

A TARGET COMPETITOR



Prof. Ercanli

Ercanli

Seyed Alireza Zarrin Mehr

Syracuse University
Introduction to Database Management Systems

Abstract:

in this project, we designed a database for a chain store that offers online and in-store sales. This design is based on the following business model:

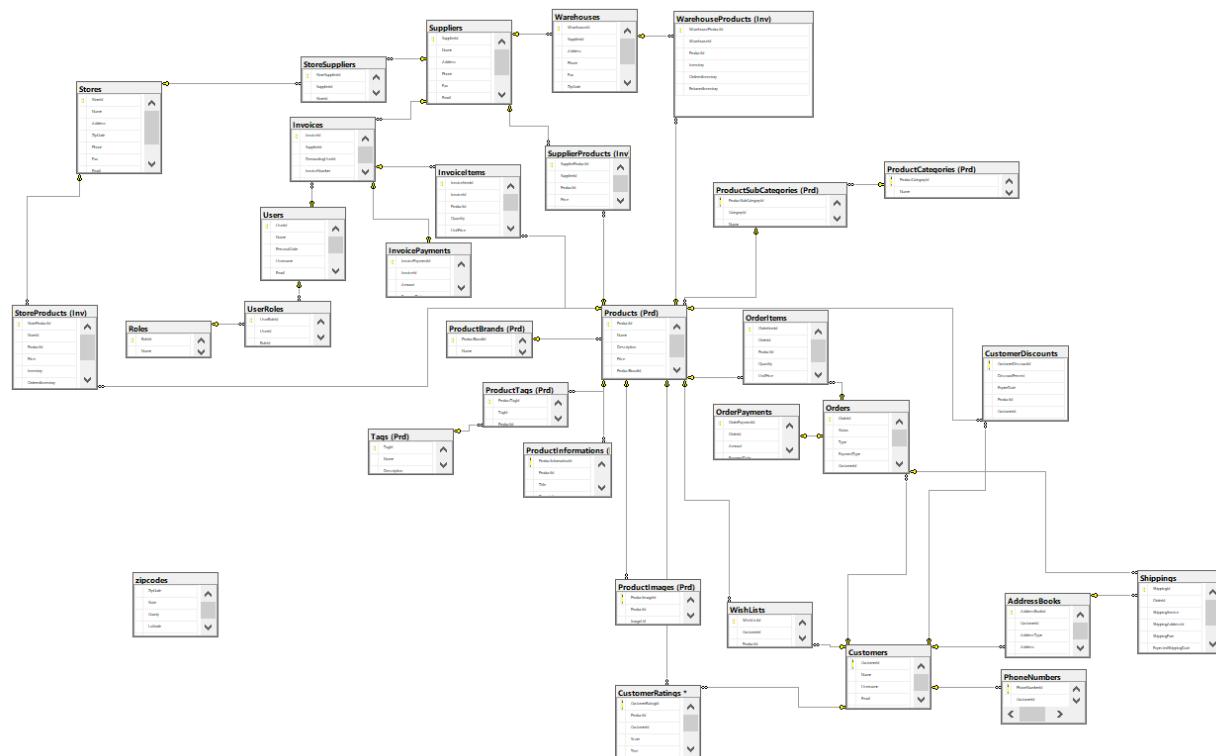
- Stores can place orders to deliver products from warehouses to stores for in-store purchases.
- Each online order can contain multiple products from one or more store warehouses and online sales will get supplied from the nearest store's product inventory.
- Multiple warehouses may have the same product.
- Each online order has a single shipping address.
- Suppliers can have multiple products in various warehouses.

Design Introduction:

The database has got tested with real data and is ready to be implemented in any chain store business. Various real-world scenarios are considered which will be introduced later in the testing section of this report. Following are some of the features of the design:

- ❖ To decrease the shipping costs, the system finds the nearest store location to the customers and the order will get shipped from there.
- ❖ If the first store does not have enough inventory to supply the order, the system will find another store (the second store) to supply the rest of the order items.
- ❖ Product quantity in stores and warehouses gets updated based on transactions.

E/R diagram:



Design considerations, table relationship

Stores-store products: one to many, each store has multiple products.

Store suppliers-stores: many to many

Store-store product: one to many

Product brands-Products: one to many

Products-product tags: many to many

Tags-product tags: one to many

Products-product information: one to many, each product may have multiple information (e.g., a laptop has information for ram and hard, etc.)

Product-product images: one to many, each product may have multiple images.

Product-customers: many to many with the wish list linking table

Product-customers: many to many with the customer ratings linking table

Customer-phone numbers: one to many, each customer may have various phones of different types.

Customer-address books: many to many, each customer may have multiple addresses (billing, shipping, etc.)

Order-shipping: each order has one shipping and each shipping is for one order

Shipping-address books: one to many, there can be multiple shipping that goes to the same address.

Customers-customer discount: one to one, each customer can have a certain amount of coupon on a product

Customer-products: many to many with customer discount linking table, each product may have a different amount of discount for different customers, and each customer may get different discounts on different products.

Product- product subcategories: each category may have different products.

Product subcategories- product categories: each category has various subcategories.

Products-warehouses: many to many, with the warehouse product linking table.

Supplier product-products: one to one

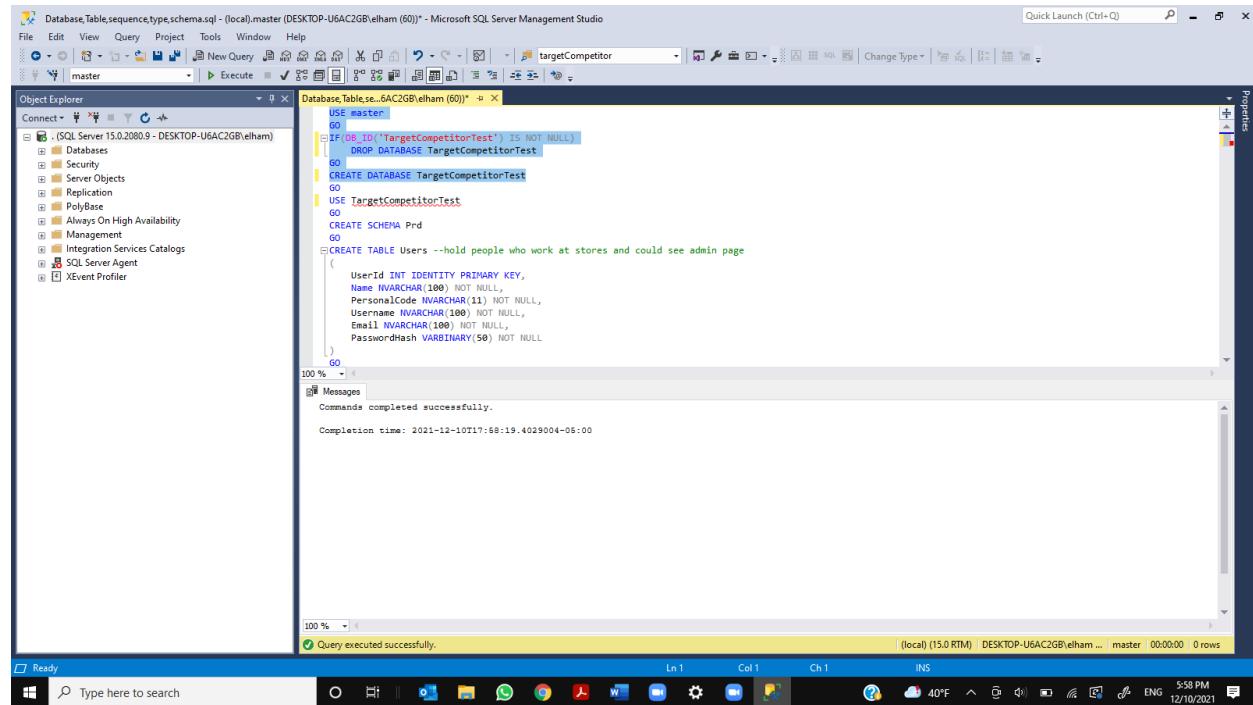
Supplier- supplier products: one to many: each supplier has multiple products.

- Warehouses deliver the products to the stores and when an online order is placed the order will get shipped from the stores to decrease the shipping costs and a fast delivery. Each supplier has various products in different warehouses

Implementation; SQL source codes with comments:

- ✓ All the objects and items in this database has got named using CamelCase naming standard.
- ✓ The design has got normalized into its 3rd normal form

```
USE master
GO
IF(DB_ID('TargetCompetitorTest') IS NOT NULL)
    DROP DATABASE TargetCompetitorTest
GO
CREATE DATABASE TargetCompetitorTest
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'SQL Server 15.0.2080.9 - DESKTOP-U6AC2GB\elham' is selected. A new query window titled 'DatabaseTable.se...6AC2GB\elham (60)*' is open, displaying the following SQL code:

```
USE master
GO
IF(DB_ID('TargetCompetitorTest') IS NOT NULL)
    DROP DATABASE TargetCompetitorTest
GO
CREATE DATABASE TargetCompetitorTest
GO
USE TargetCompetitorTest
GO
CREATE SCHEMA Prd
GO
CREATE TABLE Users --hold people who work at stores and could see admin page
(
    UserId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    PersonId DECIMAL(12,0) NOT NULL,
    Username NVARCHAR(100) NOT NULL,
    Email NVARCHAR(100) NOT NULL,
    PasswordHash VARBINARY(50) NOT NULL
)
GO
```

The 'Messages' pane at the bottom shows the command was executed successfully with a completion time of 2021-12-10T17:58:19.4029004-05:00.

Comment: here we created the database

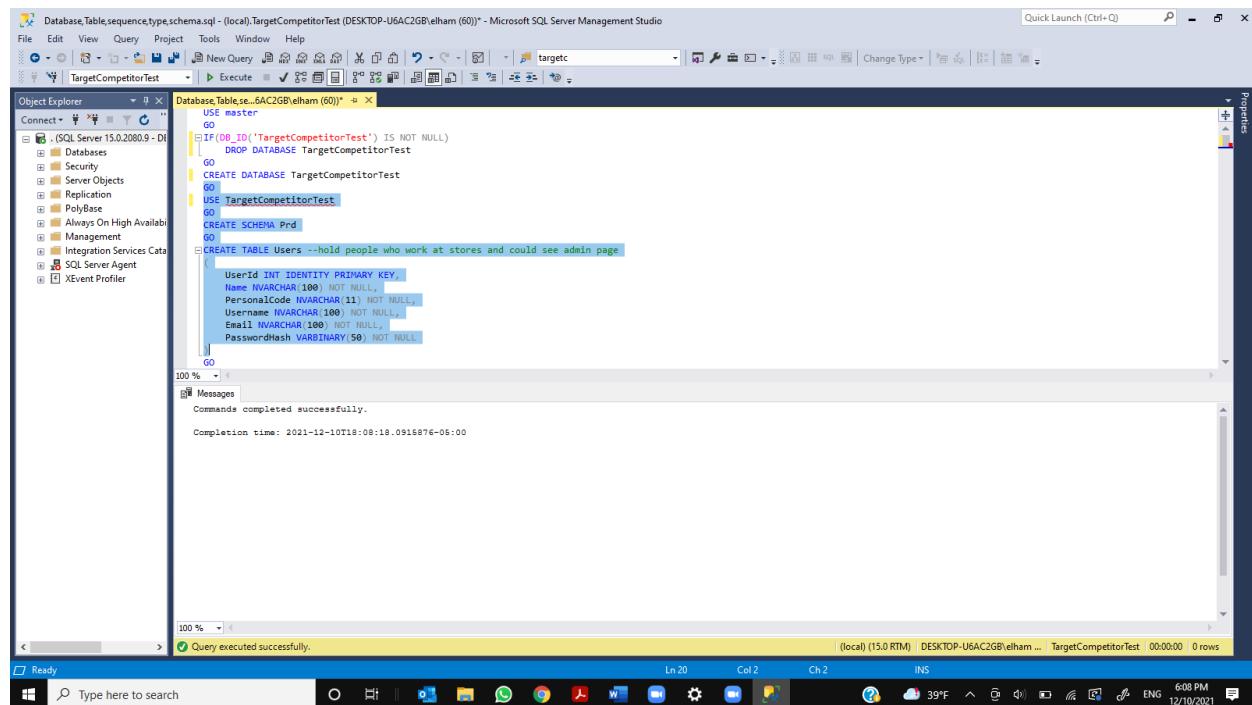
Tables with constraints

- ✓ Schema and tables have got created with columns, primary keys, foreign keys, proper data types, nullabilities, and necessary relationships and constraints

```

GO
USE TargetCompetitorTest
GO
CREATE SCHEMA Prd
GO
CREATE TABLE Users --hold people who work at stores and could see admin page
(
    UserId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    PersonalCode NVARCHAR(11) NOT NULL,
    Username NVARCHAR(100) NOTNULL,
    Email NVARCHAR(100) NOT NULL,
    PasswordHash VARBINARY(50) NOT NULL
)

```



Comment: here we created the created the users table for people who work at stores and could see admin page.

```
GO  
CREATE TABLE Roles -- supply order , product creation , ...  
(  
    RoleId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL  
)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. A query window titled 'Database.Table.sequence.type.schema.sql - (local), TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60))' displays the following T-SQL script:

```
GO  
CREATE TABLE Users -- hold people who work at stores and could see admin page  
(  
    UserId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL,  
    PersonalCode NVARCHAR(11) NOT NULL,  
    Username NVARCHAR(100) NOT NULL,  
    Email NVARCHAR(100) NOT NULL,  
    PasswordHash VARBINARY(50) NOT NULL  
)  
GO  
CREATE TABLE Roles -- supply order , product creation , ...  
(  
    RoleId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL  
)  
GO  
CREATE TABLE UserRoles  
(  
    UserRoleID INT IDENTITY PRIMARY KEY,  
    UserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId),  
    RoleId INT NOT NULL FOREIGN KEY REFERENCES Roles(RoleId)  
)
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2021-12-10T18:12:59.8188730-05:00".

Comment: here we created Roles table.

```
GO
```

```
CREATE TABLE UserRoles
(
    UserRoleID INT IDENTITY PRIMARY KEY,
    UserID INT NOT NULL FOREIGN KEY REFERENCES Users(UserID),
    RoleID INT NOT NULL FOREIGN KEY REFERENCES Roles(RoleID)
)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, a query window displays the following SQL code:

```
CREATE TABLE Roles -- supply order , product creation , ...
(
    RoleId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL
)
GO
CREATE TABLE UserRoles
(
    UserRoleID INT IDENTITY PRIMARY KEY,
    UserID INT NOT NULL FOREIGN KEY REFERENCES Users(UserID),
    RoleID INT NOT NULL FOREIGN KEY REFERENCES Roles(RoleID)
)
GO
CREATE TABLE Customers
(
    CustomerId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Username NVARCHAR(100) NOT NULL,
    Email NVARCHAR(100) NOT NULL,
    PasswordHash VARBINARY(50) NOT NULL
)
```

Below the code, the message pane shows:

- Messages: Commands completed successfully.
- Completion time: 2021-12-10T18:15:49.8734353-05:00

The status bar at the bottom indicates: (local) (15.0 RTM) | DESKTOP-U6AC2GB\elham ... | TargetCompetitorTest | 00:00:00 | 0 rows | 100 % |

Comment: here we created the user roles table.

```
GO  
CREATE TABLE Customers  
(  
    CustomerId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL,  
    Username NVARCHAR(100) NOT NULL,  
    Email NVARCHAR(100) NOT NULL,  
    PasswordHash VARBINARY(50) NOT NULL  
)
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating the 'Customers' table. The code includes a primary key constraint and several non-null columns for name, username, email, and password hash. A GO statement is present at the end of the table definition. Below the code, the 'Messages' pane shows a successful execution message: 'Commands completed successfully.' and 'Completion time: 2021-12-10T18:17:39.6370419-05:00'. The status bar at the bottom indicates the session is ready.

Comment: here we created the customer table.

GO**CREATE TABLE** PhoneNumbers**(**

```
PhoneNumberId INT IDENTITY PRIMARY KEY,  
CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
PhoneType NVARCHAR(8) NOT NULL,  
PhoneNumber NVARCHAR(10) NOT NULL
```

)

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the following T-SQL code:

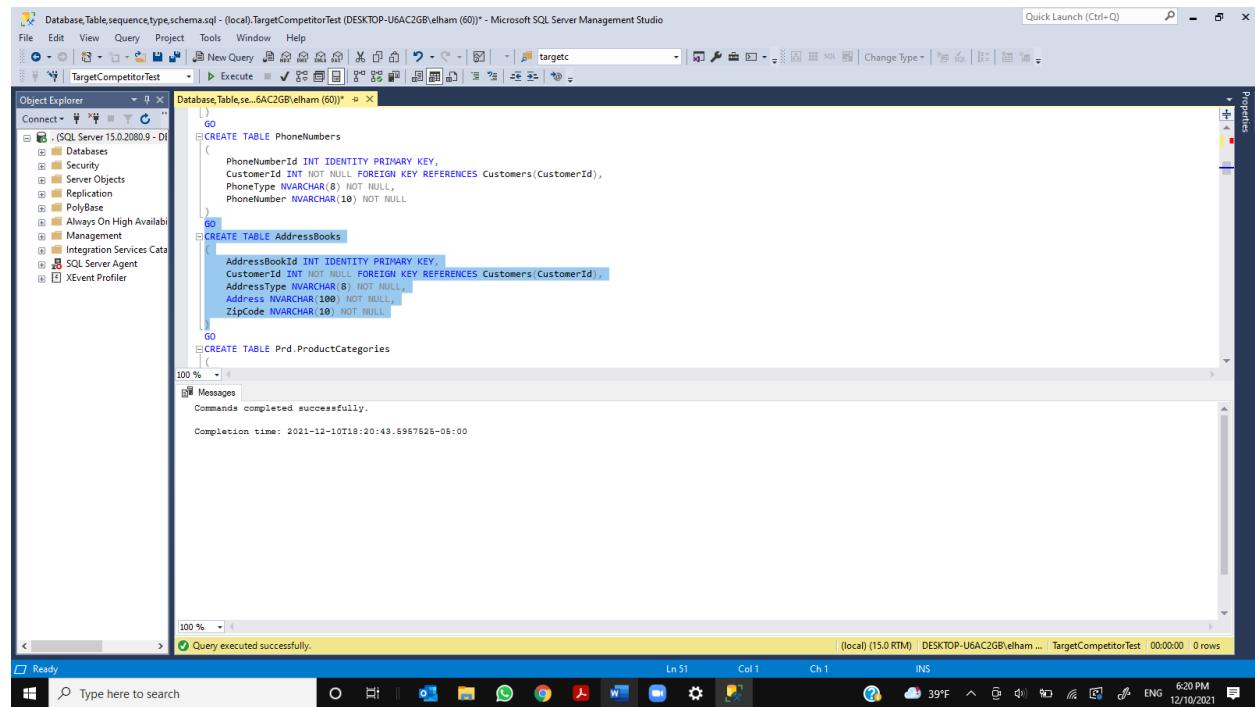
```
CREATE TABLE Customers  
(  
    CustomerId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL,  
    Username NVARCHAR(100) NOT NULL,  
    Email NVARCHAR(100) NOT NULL,  
    PasswordHash VARBINARY(50) NOT NULL  
)  
GO  
CREATE TABLE PhoneNumbers  
(  
    PhoneNumberId INT IDENTITY PRIMARY KEY,  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    PhoneType NVARCHAR(8) NOT NULL,  
    PhoneNumber NVARCHAR(10) NOT NULL  
)  
GO  
CREATE TABLE AddressBooks  
(  
    AddressBookId INT IDENTITY PRIMARY KEY,  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    Address NVARCHAR(100) NOT NULL  
)  
GO
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2021-12-10T18:17:39.6370419-06:00".

Comment: here we created the phone number table.

```
GO  
CREATE TABLE AddressBooks
```

```
(  
    AddressBookId INT IDENTITY PRIMARY KEY,  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    AddressType NVARCHAR(8) NOT NULL,  
    Address NVARCHAR(100) NOT NULL,  
    ZipCode NVARCHAR(10) NOT NULL  
)
```



Comment: here we created the address book table.

GO**CREATE TABLE** Prd.ProductCategories

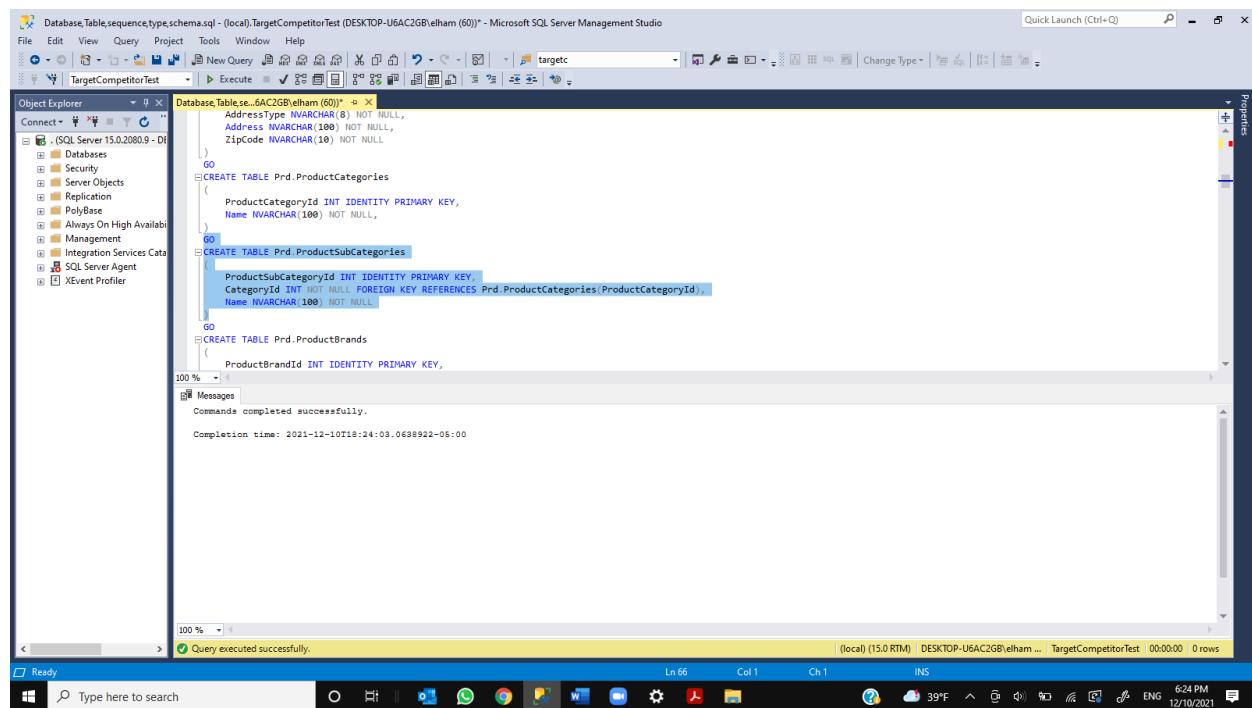
```
(  
    ProductCategoryId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL,  
)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating the 'Prd.ProductCategories' table. The code includes the table definition with an identity primary key and a non-null name constraint, followed by a GO command. Below the code, the message 'Commands completed successfully.' is displayed, along with the completion time: 2021-12-10T18:22:04.0376527-05:00. The status bar at the bottom shows the session details: (local) (15.0 RTM) | DESKTOP-U6AC2GB\elham ... | TargetCompetitorTest | 00:00:00 | 0 rows.

```
File Edit View Query Project Tools Window Help  
Quick Launch (Ctrl+Q)   
Database,Table,sequence,type,schema.sql - (local),TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio  
File Edit View Query Project Tools Window Help  
Execute ✓   
Object Explorer   
Connect ▾   
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler  
Database.Table.sequence.type.schema.sql - (local),TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) *   
PhoneNumber NVARCHAR(10) NOT NULL  
)  
GO  
CREATE TABLE AddressBooks  
(  
    AddressBookId INT IDENTITY PRIMARY KEY,  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    AddressType NVARCHAR(10) NOT NULL,  
    Address NVARCHAR(100) NOT NULL,  
    ZipCode NVARCHAR(10) NOT NULL  
)  
GO  
CREATE TABLE Prd.ProductCategories  
(  
    ProductCategoryId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL  
)  
GO  
CREATE TABLE Prd.ProductSubCategories  
(  
    ProductSubCategoryId INT IDENTITY PRIMARY KEY,  
    )  
Messages  
Commands completed successfully.  
Completion time: 2021-12-10T18:22:04.0376527-05:00  
100 %  
100 %   
Query executed successfully.  
Ln 60 Col 1 Ch 1 INS  
(local) (15.0 RTM) | DESKTOP-U6AC2GB\elham ... | TargetCompetitorTest | 00:00:00 | 0 rows  
Ready Type here to search O 39°F 6:22 PM ENG 12/10/2021
```

Comment: here we created the product categories table.

```
GO  
CREATE TABLE Prd.ProductSubCategories  
(  
    ProductSubCategoryId INT IDENTITY PRIMARY KEY,  
    CategoryId INT NOT NULL FOREIGN KEY REFERENCES  
Prd.ProductCategories(ProductCategoryId),  
    Name NVARCHAR(100) NOT NULL  
)
```



Comment: here we created the product subcategories table.

```
GO  
CREATE TABLE Prd.ProductBrands  
(  
    ProductBrandId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL  
)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, a script window displays the creation of the `Prd.ProductBrands` table. The table has one column, `ProductBrandId`, defined as an `INT` with `IDENTITY` and `PRIMARY KEY` properties, and another column, `Name`, defined as `NVARCHAR(100)` with `NOT NULL` constraint. The script concludes with a `GO` statement. The status bar at the bottom indicates the command was executed successfully.

```
File Edit View Query Project Tools Window Help  
New Query targetc  
Execute ✓                 
Database.Table.sequence.type.schema.sql - (local),TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio  
Object Explorer Properties  
Connect Connect...  
SQL Server 15.0.2080.9 - Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler  
Database.Table.sequence.type.schema.sql - (local),TargetCompetitorTest  
CREATE TABLE Prd.ProductBrands  
(  
    ProductBrandId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL  
)  
GO  
CREATE TABLE Prd.ProductSubCategories  
(  
    ProductSubCategoryId INT IDENTITY PRIMARY KEY,  
    CategoryId INT NOT NULL FOREIGN KEY REFERENCES Prd.ProductCategories(ProductCategoryId),  
    Name NVARCHAR(100) NOT NULL  
)  
GO  
CREATE TABLE Prd.Products  
(  
    ProductId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL,  
    Description NVARCHAR(250) NULL,  
    UnitPrice DECIMAL(18, 2) NOT NULL  
)  
GO  
Messages  
Commands completed successfully.  
Completion time: 2021-12-10T18:25:12.8882560-05:00  
100 %  
Query executed successfully.  
Ln 73 Col 1 Ch 1 INS  
(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows  
Ready Type here to search 6:25 PM 39°F ENG 12/10/2021
```

Comment: here we created the product brands table.

GO**CREATE TABLE** Prd.Products**(**

```
    ProductId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Description NVARCHAR(250) NULL,
    Price MONEY NOT NULL,
    ProductBrandId INT FOREIGN KEY REFERENCES Prd.ProductBrands(ProductBrandId),
    ProductSubCategoryId INT FOREIGN KEY REFERENCES
```

```
Prd.ProductSubCategories(ProductSubCategoryId)
```

)

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. A query window is open with the following T-SQL code:

```
CREATE TABLE Prd.ProductBrands
(
    ProductBrandId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL
)
GO
CREATE TABLE Prd.Products
(
    ProductId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Description NVARCHAR(250) NULL,
    Price MONEY NOT NULL,
    ProductBrandId INT FOREIGN KEY REFERENCES Prd.ProductBrands(ProductBrandId),
    ProductSubCategoryId INT FOREIGN KEY REFERENCES Prd.ProductSubCategories(ProductSubCategoryId)
)
GO
CREATE TABLE Prd.ProductImages
(
    ProductImageId INT IDENTITY PRIMARY KEY,
    ...
)
GO
```

The execution of the code has completed successfully, as indicated by the message in the status bar: "Query executed successfully".

Comment: here we created the products table.

GO**CREATE TABLE** Prd.ProductImages**(**

```
ProductImageId INT IDENTITY PRIMARY KEY,  
ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
ImageUrl NVARCHAR(500) NOT NULL
```

)

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating the 'Prd.ProductImages' table. The code includes a primary key constraint and a foreign key reference to the 'Products' table. The execution of the script was successful, as indicated by the message 'Commands completed successfully.' at the bottom of the results pane.

```
GO  
CREATE TABLE Prd.ProductImages  
(  
    ProductImageId INT IDENTITY PRIMARY KEY,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    ImageUrl NVARCHAR(500) NOT NULL  
)  
GO  
CREATE TABLE Prd.ProductInformations  
(  
    ProductInformationId INT IDENTITY PRIMARY KEY,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    Title NVARCHAR(200) NOT NULL,  
    Description NVARCHAR(250) NULL,  
    Price MONEY NOT NULL,  
    ProductBrandId INT FOREIGN KEY REFERENCES Prd.ProductBrands(ProductBrandId),  
    ProductSubCategoryId INT FOREIGN KEY REFERENCES Prd.ProductSubCategories(ProductSubCategoryId)  
)
```

Comment: here we created the product images table.

GO**CREATE TABLE** Prd.ProductInformations**(**

```
ProductInformationId INT IDENTITY PRIMARY KEY,  
ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
Title NVARCHAR(200) NOT NULL,  
Description NVARCHAR(500) NOT NULL
```

)

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the following T-SQL code:

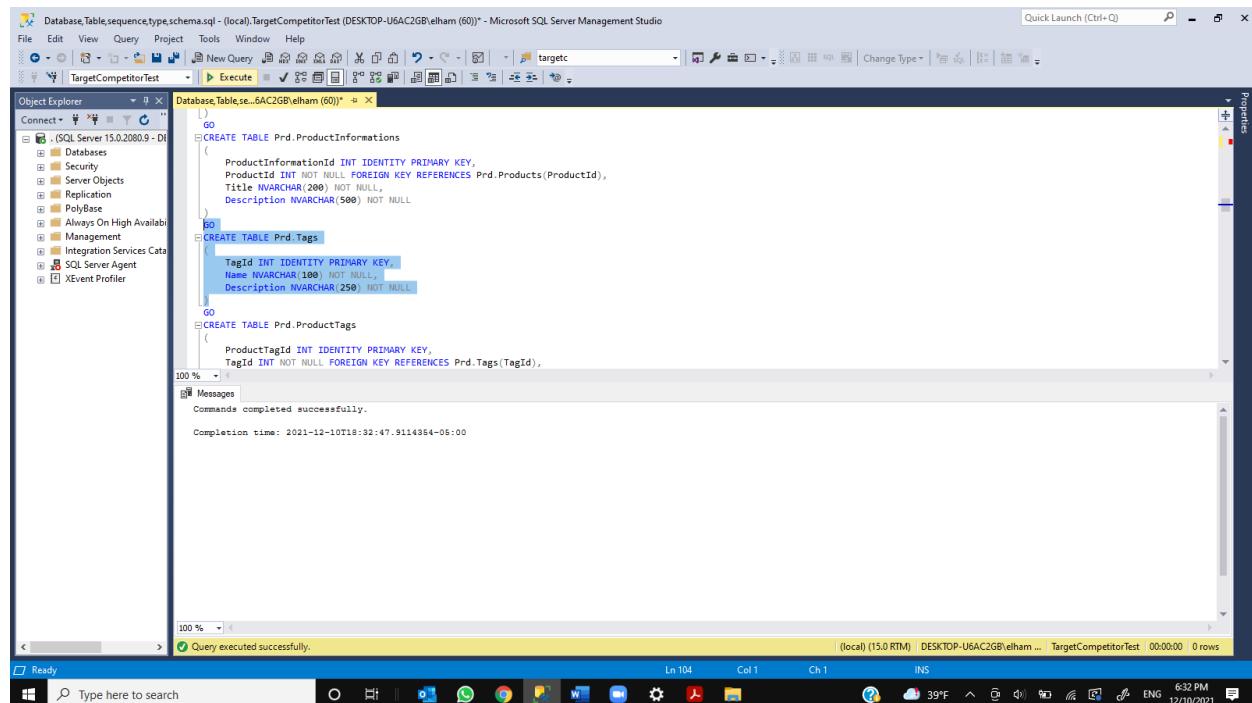
```
CREATE TABLE Prd.ProductImages  
(  
    ProductImageId INT IDENTITY PRIMARY KEY,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    ImageUrl NVARCHAR(500) NOT NULL  
)  
GO  
CREATE TABLE Prd.ProductInformations  
(  
    ProductInformationId INT IDENTITY PRIMARY KEY,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    Title NVARCHAR(200) NOT NULL,  
    Description NVARCHAR(500) NOT NULL  
)  
GO  
CREATE TABLE Prd.Tags  
(  
    TagId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL,
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2021-12-10T18:31:48.0800102-05:00".

Comment: here we created the product information table.

GO
CREATE TABLE Prd.Tags

```
(  
    TagId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(100) NOT NULL,  
    Description NVARCHAR(250) NOT NULL  
)
```



Comment: here we created the tags table.

GO**CREATE TABLE** Prd.ProductTags**(**

```
    ProductTagId INT IDENTITY PRIMARY KEY,  
    TagId INT NOT NULL FOREIGN KEY REFERENCES Prd.Tags(TagId),  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId)
```

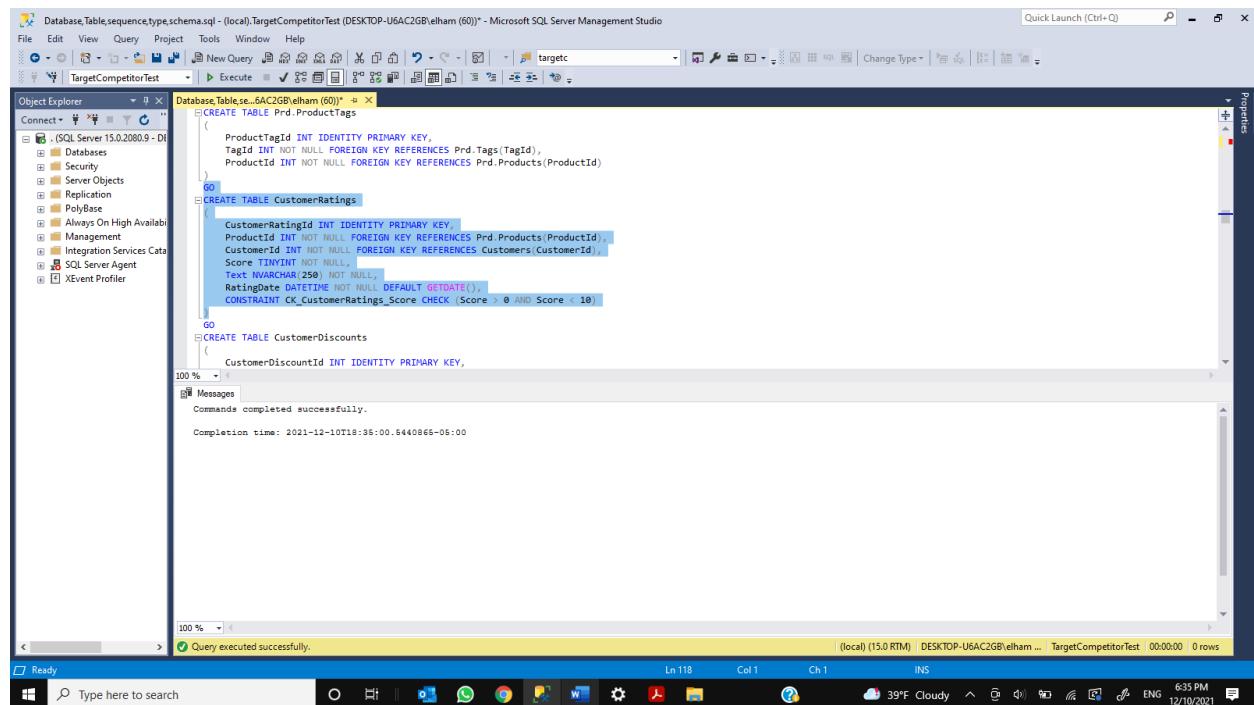
)

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is a query window containing T-SQL code. The code creates a table named 'Prd.ProductTags' with three columns: 'ProductTagId' (INT IDENTITY PRIMARY KEY), 'TagId' (INT NOT NULL FOREIGN KEY REFERENCES 'Prd.Tags'), and 'ProductId' (INT NOT NULL FOREIGN KEY REFERENCES 'Prd.Products'). The code is preceded by a GO statement and followed by a CREATE TABLE statement for 'CustomerRatings'. The status bar at the bottom indicates the command was executed successfully.

```
File Edit View Query Project Tools Window Help  
New Query Save All Close All New Query Save As... Open Recent... Execute Cancel Run Stop Refresh Properties  
Database,Table,sequence,type,schema.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio  
File Edit View Project Tools Window Help  
New Query Save All Close All New Query Save As... Open Recent... Execute Cancel Run Stop Refresh Properties  
Object Explorer  
Connect Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler  
Database,Table,sequence,type,schema.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60))  
CREATE TABLE Prd.ProductTags  
(  
    ProductTagId INT IDENTITY PRIMARY KEY,  
    TagId INT NOT NULL FOREIGN KEY REFERENCES Prd.Tags(TagId),  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId)  
GO  
CREATE TABLE CustomerRatings  
(  
    CustomerRatingId INT IDENTITY PRIMARY KEY,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    Score TINYINT NOT NULL,  
    Text NVARCHAR(250) NOT NULL,  
    RatingDate DATETIME NOT NULL DEFAULT GETDATE(),  
    RatingText NVARCHAR(250) NOT NULL  
);  
GO  
100 % < >  
Messages  
Commands completed successfully.  
Completion time: 2021-12-10T18:34:11.7998184-05:00  
Ln 111 Col 1 Ch 1 INS  
Ready Type here to search (local) (15.0 RTM) | DESKTOP-U6AC2GB\elham ... | TargetCompetitorTest | 00:00:00 | 0 rows  
6:34 PM 39°F ENG 12/10/2021
```

Comment: here we created the product tags linking table.

```
GO  
CREATE TABLE CustomerRatings  
(  
    CustomerRatingId INT IDENTITY PRIMARY KEY,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    Score DECIMAL(4,2) NOT NULL,  
    Text NVARCHAR(250) NOT NULL,  
    RatingDate DATETIME NOT NULL DEFAULT GETDATE(),  
    CONSTRAINT CK_CustomerRatings_Score CHECK (Score > 0 AND Score < 10)  
)
```



Comment: here we created the customer ratings table.

```

GO
CREATE TABLE CustomerDiscounts
(
    CustomerDiscountId INT IDENTITY PRIMARY KEY,
    DiscountPercent DECIMAL(3,0) NOT NULL,
    ExpireDate DATETIME NOT NULL,
    ProductId INT FOREIGN KEY REFERENCES Prd.Products(ProductId),
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),
    CONSTRAINT CK_CustomerDiscounts_Discount CHECK(DiscountPercent > 0 AND DiscountPercent
    <= 100),
    CONSTRAINT CK_CustomerDiscounts_ExpireDate CHECK(ExpireDate > GETDATE())
)

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. A query window titled 'Database_Table_sequence_type_schema.sql - (local)\TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60))' displays the following T-SQL code:

```

GO
CREATE TABLE CustomerDiscounts
(
    CustomerDiscountId INT IDENTITY PRIMARY KEY,
    DiscountPercent DECIMAL(3,0) NOT NULL,
    ExpireDate DATETIME NOT NULL,
    ProductId INT FOREIGN KEY REFERENCES Prd.Products(ProductId),
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),
    CONSTRAINT CK_CustomerDiscounts_Discount CHECK(DiscountPercent > 0 AND DiscountPercent
    <= 100),
    CONSTRAINT CK_CustomerDiscounts_ExpireDate CHECK(ExpireDate > GETDATE())
)

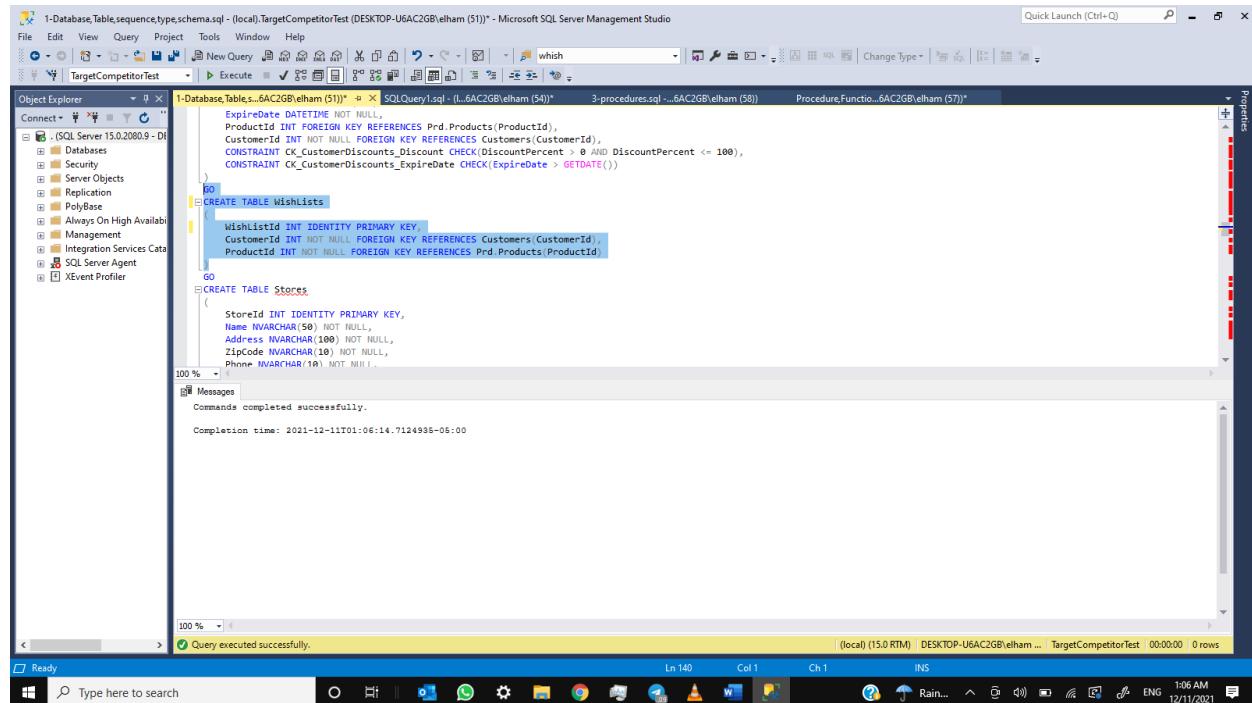
```

The code is highlighted in blue, indicating it is syntax-highlighted T-SQL. Below the code, the 'Messages' pane shows the output: 'Commands completed successfully.' and 'Completion time: 2021-12-10T18:37:02.2083011-05:00'. The status bar at the bottom right shows the system is running at 38°F Wind, 6:37 PM, and the date is 12/10/2021.

Comment: here we created the customer discount table.

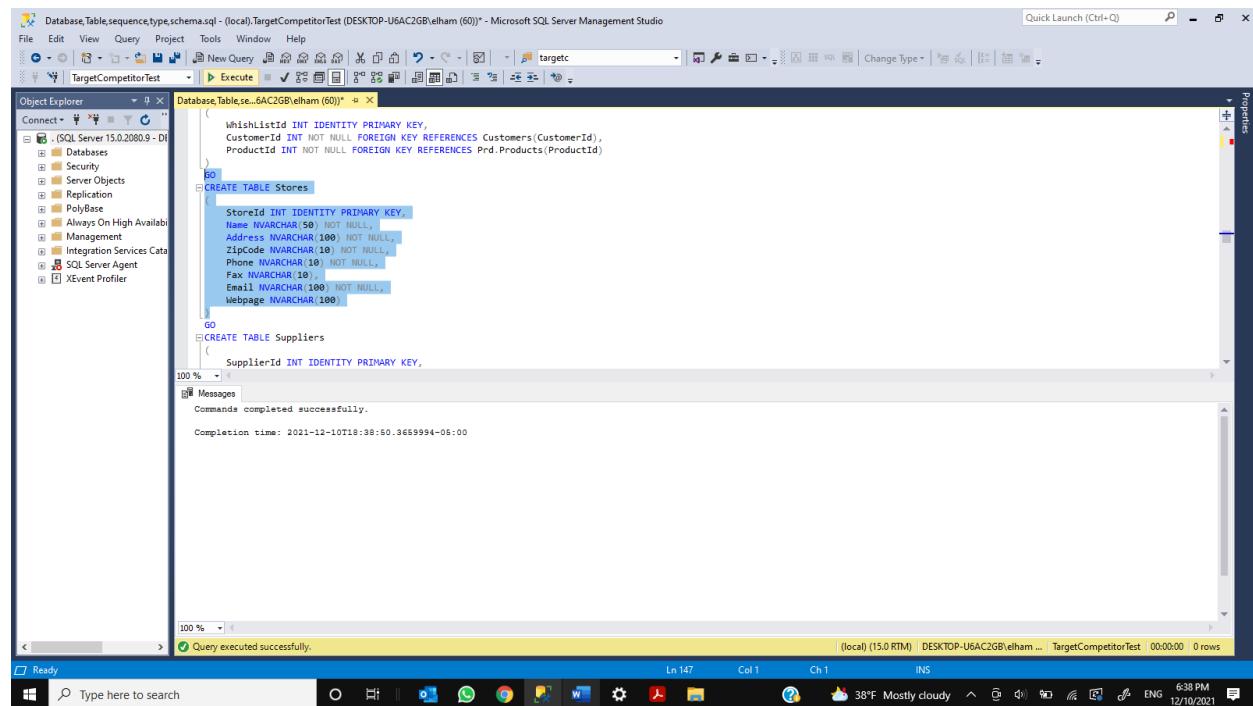
GO
CREATE TABLE WishLists

```
( WishListId INT IDENTITY PRIMARY KEY,  
CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId)  
)
```



Comment: here we created the wish lists table.

```
GO  
CREATE TABLE Stores  
(  
    StoreId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(50) NOT NULL,  
    Address NVARCHAR(100) NOT NULL,  
    ZipCode NVARCHAR(10) NOT NULL,  
    Phone NVARCHAR(10) NOT NULL,  
    Fax NVARCHAR(10),  
    Email NVARCHAR(100) NOT NULL,  
    Webpage NVARCHAR(100)  
)
```

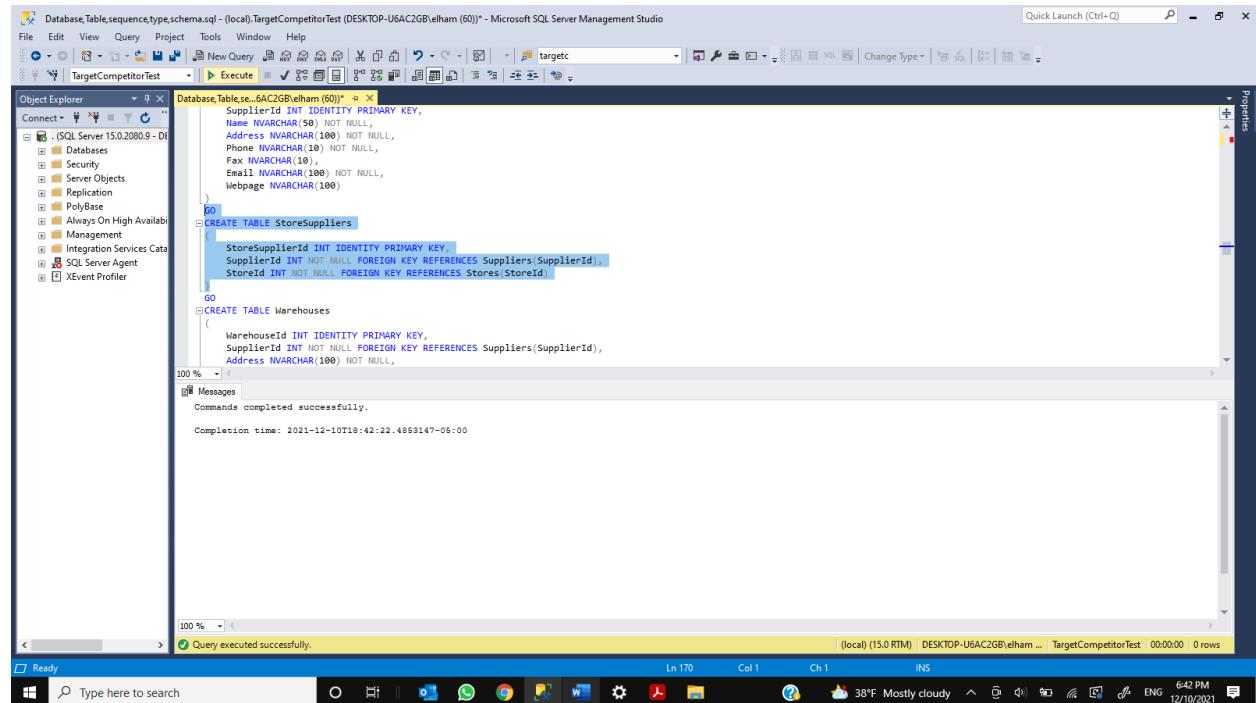


Comment: here we created the stores table.

```
GO  
CREATE TABLE Suppliers  
(  
    SupplierId INT IDENTITY PRIMARY KEY,  
    Name NVARCHAR(50) NOT NULL,  
    Address NVARCHAR(100) NOT NULL,  
    Phone NVARCHAR(10) NOT NULL,  
    Fax NVARCHAR(10),  
    Email NVARCHAR(100) NOT NULL,  
    Webpage NVARCHAR(100)  
)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating the 'Suppliers' table. The code defines the table structure with columns for SupplierId (primary key, identity), Name (NVARCHAR(50)), Address (NVARCHAR(100)), Phone (NVARCHAR(10)), Fax (NVARCHAR(10)), Email (NVARCHAR(100)), and Webpage (NVARCHAR(100)). The 'GO' command is used to execute the CREATE TABLE statement. Below the code, another 'GO' command is present, followed by a CREATE TABLE statement for 'StoreSuppliers'. The status bar at the bottom indicates 'Query executed successfully.' and shows the completion time as 2021-12-10T18:41:19.4910909-05:00.

Comment: here we created the supplier's table.

GO**CREATE TABLE** StoreSuppliers**(****StoreSupplierId INT IDENTITY PRIMARY KEY,**
SupplierId INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierId),
StoreId INT NOT NULL FOREIGN KEY REFERENCES Stores(StoreId)**)**

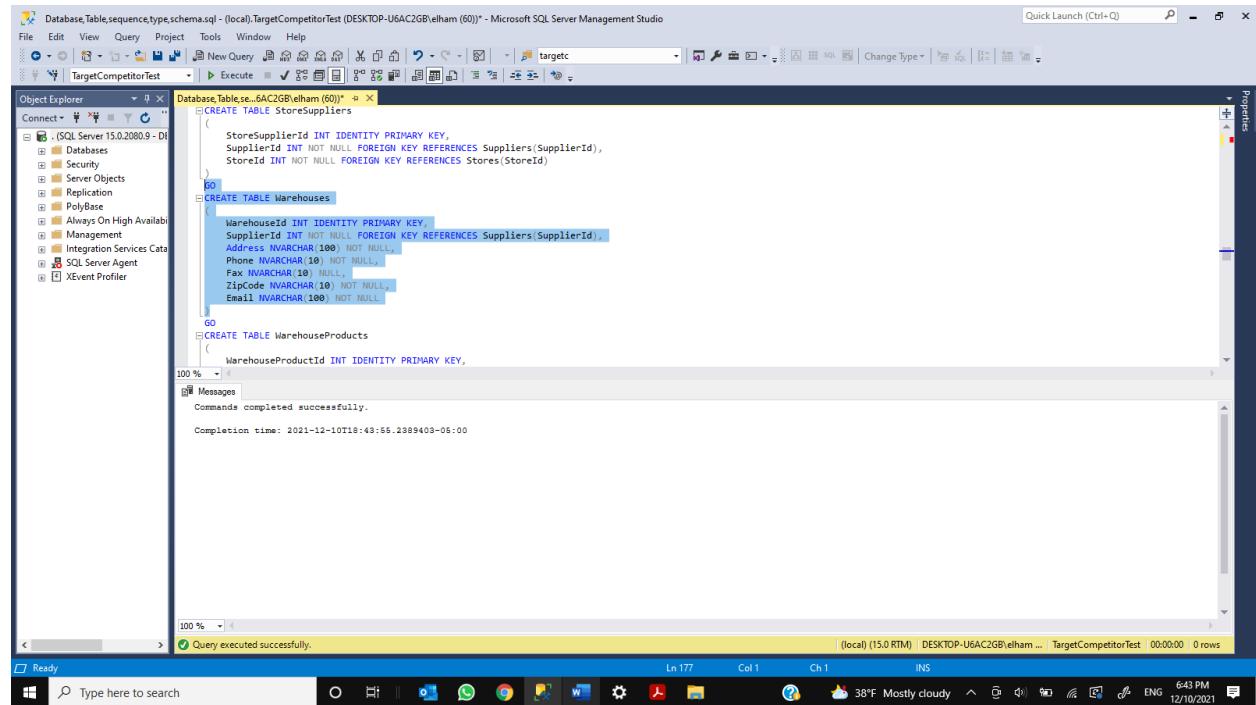
The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer on the left, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating the 'StoreSuppliers' table. The code includes a primary key constraint on 'StoreSupplierId' and foreign key constraints linking 'SupplierId' to 'Suppliers(SupplierId)' and 'StoreId' to 'Stores(StoreId)'. The 'Messages' section at the bottom of the query window shows a green checkmark indicating that the command was executed successfully. The status bar at the bottom right shows the completion time as 2021-12-10T18:42:22.4853147-05:00.

```
GO
CREATE TABLE StoreSuppliers
(
    StoreSupplierId INT IDENTITY PRIMARY KEY,
    SupplierId INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierId),
    StoreId INT NOT NULL FOREIGN KEY REFERENCES Stores(StoreId)
)
GO
CREATE TABLE Warehouses
(
    WarehouseId INT IDENTITY PRIMARY KEY,
    SupplierId INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierId),
    Address NVARCHAR(100) NOT NULL,
    Email NVARCHAR(100) NOT NULL,
    Webpage NVARCHAR(100)
)
GO
CREATE TABLE Suppliers
(
    SupplierId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(50) NOT NULL,
    Address NVARCHAR(100) NOT NULL,
    Phone NVARCHAR(10) NOT NULL,
    Fax NVARCHAR(10),
    Email NVARCHAR(100) NOT NULL,
    Webpage NVARCHAR(100)
)
GO
CREATE TABLE Stores
(
    StoreId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(50) NOT NULL,
    Address NVARCHAR(100) NOT NULL,
    Phone NVARCHAR(10) NOT NULL,
    Fax NVARCHAR(10),
    Email NVARCHAR(100) NOT NULL,
    Webpage NVARCHAR(100)
)
GO
```

Comment: here we created the store-suppliers linking table.

GO**CREATE TABLE** Warehouses**(**

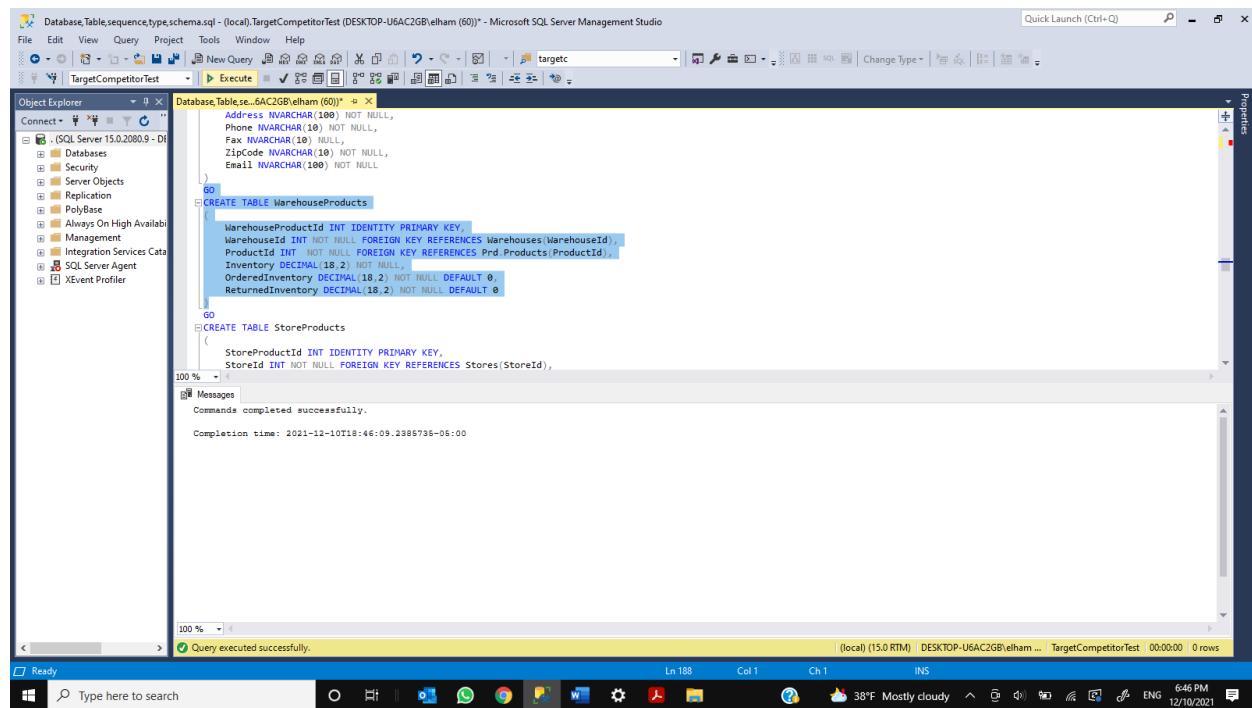
```
Warehouseld INT IDENTITY PRIMARY KEY,  
SupplierId INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierId),  
Address NVARCHAR(100) NOT NULL,  
Phone NVARCHAR(10) NOT NULL,  
Fax NVARCHAR(10) NULL,  
ZipCode NVARCHAR(10) NOT NULL,  
Email NVARCHAR(100) NOT NULL
```

)

Comment: here we created the warehouses table.

GO
CREATE TABLE WarehouseProducts

```
(  
    WarehouseProductId INT IDENTITY PRIMARY KEY,  
    WarehouseId INT NOT NULL FOREIGN KEY REFERENCES Warehouses(WarehouseId),  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    Inventory DECIMAL(18,2) NOT NULL,  
    OrderedInventory DECIMAL(18,2) NOT NULL DEFAULT 0,  
    ReturnedInventory DECIMAL(18,2) NOT NULL DEFAULT 0  
)
```



Comment: here we created the warehouse-products linking table. There are columns to show the inventory, ordered inventory, and returned inventory.

```

USE TargetCompetitorTest
GO
CREATE TABLE inv.StoreProducts
(
    StoreProductId INT IDENTITY PRIMARY KEY,
    StoreId INT NOT NULL FOREIGN KEY REFERENCES Stores(StoreId),
    ProductId INT NOT NULL FOREIGN KEY REFERENCES prd.Products(ProductId),
    Price MONEY NOT NULL,
    Inventory DECIMAL(18,2) NOT NULL,
    OrderedInventory DECIMAL(18,2) NOT NULL DEFAULT 0,
    ReturnedInventory DECIMAL(18,2) NOT NULL DEFAULT 0
)

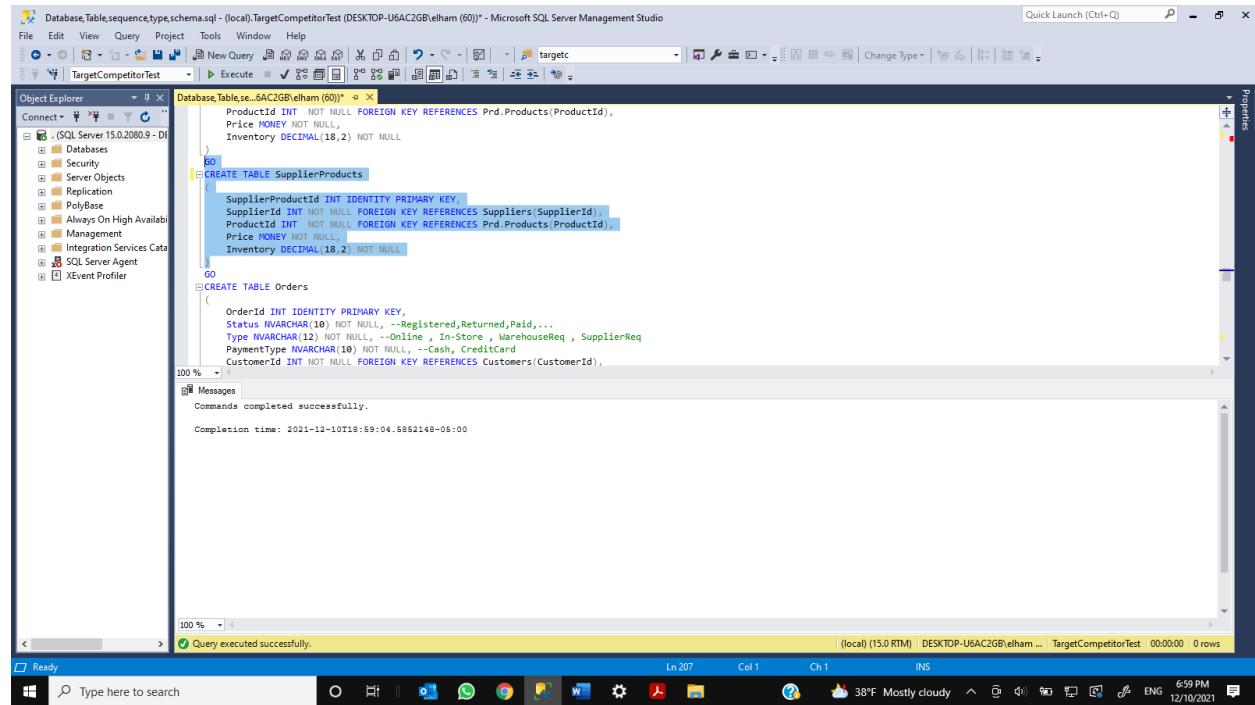
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists several databases and their objects, including 'TargetCompetitorTest' which contains tables like 'Stores', 'Products', and 'Customers'. The central pane displays the T-SQL code for creating the 'inv.StoreProducts' table. The code defines a primary key 'StoreProductId' with an identity increment, foreign keys to 'Stores' and 'Products', and columns for price, inventory, ordered inventory, and returned inventory. The 'Messages' pane at the bottom indicates that the command completed successfully.

Comment: here we created the store-products linking table.

```
GO
CREATE TABLE SupplierProducts
```

```
(  
    SupplierProductId INT IDENTITY PRIMARY KEY,  
    SupplierId INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierId),  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    Price MONEY NOT NULL,  
    Inventory DECIMAL(18,2) NOT NULL  
)
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating the 'SupplierProducts' table. The code includes the table definition with columns for SupplierProductId (identity primary key), SupplierId (foreign key to Suppliers), ProductId (foreign key to Prd.Products), Price (money type), and Inventory (decimal type). The 'Messages' section at the bottom of the query window shows a green checkmark indicating 'Query executed successfully.' and the completion time '2021-12-10T18:59:04.5852148-05:00'.

```
File Edit View Query Project Tools Window Help
Database.Table.sequence.type.schema.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio
Object Explorer
Connect
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
Database.Table.sequence.schema.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60))
ProductID INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductID),
Price MONEY NOT NULL,
Inventory DECIMAL(18,2) NOT NULL
)
GO
CREATE TABLE SupplierProducts
(
    SupplierProductId INT IDENTITY PRIMARY KEY,
    SupplierId INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierId),
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),
    Price MONEY NOT NULL,
    Inventory DECIMAL(18,2) NOT NULL
)
GO
CREATE TABLE Orders
(
    OrderId INT IDENTITY PRIMARY KEY,
    Status NVARCHAR(10) NOT NULL, --Registered,Returned,Paid...
    Type NVARCHAR(12) NOT NULL, --Online , In-Store , WarehouseReq , SupplierReq
    PaymentType NVARCHAR(10) NOT NULL, --Cash, CreditCard
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),
)
GO
Messages
Commands completed successfully.
Completion time: 2021-12-10T18:59:04.5852148-05:00
100 %
Ready
Type here to search
Ln 207 Col 1 Ch 1 INS
((local) (15.0 RTM) | DESKTOP-U6AC2GB\elham ... | TargetCompetitorTest | 00:00:00 | 0 rows
6:59 PM 38°F Mostly cloudy ENG 12/10/2021
```

Comment: here we created the supplier-products linking table.

```
GO
```

```
CREATE TABLE Orders
```

```
(
```

```
    OrderId INT IDENTITY PRIMARY KEY,  
    Status NVARCHAR(10) NOT NULL, --Registered,Returned,Paid,...  
    Type NVARCHAR(12) NOT NULL, --Online , In-Store , WarehouseReq , SupplierReq  
    PaymentType NVARCHAR(10) NOT NULL, --Cash, CreditCard  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    OrderDate DATETIME NOT NULL DEFAULT GETDATE()
```

```
)
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. A query window is open with the following T-SQL code:

```
Database,Table,sequence,type,schema.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (00)) - Microsoft SQL Server Management Studio
```

```
File Edit View Query Project Tools Window Help
```

```
Object Explorer
```

```
Connect (SQL Server 15.0.2080.9 - D)
```

```
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
```

```
targetc Execute
```

```
GO
```

```
CREATE TABLE Orders
```

```
    OrderId INT IDENTITY PRIMARY KEY,  
    Status NVARCHAR(10) NOT NULL, --Registered,Returned,Paid,...  
    Type NVARCHAR(12) NOT NULL, --Online , In-Store , WarehouseReq , SupplierReq  
    PaymentType NVARCHAR(10) NOT NULL, --Cash, CreditCard  
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),  
    OrderDate DATETIME NOT NULL DEFAULT GETDATE()
```

```
GO
```

```
CREATE TABLE OrderItems
```

```
    OrderItemId INT IDENTITY PRIMARY KEY,
```

```
100 % < >
```

```
Messages
```

```
Commands completed successfully.
```

```
Completion time: 2021-12-10T19:00:21.6837672-05:00
```

```
100 % < >
```

```
Query executed successfully.
```

```
Ln 216 Col 1 Ch 1 INS
```

```
(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows
```

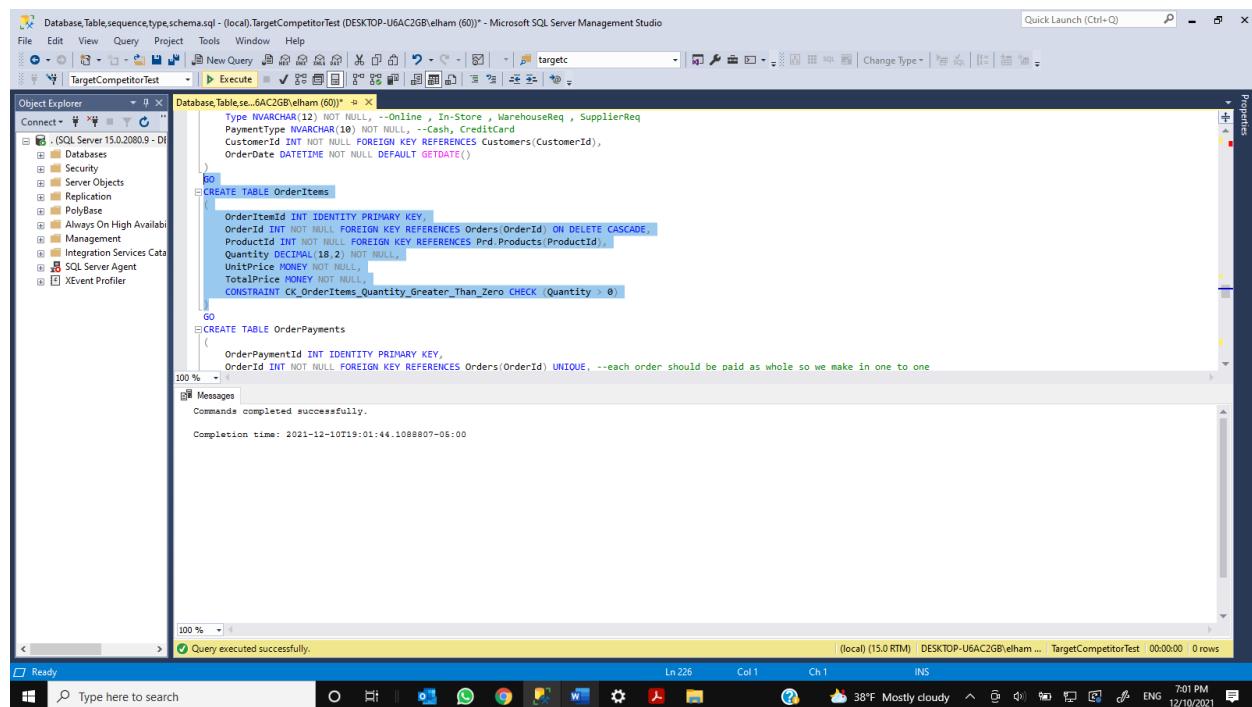
```
Type here to search
```

```
O 38°F Mostly cloudy 7:00 PM ENG 12/10/2021
```

The code creates the 'Orders' table with columns: OrderId (INT IDENTITY PRIMARY KEY), Status (NVARCHAR(10) NOT NULL), Type (NVARCHAR(12) NOT NULL), PaymentType (NVARCHAR(10) NOT NULL), CustomerId (INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId)), and OrderDate (DATETIME NOT NULL DEFAULT GETDATE()). It also creates the 'OrderItems' table with a primary key column OrderItemId (INT IDENTITY PRIMARY KEY). The command is completed successfully at 2021-12-10T19:00:21.6837672-05:00.

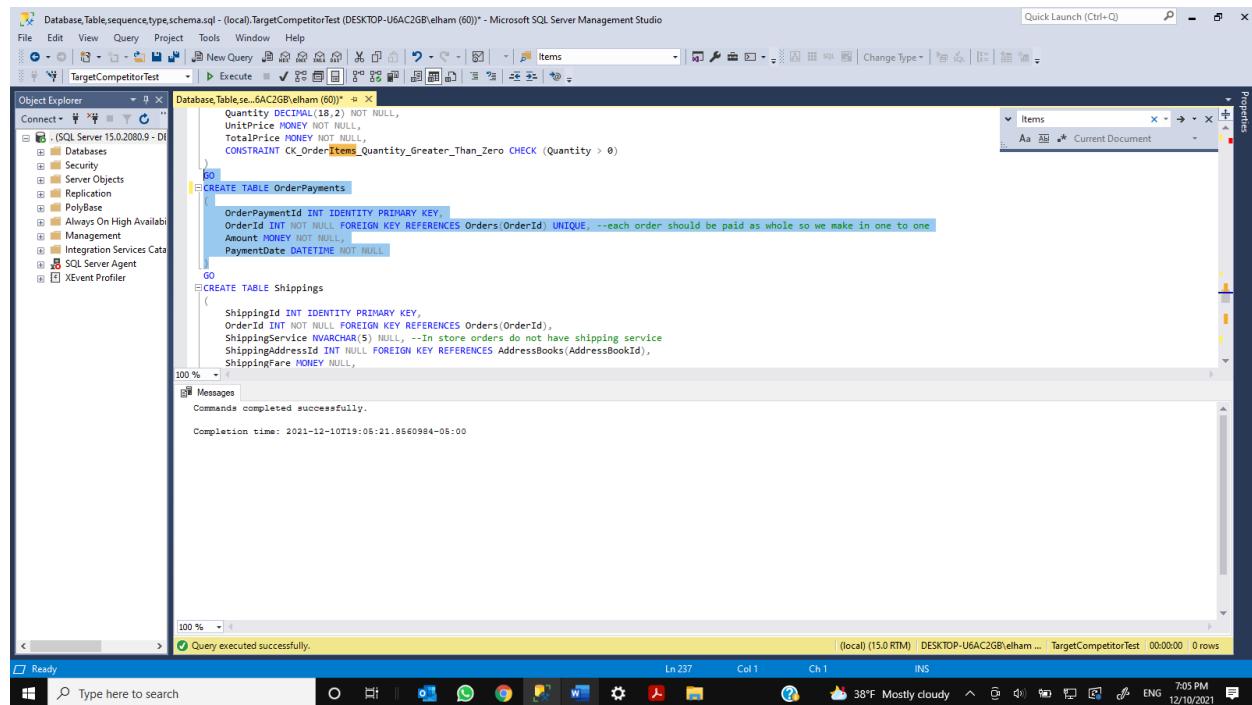
Comment: here we created the orders table.

```
GO  
CREATE TABLE OrderItems  
(  
    OrderItemId INT IDENTITY PRIMARY KEY,  
    OrderId INT NOT NULL FOREIGN KEY REFERENCES Orders(OrderId) ON DELETE CASCADE,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    Quantity DECIMAL(18,2) NOT NULL,  
    UnitPrice MONEY NOT NULL,  
    TotalPrice MONEY NOT NULL,  
    CONSTRAINT CK_OrderItems_Quantity_Greater_Than_Zero CHECK (Quantity > 0)  
)
```



Comment: here we created the order-items linking table.

```
GO  
CREATE TABLE OrderPayments  
(  
    OrderPaymentId INT IDENTITY PRIMARY KEY,  
    OrderId INT NOT NULL FOREIGN KEY REFERENCES Orders(OrderId) UNIQUE, --each order  
should be paid as whole so we make in one to one  
    Amount MONEY NOT NULL,  
    PaymentDate DATETIME NOT NULL  
)
```



Comment: here we created the order payment table.

```
GO
```

```
CREATE TABLE Shipments
```

```
(
```

```
    ShippingId INT IDENTITY PRIMARY KEY,  
    OrderId INT NOT NULL FOREIGN KEY REFERENCES Orders(OrderId),  
    ShippingService NVARCHAR(5) NULL, --In store orders do not have shipping service  
    ShippingAddressId INT NULL FOREIGN KEY REFERENCES AddressBooks(AddressBookId),  
    ShippingFare MONEY NULL,  
    ExpectedShippingDate DATETIME NULL,  
    ActualShippingDate DATETIME NULL
```

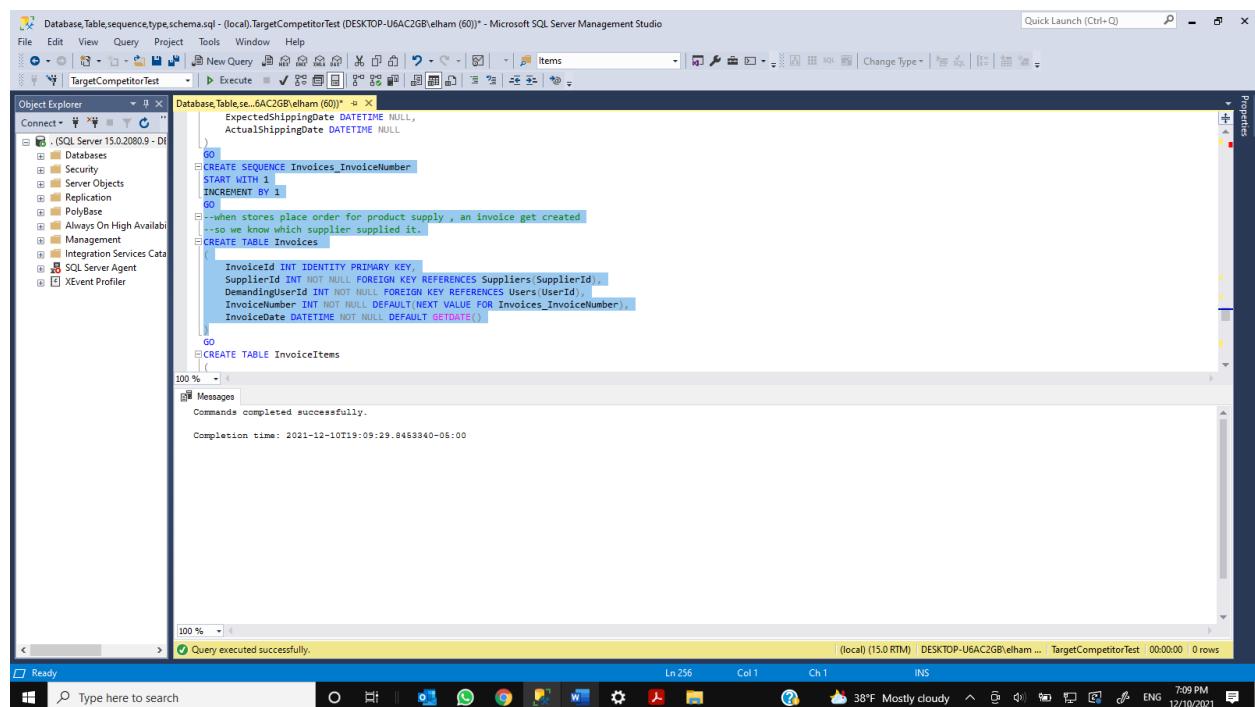
```
)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating the 'Shipments' table. The code includes a primary key constraint on 'ShippingId', a foreign key constraint on 'OrderId' referencing the 'Orders' table, and several other columns like 'ShippingService' and 'ShippingFare'. The code concludes with a 'GO' command. Below the code, the 'Messages' pane shows the message 'Commands completed successfully.' and the completion time '2021-12-10T19:06:39.6648805-05:00'. At the bottom of the screen, the Windows taskbar is visible with various icons.

```
File Edit View Query Project Tools Window Help  
New Query New Item Items Change Type  
Execute  
Database,Table,sequence,type,schema.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio  
Object Explorer  
Connect ▾ Database  
File System Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler  
Database Table Sequence Type Schema  
CREATE TABLE Shipments  
(  
    ShippingId INT IDENTITY PRIMARY KEY,  
    OrderId INT NOT NULL FOREIGN KEY REFERENCES Orders(OrderId) UNIQUE, --each order should be paid as whole so we make in one to one  
    Amount MONEY NOT NULL,  
    PaymentDate DATETIME NOT NULL  
)  
CREATE SEQUENCE Invoices InvoiceNumber  
100 % ▾  
Messages  
Commands completed successfully.  
Completion time: 2021-12-10T19:06:39.6648805-05:00  
Ln 245 Col 1 Ch 1 INS  
Ready Type here to search (local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows  
O 38°F Mostly cloudy 7:06 PM ENG 12/10/2021
```

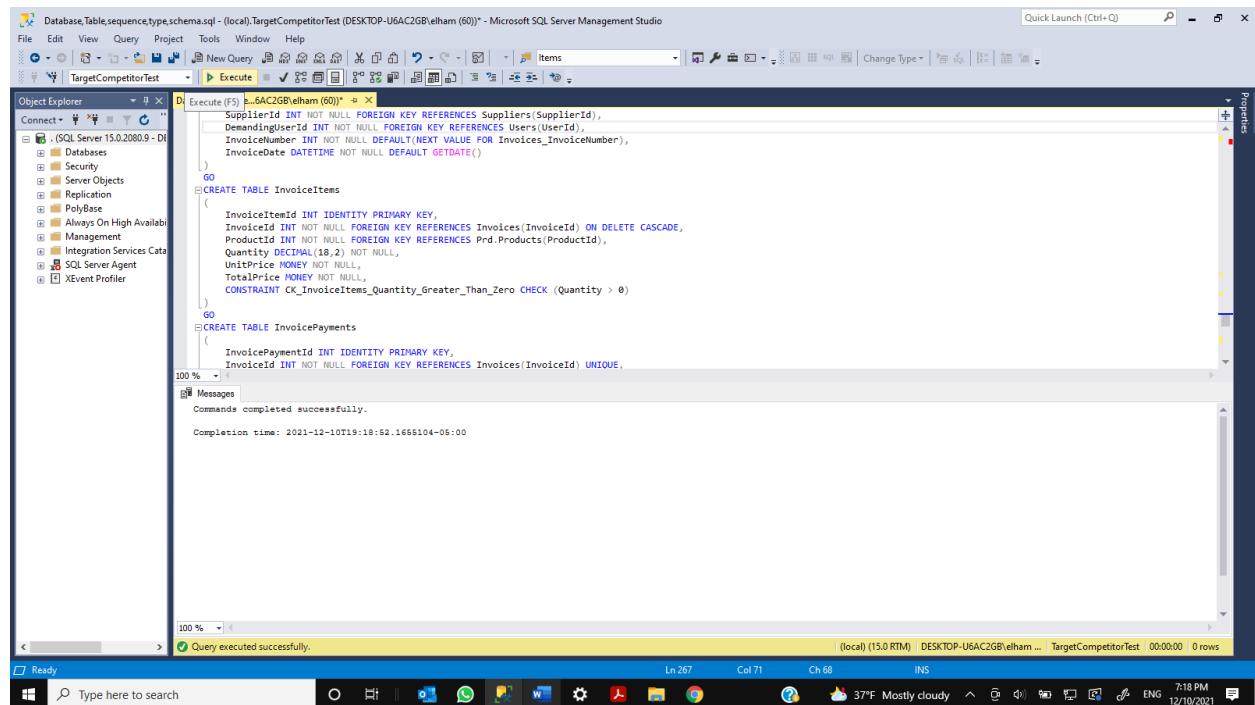
Comment: here we created the shipping table.

```
GO
CREATE SEQUENCE Invoices_InvoiceNumber
START WITH 1
INCREMENT BY 1
GO
--when stores place order for product supply , an invoice get created
--so we know which supplier supplied it.
CREATE TABLE Invoices
(
    InvoiceId INT IDENTITY PRIMARY KEY,
    SupplierId INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierId),
    DemandingUserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId),
    InvoiceNumber INT NOT NULL DEFAULT(NEXT VALUE FOR Invoices_InvoiceNumber),
    InvoiceDate DATETIME NOT NULL DEFAULT GETDATE()
)
```



Comment: Here we created the invoices table. We also created a sequence to create numbers for the times when stores place an order for product supply. (An invoice gets created to inform us about the supplier who supplied it).

```
GO  
CREATE TABLE InvoiceItems  
(  
    InvoiceItemId INT IDENTITY PRIMARY KEY,  
    InvoiceId INT NOT NULL FOREIGN KEY REFERENCES Invoices(InvoiceId) ON DELETE CASCADE,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    Quantity DECIMAL(18,2) NOT NULL,  
    UnitPrice MONEY NOT NULL,  
    TotalPrice MONEY NOT NULL,  
    CONSTRAINT CK_InvoiceItems_Quantity_Greater_Than_Zero CHECK (Quantity > 0)  
)
```



Comment: here we created the invoice items table.

GO**CREATE TABLE** InvoicePayments**(**

```
InvoicePaymentId INT IDENTITY PRIMARY KEY,  
InvoiceId INT NOT NULL FOREIGN KEY REFERENCES Invoices(InvoiceId) UNIQUE,  
Amount MONEY NOT NULL,  
PaymentDate DATETIME NOT NULL
```

)

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure, including the TargetCompetitorTest database and its objects like Databases, Security, and Server Objects. The main pane contains a query window with the following T-SQL code:

```
CREATE TABLE InvoicePayments  
(  
    InvoicePaymentId INT IDENTITY PRIMARY KEY,  
    InvoiceId INT NOT NULL FOREIGN KEY REFERENCES Invoices(InvoiceId) ON DELETE CASCADE,  
    ProductId INT NOT NULL FOREIGN KEY REFERENCES Prd.Products(ProductId),  
    Quantity DECIMAL(18,2) NOT NULL,  
    UnitPrice MONEY NOT NULL,  
    TotalPrice MONEY NOT NULL,  
    CONSTRAINT CK_InvoiceItems_Quantity_Greater_Than_Zero CHECK (Quantity > 0)  
)  
GO  
CREATE TYPE OrderProduct AS TABLE  
(  
    ProductId INT,  
    Quantity Decimal(18,2)
```

The status bar at the bottom indicates "Commands completed successfully." and "Completion time: 2021-12-10T19:20:27.5511974-06:00". The taskbar at the bottom of the screen shows various application icons.

Comment: here we created the invoice payment table.

```

GO
CREATE TYPE OrderProduct AS TABLE
(
    ProductId INT,
    Quantity Decimal(18,2)
)
GO
CREATE TYPE WarehouseProduct AS TABLE
(
    ProductId INT,
    Quantity Decimal(18,2),
    StoreId INT NULL
)
GO
CREATE SCHEMA Inv;
GO
ALTER SCHEMA Inv TRANSFER StoreProducts;
ALTER SCHEMA Inv TRANSFER WarehouseProducts;
ALTER SCHEMA Inv TRANSFER SupplierProducts;

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "Database,Table,sequence,type,schema.sql - (local)\TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio". The main window displays a query editor with the following SQL code:

```

USE [TargetCompetitorTest]
GO
CREATE TYPE OrderProduct AS TABLE
(
    ProductId INT,
    Quantity Decimal(18,2)
)
GO
CREATE TYPE WarehouseProduct AS TABLE
(
    ProductId INT,
    Quantity Decimal(18,2),
    StoreId INT NULL
)
GO
CREATE SCHEMA Inv;
GO
ALTER SCHEMA Inv TRANSFER StoreProducts;
ALTER SCHEMA Inv TRANSFER WarehouseProducts;
ALTER SCHEMA Inv TRANSFER SupplierProducts;

```

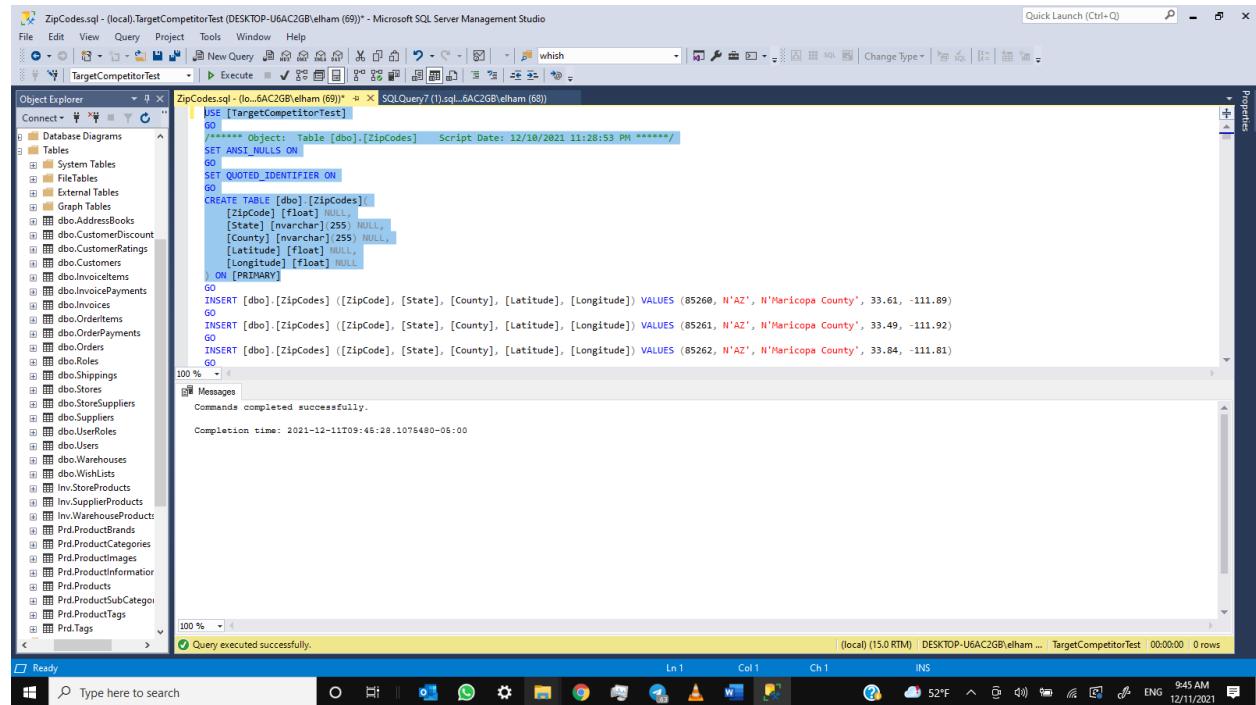
The code is being executed, as indicated by the "Execute" button in the toolbar. The status bar at the bottom right shows "Query executed successfully." and "Completion time: 2021-12-10T19:28:07.7294641-05:00". The system tray at the bottom indicates it's 7:28 PM, 37°F, Mostly cloudy, and the date is 12/10/2021.

Comment: here we created the types and schemas.

```

USE [TargetCompetitorTest]
GO
CREATE TABLE [dbo].[ZipCodes](
    [ZipCode] [float] NULL,
    [State] [nvarchar](255) NULL,
    [County] [nvarchar](255) NULL,
    [Latitude] [float] NULL,
    [Longitude] [float] NULL
) ON [PRIMARY]

```



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like tables, stored procedures, and triggers. The central pane displays the SQL script for creating the ZipCodes table. The status bar at the bottom indicates the query was executed successfully.

```

USE [TargetCompetitorTest]
GO
CREATE TABLE [dbo].[ZipCodes](
    [ZipCode] [float] NULL,
    [State] [nvarchar](255) NULL,
    [County] [nvarchar](255) NULL,
    [Latitude] [float] NULL,
    [Longitude] [float] NULL
) ON [PRIMARY]

```

Comment: here we created the zip codes table to insert all the zip codes and the longitude and latitude to be able to calculate the distances.

Transactions & Stored procedures:

1)

```
USE TargetCompetitorTest
GO
CREATE PROCEDURE CustomerReviews
@IN_CustomerId INT
AS
    SELECT p.Name ProductName , Score , Text , RatingDate
    FROM CustomerRatings cr
    JOIN prd.Products p ON cr.ProductId = p.ProductId
    JOIN Customers c ON cr.CustomerId = c.CustomerId
    WHERE cr.CustomerId = @IN_CustomerId
GO
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is a query window containing T-SQL code to create a stored procedure named 'CustomerReviews'. The code includes parameters for customer ID and joins between 'CustomerRatings', 'Products', and 'Customers' tables to select product names, scores, text, and rating dates. A 'GO' statement is present at the end of the procedure definition. The status bar at the bottom of the screen displays the message 'Query executed successfully.' along with other system information.

Comment: here we created a stored procedure to show reviews for a specific customer.

2)

GO

CREATE PROCEDURE ProductReviews

@IN_ProductId INT

AS

```
    SELECT c.Name CustomerName , Score , Text , RatingDate
    FROM CustomerRatings cr
    JOIN prd.Products p ON cr.ProductId = p.ProductId
    JOIN Customers c ON cr.CustomerId = c.CustomerId
    WHERE cr.ProductId = @IN_ProductId
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating a stored procedure:

```
CREATE PROCEDURE ProductReviews
@IN_ProductId INT
AS
    SELECT c.Name CustomerName , Score , Text , RatingDate
    FROM CustomerRatings cr
    JOIN prd.Products p ON cr.ProductId = p.ProductId
    JOIN Customers c ON cr.CustomerId = c.CustomerId
    WHERE cr.ProductId = @IN_ProductId
```

The 'Messages' pane at the bottom left shows the execution results:

```
Commands completed successfully.
```

The status bar at the bottom right indicates the session details: (local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows.

Comment: here we created a stored procedure to show product reviews.

3-4)

```

CREATE OR ALTER PROCEDURE PlaceCustomerOrder
    @CustomerId INT,
    @PaymentType NVARCHAR(10),
    @OrderType NVARCHAR(7),
    @ShippingService NVARCHAR(5),
    @ShippingAddressId INT NULL,
    @ShippingFare MONEY,
    @ExpectedShippingDate DATETIME,
    @PaymentAmount MONEY,
    @PaymentDate DATETIME,
    @Products OrderProduct READONLY
AS
BEGIN TRAN CustomerOrder
BEGIN TRY

    DECLARE @OrderId INT
    DECLARE @OrderStatus NVARCHAR(10) = IIF(@PaymentType = 'Not-Paid', 'Registered', 'Paid')

    IF EXISTS (
        SELECT * FROM @Products p1
        LEFT JOIN prd.Products p2 ON p1.ProductId = p2.ProductId
        WHERE p2.ProductId IS NULL
    )
        THROW 50001, 'Invalid products ids detected', 1;

    IF EXISTS (
        SELECT * FROM @Products p1
        LEFT JOIN prd.Products p2 ON p1.ProductId = p2.ProductId
        WHERE p1.Quantity = 0
    )
        THROW 50002, 'Products quantity cannot be zero', 1;

    IF @OrderType = 'Online' AND (@ShippingAddressId IS NULL OR @ShippingService IS NULL)
        THROW 50003, 'Online orders should have shipping address and service specified', 1;

    DECLARE @NotEnough NVARCHAR(MAX) = 'Not enough product inventory' + (SELECT
        STRING_AGG(inv, ', ') FROM (
            SELECT CAST(p.ProductId AS VARCHAR) + ' Inventory : ' + CAST(SUM(sp.Inventory) AS
        VARCHAR) inv
            FROM inv.StoreProducts sp
            JOIN @Products p ON sp.ProductId = p.ProductId
            GROUP BY p.ProductId, p.Quantity
            HAVING SUM(sp.Inventory) < p.Quantity
        )a
    )

    IF @NotEnough IS NOT NULL
        THROW 50004, @NotEnough, 1;

    INSERT Orders
    VALUES(@OrderStatus, @OrderType, @PaymentType, @CustomerId, GETDATE())

    SET @OrderId = SCOPE_IDENTITY()

    INSERT OrderItems
    SELECT      @OrderId,
                p2.ProductId,

```

```

        p1.Quantity,
        IIF(cd.DiscountPercent IS NOT NULL,
            p2.Price - p2.Price * cd.DiscountPercent/100 ,
            p2.Price),
        IIF(cd.DiscountPercent IS NOT NULL,
            p2.Price - p2.Price * cd.DiscountPercent/100 ,
            p2.Price) * p1.Quantity
    FROM @Products p1
    JOIN prd.Products p2 ON p1.ProductId = p2.ProductId
    LEFT JOIN CustomerDiscounts cd ON p2.ProductId = cd.ProductId AND cd.CustomerId =
    @CustomerId AND cd.ExpireDate < GETDATE()
    INSERT OrderPayments
    VALUES (@OrderId , @PaymentAmount , @PaymentDate)

    IF @OrderType = 'Online'
    BEGIN
        INSERT Shippings
        VALUES (@OrderId , @ShippingService , @ShippingAddressId , @ShippingFare,
    @ExpectedShippingDate ,NULL)
    END

    ;WITH StoreInventoryUpdate
    AS
    (
        SELECT
            p.ProductId,
            sp.StoreId,
            sp.StoreProductId,
            dbo.DistanceOfTwoPoint(sz.Latitude , sz.Longitude , az.Latitude , az.Longitude)
Distance,
            p.Quantity CustomerRequestedQuantity ,
            sp.Inventory,
            a.AddressBookId,
            ISNULL(p.Quantity - SUM(sp.Inventory)
            OVER(
                PARTITION BY P.ProductId
                ORDER BY dbo.DistanceOfTwoPoint(sz.Latitude ,
sz.Longitude , az.Latitude , az.Longitude)
                ROWS BETWEEN UNBOUNDED PRECEDING AND 1
PRECEDING) , IIF(p.Quantity > sp.Inventory , sp.Inventory , p.Quantity)) RemainedRequestedInventory
        FROM @Products p
        JOIN inv.StoreProducts sp ON p.ProductId = sp.ProductId
        JOIN Stores s ON sp.StoreId = s.StoreId
        JOIN ZipCodes sz ON s.ZipCode = sz.ZipCode
        JOIN AddressBooks a ON 1 = 1
        JOIN ZipCodes az ON az.ZipCode = a.ZipCode
        WHERE a.AddressBookId = @ShippingAddressId
    )
    UPDATE StoreInventoryUpdate
    SET
        Inventory = IIF(CustomerRequestedQuantity < Inventory ,
                    Inventory - CustomerRequestedQuantity ,
                    IIF(RemainedRequestedInventory >= Inventory , 0 ,
        Inventory - RemainedRequestedInventory))
    WHERE RemainedRequestedInventory >= 0
    DELETE inv.StoreProducts

```

```

WHERE Inventory = 0

    COMMIT TRAN CustomerOrder
END TRY
BEGIN CATCH
    ROLLBACK TRAN CustomerOrder
    PRINT 'Error occurred'
    PRINT 'ErrorNumber:' + CAST(ERROR_NUMBER() AS NVARCHAR(MAX))
    PRINT 'Message : ' + ERROR_MESSAGE()
END CATCH

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with a tree view of the database structure. The center pane shows the script for the 'PlaceCustomerOrder' stored procedure. The right pane shows the Properties window. At the bottom, the Messages pane displays the output: 'Commands completed successfully.' and 'Completion time: 2021-12-10T21:39:45.9419640-05:00'. The status bar at the bottom right shows the system information: (local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows.

Comments: this procedure/transaction performs the following tasks

- ✓ Detection of various errors like invalid product id, products with a zero quantity, orders with no shipping address, and shipping services.
- ✓ Inserting orders, the related dates, payments, and the shipping information
- ✓ Calculation of the price and the applied discounts if there are any.
- ✓ Updating the inventory of stores (by using a CTE)
- ✓ Deleting the products which have no inventory value.

5-6)

```

USE TargetCompetitorTest
GO
CREATE OR ALTER PROCEDURE DeliverProductToStoreFromWarehouse
@PaymentType NVARCHAR(10),
@PaymentAmount MONEY,
@PaymentDate DATETIME,
@SupplierId INT,
@DemandingUserId INT,
@Products WarehouseProduct READONLY
AS
BEGIN TRAN
BEGIN TRY
    DECLARE @InvoiceId INT

    IF EXISTS (
        SELECT * FROM @Products p1
        LEFT JOIN Products p2 ON p1.ProductId = p2.ProductId
        WHERE p2.ProductId IS NULL
    )
        THROW 50001, 'Invalid products ids detected', 1;

    IF EXISTS (
        SELECT * FROM @Products p1
        LEFT JOIN Products p2 ON p1.ProductId = p2.ProductId
        WHERE p1.Quantity = 0
    )
        THROW 50002, 'Products quantity cannot be zero', 1;

    DECLARE @NotEnough NVARCHAR(MAX) = 'Not enough product inventory' + (SELECT
        STRING_AGG(inv, ', ') FROM (
            SELECT CAST(p.ProductId AS VARCHAR) + ' Inventory : ' + CAST(SUM(wp.Inventory) AS
                VARCHAR) inv
            FROM WarehouseProducts wp
            JOIN @Products p ON wp.ProductId = p.ProductId
            JOIN Warehouses w ON wp.WarehouseId = w.WarehouseId
            WHERE w.SupplierId = @SupplierId
            GROUP BY p.ProductId, p.Quantity
            HAVING SUM(wp.Inventory) < p.Quantity
        )a
    )

    IF @NotEnough IS NOT NULL
        THROW 50004, @NotEnough, 1;

    INSERT Invoices (SupplierId, DemandingUserId, InvoiceDate)
    VALUES (@SupplierId, @DemandingUserId, GETDATE())

    SET @InvoiceId = SCOPE_IDENTITY()

    INSERT InvoiceItems
    SELECT      @InvoiceId,
                p1.ProductId,
                p1.Quantity,
                p2.Price,
                p1.Quantity * p2.Price
    FROM @Products p1

```

```

JOIN Products p2 ON p1.ProductId = p2.ProductId

INSERT InvoicePayments
VALUES(@InvoiceId, @PaymentAmount, @PaymentDate)

MERGE INTO StoreProducts sp
USING(
    SELECT ip1.ProductId, ip1.Quantity, ip1.StoreId, ip2.Price
    FROM @Products ip1
    JOIN Products ip2 ON ip1.ProductId = ip2.ProductId
) p (ProductId, Quantity, StoreId, Price)
ON sp.ProductId = p.ProductId AND sp.StoreId = p.StoreId
WHEN MATCHED THEN
    UPDATE SET sp.Inventory = sp.Inventory + p.Quantity
WHEN NOT MATCHED THEN
    INSERT(StoreId, ProductId, Price, Inventory)
    VALUES(p.StoreId, p.ProductId, p.Price, p.Quantity);

;WITH WarehouseInventories
AS(
    SELECT      p.ProductId,
                s.StoreId,
                sp.StoreProductId,
                dbo.DistanceOfTwoPoint(sz.Latitude, sz.Longitude, wz.Latitude,
wz.Longitude) Distance,
                p.Quantity CustomerRequestedQuantity,
                wp.Inventory WarehouseInventory,
                wp.OrderedInventory,
                ISNULL(p.Quantity - SUM(wp.Inventory)
OVER(
            PARTITION BY P.ProductId
            ORDER BY dbo.DistanceOfTwoPoint(sz.Latitude,
sz.Longitude, wz.Latitude, wz.Longitude)
            ROWS BETWEEN UNBOUNDED PRECEDING
AND 1 PRECEDING), IIF(p.Quantity > wp.Inventory, wp.Inventory, p.Quantity))
RemainedRequestedInventory
        FROM @Products p
        JOIN WarehouseProducts wp ON p.ProductId = wp.ProductId AND wp.Inventory != 0
        JOIN Stores s ON p.StoreId = s.StoreId
        JOIN StoreSuppliers ss ON s.StoreId = ss.StoreId
        JOIN Warehouses w ON wp.WarehouseId = w.WarehouseId AND w.SupplierId =
ss.SupplierId
        LEFT JOIN StoreProducts sp ON sp.ProductId = wp.ProductId AND sp.StoreId = p.StoreId
        JOIN ZipCodes wz ON w.ZipCode = wz.ZipCode
        JOIN ZipCodes sz ON sz.ZipCode = s.ZipCode
)
UPDATE WarehouseInventories
SET OrderedInventory = OrderedInventory + RemainedRequestedInventory,
WarehouseInventory = IIF(CustomerRequestedQuantity < WarehouseInventory,
                           WarehouseInventory - CustomerRequestedQuantity,
                           IIF(RemainedRequestedInventory >= WarehouseInventory, 0, WarehouseInventory -
RemainedRequestedInventory))
WHERE RemainedRequestedInventory > 0

COMMIT TRAN
END TRY
BEGIN CATCH

```

```

ROLLBACK TRAN
PRINT 'Error occurred'
PRINT 'ErrorNumber:' + CAST(ERROR_NUMBER() AS NVARCHAR(MAX))
PRINT 'Message : ' + ERROR_MESSAGE()
END CATCH

```

```

USE [StoreProductSupply_AAC2GB\elham (67)]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[StoreProductSupply_AAC2GB\elham (67)]
    @CustomerCode NVARCHAR(50),
    @WarehouseCode NVARCHAR(50)
AS
BEGIN
    BEGIN TRY
        UPDATE WarehouseInventories
        SET OrderedInventory = OrderedInventory - RemainedRequestedInventory,
            WarehouseInventory = ISNULL(CustomerRequestedQuantity - WarehouseInventory,
                ISNULL(RemainedRequestedInventory - WarehouseInventory, 0)) - WarehouseInventory
            WHERE RemainedRequestedInventory > 0
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred'
        PRINT 'ErrorNumber:' + CAST(ERROR_NUMBER() AS NVARCHAR(MAX))
        PRINT 'Message : ' + ERROR_MESSAGE()
    END CATCH
END

```

Completion time: 2021-12-10T22:56:09.5170094-06:00

Query executed successfully.

Comment: This transaction/procedure performs the following tasks.

- ✓ Inserting data into invoices like supplier information, user information and invoice date.
- ✓ Updating store product if there is existing inventory.
- ✓ Inserting store product if there is no existing inventory
- ✓ Reducing inventory of warehouse which supplied the product.

7)

```
BEGIN TRY
BEGIN TRAN NewProductAssignmentToWarehouse

DECLARE @ProductBrandId INT = 3
DECLARE @ProductSubCategoryId INT = 40
DECLARE @Name NVARCHAR(100)='Cat food'
DECLARE @Description NVARCHAR(250)=NULL
DECLARE @Price MONEY = 50
DECLARE @WarehouseId INT = 1
DECLARE @Inventory INT = 10
DECLARE @ProductId INT

IF NOT EXISTS (SELECT 1 FROM TargetCompetitorTest.Prd.ProductBrands WHERE
ProductBrandId = @ProductBrandId)
    THROW 50010 , 'Product brand does not exists' , 1;

IF NOT EXISTS (SELECT 1 FROM TargetCompetitorTest.Prd.ProductSubCategories WHERE
ProductSubCategoryId = @ProductSubCategoryId)
    THROW 50011 , 'Product sub category does not exists' , 1;

IF NOT EXISTS (SELECT 1 FROM TargetCompetitorTest.dbo.Warehouses WHERE WarehouseId =
@WarehouseId)
    THROW 50012 , 'Warehouse does not exists' , 1;

INSERT TargetCompetitorTest.Prd.Products
VALUES(@Name , @Description , @Price , @ProductBrandId , @ProductSubCategoryId)

SET @ProductId = SCOPE_IDENTITY()

INSERT TargetCompetitorTest.Inv.WarehouseProducts
VALUES(@WarehouseId , @ProductId , @Inventory , 0 ,0)

COMMIT TRAN NewProductAssignmentToWarehouse
END TRY
BEGIN CATCH
    ROLLBACK TRAN NewProductAssignmentToWarehouse

    PRINT 'Error Occured : '
    PRINT 'Error Number:' + CAST(ERROR_NUMBER() AS NVARCHAR(MAX))
    PRINT 'Error Message : ' + ERROR_MESSAGE()
END CATCH
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the database structure of 'TargetCompetitorTest'. The right pane contains a query window titled 'SQLQuery1.sql - AHMAD-F15.TargetCompetitorTest (AHMAD-F15\ahmads (62))'.

```
IF NOT EXISTS (SELECT 1 FROM TargetCompetitorTest.dbo.Warehouses WHERE WarehouseId = @WarehouseId)
    THROW 500012 , 'Warehouse does not exists' , 1 ;

INSERT TargetCompetitorTest.Prd.Products
VALUES (@Name , @Description , @Price , @ProductBrandId , @ProductSubCategoryId);

SET @ProductId = SCOPE_IDENTITY();

INSERT TargetCompetitorTest.Inv.WarehouseProducts
VALUES (@WarehouseId , @ProductId , @Inventory , 0 , 0);

COMMIT TRAN NewProductAssignmentToWarehouse
END TRY
BEGIN CATCH
    ROLLBACK TRAN NewProductAssignmentToWarehouse

    PRINT 'Error Ocurred : '
    PRINT 'Error Number : ' + CAST(ERROR_NUMBER() AS NVARCHAR(MAX))
    PRINT 'Error Message : ' + ERROR_MESSAGE()
END CATCH
```

The 'Messages' pane at the bottom shows the execution results:

```
(1 row affected)
(1 row affected)
```

Completion time: 2021-12-14T16:57:20.0445998+03:30

At the bottom of the window, it says 'Query executed successfully.'

Comment: here we created a transaction that adds a new product to warehouse.

8)

```
BEGIN TRY
BEGIN TRAN NewTagAssignmentToProduct

    DECLARE @ProductId INT = 104
    DECLARE @Name NVARCHAR(100) = 'Tools'
    DECLARE @Description NVARCHAR(MAX) = 'Powerful tools'

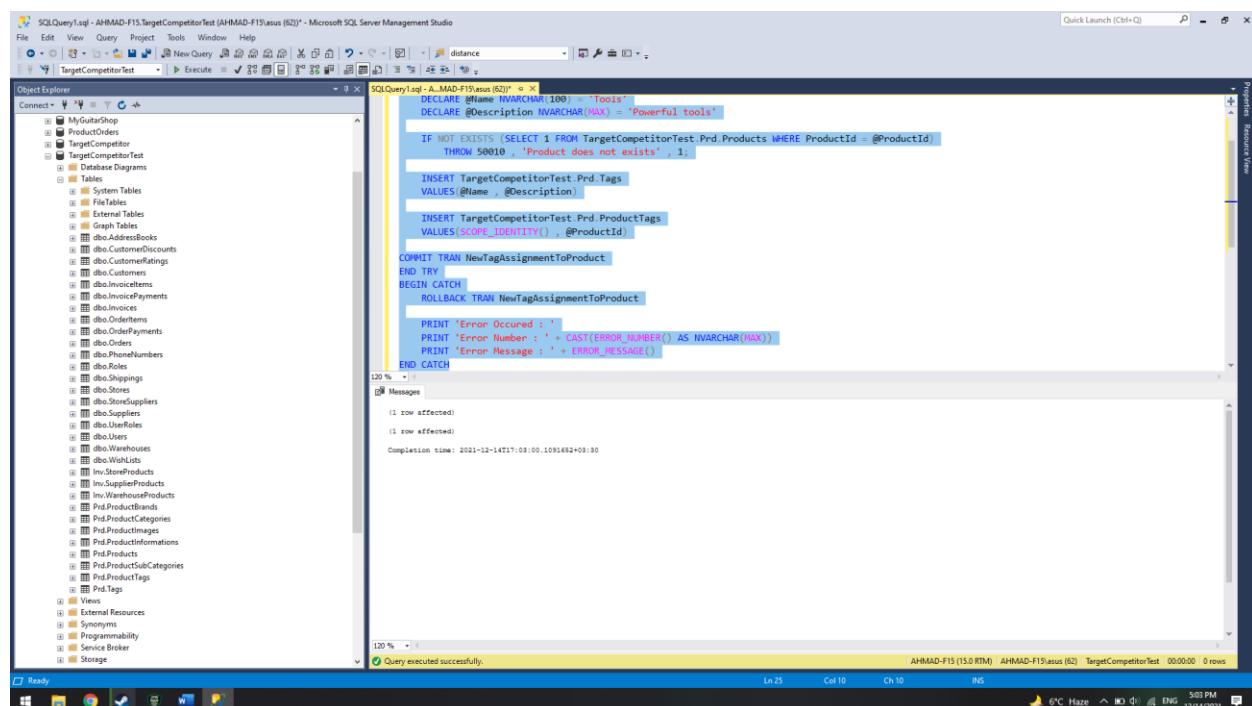
    IF NOT EXISTS (SELECT 1 FROM TargetCompetitorTest.Prd.Products WHERE ProductId = @ProductId)
        THROW 50010 , 'Product does not exists' , 1;

    INSERT TargetCompetitorTest.Prd.Tags
    VALUES(@Name , @Description)

    INSERT TargetCompetitorTest.Prd.ProductTags
    VALUES(SCOPE_IDENTITY() , @ProductId)

COMMIT TRAN NewTagAssignmentToProduct
END TRY
BEGIN CATCH
    ROLLBACK TRAN NewTagAssignmentToProduct

    PRINT 'Error Occured : '
    PRINT 'Error Number : ' + CAST(ERROR_NUMBER() AS NVARCHAR(MAX))
    PRINT 'Error Message : ' + ERROR_MESSAGE()
END CATCH
```



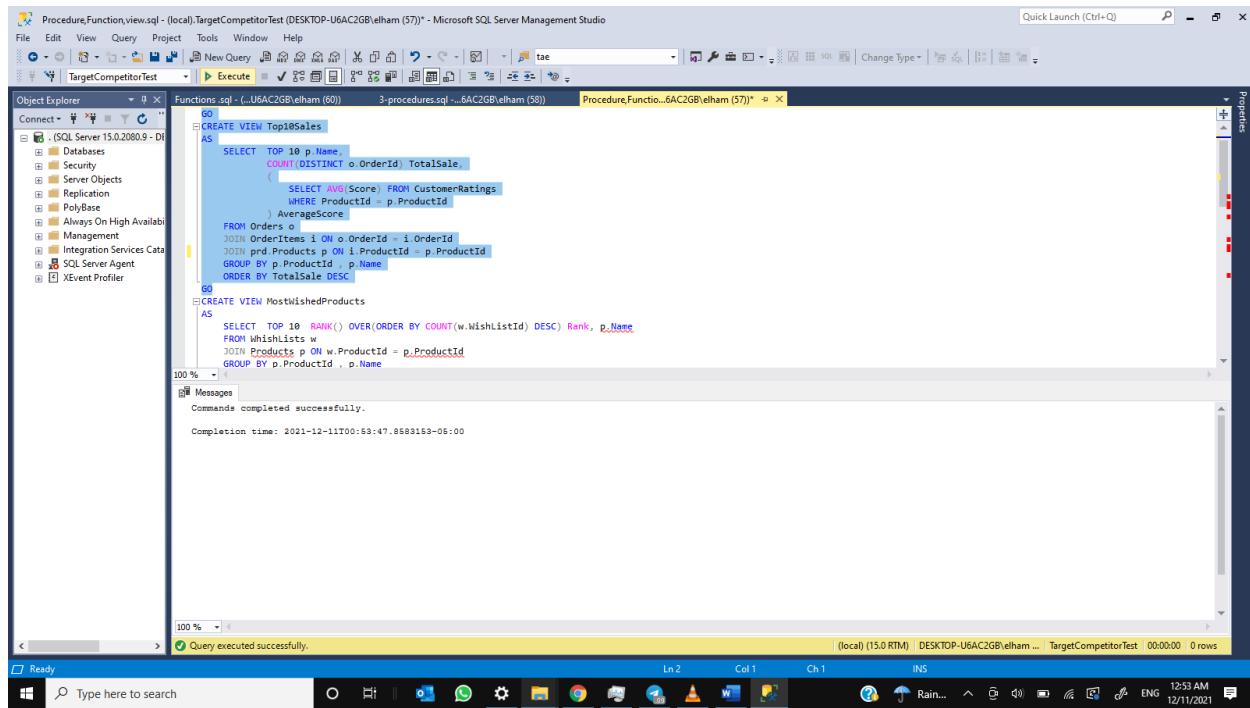
Comment: This transaction assign new tags to products.

Views :

```

USE TargetCompetitorTest
GO
CREATE VIEW Top10Sales
AS
    SELECT      TOP 10 p.Name,
                COUNT(DISTINCT o.OrderId) TotalSale,
                (
                    SELECT AVG(Score) FROM CustomerRatings
                    WHERE ProductId = p.ProductId
                ) AverageScore
    FROM Orders o
    JOIN OrderItems i ON o.OrderId = i.OrderId
    JOIN prd.Products p ON i.ProductId = p.ProductId
    GROUP BY p.ProductId, p.Name
    ORDER BY TotalSale DESC

```



Comment: This view shows top 10 product regarding their total sales.

```
GO  
USE TargetCompetitorTest  
GO  
CREATE VIEW MostWishedProducts  
AS  
SELECT      TOP 10 RANK() OVER(ORDER BY COUNT(w.WishListId) DESC) Rank, p.Name  
FROM WishLists w  
JOIN prd.Products p ON w.ProductId = p.ProductId  
GROUP BY p.ProductId , p.Name  
ORDER BY COUNT(w.WishListId) DESC
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (54)) - Microsoft SQL Server Management Studio' contains the T-SQL code for creating the 'MostWishedProducts' view. The code includes a temporary table '#TempScore', a SELECT statement from 'Orders' and 'OrderItems', and the final CREATE VIEW statement. The 'Messages' pane at the bottom indicates that the commands were executed successfully. The system tray at the bottom right shows the date and time as 12/11/2021 1:10 AM.

```
File Edit View Query Project Tools Window Help  
... New Query ... whish ... Change Type ...  
... TargetCompetitorTest ... Execute ...  
Object Explorer Properties  
File Connect ... Help  
... Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler  
... (SQL Server 15.0.2080.9 - DE...  
... #TempScore  
    WHERE ProductId = p.ProductId  
    FROM Orders o  
    JOIN OrderItems i ON o.OrderId = i.OrderId  
    JOIN prd.Products p ON i.ProductId = p.ProductId  
    GROUP BY p.ProductId , p.Name  
    ORDER BY TotalSale DESC  
GO  
USE TargetCompetitorTest  
GO  
CREATE VIEW MostWishedProducts  
AS  
SELECT      TOP 10 RANK() OVER(ORDER BY COUNT(w.WishListId) DESC) Rank, p.Name  
FROM WishLists w  
JOIN prd.Products p ON w.ProductId = p.ProductId  
GROUP BY p.ProductId , p.Name  
ORDER BY COUNT(w.WishListId) DESC  
100 % 100 %  
Messages Commands completed successfully.  
Completion time: 2021-12-11T01:10:32.1457794-06:00  
100 % 100 %  
Ready (local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows  
Ln 16 Col 1 Ch 1 INS Rain... 1:10 AM ENG 12/11/2021
```

Comment: This view shows the most wished products.

```
USE TargetCompetitorTest
GO
CREATE VIEW CustomersWithNearbyStores
AS
    SELECT c.Name ,  dbo.CustomerNearbyStores(c.CustomerId , 2000) NearbyStores
    FROM Customers c
```

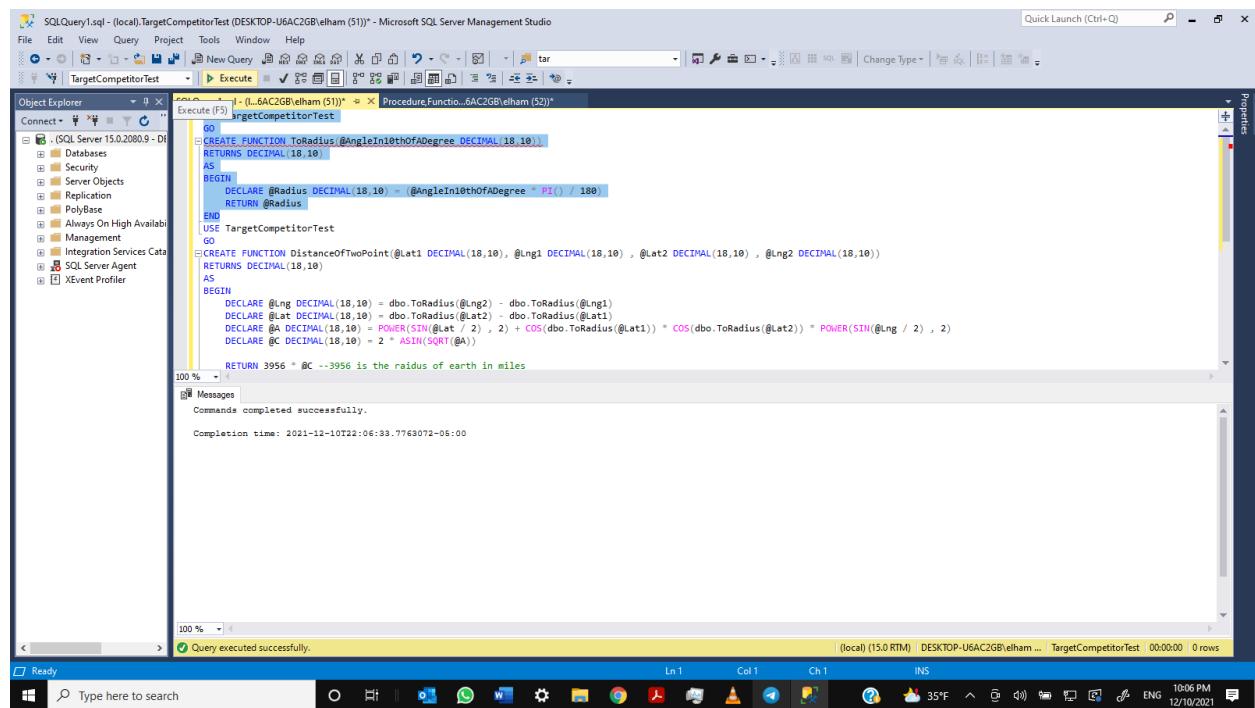
The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'Views.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (51)) - Microsoft SQL Server Management Studio' contains the T-SQL code for creating a view. The code includes a USE statement for the database, a GO statement, a CREATE VIEW statement with a view name, an AS keyword, and a SELECT statement that joins the 'Customers' table with a user-defined function 'dbo.CustomerNearbyStores'. The code is highlighted in blue. Below the code, the 'Messages' pane shows the output: 'Commands completed successfully.' and 'Completion time: 2021-12-11T08:33:03.0762302-05:00'. At the bottom of the screen, the Windows taskbar is visible with various icons.

Comment: this view use CustomerNearbyStores to show the stores near the customers

Functions :

1)

```
USE TargetCompetitorTest
GO
CREATE FUNCTION ToRadius(@AngleIn10thOfADegree DECIMAL(18,10))
RETURNS DECIMAL(18,10)
AS
BEGIN
    DECLARE @Radius DECIMAL(18,10)=(@AngleIn10thOfADegree * PI() / 180)
    RETURN @Radius
END
```



Comment: the output of this function will be used to calculate the distance between two points on earth.

2)

```

USE TargetCompetitorTest
GO
CREATE FUNCTION DistanceOfTwoPoint(@Lat1 DECIMAL(18,10), @Lng1 DECIMAL(18,10) , @Lat2
DECIMAL(18,10) , @Lng2 DECIMAL(18,10))
RETURNS DECIMAL(18,10)
AS
BEGIN
    DECLARE @Lng DECIMAL(18,10)=dbo.ToRadius(@Lng2)-dbo.ToRadius(@Lng1)
    DECLARE @Lat DECIMAL(18,10)=dbo.ToRadius(@Lat2)-dbo.ToRadius(@Lat1)
    DECLARE @A DECIMAL(18,10)=POWER(SIN(@Lat / 2) , 2)+COS(dbo.ToRadius(@Lat1))*
COS(dbo.ToRadius(@Lat2))*POWER(SIN(@Lat / 2) , 2)
    DECLARE @C DECIMAL(18,10)=2 * ASIN(SQRT(@A))

    RETURN 3956 * @C --3956 is the radius of earth in miles
END

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'TargetCompetitorTest' is selected. In the center pane, a query window displays the T-SQL code for creating two functions. The first function, 'ToRadius', converts an angle in degrees to radians. The second function, 'DistanceOfTwoPoint', calculates the distance between two points on Earth given their latitude and longitude. The code uses several mathematical functions like PI(), SIN(), COS(), POWER(), and ASIN() to perform the calculations.

```

SQLQuery1.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (51)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Quick Launch (Ctrl+Q) P X
Object Explorer
Connect Connect to Server
SQL Server 15.0.2080.9 - DESKTOP-U6AC2GB\elham (51)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
TargetCompetitorTest
Execute (F5) q1-(L-6AC2GB\elham (51)) Procedure.Functio...6AC2GB\elham (52)*
--CREATE FUNCTION ToRadius(@AngleInDegree DECIMAL(18,10))
--AS
--BEGIN
--    DECLARE @Radius DECIMAL(18,10) = (@AngleInDegree * PI() / 180)
--    RETURN @Radius
--END
USE TargetCompetitorTest
GO
CREATE FUNCTION DistanceOfTwoPoint(@Lat1 DECIMAL(18,10) , @Lng1 DECIMAL(18,10) , @Lat2 DECIMAL(18,10) , @Lng2 DECIMAL(18,10))
RETURNS DECIMAL(18,10)
AS
BEGIN
    DECLARE @Lng DECIMAL(18,10)=dbo.ToRadius(@Lng2)-dbo.ToRadius(@Lng1)
    DECLARE @Lat DECIMAL(18,10)=dbo.ToRadius(@Lat2)-dbo.ToRadius(@Lat1)
    DECLARE @A DECIMAL(18,10)=POWER(SIN(@Lat / 2) , 2)+COS(dbo.ToRadius(@Lat1)) * COS(dbo.ToRadius(@Lat2)) * POWER(SIN(@Lat / 2) , 2)
    DECLARE @C DECIMAL(18,10)=2 * ASIN(SQRT(@A))

    RETURN 3956 * @C --3956 is the radius of earth in miles
END

```

Messages

Commands completed successfully.

Completion time: 2021-12-10T22:08:10.3044921-06:00

LN 10

SQLQuery1.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (51)) - Microsoft SQL Server Management Studio

Ready

Type here to search

100 %

Query executed successfully.

LN 10

SQLQuery1.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (51)) - Microsoft SQL Server Management Studio

LN 10

10:08 PM 12/10/2021

Comment: This function calculates the distance of two points which will be useful to find the nearest store to ship from.

3)

```
USE TargetCompetitorTest
GO
CREATE FUNCTION AverageProductScore(@ProductId INT)
RETURNS DECIMAL(4,2)
BEGIN
    DECLARE @AVG DECIMAL(4,2)

    SET @AVG = (
        SELECT AVG(Score) FROM CustomerRatings
        WHERE ProductId = @ProductId
    )

    RETURN IIF(@AVG IS NULL, 0, @AVG)
END
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the creation of a function named AverageProductScore. The command is as follows:

```
USE TargetCompetitorTest
GO
CREATE FUNCTION AverageProductScore(@ProductId INT)
RETURNS DECIMAL(4,2)
BEGIN
    DECLARE @AVG DECIMAL(4,2)

    SET @AVG = (
        SELECT AVG(Score) FROM CustomerRatings
        WHERE ProductId = @ProductId
    )

    RETURN IIF(@AVG IS NULL, 0, @AVG)
END
```

The execution results show the command completed successfully with a completion time of 2021-12-11T00:42:49.2268827-05:00.

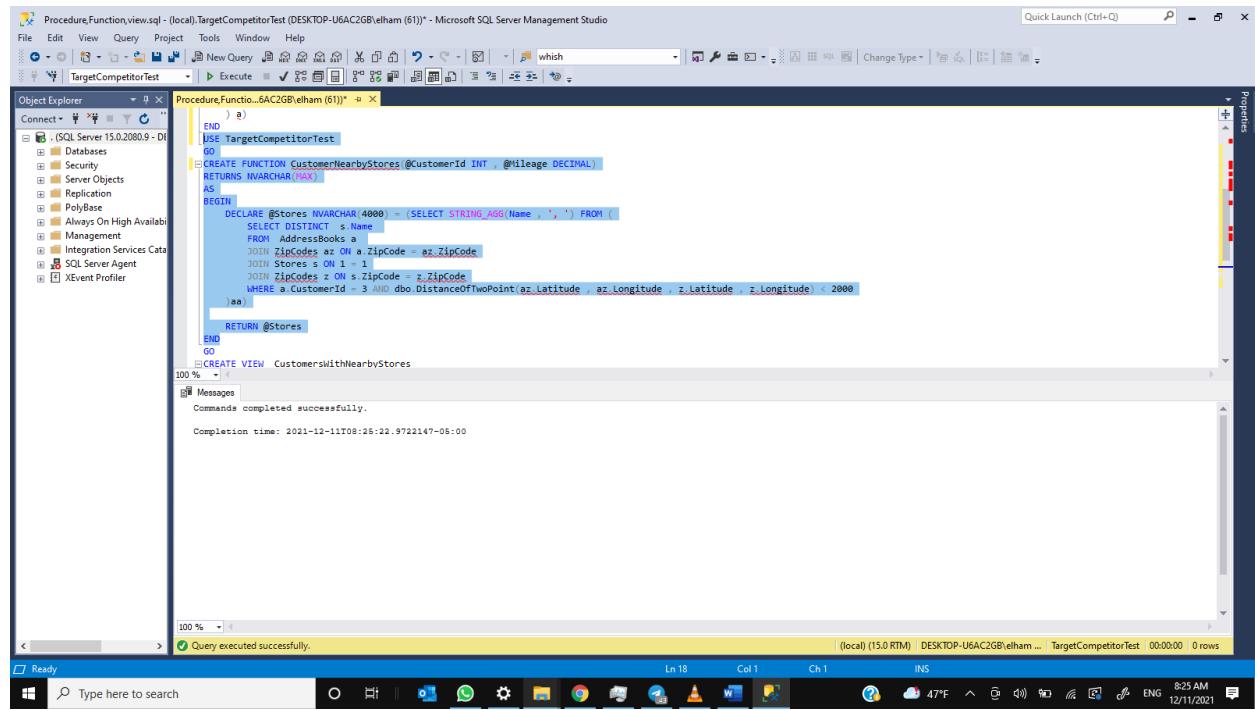
Comment: This function calculates the average product scores.

4)

```

USE TargetCompetitorTest
GO
CREATE FUNCTION CustomerNearbyStores(@CustomerId INT, @Mileage DECIMAL)
RETURNS NVARCHAR(MAX)
AS
BEGIN
    DECLARE @Stores NVARCHAR(4000) = (SELECT STRING_AGG(Name, ',') FROM (
        SELECT DISTINCT s.Name
        FROM AddressBooks a
        JOIN ZipCodes az ON a.ZipCode = az.ZipCode
        JOIN Stores s ON 1 = 1
        JOIN ZipCodes z ON s.ZipCode = z.ZipCode
        WHERE a.CustomerId = 3 AND dbo.DistanceOfTwoPoint(az.Latitude, az.Longitude,
        z.Latitude, z.Longitude) < 2000
    )aa)
    RETURN @Stores
END

```



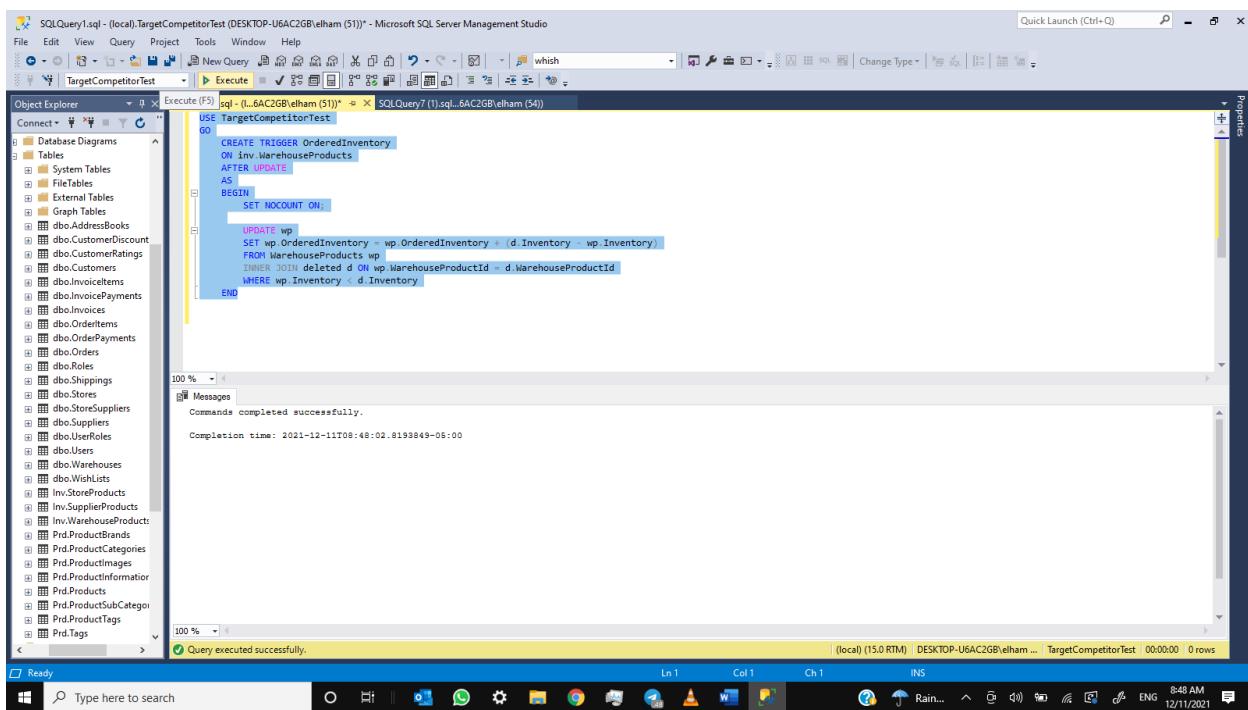
Comment: output of this function will be used in one of the views to show stores near customers.

Triggers:

1)

```
USE TargetCompetitorTest
GO
CREATE TRIGGER OrderedInventory
ON inv.WarehouseProducts
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE wp
    SET wp.OrderedInventory = wp.OrderedInventory + (d.Inventory - wp.Inventory)
    FROM WarehouseProducts wp
    INNER JOIN deleted d ON wp.WarehouseProductId = d.WarehouseProductId
    WHERE wp.Inventory < d.Inventory
END
```



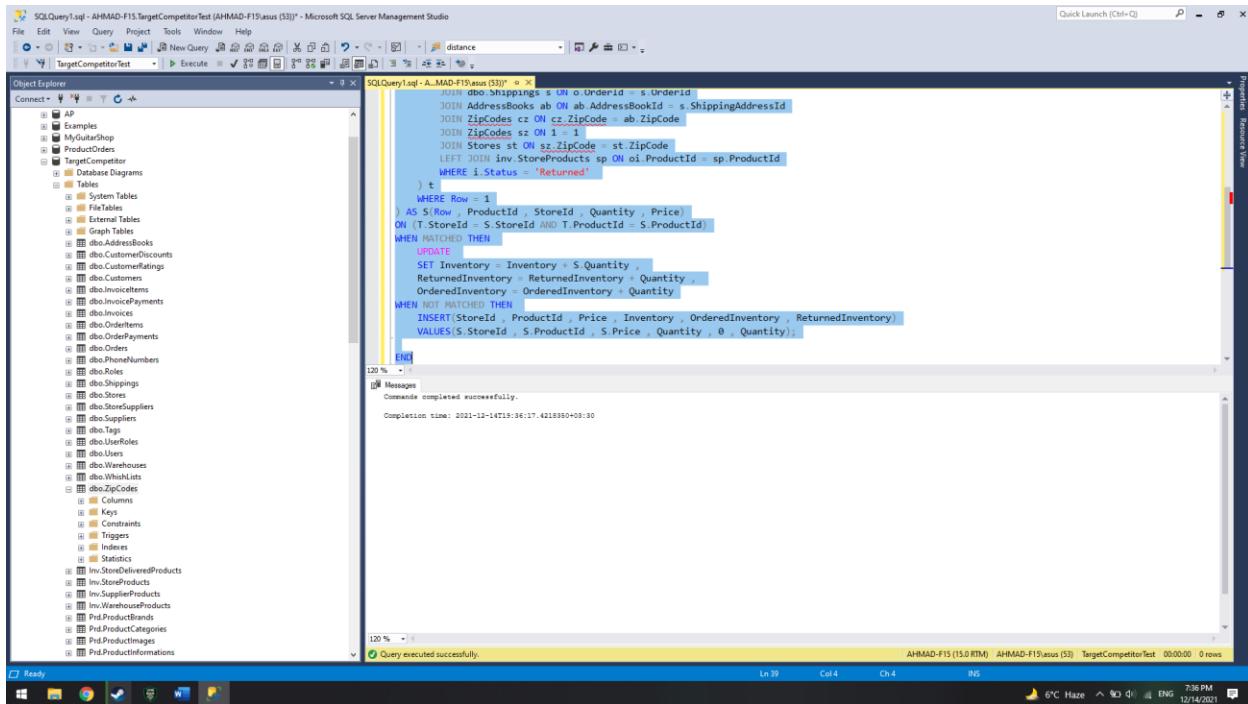
Comment: This trigger updates the warehouse product table column named ordered inventory. This will be useful for the times one of the products them has been ordered

2)

```
USE TargetCompetitorTest
GO
CREATE TRIGGER ReturnedOrderInventoryFix
ON TargetCompetitorTest.dbo.Orders
AFTER UPDATE
AS
BEGIN

MERGE inv.StoreProducts AS T
USING (
    SELECT Row ,ProductId,StoreId , Quantity , UnitPrice FROM (
        SELECT ROW_NUMBER() OVER(
            PARTITION BY oi.ProductId
            ORDER BY dbo.DistanceOfTwoPoint(cz.Latitude , cz.Longitude , sz.Latitude ,
sz.Longitude)
        ) Row , oi.ProductId , st.StoreId , oi.Quantity , oi.UnitPrice
        FROM dbo.Orders o
        JOIN Inserted i ON o.OrderId = i.OrderId
        JOIN dbo.OrderItems oi ON o.OrderId = oi.OrderId
        JOIN dbo.Shippings s ON o.OrderId = s.OrderId
        JOIN AddressBooks ab ON ab.AddressBookId = s.ShippingAddressId
        JOIN ZipCodes cz ON cz.ZipCode = ab.ZipCode
        JOIN ZipCodes sz ON 1 = 1
        JOIN Stores st ON sz.ZipCode = st.ZipCode
        LEFT JOIN inv.StoreProducts sp ON oi.ProductId = sp.ProductId
        WHERE i.Status = 'Returned'
    ) t
    WHERE Row = 1
) AS S(Row , ProductId , StoreId , Quantity , Price)
ON (T.StoreId = S.StoreId AND T.ProductId = S.ProductId)
WHEN MATCHED THEN
    UPDATE
        SET Inventory = Inventory + S.Quantity ,
        ReturnedInventory = ReturnedInventory + Quantity ,
        OrderedInventory = OrderedInventory + Quantity
WHEN NOT MATCHED THEN
    INSERT(StoreId , ProductId , Price , Inventory , OrderedInventory , ReturnedInventory)
    VALUES(S.StoreId , S.ProductId , S.Price , Quantity , 0 , Quantity);

END
```



The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The left pane displays the Object Explorer with the database 'TargetCompetitorTest' selected. The right pane contains a query window titled 'SQLQuery1.sql - AHMAD-F15.TargetCompetitorTest (AHMAD-F15\ahmed (S3))' containing the following T-SQL code:

```

CREATE TRIGGER [dbo].[Trg_InventoryUpdate]
ON [dbo].[Shipments]
FOR RETURNED
AS
BEGIN
    UPDATE [dbo].[Inventory]
    SET [Quantity] = [Quantity] + S.[Quantity]
    FROM [dbo].[Shipments] S
    JOIN [dbo].[AddressBooks] ab ON ab.AddressBookId = s.ShippingAddressId
    JOIN [dbo].[ZipCodes] cz ON cz.ZipCode = ab.ZipCode
    JOIN [dbo].[ZipCodes] sz ON sz.ZipCode = st.ZipCode
    LEFT JOIN [dbo].[StoreProducts] sp ON oi.ProductId = sp.ProductId
    WHERE i.Status = 'Returned'
    ) t
    WHERE Row = 1
) AS S(Row , ProductId , StoreId , Quantity , Price)
ON (T.StoreId = S.StoreId AND T.ProductId = S.ProductId)
WHEN MATCHED THEN
    UPDATE
        SET Inventory = S.Quantity
        ReturnedInventory = ReturnedInventory + Quantity ,
        OrderedInventory = OrderedInventory + Quantity
WHEN NOT MATCHED THEN
    INSERT(StoreId , ProductId , Price , Inventory , OrderedInventory , ReturnedInventory)
    VALUES(S.StoreId , S.ProductId , S.Price , Quantity , 0 , Quantity);
END

```

The status bar at the bottom indicates 'Query executed successfully.' and 'Completion time: 2021-12-14T19:36:17.4218890+03:00'.

Comment: Here we created a trigger that updates the inventory when a product has got returned. This trigger finds the nearest store and increases its inventory

3)

```

CREATE TRIGGER InvoiceRoleCheck
ON Invoices
FOR INSERT
AS
BEGIN

    IF NOT EXISTS (
        SELECT * FROM UserRoles ur
        JOIN Roles r ON ur.RoleId = r.RoleId
        JOIN inserted i ON i.DemandingUserId = ur.UserId
        WHERE r.Name = 'StoreProductDemand'
    )
        THROW 60001 , 'Insufficient access for user to place order' , 1;

END

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists several databases and their objects, including AP, Examples, MyGuitarShop, ProductOrder, and TargetCompetitor. The TargetCompetitor database is selected. The SQL Query Editor window on the right contains the T-SQL code for creating the trigger. The code defines a trigger named 'InvoiceRoleCheck' on the 'Invoices' table for the 'INSERT' event. It uses an IF NOT EXISTS block to check if a user has the 'StoreProductDemand' role. If the condition is false, it throws an error with code 60001 and message 'Insufficient access for user to place order'. The trigger ends with an END keyword. Below the code, the 'Messages' pane shows the command completed successfully with a completion time of 2021-12-14T19:41:17.3444979+03:30.

```

CREATE TRIGGER InvoiceRoleCheck
ON Invoices
FOR INSERT
AS
BEGIN

    IF NOT EXISTS (
        SELECT * FROM UserRoles ur
        JOIN Roles r ON ur.RoleId = r.RoleId
        JOIN inserted i ON i.DemandingUserId = ur.UserId
        WHERE r.Name = 'StoreProductDemand'
    )
        THROW 60001 , 'Insufficient access for user to place order' , 1;

END

```

Comment: This trigger checks the if the user is authorized to insert data into invoices and throws an error when there is no enough access.

4)

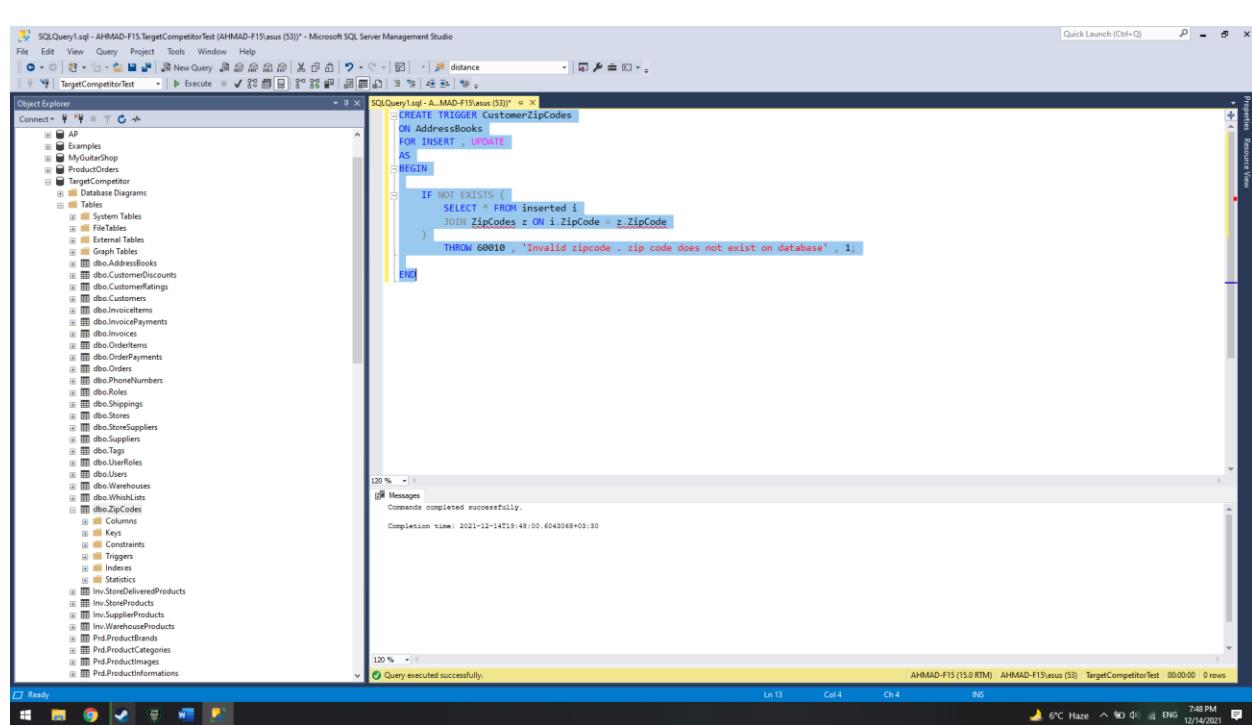
```

CREATE TRIGGER CustomerZipCodes
ON AddressBooks
FOR INSERT , UPDATE
AS
BEGIN

    IF NOT EXISTS (
        SELECT * FROM inserted i
        JOIN ZipCodes z ON i.ZipCode = z.ZipCode
    )
        THROW 60010 , 'Invalid zipcode . zip code does not exist on database' , 1;

END

```



Comment: This trigger fires and then throws an error if an invalid zip code is entered into database.

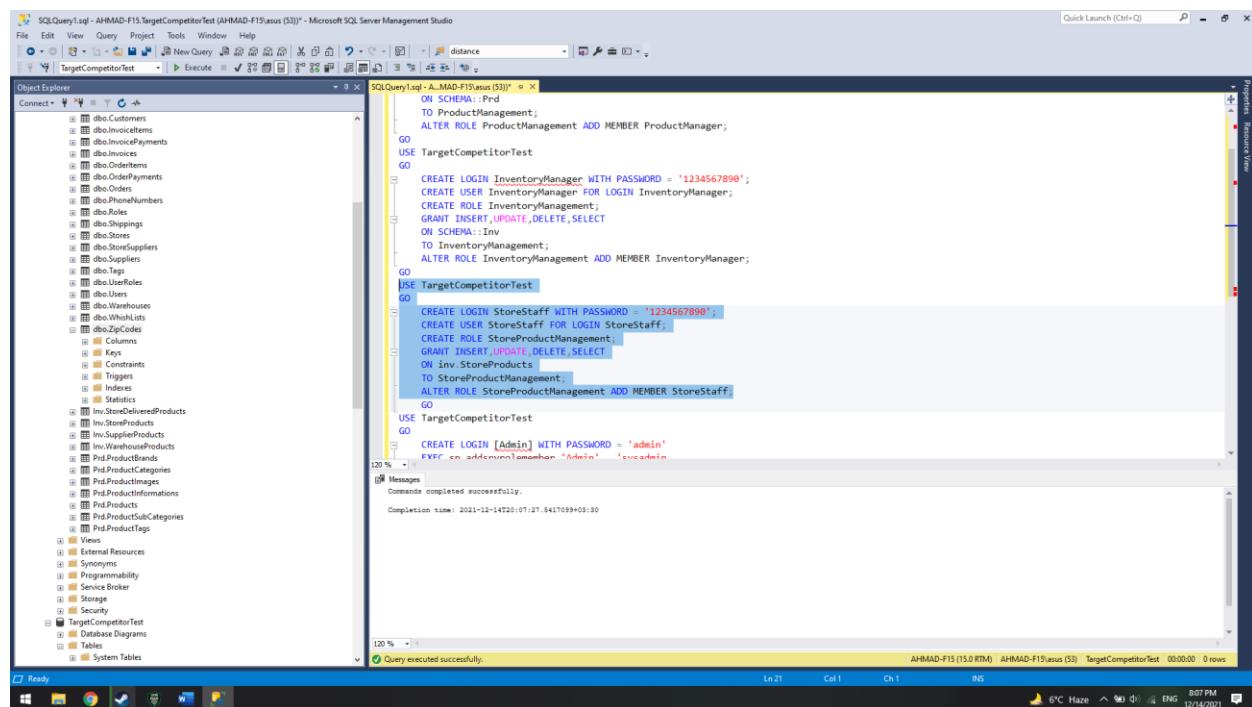
Scripts to create users with various security levels, passwords, and roles

1)

USE TargetCompetitorTest

GO

```
CREATE LOGIN StoreStaff WITH PASSWORD = '1234567890';
CREATE USER StoreStaff FOR LOGIN StoreStaff;
CREATE ROLE StoreProductManagement;
GRANT INSERT,UPDATE,DELETE,SELECT
ON inv.StoreProducts
TO StoreProductManagement;
ALTER ROLE StoreProductManagement ADD MEMBER StoreStaff;
```



Comment: Here we wrote a script that allows Store staff to perform DML on store products table.

2)

```
USE TargetCompetitorTest
GO
```

```
CREATE LOGIN InventoryManager WITH PASSWORD = '1234567890';
CREATE USER InventoryManager FOR LOGIN InventoryManager;
CREATE ROLE InventoryManagement;
GRANT INSERT,UPDATE,DELETE,SELECT
ON SCHEMA::Inv
TO InventoryManagement;
ALTER ROLE InventoryManagement ADD MEMBER InventoryManager;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Customers, Invoices, Payments, etc. The central pane displays a query window titled 'SQLQuery1.sql' containing the following T-SQL script:

```
CREATE LOGIN ProductManager WITH PASSWORD = '1234567890';
CREATE ROLE ProductManagement;
GRANT INSERT, UPDATE , DELETE , SELECT
ON SCHEMA::Prod
TO ProductManagement;
ALTER ROLE ProductManagement ADD MEMBER ProductManager;
GO
USE TargetCompetitorTest
GO
CREATE LOGIN InventoryManager WITH PASSWORD = '1234567890';
CREATE USER InventoryManager FOR LOGIN InventoryManager;
CREATE ROLE InventoryManagement;
GRANT INSERT,UPDATE,DELETE,SELECT
ON SCHEMA::Inv
TO InventoryManagement;
ALTER ROLE InventoryManagement ADD MEMBER InventoryManager;
GO
USE TargetCompetitorTest
GO
CREATE LOGIN StoreStaff WITH PASSWORD = '1234567890';
CREATE USER StoreStaff FOR LOGIN StoreStaff;
CREATE ROLE StoreProductManagement;
GRANT INSERT,UPDATE,DELETE,SELECT
ON Inv.StoreProducts
TO StoreProductManagement;
ALTER ROLE StoreProductManagement ADD MEMBER StoreStaff;
GO
```

The status bar at the bottom indicates 'Query executed successfully.' and shows the completion time as 2021-12-14T20:07:27.5417099+03:30.

Comment: Here we wrote a script that allows inventory manager to perform DML on INV schema.

3)

```
USE TargetCompetitorTest  
GO
```

```
CREATE LOGIN [Admin] WITH PASSWORD = 'admin'  
EXEC sp_addsrvrolemember 'Admin', 'sysadmin'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database schema for 'TargetCompetitorTest' with various tables like Customers, Invoices, and Products. The main window contains a query editor with the following T-SQL script:

```
CREATE LOGIN InventoryManager WITH PASSWORD = '1234567890';  
CREATE USER InventoryManager FOR LOGIN InventoryManager;  
CREATE ROLE InventoryManagement;  
GRANT INSERT,UPDATE,DELETE,SELECT  
ON SCHEMA::Inv  
TO InventoryManagement;  
ALTER ROLE InventoryManagement ADD MEMBER InventoryManager;  
GO  
USE TargetCompetitorTest  
GO  
CREATE LOGIN StoreStaff WITH PASSWORD = '1234567890';  
CREATE USER StoreStaff FOR LOGIN StoreStaff;  
CREATE ROLE StoreProductManagement;  
GRANT INSERT,UPDATE,DELETE,SELECT  
ON Inv.StoreProducts  
TO StoreProductManagement;  
ALTER ROLE StoreProductManagement ADD MEMBER StoreStaff;  
GO  
USE TargetCompetitorTest  
GO  
CREATE LOGIN [Admin] WITH PASSWORD = 'admin'  
EXEC sp_addsrvrolemember 'Admin', 'sysadmin'
```

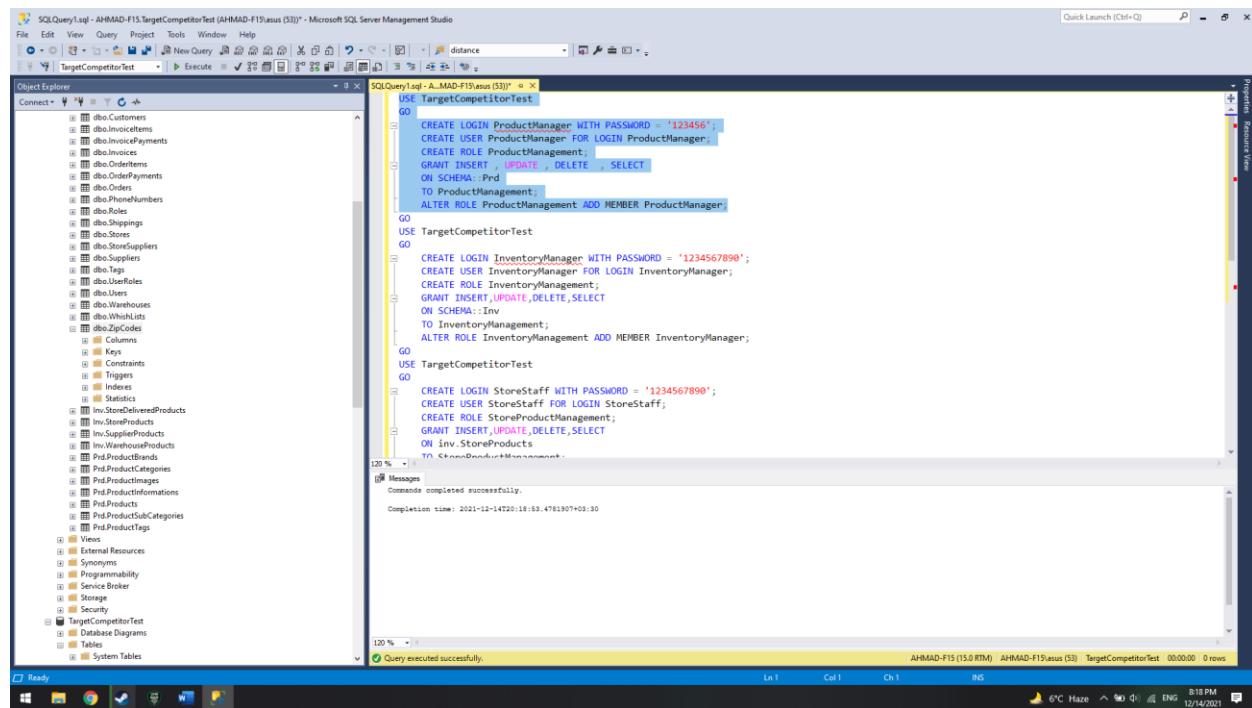
The 'Messages' pane at the bottom shows the command completed successfully with a completion time of 2021-12-14T20:11:58.0723264+00:00.

Comment: Here we wrote a script that gives full access to sysadmin

4)

USE TargetCompetitorTest
GO

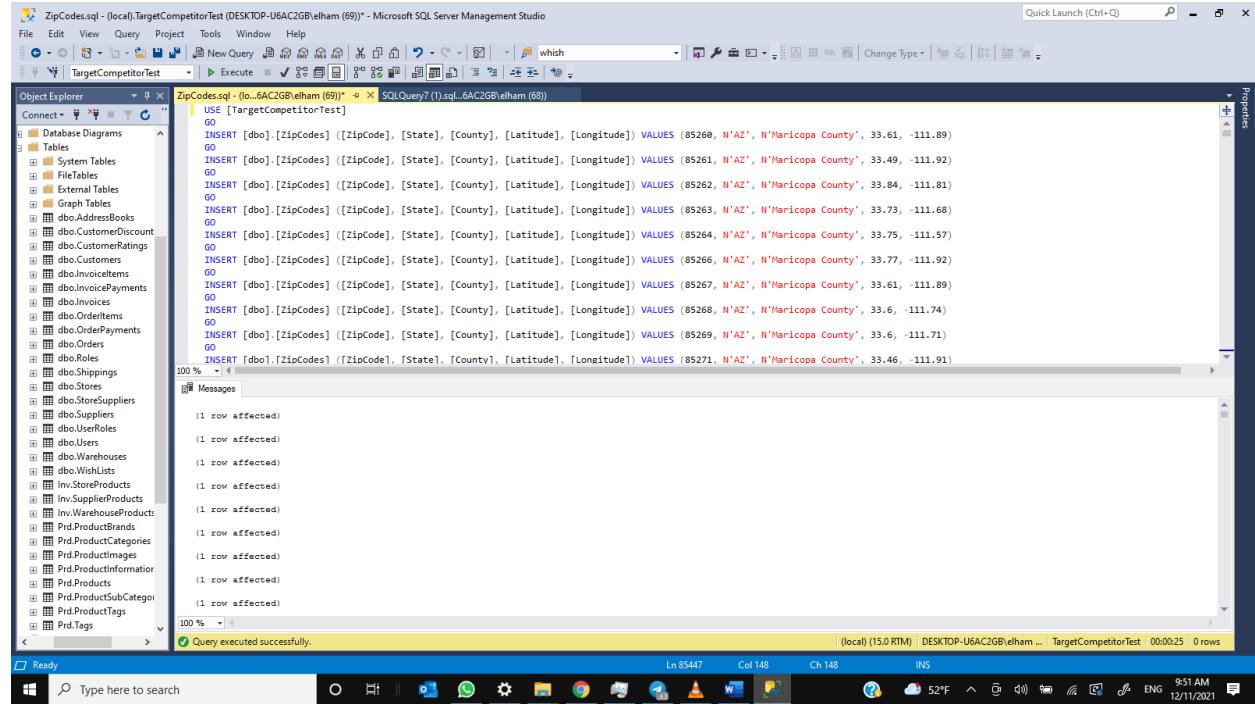
```
CREATE LOGIN ProductManager WITH PASSWORD = '123456';
CREATE USER ProductManager FOR LOGIN ProductManager;
CREATE ROLE ProductManagement;
GRANT INSERT , UPDATE , DELETE , SELECT
ON SCHEMA::Prd
TO ProductManagement;
ALTER ROLE ProductManagement ADD MEMBER ProductManager;
```



Comment: Here we wrote a script that allows product manager to perform DML on PRD schema.

Database population with test data:

In this section, we insert data to be able to start testing the database. we start by inserting all the zip codes of all the regions in the country.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like tables, stored procedures, and triggers. The main pane displays a T-SQL script named 'ZipCodes.sql' which contains multiple INSERT statements for the 'ZipCodes' table. The script inserts data for several zip codes in Maricopa County, Arizona, with coordinates ranging from approximately 33.61, -111.89 to 33.73, -111.68. The status bar at the bottom right indicates '0 rows' affected, and the message bar says 'Query executed successfully.'

```

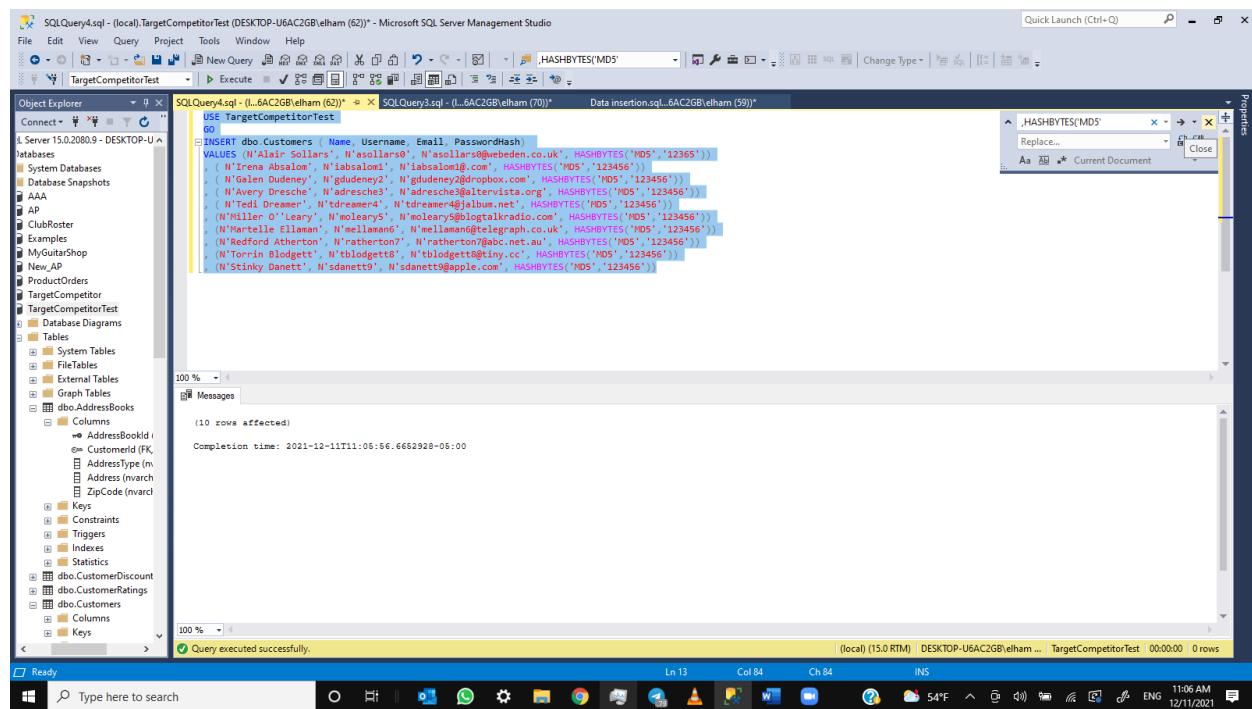
USE [TargetCompetitorTest]
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85260, 'AZ', 'Maricopa County', 33.61, -111.89)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85261, 'AZ', 'Maricopa County', 33.49, -111.92)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85262, 'AZ', 'Maricopa County', 33.84, -111.81)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85263, 'AZ', 'Maricopa County', 33.73, -111.68)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85264, 'AZ', 'Maricopa County', 33.75, -111.57)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85266, 'AZ', 'Maricopa County', 33.77, -111.92)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85267, 'AZ', 'Maricopa County', 33.61, -111.89)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85268, 'AZ', 'Maricopa County', 33.6, -111.74)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85269, 'AZ', 'Maricopa County', 33.6, -111.71)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85271, 'AZ', 'Maricopa County', 33.46, -111.91)
GO

```

Comment: you can find the codes in appendix section (part1)

```
USE TargetCompetitorTest
GO
```

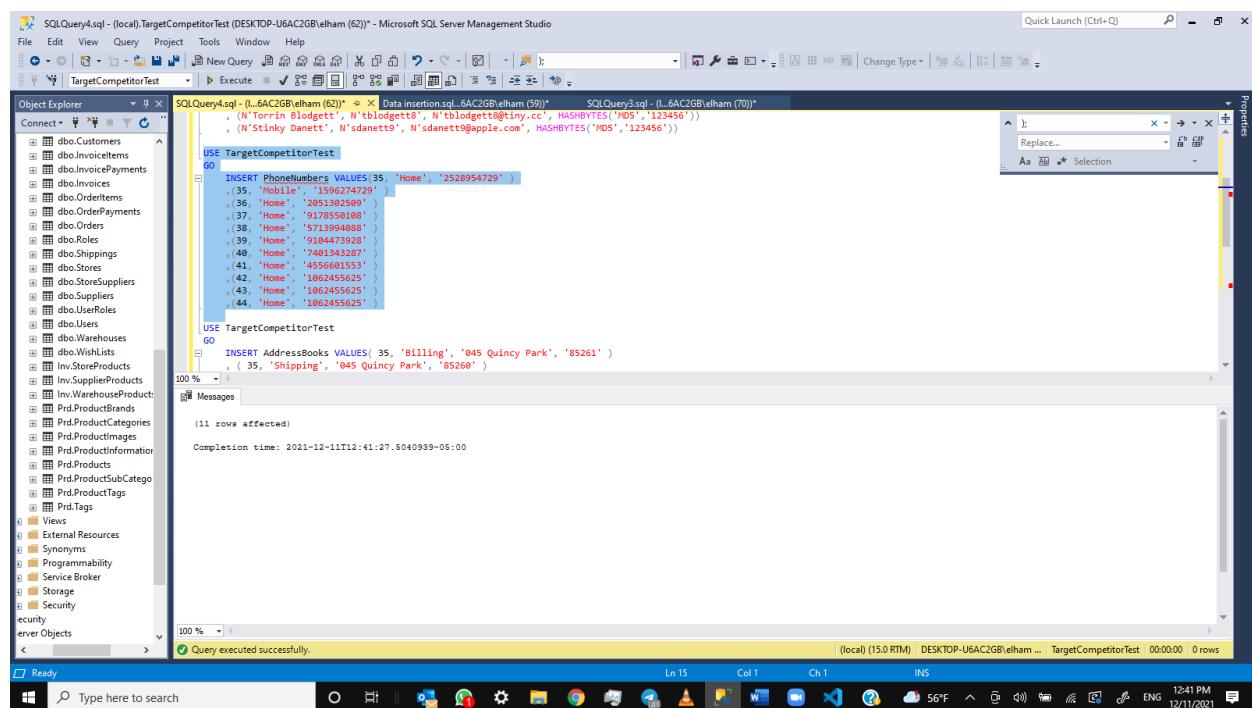
```
INSERT dbo.Customers (Name, Username, Email, PasswordHash)
VALUES (N'Alair Sollars', N'asollars0', N'asollars0@webeden.co.uk', HASHBYTES('MD5','123456'))
, ( N'Irena Absalom', N'iabsalom1', N'iabsalom1@.com', HASHBYTES('MD5','123456'))
, ( N'Galen Dudeney', N'gdudeney2', N'gdudeney2@dropbox.com', HASHBYTES('MD5','123456'))
, ( N'Avery Dresche', N'adresche3', N'adresche3@altervista.org', HASHBYTES('MD5','123456'))
, ( N'Tedi Dreamer', N'tdreamer4', N'tdreamer4@jalbum.net', HASHBYTES('MD5','123456'))
, (N'Miller O'Leary', N'moleary5', N'moleary5@blogtalkradio.com', HASHBYTES('MD5','123456'))
, (N'Martelle Ellaman', N'mellaman6', N'mellaman6@telegraph.co.uk', HASHBYTES('MD5','123456'))
, (N'Redford Atherton', N'ratherton7', N'ratherton7@abc.net.au', HASHBYTES('MD5','123456'))
, (N'Torrin Blodgett', N'tblodgett8', N'tblodgett8@tiny.cc', HASHBYTES('MD5','123456'))
, (N'Stinky Danett', N'sdanett9', N'sdanett9@apple.com', HASHBYTES('MD5','123456'))
```



Comment: here we inserted name, username, email and password for ten customers.

USE TargetCompetitorTest
GO

```
INSERT PhoneNumbers VALUES(35, 'Home', '2528954729')
,(35, 'Mobile', '1596274729')
,(36, 'Home', '2051302509')
,(37, 'Home', '9178550108')
,(38, 'Home', '5713994088')
,(39, 'Home', '9104473928')
,(40, 'Home', '7401343287')
,(41, 'Home', '4556601553')
,(42, 'Home', '1062455625')
,(43, 'Home', '1062455625')
,(44, 'Home', '1062455625')
```



Comment: phone numbers have got inserted for each customer. As can be seen, each customer can have various phone numbers with different types (home, work, mobile)

```
USE TargetCompetitorTest
GO
```

```
INSERT AddressBooks VALUES(35, 'Billing', '045 Quincy Park', '85261')
, ( 35, 'Shipping', '045 Quincy Park', '85260' )
, ( 36, 'Shipping', '1101 Coleman Terrace', '85262' )
, ( 37, 'Shipping', '2 Spenser Avenue', '85263' )
, ( 38, 'Shipping', '2253 Prairie Rose Place', '85264' )
, ( 39, 'Shipping', '28054 Nova Crossing', '85266' )
, ( 40, 'Shipping', '24 Rutledge Junction', '85267' )
, ( 41, 'Shipping', '83 Gerald Terrace', '85268' )
, ( 42, 'Shipping', '7031 Gina Place', '85269' )
, ( 43, 'Shipping', '0 New Castle Street', '85271' )
, ( 44, 'Shipping', '43487 Sunnyside Parkway', '85272')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'TargetCompetitorTest' is selected. In the center pane, a query window displays the following T-SQL code:

```
USE TargetCompetitorTest
GO
INSERT AddressBooks VALUES(35, 'Billing', '045 Quincy Park', '85261')
, ( 35, 'Shipping', '045 Quincy Park', '85260' )
, ( 36, 'Shipping', '1101 Coleman Terrace', '85262' )
, ( 37, 'Shipping', '2 Spenser Avenue', '85263' )
, ( 38, 'Shipping', '2253 Prairie Rose Place', '85264' )
, ( 39, 'Shipping', '28054 Nova Crossing', '85266' )
, ( 40, 'Shipping', '24 Rutledge Junction', '85267' )
, ( 41, 'Shipping', '83 Gerald Terrace', '85268' )
, ( 42, 'Shipping', '7031 Gina Place', '85269' )
, ( 43, 'Shipping', '0 New Castle Street', '85271' )
, ( 44, 'Shipping', '43487 Sunnyside Parkway', '85272')
```

The 'Messages' pane at the bottom shows the results of the execution:

```
(11 rows affected)
Completion time: 2021-12-11T12:46:04.6599821-05:00
```

The status bar at the bottom right indicates 'Query executed successfully.'

Comment: Addresses have got inserted for each customer. As can be seen, each customer can have various addresses with different types (billing, shipping)

```
USE TargetCompetitorTest
GO
```

```
INSERT Prd.ProductCategories VALUES('Home & Kitchen')
, ('Sports & Outdoors')
, ('Toys & Games')
, ('Beauty & Personal Care')
, ('Health, Household & Baby Care')
, ('Kitchen & Dining')
, ('Office Products')
, ('Garden & Outdoor')
, ('Tools & Home Improvement')
, ('Pet Supplies')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there are several database objects listed under the 'TargetCompetitorTest' database. In the center, the SQL Query Editor window contains the following SQL code:

```
USE TargetCompetitorTest
GO
INSERT Prd.ProductCategories VALUES('Home & Kitchen')
, ('Sports & Outdoors')
, ('Toys & Games')
, ('Beauty & Personal Care')
, ('Health, Household & Baby Care')
, ('Kitchen & Dining')
, ('Office Products')
, ('Garden & Outdoor')
, ('Tools & Home Improvement')
, ('Pet Supplies')
```

The 'Messages' pane at the bottom of the editor shows the results of the execution:

```
(10 rows affected)
Completion time: 2021-12-11T12:59:33.4364779-06:00
```

In the status bar at the bottom right, it says 'Query executed successfully.' and provides system information like the date and time.

Comment: here we identified product categories. Next step would be defining product's subcategories.

```
USE TargetCompetitorTest
```

```
GO
```

```
INSERT Prd.ProductSubCategories VALUES(1, 'Furniture & Décor')
, (1, 'Kitchen & Dining')
, (1, 'Bedding & Bath')
, (1, 'Patio, Lawn, & Garden')
, (2, 'Archery')
, (2, 'Camping')
, (2, 'Canoeing')
, (2, 'Climbing')
, (3, 'Board Games')
, (3, 'Toy Foam Blasters & Guns')
, (3, 'Electronic Learning & Education Toys')
, (3, 'Kids' Electronics')
, (4, 'Facial Skin Care Products')
, (4, 'Body Skin Care Products')
, (4, 'Eye Treatment Products')
, (4, 'Bath & Bathing Accessories')
, (5, 'Household Supplies')
, (5, 'Health Care Products')
, (5, 'Baby Care Products')
, (5, 'Industrial & Scientific')
, (6, 'Dining Room Furniture')
, (6, 'Kitchen Utensils & Gadgets')
, (6, 'Kitchen Small Appliances')
, (6, 'Dining & Entertaining')
, (7, 'Desk Accessories')
, (7, 'Workspace Organizers')
, (7, 'Paper & Printable Media')
, (7, 'Filing Products')
, (8, 'Outdoor Benches')
, (8, 'Planter Raised Beds')
, (8, 'Deck Boxes')
, (8, 'Patio Conversation Sets')
, (9, 'Power Tools')
, (9, 'Hand Tools')
, (9, 'Power Tool Parts & Accessories')
, (9, 'Tool Boxes')
, (10, 'Dog Housebreaking Supplies')
, (10, 'Dog Beds & Furniture')
, (10, 'Dog Carriers & Travel Products')
, (10, 'Dog Bowls & Dishes')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left is the Object Explorer pane, which lists various database objects like Customers, InvoiceItems, Payments, Invoices, OrderItems, OrderPayments, Orders, Roles, Shippings, Stores, StoreSuppliers, Suppliers, UserRoles, Users, Warehouses, Wishlists, and several Product-related tables and views. The main central area contains two query windows: 'SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62))' and 'SQLQuery3.sql - (local).6AC2GB\elham (59)'. The 'SQLQuery4.sql' window displays T-SQL code for creating a new database and inserting data into the 'Prod.ProductSubCategories' table. The 'Messages' pane at the bottom indicates '(40 rows affected)' and a completion time of '2021-12-11T13:41:58.4231919-05:00'. The status bar at the bottom right shows '(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows'.

```
(Sports & Outdoors)
(Toys & Games)
(Health & Personal Care)
(Kitchen, Household & Baby Care')
(Office Products)
(Garden & Outdoor')
(Tools & Home Improvement')
(Pet Supplies)

GO
USE TargetCompetitorTest
GO

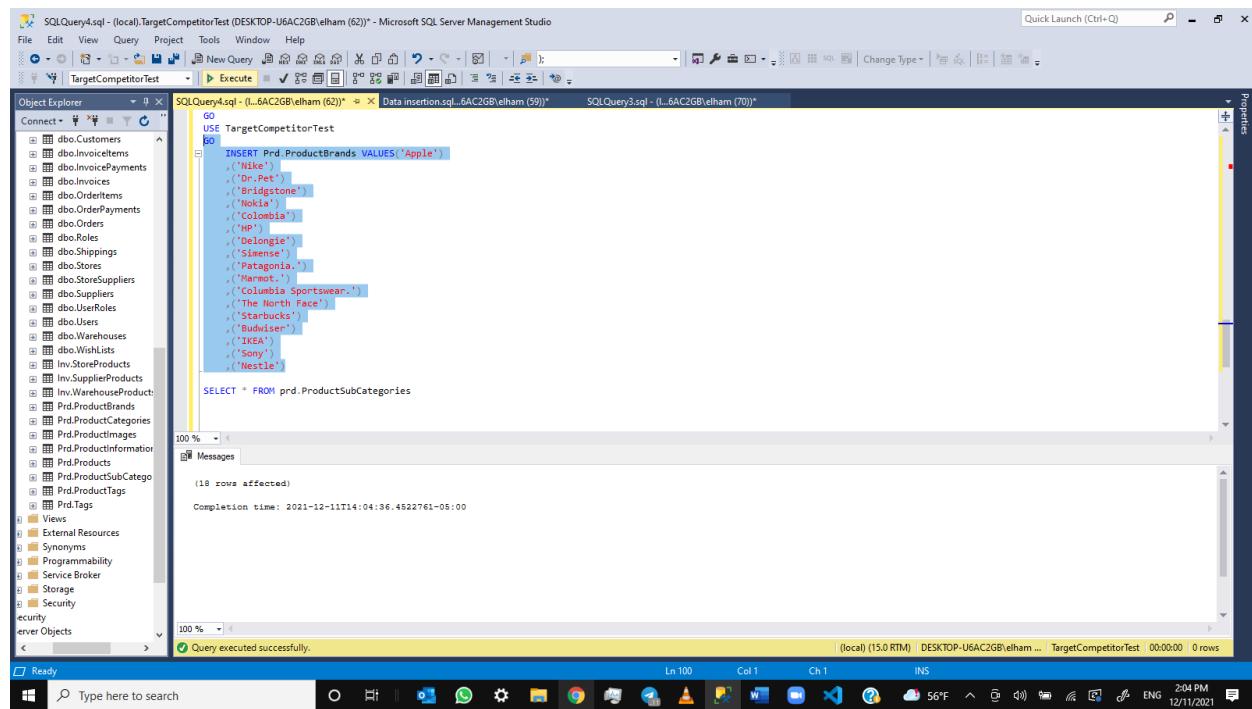
INSERT Prod.ProductSubCategories VALUES(1, 'Furniture & Décor')
, (1, 'Kitchen & Dining')
, (1, 'Bedding & Bath')
, (1, 'Patio, Lawn, & Garden')
, (2, 'Archery')
, (2, 'Camping')
, (2, 'Canoeing')
, (2, 'Climbing')
, (3, 'Board Games')

(40 rows affected)
```

Comment: Here we identified four subcategories for each category.

GO

```
INSERT Prd.ProductBrands VALUES('Apple')
,('Nike')
,('Dr.Pet')
,('Bridgestone')
,('Nokia')
,('Colombia')
,('HP')
,('Delongie')
,('Simense')
,('Patagonia.')
,('Marmot.')
,('Columbia Sportswear.')
,('The North Face')
,('Starbucks')
,('Budwiser')
,('IKEA')
,('Sony')
,('Nestle')
```



Comment: Some brands got inserter to product brands table.

```
USE TargetCompetitorTest
GO
```

```
INSERT prd.Products VALUES('Dog food',NULL,'$10',3,40 )
, ('Drilling tool box',NULL,'$200',9,36 )
, ('White barbecue set',NULL,'$1000',16,32 )
, ('A4 paper for HP13240',NULL,'$5.50',7,27 )
, ('Powerfull mixer 2',NULL,'$36',8,23 )
, ('Chocolate flavored multivitamin', 'Imported from italy','$22',18,19 )
, ('Climbing boots',NULL,'$260',13,8 )
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays a tree view of database objects under the 'TargetCompetitorTest' database, including tables like Customers, Invoiceltems, InvoicePayments, Invoices, OrderItems, Orders, Roles, Shippings, Stores, and various Product-related tables. The central pane contains two tabs: 'SQLQuery4.sql - (L...6AC2GB\elham (62))' and 'Data insertion.sql - (L...6AC2GB\elham (59))'. The 'Data insertion.sql' tab contains the SQL code shown above. The 'Messages' pane at the bottom shows the execution results: '(7 rows affected)' and 'Completion time: 2021-12-11T14:35:00.8528722-05:00'. A yellow status bar at the bottom right indicates 'Query executed successfully.' The system tray at the bottom of the screen shows the date and time as '12/11/2021 2:35 PM'.

Comment: here real-world products has got entered and the subcategory and brand has got set.

```
USE TargetCompetitorTest
```

```
GO
```

```
INSERT Prd.ProductImages VALUES (101,
'https://cdn.shopify.com/s/files/1/0355/3833/7836/products/OpenFarm_Dry_Puppy_Food_Product_3.png?v=1609777401')
,(102, 'https://image.made-in-china.com/202f0j00GTofLBCPgnqW/Wholesale-Custom-SGS-Household-Car-Repairing-Hand-Tool-Set.jpg')
,(103, 'https://cdn.shopify.com/s/files/1/2503/7608/products/KT130WH-ST_480x384.jpg?v=1574885326')
,(104, 'https://m.media-amazon.com/images/I/71+iSOR3KNL._AC_SL1500_.jpg')
,(105, 'https://m.media-amazon.com/images/I/61WUyV25oVL._AC_SX466_.jpg')
,(106, 'https://img.thrivemarket.com/store/full/8/5/858982004307-1_1_1.jpg')
,(107, 'https://i1.wp.com/www.northwestpineguides.com/wp-content/uploads/2018/04/la-sportiva-nepal-evo-gore-tex-mountaineering-boots.Yellow.01.jpg?fit=1146%2C1146&ssl=1')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'TargetCompetitorTest' is selected. In the center pane, there is a query window titled 'Data insertion.sql - not connected'. The query is:

```
USE TargetCompetitorTest
GO
INSERT Prd.ProductImages VALUES (101, 'https://cdn.shopify.com/s/files/1/0355/3833/7836/products/OpenFarm_Dry_Puppy_Food_Product_3.png?v=1609777401')
,(102, 'https://image.made-in-china.com/202f0j00GTofLBCPgnqW/Wholesale-Custom-SGS-Household-Car-Repairing-Hand-Tool-Set.jpg')
,(103, 'https://cdn.shopify.com/s/files/1/2503/7608/products/KT130WH-ST_480x384.jpg?v=1574885326')
,(104, 'https://m.media-amazon.com/images/I/71+iSOR3KNL._AC_SL1500_.jpg')
,(105, 'https://m.media-amazon.com/images/I/61WUyV25oVL._AC_SX466_.jpg')
,(106, 'https://img.thrivemarket.com/store/full/8/5/858982004307-1_1_1.jpg')
,(107, 'https://i1.wp.com/www.northwestpineguides.com/wp-content/uploads/2018/04/la-sportiva-nepal-evo-gore-tex-mountaineering-boots.Yellow.01.jpg?fit=1146%2C1146&ssl=1')
USE TargetCompetitorTest
GO
```

In the 'Messages' pane, it shows '(7 rows affected)' and the completion time '2021-12-12T00:51:59.9846100-05:00'. At the bottom of the screen, the taskbar shows various open applications and the system status.

Comment: here we created the product images table which contains a link to the product's image.

```
USE TargetCompetitorTest
GO
```

```
INSERT Prd.ProductInformations VALUES(101,'Background','Pet food is a specialty food for domesticated animals that is formulated according to their nutritional needs. Pet food generally consists of meat, meat byproducts, cereals, grain, vitamins, and minerals.')
,(102,'Product details','Open the toolbox and get ready for role-play fun with the electronic fix-it tray, drill, hammer, wrench, nails and screws')
,(103,'Rectangular Folding Table','Off White Portable Folding Utility Table, Plastic Picnic Party Dining Camp Table for Parties Picnic Barbecue Outdoor')
,(104,'Acid free paper','A vast majority of HP photo and specialty papers for inkjet printers meet the traditional definition of acid-free and lignin-free')
,(105,'All you need','One mixer meets your two needs. This mixer can be detach from the base as a lightweight speed hand mixer')
,(106,'Only 2g of sugar per piece','Chocolate multivitamin for kids that is made with Fair-Trade milk chocolate, non-GMO ingredients and only 2g of sugar per piece')
,(107,'Specific design','Mountaineering boots are a type of footwear used in mountain climbing. They are designed specifically for moving over harsh terrain.')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Customers, Invoicetems, InvoicePayments, etc. The central SQL Query Editor window contains the following SQL code:

```
USE TargetCompetitorTest
GO
INSERT Prd.ProductInformations VALUES(101,'Background','Pet food is a specialty food for domesticated animals that is formulated according to their nutritional needs. Pet food generally consists of meat, meat byproducts, cereals, grain, vitamins, and minerals.')
,(102,'Product details','Open the toolbox and get ready for role-play fun with the electronic fix-it tray, drill, hammer, wrench, nails and screws')
,(103,'Rectangular Folding Table','Off White Portable Folding Utility Table, Plastic Picnic Party Dining Camp Table for Parties Picnic Barbecue Outdoor')
,(104,'Acid free paper','A vast majority of HP photo and specialty papers for inkjet printers meet the traditional definition of acid-free and lignin-free')
,(105,'All you need','One mixer meets your two needs. This mixer can be detach from the base as a lightweight speed hand mixer')
,(106,'Only 2g of sugar per piece','Chocolate multivitamin for kids that is made with Fair-Trade milk chocolate, non-GMO ingredients and only 2g of sugar per piece')
,(107,'Specific design','Mountaineering boots are a type of footwear used in mountain climbing. They are designed specifically for moving over harsh terrain.')

SELECT * FROM prd.Products
```

The Messages pane at the bottom indicates "7 rows affected". The status bar at the bottom right shows the completion time as 2021-12-12T01:35:59.7209084-05:00.

```
USE TargetCompetitorTest
GO
```

```
INSERT Orders VALUES ('Returned', 'Online', 'CreditCard', 35, DEFAULT)
, ('Registered', 'Online', 'CreditCard', 37, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 38, DEFAULT)
, ('Paid', 'In-Store', 'CreditCard', 36, DEFAULT)
, ('Returned', 'In-Store', 'CreditCard', 39, DEFAULT)
, ('Paid', 'In-Store', 'Cash', 40, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 41, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 42, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 43, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 44, DEFAULT)
```

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "Data insertion-.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio". The main window displays a T-SQL script for inserting data into the "Orders" table. The script includes a multi-line comment explaining the purpose of the data. The execution message at the bottom indicates that 10 rows were affected. The status bar at the bottom right shows the completion time as 2021-12-13T00:27:45.3039093-05:00.

```
GO
USE TargetCompetitorTest
GO
INSERT Orders VALUES ('Returned', 'Online', 'CreditCard', 35, DEFAULT)
, ('Registered', 'Online', 'CreditCard', 37, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 38, DEFAULT)
, ('Paid', 'In-Store', 'CreditCard', 36, DEFAULT)
, ('Returned', 'In-Store', 'CreditCard', 39, DEFAULT)
, ('Paid', 'In-Store', 'Cash', 40, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 41, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 42, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 43, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 44, DEFAULT)

(
    OrderId INT IDENTITY PRIMARY KEY,
    Status NVARCHAR(10) NOT NULL, --Returned,Paid,....
    Type NVARCHAR(12) NOT NULL , --Online , In-Store , WarehouseReq , SupplierReq
    PaymentType NVARCHAR(10) NOT NULL, --Cash, CreditCard
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),
    OrderDate DATETIME NOT NULL
)
GO
--Mountaineering boots are a type of footwear used in mountain climbing. They are designed specifically for moving over harsh terrain.
--Inserted 10 rows of data into the Orders table.
```

(10 rows affected)

Completion time: 2021-12-13T00:27:45.3039093-05:00

(local) (15.0 RTM) | DESKTOP-U6AC2GB\elham ... | TargetCompetitorTest | 00:00:00 | 0 rows

Ready Type here to search Ln 150 Col 1 Ch 1 INS

Comment: Here we entered some orders.

USE TargetCompetitorTest

GO

```
INSERT Stores VALUES('Westcott', '504 Westott Ave', '13210', '9143875429', '9144875429',  
'Westcott303@TCT.com', 'TCT.com/Westcott303')
```

('Mckinley County', '504 Mckinley County' '87302', '5931547862', '5931547863',
'504MckinleyCounty@TCT.com', 'TCT.com/504MckinleyCounty')

,('Lubbock County', '928 Lubbock County', '13210', '318524988', '9318524987',
'928LubbockCounty@TCT.com', 'TCT.com/928LubbockCounty')

,('Amelia County', '951 Amelia County', '23822', '7852147963', '7

(Oklahoma County, '630 Oklahoma County', '73127', '2951753963', '2951753964',
(TCT.com/951AmeliaCounty)')

(('McDowell County', '523 McDowell County', '28749', '4159785248', '4159785249',
('630OklahomaCounty@TCT.com', 'TCT.com/630OklahomaCounty'))

(.'McDowell County', '23 McDowell County', '23445', '4159553404', '+1337552-523McDowellCounty@TCT.com', 'TCT.Com/523McDowellCounty')
(.'Allen County', '234 Allen County', '46861', '4159555412', '4159555413', '234Allen

(. 'Alchin County' '234 Alchin County' '46881' '4155555412' '415555
TCT.com/234AllenCounty')
. ('Chippewa County' '214 Chippewa County' '54748' '1455888763')

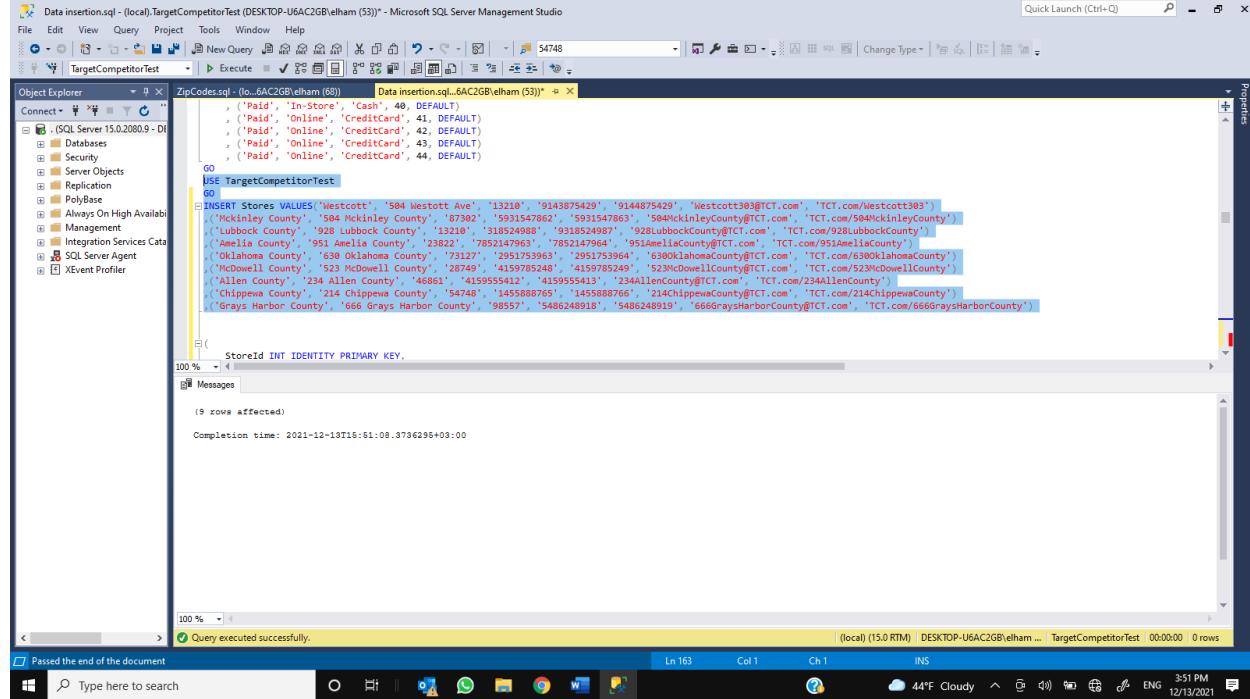
Chippewa County, 214 Chippewa County, 347-8, 1455887-83, 1455887-88, 214ChippewaCounty@TCT.com; TCT.com/214ChippewaCounty)

214 Grays Harbor County ('Grays Harbor County' '66)

,(Grays Harbor County , 666 Grays Harbor County , 98537 , 3488248918 , 3488248918 , '666GraysHarborCounty@TCT.com' 'TCT.com/666GraysHarborCounty')

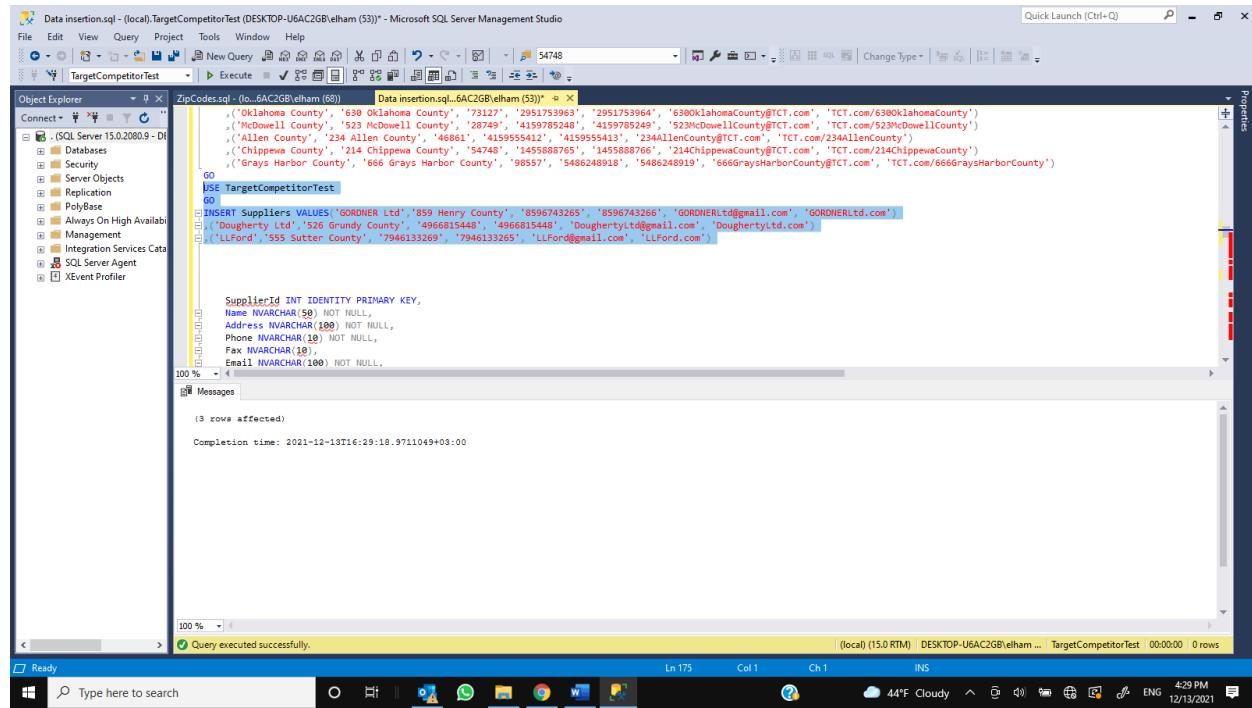
666GraysHarborCounty@CTC.com , CTC.com/666GraysHarborCo

Data insertion.sql - [local].TargetCompetitorTest (DESKTOP-LU6AC2GB\elham (53)) - Microsoft SQL Server Management Studio



Comment: The information of stores has got entered in the store table.

```
USE TargetCompetitorTest
GO
INSERT Suppliers VALUES('GORDNER Ltd', '859 Henry County', '8596743265', '8596743266',
'GORDNERLtd@gmail.com', 'GORDNERLtd.com')
,('Dougherty Ltd', '526 Grundy County', '4966815448', '4966815448', 'DoughertyLtd@gmail.com',
'DoughertyLtd.com')
,('LLFord', '555 Sutter County', '7946133269', '7946133265', 'LLFord@gmail.com', 'LLFord.com')
```

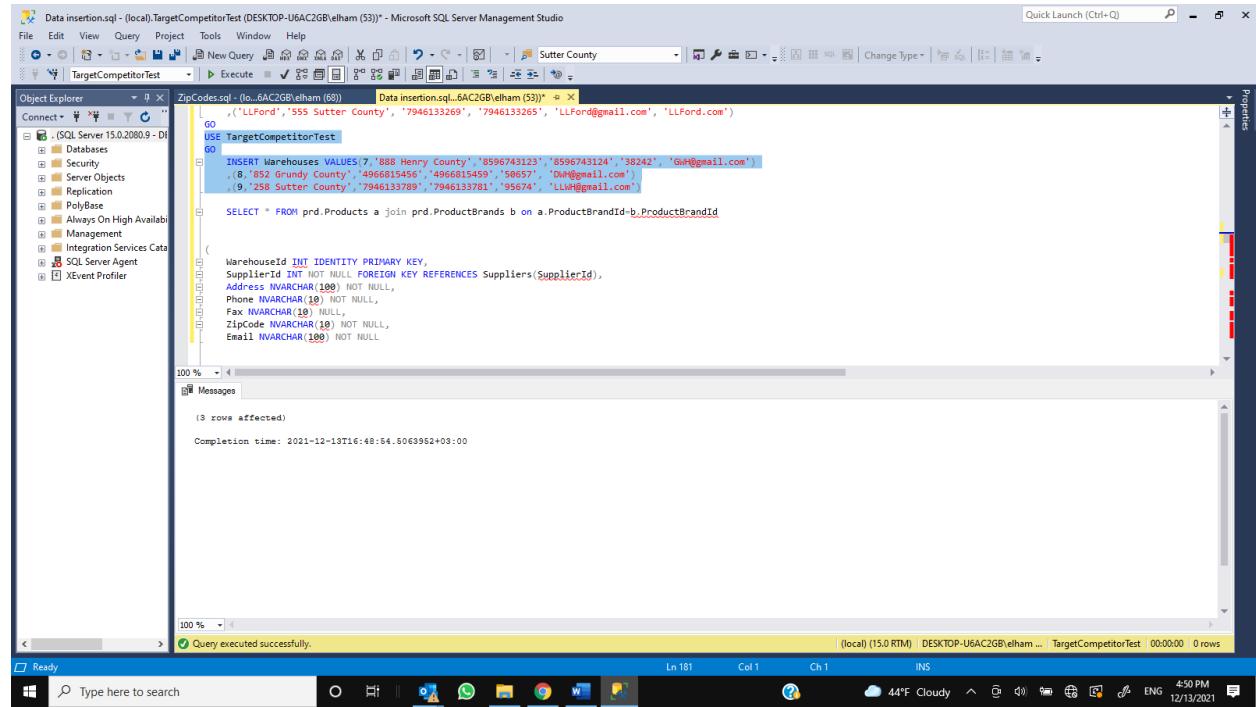


Comment: Here we entered our main three suppliers.

```
USE TargetCompetitorTest
```

```
GO
```

```
INSERT Warehouses VALUES(7,'888 Henry County','8596743123','8596743124','38242','GWH@gmail.com')  
,(8,'852 Grundy County','4966815456','4966815459','50657','DWH@gmail.com')  
,(9,'258 Sutter County','7946133789','7946133781','95674','LLWH@gmail.com')
```



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'Data insertion.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (53))' is open. The code entered is:

```
USE TargetCompetitorTest  
GO  
INSERT Warehouses VALUES(7,'888 Henry County','8596743123','8596743124','38242','GWH@gmail.com')  
,(8,'852 Grundy County','4966815456','4966815459','50657','DWH@gmail.com')  
,(9,'258 Sutter County','7946133789','7946133781','95674','LLWH@gmail.com')  
  
SELECT * FROM prd.Products a join prd.ProductBrands b on a.ProductBrandId=b.ProductBrandId
```

The execution message at the bottom of the window says: 'Query executed successfully.' and 'Completion time: 2021-12-13T16:48:54.5063952+03:00'. The status bar at the bottom right shows: '(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 0 rows'.

Comment: Here we entered our main three warehouses.

```
USE TargetCompetitorTest  
GO
```

```
INSERT Wishlists VALUES(35,108)  
,(36,107)  
,(37,106)  
,(38,105)  
,(39,104)  
,(40,103)  
,(41,102)  
,(42,101)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like MyCustomer, OrderDetails, TargetCompetitor, and TargetCompetitorTest. The TargetCompetitorTest database is selected. The center pane displays the following SQL code:

```
USE TargetCompetitorTest  
GO  
INSERT Wishlists VALUES(35,108)  
,(36,107)  
,(37,106)  
,(38,105)  
,(39,104)  
,(40,103)  
,(41,102)  
,(42,101)
```

The status bar at the bottom indicates "Query executed successfully." The taskbar at the bottom right shows the system date and time as 12/14/2021.

Comment: here we updated the Wish lists table data.

Testing, database's reliability demonstration and logic:

Transaction and stored procedures Testing:

```
DECLARE @IN_CustomerId INT
SET @IN_CustomerId= 36
EXEC CustomerReviews @IN_CustomerId
```

The screenshot shows the SQL Server Management Studio interface. In the center, a query window displays the following T-SQL code:

```
USE TargetCompetitorTest
DECLARE @IN_CustomerId INT
SET @IN_CustomerId= 36
EXEC CustomerReviews @IN_CustomerId
```

Below the code, the results grid shows the following data:

ProductName	Score	Test	RatingDate
Dog feed	3	Test	2021-12-14 20:47:06.820
Drilling tool box	4	Test	2021-12-14 20:47:06.820
White barbecue set	5	Test	2021-12-14 20:47:06.820
A4 paper for HP13240	1	Test	2021-12-14 20:47:06.820
Powerful mixer 2	4	Test	2021-12-14 20:47:06.820
Chocolate flavored multivitamin	2	Test	2021-12-14 20:47:06.820
Drilling boars	4	Test	2021-12-14 20:47:06.820
Cat food	2	Test	2021-12-14 20:47:06.820

At the bottom of the screen, the status bar indicates "Query executed successfully." and shows system information like "AHMAD-F15 (15.0 RTM) AHMAD-F15\ahmads (S3) TargetCompetitorTest 00:00:00 8 rows".

Comment: by running procedures we see that “White barbecue set” has the highest score of 5 and may be considered for promotion.

```
USE TargetCompetitorTest
GO
DECLARE @IN_ProductId INT
SET @IN_ProductId= 108
EXEC ProductReviews @IN_ProductId
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery1.sql - AHMAD-F15.TargetCompetitorTest (AHMAD-F15\asus (53)) - Microsoft SQL Server Management Studio". The main window has a "Results" tab open, displaying the output of the executed query:

CustomerName	Score	Text	RatingDate
Stinky Danett	2	Test	2021-12-14 20:47:06.820

The status bar at the bottom indicates "Query executed successfully." and shows system information like "AHMAD-F15 (15.0 RTM) AHMAD-F15\asus (53) TargetCompetitorTest 00:00:00 1 rows".

Comment: by running procedures we see that the product 108 has one review from Stinky Danett who gave a score of 2 to the product.

```

SELECT * FROM Orders
SELECT * FROM OrderItems
SELECT * FROM OrderPayments
SELECT * FROM Shipments
SELECT * FROM inv.StoreProducts

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists several databases and their objects, including 'MyCustomerShop', 'ProductOrders', 'TargetCompetitor', and 'TargetCompetitorTest'. The 'Tables' node under 'TargetCompetitorTest' is expanded, showing tables like 'AddressBooks', 'CustomerDiscounts', 'CustomerRatings', 'Customer', 'InvoiceItems', 'InvoicePayments', 'Invoices', 'OrderItems', 'OrderPayments', 'OrderProducts', 'PhoneNumbers', 'Roles', 'Shipments', 'Stores', 'SupplierSuppliers', 'SupplierProducts', 'UserRoles', 'Users', 'Warehouses', and 'WishLists'. The 'Scripting' tab in the top ribbon is selected.

The main query window contains the following T-SQL code:

```

DECLARE @ShippingFare MONEY = 34
DECLARE @ExpectedShippingDate DATETIME = GETDATE()
DECLARE @PaymentAmount MONEY = 1200
DECLARE @PaymentDate DATETIME = GETDATE()
DECLARE @Products OrderProduct

INSERT @Products
VALUES(101,3),(102,2)

EXEC PlaceCustomerOrder @CustomerId, @PaymentType , @OrderType , @ShippingService , @ShippingAddressId , @ShippingFare , @ExpectedShippingDate ,
@PaymentAmount , @PaymentDate , @Products

SELECT * FROM Orders
SELECT * FROM OrderItems
SELECT * FROM OrderPayments
SELECT * FROM Shipments
SELECT * FROM inv.StoreProducts

```

The 'Results' tab shows the output of the last SELECT statement, which returns two rows of data:

StoreProductId	StoreId	Productid	Price	Inventory	OrderedInventory	ReturnedInventory
1	3	101	30.00	30.00	3.00	0.00
2	4	102	35.00	10.00	2.00	0.00

A yellow status bar at the bottom indicates: 'Query executed successfully.' and 'AHMAD-F15 (15.0 RTM) AHMAD-F15\jesus (53) TargetCompetitorTest 00:00:00 2 rows'.

Comment: before running the procedure & transaction there is no orders in the database.

```

DECLARE @CustomerId INT = 36
DECLARE @PaymentType NVARCHAR(10) = 'CreditCard'
DECLARE @OrderType NVARCHAR(7) = 'Online'
DECLARE @ShippingService NVARCHAR(5) = 'Fedex'
DECLARE @ShippingAddressId INT = 12
DECLARE @ShippingFare MONEY = 34
DECLARE @ExpectedShippingDate DATETIME = GETDATE()
DECLARE @PaymentAmount MONEY = 1200
DECLARE @PaymentDate DATETIME = GETDATE()
DECLARE @Products OrderProduct

INSERT @Products
VALUES(101,3),(102,2)

EXEC PlaceCustomerOrder @CustomerId,@PaymentType , @OrderType , @ShippingService ,
@ShippingAddressId , @ShippingFare , @ExpectedShippingDate,
@PaymentAmount ,@PaymentDate , @Products

```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - AHMAD-F15.TargetCompetitorTest (AHMAD-F15\ahmuss (53))' displays the provided T-SQL script. The script declares variables for customer ID, payment type, order type, shipping service, shipping address ID, shipping fare, expected shipping date, payment amount, payment date, and products. It then inserts two rows into the @Products temporary table and executes the stored procedure PlaceCustomerOrder with these parameters. The execution results show six rows affected (two rows inserted into @Products and four rows selected from the PlaceCustomerOrder execution). The status bar at the bottom indicates the query was executed successfully.

Comment: by running the script we get the following results.

```

SELECT * FROM Orders
SELECT * FROM OrderItems
SELECT * FROM OrderPayments
SELECT * FROM Shipments
SELECT * FROM inv.StoreProducts

```

```

SQLQuery1.sql - AHMAD-F15\ahmads (SS) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Quick Launch (Ctrl+Q) Distance
TargetCompetitorTest Execute
Object Explorer
Connect+ MyCustomerShop ProductOrders TargetCompetitor
TargetCompetitorTest Database Diagrams
Tables System Tables PartTables External Tables Graph Tables
dbo.AddressBooks dbo.CustomerDiscounts dbo.CustomerRatings dbo.Customers
dbo.InvoiceItems dbo.InvoicePayments dbo.Invoices dbo.OrderItems
dbo.OrderPayments dbo.Orders dbo.PhoneNumbers dbo.Roles
dbo.Shippings dbo.Stores dbo.SupplierSuppliers dbo.Suppliers
dbo.UserRoles dbo.Users dbo.Warehouses dbo.WishLists
Inv.Products Inv.Catagories Inv.Constraints Inv.Keys
Inv.Triggers Inv.Indexes Inv.Triggers
Inv.SupplierProducts Inv.WarehouseProducts Pd.ProductBrands
Pd.ProductCategories Pd.ProductImages Pd.ProductNames
Pd.ProductDescriptions Pd.ProductFormations Pd.ProductForms
Pd.ProductSubCategories Pd.ProductTags Pd.Tags
SELECT * FROM Orders
SELECT * FROM OrderItems
SELECT * FROM OrderPayments
SELECT * FROM Shipments
SELECT * FROM inv.StoreProducts

```

Results

Orderid	Status	Type	PaymentType	CustomerId	OrderDate
13	Paid	Online	CreditCard	36	2021-12-14 21:20:44.130

OrderItemId	OrderId	ProductId	Quantity	UnitPrice	TotalPrice
9	13	101	1.00	10.00	10.00
6	13	102	2.00	200.00	400.00

OrderPaymentId	OrderId	Amount	PaymentDate
3	13	1200.00	2021-12-14 21:20:44.090

ShippingId	OrderId	ShippingService	ShippingAddressId	ShippingFare	ExpectedShippingDate	ActualShippingDate
1	3	13	12	34.00	2021-12-14 21:20:44.090	NULL

StoreProductid	Shelfid	ProductId	Price	Inventory	OrderdInventory	ReturnedInventory
1	3	101	30.00	27.00	3.00	0.00
2	4	102	35.00	8.00	2.00	0.00

Query executed successfully.

Comment: here we can see the ordered inventory decreased.

```
SELECT * FROM Invoices
SELECT * FROM InvoiceItems
SELECT * FROM InvoicePayments
SELECT * FROM inv.WarehouseProducts
SELECT * FROM inv.StoreProducts
```

Comment: before running the procedure and transaction we get the above result.

```

DECLARE @Date DATETIME = GETDATE()
DECLARE @Products WarehouseProduct
INSERT @Products
VALUES(108 , 3 , 7),(110 , 4 , 7)

```

```

EXEC DeliverProductToStoreFromWarehouse 'CreditCard' , 4500 , @Date , 7 , 1, @Products

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a database named 'TargetCompetitorTest'. In the center pane, a query window displays the following T-SQL code:

```

DECLARE @Date DATETIME = GETDATE()
DECLARE @Products WarehouseProduct
INSERT @Products
VALUES(108 , 3 , 7),(110 , 4 , 7)

EXEC DeliverProductToStoreFromWarehouse 'CreditCard' , 4500 , @Date , 7 , 1, @Products

```

Below the code, the results of the execution are shown in the Messages pane:

```

(2 rows affected)
(1 row affected)
(2 rows affected)
(1 row affected)
(2 rows affected)
(0 rows affected)

Completion time: 2021-12-14T21:46:21.721387+08:00

```

At the bottom of the screen, the status bar indicates "Query executed successfully." The system tray shows the date and time as "12/14/2021".

Comment: here we ran the script and we got the following result.

```

SELECT * FROM Invoices
SELECT * FROM InvoiceItems
SELECT * FROM InvoicePayments
SELECT * FROM inv.WarehouseProducts
SELECT * FROM inv.StoreProducts

```

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the session is titled "SQLQuery4.sql - AHMAD-F15.TargetCompetitorTest (AHMAD-F15\ahm...)".

The Object Explorer pane on the left lists various database objects such as Statistics, Inv.SupplierProducts, Inv.WarehouseProducts, Pwd.ProductBrands, Pwd.ProductCategories, Pwd.ProductTags, Pwd.Tags, External Resources, Synonyms, Programmability (including Stored Procedures like dbo.DeliverProductToStoreFromWarehouse, dbo.PlaceCustomerOrder, and dbo.ProductReviews), Functions, Database Triggers, Assemblies, Types, Rules, Defaults, Plan Guides, Sequences, Service Broker, Service, and Security.

The main Results pane displays five result sets:

- Result Set 1:** A single row from the Invoices table. (InvoiceId: 5, SupplierId: 7, DemandingUserId: 1, InvoiceNumber: 5, InvoiceDate: 2021-12-14 21:46:21.710)
- Result Set 2:** Two rows from the InvoiceItems table. (InvoiceItemId: 9, InvoiceId: 5, ProductId: 108, Quantity: 3.00, UnitPrice: 50.00, TotalPrice: 150.00; InvoiceItemId: 10, InvoiceId: 5, ProductId: 110, Quantity: 4.00, UnitPrice: 50.00, TotalPrice: 200.00)
- Result Set 3:** One row from the InvoicePayments table. (InvoicePaymentId: 1, InvoiceId: 5, Amount: 4500.00, PaymentDate: 2021-12-14 21:46:21.653)
- Result Set 4:** Two rows from the WarehouseProducts table. (WarehouseProductId: 1, WarehouseId: 1, ProductId: 108, Inventory: 7.00, OrderedInventory: 3.00, ResumedInventory: 0.00; WarehouseProductId: 2, WarehouseId: 3, ProductId: 110, Inventory: 6.00, OrderedInventory: 4.00, ResumedInventory: 0.00)
- Result Set 5:** Four rows from the StoreProducts table. (StoreProductId: 3, StoreId: 7, ProductId: 101, Price: 30.00, Inventory: 27.00, OrderedInventory: 3.00, ResumedInventory: 0.00; StoreProductId: 4, StoreId: 7, ProductId: 102, Price: 35.00, Inventory: 8.00, OrderedInventory: 2.00, ResumedInventory: 0.00; StoreProductId: 5, StoreId: 7, ProductId: 108, Price: 50.00, Inventory: 9.00, OrderedInventory: 0.00, ResumedInventory: 0.00; StoreProductId: 6, StoreId: 7, ProductId: 110, Price: 50.00, Inventory: 12.00, OrderedInventory: 0.00, ResumedInventory: 0.00)

At the bottom of the results pane, it says "Query executed successfully." The status bar at the bottom right shows "AHMAD-F15 (15.0 RTM) AHMAD-F15\ahm... TargetCompetitorTest 00:00:00 10 rows." The system tray shows icons for network, battery, and time (6°C Haze, 9:47 PM, 12/14/2021).

Comment: after running the script we see that inventory and ordered inventory has got updated.

View Testing:

```
SELECT * FROM Top10Sales
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window with the following content:

```
SELECT * FROM Top10Sales
```

Below the query window, the Object Explorer pane is visible, showing a tree structure of database objects. The 'Tables' node under the 'TargetCompetitorTest' database is expanded, displaying various tables such as ProductOrders, CustomerCounts, Customers, InvoiceItems, InvoicePayments, OrderItems, OrderPayments, Orders, PhoneNumbers, Roles, StoreSettings, Stores, StoreSuppliers, Suppliers, UserRoles, Users, and WishLists.

The results pane at the bottom displays the output of the query:

Name	TotalSale	AverageScore
Dilling tool box	1	4
Dog food	1	3

At the bottom right of the results pane, it says "Query executed successfully." Below the results pane, the status bar shows "AHMAD-F15 (15.0 RTM) AHMAD-F15\ahmads (S) TargetCompetitorTest 00:00:00 2 rows".

Comment: here we can see the top10 sales.

```
SELECT * FROM MostWishedProducts
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window containing the following SQL code:

```
SELECT * FROM MostWishedProducts
```

Below the query window, the results pane displays a table titled "Results". The table has two columns: "Rank" and "Name". The data is as follows:

Rank	Name
1	1 Revived Revived multivitamin
2	2 Climbing boots
3	3 Cat food
4	3 Dog food
5	3 Drilling tool box
6	6 White barbecue set
7	6 All paper for printer 3240
8	9 Powerade flavor 2

At the bottom of the screen, the taskbar shows various icons, and the system tray indicates the date and time as 12/14/2021 at 10:31 PM.

Comment: here we can see the most wished products.

```
SELECT * FROM CustomersWithNearbyStores
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid titled 'Results' with a single row labeled 'NearbyStores'. The data in the grid is as follows:

Name	Address	City	State	Zip
1. Linda Siler	Allen County, Amelia County, Chippewa County, Gray Harbor County, McDowell County, Makenley County, Oklahoma County			
2. Ivana Alcantar	Allen County, Amelia County, Chippewa County, Gray Harbor County, McDowell County, Makenley County, Oklahoma County			
3. Galen Duremey	Allen County, Amelia County, Chippewa County, Lubbock County, McDowell County, Makenley County, Oklahoma County, Westcott			
4. Avery Dremer	Allen County, Amelia County, Chippewa County, Lubbock County, McDowell County, Makenley County, Oklahoma County, Westcott			
5. Ted Dremer	Allen County, Amelia County, Chippewa County, Lubbock County, McDowell County, Makenley County, Oklahoma County, Westcott			
6. Marcella Etman	Allen County, Amelia County, Chippewa County, Gray Harbor County, McDowell County, Makenley County, Oklahoma County, Westcott			
7. Radford Atherton	Allen County, Amelia County, Chippewa County, Gray Harbor County, McDowell County, Makenley County, Oklahoma County			
8. Tom Blodgett	Allen County, Amelia County, Chippewa County, Gray Harbor County, McDowell County, Makenley County, Oklahoma County			
9. Stinkly Denett	Allen County, Amelia County, Chippewa County, Gray Harbor County, McDowell County, Makenley County, Oklahoma County			
10. Shirley Denett	Allen County, Amelia County, Chippewa County, Gray Harbor County, McDowell County, Makenley County, Oklahoma County			

Below the results grid, a message says 'Query executed successfully.' At the bottom of the screen, the system status bar shows 'AHMAD-F15 (13.0 RTM) AHMAD-F15\ahmad (62) TargetCompetitorTest 00:00:18 10 rows' and the date '12/14/2021'.

Comment: here we get the nearby stores for each customer.

Function Testing:

`SELECT dbo.ToRadius(93)`

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like tables, stored procedures, and functions. The main query window contains the following SQL code:

```
SELECT * FROM CustomerRatings
SELECT dbo.AverageProductScore(104)
SELECT dbo.ToRadius(93)
SELECT dbo.CustomerNearbyStores(35,1000)
SELECT dbo.DistanceOfTwoPoint(34.75,-111.77,43.29,-72)
```

The results pane shows a single row of output from the `dbo.ToRadius` call:

(No column name)
1.6231562044

At the bottom, a status bar indicates "Query executed successfully." The system tray at the bottom of the screen shows the date and time as 12/14/2021 11:07 PM.

Comment: by entering the 93 we get 1.6231562044. the output of this function will be used to calculate the distance between two points on earth.

```
SELECT dbo.DistanceOfTwoPoint(34.75, -111.77, 43.29, -72)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays a tree view of database objects under the schema 'dbo'. In the center, the Query Editor pane contains the following SQL code:

```
SELECT dbo.AverageProductScore(104)
SELECT dbo.ToRadius(93)
SELECT dbo.CustomerNearbyStores(35,1000)
SELECT dbo.DistanceOfTwoPoint(34.75, -111.77, 43.29, -72)
```

The results pane below the query editor shows a single row of output:

(No column name)
2190.8469958640

At the bottom of the screen, the taskbar shows the system status: AHMAD-F15 (15.0 RTM) | AHMAD-F15\ahmads (62) | TargetCompetitorTest | 00:00:00 | 1 rows.

Comment: by entering (34.75, -111.77, 43.29, -72) we get 2190.84.

```
SELECT dbo.AverageProductScore(104)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery4.sql - AHMAD-F15.TargetCompetitorTest (AHMAD-F15.usus (62)) - Microsoft SQL Server Management Studio". The main area displays a query window with the following T-SQL code:

```
SELECT * FROM CustomerRatings
SELECT dbo.AverageProductScore(104)
SELECT dbo.ToRadius(93)
SELECT dbo.CustomerNearbyStores(35,1000)
SELECT dbo.DistanceOfTwoPoint(34.75, -111.77, 43.29, -72)
```

The results pane shows a single row with value 150. The object explorer on the left lists various database objects such as tables (products, customers, invoices, etc.), stored procedures, functions, and other system objects. The status bar at the bottom indicates "Query executed successfully." and "AHMAD-F15 (15.0 RTM) AHMAD-F15.usus (62) TargetCompetitorTest 00:00:00 1 rows".

Comment: by entering 104 as product ID we get the average score of 1.5

```
SELECT dbo.CustomerNearbyStores(35,1000)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays a tree view of database objects including tables (e.g., dbo.CustomerRatings, dbo.Customers), stored procedures (e.g., dbo.CustomerNearbyStores, dbo.DistanceOfTwoPoint), and functions (e.g., dbo.AverageProductScore, dbo.ToRadius). The main query window on the right contains the following T-SQL code:

```
SELECT * FROM CustomerRatings
SELECT dbo.AverageProductScore(104)
SELECT dbo.ToRadius(93)
SELECT dbo.CustomerNearbyStores(35,1000)
SELECT dbo.DistanceOfTwoPoint(34.75,-111.77,43.29,-72)
```

The Results tab shows the output of the last query, which is a single row:

1	Allen County, Amelia County, Chippewa County, Grays Harbor County, McDowell County, McKinley County, Oklahoma County
1	Allen County, Amelia County, Chippewa County, Grays Harbor County, McDowell County, McKinley County, Oklahoma County

At the bottom of the screen, the taskbar shows the operating system environment, including the Start button, taskbar icons, and system status information like battery level, temperature, and date/time.

Comment: by entering the customer ID we get the stores nearby customers.

Trigger Testing:

```
INSERT AddressBooks VALUES (41, 'billing', 'wrong address', '00000')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, a query window displays the following SQL statement:

```
INSERT AddressBooks VALUES (41, 'billing', 'wrong address', '00000')
```

In the bottom right corner of the query window, there is an error message:

Msg 40010, Level 16, State 1, Procedure CustomerZipCodes, Line 11 [Batch Start line 15]
Invalid zipcode . zip code does not exist on database
Completion time: 2021-12-14T23:20:39.5805413+08:00

The status bar at the bottom of the screen shows the following information:

AHMAD-F15 (15.0 RTM) AHMAD-F15\ahmads (62) TargetCompetitorTest 200000 0 rows
6°C Haze 11:21 PM ENG 12/14/2021

Comment: the trigger cached a wrong zip code and threw “Invalid zipcode . zip code does not exist on database”

Conclusions:

creating a database for a sale business seems overwhelming but is a truly interesting task. How to create tables and organize them is the infrastructure which I really paid attention to. For the second step I taught about the real word requirement of such business and made a business plan and created and tested all the scripts which may be useful in these types of businesses.

This project was really helpful in reviewing all the materials we have learned during the semester

Appendix: screenshots from SQL Server for codes, testing, etc.

Part1:

```
USE [TargetCompetitorTest]
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85260, N'AZ',
N'Maricopa County', 33.61, -111.89)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85261, N'AZ',
N'Maricopa County', 33.49, -111.92)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85262, N'AZ',
N'Maricopa County', 33.84, -111.81)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85263, N'AZ',
N'Maricopa County', 33.73, -111.68)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85264, N'AZ',
N'Maricopa County', 33.75, -111.57)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85266, N'AZ',
N'Maricopa County', 33.77, -111.92)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85267, N'AZ',
N'Maricopa County', 33.61, -111.89)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85268, N'AZ',
N'Maricopa County', 33.6, -111.74)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85269, N'AZ',
N'Maricopa County', 33.6, -111.71)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85271, N'AZ',
N'Maricopa County', 33.46, -111.91)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85272, N'AZ', N'Pinal
County', 32.85, -111.97)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85273, N'AZ', N'Pinal
County', 33.28, -111.11)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85274, N'AZ',
N'Maricopa County', 33.38, -111.87)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85275, N'AZ',
N'Maricopa County', 33.45, -111.76)
GO
INSERT [dbo].[ZipCodes] ([ZipCode], [State], [County], [Latitude], [Longitude]) VALUES (85277, N'AZ',
N'Maricopa County', 33.45, -111.72)
GO
.
.
.
```

You can find all the codes in the files attached to this report
Part 2: table verification

```
SELECT * FROM Prd.ProductCategories
```

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62))- Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Object Explorer Task List Home Back Forward Refresh Stop Change Type

TargetCompetitorTest

Execute Data insertion.sql (L..6AC2GB\elham (59)) SQLQuery3.sql (L..6AC2GB\elham (70))

Data insertion.sql (L..6AC2GB\elham (59))
, [44, 'Shipping', '43487 Sunnyside Parkway', '85272')
GO
USE TargetCompetitorTest
GO
INSERT Prd.ProductCategories VALUES('Home & Kitchen')
, ('Sports & Outdoors')
, ('Toys & Games')
, ('Beauty & Personal Care')
, ('Health, Household & Baby Care')
, ('Kitchen & Dining')
, ('Office Products')
, ('Garden & Outdoor')
, ('Tools & Home Improvement')
, ('Pet Supplies')

SELECT * FROM Prd.ProductCategories

DELETE FROM AddressBooks

100 %

Results Messages

ProductCategoryId	Name
1	Home & Kitchen
2	Sports & Outdoors
3	Toys & Games
4	Beauty & Personal Care
5	Health, Household & Baby Care
6	Kitchen & Dining
7	Office Products
8	Garden & Outdoor
9	Tools & Home Improvement
10	Pet Supplies

Query executed successfully.

(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 10 rows

Ready

Ln 56 Col 1 Ch 1 INS

56° F 10:01 PM

ENG 12/11/2021

SELECT * FROM Customers

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62))- Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Recent Objects Task List Object Explorer

TargetCompetitorTest

Execute

Change Type

Quick Launch (Ctrl+Q)

Object Explorer

Connect

dbo.Customers
dbo.InvoiceItems
dbo.InvoicePayments
dbo.Invoices
dbo.OrderDetails
dbo.OrderPayments
dbo.Orders
dbo.Roles
dbo.Shippings
dbo.Stores
dbo.StoreSuppliers
dbo.Suppliers
dbo.UserRoles
dbo.Users
dbo.Warehouses
dbo.WishLists
Inv.StoreProducts
Inv.SupplierProducts
Inv.WarehouseProduct
Prd.ProductBrands
Prd.ProductCategories
Prd.ProductImages
Prd.ProductInformation
Prd.Products
Prd.ProductSubCategory
Prd.ProductTags
Prd.Tags

Views

External Resources

Synonyms

Programmability

Service Broker

Storage

Security

Server Objects

Ready

Type here to search

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62)) - Microsoft SQL Server Management Studio

SQLQuery3.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (70))

INSERT Prd.ProductCategories VALUES('Home & Kitchen')
('Sports & Outdoors')
('Toys & Games')
('Beauty & Personal Care')
('Health, Household & Baby Care')
('Kitchen & Dining')
('Office Products')
('Garden & Outdoor')
('Tools & Home Improvement')
('Pet Supplies')

SELECT * FROM Customers

DELETE FROM AddressBooks

100 %

Result Messages

CustomerID	Name	Username	Email	PasswordHash
1	Alair Solars	asolars0	asolars@webeden.co.uk	0x425ED63A470235F39953579424997
2	Irena Absalom	iabsalon1	iabsalon1@com	0xE10ADC3949B4594BBE56E05720F883E
3	Galen Dineyden	gdineyden2	gdineyden2@dropbox.com	0xE10ADC3949B4594BBE56E05720F883E
4	Avery Deschre	adeschre3	adeschre3@avervista.org	0xE10ADC3949B4594BBE56E05720F883E
5	Tedi Dreameer	tdreamer4	tdreamer4@album.net	0xE10ADC3949B4594BBE56E05720F883E
6	Miller O'Leary	moleary5	moleary5@blogtalkradio.com	0xE10ADC3949B4594BBE56E05720F883E
7	Marielle Elham	mellamani6	mellamani6@telegraph.co.uk	0xE10ADC3949B4594BBE56E05720F883E
8	Redford Atherton	ratherton7	ratherton7@abc.net.au	0xE10ADC3949B4594BBE56E05720F883E
9	Tomm Blodgett	tblodgett8	tblodgett8@tiny.cc	0xE10ADC3949B4594BBE56E05720F883E
10	Shirley Danett	sdanett9	sdanett9@apple.com	0xE10ADC3949B4594BBE56E05720F883E

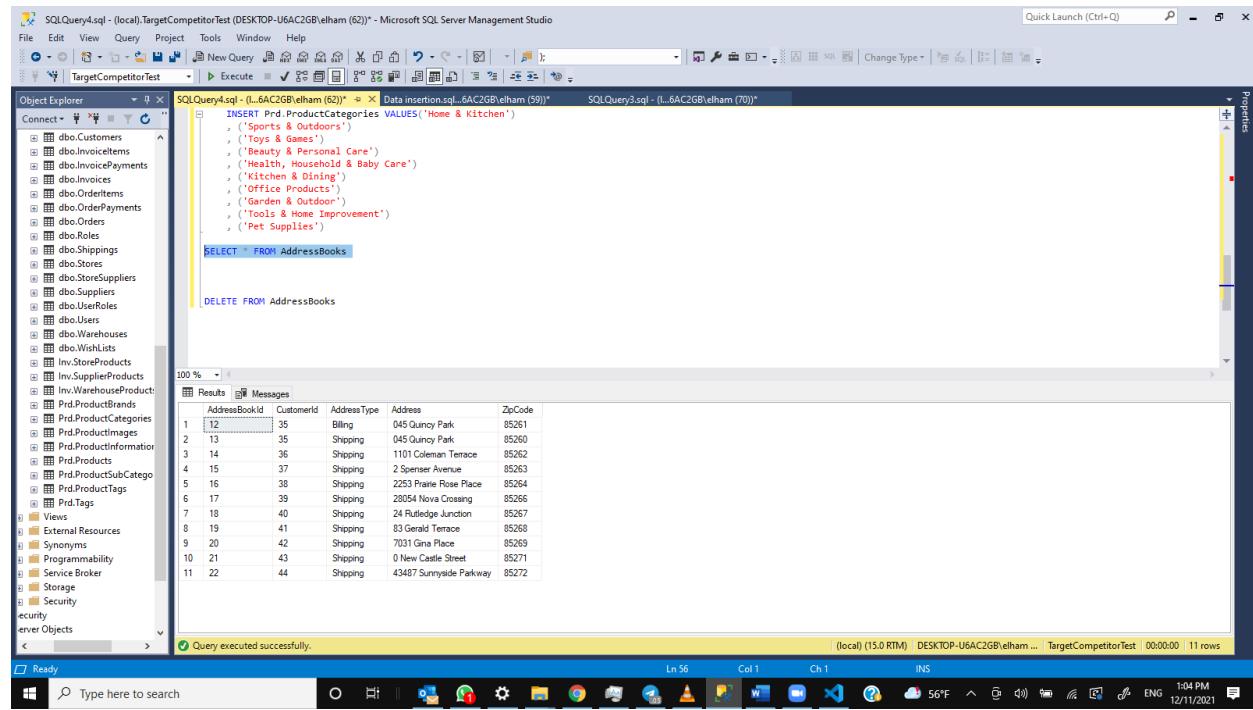
Query executed successfully.

(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 | 10 rows

Ln 56 Col 1 Ch 1 INS

10:03 PM 12/11/2021

SELECT * FROM AddressBooks



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Customers, Invoicetems, and OrderPayments. The main pane displays a query window with the following content:

```

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
... Execute ... SQLServer3 - (L..6AC2GB\elham (59))
[SQLQuery4.sql - (L..6AC2GB\elham (62))] [Data insertion.sql...6AC2GB\elham (59)] [SQLQuery3.sql - (L..6AC2GB\elham (70))]
[...]
INSERT INTO ProductCategories VALUES('Home & Kitchen')
    ('Sports & Outdoors')
    ('Toys & Games')
    ('Beauty & Personal Care')
    ('Health, Household & Baby Care')
    ('Kitchen & Dining')
    ('Office Products')
    ('Garden & Outdoor')
    ('Tools & Home Improvement')
    ('Pet Supplies')

SELECT * FROM AddressBooks

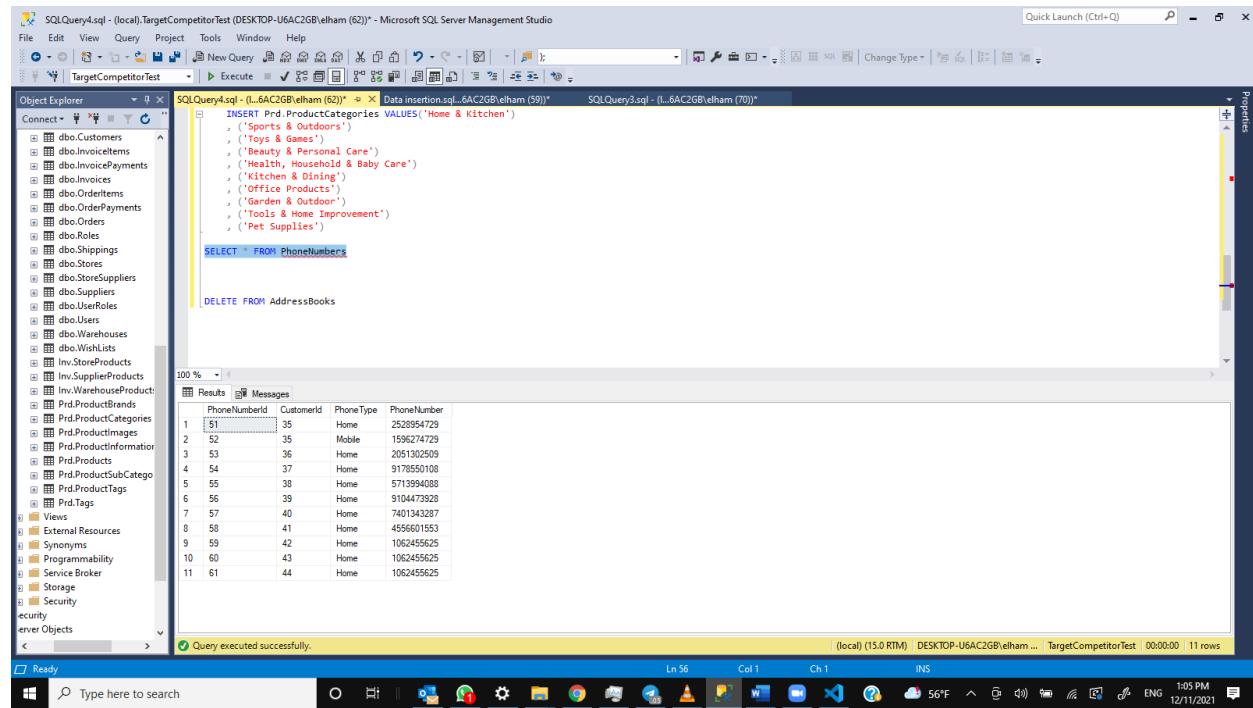
DELETE FROM AddressBooks

100 %
Results Messages
AddressBookId CustomerId AddressType Address ZipCode
1 12 35 Billing 045 Quincy Park 85261
2 13 35 Shipping 045 Quincy Park 85260
3 14 36 Shipping 1101 Coleman Terrace 85262
4 15 37 Shipping 2 Spenser Avenue 85263
5 16 38 Shipping 2253 Prairie Rose Place 85264
6 17 39 Shipping 28054 Nova Crossing 85266
7 18 40 Shipping 24 Rutledge Junction 85267
8 19 41 Shipping 83 Gerald Terrace 85268
9 20 42 Shipping 7031 Gina Place 85269
10 21 43 Shipping 0 New Castle Street 85271
11 22 44 Shipping 43487 Sunnyside Parkway 85272

```

The status bar at the bottom indicates "Query executed successfully." and "11 rows".

SELECT * FROM PhoneNumbers



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Customers, Invoicetems, and OrderPayments. The main pane displays a query window with the following content:

```

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
... Execute ... SQLServer3 - (L..6AC2GB\elham (59))
[SQLQuery4.sql - (L..6AC2GB\elham (62))] [Data insertion.sql...6AC2GB\elham (59)] [SQLQuery3.sql - (L..6AC2GB\elham (70))]
[...]
INSERT INTO ProductCategories VALUES('Home & Kitchen')
    ('Sports & Outdoors')
    ('Toys & Games')
    ('Beauty & Personal Care')
    ('Health, Household & Baby Care')
    ('Kitchen & Dining')
    ('Office Products')
    ('Garden & Outdoor')
    ('Tools & Home Improvement')
    ('Pet Supplies')

SELECT * FROM PhoneNumbers

DELETE FROM AddressBooks

100 %
Results Messages
PhoneNumberId CustomerId PhoneType PhoneNumber
1 51 35 Home 2528954729
2 52 35 Mobile 1596274729
3 53 36 Home 2051302509
4 54 37 Home 9178550108
5 55 38 Home 5713994088
6 56 39 Home 9104473928
7 57 40 Home 7401342287
8 58 41 Home 4556601553
9 59 42 Home 1062455625
10 60 43 Home 1062455625
11 61 44 Home 1062455625

```

The status bar at the bottom indicates "Query executed successfully." and "11 rows".

`SELECT * FROM prd.ProductSubCategories`

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Customers, Invoices, and OrderItems. The central Results grid displays the following data from the ProductSubCategories table:

ProductSubCategoryId	CategoryId	Name
1	1	Furniture & Décor
2	1	Kitchen & Dining
3	1	Bedding & Bath
4	1	Patio, Lawn, & Garden
5	2	Archery
6	2	Camping
7	2	Canoeing
8	2	Climbing
9	3	Board Games
10	3	Toy Foam Blasters & Guns
11	3	Electronic Learning & Education Toys
12	3	Kids' Electronics
13	4	Facial Skin Care Products
14	4	Body Skin Care Products
15	4	Eye Treatment Products
16	4	Bath & Bathing Accessories
17	5	Household Supplies
18	5	Health Care Products
19	5	Baby Care Products
20	5	Industrial & Scientific
21	6	Dining Room Furniture
22	6	Kitchen Utensils & Gadgets
23	6	Kitchen Small Appliances
24	6	Dining & Entertaining
25	7	Desk Accessories
26	7	Workspace Organizers
27	7	Paper & Printable Media
28	7	Filing Products

At the bottom, a message indicates "Query executed successfully." The status bar shows "(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 40 rows".

`SELECT * FROM prd.ProductBrands`

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects. The central Results grid displays the following data from the ProductBrands table:

ProductBrandId	Name
1	Apple
2	Nike
3	Dr. Pepper
4	Bridgestone
5	Nokia
6	Colombia
7	HP
8	Delonghi
9	Simense
10	Patagonia
11	Marmot
12	Columbia
13	The North Face
14	Starbucks
15	Budweiser
16	IKEA
17	Sony

At the bottom, a message indicates "Query executed successfully." The status bar shows "(local) (15.0 RTM) DESKTOP-U6AC2GB\elham ... TargetCompetitorTest 00:00:00 18 rows".

```
SELECT P.Name,C.Name,B.Name  
FROM prd.Products P  
JOIN prd.ProductSubCategories C ON P.ProductSubCategoryId=C.ProductSubCategoryId  
JOIN prd.ProductBrands B ON B.ProductBrandId=P.ProductBrandId
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to 'SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62)) - Microsoft SQL Server Management Studio'. The 'Object Explorer' pane on the left lists various database objects under 'dbo'. The 'SQL Query Editor' pane contains two queries: 'Data insertion.sql' and 'SQLQuery3.sql'. The results of the 'Data insertion.sql' query are displayed in a table, showing product details like name, price, and category. The results of the 'SQLQuery3.sql' query are also shown. A status bar at the bottom right shows the execution time as 00:00:00.7 rows.

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Open Recent Object Explorer Task List Properties

SQLQuery4.sql - (local).TargetCompetitorTest Data insertion.sql - (local).TargetCompetitorTest SQLQuery3.sql - (local).TargetCompetitorTest

Execute Save All Undo Redo Cut Copy Paste Find Replace Select All Clear All

Change Type

Object Explorer

Connect

dbo.Customers
dbo.invoiceItems
dbo.invoicePayments
dbo.invoices
dbo.orderDetails
dbo.orderItems
dbo.orderPayments
dbo.Orders
dbo.Roles
dbo.Shippings
dbo.Stores
dbo.StoreSuppliers
dbo.Suppliers
dbo.UserRoles
dbo.Users
dbo.Warehouses
dbo.WishLists
Inv.StoreProducts
Inv.SupplierProducts
Inv.WarehouseProduct
Prd.ProductBrands
Prd.ProductCategories
Prd.ProductImages
Prd.ProductInformation
Prd.Products
Prd.ProductSubCatego
Prd.ProductTags
Prd.Tags
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security

Server Objects

Ready

Type here to search

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (62))

Data insertion.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (59))

SQLQuery3.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (70))

INSERT prd.Products VALUES('Dog food',NULL,'\$18',3,40)
, ('Drilling tool box',NULL,'\$200',9,36)
, ('White barbecue set',NULL,'\$1000',16,32)
, ('A4 paper for HP13240',NULL,'\$5.56',7,27)
, ('Powerfull mixer 2',NULL,'\$58',8,23)
, ('Chocolate flavored multivitamin','Imported from italy','\$22',18,19)
, ('Climbing boots',NULL,'\$269',13,8)

SELECT P.Name, C.Name, B.Name
FROM prd.Products P
JOIN prd.ProductsSubCategories C ON P.ProductSubCategoryId=C.ProductSubCategoryId
JOIN prd.Brands B ON B.ProductBrandId=P.ProductBrandId

Results Messages

	Name	Name	Name
1	Dog food	Dog Bowls & Dishes	Dr. Pet
2	Drilling tool box	Tool Boxes	Simeise
3	White barbecue set	Patio Conversation Sets	IKEA
4	A4 paper for HP13240	Paper & Printable Media	HP
5	Powerfull mixer 2	Kitchen Small Appliances	Delongie
6	Chocolate flavored multivitamin	Baby Care Products	Nestle
7	Climbing boots	Climbing	The North Face

Query executed successfully.

(local) (15.0 RTM) | DESKTOP-U6AC2GB\elham ... | TargetCompetitorTest | 00:00:00 | 7 rows

Ln 131 Col 1 Ch 1 INS

2:41 PM 12/11/2021

```
SELECT * FROM prd.ProductImages
```

SQLQuery4.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (59)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
Object Explorer
Connect TargetCompetitorTest
File Edit View Query Project Tools Window Help
SQLQuery4.sql - (L..6AC2GB\elham (59)) Data insertion.sql - not connected*
GO
INSERT prd.ProductImages VALUES (101, 'https://cdn.shopify.com/s/files/1/0355/3833/7836/products/OpenFarm_Dry_Puppy_Food_Product_3.png?v=1609777401')
, (102, 'https://image.made-in-china.com/02f8j006TfLBCPngq/Wholesale-Custom-SGS-Household-Car-Repairing-Hand-Tool-Set.jpg?v=1574485326')
, (103, 'https://cdn.shopify.com/s/files/1/2593/7688/products/KT13RMH-ST_480x384.jpg?v=1574485326')
, (104, 'https://m.media-amazon.com/images/I/77i15Q8KNL.AC_SL1500_.Jpg')
, (105, 'https://m.media-amazon.com/images/I/7610yv2s0L.AC_SX466_.Jpg')
, (106, 'https://img.thrivemarket.com/store/ful/78/5/858892094307-1_L.Jpg')
, (107, 'https://11.wp.com/www.northwestalpineguides.com/wp-content/uploads/2018/04/la-sportiva-nepal-evo-gore-tex-mountaineering-boots.Yellow.01.jpg?fit=1146%2C1146&ssl=1')

USE TargetCompetitorTest
GO
SELECT * FROM prd.ProductImages

DELETE FROM AddressBooks

```

Results

ProductImageId	ProductId	ImageURL
1	1	https://cdn.shopify.com/s/files/1/0355/3833/7836/prod...
2	2	https://image.made-in-china.com/02f8j006TfLBCPngq...
3	3	https://cdn.shopify.com/s/files/1/2593/7688/products/...
4	4	https://m.media-amazon.com/images/I/77i15Q8KNL...
5	5	https://m.media-amazon.com/images/I/7610yv2s0L.AC_SX466...
6	6	https://img.thrivelmarket.com/store/ful/78/5/858892094307-1_L.Jpg
7	7	https://11.wp.com/www.northwestalpineguides.com/wp...

Query executed successfully.

[SELECT * FROM Orders](#)

Data insertion.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (60)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
Object Explorer
Connect TargetCompetitorTest
File Edit View Query Project Tools Window Help
Data insertion.sql - sq..6AC2GB\elham (60)* Data insertion.sql - not connected*
GO
INSERT Orders VALUES ('Returned', 'Online', 'CreditCard', 35, DEFAULT)
, ('Registered', 'Online', 'CreditCard', 37, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 38, DEFAULT)
, ('Paid', 'In-Store', 'CreditCard', 36, DEFAULT)
, ('Returned', 'In-Store', 'CreditCard', 39, DEFAULT)
, ('Paid', 'In-Store', 'Cash', 40, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 41, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 42, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 43, DEFAULT)
, ('Paid', 'Online', 'CreditCard', 44, DEFAULT)

SELECT * FROM Orders

(
    OrderId INT IDENTITY PRIMARY KEY,
    Status NVARCHAR(10) NOT NULL, --Returned,Returned,Paid...
    Type NVARCHAR(12) NOT NULL, --Online , In-Store , WarehouseReq , SupplierReq
    PaymentType NVARCHAR(10) NOT NULL, --Cash, CreditCard
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerId),
    OrderDate DATETIME NOT NULL DEFAULT GETDATE()
)
```

Results

OrderId	Status	Type	PaymentType	CustomerId	OrderDate
1	Returned	Online	CreditCard	35	2021-12-13 00:27:45.293
2	Registered	Online	CreditCard	37	2021-12-13 00:27:45.293
3	Paid	Online	CreditCard	38	2021-12-13 00:27:45.293
4	Paid	In-Store	CreditCard	36	2021-12-13 00:27:45.293
5	Returned	In-Store	CreditCard	39	2021-12-13 00:27:45.293
6	Paid	In-Store	Cash	40	2021-12-13 00:27:45.293
7	Paid	Online	CreditCard	41	2021-12-13 00:27:45.293
8	Paid	Online	CreditCard	42	2021-12-13 00:27:45.293
9	Paid	Online	CreditCard	43	2021-12-13 00:27:45.293
10	Paid	Online	CreditCard	44	2021-12-13 00:27:45.293

Query executed successfully.

[SELECT * FROM Stores](#)

Microsoft SQL Server Management Studio - Data insertion.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (53)) - Microsoft SQL Server Management Studio

```

USE TargetCompetitorTest
GO
INSERT Stores VALUES('Westcott', '504 Westcott Ave', '13210', '9143875429', '9144875429', 'Westcott303@TCT.com', 'TCT.com/Westcott303')
,'McKinley County', '504 McKinley County', '87302', '5931547862', '5931547863', '594McKinleyCounty@TCT.com', 'TCT.com/594McKinleyCounty'
,'Lubbock County', '928 Lubbock County', '13218', '29124988', '9318524987', '928LubbockCounty@TCT.com', 'TCT.com/928LubbockCounty'
,'Amelia County', '951 Amelia County', '23822', '7852147963', '7852147964', '951AmeliaCounty@TCT.com', 'TCT.com/951AmeliaCounty'
,'Oklahoma County', '630 Oklahoma County', '73127', '28749', '2951753964', '630OklahomaCounty@TCT.com', 'TCT.com/630OklahomaCounty'
,'McDowell County', '523 McDowell County', '28749', '4159785248', '4159785249', '523McDowellCounty@TCT.com', 'TCT.com/523McDowellCounty'
,'Allen County', '234 Allen County', '46861', '28749', '2951753964', '234AllenCounty@TCT.com', 'TCT.com/234AllenCounty'
,'Chippewa County', '214 Chippewa County', '54748', '145588765', '145588766', '214ChippewaCounty@TCT.com', 'TCT.com/214ChippewaCounty'
,'Grays Harbor County', '666 Grays Harbor County', '98557', '5486248918', '5486248919', '666GraysHarborCounty@TCT.com', 'TCT.com/666GraysHarborCounty'
SELECT * FROM Stores
(
    StoreId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(50) NOT NULL,
    Address NVARCHAR(100) NOT NULL,
    ZipCode NVARCHAR(10) NOT NULL,
    Phone NVARCHAR(10) NOT NULL,
    Fax NVARCHAR(10),
    Email NVARCHAR(100) NOT NULL,
)
```

Results

StoreId	Name	Address	ZipCode	Phone	Fax	Email	Webpage
1	Westcott	504 Westcott Ave	13210	9143875429	9144875429	Westcott303@TCT.com	TCT.com/Westcott303
2	McKinley County	504 McKinley County	87302	5931547862	5931547863	594McKinleyCounty@TCT.com	TCT.com/594McKinleyCounty
3	Lubbock County	928 Lubbock County	13218	29124988	9318524987	928LubbockCounty@TCT.com	TCT.com/928LubbockCounty
4	Amelia County	951 Amelia County	23822	7852147963	7852147964	951AmeliaCounty@TCT.com	TCT.com/951AmeliaCounty
5	Oklahoma County	630 Oklahoma County	73127	28749	2951753964	630OklahomaCounty@TCT.com	TCT.com/630OklahomaCounty
6	McDowell County	523 McDowell County	28749	4159785248	4159785249	523McDowellCounty@TCT.com	TCT.com/523McDowellCounty
7	Allen County	234 Allen County	46861	145588765	145588766	234AllenCounty@TCT.com	TCT.com/234AllenCounty
8	Chippewa County	214 Chippewa County	54748	145588765	145588766	214ChippewaCounty@TCT.com	TCT.com/214ChippewaCounty
9	Grays Harbor County	666 Grays Harbor County	98557	5486248918	5486248919	666GraysHarborCounty@TCT.com	TCT.com/666GraysHarborCounty

Query executed successfully.

[SELECT * FROM Suppliers](#)

Microsoft SQL Server Management Studio - Data insertion.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (53)) - Microsoft SQL Server Management Studio

```

USE TargetCompetitorTest
GO
USE TargetCompetitorTest
GO
INSERT Suppliers VALUES('GORDNER Ltd.', '859 Henry County', '8596743265', '8596743266', 'GORDNERLtd@gmail.com', 'GORDNERLtd.com')
,'Dougherty Ltd.', '526 Grundy County', '4966815448', '4966815448', 'DoughertyLtd@gmail.com', 'DoughertyLtd.com')
,'LLFord', '555 Sutter County', '7946133269', '7946133265', 'LLFord@gmail.com', 'LLFord.com')
SELECT * FROM Suppliers
(
    SupplierId INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(50) NOT NULL,
    Address NVARCHAR(100) NOT NULL,
    Phone NVARCHAR(10) NOT NULL,
    Fax NVARCHAR(10),
    Email NVARCHAR(100) NOT NULL,
    Webpage NVARCHAR(100)
)
```

Results

SupplierId	Name	Address	Phone	Fax	Email	Webpage
1	GORDNER Ltd.	859 Henry County	8596743265	8596743266	GORDNERLtd@gmail.com	GORDNERLtd.com
2	Dougherty Ltd.	526 Grundy County	4966815448	4966815448	DoughertyLtd@gmail.com	DoughertyLtd.com
3	LLFord	555 Sutter County	7946133269	7946133265	LLFord@gmail.com	LLFord.com

Query executed successfully.

[SELECT * FROM Warehouses](#)

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "Data insertion.sql - (local).TargetCompetitorTest (DESKTOP-U6AC2GB\elham (53)) - Microsoft SQL Server Management Studio" is open. The code in the window is:

```
GO
USE TargetCompetitorTest
GO
INSERT Warehouses VALUES(7,'888 Henry County','8596743123','8596743124','38242', 'GWH@gmail.com')
,(8,'852 Grundy County','4966815456','4966815459','50657', 'DWH@gmail.com')
,(9,'258 Sutter County','7946133789','7946133781','95674', 'LLWH@gmail.com')

SELECT * FROM Warehouses
```

The results pane shows the following data:

	WarehouseId	SupplierId	Address	Phone	Fax	ZipCode	Email
1	1	7	888 Henry County	8596743123	8596743124	38242	GWH@gmail.com
2	2	8	852 Grundy County	4966815456	4966815459	50657	DWH@gmail.com
3	3	9	258 Sutter County	7946133789	7946133781	95674	LLWH@gmail.com

At the bottom of the screen, the taskbar shows the following information: Ready, Type here to search, and system icons for File Explorer, Task View, Start, Taskbar settings, and Network. The system tray shows the date and time as 12/13/2021, 4:52 PM, with a battery level of 44%.