',NULL)
SET IDENTITY_INSERT OrderItems OFF

SET IDENTITY_INSERT ReturnItems ON
INSERT ReturnItems (ReturnItemID, OrderID, ProductID, ActualUnitPrice,
Quantity, ReturnStatusID, WarehouseID, ShipperID, ReturnStartDate,
ActualReceiveDate )
VALUES
( 1, 2, 1, 40, 1, 3, 1, 2, '2019-11-08 10:00:00','2019-11-16 10:00:00')
SET IDENTITY_INSERT ReturnItems OFF
```

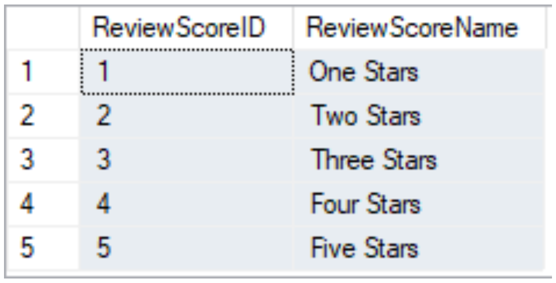
The data fill script runs successfully as indicted. The details of the data filled are indication of how these tables are intended to be used. They are in the following pages.



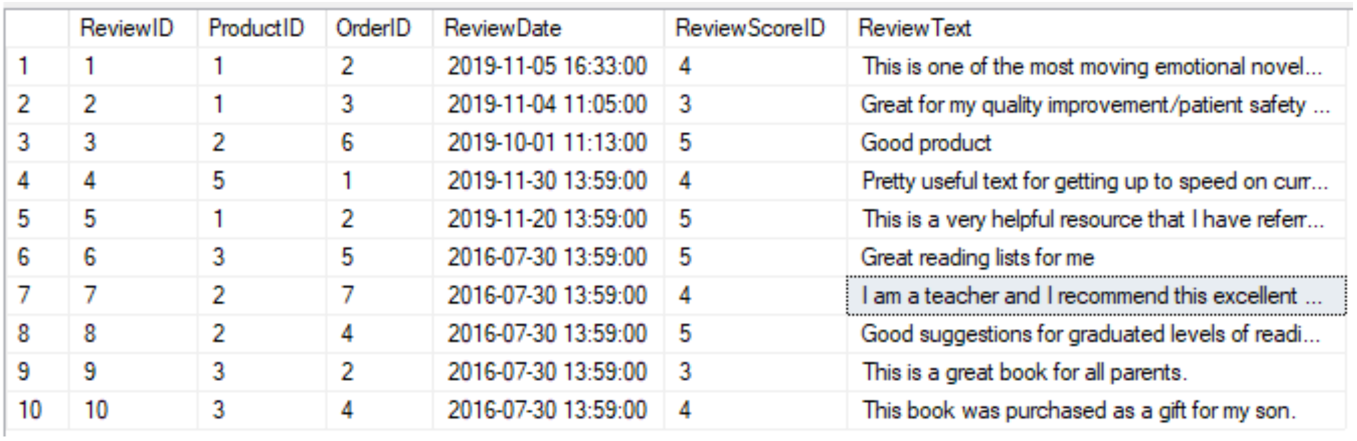
```
(10 rows affected)

(5 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)

(5 rows affected)

(1 row affected)

Completion time: 2019-12-05T06:21:09.4617035-08:00
```

This is the contents of the Reviews table.
Each reviews by a customer (reference OrderID) on a product (reference ProductID) are logged here. The ReviewScores table defines the definition of rating (1 star to 5 stars)

Orders (1 to many) ReviewScores
Products (1 to many) ReviewScores
ReviewScores (1 to many) ReviewScores



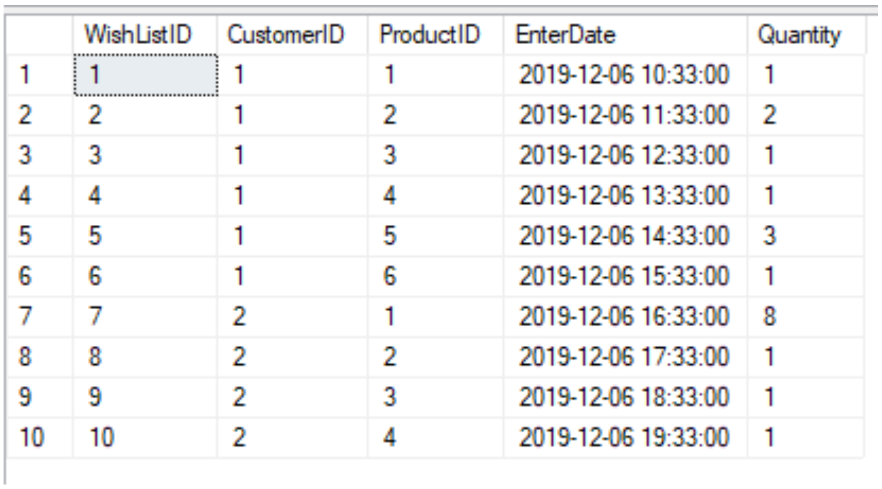
	ReviewScoreID	ReviewScoreName
1	1	One Stars
2	2	Two Stars
3	3	Three Stars
4	4	Four Stars
5	5	Five Stars



	ReviewID	ProductID	OrderID	ReviewDate	ReviewScoreID	ReviewText
1	1	1	2	2019-11-05 16:33:00	4	This is one of the most moving emotional novel...
2	2	1	3	2019-11-04 11:05:00	3	Great for my quality improvement/patient safety ...
3	3	2	6	2019-10-01 11:13:00	5	Good product
4	4	5	1	2019-11-30 13:59:00	4	Pretty useful text for getting up to speed on curr...
5	5	1	2	2019-11-20 13:59:00	5	This is a very helpful resource that I have referr...
6	6	3	5	2016-07-30 13:59:00	5	Great reading lists for me
7	7	2	7	2016-07-30 13:59:00	4	I am a teacher and I recommend this excellent ...
8	8	2	4	2016-07-30 13:59:00	5	Good suggestions for graduated levels of readi...
9	9	3	2	2016-07-30 13:59:00	3	This is a great book for all parents.
10	10	3	4	2016-07-30 13:59:00	4	This book was purchased as a gift for my son.

This is the contents of the WishLists table.
Each rWishList items (potential Shopping Cart items) by a customer (reference CustomerID) on a product (reference ProductID) are logged here.

Customers (1 to many) ReviewScores
Products (1 to many) ReviewScores

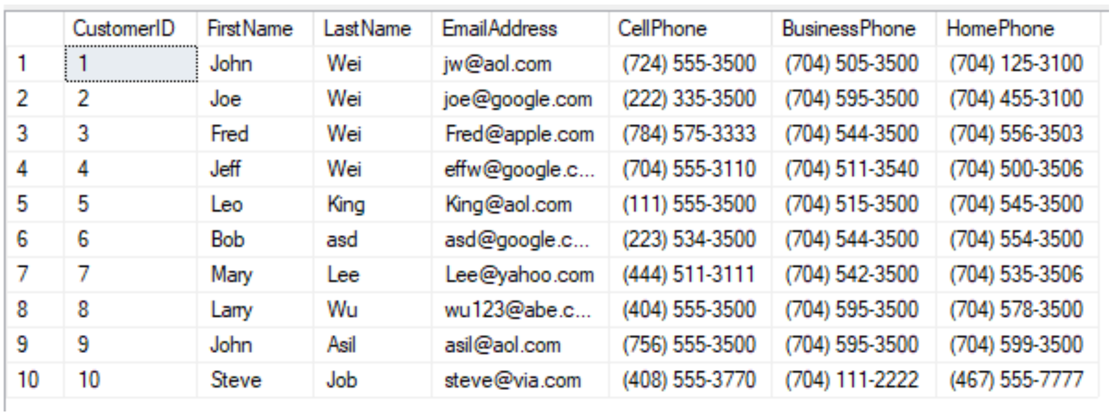


	WishListID	CustomerID	ProductID	EnterDate	Quantity
1	1	1	1	2019-12-06 10:33:00	1
2	2	1	2	2019-12-06 11:33:00	2
3	3	1	3	2019-12-06 12:33:00	1
4	4	1	4	2019-12-06 13:33:00	1
5	5	1	5	2019-12-06 14:33:00	3
6	6	1	6	2019-12-06 15:33:00	1
7	7	2	1	2019-12-06 16:33:00	8
8	8	2	2	2019-12-06 17:33:00	1
9	9	2	3	2019-12-06 18:33:00	1
10	10	2	4	2019-12-06 19:33:00	1

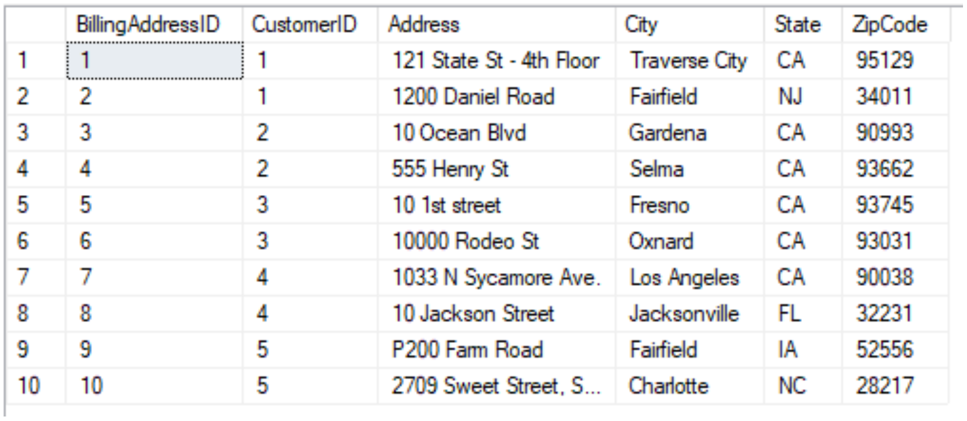
Customer information are stored in Customers, BillingAddress and Shipping Address tables.

Customers (1 to many) BillingAddress
Customers (1 to many) Shipping Address

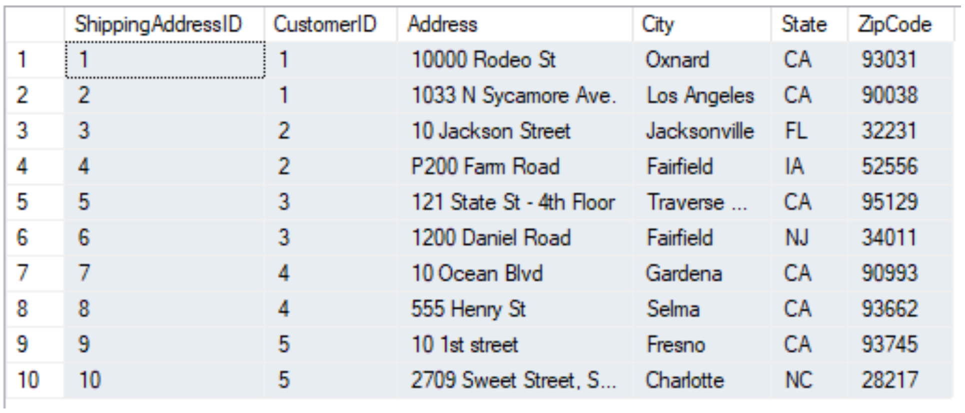
Customers, BillingAddress and Shipping Address tables :



	CustomerID	FirstName	LastName	EmailAddress	CellPhone	BusinessPhone	HomePhone
1	1	John	Wei	jw@aol.com	(724) 555-3500	(704) 505-3500	(704) 125-3100
2	2	Joe	Wei	joe@google.com	(222) 335-3500	(704) 595-3500	(704) 455-3100
3	3	Fred	Wei	Fred@apple.com	(784) 575-3333	(704) 544-3500	(704) 556-3503
4	4	Jeff	Wei	effw@google.c...	(704) 555-3110	(704) 511-3540	(704) 500-3506
5	5	Leo	King	King@aol.com	(111) 555-3500	(704) 515-3500	(704) 545-3500
6	6	Bob	asd	asd@google.c...	(223) 534-3500	(704) 544-3500	(704) 554-3500
7	7	Mary	Lee	Lee@yahoo.com	(444) 511-3111	(704) 542-3500	(704) 535-3506
8	8	Larry	Wu	wu123@abe.c...	(404) 555-3500	(704) 595-3500	(704) 578-3500
9	9	John	Asil	asil@aol.com	(756) 555-3500	(704) 595-3500	(704) 599-3500
10	10	Steve	Job	steve@via.com	(408) 555-3770	(704) 111-2222	(467) 555-7777



	BillingAddressID	CustomerID	Address	City	State	ZipCode
1	1	1	121 State St - 4th Floor	Traverse City	CA	95129
2	2	1	1200 Daniel Road	Fairfield	NJ	34011
3	3	2	10 Ocean Blvd	Gardena	CA	90993
4	4	2	555 Henry St	Selma	CA	93662
5	5	3	10 1st street	Fresno	CA	93745
6	6	3	10000 Rodeo St	Oxnard	CA	93031
7	7	4	1033 N Sycamore Ave.	Los Angeles	CA	90038
8	8	4	10 Jackson Street	Jacksonville	FL	32231
9	9	5	P200 Farm Road	Fairfield	IA	52556
10	10	5	2709 Sweet Street, S...	Charlotte	NC	28217

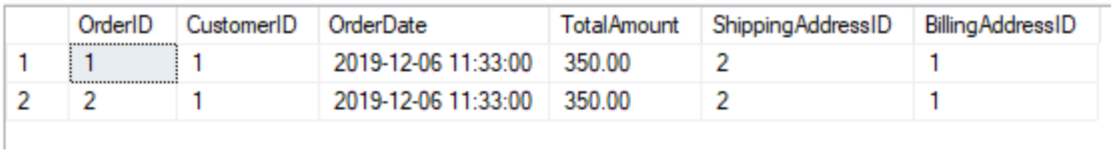


	ShippingAddressID	CustomerID	Address	City	State	ZipCode
1	1	1	10000 Rodeo St	Oxnard	CA	93031
2	2	1	1033 N Sycamore Ave.	Los Angeles	CA	90038
3	3	2	10 Jackson Street	Jacksonville	FL	32231
4	4	2	P200 Farm Road	Fairfield	IA	52556
5	5	3	121 State St - 4th Floor	Traverse ...	CA	95129
6	6	3	1200 Daniel Road	Fairfield	NJ	34011
7	7	4	10 Ocean Blvd	Gardena	CA	90993
8	8	4	555 Henry St	Selma	CA	93662
9	9	5	10 1st street	Fresno	CA	93745
10	10	5	2709 Sweet Street, S...	Charlotte	NC	28217

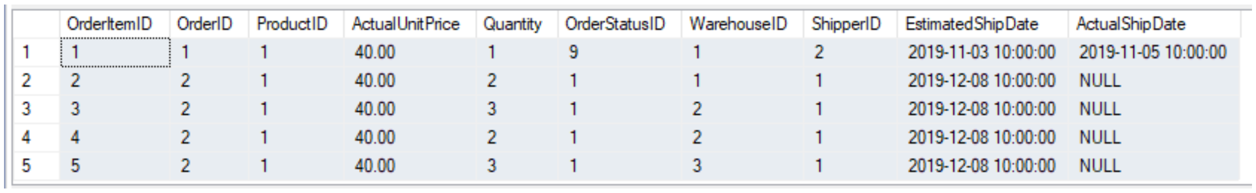
Each order will generate one row in Orders table and multiple rows in OrderItems table. It might generate row in ReturnItems table if there is return or exchange request. i

Customers (1 to many) Orders
Orders (1 to many) OrderItems
Orders (1 to many) ReturnItems
OrderStatus (1 to many) OrderItems
ReturnStatus (1 to many) ReturnItems

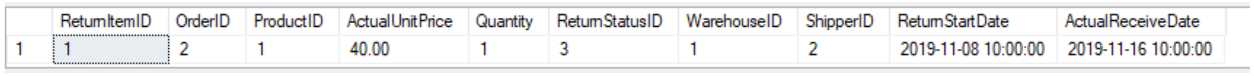
Orders, OrderItems and ReturnItems tables
and supporting tables : OrderStatus, ReturnStatus, Shippers



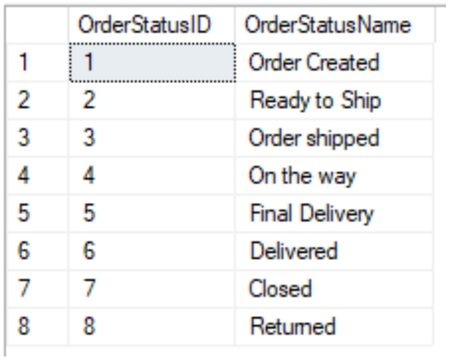
	OrderID	CustomerID	OrderDate	TotalAmount	ShippingAddressID	BillingAddressID
1	1	1	2019-12-06 11:33:00	350.00	2	1
2	2	1	2019-12-06 11:33:00	350.00	2	1



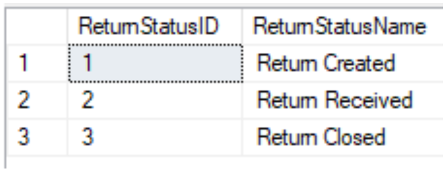
	OrderItemID	OrderID	ProductID	ActualUnitPrice	Quantity	OrderStatusID	WarehouseID	ShipperID	EstimatedShipDate	ActualShipDate
1	1	1	1	40.00	1	9	1	2	2019-11-03 10:00:00	2019-11-05 10:00:00
2	2	2	1	40.00	2	1	1	1	2019-12-08 10:00:00	NULL
3	3	2	1	40.00	3	1	2	1	2019-12-08 10:00:00	NULL
4	4	2	1	40.00	2	1	2	1	2019-12-08 10:00:00	NULL
5	5	2	1	40.00	3	1	3	1	2019-12-08 10:00:00	NULL



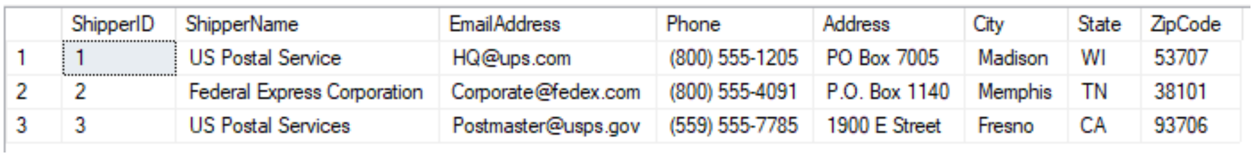
	ReturnItemID	OrderID	ProductID	ActualUnitPrice	Quantity	ReturnStatusID	WarehouseID	ShipperID	ReturnStartDate	ActualReceiveDate
1	1	2	1	40.00	1	3	1	2	2019-11-08 10:00:00	2019-11-16 10:00:00



	OrderStatusID	OrderStatusName
1	1	Order Created
2	2	Ready to Ship
3	3	Order shipped
4	4	On the way
5	5	Final Delivery
6	6	Delivered
7	7	Closed
8	8	Returned



	ReturnStatusID	ReturnStatusName
1	1	Return Created
2	2	Return Received
3	3	Return Closed

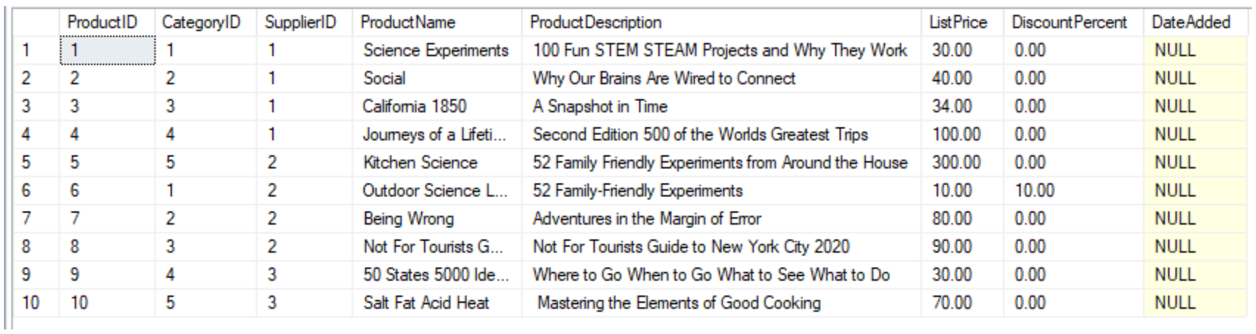


	ShipperID	ShipperName	EmailAddress	Phone	Address	City	State	ZipCode
1	1	US Postal Service	HQ@ups.com	(800) 555-1205	PO Box 7005	Madison	WI	53707
2	2	Federal Express Corporation	Corporate@fedex.com	(800) 555-4091	P.O. Box 1140	Memphis	TN	38101
3	3	US Postal Services	Postmaster@usps.gov	(559) 555-7785	1900 E Street	Fresno	CA	93706

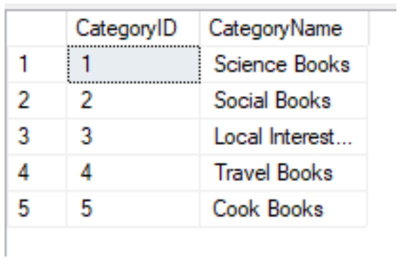
Product information are stored in Products table. The Categories table and Suppliers table provide the definition of product category definition and the information of the product suppliers.

Categories (1 to many) Products
Suppliers (1 to many) Products

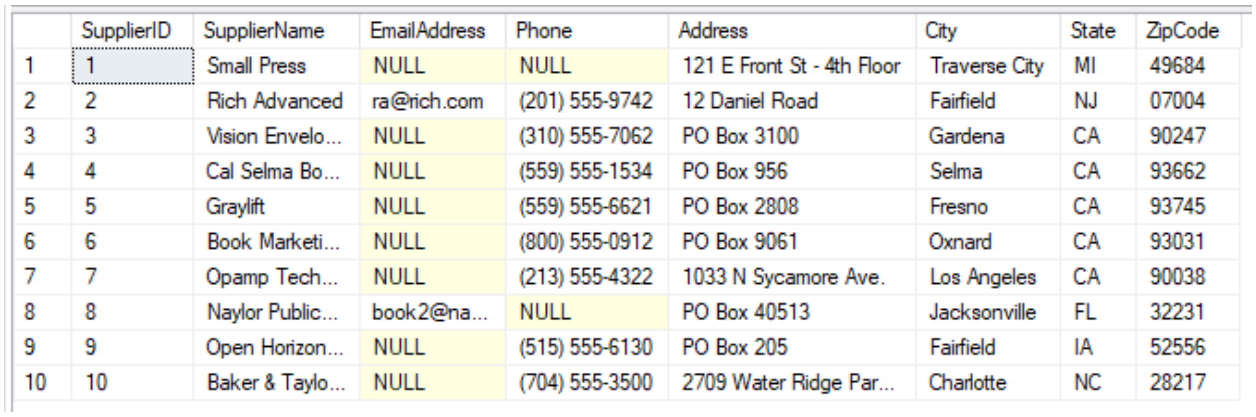
Products, Categories, and Suppliers tables :



	ProductID	CategoryID	SupplierID	ProductName	ProductDescription	ListPrice	DiscountPercent	DateAdded
1	1	1	1	Science Experiments	100 Fun STEM STEAM Projects and Why They Work	30.00	0.00	NULL
2	2	2	1	Social	Why Our Brains Are Wired to Connect	40.00	0.00	NULL
3	3	3	1	California 1850	A Snapshot in Time	34.00	0.00	NULL
4	4	4	1	Journeys of a Lifeti...	Second Edition 500 of the Worlds Greatest Trips	100.00	0.00	NULL
5	5	5	2	Kitchen Science	52 Family Friendly Experiments from Around the House	300.00	0.00	NULL
6	6	1	2	Outdoor Science L...	52 Family-Friendly Experiments	10.00	10.00	NULL
7	7	2	2	Being Wrong	Adventures in the Margin of Error	80.00	0.00	NULL
8	8	3	2	Not For Tourists G...	Not For Tourists Guide to New York City 2020	90.00	0.00	NULL
9	9	4	3	50 States 5000 Ide...	Where to Go When to Go What to See What to Do	30.00	0.00	NULL
10	10	5	3	Salt Fat Acid Heat	Mastering the Elements of Good Cooking	70.00	0.00	NULL



	CategoryID	CategoryName
1	1	Science Books
2	2	Social Books
3	3	Local Interest...
4	4	Travel Books
5	5	Cook Books

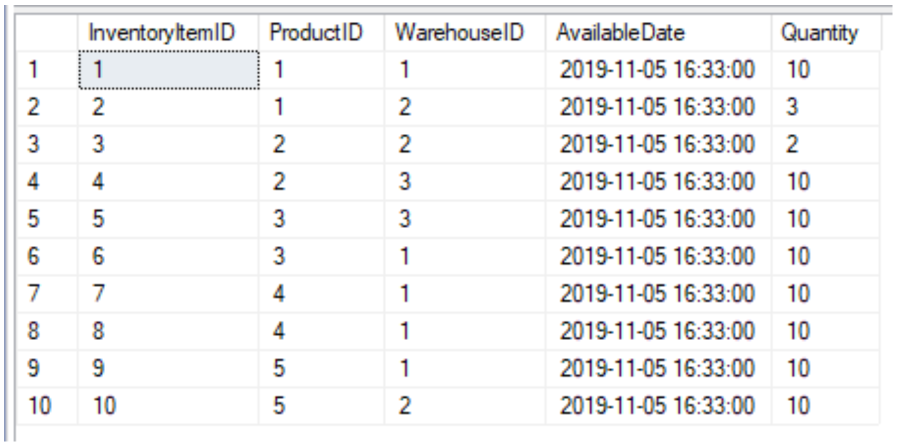


	SupplierID	SupplierName	EmailAddress	Phone	Address	City	State	ZipCode
1	1	Small Press	NULL	NULL	121 E Front St - 4th Floor	Traverse City	MI	49684
2	2	Rich Advanced	ra@rich.com	(201) 555-9742	12 Daniel Road	Fairfield	NJ	07004
3	3	Vision Envelo...	NULL	(310) 555-7062	PO Box 3100	Gardena	CA	90247
4	4	Cal Selma Bo...	NULL	(559) 555-1534	PO Box 956	Selma	CA	93662
5	5	Graylift	NULL	(559) 555-6621	PO Box 2808	Fresno	CA	93745
6	6	Book Marketi...	NULL	(800) 555-0912	PO Box 9061	Oxnard	CA	93031
7	7	Opamp Tech...	NULL	(213) 555-4322	1033 N Sycamore Ave.	Los Angeles	CA	90038
8	8	Naylor Public...	book2@na...	NULL	PO Box 40513	Jacksonville	FL	32231
9	9	Open Horizon...	NULL	(515) 555-6130	PO Box 205	Fairfield	IA	52556
10	10	Baker & Taylo...	NULL	(704) 555-3500	2709 Water Ridge Par...	Charlotte	NC	28217

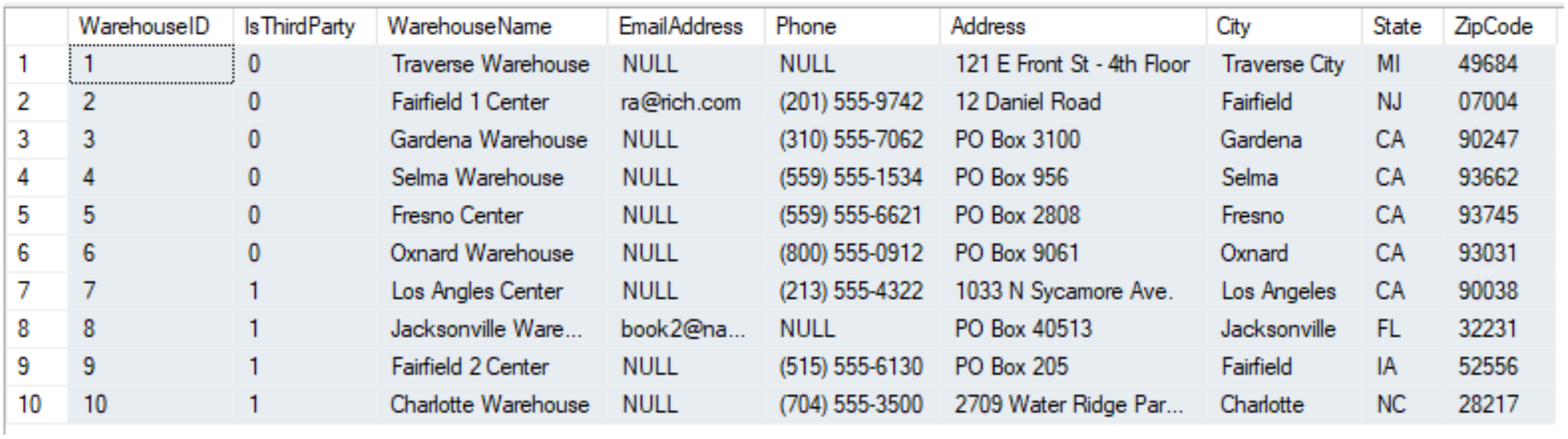
Inventory information are stored in InventoryItems table. Warehouses table and Products table store the product information and warehouse location the inventory is physically stored.

Categories (1 to many) InventoryItems
Suppliers (1 to many) InventoryItems

InventoryItems and Warehouses tables :



	InventoryItemID	ProductID	WarehouseID	AvailableDate	Quantity
1	1	1	1	2019-11-05 16:33:00	10
2	2	1	2	2019-11-05 16:33:00	3
3	3	2	2	2019-11-05 16:33:00	2
4	4	2	3	2019-11-05 16:33:00	10
5	5	3	3	2019-11-05 16:33:00	10
6	6	3	1	2019-11-05 16:33:00	10
7	7	4	1	2019-11-05 16:33:00	10
8	8	4	1	2019-11-05 16:33:00	10
9	9	5	1	2019-11-05 16:33:00	10
10	10	5	2	2019-11-05 16:33:00	10



	WarehouseID	IsThirdParty	WarehouseName	EmailAddress	Phone	Address	City	State	ZipCode
1	1	0	Traverse Warehouse	NULL	NULL	121 E Front St - 4th Floor	Traverse City	MI	49684
2	2	0	Fairfield 1 Center	ra@rich.com	(201) 555-9742	12 Daniel Road	Fairfield	NJ	07004
3	3	0	Gardena Warehouse	NULL	(310) 555-7062	PO Box 3100	Gardena	CA	90247
4	4	0	Selma Warehouse	NULL	(559) 555-1534	PO Box 956	Selma	CA	93662
5	5	0	Fresno Center	NULL	(559) 555-6621	PO Box 2808	Fresno	CA	93745
6	6	0	Oxnard Warehouse	NULL	(800) 555-0912	PO Box 9061	Oxnard	CA	93031
7	7	1	Los Angles Center	NULL	(213) 555-4322	1033 N Sycamore Ave.	Los Angeles	CA	90038
8	8	1	Jacksonville Ware...	book2@na...	NULL	PO Box 40513	Jacksonville	FL	32231
9	9	1	Fairfield 2 Center	NULL	(515) 555-6130	PO Box 205	Fairfield	IA	52556
10	10	1	Charlotte Warehouse	NULL	(704) 555-3500	2709 Water Ridge Par...	Charlotte	NC	28217

(B.2) This is the script to restore the foreign key constraints after data are populated into tables.

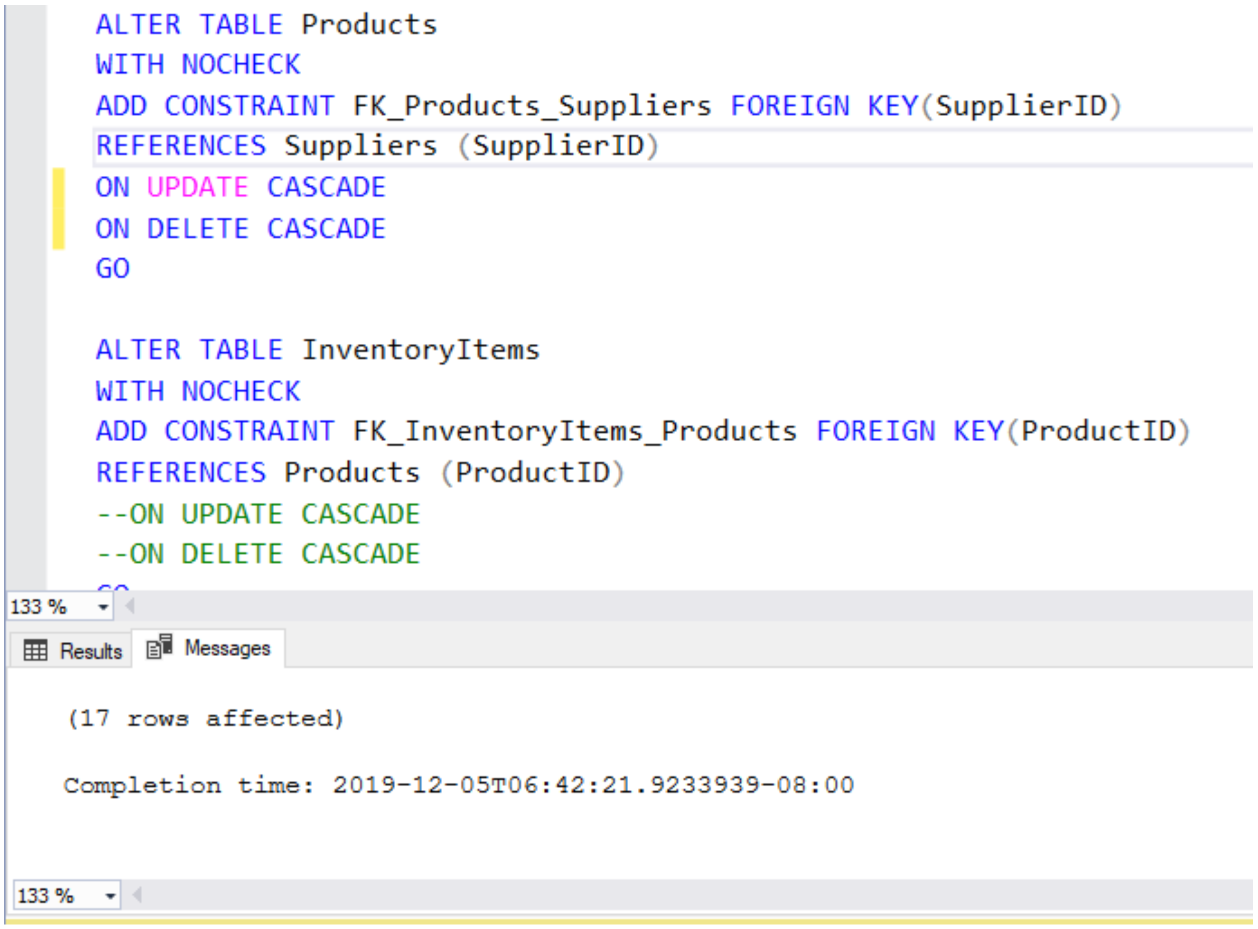
```
-- This is the script to add reference foreign key reference constraints

USE MyOnline;

ALTER TABLE BillingAddress
WITH NOCHECK
ADD CONSTRAINT FK_BillingAddress_Customers FOREIGN KEY(CustomerID)
REFERENCES Customers (CustomerID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE ShippingAddress
WITH NOCHECK
ADD CONSTRAINT FK_ShippingAddress_Customers FOREIGN KEY
(CustomerID)
REFERENCES Customers (CustomerID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE Products
WITH NOCHECK
ADD CONSTRAINT FK_Products_Categories FOREIGN KEY(CategoryID)
REFERENCES Categories (CategoryID)
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE Products
WITH NOCHECK
ADD CONSTRAINT FK_Products_Suppliers FOREIGN KEY(SupplierID)
REFERENCES Suppliers (SupplierID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE InventoryItems
WITH NOCHECK
ADD CONSTRAINT FK_InventoryItems_Products FOREIGN KEY(ProductID)
REFERENCES Products (ProductID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
```

```
GO

ALTER TABLE InventoryItems
WITH NOCHECK
ADD CONSTRAINT FK_InventoryItems_Warehouses FOREIGN KEY
(WarehouseID)
REFERENCES Warehouses (WarehouseID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE WishLists
WITH NOCHECK
ADD CONSTRAINT FK_WishLists_Customers FOREIGN KEY(CustomerID)
REFERENCES Customers (CustomerID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE WishLists
WITH NOCHECK
ADD CONSTRAINT FK_WishLists_Products FOREIGN KEY(ProductID)
REFERENCES Products (ProductID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE Orders
WITH NOCHECK
ADD CONSTRAINT FK_Orders_Customers FOREIGN KEY(CustomerID)
REFERENCES Customers (CustomerID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE Orders
WITH NOCHECK
ADD CONSTRAINT FK_Orders_ShippingAddress FOREIGN  KEY
(ShippingAddressID)
REFERENCES ShippingAddress (ShippingAddressID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE Orders
WITH NOCHECK
```

```
ADD CONSTRAINT FK_Orders_BillingAddress FOREIGN KEY(BillingAddressID)
REFERENCES BillingAddress (BillingAddressID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE Reviews
WITH NOCHECK
ADD CONSTRAINT FK_Reviews_ShippingAddress FOREIGN KEY(ProductID)
REFERENCES Products (ProductID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE Reviews
WITH NOCHECK
ADD CONSTRAINT FK_Reviews_Orders FOREIGN KEY(OrderID) REFERENCES
Orders (OrderID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE Reviews
WITH NOCHECK
ADD CONSTRAINT FK_Reviews_ReviewScores FOREIGN KEY(ReviewScoreID)
REFERENCES ReviewScores (ReviewScoreID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE OrderItems
WITH NOCHECK
ADD CONSTRAINT FK_OrderItems_Orders FOREIGN KEY(OrderID)
REFERENCES Orders (OrderID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE OrderItems
WITH NOCHECK
ADD CONSTRAINT FK_OrderItems_Products FOREIGN KEY(ProductID)
REFERENCES Products (ProductID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO
```

```
ALTER TABLE OrderItems
WITH NOCHECK
ADD CONSTRAINT FK_OrderItems_OrderStatus FOREIGN KEY(OrderStatusID)
REFERENCES OrderStatus (OrderStatusID)
ON UPDATE NO ACTION
ON DELETE NO ACTION
GO

ALTER TABLE OrderItems
WITH NOCHECK
ADD CONSTRAINT FK_OrderItems_Warehouses FOREIGN KEY(WarehouseID)
REFERENCES Warehouses (WarehouseID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE OrderItems
WITH NOCHECK
ADD CONSTRAINT FK_OrderItems_Shippers FOREIGN KEY(ShipperID)
REFERENCES Shippers (ShipperID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE ReturnItems
WITH NOCHECK
ADD CONSTRAINT FK_ReturnItems_Orders FOREIGN KEY(OrderID)
REFERENCES Orders (OrderID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE ReturnItems
WITH NOCHECK
ADD CONSTRAINT FK_ReturnItems_Products FOREIGN KEY(ProductID)
REFERENCES Products (ProductID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE ReturnItems
WITH NOCHECK
ADD CONSTRAINT FK_ReturnItems_ReturnStatus FOREIGN KEY
(ReturnStatusID)
REFERENCES ReturnStatus (ReturnStatusID)
ON UPDATE NO ACTION
ON DELETE NO ACTION
GO

ALTER TABLE ReturnItems
WITH NOCHECK
ADD CONSTRAINT FK_ReturnItems_Warehouses FOREIGN KEY(WarehouseID)
REFERENCES Warehouses (WarehouseID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

ALTER TABLE ReturnItems
WITH NOCHECK
ADD CONSTRAINT FK_ReturnItems_Shippers FOREIGN KEY(ShipperID)
REFERENCES Shippers (ShipperID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
GO

SELECT * FROM sys.tables;
```

The script executed successfully and the tables were populated with sample data



```
ALTER TABLE Products
WITH NOCHECK
ADD CONSTRAINT FK_Products_Suppliers FOREIGN KEY(SupplierID)
REFERENCES Suppliers (SupplierID)
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE InventoryItems
WITH NOCHECK
ADD CONSTRAINT FK_InventoryItems_Products FOREIGN KEY(ProductID)
REFERENCES Products (ProductID)
--ON UPDATE CASCADE
--ON DELETE CASCADE
```

```
(17 rows affected)

Completion time: 2019-12-05T06:42:21.9233939-08:00
```

(C) Create the permission and database role

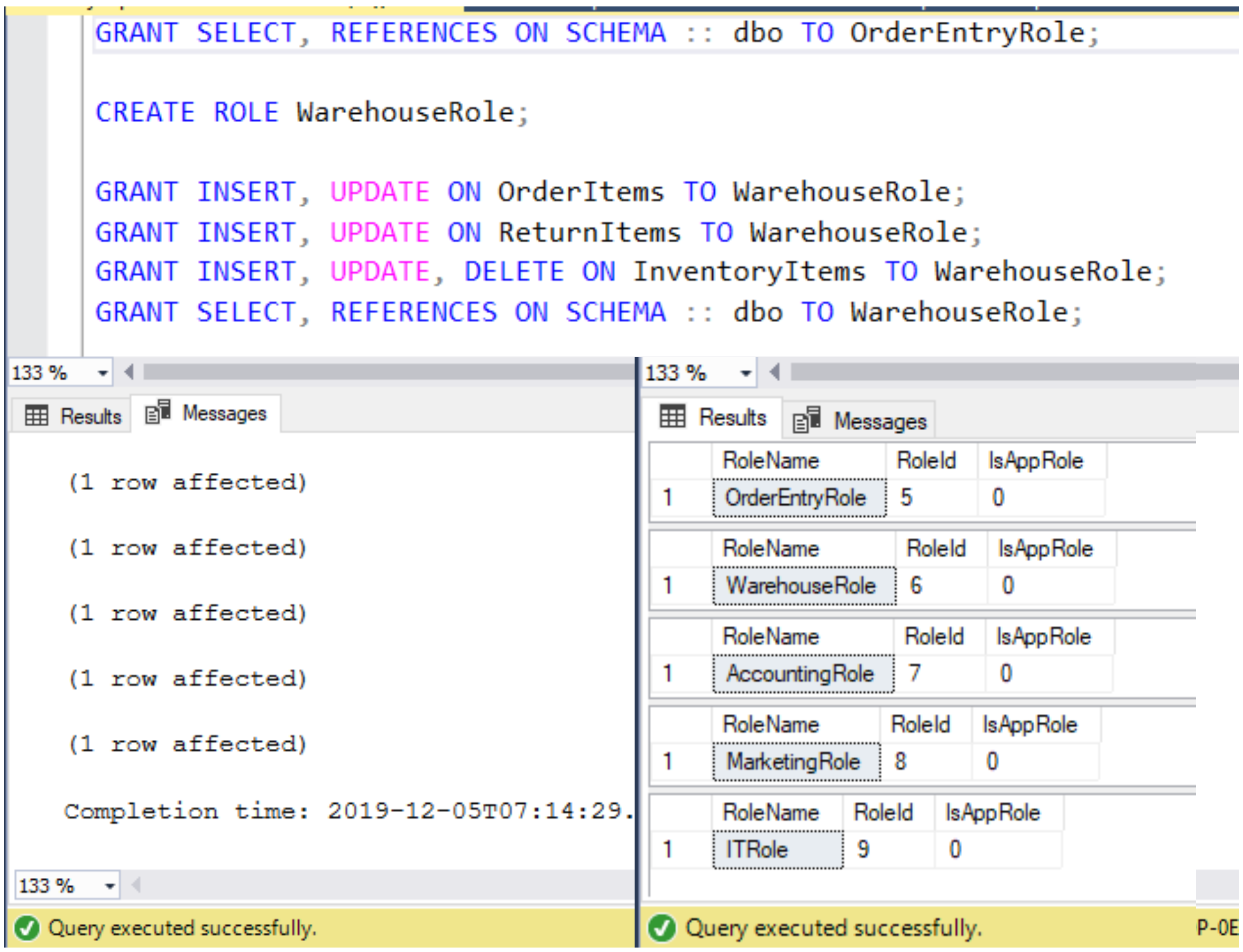
The source code of the database role and database & object permission creation is listed as following:

```
-- Create five different application roles to for different peoples to work in
--        different duties.  There are
--                Order-Entry role who take customer order
--                Warehouse role handles warehouse inventory, shipment and returns.
--                Accounting role archived finished order
--                Marketing role handles product definition, promotion,
--                                responds customer review and wishlist
--                IT role who has the most privilege to make large scale database update
--
--        Application role is created for frontend software development.
--
-- Server login and database user to 'ADD MEMBER' by IT

USE MyOnline;
GO

CREATE ROLE OrderEntryRole;

GRANT INSERT, UPDATE ON Orders TO OrderEntryRole;
GRANT INSERT, UPDATE ON OrderItems TO OrderEntryRole;
GRANT INSERT, UPDATE ON ReturnItems TO OrderEntryRole;
GRANT UPDATE ON InventoryItems TO OrderEntryRole;
GRANT INSERT ON WishLists TO OrderEntryRole;
GRANT INSERT, UPDATE ON Customers TO OrderEntryRole;
GRANT INSERT, UPDATE ON ShippingAddress TO OrderEntryRole;
GRANT INSERT, UPDATE ON BillingAddress TO OrderEntryRole;
GRANT SELECT, REFERENCES ON SCHEMA :: dbo TO OrderEntryRole;

CREATE ROLE WarehouseRole;

GRANT INSERT, UPDATE ON OrderItems TO WarehouseRole;
GRANT INSERT, UPDATE ON ReturnItems TO WarehouseRole;
GRANT INSERT, UPDATE, DELETE ON InventoryItems TO WarehouseRole;
GRANT SELECT, REFERENCES ON SCHEMA :: dbo TO WarehouseRole;

CREATE ROLE AccountingRole;

GRANT INSERT, UPDATE, DELETE ON Orders TO AccountingRole;
GRANT INSERT, UPDATE, DELETE ON OrderItems TO AccountingRole;
GRANT INSERT, UPDATE, DELETE ON ReturnItems TO AccountingRole;
GRANT INSERT, UPDATE, DELETE ON InventoryItems TO AccountingRole;
GRANT SELECT, REFERENCES ON SCHEMA :: dbo TO AccountingRole;

CREATE ROLE MarketingRole;

GRANT INSERT, UPDATE, DELETE ON Products TO MarketingRole;
GRANT INSERT, UPDATE, DELETE ON Categories TO MarketingRole;
GRANT INSERT, UPDATE, DELETE ON Reviews TO MarketingRole;
GRANT SELECT, REFERENCES ON SCHEMA :: dbo TO MarketingRole;

CREATE ROLE ITRole;

GRANT SELECT, INSERT, UPDATE, DELETE, REFERENCES, EXECUTE, ALTER
ON SCHEMA :: dbo TO ITRole;

CREATE APPLICATION ROLE MyOnlineApplicationRole
        WITH PASSWORD = 'passpasswordword',
        DEFAULT_SCHEMA = dbo;

GRANT SELECT, REFERENCES ON Products TO MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON WishLists TO
MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON Reviews TO
MyOnlineApplicationRole;
GRANT SELECT, UPDATE, REFERENCES ON Customers TO
MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON BillingAddress TO
MyOnlineApplicationRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON ShippingAddress TO
MyOnlineApplicationRole;
GO

EXEC sp_HelpRole OrderEntryRole;
EXEC sp_HelpRole WarehouseRole;
EXEC sp_HelpRole AccountingRole;
EXEC sp_HelpRole MarketingRole;
EXEC sp_HelpRole ITRole;

-- Server login and database user to 'ADD MEMBER' by IT when the database is
released.
/*
CREATE LOGIN LOGIN001 FROM WINDOWS WITH DEFAULT_DATABASE = MyOnline;
CREATE LOGIN LOGIN002 FROM WINDOWS WITH DEFAULT_DATABASE = MyOnline;
CREATE LOGIN LOGIN003 FROM WINDOWS WITH DEFAULT_DATABASE = MyOnline;
CREATE LOGIN LOGIN004 FROM WINDOWS WITH DEFAULT_DATABASE = MyOnline;
CREATE LOGIN LOGIN005 FROM WINDOWS WITH DEFAULT_DATABASE = MyOnline;
```

CREATE USER MyOnlineEmployee001 WITH DEFAULT_SCHEMA = dbo;
CREATE USER MyOnlineEmployee002 WITH DEFAULT_SCHEMA = dbo;
CREATE USER MyOnlineEmployee003 WITH DEFAULT_SCHEMA = dbo;
CREATE USER MyOnlineEmployee004 WITH DEFAULT_SCHEMA = dbo;
CREATE USER MyOnlineEmployee005 WITH DEFAULT_SCHEMA = dbo;

ALTER ROLE OrderEntry ADD MEMBER MyOnlinEmployee001;
ALTER ROLE OrderEntry ADD MEMBER MyOnlinEmployee002;
ALTER ROLE OrderEntry ADD MEMBER MyOnlinEmployee003;
ALTER ROLE OrderEntry ADD MEMBER MyOnlinEmployee004;
ALTER ROLE OrderEntry ADD MEMBER MyOnlinEmployee005;
*/

The script executed successfully



```
GRANT SELECT, REFERENCES ON SCHEMA :: dbo TO OrderEntryRole;

CREATE ROLE WarehouseRole;

GRANT INSERT, UPDATE ON OrderItems TO WarehouseRole;
GRANT INSERT, UPDATE ON ReturnItems TO WarehouseRole;
GRANT INSERT, UPDATE, DELETE ON InventoryItems TO WarehouseRole;
GRANT SELECT, REFERENCES ON SCHEMA :: dbo TO WarehouseRole;
```

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2019-12-05T07:14:29.
```

Query executed successfully.

	RoleName	RoleId	IsAppRole
1	OrderEntryRole	5	0

	RoleName	RoleId	IsAppRole
1	WarehouseRole	6	0

	RoleName	RoleId	IsAppRole
1	AccountingRole	7	0

	RoleName	RoleId	IsAppRole
1	MarketingRole	8	0

	RoleName	RoleId	IsAppRole
1	ITRole	9	0

Query executed successfully.

(D) Views, Functions, Stored Procedures and reports

The following are the Views, Functions, and Stored Procedure I created   I think VIEW, Stored Procedure, and Function are very application specific and can be wisely selected after the frontend program structure is planned and the frontend program architect can request the kind of view they need in order to make the program efficient.

The source code of this portion is listed as below:

```
 -- This script contains the creation of the Views, Functions, and Stored procedure.
 -- Five reports are generated

USE MyOnline;
GO

————————————————————————————————————————

-- View for customer billing information

IF OBJECT_ID ('CustomerShippingAddressView') IS NOT NULL
DROP VIEW CustomerShippingAddressView;
GO

CREATE VIEW CustomerShippingAddressView
AS
SELECT Customers.CustomerID, LastName, FirstName, CellPhone,
'Shipping' AS Type,
ShippingAddress.ZipCode AS State,
ShippingAddress.ZipCode AS ZipCode
FROM Customers
JOIN ShippingAddress ON Customers.CustomerID=ShippingAddress.CustomerID
GO

————————————————————————————————————————

-- View for customer billing information

IF OBJECT_ID ('CustomerBillingAddressView') IS NOT NULL
DROP VIEW CustomerBillingAddressView;
GO

CREATE VIEW CustomerBillingAddressView
AS
SELECT Customers.CustomerID, LastName, FirstName, CellPhone,
'Billing' AS Type,
BillingAddress.ZipCode AS State,
BillingAddress.ZipCode AS ZipCode
FROM Customers
JOIN BillingAddress ON Customers.CustomerID=BillingAddress.CustomerID
GO
```

————————————————————————————————————

```
-- View for order item summary

IF OBJECT_ID ('OrderItemsSummaryView') IS NOT NULL
DROP VIEW OrderItemsSummaryView;
GO

CREATE VIEW OrderItemsSummaryView
AS
WITH OrderDetails AS
(       SELECT OrderID, OrderItemID, ProductName, Quantity, OrderStatusName
   FROM OrderItems
        JOIN OrderStatus
        ON OrderItems.OrderStatusID = OrderStatus.OrderStatusID
        JOIN Products
        ON OrderItems.ProductID = Products.ProductID
)
SELECT  Orders.OrderID AS OrderID,
                Orders.OrderDate AS OrderDate,
                OrderDetails.OrderItemID AS ItemID,
                OrderDetails.ProductName AS ProductName,
                OrderDetails.Quantity AS Quantity,
                'Order Item' AS Type,
                OrderDetails.OrderStatusName AS Status
FROM Orders JOIN OrderDetails
ON Orders.OrderID = OrderDetails.OrderID;
GO
```

————————————————————————————————————

```
-- View for return item summary

IF OBJECT_ID ('ReturnItemsSummaryView') IS NOT NULL
DROP VIEW ReturnItemsSummaryView;
GO

CREATE VIEW ReturnItemsSummaryView
AS
```

```
WITH ReturnDetails AS
(        SELECT OrderID, ReturnItemID, ProductName, Quantity, ReturnStatusName
   FROM ReturnItems
         JOIN ReturnStatus
         ON ReturnItems.ReturnStatusID = ReturnStatus.ReturnStatusID
         JOIN Products
         ON ReturnItems.ProductID = Products.ProductID
)
SELECT  Orders.OrderID AS OrderID,
                 Orders.OrderDate AS OrderDate,
                 ReturnDetails.ReturnItemID AS ItemID,
                 ReturnDetails.ProductName AS ProductName,
                 ReturnDetails.Quantity AS Quantity,
                 'Return Item' AS Type,
                 ReturnDetails.ReturnStatusName AS Status
FROM Orders JOIN ReturnDetails
ON Orders.OrderID = ReturnDetails.OrderID;
GO
```

———————————————————————————————————

```
-- function to calculate the shipping cost

IF OBJECT_ID ('fn_ShippingTotal') IS NOT NULL
DROP FUNCTION fn_ShippingTotal;
GO

CREATE FUNCTION fn_ShippingTotal
                 (@WarehouseID INT = 0, @ShippingAddressID INT = 0)
                 RETURNS MONEY
BEGIN
         -- The shipping cost is normally set by shipping service provider
         -- with the zone definition.  It's highly frontend-specific, not
         -- related to database design. This function will provide a simple
         -- price without the knowledge of actual zoning.
         DECLARE @Source INT;
         DECLARE @Target INT;
         DECLARE @Fare MONEY;
         SELECT @Source = State FROM Warehouses
                                 WHERE WarehouseID = @WarehouseID;
         SELECT @Target = State FROM ShippingAddress
                                 WHERE ShippingAddressID = @ShippingAddressID;
         if @Source = @Target
                 SET @Fare = 2.00
         Else
                 SET @Fare = 5.00    ;
```

```
   RETURN (@Fare);
END;
GO
```

———————————————————————————————————

```
-- stored procedure to send shipping notice to warehouse

IF OBJECT_ID ('sp_ShippingNotice') IS NOT NULL
DROP PROC sp_ShippingNotice;
GO

CREATE PROC sp_ShippingNotice
AS
SELECT OrderItems.OrderItemID,
          OrderItems.EstimatedShipDate,
          OrderItems.ProductID,
          Products.ProductName,
          OrderItems.Quantity,
          Shippers.ShipperName,
          Customers.LastName + '. ' + Customers.FirstName AS Name,
          ShippingAddress.Address + ', '+ ShippingAddress.City + ' '+
          ShippingAddress.State+ ', '+ShippingAddress.ZipCode AS Address
FROM OrderItems
JOIN Warehouses ON OrderItems.WarehouseID = Warehouses.WarehouseID
JOIN Products ON OrderItems.ProductID = Products.ProductID
JOIN Shippers ON OrderItems.ShipperID = Shippers.ShipperID
JOIN Orders ON OrderItems.OrderID = Orders.OrderID
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
JOIN ShippingAddress ON Orders.ShippingAddressID =
ShippingAddress.ShippingAddressID
ORDER BY Warehouses.WarehouseID ASC;
```

———————————————————————————————————

```
-- store procedure to send alert of low review score

IF OBJECT_ID ('sp_LowReviewAlert') IS NOT NULL
DROP PROC sp_LowReviewAlert;
GO

CREATE PROC sp_LowReviewAlert
AS
SELECT
                ReviewDate AS Date,
                ReviewScoreName AS Score,
                Reviews.ProductID,
                ProductName AS Product,
                Reviews.OrderID,
                CustomerID AS Customer,
                ReviewText AS Review
FROM Reviews
JOIN Products ON Reviews.ProductID = Products.ProductID
JOIN Orders ON Reviews.OrderID = Orders.OrderID
JOIN ReviewScores ON Reviews.ReviewScoreID = ReviewScores.ReviewScoreID
WHERE Reviews.ReviewScoreID <=3
ORDER BY Reviews.ReviewScoreID ASC;
```

———————————————————————————————————

```
-- stored procedure to calculate order price total

IF OBJECT_ID ('sp_PriceTotal') IS NOT NULL
DROP PROC sp_PriceTotal;
GO

CREATE PROC sp_PriceTotal
                @OrderID    INT,
                @Price          MONEY      OUTPUT
AS
        DECLARE @OrderTotal MONEY;
        DECLARE @ReturnTotal MONEY;
        SELECT @OrderTotal = SUM(ActualUnitPrice * Quantity)
          FROM OrderItems
          WHERE OrderID = @OrderID;
        SELECT @ReturnTotal = SUM(ActualUnitPrice * Quantity)
          FROM ReturnItems
          WHERE OrderID = @OrderID;
        SET @Price = @OrderTotal - @ReturnTotal;
GO
```

———————————————————————————————————

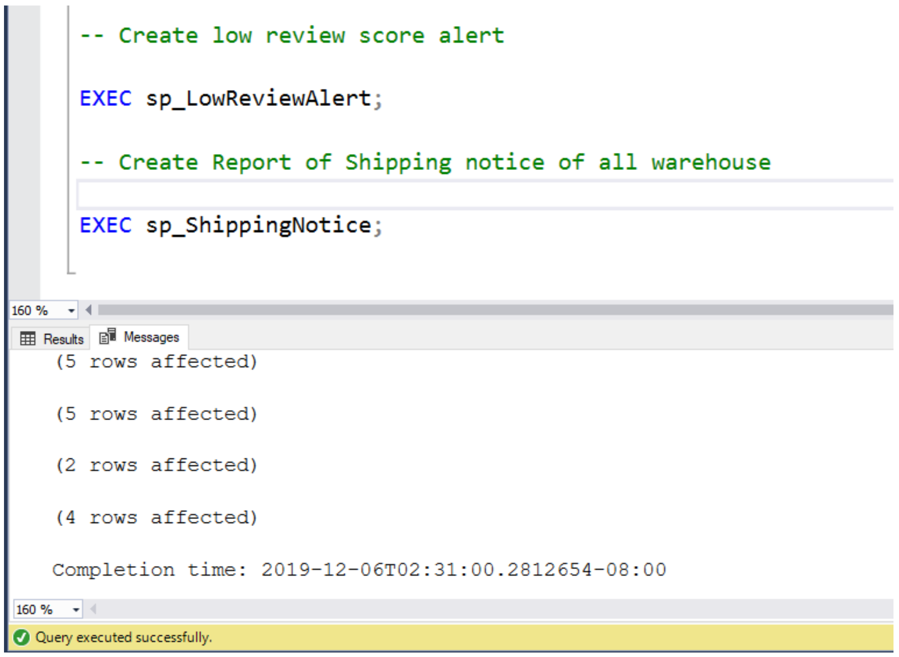
```
-- stored procedure to create printing label

IF OBJECT_ID ('sp_PrintLabel') IS NOT NULL
DROP PROC sp_PrintLabel;
GO

CREATE PROC sp_PrintLabel
                @ShippingAddressID       INT
```

```
AS
SELECT LastName + CHAR (13) + CHAR (10)
                    + Address + CHAR (13) + CHAR (10)
                    + City + ' ' + State + ZipCode
FROM ShippingAddress JOIN Customers
ON ShippingAddress.CustomerID = Customers.CustomerID
WHERE ShippingAddress.ShippingAddressID = @ShippingAddressID;
GO
```

———————————————————————————————————

```
-- Create customer information summary report
SELECT * FROM CustomerShippingAddressView
UNION
SELECT * FROM CustomerBillingAddressView;

-- Create order item summary report
SELECT * FROM OrderItemsSummaryView
UNION
SELECT * FROM ReturnItemsSummaryView
ORDER BY OrderID;
SELECT * FROM OrderItemsSummaryView
SELECT * FROM ReturnItemsSummaryView;

-- Create low review score alert
EXEC sp_LowReviewAlert;

-- Create Report of Shipping notice of all warehouse
EXEC sp_ShippingNotice;
```

The script executed successfully.

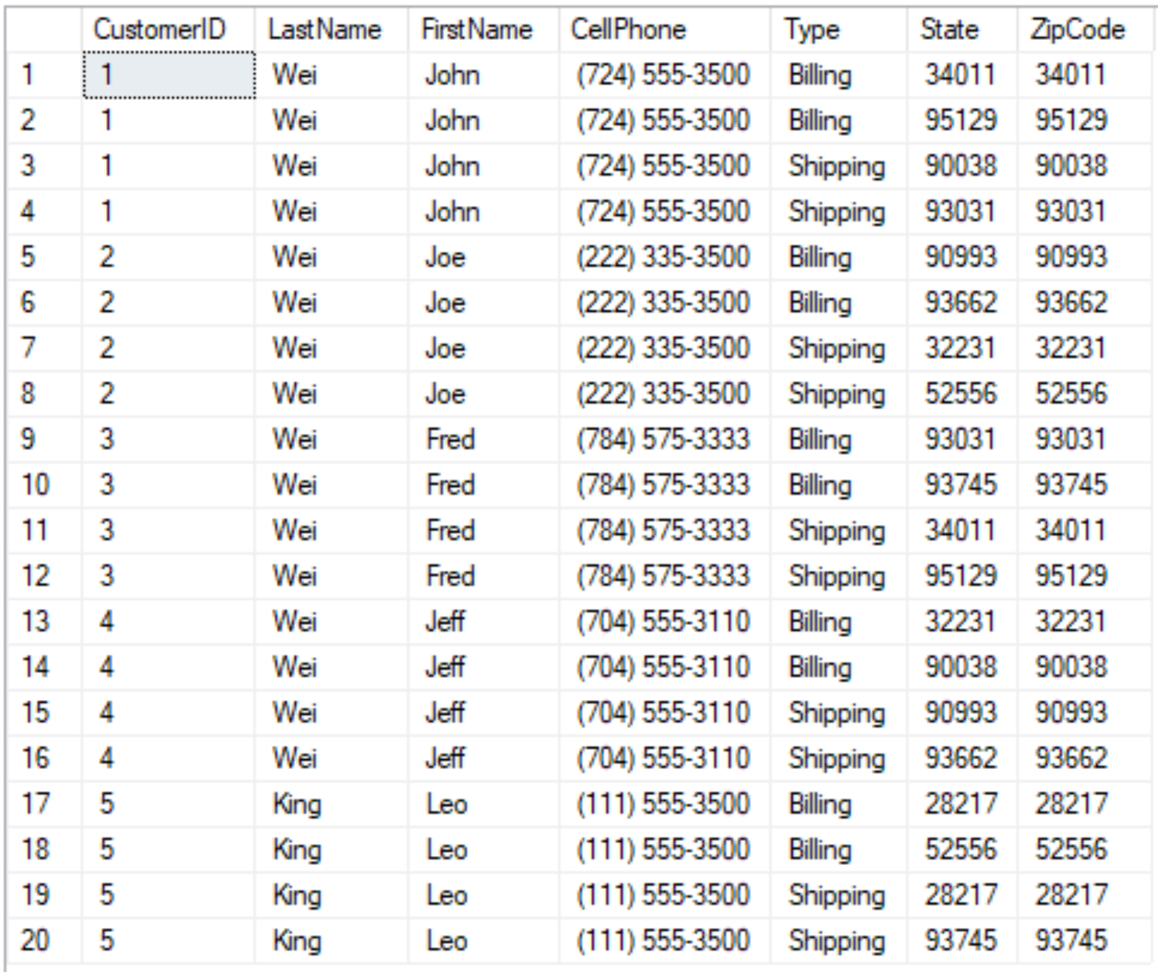


```
-- Create low review score alert

EXEC sp_LowReviewAlert;

-- Create Report of Shipping notice of all warehouse

EXEC sp_ShippingNotice;
```

```
(5 rows affected)

(5 rows affected)

(2 rows affected)

(4 rows affected)

Completion time: 2019-12-06T02:31:00.2812654-08:00
```

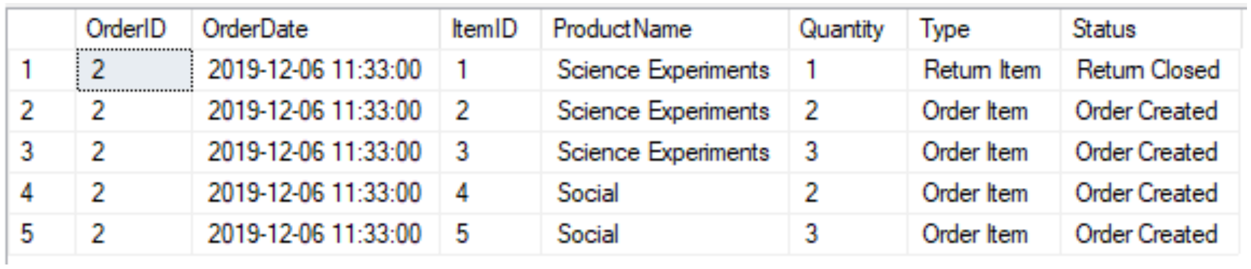
Query executed successfully.

This is the report of customers' shipping addresses and billing addresses. This database is capable of maintain multiple addresses for each customer and maintain the online transactions with them.



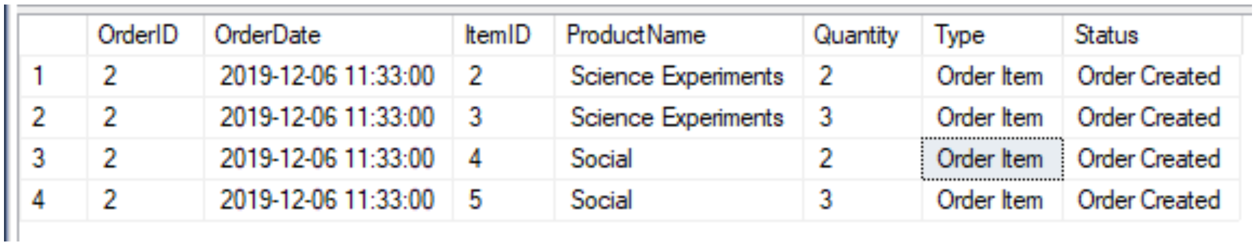
	CustomerID	LastName	FirstName	CellPhone	Type	State	ZipCode
1	1	Wei	John	(724) 555-3500	Billing	34011	34011
2	1	Wei	John	(724) 555-3500	Billing	95129	95129
3	1	Wei	John	(724) 555-3500	Shipping	90038	90038
4	1	Wei	John	(724) 555-3500	Shipping	93031	93031
5	2	Wei	Joe	(222) 335-3500	Billing	90993	90993
6	2	Wei	Joe	(222) 335-3500	Billing	93662	93662
7	2	Wei	Joe	(222) 335-3500	Shipping	32231	32231
8	2	Wei	Joe	(222) 335-3500	Shipping	52556	52556
9	3	Wei	Fred	(784) 575-3333	Billing	93031	93031
10	3	Wei	Fred	(784) 575-3333	Billing	93745	93745
11	3	Wei	Fred	(784) 575-3333	Shipping	34011	34011
12	3	Wei	Fred	(784) 575-3333	Shipping	95129	95129
13	4	Wei	Jeff	(704) 555-3110	Billing	32231	32231
14	4	Wei	Jeff	(704) 555-3110	Billing	90038	90038
15	4	Wei	Jeff	(704) 555-3110	Shipping	90993	90993
16	4	Wei	Jeff	(704) 555-3110	Shipping	93662	93662
17	5	King	Leo	(111) 555-3500	Billing	28217	28217
18	5	King	Leo	(111) 555-3500	Billing	52556	52556
19	5	King	Leo	(111) 555-3500	Shipping	28217	28217
20	5	King	Leo	(111) 555-3500	Shipping	93745	93745

This is the report of details of the shipments and returns for a single order (orderID = 2). The summary is small since the limited data populated into the table. In theory, a product item could be shipped and returned multiple-times. Although it's unlikely to happen in real world but that's a possibility the database has to handle if it does happens.



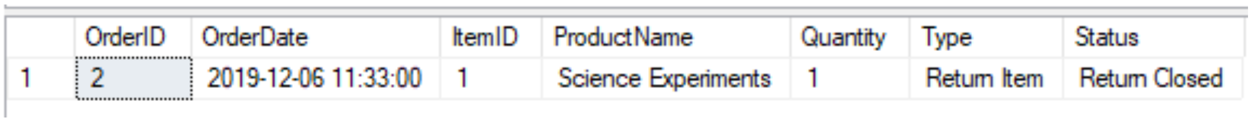
	OrderID	OrderDate	ItemID	ProductName	Quantity	Type	Status
1	2	2019-12-06 11:33:00	1	Science Experiments	1	Return Item	Return Closed
2	2	2019-12-06 11:33:00	2	Science Experiments	2	Order Item	Order Created
3	2	2019-12-06 11:33:00	3	Science Experiments	3	Order Item	Order Created
4	2	2019-12-06 11:33:00	4	Social	2	Order Item	Order Created
5	2	2019-12-06 11:33:00	5	Social	3	Order Item	Order Created

This report is the summary of order items. Each items represents some quantity of a product to be shipped by certain warehouse. Since the flexibility provided by the system, A product ordered in one order can be divided into shipment from several different warehouses.



	OrderID	OrderDate	ItemID	ProductName	Quantity	Type	Status
1	2	2019-12-06 11:33:00	2	Science Experiments	2	Order Item	Order Created
2	2	2019-12-06 11:33:00	3	Science Experiments	3	Order Item	Order Created
3	2	2019-12-06 11:33:00	4	Social	2	Order Item	Order Created
4	2	2019-12-06 11:33:00	5	Social	3	Order Item	Order Created

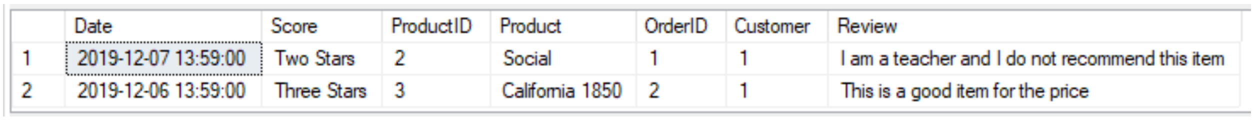
This is the report of the return items. This is the shortest report since limited amount of data.



	OrderID	OrderDate	ItemID	ProductName	Quantity	Type	Status
1	2	2019-12-06 11:33:00	1	Science Experiments	1	Return Item	Return Closed

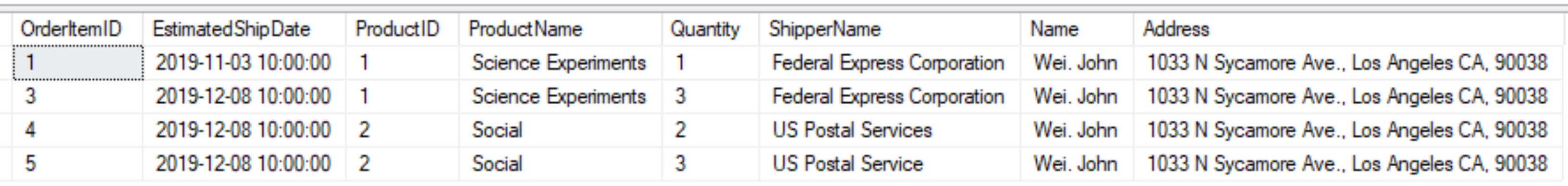
This is the report of low score reviews created by customers. Due to limited amount of data, the summary is very small.

Marveling team can review the customers' review on products to make product availability adjustment. This summary is generated by reference several tables. Actually, this summary can be more meaningful if we have large amount of data showing the actual sales of the products to compare against these low customer review scores. That will make the decision making processes much more fitted to the market.



	Date	Score	ProductID	Product	OrderID	Customer	Review
1	2019-12-07 13:59:00	Two Stars	2	Social	1	1	I am a teacher and I do not recommend this item
2	2019-12-06 13:59:00	Three Stars	3	California 1850	2	1	This is a good item for the price

This report is the shipment 'work order' for the warehouse. Due to limited data, the table is very small. Basically the warehouse will receive the shipments to be made, the expected shipment date (or the deadline to make the shipment happen), the product name, quantity, the shipping services selected by system based on cost consideration, and target shipping name and address.

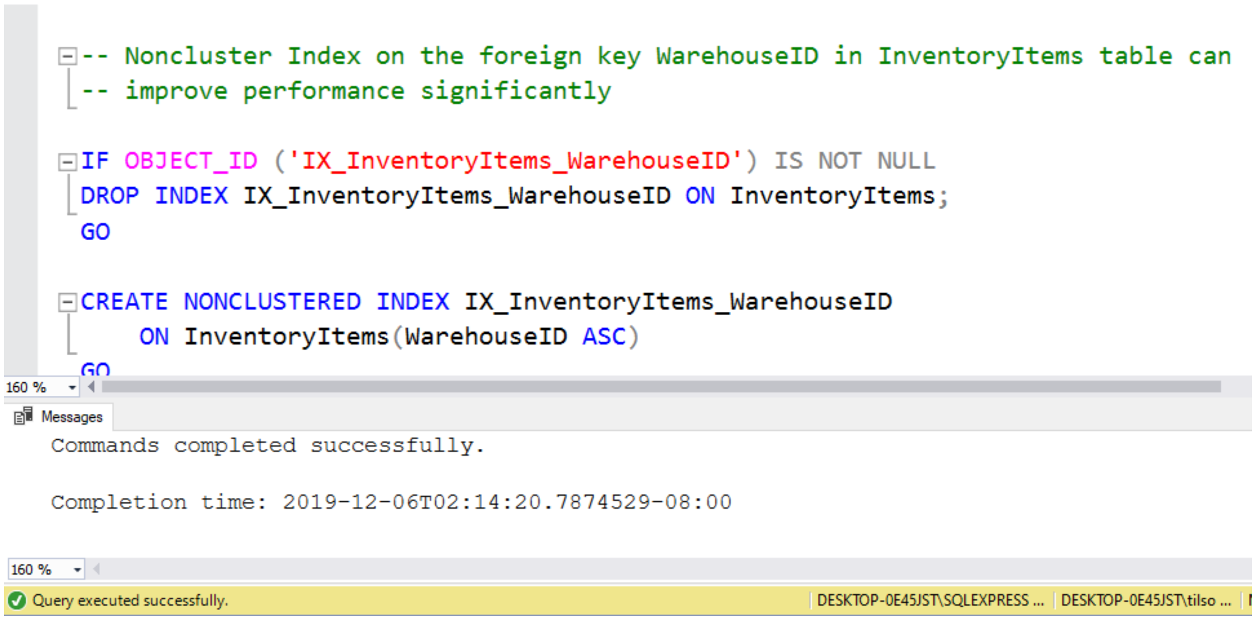


OrderItemID	EstimatedShipDate	ProductID	ProductName	Quantity	ShipperName	Name	Address
1	2019-11-03 10:00:00	1	Science Experiments	1	Federal Express Corporation	Wei. John	1033 N Sycamore Ave., Los Angeles CA, 90038
3	2019-12-08 10:00:00	1	Science Experiments	3	Federal Express Corporation	Wei. John	1033 N Sycamore Ave., Los Angeles CA, 90038
4	2019-12-08 10:00:00	2	Social	2	US Postal Services	Wei. John	1033 N Sycamore Ave., Los Angeles CA, 90038
5	2019-12-08 10:00:00	2	Social	3	US Postal Service	Wei. John	1033 N Sycamore Ave., Los Angeles CA, 90038

(E) Index Creation for performance Improvement

```
-- This script create INDEX on MyOnline database to improve the performance

USE MyOnline;
GO

-- Noncluster Index on the foreign key OrderID in OrderItems table can improve
-- performance significantly

IF OBJECT_ID ('IX_OrderItems_OrderID') IS NOT NULL
DROP INDEX IX_OrderItems_OrderID ON OrderItems;
GO

CREATE NONCLUSTERED INDEX IX_OrderItems_OrderID
ON OrderItems(OrderID ASC)
GO

-- Noncluster Index on the foreign key OrderID in ReturnItems table can improve
-- performance significantly

IF OBJECT_ID ('IX_ReturnItems_OrderID') IS NOT NULL
DROP INDEX IX_ReturnItems_OrderID ON ReturnItems;
GO

CREATE NONCLUSTERED INDEX IX_ReturnItems_OrderID
ON ReturnItems(OrderID ASC)
GO

-- Noncluster Index on the foreign key WarehouseID in InventoryItems table can
-- improve performance significantly

IF OBJECT_ID ('IX_InventoryItems_WarehouseID') IS NOT NULL
DROP INDEX IX_InventoryItems_WarehouseID ON InventoryItems;
GO

CREATE NONCLUSTERED INDEX IX_InventoryItems_WarehouseID
ON InventoryItems(WarehouseID ASC)
GO
```

The script executed successfully



```
-- Noncluster Index on the foreign key WarehouseID in InventoryItems table can
-- improve performance significantly

IF OBJECT_ID ('IX_InventoryItems_WarehouseID') IS NOT NULL
DROP INDEX IX_InventoryItems_WarehouseID ON InventoryItems;
GO

CREATE NONCLUSTERED INDEX IX_InventoryItems_WarehouseID
    ON InventoryItems(WarehouseID ASC)
GO
```

```
Commands completed successfully.

Completion time: 2019-12-06T02:14:20.7874529-08:00
```

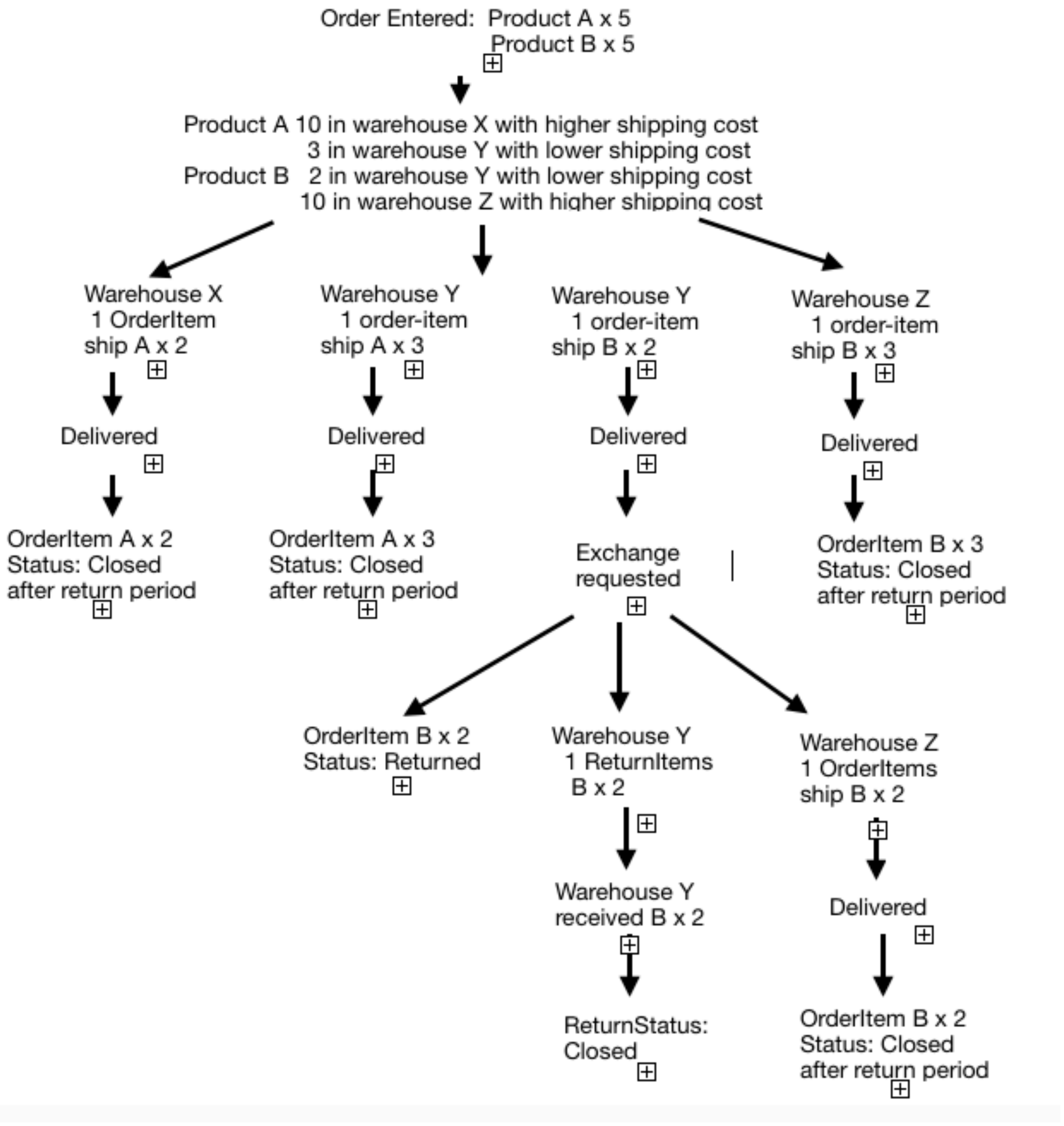
160 %
Query executed successfully.
DESKTOP-0E45JST\SQLEXPRESS ...
DESKTOP-0E45JST\tilso ...

5. Test Cases and Scenarios

The database design has been simulated with several important scenarios of actual business operations. The actually simulation in software will requires some frontend program. Without the frontend program, I tested these scenarios as following:

5.1 The transaction of an order with multiple items and returns or exchanges.

This is the main consideration influenced the design of this database. It can be explained in the case shown below.


Order Entered: Product A x 5
Product B x 5
Product A 10 in warehouse X with higher shipping cost
3 in warehouse Y with lower shipping cost
Product B 2 in warehouse Y with lower shipping cost
10 in warehouse Z with higher shipping cost
Warehouse X 1 OrderItem ship A x 2
Warehouse Y 1 order-item ship A x 3
Warehouse Y 1 order-item ship B x 2
Warehouse Z 1 order-item ship B x 3
Delivered
Delivered
Delivered
Delivered
OrderItem A x 2 Status: Closed after return period
OrderItem A x 3 Status: Closed after return period
Exchange requested
OrderItem B x 3 Status: Closed after return period
OrderItem B x 2 Status: Returned
Warehouse Y 1 ReturnItems B x 2
Warehouse Z 1 OrderItems ship B x 2
Warehouse Y received B x 2
Delivered
ReturnStatus: Closed
OrderItem B x 2 Status: Closed after return period

For one order of product A x 5 and product B x 2, seven rows of data are inserted into database. One in Orders table, five in OrderItems table and one in ReturnItems table.

Normally closed case:

Order Created > Delivered > Closed

If one item is returned:

Order Created > Delivered > Returned
Return Created > Return Received > Closed

If one item is exchanged:

Order Created > > Delivered > Returned
Return Created > Return Received > Closed
Order Created > Delivered > Closed

The inventory data in the InventoryItems table is also updated accordingly:

InStockQuantity (X, A)	3 -> 0
InStockQuantity (Y, A)	10 -> 8
InStockQuantity (Y, B)	2 -> 0 -> 2
InStockQuantity (Z, B)	10 -> 7 -> 10

5.2 The feedback of customer's review to 'change of production definition

This could be a very complicate scenario. The simple case is that the the online shopping would review and adjust the product offering after accumulated customer feedbacks/ reviews. In reality, some online shopping company actively respond to the review and sometime blind the review.

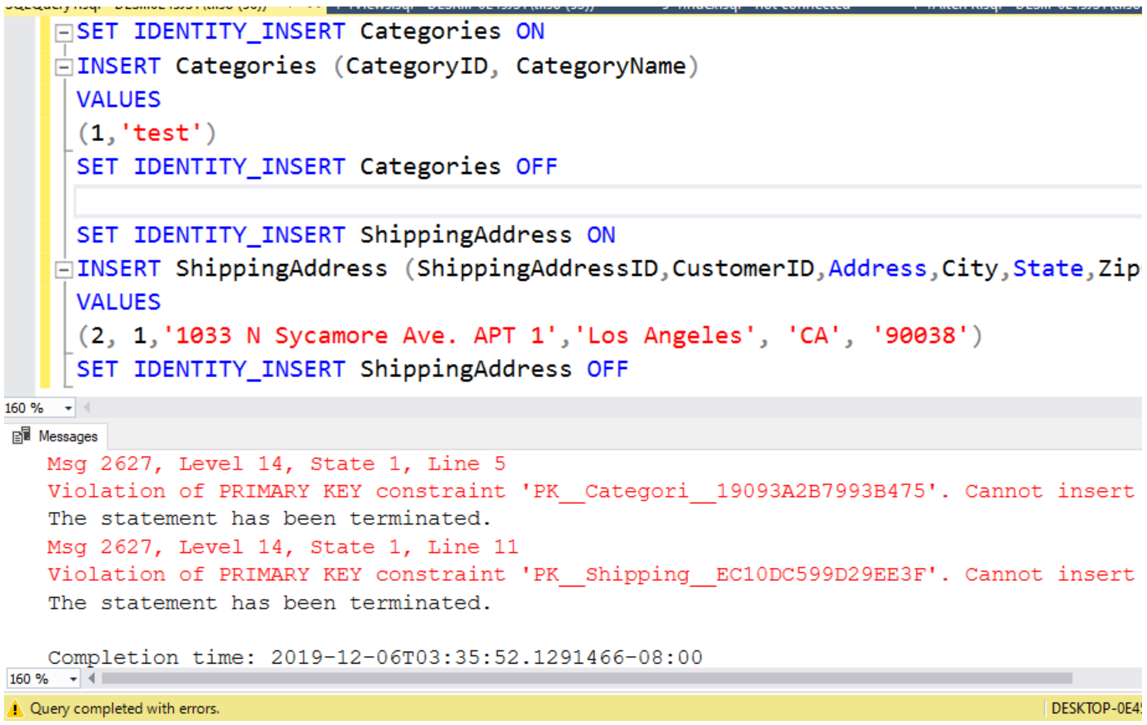
For database owner, the question is if the system should :

(a) Allow marketing to respond to review
(b) Allow marketing to update review
(c) Allow marketing to delete review.

The decision can be implemented in script, or by manual change by the IT. Since Reviews table is not a primary key table. Any manual update will not affect the data integrity.

5.3 The Referential Integrity

Foreign Key Reference constraints to prevent the referential integrity is one of the key features of relational database. The prevention of updating primary key can be handled by triggers as following.



```
SET IDENTITY_INSERT Categories ON
INSERT Categories (CategoryID, CategoryName)
VALUES
(1,'test')
SET IDENTITY_INSERT Categories OFF

SET IDENTITY_INSERT ShippingAddress ON
INSERT ShippingAddress (ShippingAddressID,CustomerID,Address,City,State,Zip
VALUES
(2, 1,'1033 N Sycamore Ave. APT 1','Los Angeles', 'CA', '90038')
SET IDENTITY_INSERT ShippingAddress OFF
```

```
Msg 2627, Level 14, State 1, Line 5
Violation of PRIMARY KEY constraint 'PK__Categori__19093A2B7993B475'. Cannot insert
The statement has been terminated.
Msg 2627, Level 14, State 1, Line 11
Violation of PRIMARY KEY constraint 'PK__Shipping__EC10DC599D29EE3F'. Cannot insert
The statement has been terminated.

Completion time: 2019-12-06T03:35:52.1291466-08:00
```

5.4 When Returned product is not accepted as a return.

When a returned product is received by warehouse or return center, it's not always the case that refund or exchange will be provided. Sometime the product is damaged by customer and be liable for the lost. These are some possible ways to handle it in case of this situation:

(a) Close the ReturnItems row with price adjusted to 0 (zero).
In this way, no credit or refund will be processed.

(b) Close the original OrderItems row, mark the ReturnItems row unacceptable. If customer want the product back, it will be send to them.

5.5 Periodical order data archive from live database to offline archive.

These three tables are closed related to transaction :

Orders
OrderItems
ReturnItems

It’s desired that closed orders should be moved to offline archive after the record is ready to retire. The criteria for a order recode to retire is :

(a) The delivered products’ return period is expired.

(b) All rows OrderItems and ReturnItems related to the Orders row are all in ‘Closed’, ‘Returned’ status.

(c) Account team has a period of time to slow-down the online shopping and make the batch move. The transaction Locking (COMMIT or ROLLBACK) should be used during the data move.

(d) A separate tag indicating the readiness could help but the data storage overhead could prevent the solution.

6. Conclusion

This database is a small but contains most of necessary components of an online shopping database system.

The practice of real world entity analysis and mapping into the database entity (tables) is quite challenging. It took me more than ten iterations to come out this version which can support the flexibility requirement. The principles of normalization and denormalization is great chance to practice the reasoning of the ‘relational database’ concept.

The implementation of views, functions, and stored procedures fill the gaps and provide more user friendly interface for frontend programmers to access the base tables. The security permissions and database roles are essential to enable the database safe and be able to run properly.

This project should be just a demonstration of how a online shopping system should be looks-like. A real-world database system could takes hundreds of peoples to implement. The capacity of the system and the speed requirement should be a complete separate deep-dive analysis.

It’s a very length project and the actual effort is more than originally expected. After the project is finish, I feel the effort is very worthy. Many thanks for review my report.