CSE581 Project Two

Shopping and Product Supply Chain Database

Yuchen Wang
905508464

A. Abstract

In this project I create a shopping database for a Target competitor, which has in store and online sales. The database stores information mainly about orders made by customers and supply chain of products. Some useful views, stored procedures, functions and triggers are also created to improve manipulation efficiency on databases.

B. Design

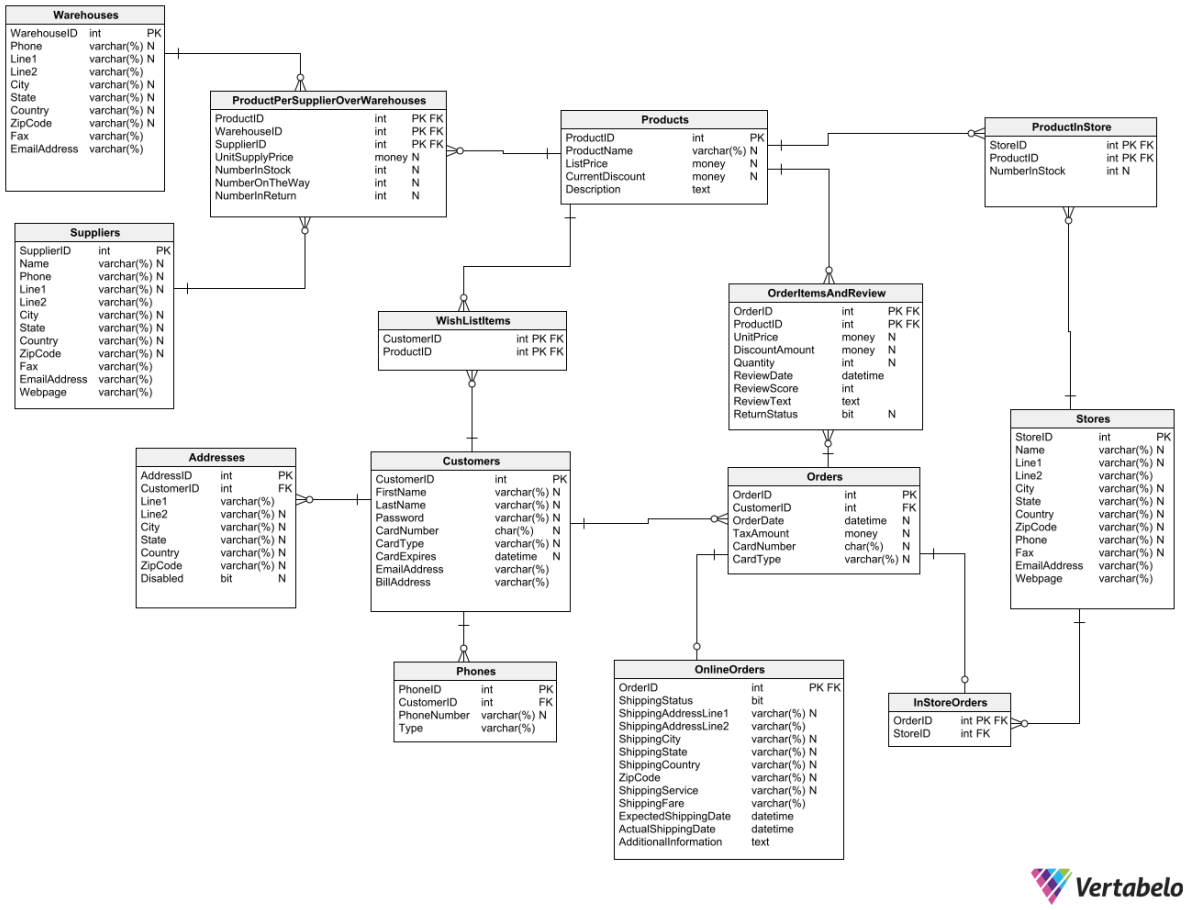
1) Information to store

1. Customers:
    a. Columns: CustomerID, FirstName, LastName, Password, Email, CardNumber, CardType, CardExpires, BillAddress
    b. Primary Key: CustomerID
    c. Related Information:
        i. Customer's wish list have been created as table 'WishListItem'
        ii. Customer's phone numbers (home, cell, business) have been created as table 'Phones'
        iii. Customer's address book (several shipping/billing addresses) have been created as table 'Addresses'
        iv. Customer's online order records have been created as table 'OnlineOrders'
        v. Customer's in-store purchase records have been created as table 'InStoreOrders'
        vi. Customer's item reviews (date, product, score, text) and returns have been stored as part of table 'OrderItemsAndReview'
    d. Design Consideration:
        i. Customer may have multiple addresses, thus create the table 'Addresses' having one to many relationship with 'Customers' table
2. Products:
    a. Columns: ProductID, ProductName, ListPrice, CurrentDiscount. Description
    b. Primary Key: ProductID
    c. Related Information:
        i. Product's customer ratings (text, score, date) has been stored as part of table 'itemOrderAndReview'
3. Suppliers:

    a. Columns: SupplierID, Name, Line1, Line2, City, State, Country, ZipCode, Phone, Fax, Email, Webpage
    b. Primary Key: SupplierID
    c. Related Information:
        i. Supplier's product information (product, number of products available, unit price) has been stored as part of table 'productPerSupplierOverWarehouses', because suppliers may also have multiple products in various warehouses.
4. Warehouses:
    a. Columns: WarehouseID, Line1, Line2, City, State, Country, ZipCode, Phone, Fax, Email
    b. Primary Key: WarehouseID
    c. Related Information
        i. Warehouse's stored products (product, number in stock, number on the way, number in return) have been stored as part of table 'productPerSupplierOverWarehouses', because multiple warehouses may have the same product and reverse the same. Hence, putting stored product information into the linking table of warehouse and product is necessary.
5. Stores:
    a. Columns: StoreID, Name, Line1, Line2, City, State, Country, ZipCode, Phone, Fax, Email, Webpage
    b. Primary Key: StoreID
    c. Related Information
        i. Store's on-selling products (product, price, quantity) have been created as table 'productInStore', because there could be multiple kinds of on-selling products in a store.
6. OnlineOrders:
    a. Columns: OrderID, ShippingStatus, ShippingAddressLine1, ShippingAddressLine2, ShippingCity, ShippingState, ShippingCountry, ZipCode, ShippingService, ShippingFare, ExpectedShippingDate, ActualShippingDate, AdditionalInformation
    b. Primary Key: OrderID
    c. Foreign Key: OrderID
    d. Related Information:
        i. order status (ready, shipped, delivered, returned), order date, order items (product, quantity, unit price, total price) have been stored as part of 'onlineOrders's parent table 'orders'
7. ProductPerSupplierOverWarehouses
    a. Columns: ProductID, WarehouseID, SupplierID, UnitSupplyPrice, NumberInStock, NumberOnTheWay, NumberInReturn
    b. Primary Key: ProductID; WarehouseID; SupplierID
    c. Foreign Key: ProductID; WarehouseID; SupplierID
8. WishListItem
    a. Columns: CustomerID; ProductID
    b. Primary key: CustomerID; ProductID
    c. Foreign key: CustomerID; ProductID
9. Addresses

    a. Columns: AddressID, CustomerID, Line1, Line2, City, State, Country, ZipCode, Disabled
    b. Primary Key: AddressID
    c. Foreign Key: CustomerID
10. Orders
    a. Columns: OrderID, CustomerID, TaxAmount, OrderDate, CardNumber, CardType
    b. Primary Key: OrderID
    c. Foreign Key: CustomerID
11. OrderItemsAndReview
    a. Columns: OrderID, ProductID, Quantity, UnitPrice, DiscountAmount, ReviewDate, ReviewScore, ReviewText, ReturnStatus
    b. Primary Key: OrderID, ProductID
    c. Foreign Key: OrderID. ProductID
12. InStoreOrders
    a. Columns: OrderID, StoreID
    b. Primary Key: OrderID, StoreID
    c. Foreign Key: OrderID, StoreID
    d. Related Information:
        i. order status (ready, shipped, delivered, returned), order date, order items (product, quantity, unit price, total price) have been stored as part of ‘InstoreOrders’s parent table ‘orders’
13. ProductInStore
    a. Columns: StoreID, ProductID, NumberInStock
    b. Primary Key: StoreID, ProductID
    c. Foreign Key: StoreID, ProductID
14. Phones
    a. Columns: PhoneID, CustomerID, PhoneNumber, Type
    b. Primary Key: PhoneID
    c. Foreign Key: CustomerID

2) E/R diagram (please use draw.io, Visual Paradigm, Vertabelo, MS Visio, or any similar tool),



3) Business logics and transactions to consider, and how to incorporate them into the design

- Quantity of a product in a store should not be more than 100
    - The trigger TooManyInStock take care this problem
- Customer may have multiple addresses
    - Addresses table take care this problem by having one to many relationship with Customers table
- Customer may have multiple phone numbers
    - Phones table take care this problem by having one to many relationship with Customers table
- Product supply prices vary with different suppliers and warehouses.
    - ProductPerSupplierOverWarehouses table take care this problem by having one to many relationship with Products table
- Each online order has single shipping address
    - OnlineOrders table take care this problem
- Each online order can contain multiple products
    - OrderItemAndReview take care this problem by having one to many relationship with OnlineOrders table's parent table, Orders table

- Multiple warehouses may have the same product
    - ProductPerSupplierOverWarehouses table take care this problem by having one to many relationship with Warehouses table
- Originating warehouse for a shipment is chosen based on the shipping time and cost (determined by the zip codes of the originating warehouse and the shipping address)
    - As both Warehouses and OnlineOrders tables have zip code columns and address columns, we can address this problem by doing calculations on zip codes and addresses.
- Suppliers may also have multiple products in various warehouses.
    - ProductPerSupplierOverWarehouses table take care this problem by having one to many relationship with both Warehouses and Supplier table
- Stores can also place orders to deliver products from warehouses to stores for in store purchases
    - If a product is out of stock in a store, the store can order the product from a warehouse by using ProductID on ProductPerSupplierOverWarehouses table to check which warehouses have the required product.
- Product's list price is relative to the supply price offered by suppliers
    - ProductPerSupplierOverWarehouses table take care this problem by having UnitSupplyPrice solemn

4). Business reports to consider.

- Most sold item
    - Can be found out by using function 'CheckProductSelling' to get the sale of each product, and then find out the most sold one
- Most popular store
    - Can be found out by using the function 'CheckProductOnSellingStore' to get the product sale at each store, then sum the sale of products to see which store sells the most products.

C. Implementation

1) Database Creation

```
USE master;
GO

IF  DB_ID('ProjectTwo') IS NOT NULL
   DROP DATABASE ProjectTwo;
GO

CREATE DATABASE ProjectTwo;
```

```
GO

USE projectTwo;

-- create the tables for the database

CREATE TABLE Products (
	ProductID	INT		PRIMARY KEY	IDENTITY,
	ProductName	VARCHAR(255)	NOT NULL	UNIQUE,
	ListPrice		MONEY		NOT
NULL,
	CurrentDiscount	MONEY		NOT NULL	DEFAULT 0.00,
	Description	TEXT		DEFAULT NULL,
);

CREATE TABLE Customers (
	CustomerID	INT		NOT NULL	PRIMARY KEY	IDENTITY,
	FirstName		VARCHAR(60)	NOT NULL,
	LastName		VARCHAR(60)	NOT NULL,
	Password		VARCHAR(60)	NOT NULL,
	CardType		VARCHAR(50)	NOT NULL,
	CardNumber		CHAR(16)	NOT NULL,
	CardExpires		DATETIME	NOT NULL,
	EmailAddress		VARCHAR(255)	DEFAULT NULL,
	BillingAddressID	INT		DEFAULT NULL
);

CREATE TABLE Warehouses (
	WarehouseID	INT		PRIMARY
KEY	IDENTITY,
	Phone		VARCHAR(12)	NOT NULL,
	Line1		VARCHAR(60)	NOT NULL,
	Line2		VARCHAR(60)		DEFAULT NULL,
	City		VARCHAR(40)	NOT NULL,
	State		VARCHAR(2)	NOT NULL,
	Country		VARCHAR(40)	NOT NULL,
	ZipCode		VARCHAR(10)	NOT NULL,
	Fax		VARCHAR(255)
DEFAULT NULL,
	EmailAddress	VARCHAR(255)		DEFAULT NULL,
);

CREATE TABLE Suppliers (
	SupplierID	INT		PRIMARY KEY
IDENTITY,
	Name		VARCHAR(40)	NOT NULL,
	Phone		VARCHAR(12)	NOT NULL,
	Line1		VARCHAR(60)	NOT NULL,
```

```
        Line2           VARCHAR(60)             DEFAULT NULL,
        City            VARCHAR(40)   NOT NULL,
        State           VARCHAR(2)    NOT NULL,
        Country                       VARCHAR(40)   NOT NULL,
        ZipCode         VARCHAR(10)   NOT NULL,
        Fax                           VARCHAR(255)
DEFAULT NULL,
        EmailAddress    VARCHAR(255)                                DEFAULT NULL,
        Webpage                       VARCHAR(255)
DEFAULT NULL,
);

CREATE TABLE Stores (
        StoreID         INT                                         PRIMARY KEY
IDENTITY,
        Name                    VARCHAR(40)   NOT NULL,
        Phone                   VARCHAR(12)         NOT NULL,
        Line1           VARCHAR(60)   NOT NULL,
        Line2           VARCHAR(60)             DEFAULT NULL,
        City            VARCHAR(40)   NOT NULL,
        State           VARCHAR(2)    NOT NULL,
        Country                       VARCHAR(40)   NOT NULL,
        ZipCode         VARCHAR(10)   NOT NULL,
        Fax                           VARCHAR(255)
DEFAULT NULL,
        EmailAddress    VARCHAR(255)                                DEFAULT NULL,
        Webpage                       VARCHAR(255)
DEFAULT NULL
);

CREATE TABLE Orders (
  OrderID         INT          PRIMARY KEY  IDENTITY,
  CustomerID      INT          REFERENCES Customers (CustomerID),
  TaxAmount       MONEY        NOT NULL,
  OrderDate       DATETIME     NOT NULL,
  CardType        VARCHAR(50)  NOT NULL,
  CardNumber      CHAR(16)     NOT NULL,
);


CREATE TABLE ProductPerSupplierOverWarehouses (
        SupplierID      INT                                         FOREIGN KEY
REFERENCES Suppliers (SupplierID),
        WarehouseID     INT                                                     FOREIGN
KEY REFERENCES Warehouses (WarehouseID),
   ProductID       INT                                              FOREIGN KEY
REFERENCES Products (ProductID),
   UnitSupplyPrice    MONEY                                     NOT NULL,
```

```
	NumberInStock	INT		NOT NULL,
	NumberOnTheWay	INT		NOT NULL,
	NumberInReturn	INT		NOT NULL,
	CONSTRAINT pkProductPerSupplierOverWarehouses PRIMARY KEY (SupplierID,
ProductID, WarehouseID)
);


CREATE TABLE ProductInStore (
	ProductID	INT		FOREIGN KEY
REFERENCES Products (ProductID),
	StoreID		INT		FOREIGN
KEY REFERENCES Stores (StoreID),
	NumberInStock	INT		NOT NULL,
	CONSTRAINT pkProductInStore PRIMARY KEY (StoreID, ProductID)
);



CREATE TABLE WishListItem (
  CustomerID	INT	FOREIGN KEY REFERENCES Customers (CustomerID),
  ProductID	INT	FOREIGN KEY REFERENCES Products (ProductID),
  CONSTRAINT pkWishListItem PRIMARY KEY (CustomerID, ProductID)
);

CREATE TABLE OrderItemsAndReview (
  OrderID	INT	REFERENCES Orders (OrderID),
  ProductID	INT	REFERENCES Products (ProductID),
  UnitPrice	MONEY		NOT NULL,
  DiscountAmount	MONEY		NOT NULL,
  Quantity	INT	NOT NULL,
  ReviewScore	INT	DEFAULT NULL,
  ReviewDate	DATETIME	DEFAULT NULL,
  ReviewTEXT	TEXT	DEFAULT NULL,
  ReturnStatus	BIT		NOT NULL	DEFAULT 0,
  CONSTRAINT pkOrderItemsAndReview PRIMARY KEY (OrderID, ProductID)
);


CREATE TABLE Addresses (
  AddressID	INT	NOT NULL	PRIMARY KEY	IDENTITY,
  CustomerID	INT	REFERENCES Customers (CustomerID),
  Line1	VARCHAR(60)	NOT NULL,
  Line2	VARCHAR(60)		DEFAULT NULL,
  City	VARCHAR(40)	NOT NULL,
  State	VARCHAR(2)	NOT NULL,
  Country	VARCHAR(40)	NOT NULL,
  ZipCode	VARCHAR(10)	NOT NULL,
```

```
  Disabled          INT           NOT NULL      DEFAULT 0
);

CREATE TABLE Phones (
          PhoneID                         INT                                   PRIMARY KEY
IDENTITY,
          CustomerID        INT                                REFERENCES Customers
(CustomerID),
          PhoneNumber           VARCHAR(12)                                   NOT NULL,
    Type                            VARCHAR(40)              DEFAULT NULL
);

CREATE TABLE InStoreOrders (
  OrderID        INT         FOREIGN KEY REFERENCES Orders (OrderID),
  StoreID        INT         FOREIGN KEY REFERENCES Stores (StoreID),
  CONSTRAINT pkInStoreOrders PRIMARY KEY (OrderID, StoreID)
);

CREATE TABLE OnlineOrders (
  OrderID                                                INT        FOREIGN KEY REFERENCES
Orders (OrderID),
  ShippingStatus                                 INT           NOT NULL      DEFAULT 0,
  ShippingAddressLine1          VARCHAR(60)    NOT NULL,
  ShippingAddressLine2          VARCHAR(60)    DEFAULT NULL,
  ShippingCity                                   VARCHAR(40)    NOT NULL,
  ShippingState                                          VARCHAR(2)     NOT NULL,
  ShippingCountry                                VARCHAR(40)    NOT NULL,
  ZipCode                                                VARCHAR(10)    NOT NULL,
  ShippingService                                VARCHAR(40)    NOT NULL,
  ShippingFare                                   VARCHAR(40)    DEFAULT NULL,
  ExceptedShippingDate                           DATETIME       DEFAULT NULL,
  ActualShippingDate                     DATETIME       DEFAULT NULL,
  AdditionalInformation         TEXT           DEFAULT NULL,
  CONSTRAINT pkOnlineOrders PRIMARY KEY (OrderID)
);
```

2) Data Insertation

```
USE projectTwo;

-- Done
DELETE OnlineOrders;
DELETE InStoreOrders;
DELETE ProductInStore;
```

```
DELETE ProductPerSupplierOverWarehouses;
DELETE WishListItem;
DELETE OrderItemsAndReview;
DELETE Phones;
DELETE Addresses;
DELETE Products;
DELETE Orders;
DELETE Customers;
DELETE Suppliers;
DELETE Warehouses;
DELETE Stores;

SET IDENTITY_INSERT Stores ON;

INSERT INTO  Stores (StoreID, Name, Phone, Line1, Line2, City, State, Country, ZipCode)
VALUES
(1, 'Syracuse Store', '201-253-4272', '105 East Ridgewood Ave.', '', 'Paramus', 'NJ',
'USA','07652'),
(2, 'Buffalo Store', '201-553-4472', '23 Rosewood Rd.', '', 'Woodcliff Lake', 'NJ',
'USA','07677'),
(3, 'NYC Store', '402-196-1876', '16285 Wendell St.', '', 'Omaha', 'NE', 'USA','68135'),
(4, 'Ithaca Store', '503-654-1292', '19470 NW Cornell Rd.', '', 'Beaverton', 'OR',
'USA','97006');

SET IDENTITY_INSERT Stores OFF;

SET IDENTITY_INSERT Warehouses ON;

INSERT INTO  Warehouses  (WarehouseID, Phone, Line1, Line2, City, State, Country,
ZipCode) VALUES
(1, '201-253-4272', '105 East Ridgewood Ave.', '', 'Paramus', 'NJ', 'USA','07652'),
(2, '201-553-4472', '23 Rosewood Rd.', '', 'Woodcliff Lake', 'NJ', 'USA','07677'),
(3, '402-196-1876', '16285 Wendell St.', '', 'Omaha', 'NE', 'USA','68135'),
(4, '503-654-1292', '19470 NW Cornell Rd.', '', 'Beaverton', 'OR', 'USA','97006');

SET IDENTITY_INSERT Warehouses OFF;

SET IDENTITY_INSERT Suppliers ON;

INSERT INTO Suppliers (SupplierID, Name, Phone, Line1, Line2, City, State, Country,
ZipCode) VALUES
(1, 'SuperV', '201-653-4272', '100 East Ridgewood Ave.', '', 'Paramus', 'NJ', 'USA','07652'),
(2, 'BigMouth', '201-553-4472', '21 Rosewood Rd.', '', 'Woodcliff Lake', 'NJ', 'USA','07677'),
(3, 'KG', '402-896-2876', '16285 Wendell St.', '', 'Omaha', 'NE', 'USA','68135'),
(4, 'Amazon', '503-654-1292', '19270 NW Cornell Rd.', '', 'Beaverton', 'OR', 'USA','97006');

SET IDENTITY_INSERT Suppliers OFF;
```

SET IDENTITY_INSERT Products ON;

INSERT INTO Products (ProductID, ProductName, Description, ListPrice, CurrentDiscount) VALUES
(1, 'Fender Stratocaster', 'The Fender Stratocaster is the electric guitar design that changed the world. New features include a tinted neck, parchment pickguard and control knobs, and a "70s-style logo. Includes select alder body, 21-fret maple neck with your choice of a rosewood or maple fretboard, 3 single-coil pickups, vintage-style tremolo, and die-cast tuning keys. This guitar features a thicker bridge block for increased sustain and a more stable point of contact with the strings. At this low price, why play anything but the real thing?\r\n\r\nFeatures:\r\n\r\n* New features:\r\n* Thicker bridge block\r\n* 3-ply parchment pick guard\r\n* Tinted neck', '699.00', '30.00'),
(2, 'Gibson Les Paul', 'This Les Paul guitar offers a carved top and humbucking pickups. It has a simple yet elegant design. Cutting-yet-rich tone?the hallmark of the Les Paul?pours out of the 490R and 498T Alnico II magnet humbucker pickups, which are mounted on a carved maple top with a mahogany back. The faded finish models are equipped with BurstBucker Pro pickups and a mahogany top. This guitar includes a Gibson hardshell case (Faded and satin finish models come with a gig bag) and a limited lifetime warranty.\r\n\r\nFeatures:\r\n\r\n* Carved maple top and mahogany back (Mahogany top on faded finish models)\r\n* Mahogany neck, "59 Rounded Les Paul\r\n* Rosewood fingerboard (Ebony on Alpine white)\r\n* Tune-O-Matic bridge with stopbar\r\n* Chrome or gold hardware\r\n* 490R and 498T Alnico 2 magnet humbucker pickups (BurstBucker Pro on faded finish models)\r\n* 2 volume and 2 tone knobs, 3-way switch', '1199.00', '30.00'),
(3, 'Gibson SG', 'This Gibson SG electric guitar takes the best of the "62 original and adds the longer and sturdier neck joint of the late "60s models. All the classic features you"d expect from a historic guitar. Hot humbuckers go from rich, sweet lightning to warm, tingling waves of sustain. A silky-fast rosewood fretboard plays like a dream. The original-style beveled mahogany body looks like a million bucks. Plus, Tune-O-Matic bridge and chrome hardware. Limited lifetime warranty. Includes hardshell case.\r\n\r\nFeatures:\r\n\r\n* Double-cutaway beveled mahogany body\r\n* Set mahogany neck with rounded "50s profile\r\n* Bound rosewood fingerboard with trapezoid inlays\r\n* Tune-O-Matic bridge with stopbar tailpiece\r\n* Chrome hardware\r\n* 490R humbucker in the neck position\r\n* 498T humbucker in the bridge position\r\n* 2 volume knobs, 2 tone knobs, 3-way switch\r\n* 24-3/4" scale', '2517.00', '52.00'),
(4, 'Yamaha FG700S', 'The Yamaha FG700S solid top acoustic guitar has the ultimate combo for projection and pure tone. The expertly braced spruce top speaks clearly atop the rosewood body. It has a rosewood fingerboard, rosewood bridge, die-cast tuners, body and neck binding, and a tortoise pickguard.\r\n\r\nFeatures:\r\n\r\n* Solid Sitka spruce top\r\n* Rosewood back and sides\r\n* Rosewood fingerboard\r\n* Rosewood bridge\r\n* White/black body and neck binding\r\n* Die-cast tuners\r\n* Tortoise pickguard\r\n* Limited lifetime warranty', '489.99', '38.00');

SET IDENTITY_INSERT Products OFF;

SET IDENTITY_INSERT Customers ON;

INSERT INTO Customers (CustomerID, EmailAddress, Password, FirstName, LastName, BillingAddressID, CardType, CardNumber, CardExpires) VALUES

```
(1, 'allan.sherwood@yahoo.com', 'c44321e51ec184a2f739318639cec426de774451', 'Allan',
'Sherwood', 1, 'Visa', '4111111111111111', '01/04/2018'),
(2, 'barryz@gmail.com', 'd9e03c0b34c57d034edda004ec8bae5d53667e36', 'Barry',
'Zimmer', 1, 'Visa', '4012888888881881', '01/08/2020'),
(3, 'christineb@solarone.com', '13ef4f968693bda97a898ece497da087b182808e', 'Christine',
'Brown', 1, 'American Express', '3782822463100005', '01/02/2017'),
(4, 'david.goldstein@hotmail.com', '2a367cbb171d78d293f40fd7d1defb31e3fb1728', 'David',
'Goldstein', 1, 'MasterCard', '5555555555554444', '01/12/2018');

SET IDENTITY_INSERT Customers OFF;

SET IDENTITY_INSERT Addresses ON;

INSERT INTO Addresses (AddressID, CustomerID, Line1, Line2, City, State, Country,
ZipCode, Disabled) VALUES
(1, 1, '100 East Ridgewood Ave.', '', 'Paramus', 'NJ', 'USA','07652', 0),
(2, 1, '21 Rosewood Rd.', '', 'Woodcliff Lake', 'NJ', 'USA','07677', 0),
(3, 2, '16285 Wendell St.', '', 'Omaha', 'NE', 'USA','68135', 0),
(4, 3, '19270 NW Cornell Rd.', '', 'Beaverton', 'OR', 'USA','97006', 0);

SET IDENTITY_INSERT Addresses OFF;

SET IDENTITY_INSERT Phones ON;

INSERT INTO Phones (PhoneID, CustomerID, PhoneNumber, Type) VALUES
(1, 1, '201-653-4472', 'mobile'),
(2, 1, '201-653-4472', 'home'),
(3, 2, '402-896-2576', 'mobile'),
(4, 3, '503-654-1291', 'mobile');

SET IDENTITY_INSERT Phones OFF;


SET IDENTITY_INSERT Orders ON;

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TaxAmount, CardType,
CardNumber) VALUES
(1, 1, '2016-03-28 09:40:28.000', 58.75, 'Visa', '4111111111111111'),
(2, 2, '2016-03-28 11:23:20.000', 21.27, 'Visa', '4012888888881881'),
(3, 1, '2016-03-29 09:44:58.000', 102.29,'Visa', '4111111111111111'),
(4, 3, '2016-03-30 15:22:31.000', 117.50, 'American Express', '3782822463100005'),
(5, 2, '2016-03-28 09:40:28.000', 81.75, 'Visa', '4111111111111111'),
(6, 4, '2016-03-28 11:23:20.000', 73.17, 'Visa', '4012888888881881'),
(7, 2, '2016-03-29 09:44:58.000', 1067.54,'Visa', '4111111111111111'),
(8, 3, '2016-03-30 15:22:31.000', 23.40, 'American Express', '3782822463100005');

SET IDENTITY_INSERT Orders OFF;
```

```
--SET IDENTITY_INSERT OrderItemsAndReview ON;

INSERT INTO OrderItemsAndReview (OrderID, ProductID, UnitPrice, DiscountAmount,
Quantity) VALUES
(1, 1, '1199.00', '359.70', 1),
(1, 2, '489.99', '186.20', 3),
(1, 3, '2517.00', '1308.84', 7),
(1, 4, '415.00', '161.85', 6),
(2, 1, '1199.00', '359.70', 12),
(2, 2, '489.99', '186.20', 4),
(2, 3, '2517.00', '1308.84', 1),
(3, 4, '415.00', '161.85', 22),
(3, 1, '1199.00', '359.70', 1),
(3, 2, '489.99', '186.20', 16),
(4, 1, '2517.00', '1308.84', 1),
(4, 3, '415.00', '161.85', 31),
(5, 1, '1199.00', '359.70', 1),
(5, 2, '489.99', '186.20', 3),
(5, 3, '2517.00', '1308.84', 7),
(6, 4, '415.00', '161.85', 6),
(6, 1, '1199.00', '359.70', 12),
(7, 2, '489.99', '186.20', 4),
(7, 3, '2517.00', '1308.84', 1),
(7, 4, '415.00', '161.85', 22),
(8, 1, '1199.00', '359.70', 1),
(8, 2, '489.99', '186.20', 16),
(8, 3, '2517.00', '1308.84', 1),
(8, 4, '415.00', '161.85', 31);

--SET IDENTITY_INSERT OrderItemsAndReview OFF;

--SET IDENTITY_INSERT WishListItem ON;

INSERT INTO WishListItem (CustomerID, ProductID) VALUES
(1, 1),
(2, 2),
(3, 3),
(1, 3);

--SET IDENTITY_INSERT WishListItem OFF;

--SET IDENTITY_INSERT ProductPerSupplierOverWarehouses ON;

INSERT INTO ProductPerSupplierOverWarehouses (WarehouseID, ProductID, SupplierID,
UnitSupplyPrice, NumberInStock, NumberOnTheWay, NumberInReturn) VALUES
(1, 1, 2, '1100.00', 423, 200, 40),
(2, 2, 3, '450.99', 421, 300, 50),
(3, 3, 1, '2500.00', 782, 700, 35),
```

```
(4, 3, 1, '422.00', 561, 100, 70);

--SET IDENTITY_INSERT ProductPerSupplierOverWarehouses OFF;

--SET IDENTITY_INSERT ProductInStore ON;
INSERT INTO ProductInStore (StoreID, ProductID, NumberInStock) VALUES
(1, 1, 27),
(2, 2, 105),
(3, 3, 33),
(1, 3, 72);

--SET IDENTITY_INSERT ProductInStore OFF;

--SET IDENTITY_INSERT WishListItem ON;

INSERT INTO InStoreOrders (OrderID, StoreID) VALUES
(1, 1),
(3, 2),
(4, 3),
(2, 3);

--SET IDENTITY_INSERT WishListItem OFF;


--SET IDENTITY_INSERT OnlineOrders ON;

INSERT INTO  OnlineOrders (OrderID, ShippingAddressLine1, ShippingAddressLine2,
ShippingCity, ShippingState, ShippingCountry, ZipCode, ShippingService, ShippingStatus)
VALUES
(5, '105 East Ridgewood Ave.', '', 'Paramus', 'NJ', 'USA','07652', 'UPS', 0),
(6, '23 Rosewood Rd.', '', 'Woodcliff Lake', 'NJ', 'USA','07677', 'EMS', 1),
(7, '16285 Wendell St.', '', 'Omaha', 'NE', 'USA','68135', 'US Post', 1),
(8, '19470 NW Cornell Rd.', '', 'Beaverton', 'OR', 'USA','97006', 'UPS', 1);

--SET IDENTITY_INSERT OnlineOrders OFF;
```

3) Views

InStoreOrdersView

```
-- Check all in store orders

USE ProjectTwo;

GO
```

```
DROP VIEW InStoreOrdersView;

GO

CREATE VIEW InStoreOrdersView
AS
SELECT Customers.CustomerID, FirstName + ' ' + LastName AS CustomerName,
Customers.EmailAddress,
OrderDate, TaxAmount, Orders.CardType, Orders.CardNumber, Stores.Name AS
StoreName,
Stores.City + ', ' + Stores.State + ', ' + Stores.Country AS StoreAddress
FROM Customers JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
JOIN InStoreOrders
ON Orders.OrderID = InStoreOrders.OrderID
JOIN Stores
ON Stores.StoreID = InStoreOrders.StoreID

GO

SELECT *
FROM InStoreOrdersView;
```

OnlineOrdersView

```
-- Check all the online orders

USE ProjectTwo;

GO

DROP VIEW OnlineOrdersView;

GO

CREATE VIEW OnlineOrdersView
AS
SELECT Customers.CustomerID, FirstName + ' ' + LastName AS CustomerName,
Customers.EmailAddress,
OrderDate, TaxAmount, Orders.CardType, Orders.CardNumber,
ShippingAddressLine1 + ', ' + ShippingCity + ', ' + ShippingState + ', ' + ShippingCountry + ', '
+ ZipCode AS ShippingAddress,
ShippingService, ShippingStatus
FROM Customers JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
JOIN OnlineOrders
```

```
ON Orders.OrderID = OnlineOrders.OrderID;

GO

SELECT *
FROM OnlineOrdersView;
```

ProductSupplyChainView

```
-- Use to see the product supply chain information

USE ProjectTwo;

GO

DROP VIEW ProductSupplyChain;

GO

CREATE VIEW ProductSupplyChain
AS
SELECT ProductName,
ProductPerSupplierOverWarehouses.ProductID,
ProductPerSupplierOverWarehouses.WarehouseID,
ProductPerSupplierOverWarehouses.SupplierID,
Description, ListPrice, CurrentDiscount
UnitSupplyPrice, NumberInStock, NumberOnTheWay, NumberInReturn,
Suppliers.Name AS SuppliersName, Suppliers.Phone AS SuppliersPhone, Suppliers.City + ',
' + Suppliers.State + ', ' +  Suppliers.Country AS SuppliersAddress,
Warehouses.Phone AS WarehousesPhone, Warehouses.City + ', ' + Warehouses.State + ', '
+ Warehouses.Country AS WarehouseAddress
FROM
ProductPerSupplierOverWarehouses LEFT JOIN Products
ON ProductPerSupplierOverWarehouses.ProductID = Products.ProductID
LEFT JOIN Suppliers
ON ProductPerSupplierOverWarehouses.SupplierID = Suppliers.SupplierID
LEFT JOIN Warehouses
ON ProductPerSupplierOverWarehouses.WarehouseID = Warehouses.WarehouseID

GO

SELECT *
FROM ProductSupplyChain;
```

AllTheOrdersView

```
-- Check all the order's details, including online orders and in-store orders

USE ProjectTwo;

GO

DROP VIEW AllTheOrdersView;

GO

CREATE VIEW AllTheOrdersView
AS
SELECT Orders.OrderID, Orders.CustomerID,
OrderDate, TaxAmount, Orders.CardType, Orders.CardNumber,
ShippingAddressLine1 + ', ' + ShippingCity + ', ' + ShippingState + ', ' + ShippingCountry + ', '
+ OnlineOrders.ZipCode AS OnlineOrderShippingAddress,
ShippingService, ShippingStatus,
Stores.Name AS StoreName,
Stores.City + ', ' + Stores.State + ', ' + Stores.Country AS StoreAddress
FROM Orders
LEFT JOIN OnlineOrders
ON Orders.OrderID = OnlineOrders.OrderID
LEFT JOIN InStoreOrders
ON Orders.OrderID = InStoreOrders.OrderID
LEFT JOIN Stores
ON Stores.StoreID = InStoreOrders.StoreID

GO

SELECT *
FROM AllTheOrdersView;
```

4) Store Procedures

spCheckProductSupplyChain

```
-- check supply chain for a specific product

USE ProjectTwo;

DROP PROC spCheckProductSupplyChain;
```

```
GO


CREATE PROC spCheckProductSupplyChain
	@ID int = 0
AS
IF @ID > 0
	BEGIN
		SELECT ProductName, ProductPerSupplierOverWarehouses.WarehouseID,
ProductPerSupplierOverWarehouses.SupplierID,
		Description, ListPrice, CurrentDiscount
		UnitSupplyPrice, NumberInStock, NumberOnTheWay, NumberInReturn,
		Suppliers.Name AS SuppliersName, Suppliers.Phone AS SuppliersPhone,
Suppliers.City + ', ' + Suppliers.State + ', ' +  Suppliers.Country AS SuppliersAddress,
		Warehouses.Phone AS WarehousesPhone, Warehouses.City + ', ' +
Warehouses.State + ', ' + Warehouses.Country AS WarehouseAddress
		FROM
		ProductPerSupplierOverWarehouses LEFT JOIN Products
		ON ProductPerSupplierOverWarehouses.ProductID = Products.ProductID
		LEFT JOIN Suppliers
		ON ProductPerSupplierOverWarehouses.SupplierID = Suppliers.SupplierID
		LEFT JOIN Warehouses
		ON ProductPerSupplierOverWarehouses.WarehouseID =
Warehouses.WarehouseID
		WHERE Products.ProductID = @ID
	END
ELSE
	PRINT 'The Input Product ID is not valid!!';

GO

EXEC spCheckProductSupplyChain @ID = 3;
```

spCheckOrderCost

```
USE ProjectTwo;

DROP PROC spCheckOrderCost;

GO

CREATE PROC spCheckOrderCost
	@ID int = 0
AS
```

```
IF @ID > 0
	BEGIN
		DECLARE @Cost int;
		DECLARE @Tax int;
		SELECT @Cost = SUM(UnitPrice * Quantity)
			FROM OrderItemsAndReview
			WHERE OrderItemsAndReview.OrderID = @ID
			GROUP BY OrderID;
		SELECT @Tax = TaxAmount
			FROM Orders
			WHERE Orders.OrderID = @ID
		PRINT 'The total cost of order ' + CAST(@ID AS VARCHAR) + ' is ' +
CAST((@COST - @TAX) AS VARCHAR);
	END;
ELSE
	PRINT 'The Input Order ID is not valid!!';

GO

EXEC spCheckOrderCost @ID = 1;
```

spCheckCustomerOnlineOrders

```
-- check online order details for a specific customer

USE ProjectTwo;

DROP PROC spCheckCustomerOnlineOrders;

GO

CREATE PROC spCheckCustomerOnlineOrders
	@ID int = 0
AS
IF @ID > 0
	BEGIN
		SELECT Customers.CustomerID, FirstName + ' ' + LastName AS
CustomerName, Customers.EmailAddress,
		OrderDate, TaxAmount, Orders.CardType, Orders.CardNumber,
		ShippingAddressLine1 + ', ' + ShippingCity + ', ' + ShippingState + ', ' +
ShippingCountry + ', ' + ZipCode AS ShippingAddress,
		ShippingService, ShippingStatus
		FROM Customers JOIN Orders
		ON Customers.CustomerID = Orders.CustomerID
		JOIN OnlineOrders
		ON Orders.OrderID = OnlineOrders.OrderID
```

```
		WHERE Customers.CustomerID = @ID;
	END
ELSE
	PRINT 'The Input Customer ID is not valid!!';

GO

EXEC spCheckCustomerOnlineOrders @ID = 2;
```

spCheckCustomerInStoreOrders

```
-- check all the in-store order of a specific customer

USE ProjectTwo;

DROP PROC spCheckCustomerInStoreOrders;

GO

CREATE PROC spCheckCustomerInStoreOrders
	@ID int = 0
AS
IF @ID > 0
	BEGIN
		SELECT Customers.CustomerID, FirstName + ' ' + LastName AS
CustomerName, Customers.EmailAddress,
		OrderDate, TaxAmount, Orders.CardType, Orders.CardNumber, Stores.Name
AS StoreName,
		Stores.City + ', ' + Stores.State + ', ' + Stores.Country AS StoreAddress
		FROM Customers JOIN Orders
		ON Customers.CustomerID = Orders.CustomerID
		JOIN InStoreOrders
		ON Orders.OrderID = InStoreOrders.OrderID
		JOIN Stores
		ON Stores.StoreID = InStoreOrders.StoreID
		WHERE Customers.CustomerID = @ID;
	END;
ELSE
	PRINT 'The Input Customer ID is not valid!!';

GO

EXEC spCheckCustomerInStoreOrders @ID = 1;
```

5) User Defined Function

fnCheckProductOnSellingStore

```
USE ProjectTwo;

DROP FUNCTION fnCheckProductOnSellingStore;

GO

CREATE FUNCTION fnCheckProductOnSellingStore(@ID INT)
RETURNS TABLE
		RETURN(
			SELECT Products.ProductName, Products.ProductID, Stores.Name
AS StoreName, NumberInStock,
			Stores.Line1 + ', ' + Stores.City + ', ' + Stores.State + ', ' +
Stores.Country + ', ' + Stores.ZipCode AS StoreAddress
			FROM Products LEFT JOIN ProductInStore
			ON Products.ProductID = ProductInStore.ProductID
			JOIN Stores
			ON Stores.StoreID = ProductInStore.StoreID)
GO

SELECT * FROM fnCheckProductOnSellingStore(3);
```

fnCheckProductSelling

```
USE ProjectTwo;

DROP FUNCTION fnCheckProductSelling;

GO

CREATE FUNCTION fnCheckProductSelling(@ID INT)
RETURNS INT
	BEGIN
		RETURN(
		SELECT SUM(UnitPrice * Quantity)
			FROM OrderItemsAndReview
			WHERE OrderItemsAndReview.ProductID = @ID
			GROUP BY ProductID)
	END;
```

```
GO

PRINT 'The total selling of product 1 is ' + CAST(dbo. fnCheckProductSelling(1) AS
VARCHAR);
```

fnCheckProductSupplyChain

```
USE ProjectTwo;

DROP FUNCTION fnCheckProductSupplyChain;

GO

CREATE FUNCTION fnCheckProductSupplyChain(@ID INT)
RETURNS TABLE
		RETURN(
			SELECT ProductName,
ProductPerSupplierOverWarehouses.WarehouseID,
ProductPerSupplierOverWarehouses.SupplierID,
			Description, ListPrice, CurrentDiscount
			UnitSupplyPrice, NumberInStock, NumberOnTheWay,
NumberInReturn,
			Suppliers.Name AS SuppliersName, Suppliers.Phone AS
SuppliersPhone, Suppliers.City + ', ' + Suppliers.State + ', ' +  Suppliers.Country AS
SuppliersAddress,
			Warehouses.Phone AS WarehousesPhone, Warehouses.City + ', ' +
Warehouses.State + ', ' + Warehouses.Country AS WarehouseAddress
			FROM
			ProductPerSupplierOverWarehouses LEFT JOIN Products
			ON ProductPerSupplierOverWarehouses.ProductID =
Products.ProductID
			LEFT JOIN Suppliers
			ON ProductPerSupplierOverWarehouses.SupplierID =
Suppliers.SupplierID
			LEFT JOIN Warehouses
			ON ProductPerSupplierOverWarehouses.WarehouseID =
Warehouses.WarehouseID
			WHERE Products.ProductID = @ID)
GO

SELECT * FROM fnCheckProductSupplyChain(3);
```

fnCheckOrderCost

```
USE ProjectTwo;

DROP FUNCTION fnCheckOrderCost;

GO

CREATE FUNCTION fnCheckOrderCost(@ID INT)
RETURNS INT
	BEGIN
		DECLARE @Cost int;
		DECLARE @Tax int;
		SELECT @Cost = SUM(UnitPrice * Quantity)
			FROM OrderItemsAndReview
			WHERE OrderItemsAndReview.OrderID = @ID
			GROUP BY OrderID;
		SELECT @Tax = TaxAmount
			FROM Orders
			WHERE Orders.OrderID = @ID;
		RETURN	@Cost - @Tax
	END;

GO

PRINT 'The total cost of order 1 is ' + CAST(dbo.fnCheckOrderCost(1) AS VARCHAR);
```

6) Triggers

CustomerNameUpdate

```
USE ProjectTwo;

DROP TRIGGER CustomerNameUpdate;

GO

CREATE TRIGGER CustomerNameUpdate
	ON Customers
	AFTER INSERT, UPDATE
AS
	UPDATE Customers
	SET LastName = UPPER(LEFT(LastName, 1)) + ' (' + LastName + ')'
	WHERE CustomerID IN (SELECT CustomerID FROM inserted);
```

CustomerUpdateLog

```
USE ProjectTwo;

DROP TRIGGER CustomerUpdateLog;

GO

CREATE TRIGGER CustomerUpdateLog
	ON Customers
	AFTER INSERT, UPDATE
AS
	SELECT Customers.FirstName + ', ' + Customers.LastName AS 'Updated Customer
Name'
	FROM Customers
	WHERE CustomerID IN (SELECT CustomerID FROM inserted);
```

OnlineOrder

```
USE ProjectTwo;

DROP TRIGGER OnlineOrder;

GO

CREATE TRIGGER OnlineOrder
	ON OnlineOrders
	AFTER INSERT
AS
	DECLARE @name VARCHAR;
	DECLARE @date DATETIME;
	SELECT @name = (Customers.FirstName + ', ' + Customers.LastName)
	FROM Customers
	WHERE CustomerID IN (SELECT CustomerID FROM inserted);
	SELECT @date = OrderDate
	FROM Orders
	WHERE CustomerID IN (SELECT CustomerID FROM inserted);
	PRINT 'Customer ' + @name + ' make an online order at ' + CAST(@date AS
VARCHAR);
```

TooManyInStock

```
CREATE TRIGGER TooManyInStock
	ON ProductInStore
	AFTER UPDATE, INSERT
AS
BEGIN
	IF EXISTS
	(SELECT NumberInStock FROM ProductInStore
	WHERE StoreID IN (SELECT StoreID FROM inserted)
	AND NumberInStock > 200)
	THROW 50002, 'Quantity of a product in a store is more than 200!!!!!', 1;
END;
```

7) Transaction

MergeStore

```
USE ProjectTwo;

GO

BEGIN TRAN;
	UPDATE InStoreOrders
	SET StoreID = 2
	WHERE StoreID = 4;

	UPDATE ProductInStore
	SET StoreID = 2
	WHERE StoreID = 4;

	DELETE Stores
	WHERE StoreID = 4;
COMMIT TRAN
```

MergeWarehoues

```
USE ProjectTwo;

GO

BEGIN TRAN;
	UPDATE ProductPerSupplierOverWarehouses
	SET WarehouseID = 2
	WHERE WarehouseID = 4;

	DELETE Warehouses
```

```
        WHERE WarehouseID = 4;
COMMIT TRAN
```

RemoveOrder

```
USE ProjectTwo;

GO

BEGIN TRAN;
        DELETE OnlineOrders
        WHERE OrderID = 4;

        DELETE InStoreOrders
        WHERE OrderID = 4;

        DELETE OrderItemsAndReview
        WHERE OrderID = 4;

        DELETE Orders
        WHERE OrderID = 4;
COMMIT TRAN
```

RemoveProduct

```
USE ProjectTwo;

GO

BEGIN TRAN;
        DELETE ProductPerSupplierOverWarehouses
        WHERE ProductID = 4;

        DELETE WishListItem
        WHERE ProductID = 4;

        DELETE ProductInStore
        WHERE ProductID = 4;

        DELETE OrderItemsAndReview
        WHERE ProductID = 4;

        DELETE Products
```

```
        WHERE ProductID = 4;
COMMIT TRAN
```

8) User Creation

Prerequisite - Roles Creation

```
USE ProjectTwo;

DROP ROLE CustomerManager;

CREATE ROLE CustomerManager;

GRANT UPDATE, INSERT
ON Customers TO CustomerManager;

GRANT UPDATE, INSERT
ON Addresses TO CustomerManager;

GRANT UPDATE, INSERT
ON Phones TO CustomerManager;

ALTER ROLE db_datareader ADD MEMBER CustomerManager
```

```
DROP ROLE OrderManager;

CREATE ROLE OrderManager;

GRANT UPDATE, INSERT
ON Orders TO OrderManager;

GRANT UPDATE, INSERT
ON OnlineOrders TO OrderManager;

GRANT UPDATE, INSERT
ON InStoreOrders TO OrderManager;

GRANT UPDATE, INSERT
ON OrderItemsAndReview TO OrderManager;

ALTER ROLE db_datareader ADD MEMBER OrderManager
```

```
DROP ROLE ProductAndStoreManager;

CREATE ROLE ProductAndStoreManager;

GRANT UPDATE, INSERT
ON Products TO ProductAndStoreManager;

GRANT UPDATE, INSERT
ON ProductInStore TO ProductAndStoreManager;

GRANT UPDATE, INSERT
ON Stores TO ProductAndStoreManager;

ALTER ROLE db_datareader ADD MEMBER ProductAndStoreManager
```

```
DROP ROLE SupplyChainManager;

CREATE ROLE SupplyChainManager;

GRANT UPDATE, INSERT
ON Products TO SupplyChainManager;

GRANT UPDATE, INSERT
ON ProductPerSupplierOverWarehouses TO SupplyChainManager;

GRANT UPDATE, INSERT
ON Warehouses TO SupplyChainManager;

GRANT UPDATE, INSERT
ON Suppliers TO SupplyChainManager;

ALTER ROLE db_datareader ADD MEMBER SupplyChainManager
```

Yuchen - Domain

```
CREATE LOGIN Domain WITH PASSWORD = '12345678',
DEFAULT_DATABASE = ProjectTwo;

CREATE USER Yuchen FOR LOGIN [Domain];
ALTER ROLE CustomerManager ADD MEMBER Yuchen
ALTER ROLE SupplyChainManager ADD MEMBER Yuchen
ALTER ROLE OrderManager ADD MEMBER Yuchen
ALTER ROLE ProductAndStoreManager ADD MEMBER Yuchen
```

Mark - Manager

```
CREATE LOGIN Manager WITH PASSWORD = '892712',
DEFAULT_DATABASE = ProjectTwo;

CREATE USER Mark FOR LOGIN [Manager];
ALTER ROLE CustomerManager ADD MEMBER Mark
ALTER ROLE OrderManager ADD MEMBER Mark
```

Jake - Intern

```
CREATE LOGIN Intern WITH PASSWORD = '122712',
DEFAULT_DATABASE = ProjectTwo;

CREATE USER Jake FOR LOGIN [Intern];
ALTER ROLE ProductAndStoreManager ADD MEMBER Jake
```

David - SeniorManager

```
CREATE LOGIN SeniorManager WITH PASSWORD = '8122712',
DEFAULT_DATABASE = ProjectTwo;

CREATE USER David FOR LOGIN [SeniorManager];
ALTER ROLE CustomerManager ADD MEMBER David
ALTER ROLE OrderManager ADD MEMBER David
ALTER ROLE ProductAndStoreManager ADD MEMBER David
```

D. Testing

1) Scenario One

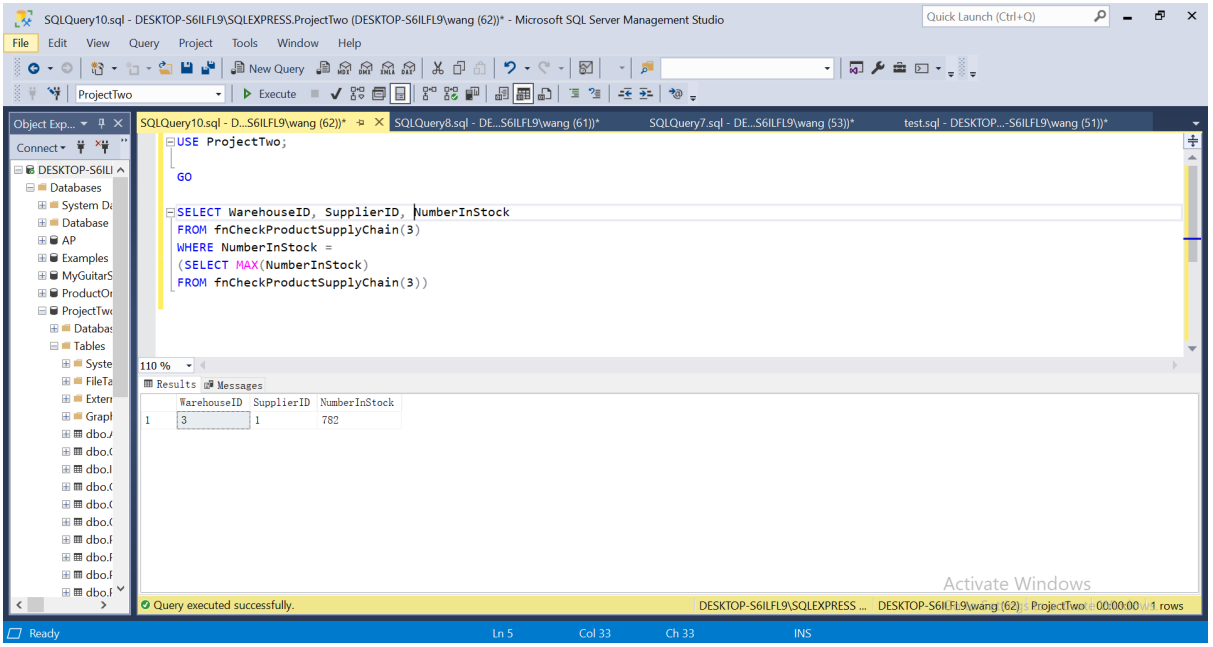
Description: I want to know which supplier in which warehouse has the most of product in stock for product ‘Gibson SG’

```
USE ProjectTwo;

GO

SELECT WarehouseID, SupplierID, NumberInStock
FROM fnCheckProductSupplyChain(3)
WHERE NumberInStock =
(SELECT MAX(NumberInStock)
```

FROM fnCheckProductSupplyChain(3))



2) Scenario Two

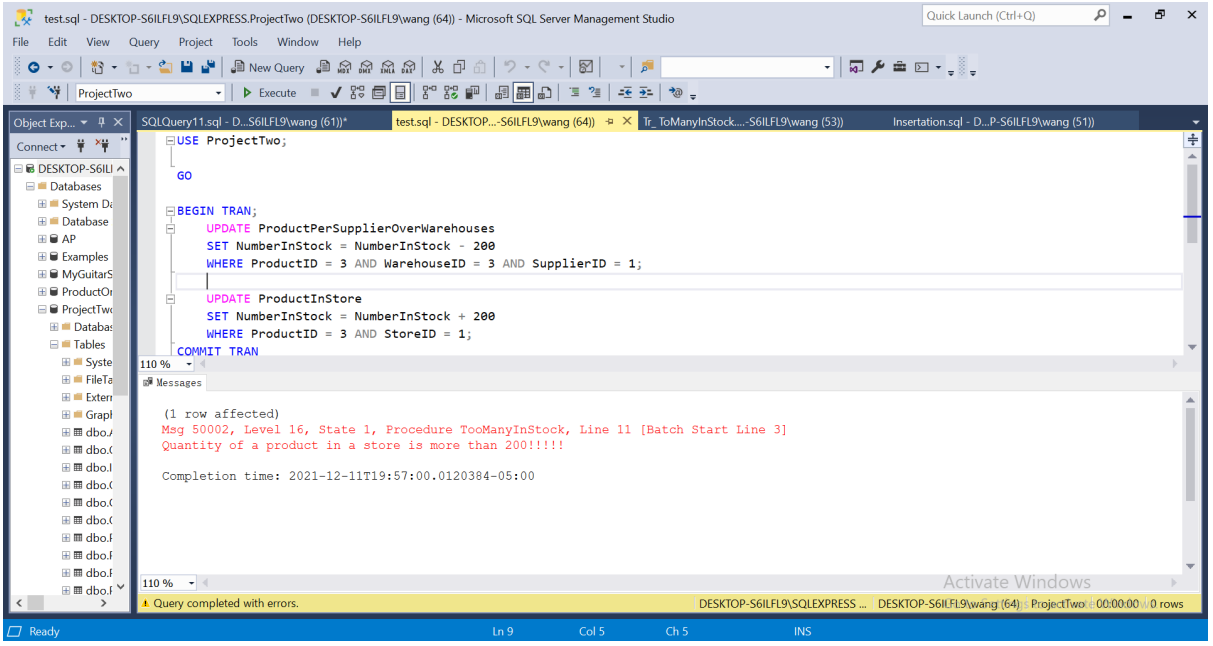
Description: Get 200 'Gibson SG' from warehouse 3 into store 1, which triggered the trigger 'TooManyInStock', as the number of 'Gibson SG' in store will become over 200.

```
USE ProjectTwo;

GO

BEGIN TRAN;
        UPDATE ProductPerSupplierOverWarehouses
        SET NumberInStock = NumberInStock - 200
        WHERE ProductID = 3 AND WarehouseID = 3 AND SupplierID = 1;

        UPDATE ProductInStore
        SET NumberInStock = NumberInStock + 200
        WHERE ProductID = 3 AND StoreID = 1;
COMMIT TRAN
```



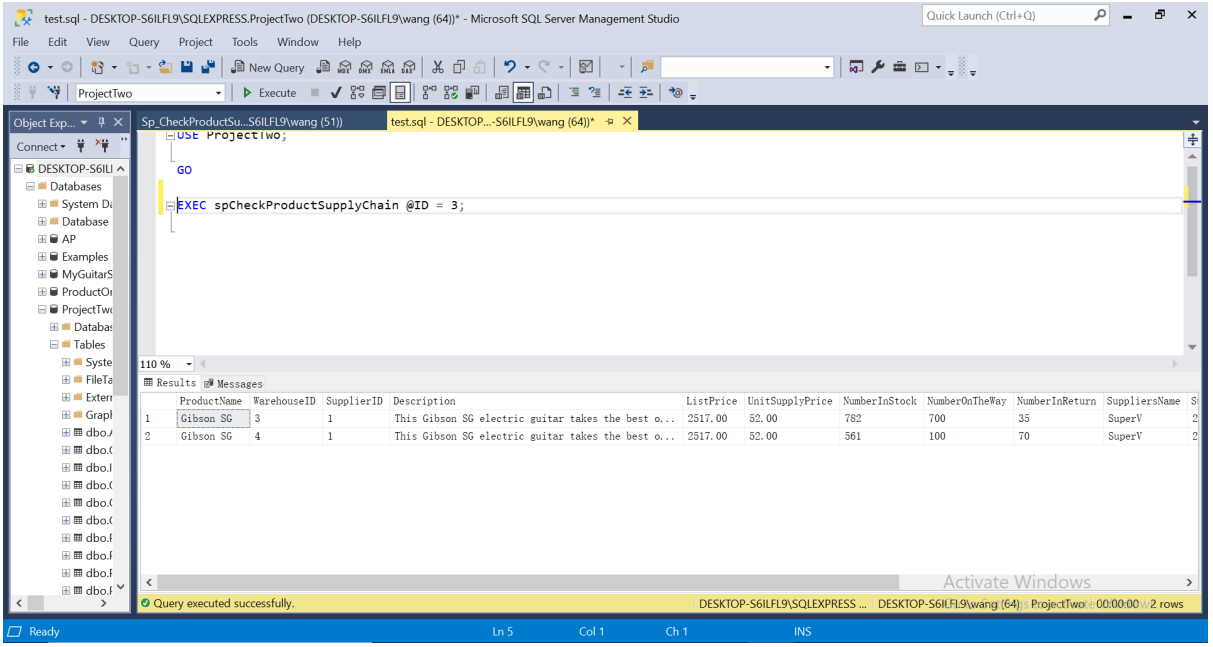
3) Scenario Three

Description: I want to know the supply chain of product ‘Gibson SG’

USE ProjectTwo;

GO

EXEC spCheckProductSupplyChain @ID = 3;



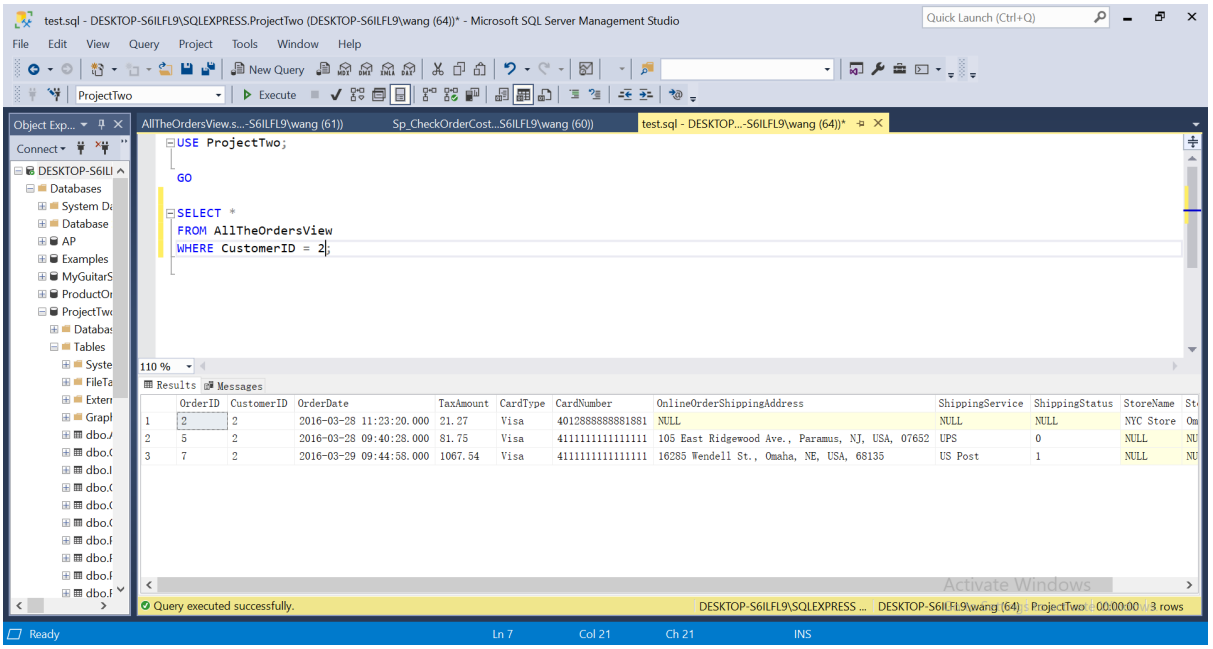
4) Scenario Four

Description: I want to know what are the orders made by the customer whose CustomerID is 2.

```
USE ProjectTwo;

GO

SELECT *
FROM AllTheOrdersView
WHERE CustomerID = 2;
```

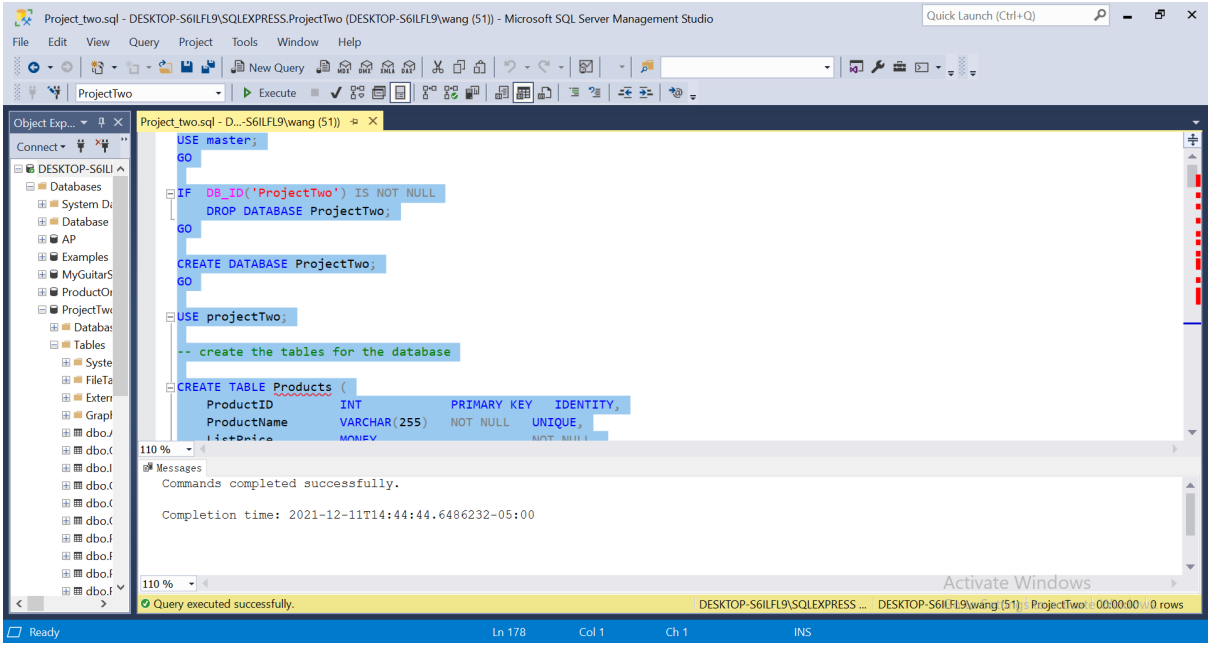


E. Conclusion

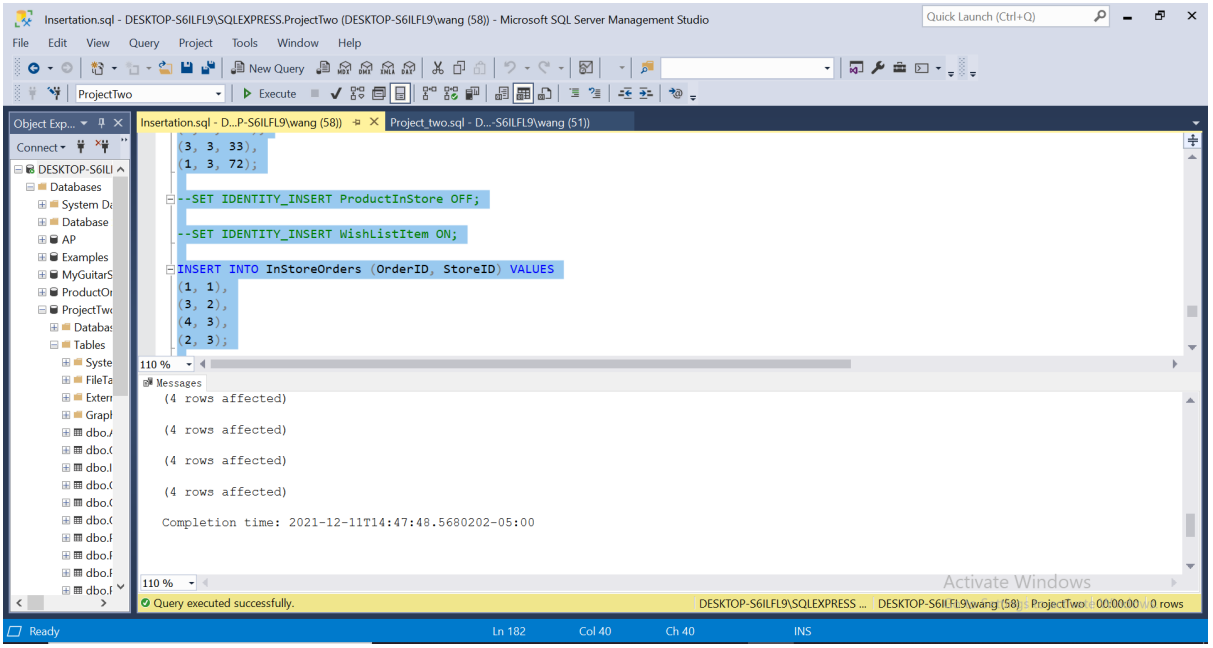
Throughout this project, I learned so much especially on creating a database, using SQL query to manipulate data in databases, and managing database security. I firstly designed a new database named ProjectTwo and normalized it into a third normal form, and set up the database with all the tables and the relative values. Then, I insert mock data into tables of the database by using INSERT. After that, I created four views in order to understand more aspects of the database by using the views to select data which related to each other in some way. I also created functions, procedures to help me select data with specific requirements., and triggers allowing me to set customized data insertion rules and automatically monitor invalid changes which behave against the rules. Besides, I attempted to create new roles and users for my database, allowing more people to manage the database with various levels of authorization. In the end, I customized some test cases to test my database design and security level. Overall, it is definitely a meaningful project and I learned a lot from it.

F. Appendix

1) Database Creation

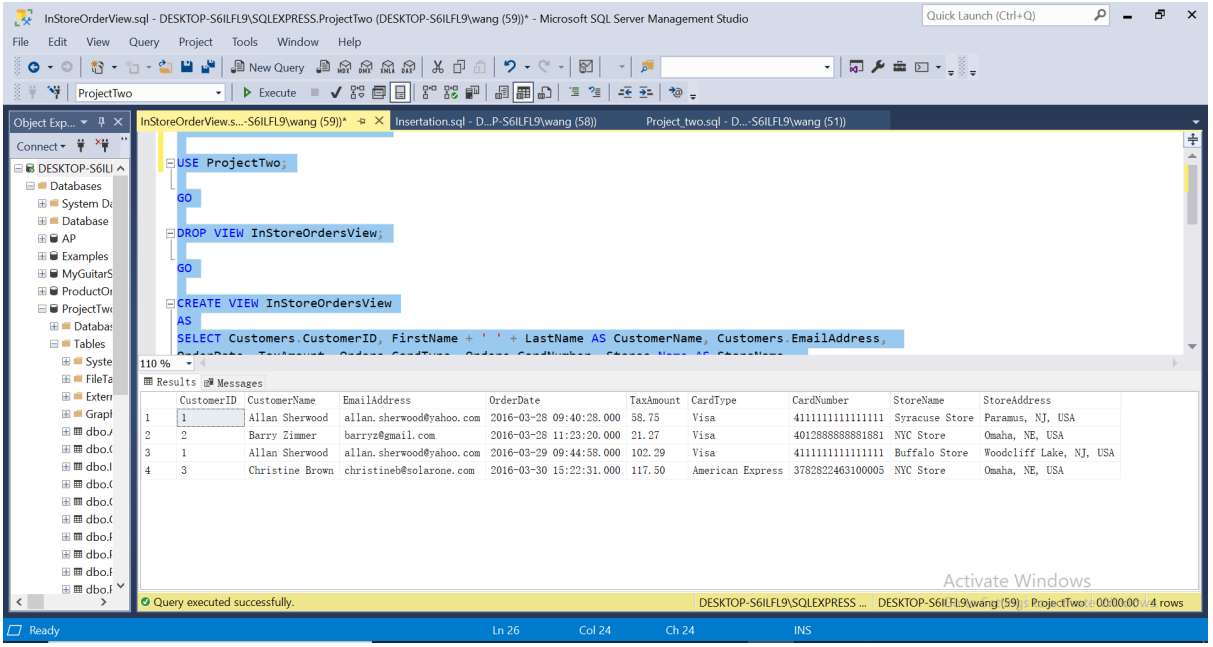


2) Data Insertation

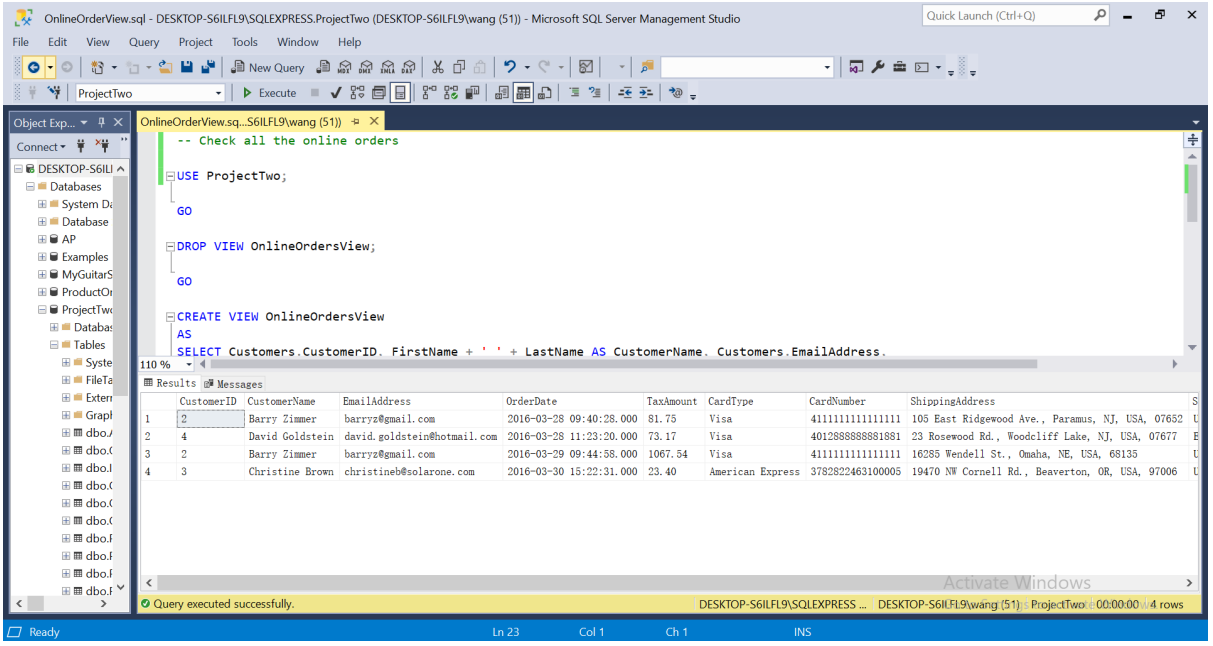


3) Views

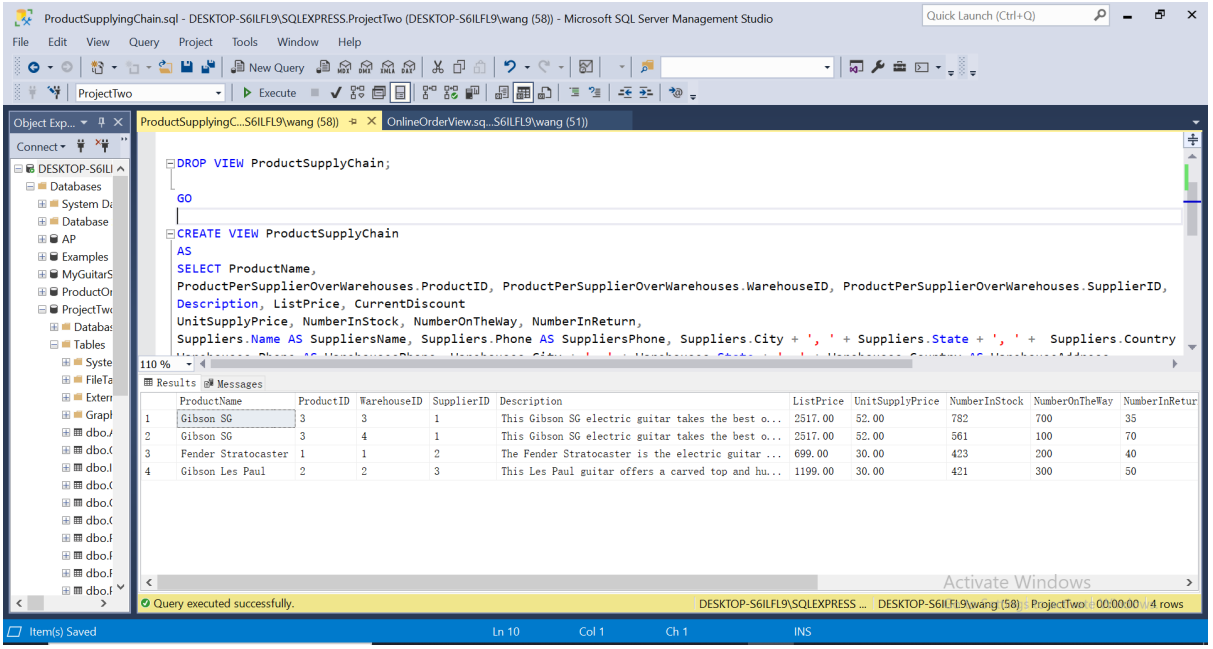
InStoreOrdersView



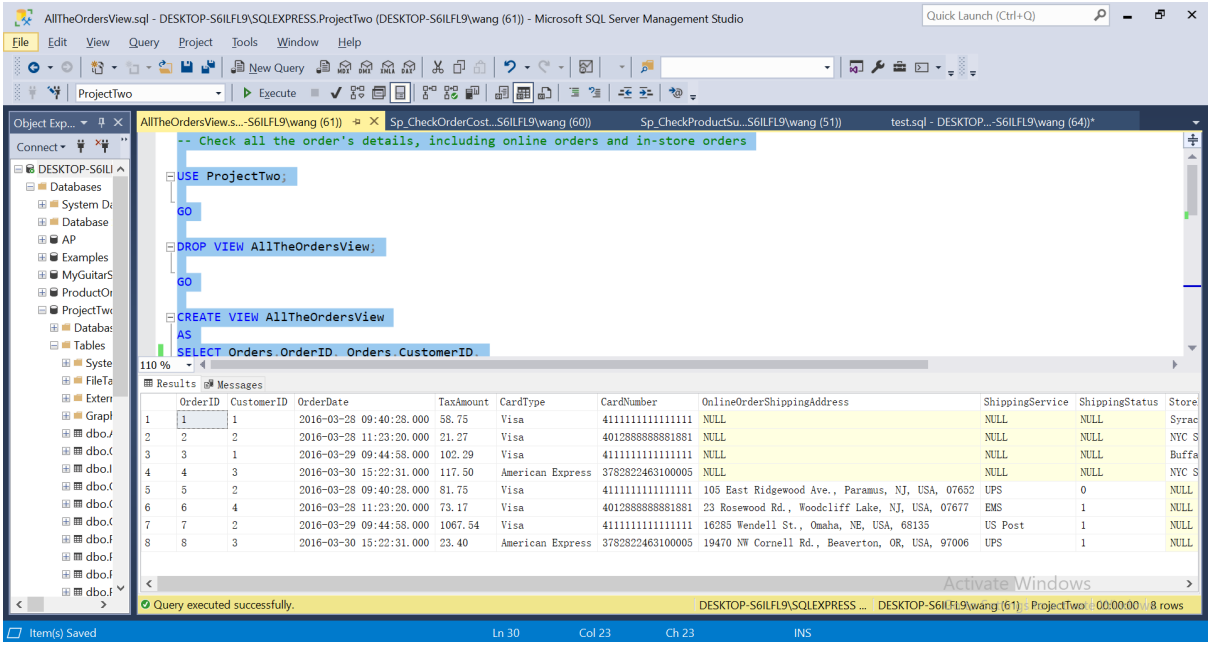
OnlineOrdersView



ProductSupplyChainView

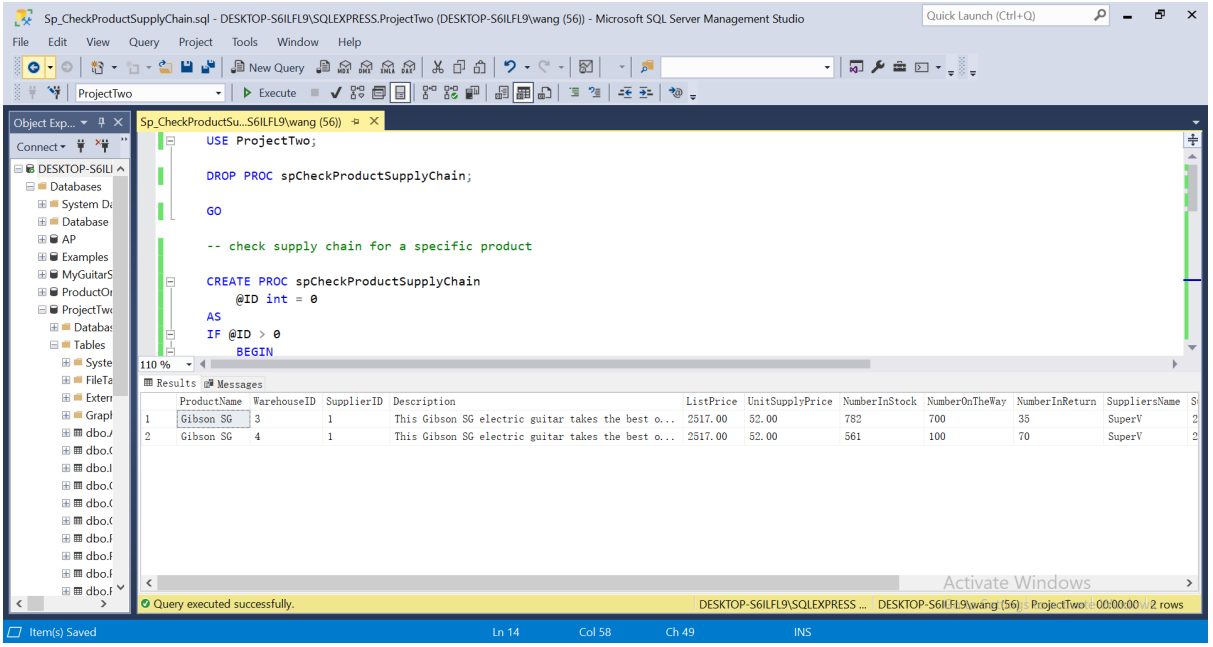


AllTheOrdersView

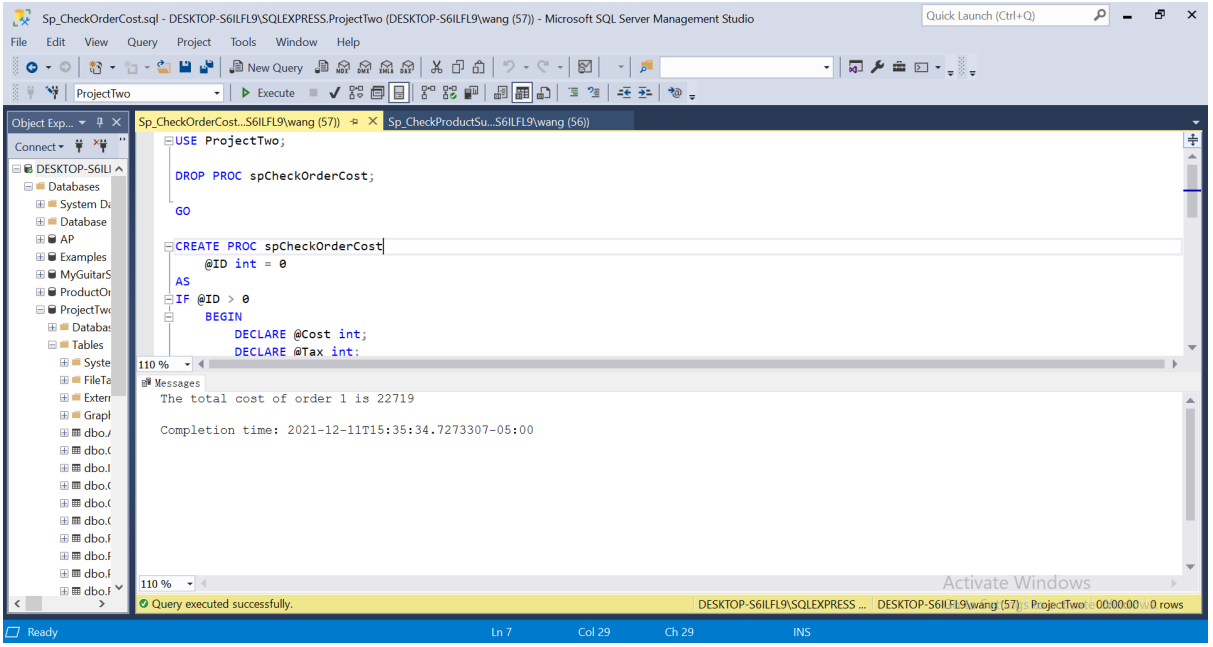


4) Store Procedures

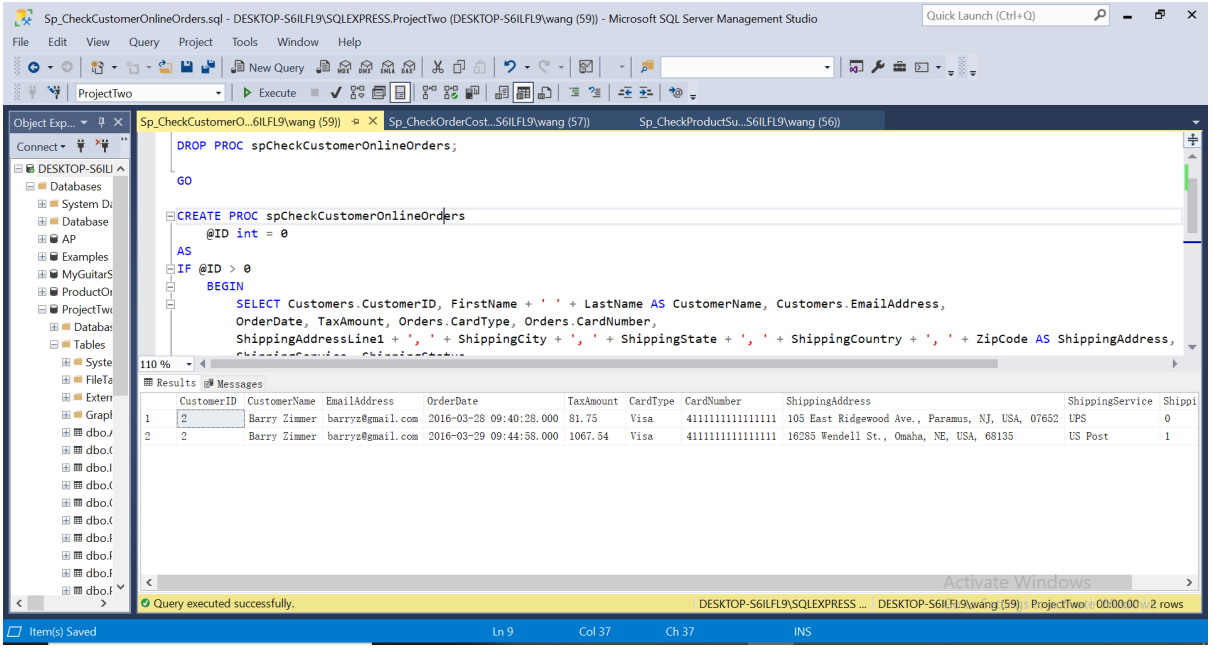
spCheckProductSupplyChain



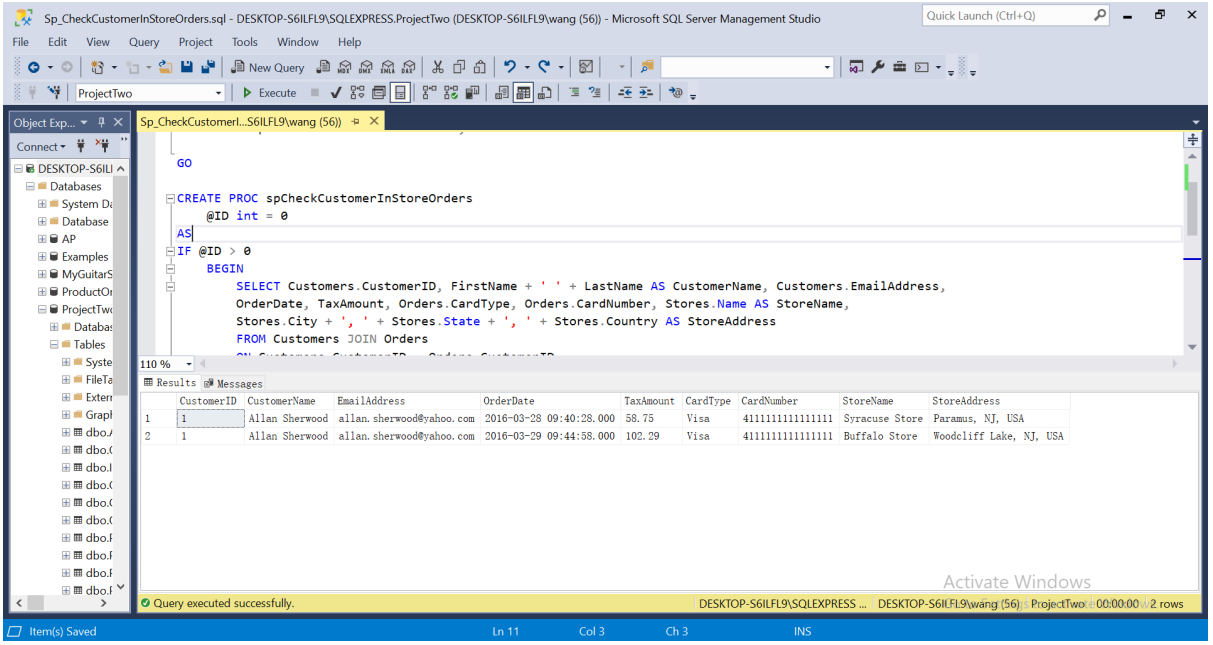
spCheckOrderCost



spCheckCustomerOnlineOrders

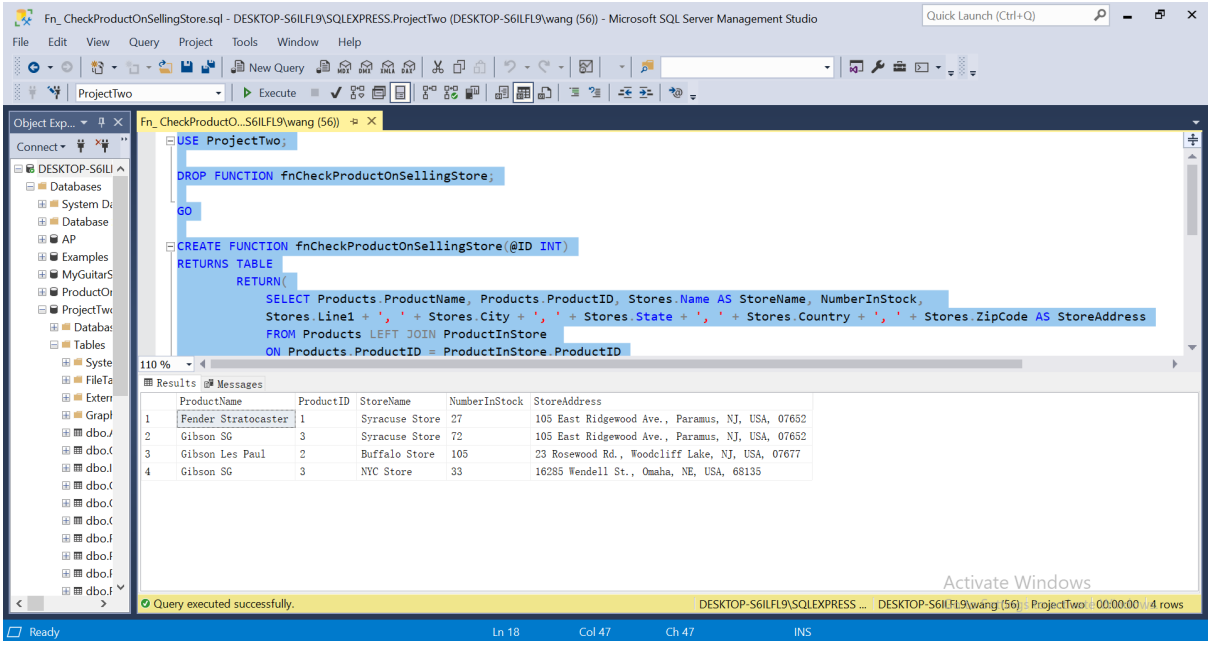


spCheckCustomerInStoreOrders

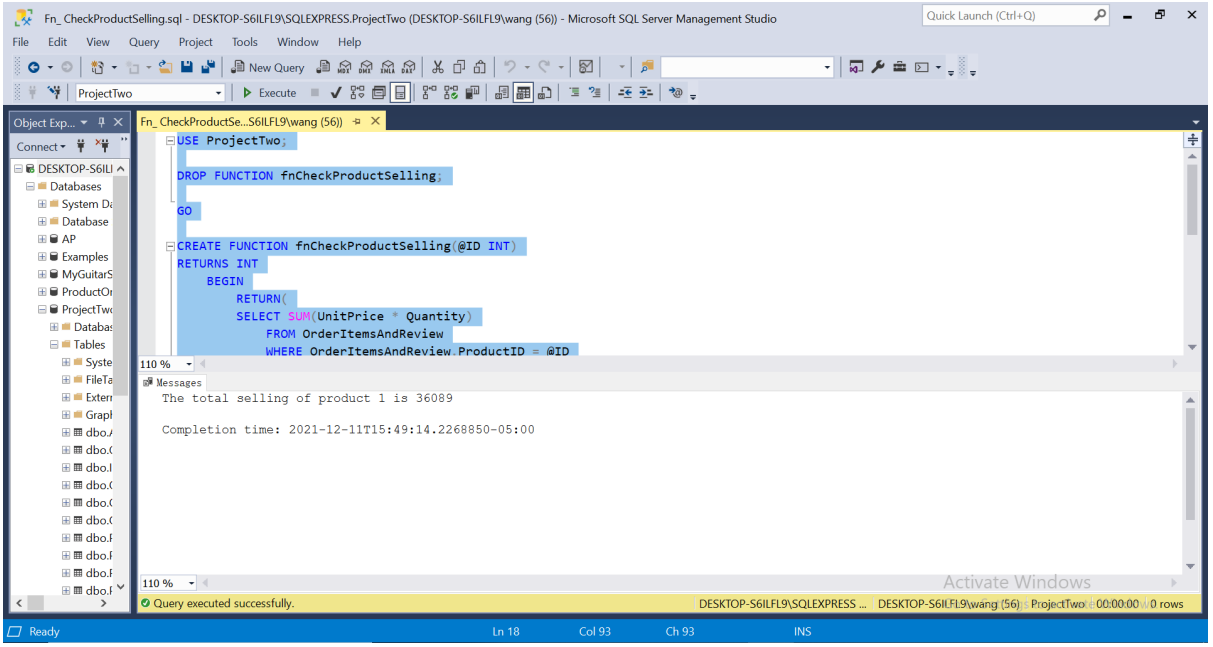


5) User Defined Function

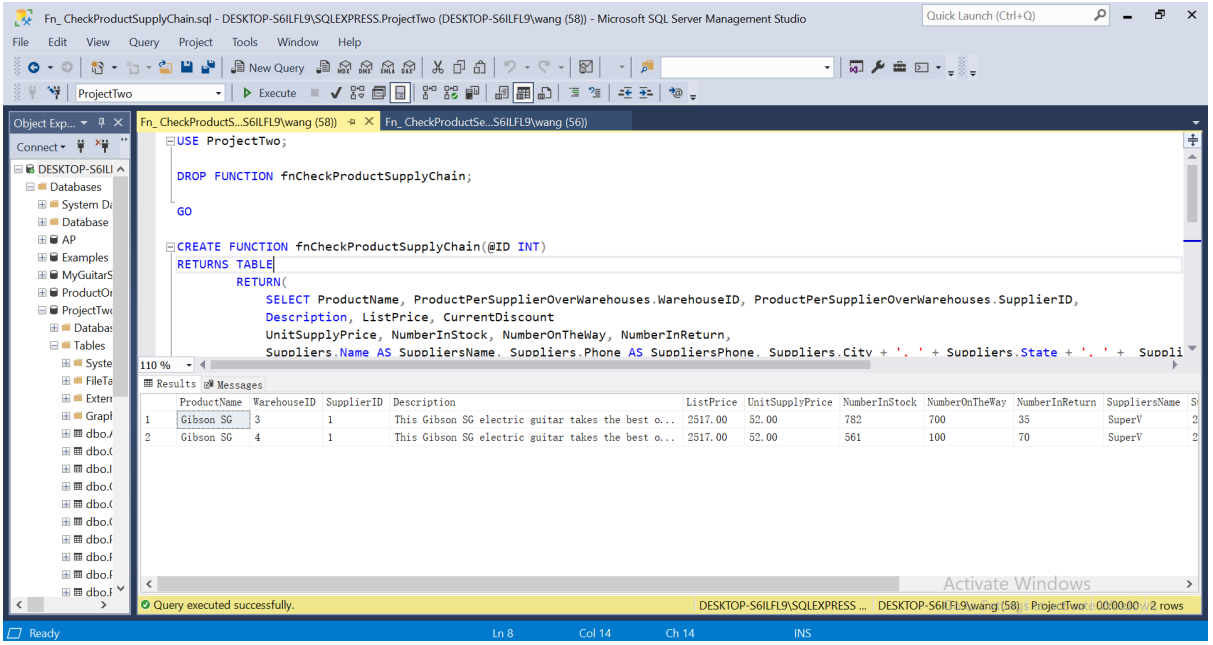
fnCheckProductOnSellingStore



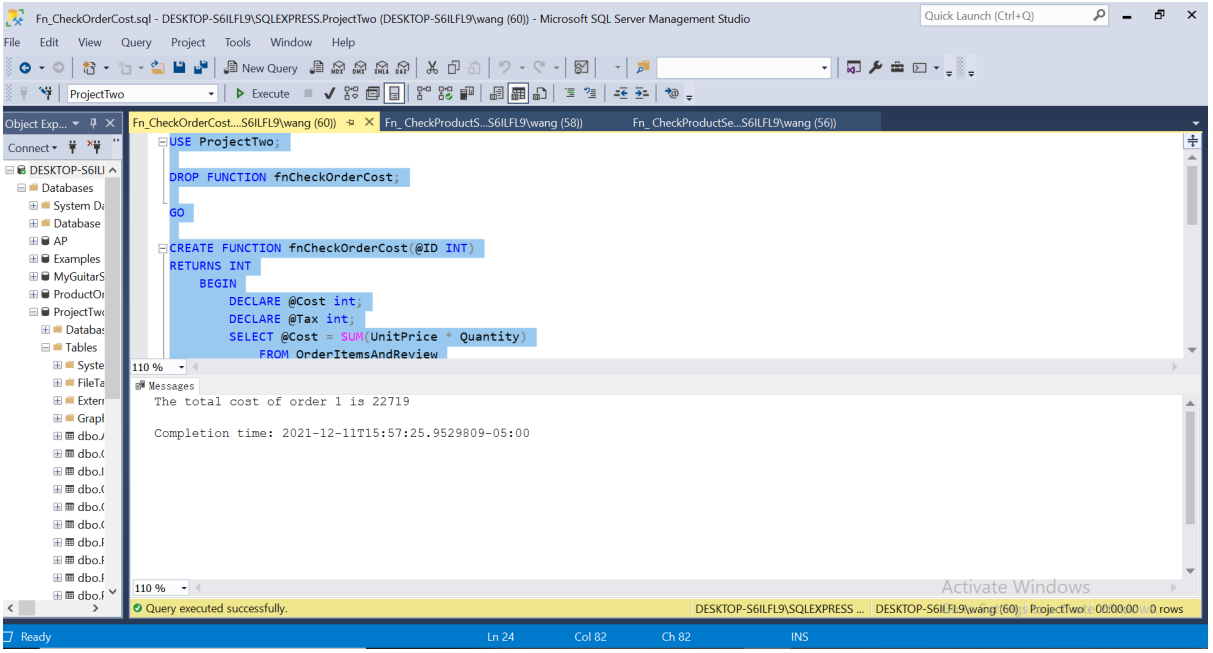
fnCheckProductSelling



fnCheckProductSupplyChain

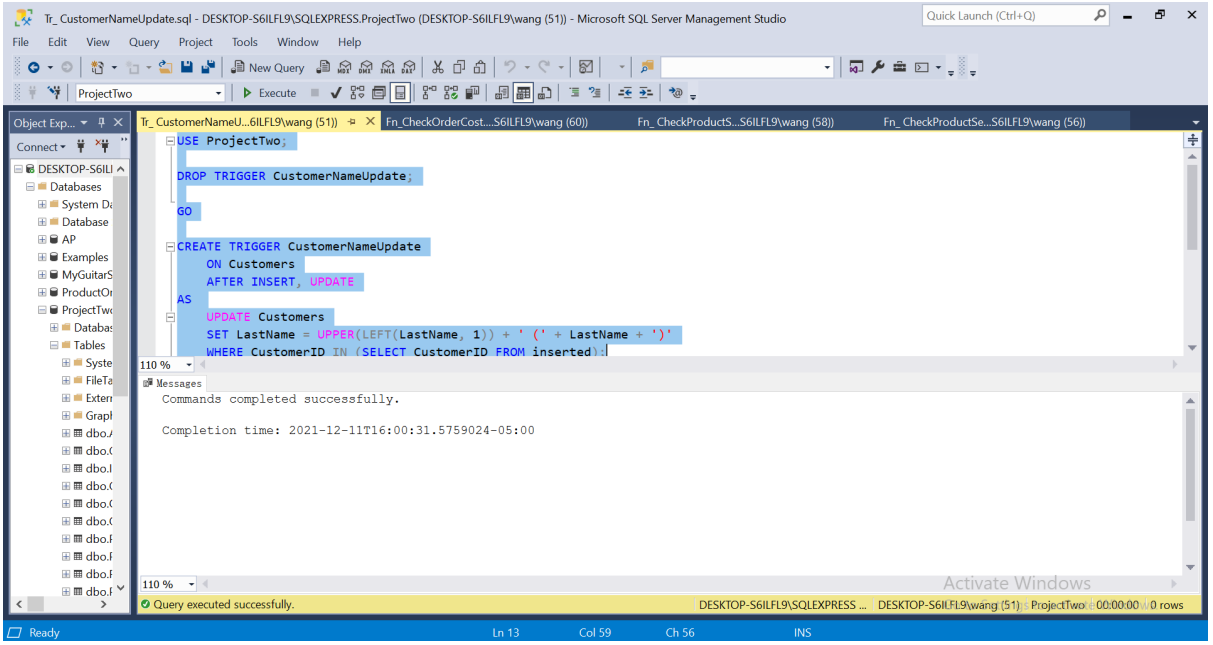


fnCheckOrderCost

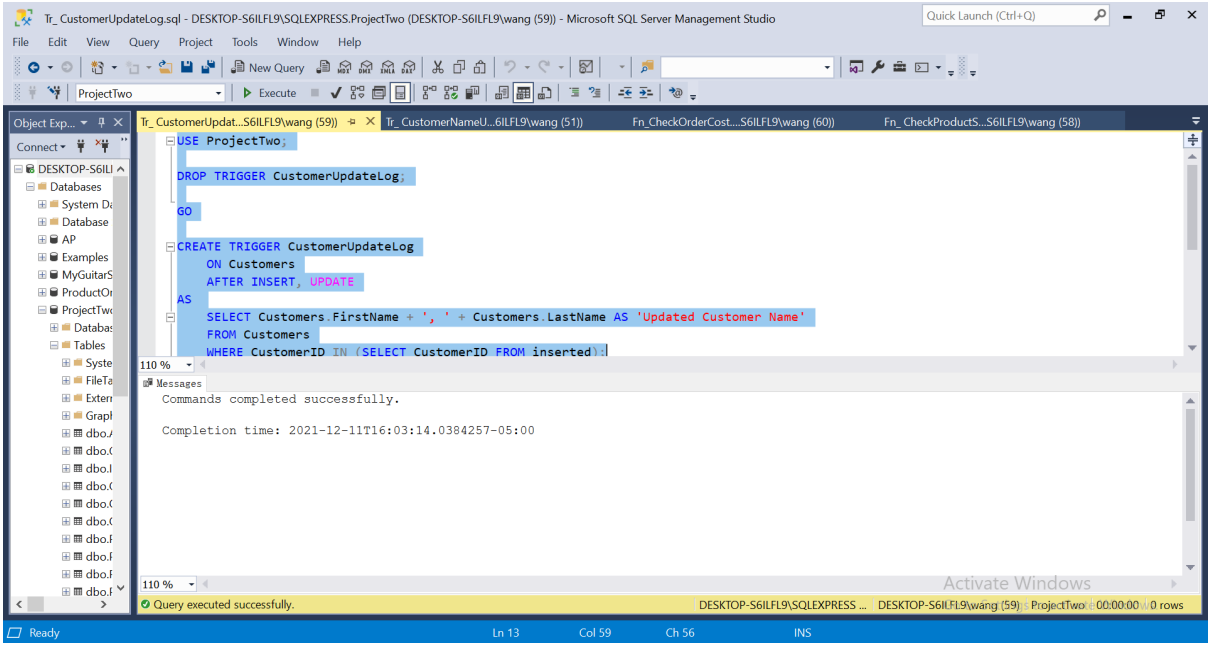


6) Triggers

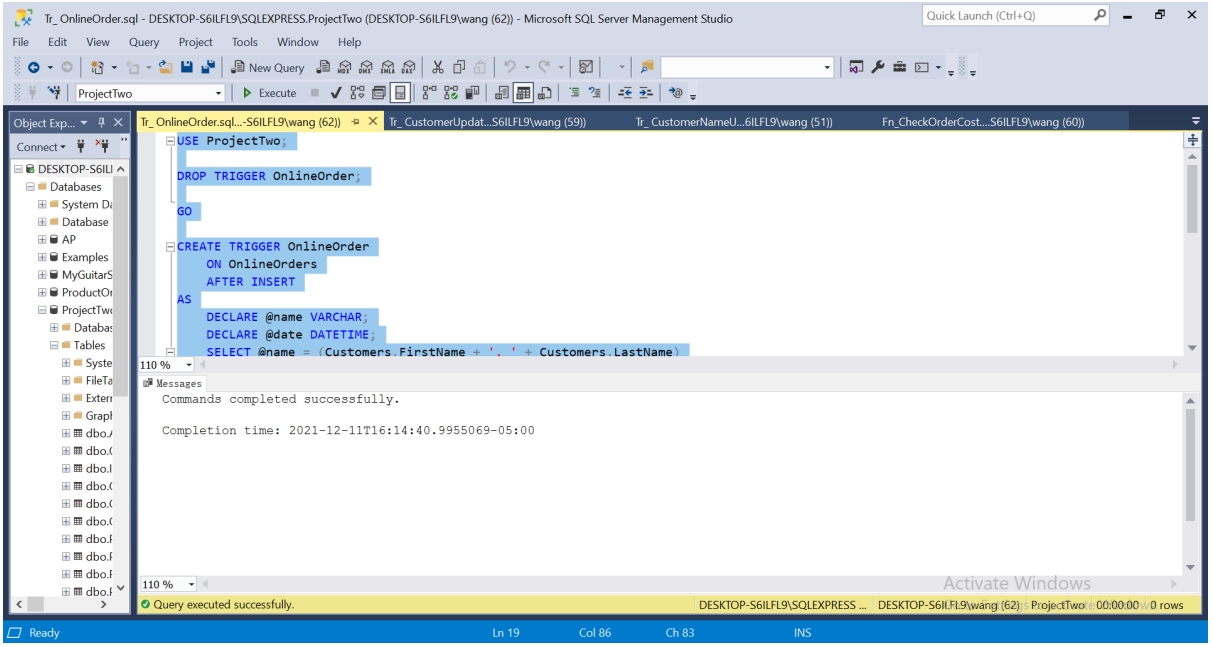
CustomerNameUpdate



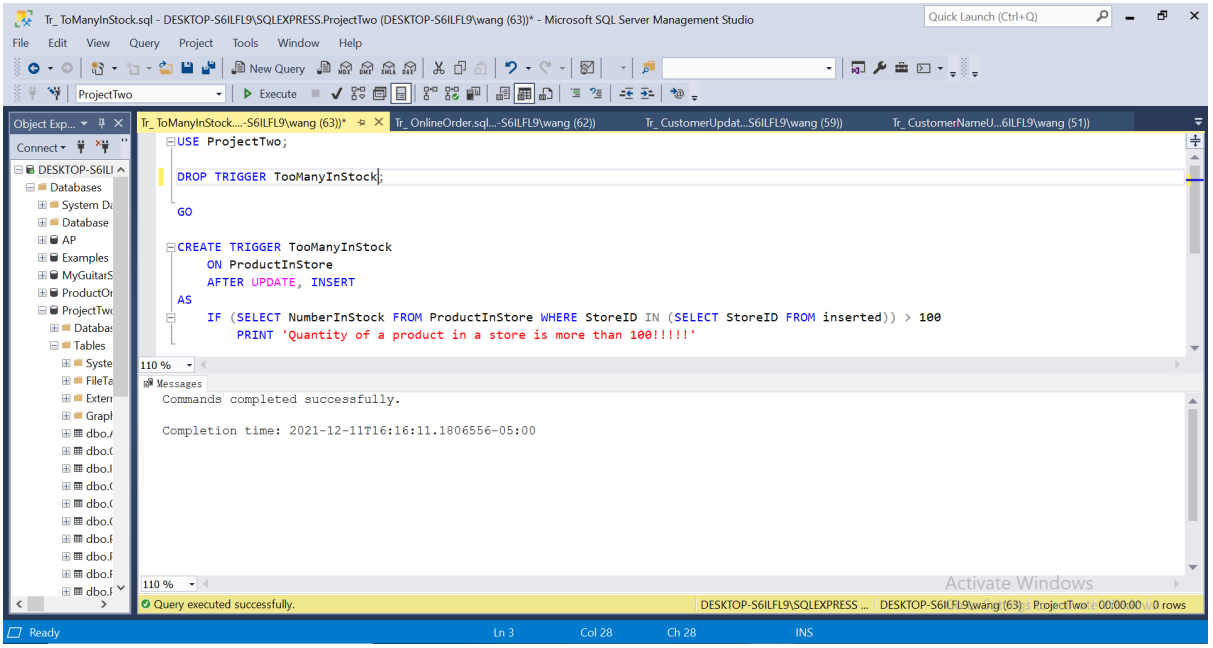
CustomerUpdateLog



OnlineOrder

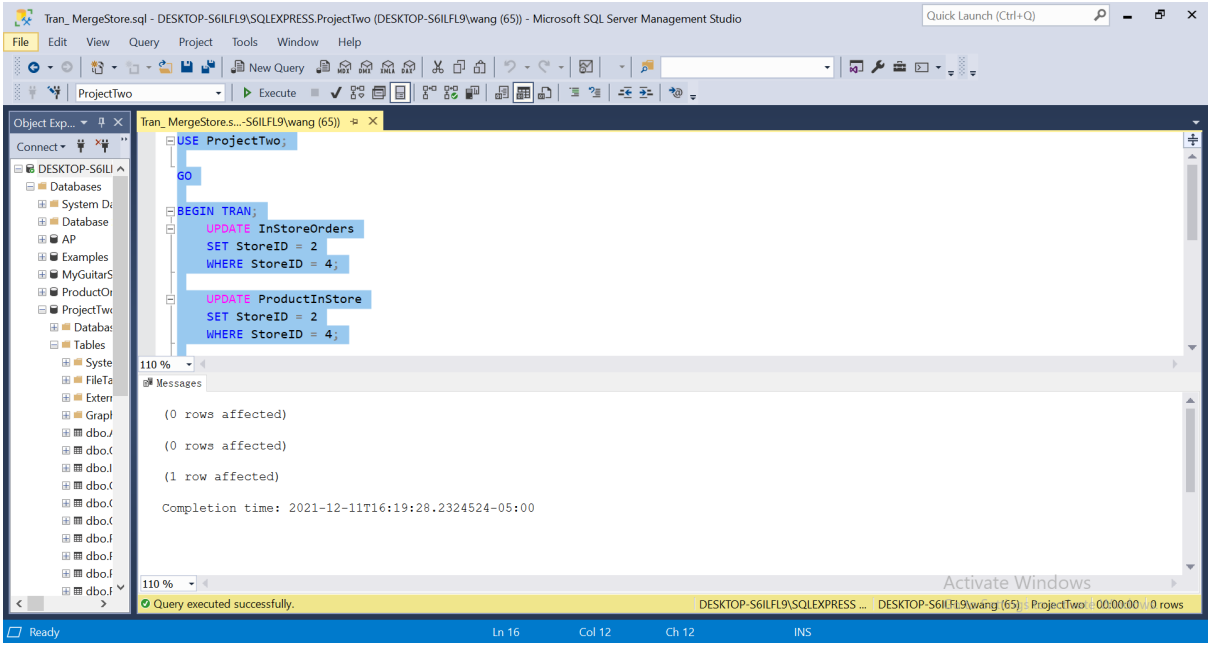


TooManyInStock

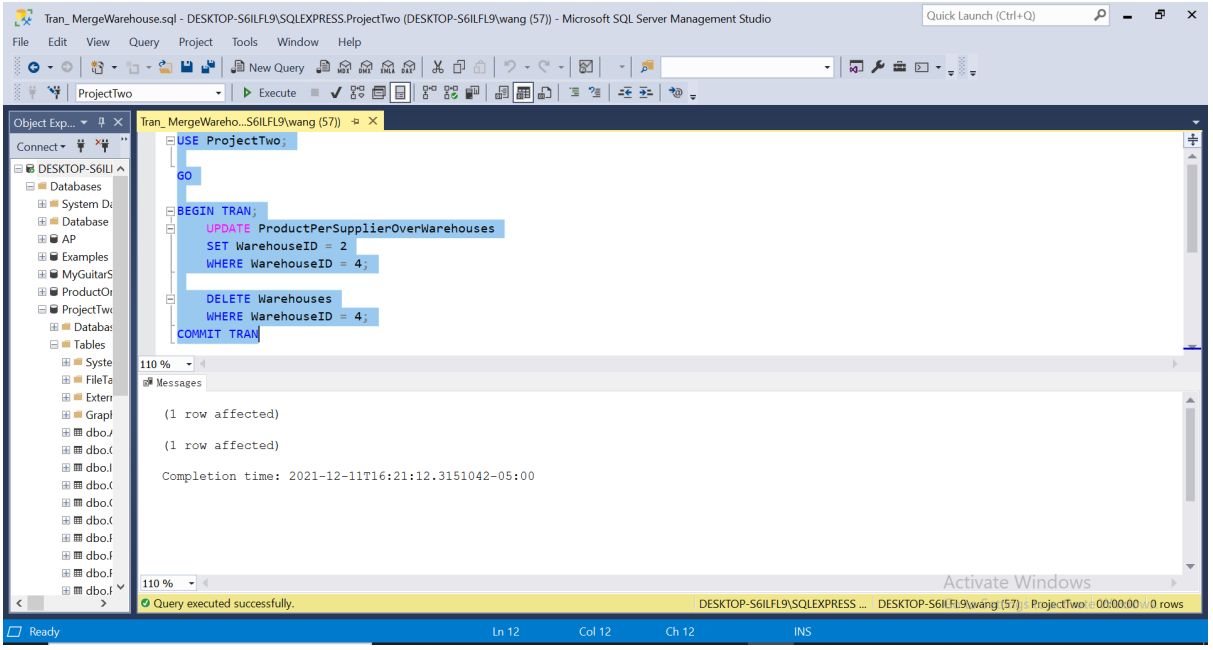


7) Transaction

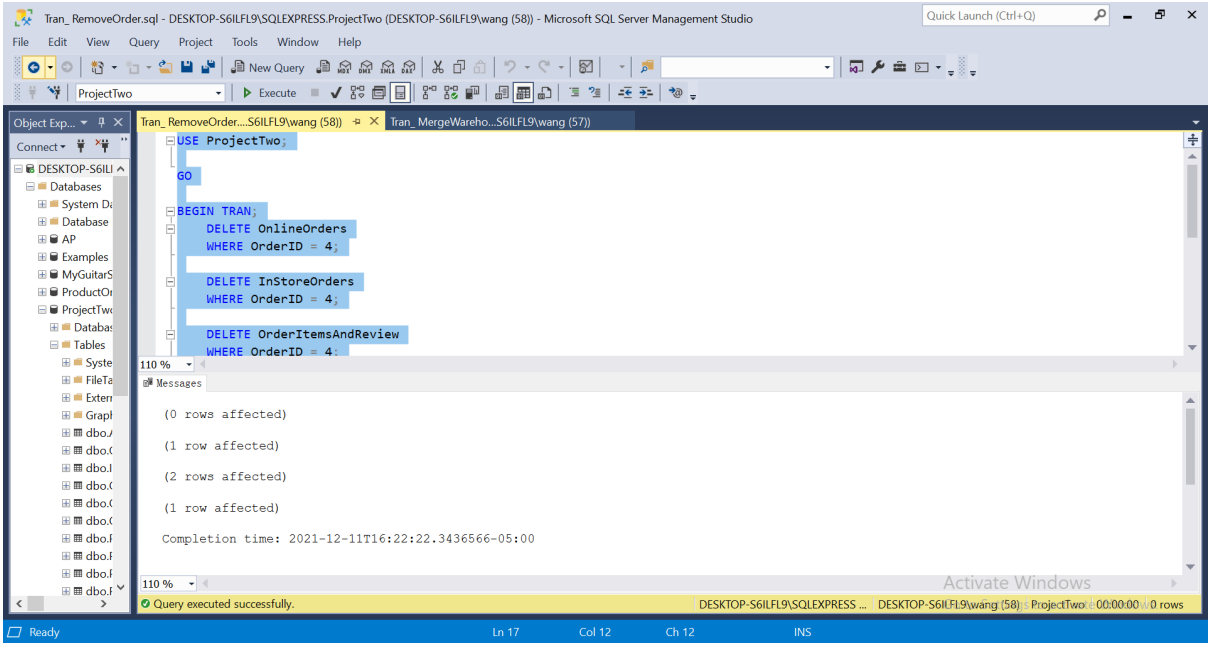
MergeStore



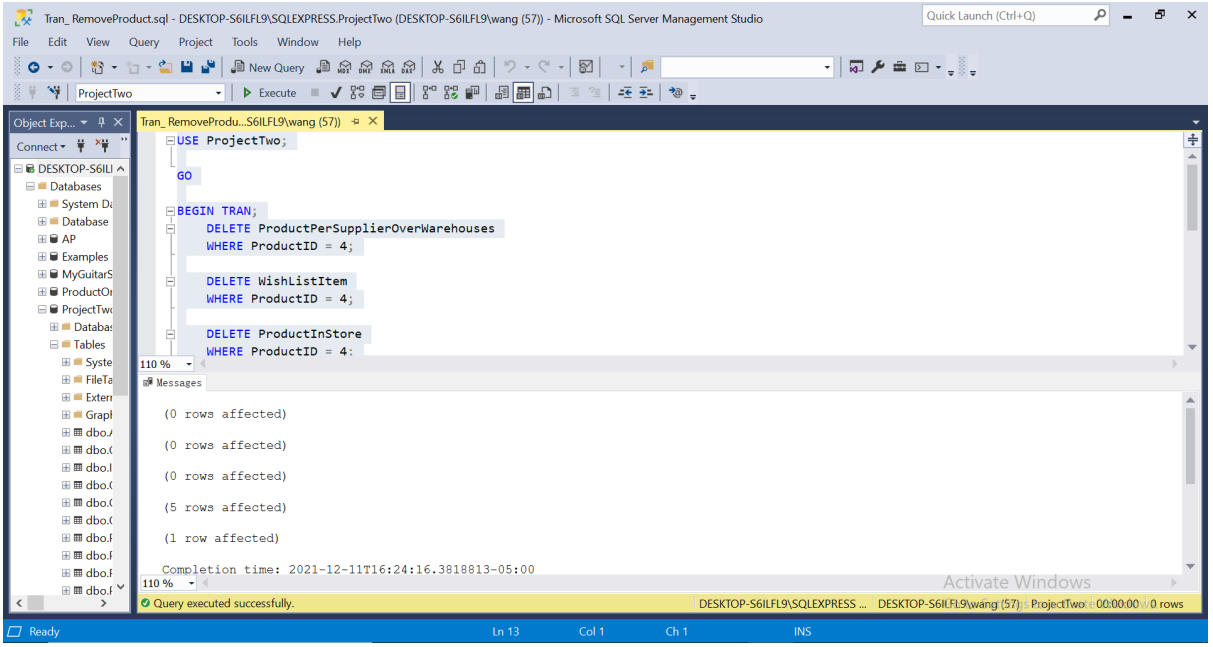
MergeWarehoues



RemoveOrder

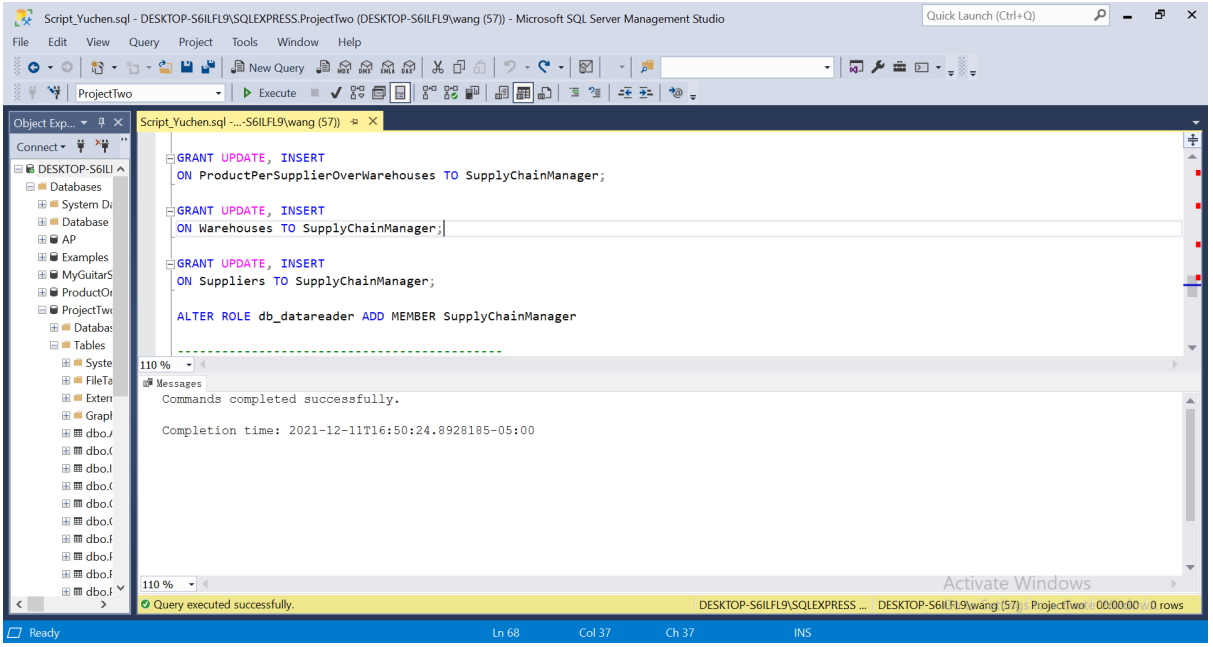


RemoveProduct

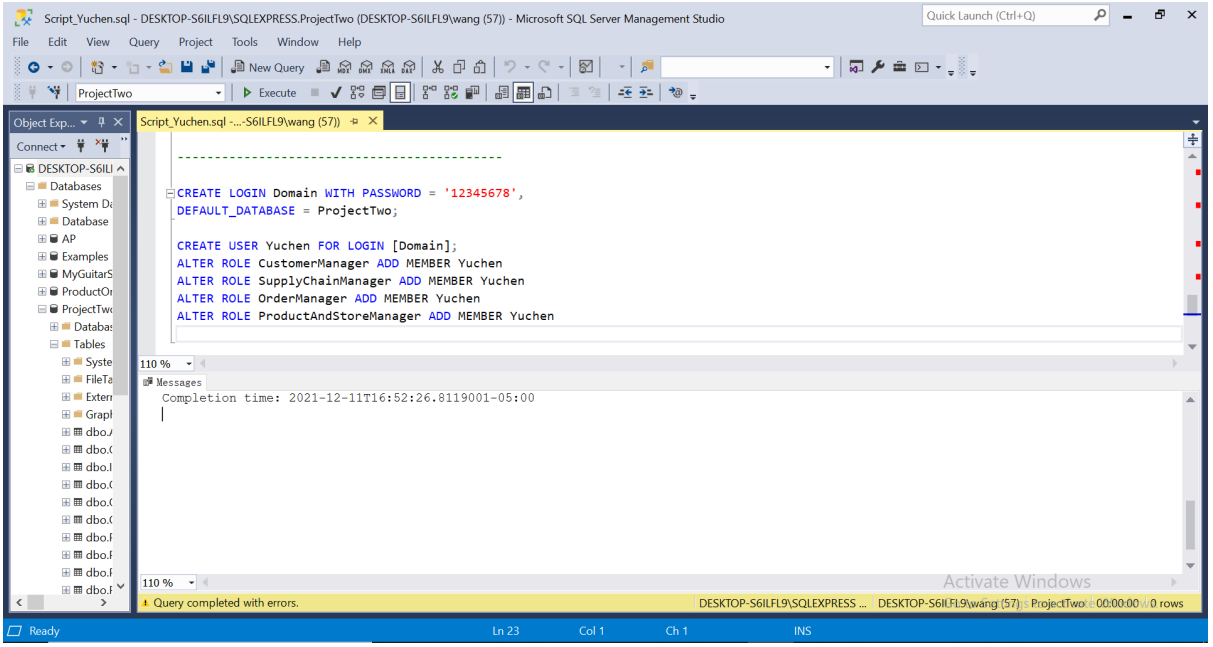


8) User Creation

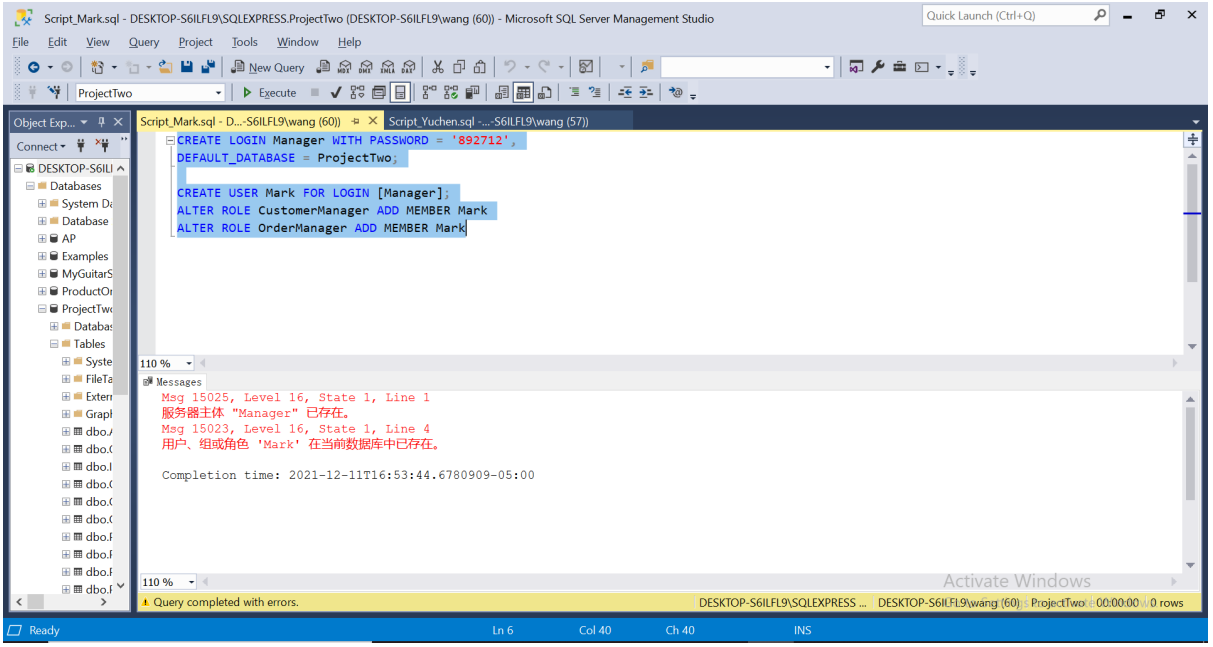
Prerequisite - Roles Creation



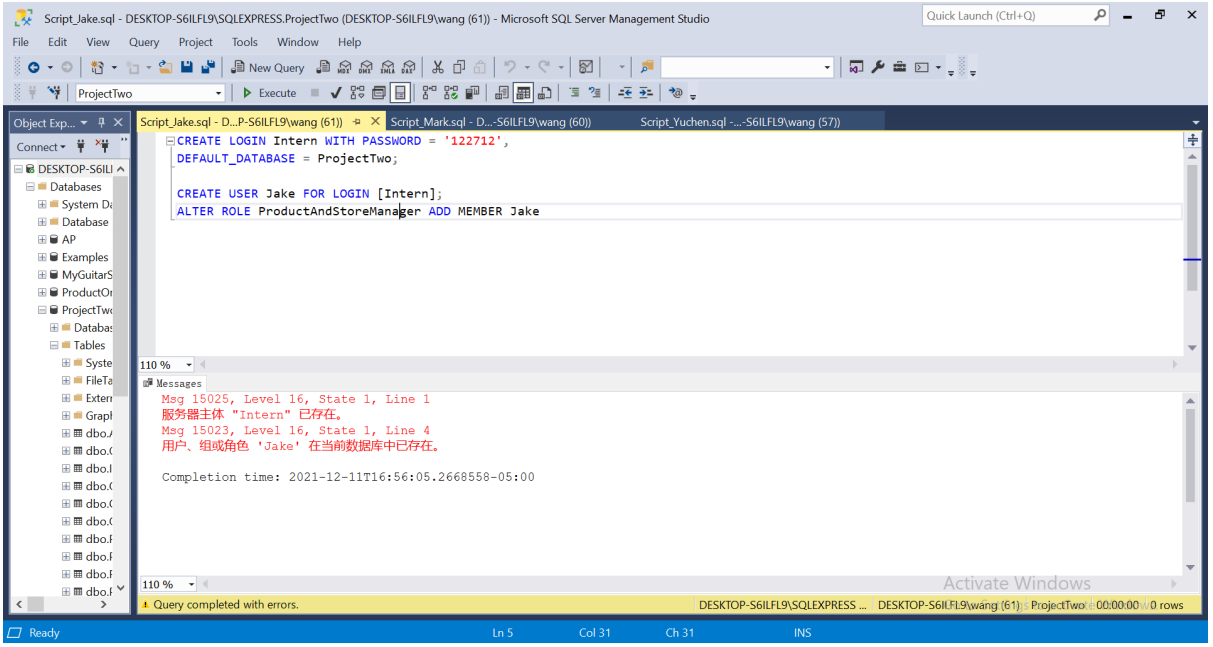
Yuchen - Domain



Mark - Manager



Jake - Intern



David - SeniorManager