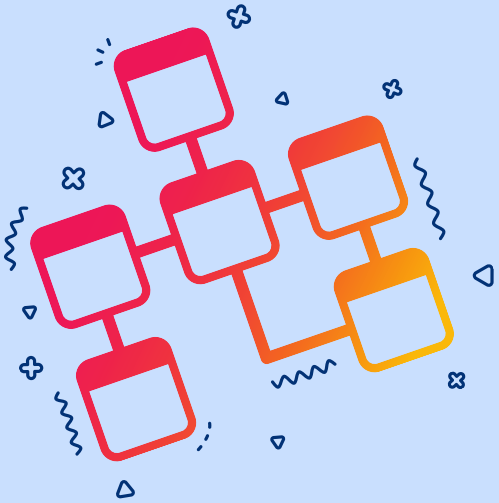
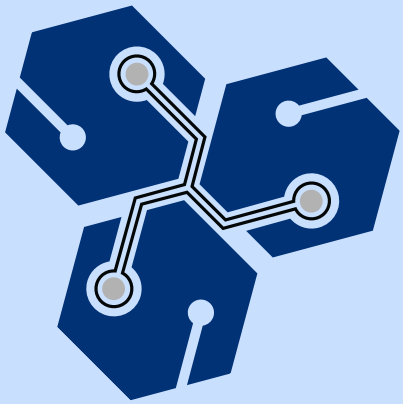


INFORMATION
MANAGEMENT RETRIEVAL
REPORT

Group Assignment:
Supermarket Information System



By - Group 77

INTRODUCTION

The scenario presented entails creating a system for a supermarket that allows the management parties to run and oversee the business operations of the supermarket.

This system is required to and is capable of restricting and controlling access via user authorization levels as well cover the stock management, billing and finance aspects of the business.

Furthermore, the solution is capable of advanced options such as providing financial reports and insights for management to make operational decisions regarding the business.

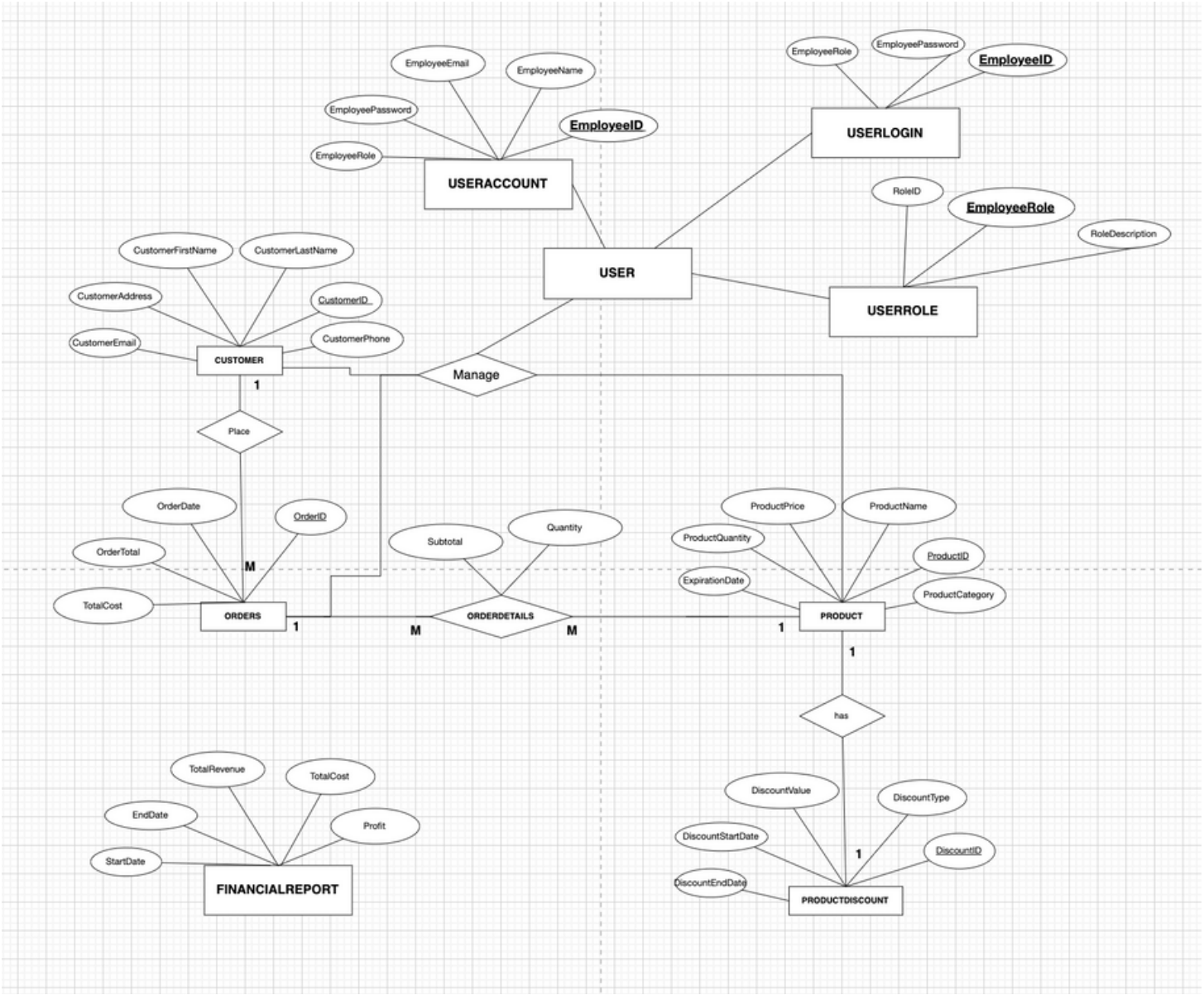
This database enables the staff to enter and maintain information about all of the store's activities. It also includes data validation mechanisms (constraints), triggers, and user-defined functions. Views, stored procedures, and user defined functions can be used by authorized parties to generate meaningful management reports and make strategic, long-term, and short-term managerial decisions for the supermarket.

The Supermarket database is intended to keep track of a retail store's products, customers, orders, and discounts. The Products table contains information about the store's products, such as the product ID, name, price, quantity, and category. Customers information, including contact information, can be found in the Customers table. The Orders table contains information about customer orders, such as the order ID, customer ID, order date, and total cost. The OrderDetails table contains information about the specific products ordered, as well as the quantity of each product in the order. These customer details are stored and can be used in the future for other purposes.

The ProductDiscounts table contains information about the discounts that are applied to products in the store, such as the discount ID, product ID, discount type, discount value, and the validity period. This database's schema also shows how these tables are related to one another via foreign keys. This also provides the ability for customers to receive special offers such as discounts.

Further details and information regarding this solution and its implementation is presented below in the rest of the report for perusal and review.

ER DIAGRAM


EmployeeEmail
EmployeeName
EmployeePassword
EmployeeID
EmployeeRole
USERACCOUNT
EmployeeRole
EmployeePassword
EmployeeID
USERLOGIN
RoleID
EmployeeRole
RoleDescription
USERROLE
CustomerFirstName
CustomerLastName
CustomerAddress
CustomerID
CustomerEmail
CustomerPhone
CUSTOMER
USER
Manage
1
Place
OrderDate
OrderID
OrderTotal
M
TotalCost
ORDERS
1
Subtotal
Quantity
M
ORDERDETAILS
M
ProductPrice
ProductName
ProductQuantity
ProductID
ExpirationDate
ProductCategory
PRODUCT
1
1
has
TotalRevenue
TotalCost
EndDate
Profit
StartDate
FINANCIALREPORT
DiscountValue
DiscountType
DiscountStartDate
1
DiscountID
DiscountEndDate
PRODUCTDISCOUNT

ER ASSUMPTIONS

- User is responsible for managing the customer, products and order and has user the details stored in UserAccounts, the authority level is stored in the UserRole (eg: admin, manager etc.) and the login details for each user is stored in UserLogin.

- The UserLogin table may be used to control access to the supermarket's systems, such as the point of sale system (not accounted for in this solution), inventory management system, and financial reporting system.

- The supermarket system to ensure that employees with different roles have the necessary access to perform their job functions, while maintaining the security of the system and data.

- Permissions are assumed, granular control over each individual permission is not accounted for. (Permission matrix is not included) for the admin login/access control component.

- Cost price is the buying/manufacturing cost of the products and product price is the selling price (with markup of profit margin). Therefore, product price should always be higher than cost price (to avoid loss).

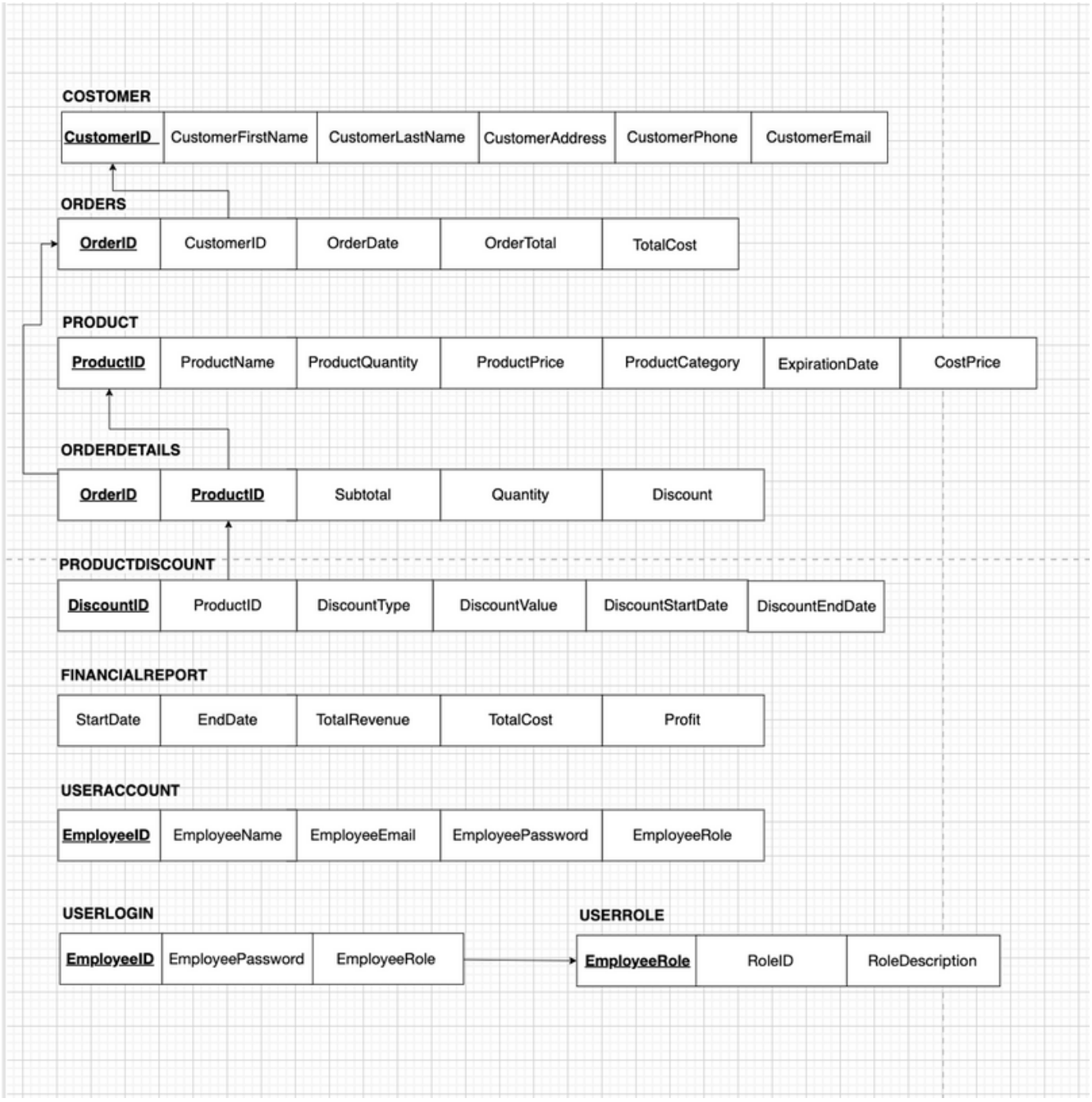
- The FinancialReport table may be updated on a regular basis, such as daily, weekly, or monthly, to track the financial performance of the supermarket over time.

- The solution doesn’t account for a customer loyalty card system.

- Assume that the supermarket is only in one location and there are no branches.

- Solution does not take employee payroll/salary into account for the financial performance aspect.

RELATIONAL MAP



DATA DICTIONARY

Table Name: Customers

Column Name	Data Type	Description
CustomerID	INT	Unique identifier for the customer
CustomerFirstName	NVARCHAR(255)	First name of the customer
CustomerLastName	NVARCHAR(255)	Last name of the customer
CustomerEmail	NVARCHAR(255)	Email address of the customer
CustomerAddress	NVARCHAR(255)	Address of the customer
CustomerPhone	NVARCHAR(255)	Phone number of the customer

Table Name: Products

Column Name	Data Type	Description
ProductID	INT	Unique identifier for the product
ProductName	NVARCHAR(255)	Name of the product
ProductPrice	DECIMAL(10, 2)	Selling price of the product
ProductQuantity	INT	Number of units of the product in stock
ProductCategory	NVARCHAR(255)	Category of the product
ExpirationDate	DATE	Expiry date of the product
CostPrice	DECIMAL(10, 2)	Cost price of the product

Table Name: Orders

Column Name	Data Type	Description
OrderID	INT	Unique identifier for the order
CustomerID	INT	Unique identifier for the customer who placed the order, foreign key referencing the 'Customers' table
OrderDate	DATE	Date on which the order was placed
OrderTotal	DECIMAL(10, 2)	Total of the order
TotalCost	INT	Total cost of the order for the supermarket

DATA DICTIONARY

Table Name: OrderDetails

Column Name	Data Type	Description
OrderID	INT	Unique identifier for the order, foreign key referencing the 'Orders' table
ProductID	INT	Unique identifier for the product, foreign key referencing the 'Products' table
Quantity	INT	Number of units of the product in the order
Subtotal	DECIMAL(10, 2)	Subtotal cost for the units of the product in the order
Discount	DECIMAL(10, 2)	Order Discounts

Table Name: ProductDiscounts

Column Name	Data Type	Description
DiscountID	INT	Unique identifier for the discount
ProductID	INT	Unique identifier for the product the discount applies to, foreign key referencing the 'Product' table
DiscountType	NVARCHAR(255)	Type of the discount, like "BOGO" for Buy one get one free or "Percentage" for percentage off
DiscountValue	DECIMAL(10,2)	Value of the discount, either a percentage or a fixed amount
DiscountStartDate	DATE	The date from which the discount starts
DiscountEndDate	DATE	The date on which the discount ends

Table Name: FinancialReport

Column Name	Data Type	Description
StartDate	date	The start date of the financial period being reported on.
EndDate	date	The end date of the financial period being reported on.
TotalRevenue	decimal(10,2)	The total amount of revenue generated during the financial period.
TotalCost	decimal(10,2)	The total amount of cost incurred during the financial period.
Profit	decimal(10,2)	The profit generated during the financial period (calculated as TotalRevenue - TotalCost).

DATA DICTIONARY

Table Name: UserAccount

Column Name	Data Type	Description
EmployeeID	INT	Primary key, unique identifier for each employee
EmployeeName	VARCHAR(50)	Employee's name
EmployeeEmail	VARCHAR(255)	Employee's email address
EmployeePassword	VARCHAR(255)	Employee's password
EmployeeRole	VARCHAR(20)	Employee's role (e.g. Manager, Employee)

Table Name: UserRole

Column Name	Data Type	Description
EmployeeRole	VARCHAR(20)	Employee's role, it is a primary key
RoleID	INT	Unique identifier for the role
RoleDescription	VARCHAR(255)	Description of the Role

Table Name: UserLogin

Column Name	Data Type	Description
EmployeeID	INT	Unique identifier for each employee
EmployeePassword	VARCHAR(255)	Employee's password, not null value
EmployeeRole	VARCHAR(20)	Employee's role, it references the UserRole table, not null value

CREATING TABLES AND CONSTRAINTS

```
-- Create the database
CREATE DATABASE Supermarket;
GO

USE Supermarket;
GO

-- Create a table for products
CREATE TABLE Products
(
   ProductID int PRIMARY KEY,
   ProductName nvarchar(255) NOT NULL,
   ProductPrice decimal(10, 2) NOT NULL,
   ProductQuantity int NOT NULL,
   ProductCategory nvarchar(255) NOT NULL
);

-- Create a table for customers
CREATE TABLE Customers
(
   CustomerID int PRIMARY KEY,
   CustomerFirstName nvarchar(255) NOT NULL,
   CustomerLastName nvarchar(255) NOT NULL,
   CustomerEmail nvarchar(255) NOT NULL,
   CustomerAddress nvarchar(255) NOT NULL,
   CustomerPhone nvarchar(255) NOT NULL
);

-- Create a table for orders
CREATE TABLE Orders
(
   OrderID int PRIMARY KEY,
   CustomerID int FOREIGN KEY REFERENCES Customers(CustomerID),
   OrderDate date NOT NULL,
   OrderTotal decimal(10, 2) NOT NULL
);

 -- Create a table for order details
 CREATE TABLE OrderDetails
 (
    OrderID int FOREIGN KEY REFERENCES Orders(OrderID),
    ProductID int FOREIGN KEY REFERENCES Products(ProductID),
    Quantity int NOT NULL,
    Subtotal decimal(10, 2) NOT NULL
 );
```

CREATING TABLES AND CONSTRAINTS

```
CREATE TABLE ProductDiscounts
(
   DiscountID int PRIMARY KEY,
   ProductID int FOREIGN KEY REFERENCES Products(ProductID),
   DiscountType nvarchar(255) NOT NULL,
   DiscountValue decimal(10,2) NOT NULL,
   DiscountStartDate date NOT NULL,
   DiscountEndDate date NOT NULL
);
```

```
CREATE TABLE FinancialReport
(
   StartDate DATE,
   EndDate DATE,
   TotalRevenue DECIMAL(10,2),
   TotalCost DECIMAL(10,2),
   Profit DECIMAL(10,2)
);
```

```
CREATE TABLE UserAccount
(
   EmployeeID INT PRIMARY KEY IDENTITY(1,1),
   EmployeeName VARCHAR(50) NOT NULL,
   EmployeeEmail VARCHAR(255) NOT NULL,
   EmployeePassword VARCHAR(255) NOT NULL,
   EmployeeRole VARCHAR(20) NOT NULL
);
```

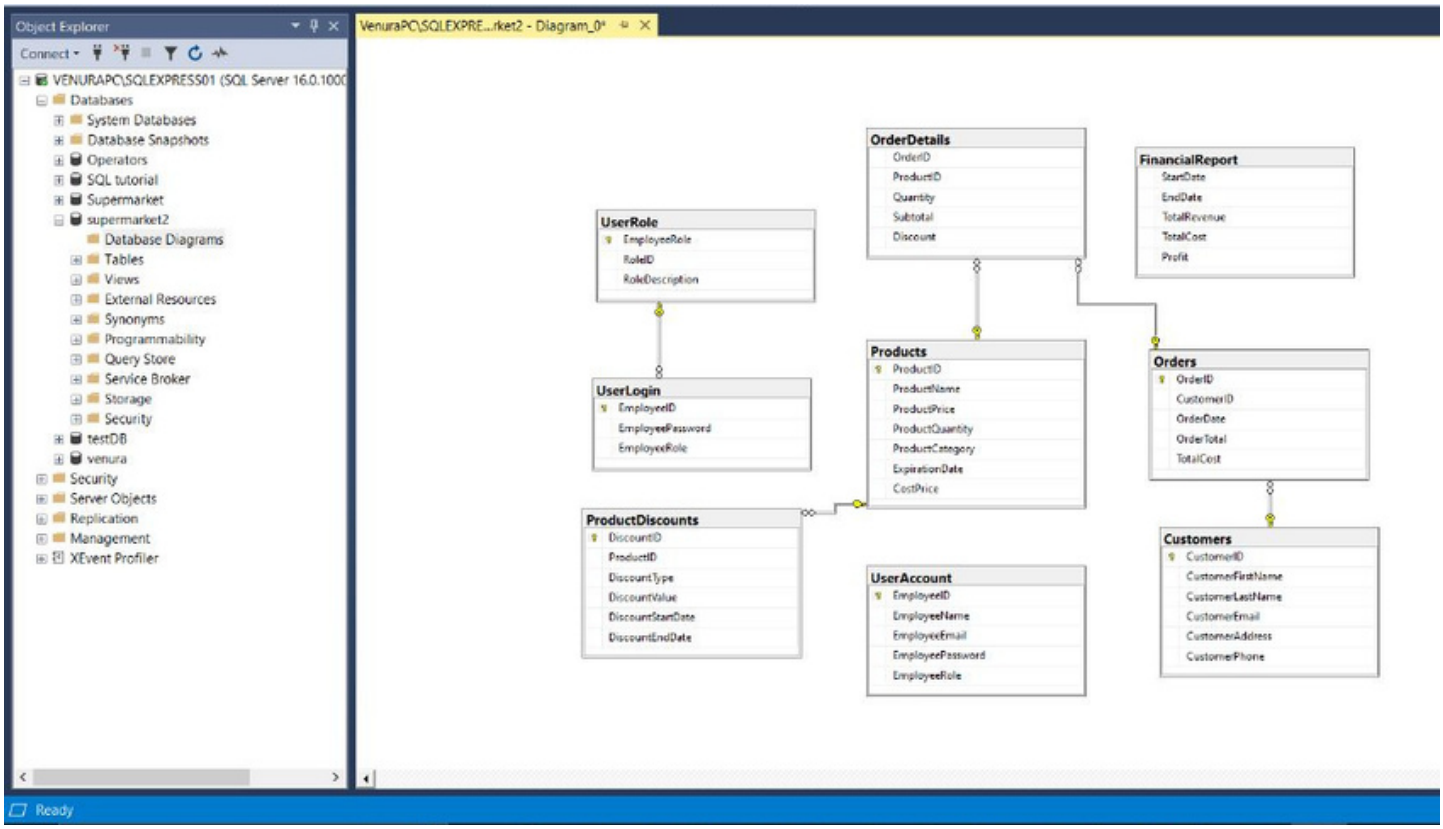
```
CREATE TABLE UserRole
(
      EmployeeRole VARCHAR(20) PRIMARY KEY,
      RoleID INT,
      RoleDescription VARCHAR(100)
);
```

```
CREATE TABLE UserLogin
(
     EmployeeID INT PRIMARY KEY IDENTITY (1,1),
     EmployeePassword VARCHAR(255) NOT NULL,
     EmployeeRole VARCHAR(20) FOREIGN KEY REFERENCES UserRole(EmployeeRole)
);
```

Please Note:

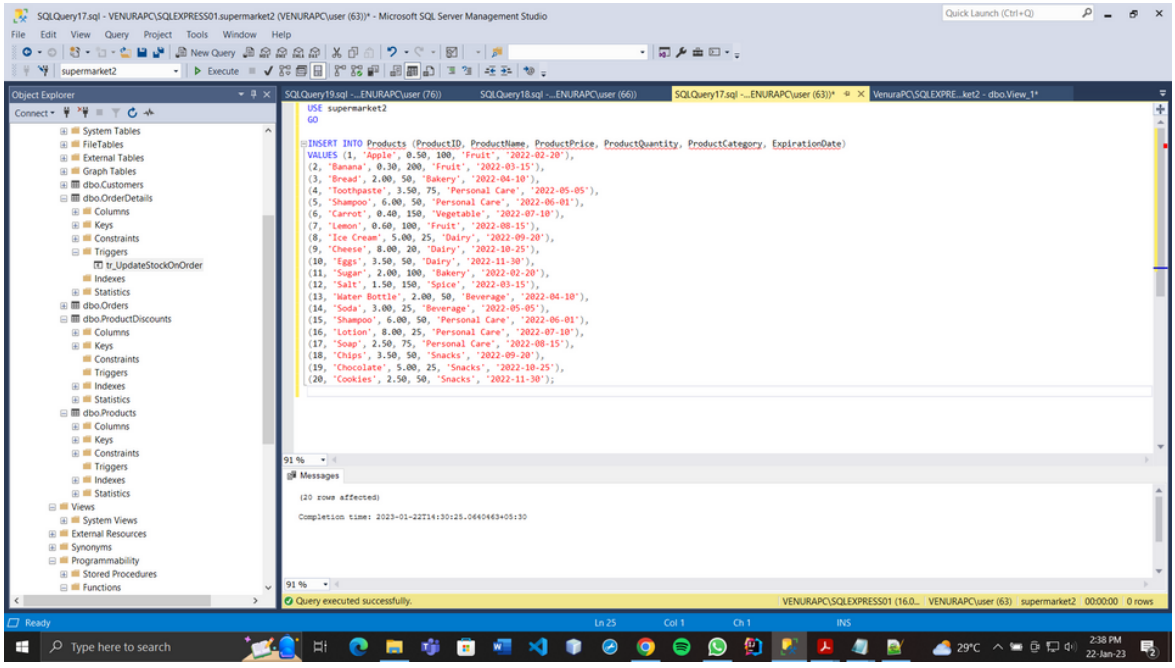
That these are the initial code snippets we used to build the database but on the process of building this database and some tables has been altered accordingly to move forward with the new triggers, user-defined functions, stored procedures and views .

DATABASE DIAGRAM

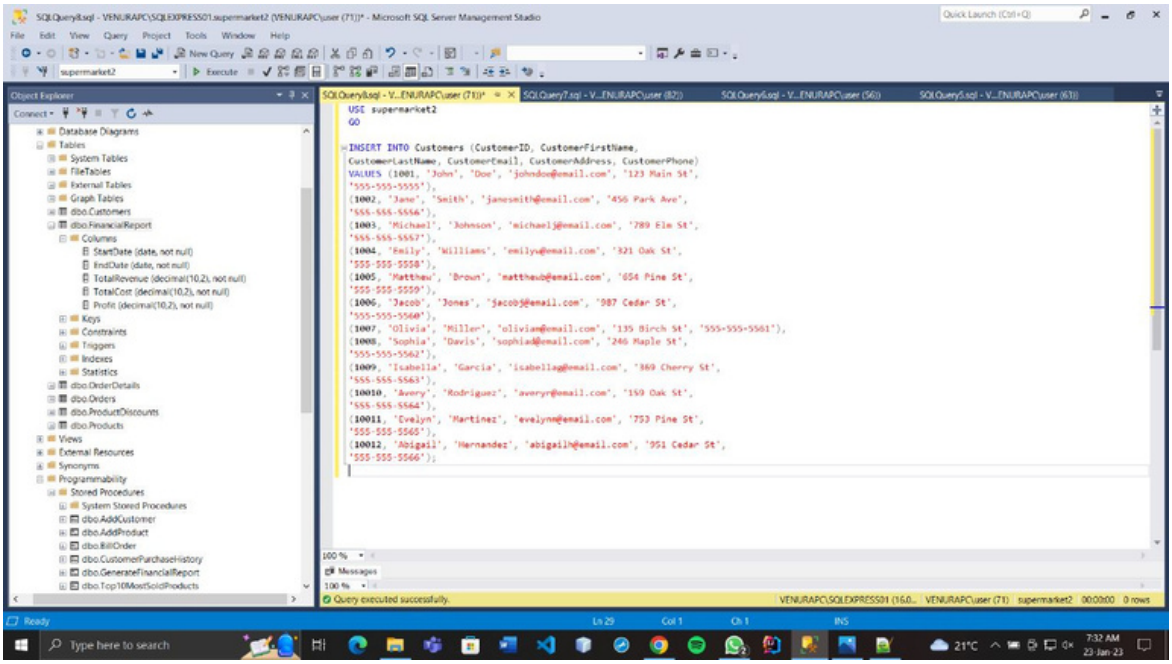


SAMPLE RECORDS

Data insertion for the table Products

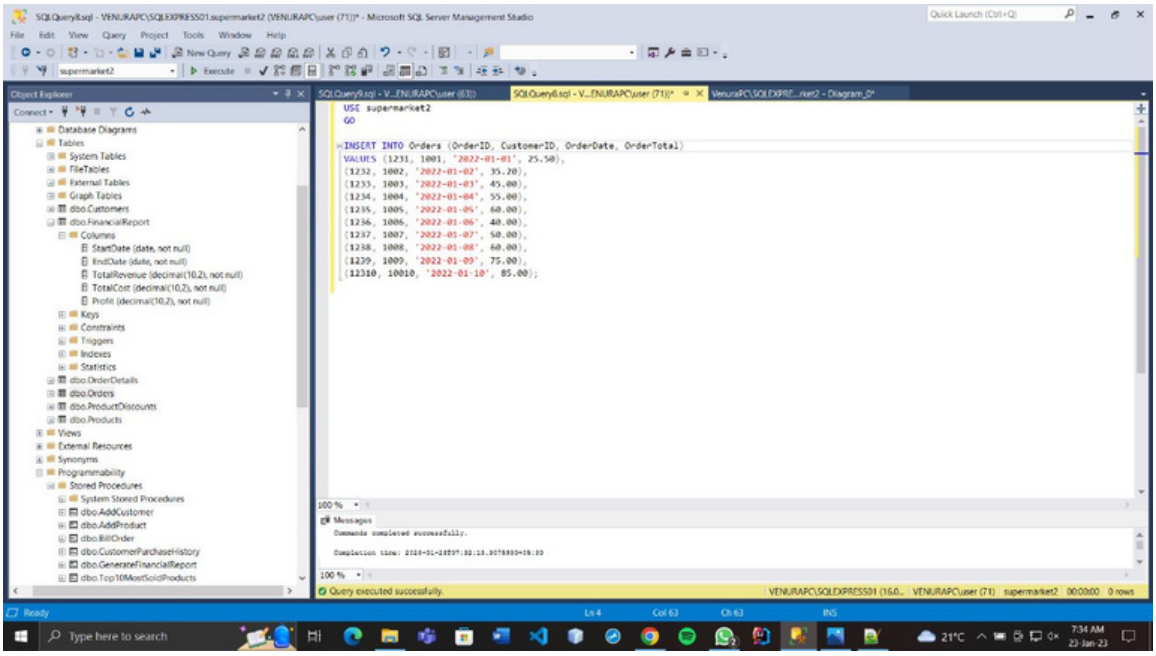


Data insertion for the table Products

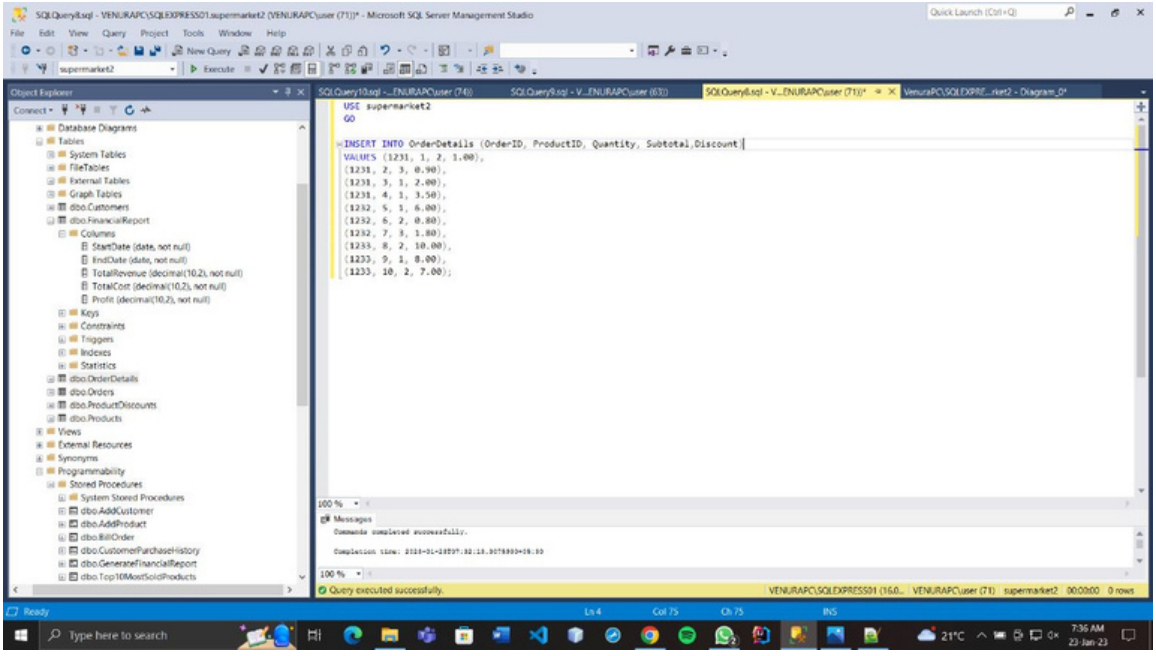


SAMPLE RECORDS

Data insertion for the table Orders

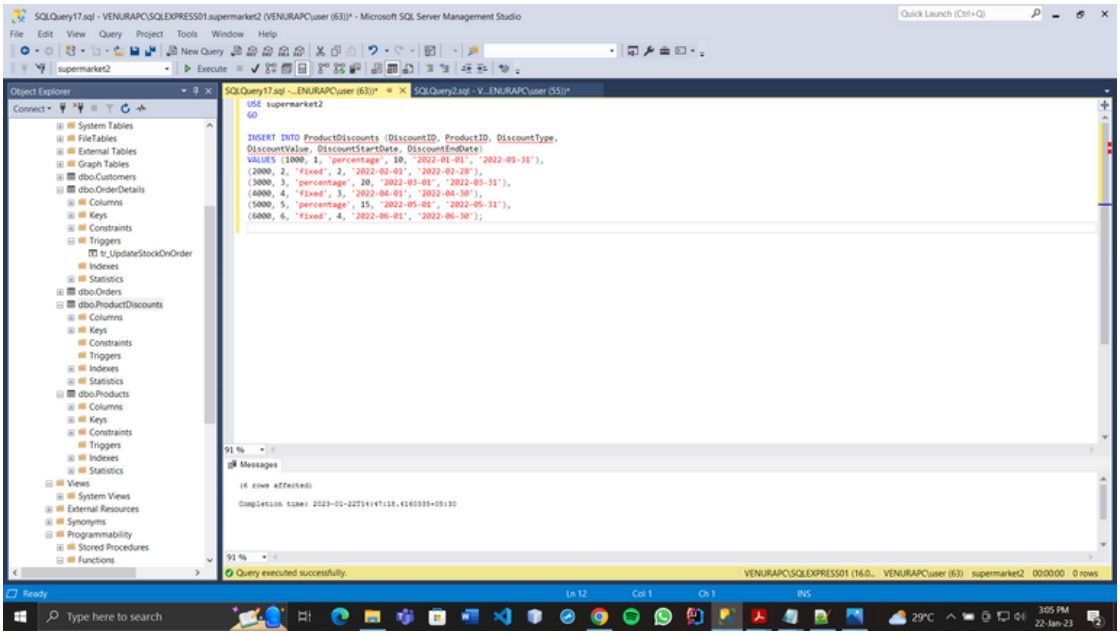


Data insertion for the table OrderDetails

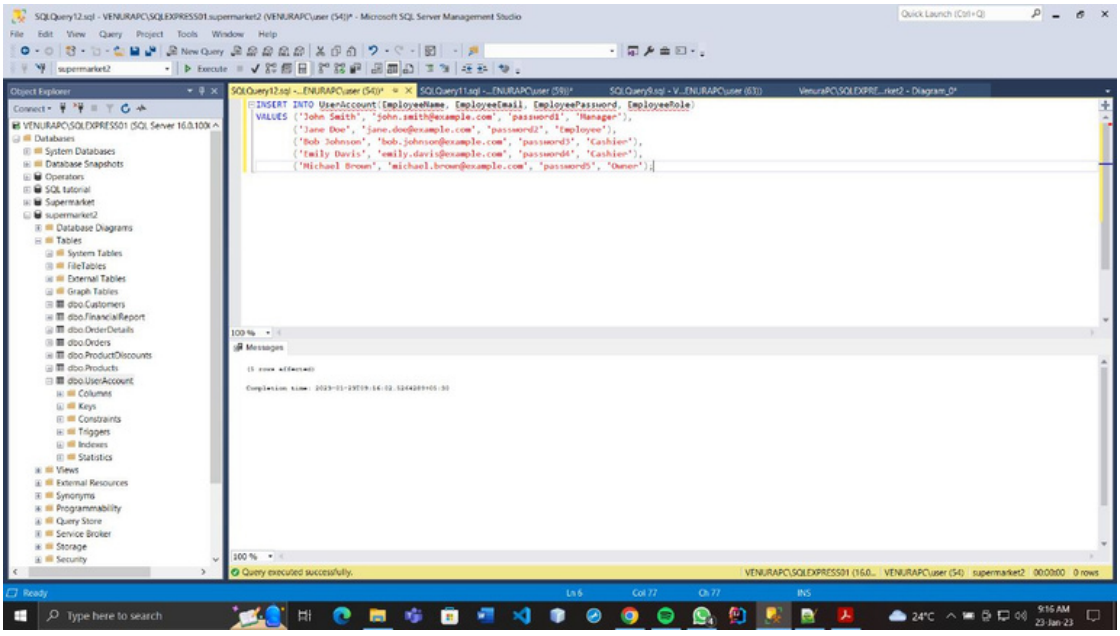


SAMPLE RECORDS

Data insertion for the table ProductDiscounts

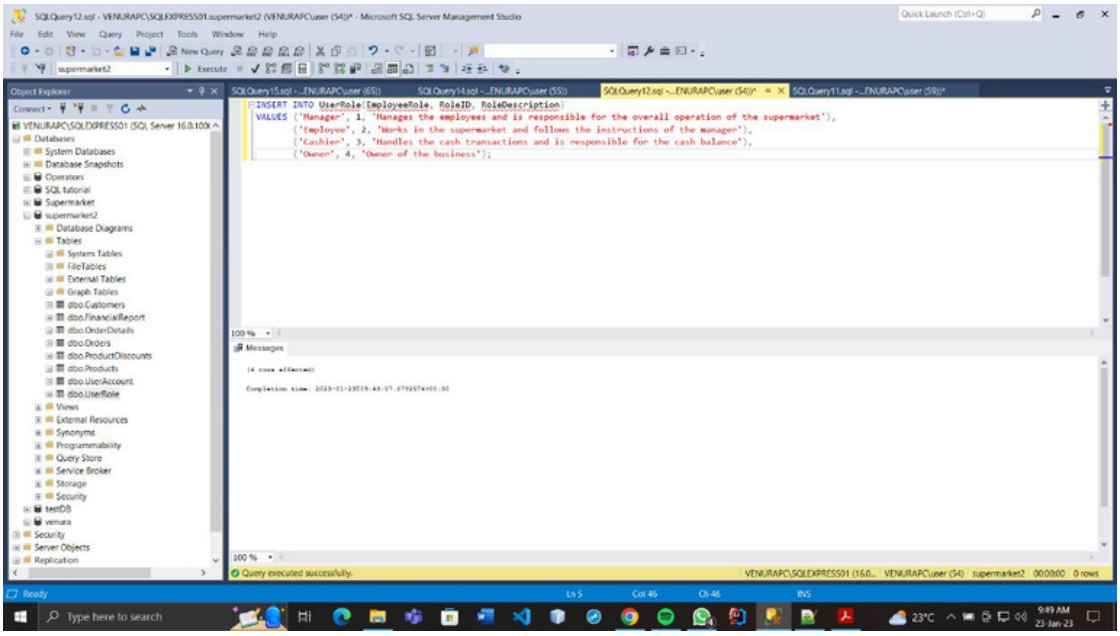


Data insertion for the table UserAccount

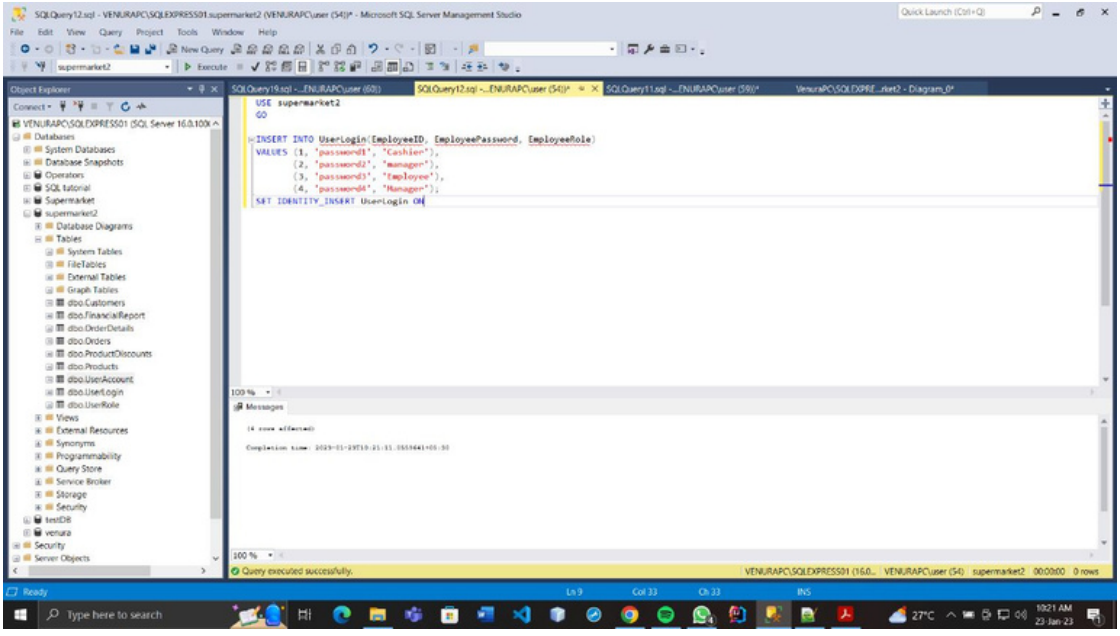


SAMPLE RECORDS

Data insertion for the table UserRole



Data insertion for the table UserLogin

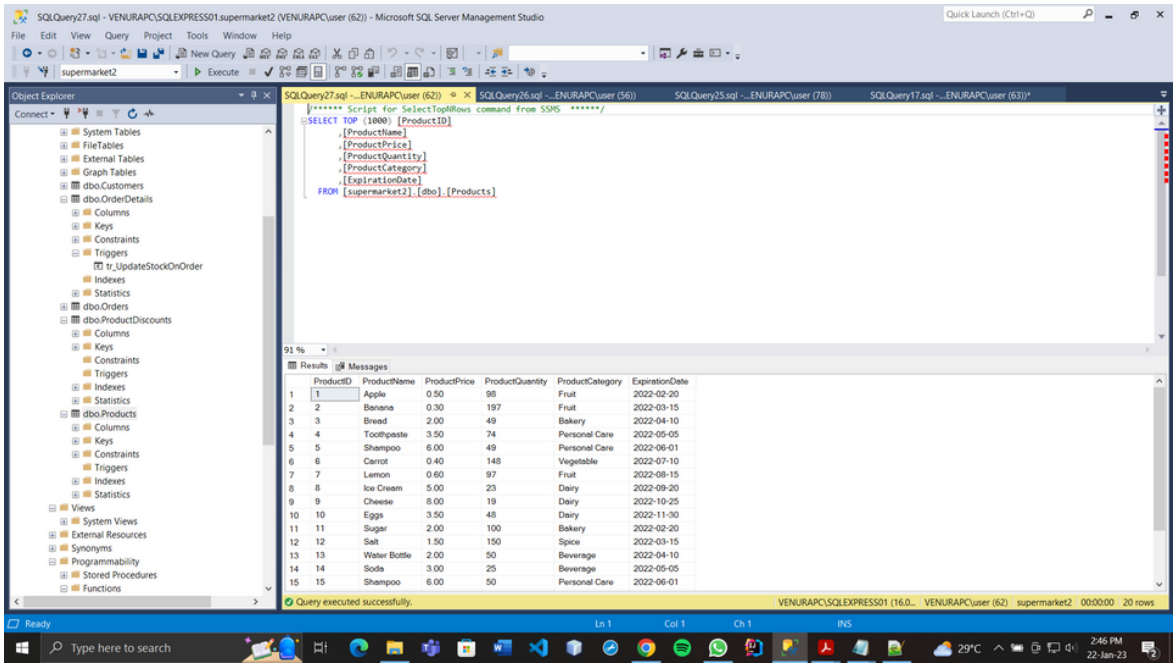


Please Note:

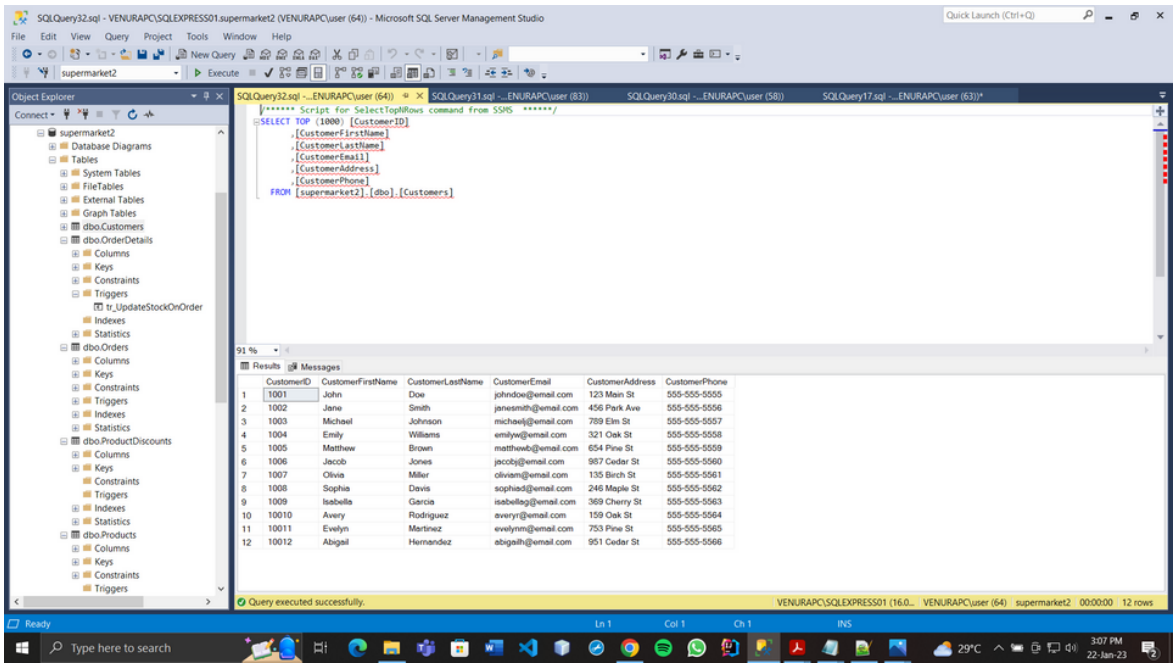
That these are the initial code snippets we used to build the database but on the process of building this database and some tables has been altered accordingly by inserting new data to move forward with the new triggers, user-defined functions, stored procedures and views .

ALL TABLES

Products Table

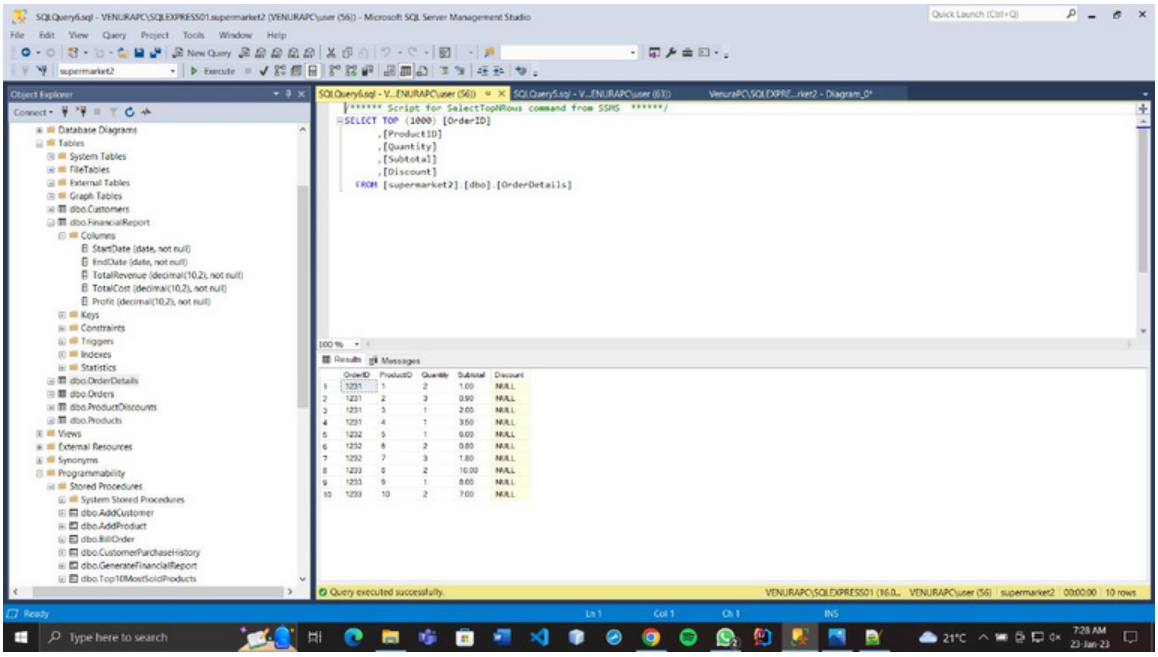


Customers Table

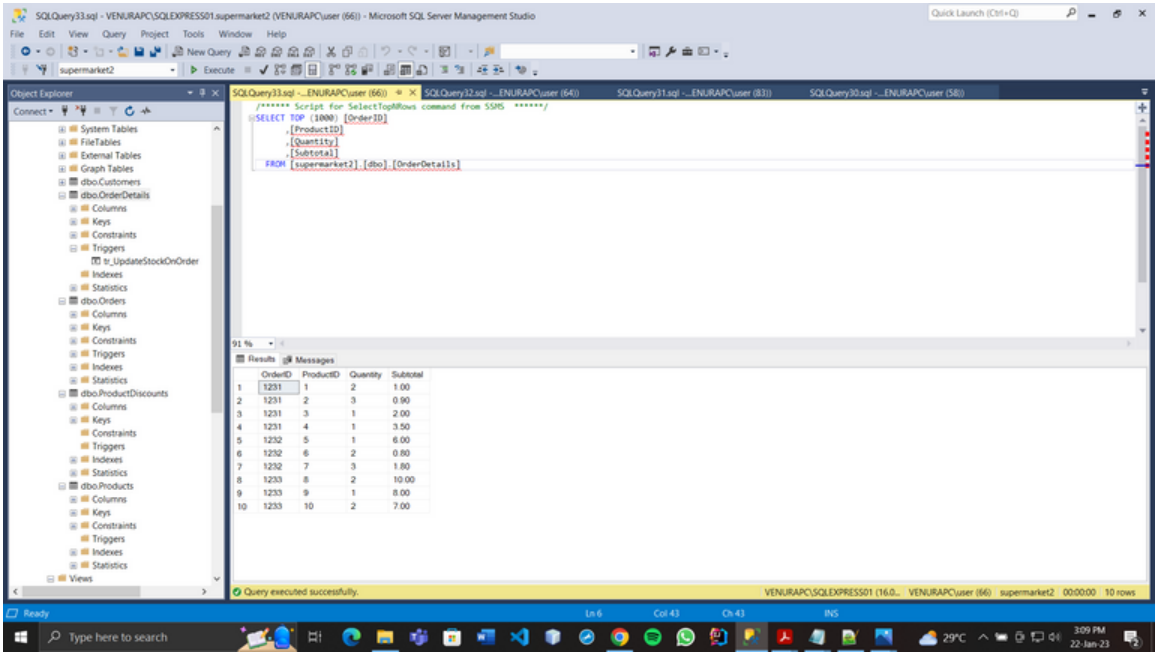


ALL TABLES

Orders Table

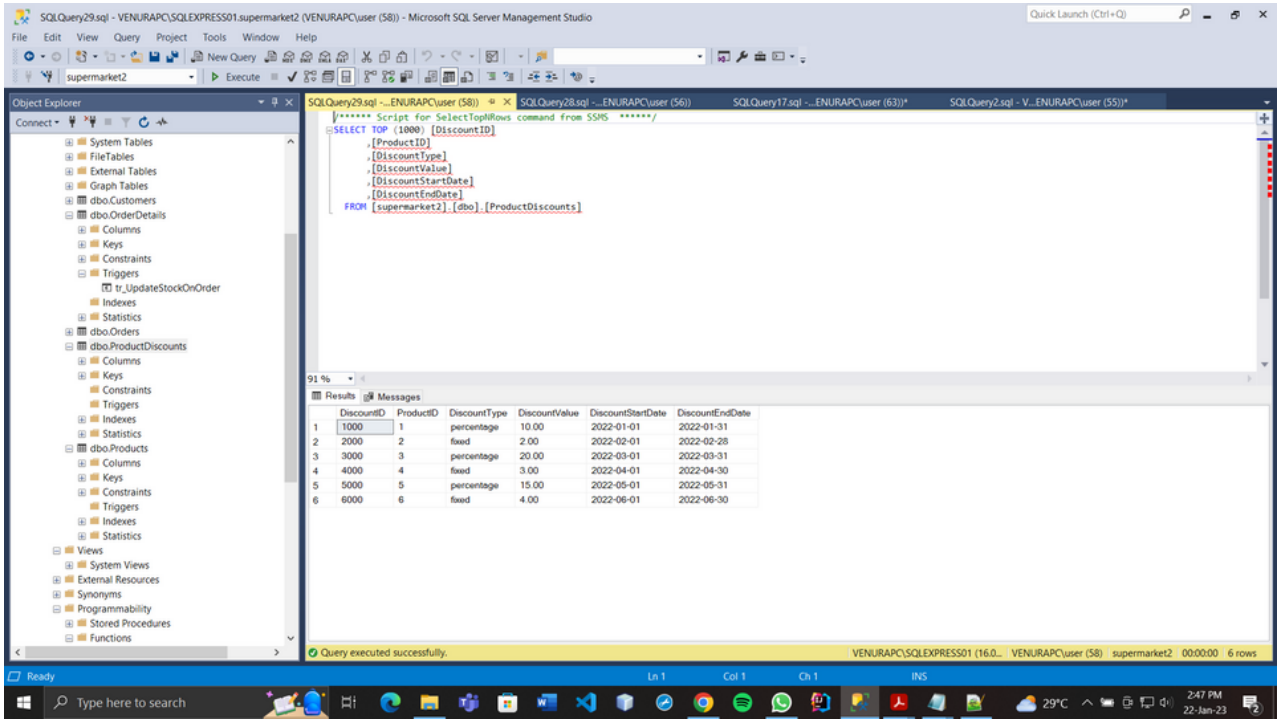


OrderDetails Table

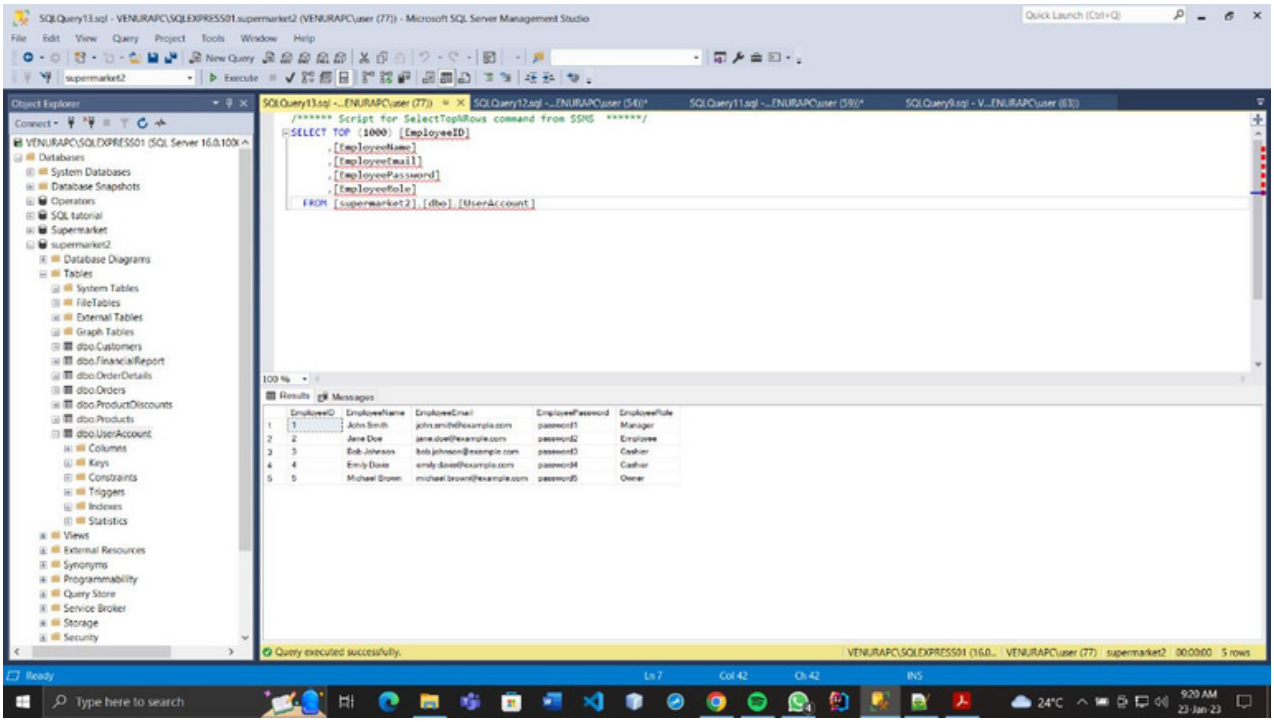


ALL TABLES

Product Discounts Table

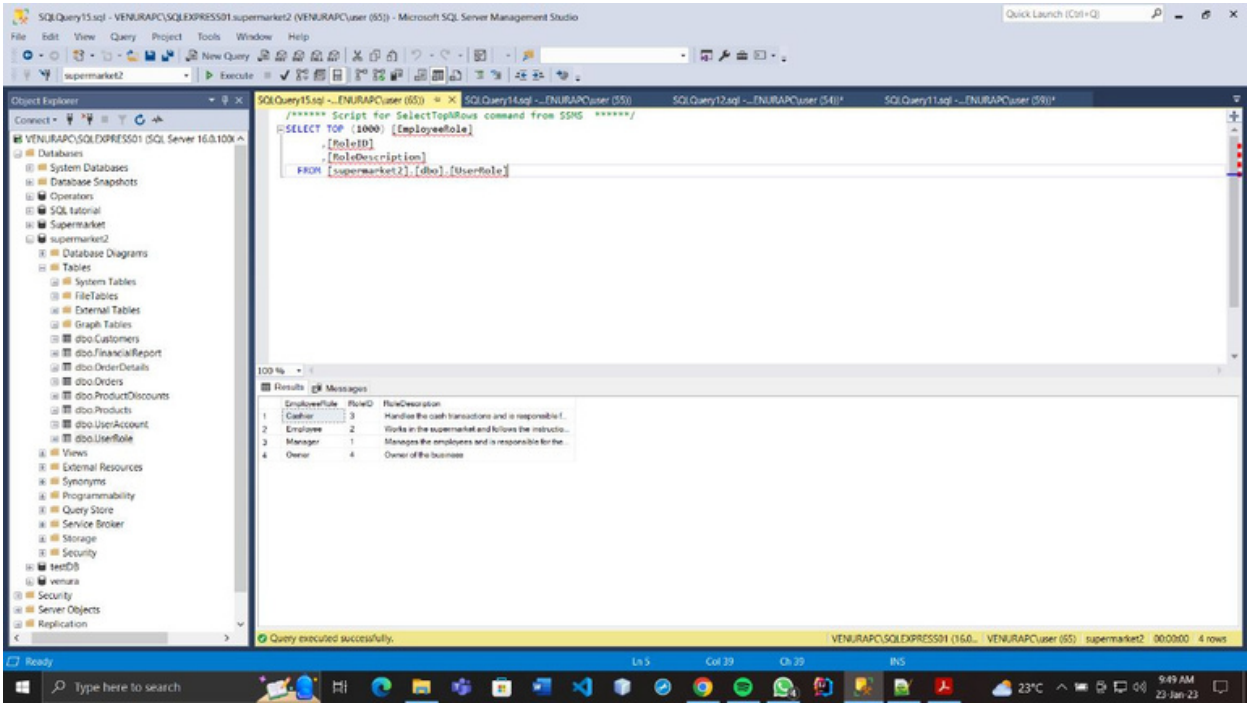


User Account Table

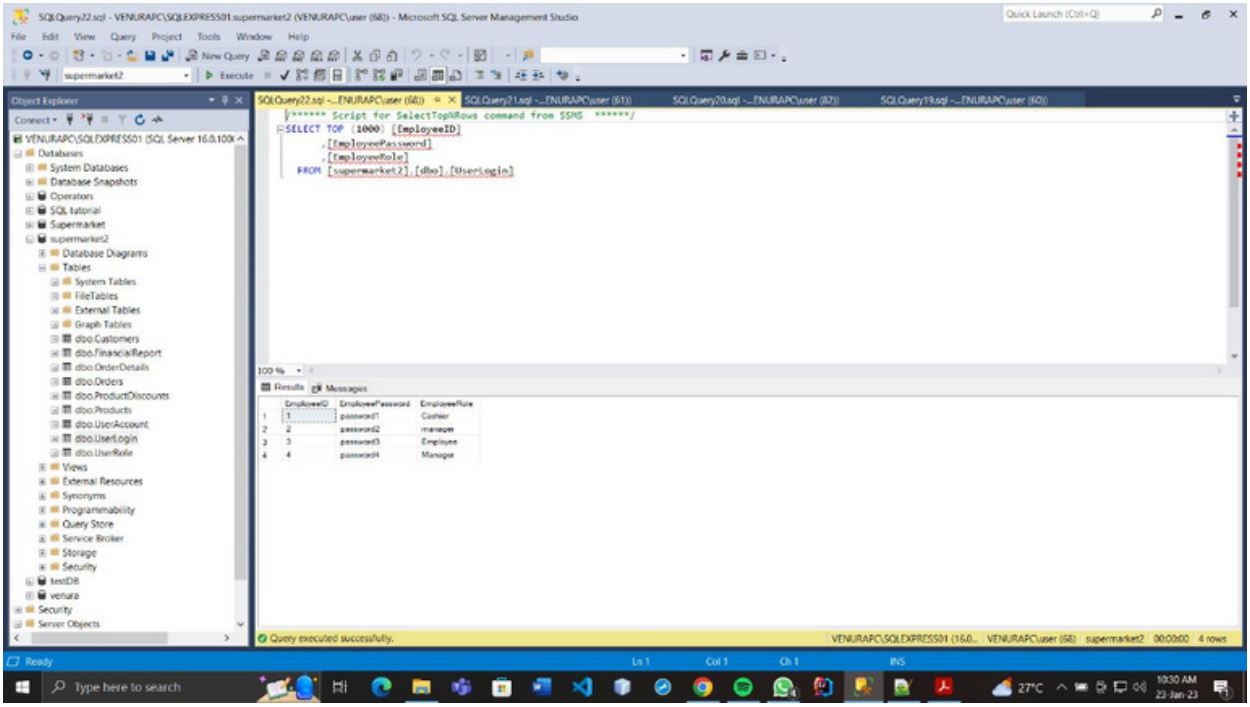


ALL TABLES

User Role Table

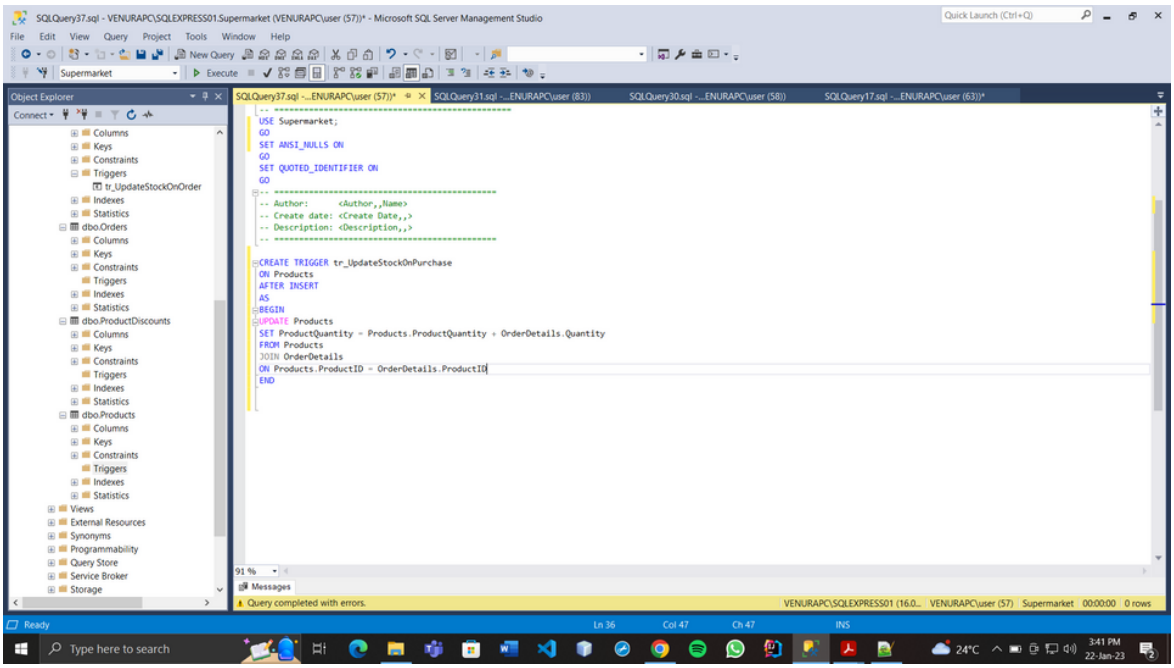


User Login Table

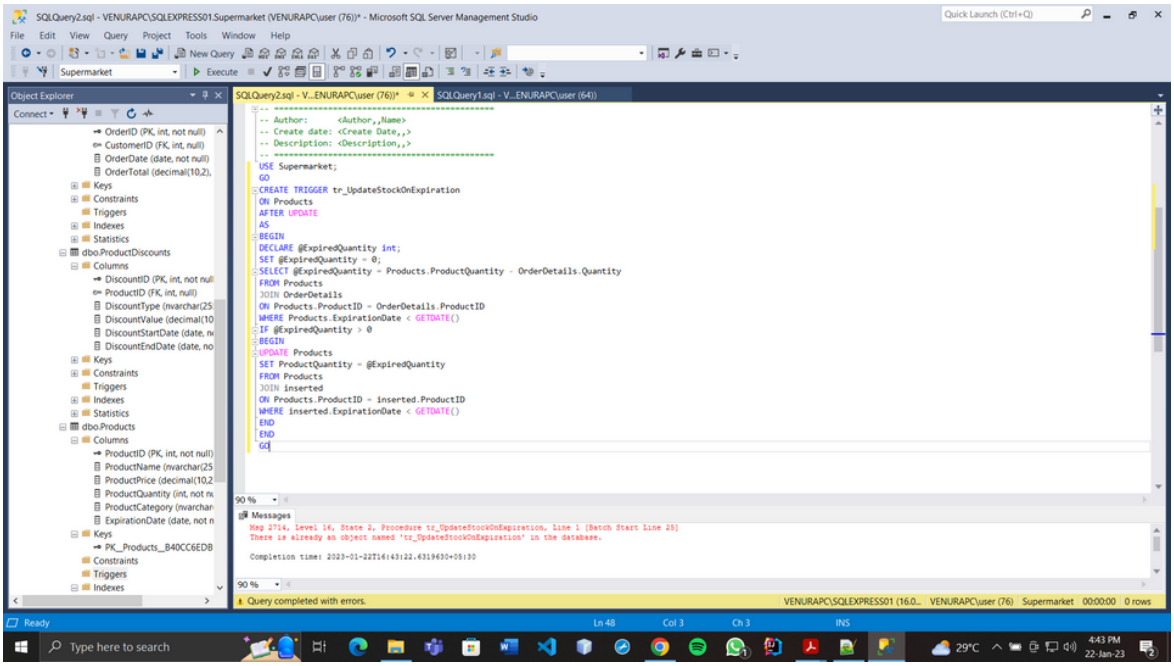


TRIGGERS

Trigger to Update Stock on Purchase

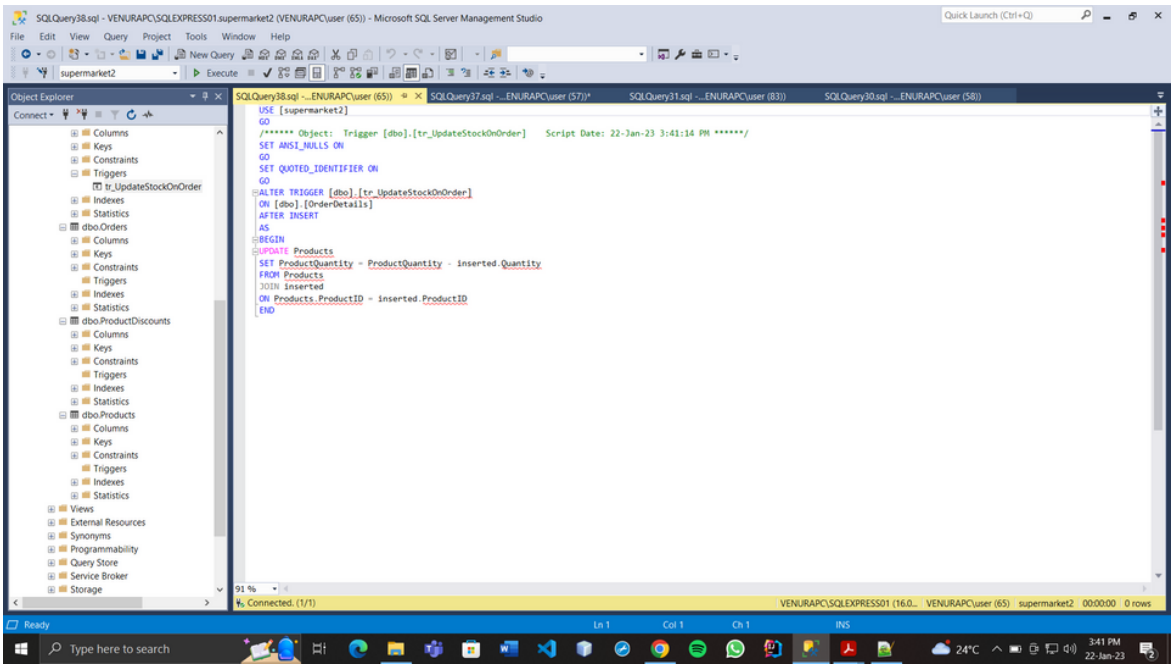


Trigger to Update Stock on Expiration

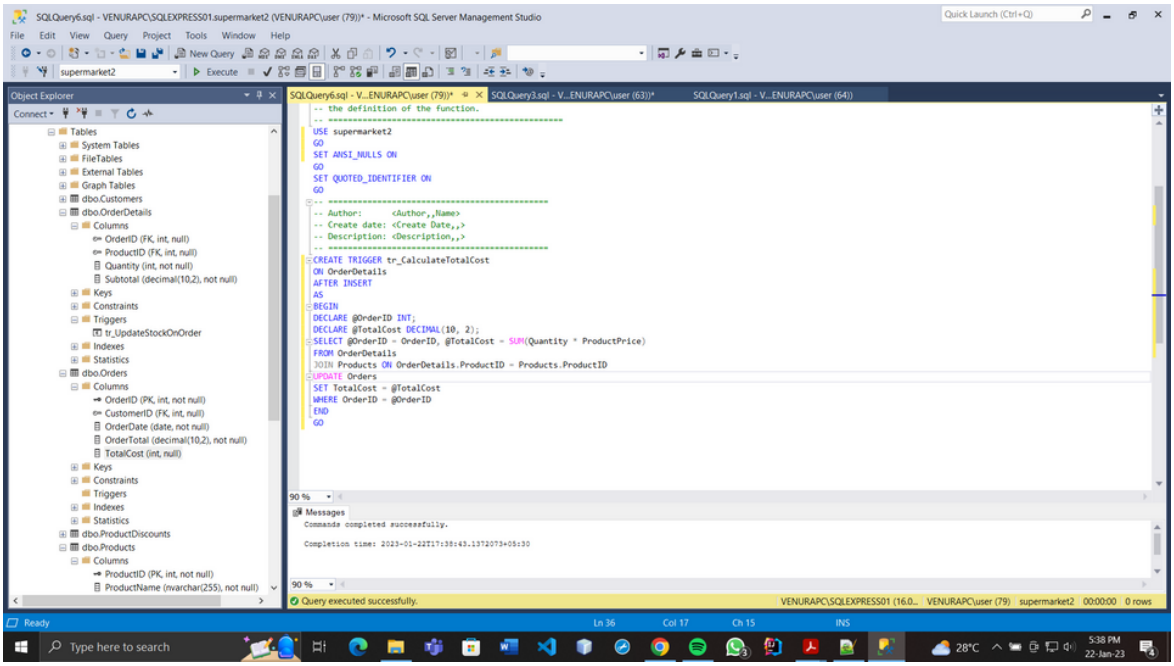


TRIGGERS

Trigger to Update Stock on Order

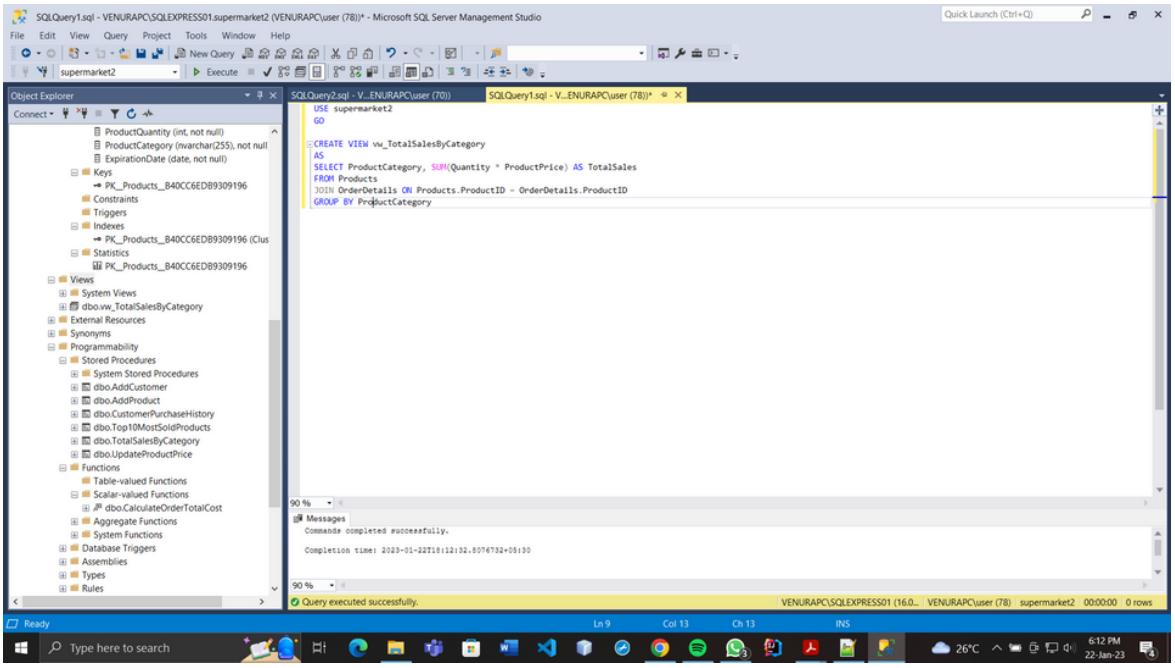


Trigger to Calculate Total Cost

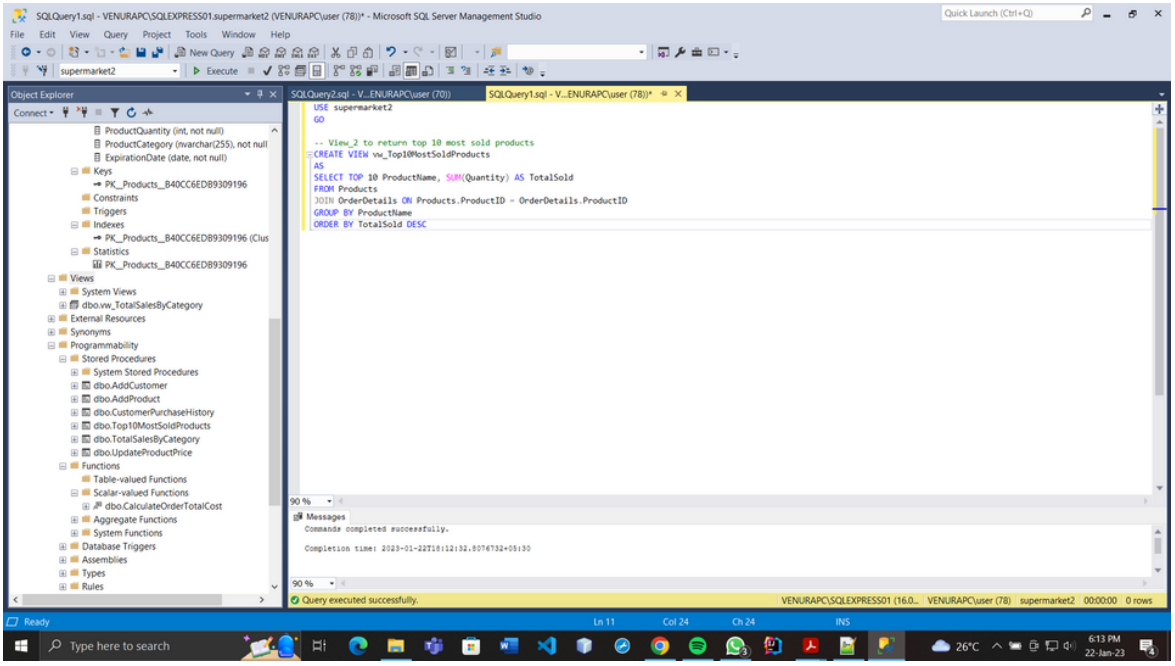


DATABASE VIEWS

View to get the Total Sales by Category

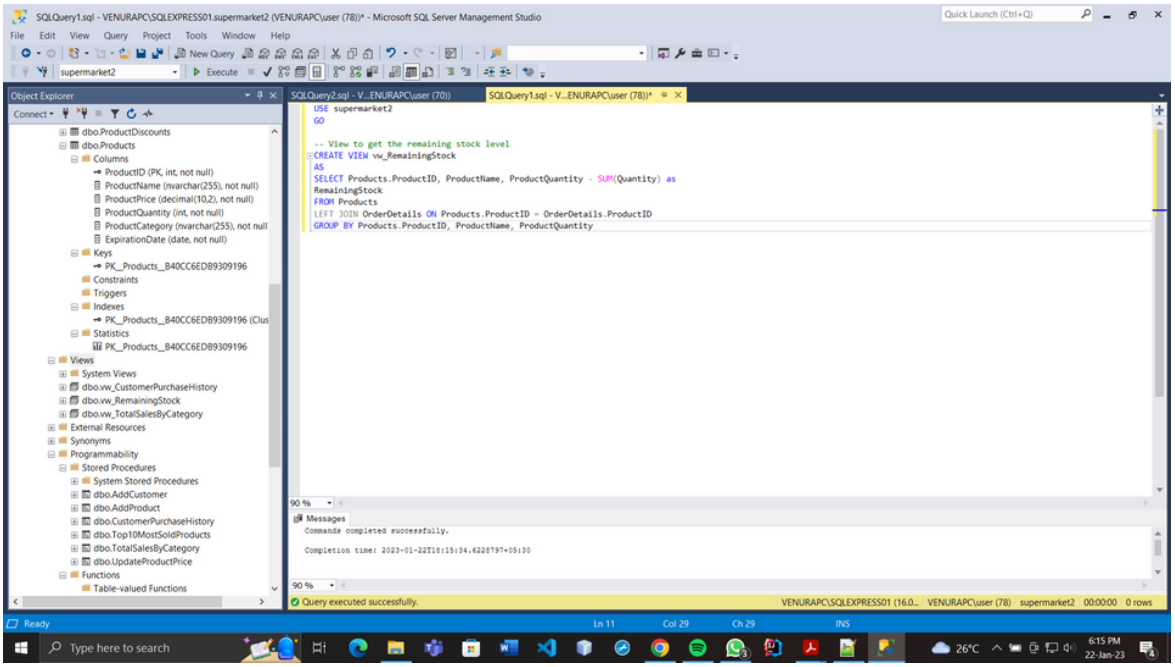


View to get the top 10 most sold Products

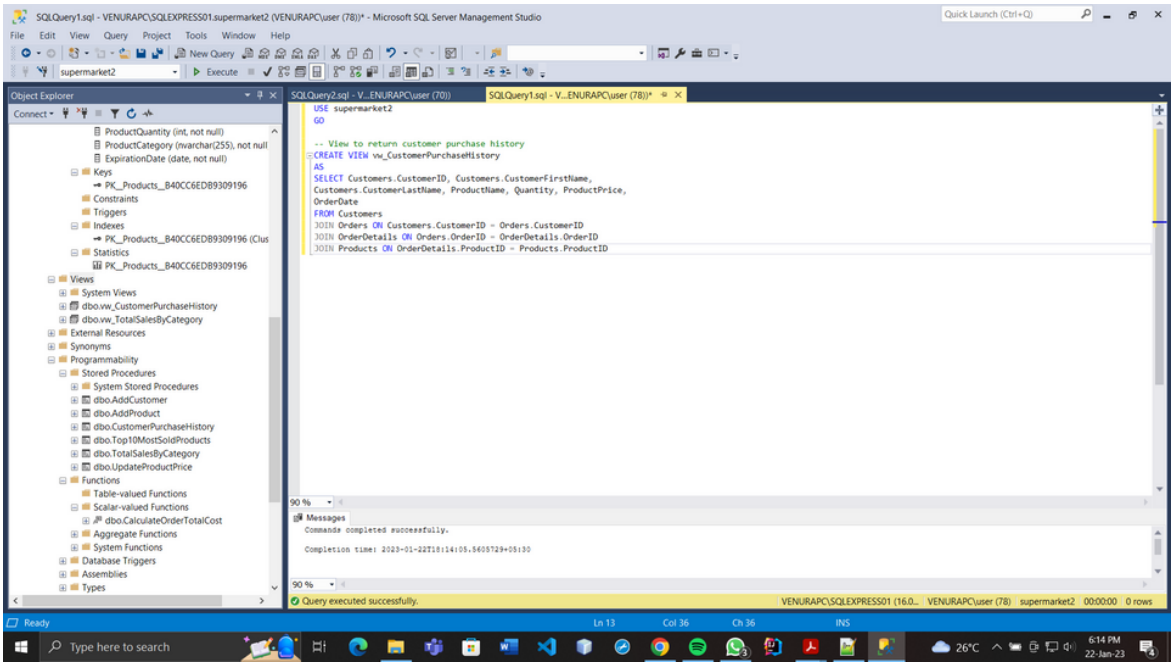


DATABASE VIEWS

View to get the Remaining Stock Level

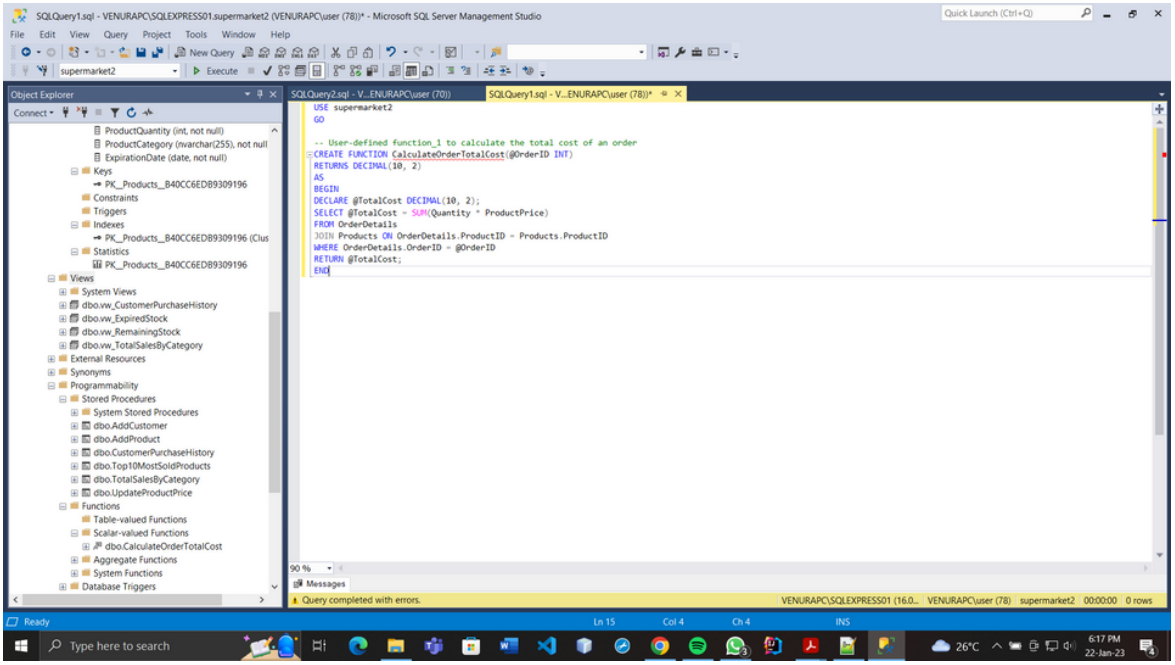


View to get the Customer Purchase History

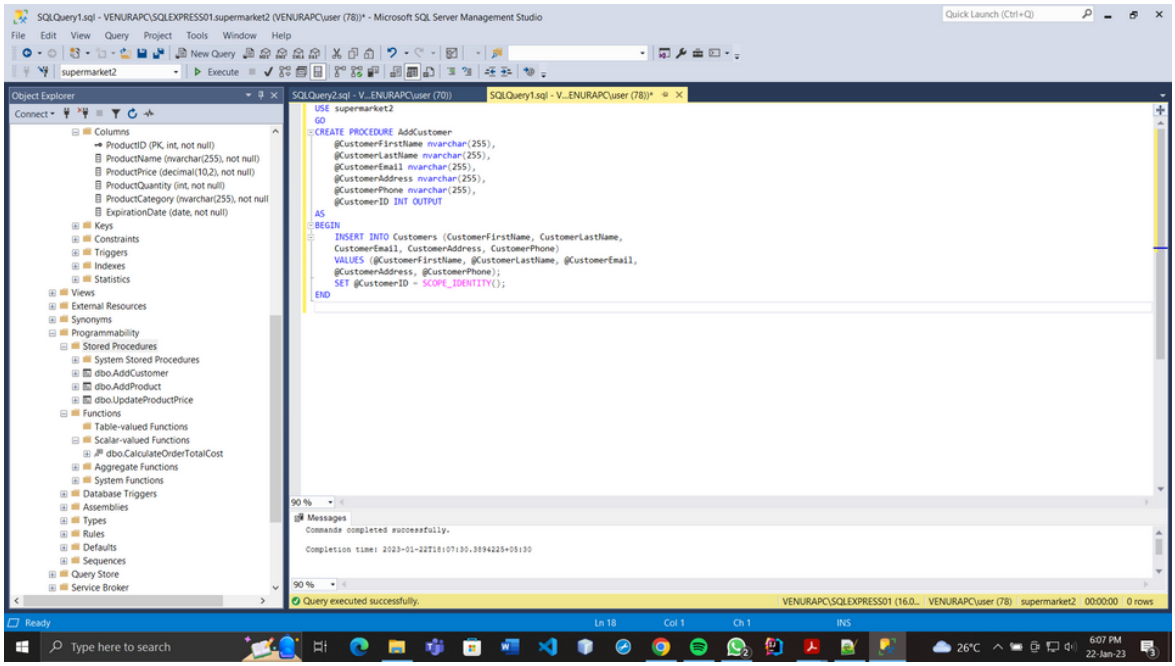


DATABASE FUNCTIONS

User Defined Function to Calculate Total Cost

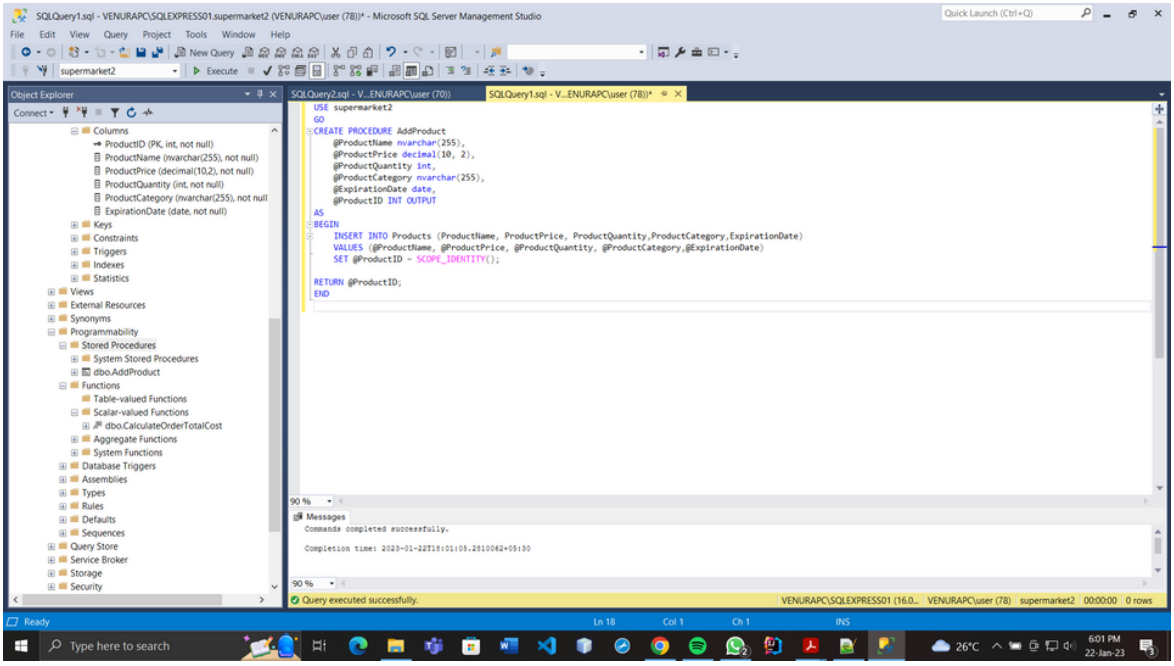


Function inside Procedure to Add Customer

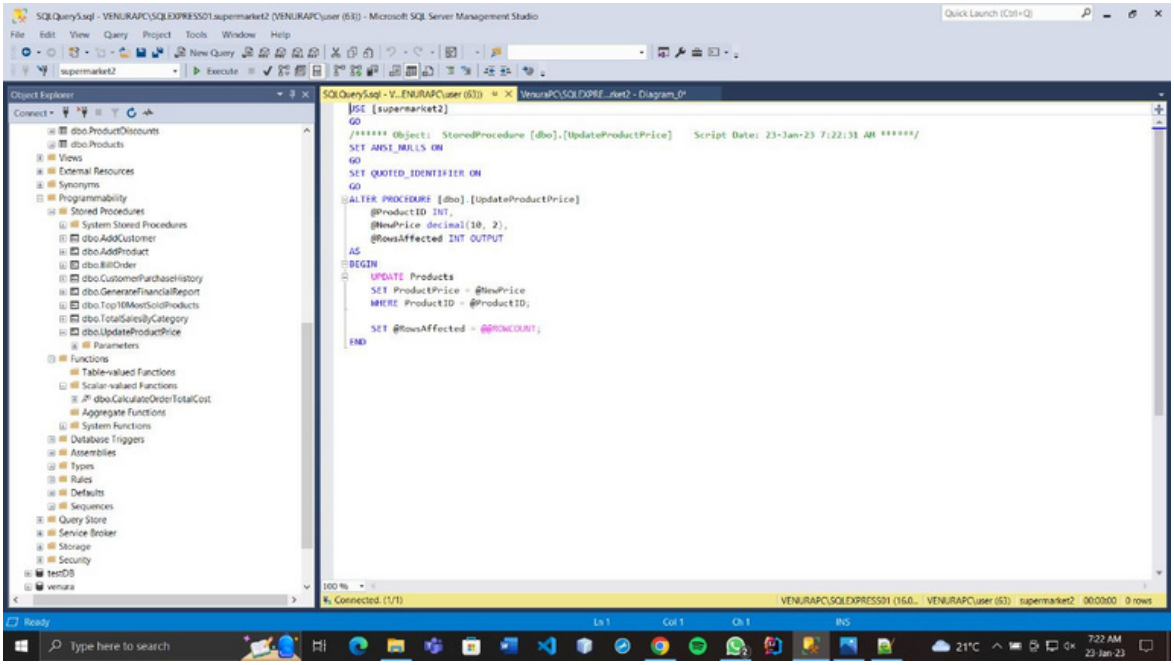


DATABASE FUNCTIONS

Function inside Procedure to Add Product

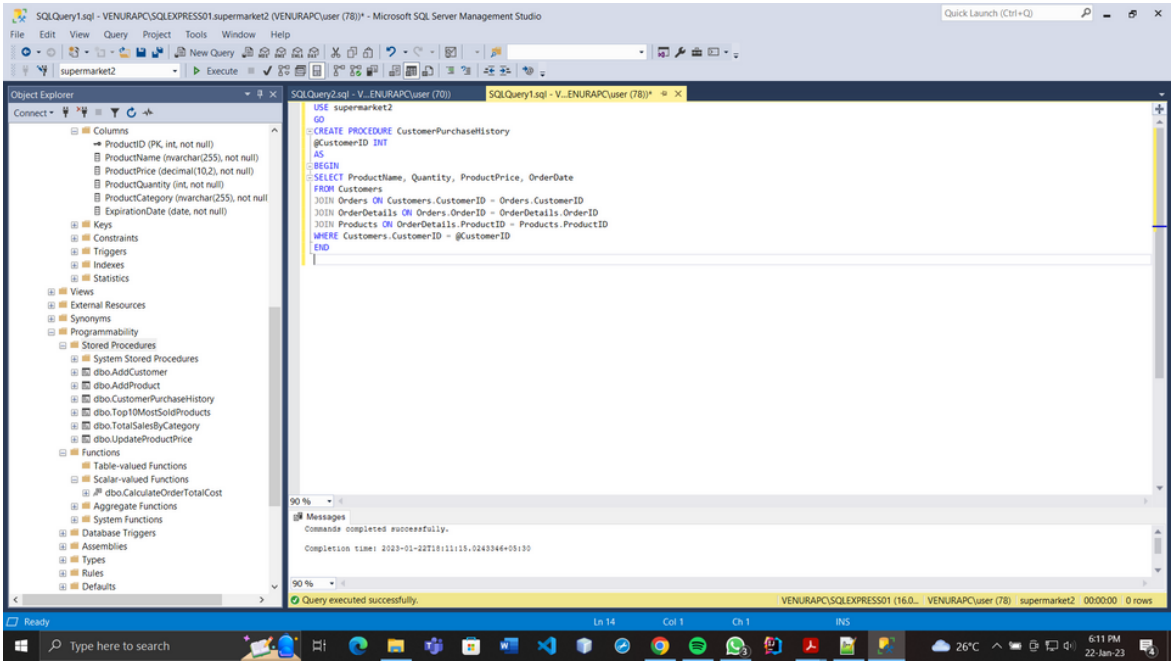


Function inside Procedure to Update Product Price

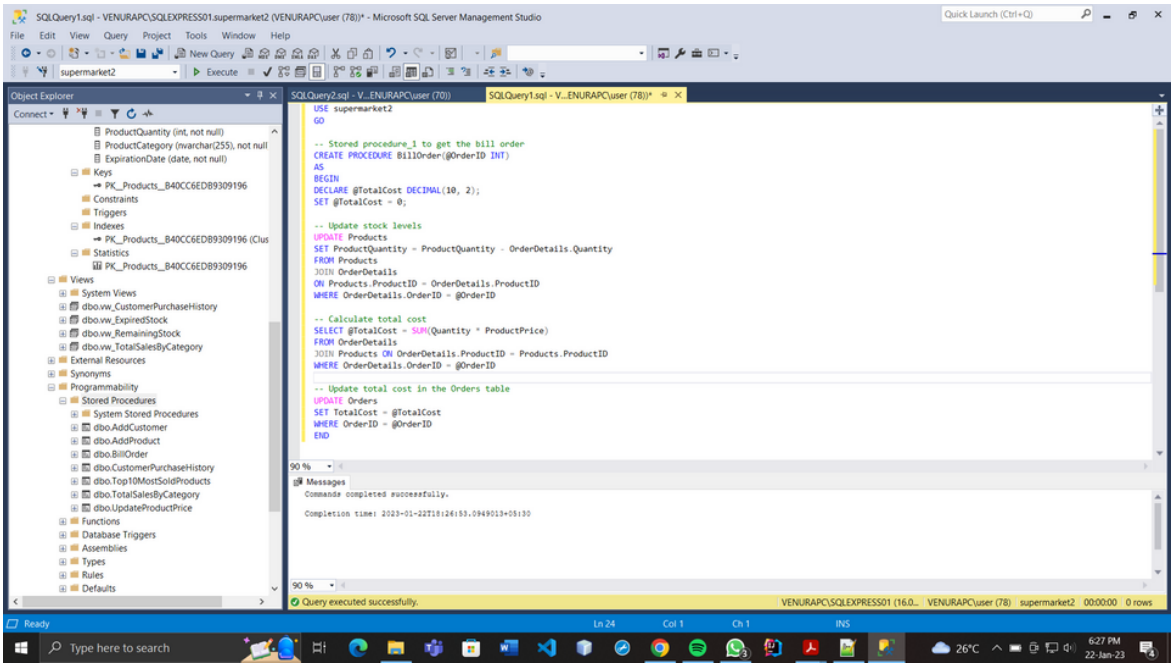


STORED PROCEDURES

Stored Procedure to get the Customer Purchase History

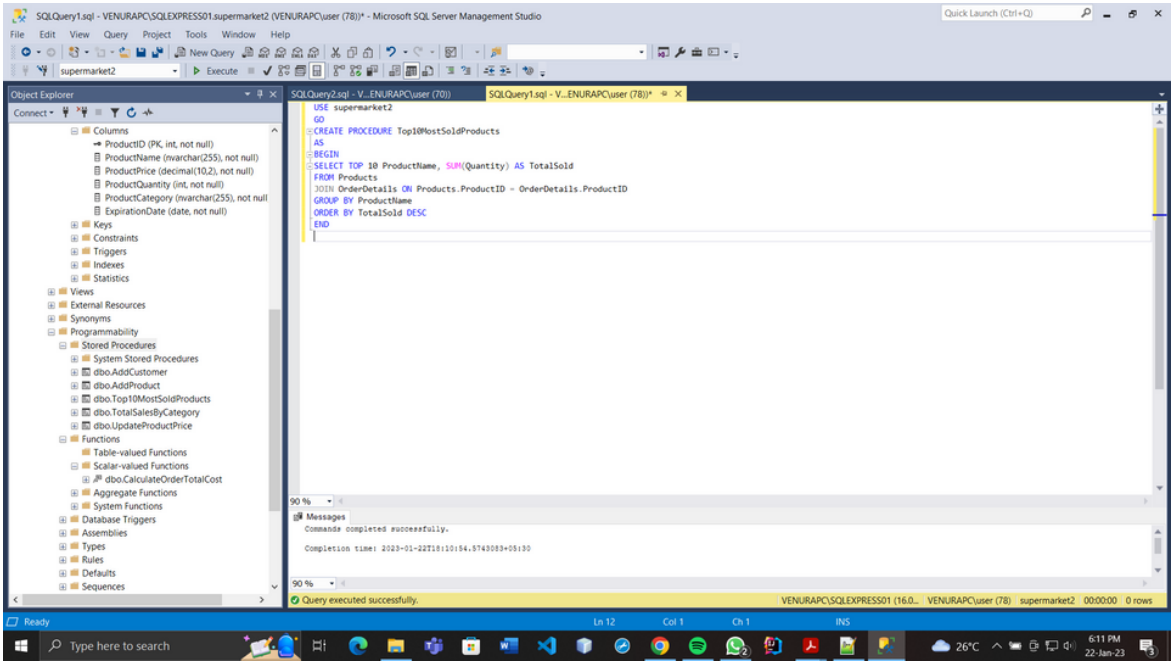


Stored Procedure to get the Bill Order

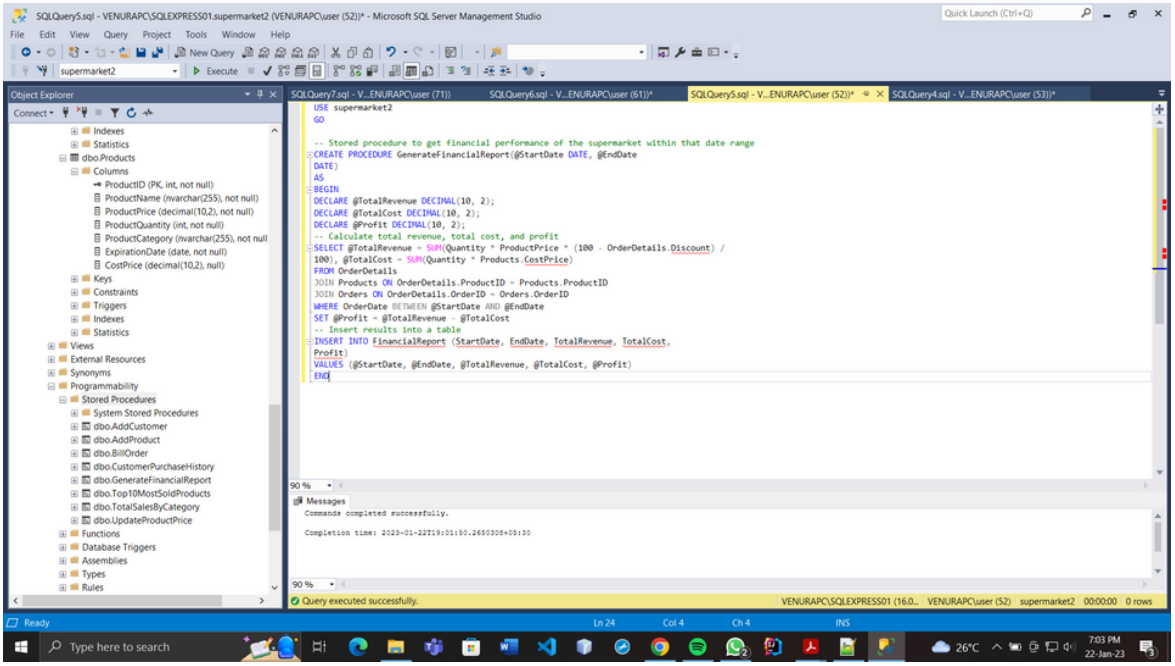


STORED PROCEDURES

Stored Procedure to get the Top 10 most sold Products

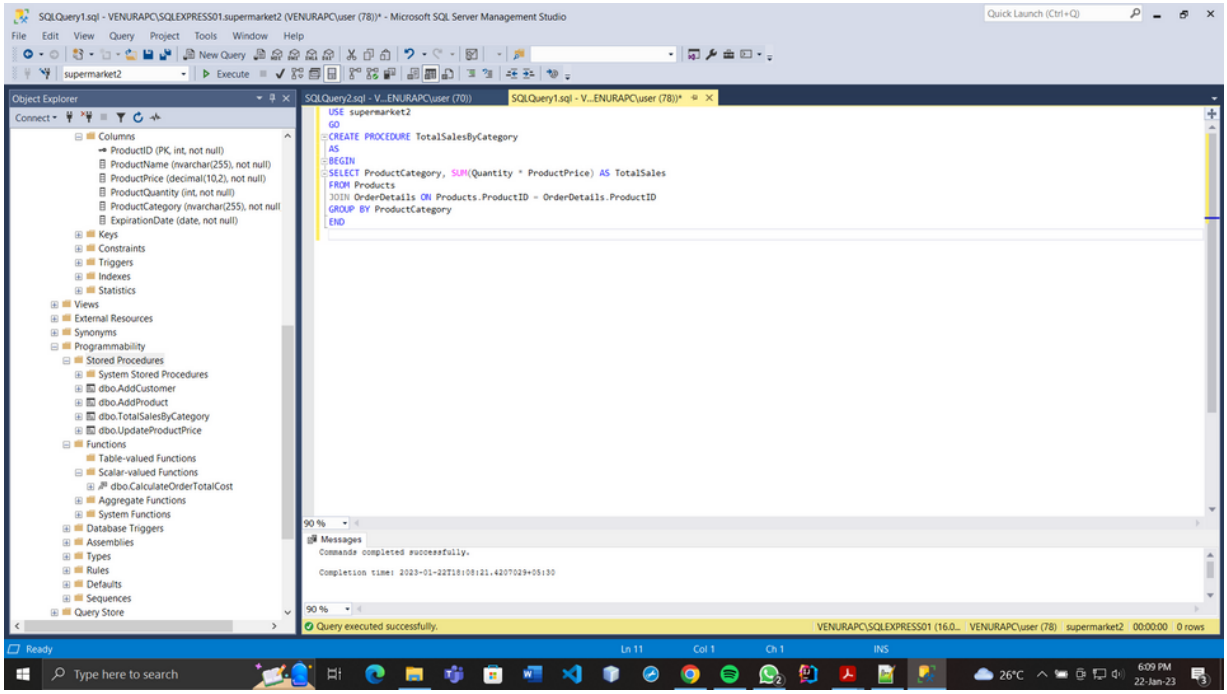


Stored Procedure to Generate Financial Report



STORED PROCEDURES

Stored Procedure to get the Total Sales by Category



CRITICAL EVALUATION

- In our solution we have implemented the ability to restrict and control access levels to the system. Only the users (such as admin/owner) with the necessary permissions can perform operations or modifications to the system and the database.
- Also we provide a detailed financial report providing an analysis of the financial performance of the business and other insights
- One aspect that we can improve upon and isn't covered in depth, in our system is the payment tracking feature. For example: Providing the ability to track the payment mode of the customers, and other insight into customer purchase payment trends.
- Additionally, the ability to evaluate supplier performance and history as well as other data to further provide insight into the performance of the business is not currently supported in our system.
- These shortcomings can be improved by tracking, collecting and storing the necessary data related to these metrics as well as coming up with effective methods to analyse this data in order to get useful reports that can lead to better decision making from the parties in charge of management and operation of the supermarket.
- In conclusion we believe our system manages to cover all the requirements and various verticals and aspects of the business and provides an effective solution to the management in order for them to run and oversee the supermarket business operations effectively and efficiently. Overall, the system is a complete solution to the problem scenario presented.

FUTURE IMPLEMENTATIONS

- One feature that can be looked into further and implemented in the future is to expand on the permissions aspect of the system. Providing granular control of permissions such as read, write, update, delete, etc. through the system by assigning permissions to the users of the system.
- Also, the advanced reporting aspect can be improved. Providing deeper insight using various data analysis methods to analyze trends, history and provide accurate predictions that are tailormade according to the data in the system. These can be generated in the form of additional reports.
- Following up on the advanced data analysis even the accounting feature can be further improved by implementing advanced accounting analytics and techniques to further get an accurate picture/ representation for the owners, admins or any other management party to accurately make informed decisions that critical in day-to-day operations of the supermarket
- The system can also be further expanded upon, depending on multiple locations or branches that are opened by the business in the future and all of them can be connected to this system to manage all operations smoothly.

TEAM AND WORKBREAKDOWN

PU ID: 10820799
Name: HIKKADUWAGE SILVA

Planning and design and evaluation ER, Report, Create views, Create triggers normalization, SQL coding to create DB

PU ID: 10818175
Name: VENURA DEEGALLA

Access Level creation, ER, Report, Create triggers, Create functions, SQL coding to create DB tables

PU ID: 10818172
Name: SHAUN HENNEDIGE

Planning and design, Database Diagram,, Data dictionary, Report, Create stored procedures.

PU ID: 10750069
NAME: MADHAVA DOMBAGAHA

Presentation, ER, SQL coding to insert data into tables, Report.

PU ID: 10818163
Name: RATHNAYAKA RATHNAYAKA

Presentation, SQL coding to create DB tables, Data dictionary, Relational Schema

PU ID: 10819571
Name: DILEEP ABEYSINGHE

Presentation, Relational Schema, SQL coding to create DB tables, insert data into tables,