Database Design Document (DBDD)

Mocca Café

Date: 11/18/2019

Name of Team: Peak Performers

Members names ASC:

Al Barwany, Nawras

Gahramanova, Darya

Liburd, Kaley

Morino, Natalia

Sharif, Mahir Muhtasim

Version	**Description**
1.0	First released draft
2.0	1. Mocca ERD 2. Mocca EERD 3. Mocca Relational Schema 4. Mocca Data Dictionary Header 5. Mocca Data Dictionary
3.0	Summary of changes: 1. Updated ERD 2. Updated EERD 3. Updates RS 1. Updated Mocca Data Dictionary a. Create data type phone(10) b. Change all data type for Phone to phone(10) in Customer, Employee and Supplier Tables. c. Created check for State values in Employee, Supplier and Customer Tables. State is like [A-Z][A-Z] d. State is restricted to char(5) e. Created check for Reward in Customer Table. Reward is like [Y] OR [N] f. Created check for Type in Order Table. Type is like [Online] OR [In-store] g. Created check for QtyOrder in Order Item Table. QtyOrder >=0 h. Created check for Type in Menu Item Table. Type is like [C] OR [T] OR [P] OR [S] i. Created check for QtyUsed in Ingredient Table. Qty Used is >0 j. Created check for QtyOnHand in Product Table. Qty On Hand >=0

	k. Created check for QtyProduct in Product Supply. Qty Product >=0 l. Created check for Type in Supplier Type Table. Type like [C] OR [R] 2. Rules: 3. Defaults: 3. SPROCS 4. Table Views
4.0	Summary of changes: 1. Table View Reports 2. User-Acceptance Test

Purpose

Peak Performers prepared this document to design a data model listing the major entities, attributes, and relations that make up Mocca Cafeé

Narrative

Mocca is a small coffee shop that serves local products in the Tampa Bay area. Customer can purchase in store and through different applications such as: Uber, GrubHub, and DoorDash. Every order is tracked by a unique order identifier. Mocca also wants to know what products are in each order and if the order is online or in store.

Customers can choose from a variety of menu items. Store offers coffee, teas, pastries and sandwiches. Customer can choose to join the rewards program to gain points toward free

items and enjoy special offers. In order to join, customers are asked for their name, phone number, email, and address.

Menu items are made of one or more products. One product can be used for different menu items. Each menu item is tracked by its unique ID and menu item type. For each menu item ordered by a customer, Mocca tracks the amount and unit price.

Products are delivered to Mocca by different suppliers. The same product can be delivered from different suppliers, and one supplier can deliver multiple types of products. Stores tracks supplier by name, location, phone number, and email. Products are tracked by product IDs and prices.

The store is operated by three baristas in each shift. Baristas can serve as cashiers, as well. Cashier takes customer's order and checks him out. The number of shifts each barista works varies per week. Moreover, every time the store is open it has a manager that supervises other employees. Employees are tracked by employee number, address, and phone number. Employees are paid hourly. Hourly Rate might vary between employees.

Business Rules

- Customer places one or many orders. An order can be placed by one customer.
- Customer is helped by one barista. A barista helps one or many customers.
- An order is prepared by one barista. A barista prepares none or many orders.
- An order has one or many menu items. A menu item can be in one or many orders.

- A menu item is has one or many products. A product can be in one or many menu items.
- Supplier delivers one or many products. A product can be delivered by one or many suppliers.

Actors/Roles

- Customer: places orders, is helped by barista
- Supplier: delivers products
- Employee: helps customers, prepares orders
- Menu item: has products, is in orders
- Product: is in menu items, delivered by supplier
- Order: placed by customer, prepared by barista, has menu items.

Entities with nested attributes

- CUSTOMER

  CustID

  Name

  (First Name, Last Name)

  Address

  (Street, City, State, Zip Code)

  Phone

  Email

  Reward

- ORDER

  OrderID

  Type

- EMPLOYEE

  EmpID

  Name

  (First Name, Last Name)

  Address

  (Street, City, State, Zip Code)

  NoHours

  HourlyRate

  [Salary]

  MangID

- SUPPLIER

  SuppID

  Name

  Address

  (Street, City, State, Zip Code)

  Phone number

  Email

{Type}

- MENUITEM

  ItemID

  Name

  Price

  Type

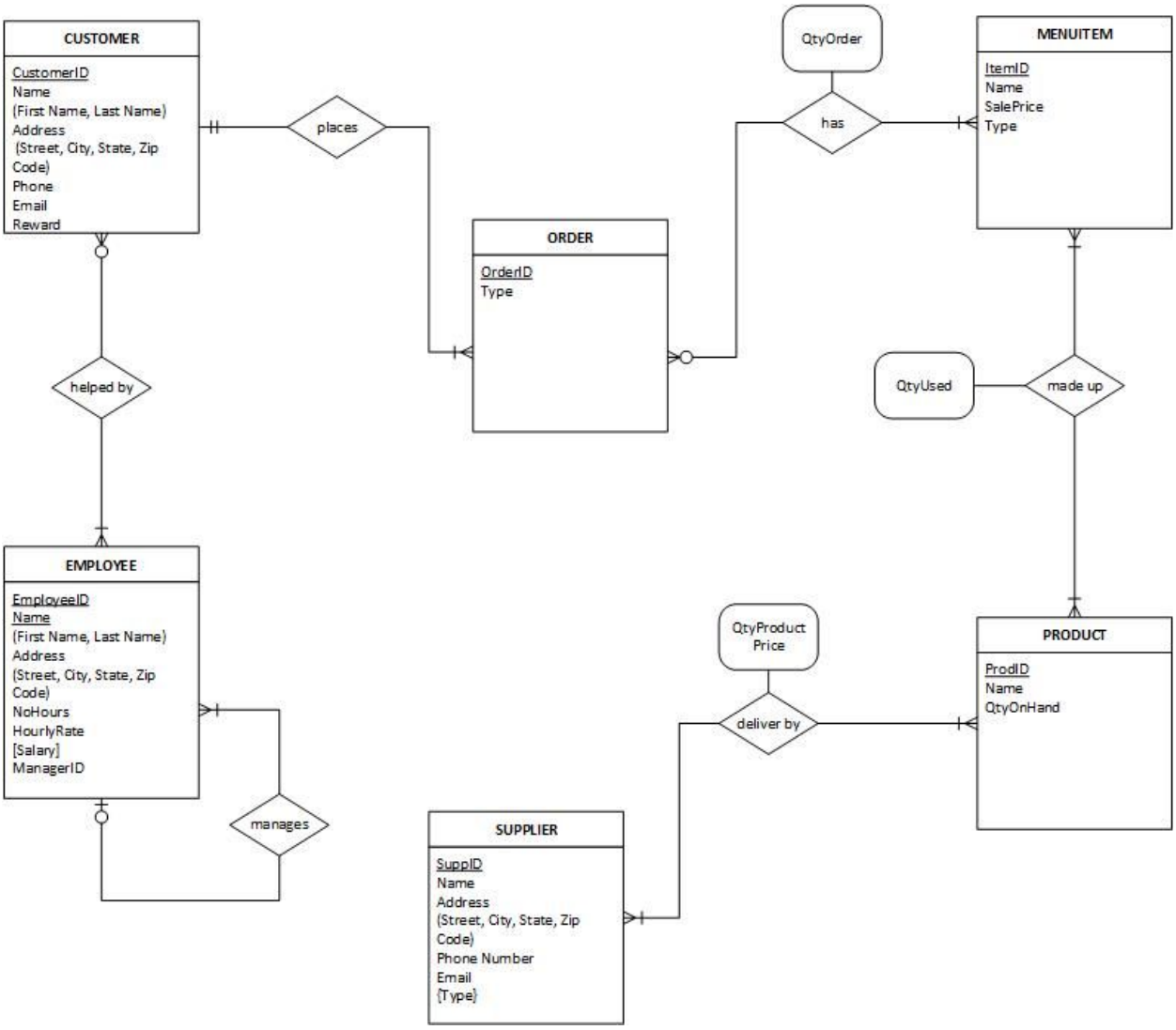
- PRODUCT

  ProdID

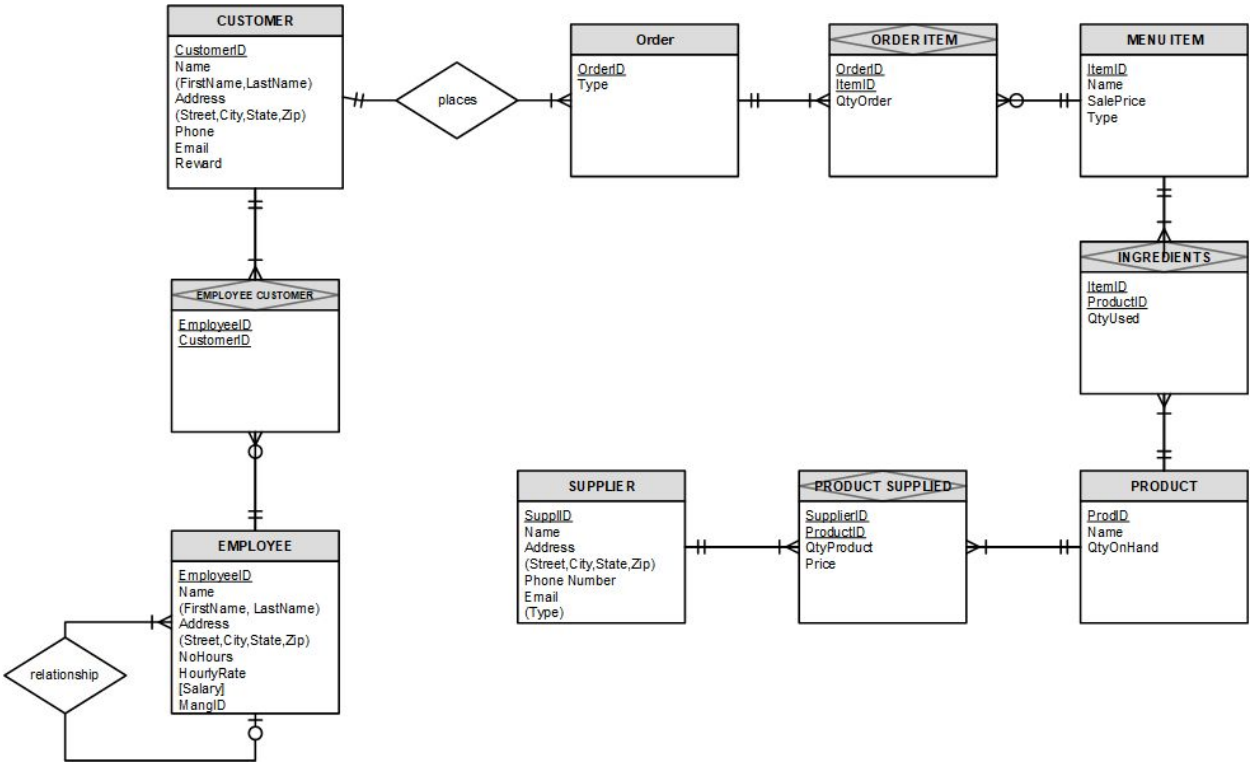
  Name

  QtyOnHand

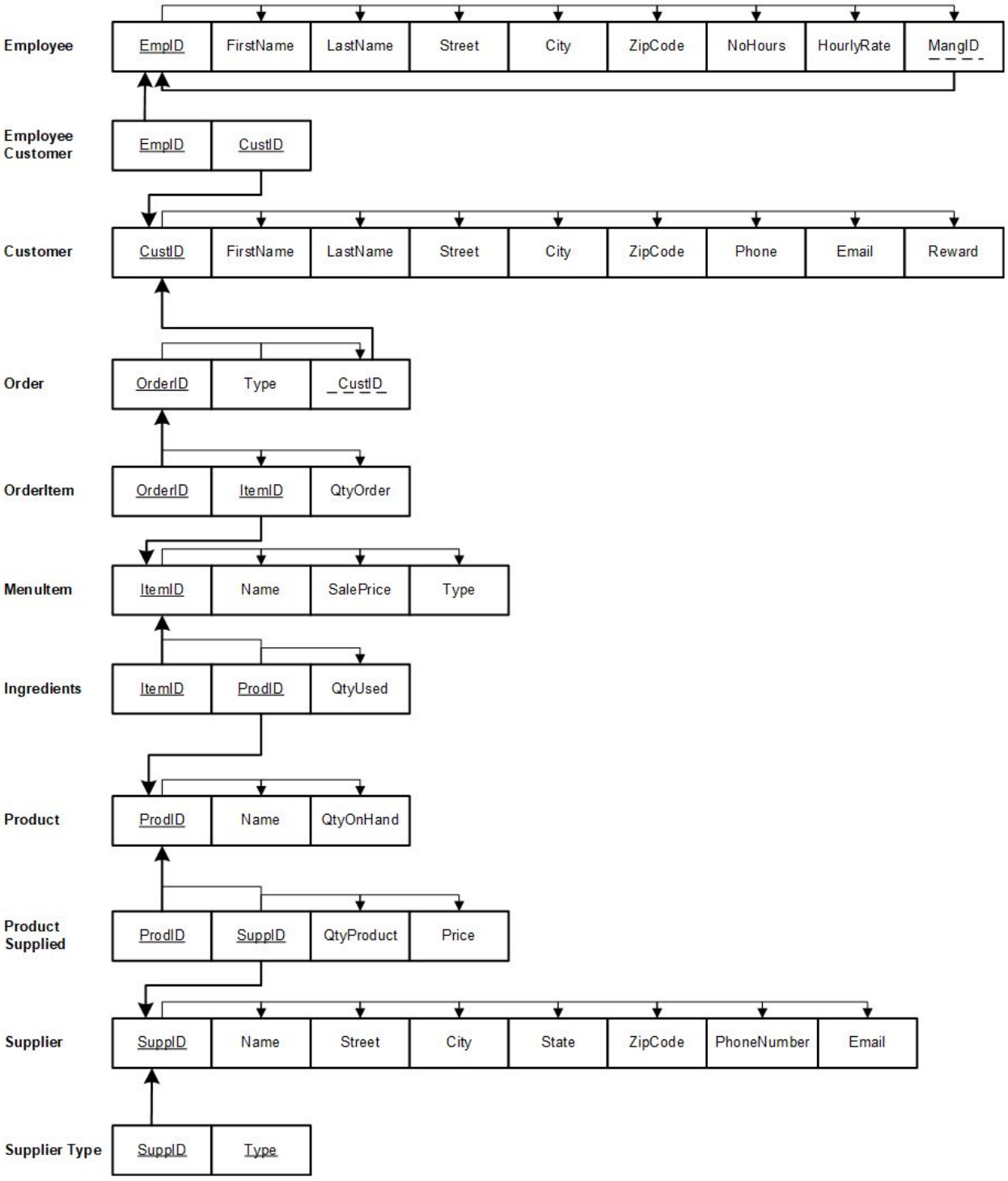
ERD


CUSTOMER
CustomerID
Name
(First Name, Last Name)
Address
(Street, City, State, Zip Code)
Phone
Email
Reward
places
ORDER
OrderID
Type
QtyOrder
has
MENUITEM
ItemID
Name
SalePrice
Type
helped by
QtyUsed
made up
EMPLOYEE
EmployeeID
Name
(First Name, Last Name)
Address
(Street, City, State, Zip Code)
NoHours
HourlyRate
[Salary]
ManagerID
manages
QtyProduct Price
deliver by
PRODUCT
ProdID
Name
QtyOnHand
SUPPLIER
SuppID
Name
Address
(Street, City, State, Zip Code)
Phone Number
Email
{Type}

EERD


CUSTOMER
CustomerID
Name
(FirstName,LastName)
Address
(Street,City,State,Zip)
Phone
Email
Reward
places
Order
OrderID
Type
ORDER ITEM
OrderID
ItemID
QtyOrder
MENU ITEM
ItemID
Name
SalePrice
Type
INGREDIENTS
ItemID
ProductID
QtyUsed
EMPLOYEE CUSTOMER
EmployeeID
CustomerID
SUPPLIER
SupplID
Name
Address
(Street,City,State,Zip)
Phone Number
Email
(Type)
PRODUCT SUPPLIED
SupplierID
ProductID
QtyProduct
Price
PRODUCT
ProdID
Name
QtyOnHand
EMPLOYEE
EmployeeID
Name
(FirstName, LastName)
Address
(Street,City,State,Zip)
NoHours
HourlyRate
[Salary]
MangID
relationship

Relational Schema



Data Dictionary Summary Header

Employee (EmployeeID, FirstName, LastName, Street, City, State, ZipCode, NoHours, HourlyRate, *ManagerID*)

Employee Customer (EmployeeID, CustomerID)

Customer (CustomerID, FirstName, LastName, Street, City, State, ZipCode, Phone, Email, Reward)

Order (OrderID, Type, *CustomerID*)

Order Item (OrderID, ItemID, QtyOrder)

Menu Item (ItemID, Name, SalePrice, Type)

Product (ProdID, Name, QtyOnHand)

Supplier (SuppID, Name, Street, City, State, ZipCode, PhoneNumber, Email)

Product Supply(ProdID, SuppID, QtyProduct, Price)

Supplier Type (SuppID, Type)

Data Dictionary

Table: **Employee**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
EmpID	**PK**; Unique sequential employee ID number	int		Y						Y

FirstName	First name of the employee	varchar	20							
LastName	Last name of the employee	varchar	20							
Street	Street of the employee	varchar	20							
City	City of the employee	varchar	20							
State	State of the employee	char	2					LIKE '[A-Z][A-Z]'		
ZipCode	Zip code of the employee	char	5					Like '[0-9][0-9][0-9][0-9][0-9]'		
NoHours	Number of Hours employee has worked	int							Y	
HourlyRate	Phone number of the employee	int								
MangerID	**FK** : Unique Manager ID	int							Y	

Table: **Employee Customer**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
EmpID	**CPK:FK;** To Employee table	int								Y

CustomerID	**CPK:FK** To Customer Table	int								Y

Table: **Customer**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
CustomerID	**PK;** Unique sequential customer ID number	int		Y						Y
FirstName	First name of the Customer	varchar	20							
LastName	Last name of the customer	varchar	20							
Street	Street of the customer	varchar	20							
City	City of the customer	varchar	20							
State	State of the customer	char	2					LIKE '[A-Z][A-Z]'		
ZipCode	Zip code of the customer	char	5					Like '[0-9][0-9][0-9][0-9][0-9]'		
Phone	Phone number of the customer	phone	10						Y	

Email	Email of the customer	Varchar	30						Y	
Reward	If customer is enrolled in Reward	char	1					([Type]='(Y)' OR [Type]='(N)')		

Table: **Order**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
OrderID	**PK;** Unique sequential employee ID number	int		Y						Y
Type	Order type	varchar	10					([Type]='(Online)' OR [Type]='(In-Store)')		
CustID	**FK:** Customer ID for customer table	int								

Table: **OrderItem**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
OrderID	**CPK:FK;** Order ID to Order table	int								Y
ItemID	**CPK:FK;** Item ID to MenuItem table	int								Y

QtyOrder	Order Amount	int						>=0		

Table: **MenuItem**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
ItemID	**PK;** Unique sequential item ID number	int		Y						Y
Name	Name of Item	varchar	30							
SalePrice	Price of item	int								
Type	Item Type (Coffee, Tea, Pastry, Sandwich)	char	1					([Type]='(C)' OR [Type]='(T)') OR [Type]='(P)' OR [Type]='(S)')		

Table: **Ingredient**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
ItemID	**CPK:FK;** Item Id to MenuItem table	int								Y
ProdID	**CPK:FK;** Item Id to Product table	int								Y
QtyUsed	Amount of Products used	int						>0		

Table: **Product**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index

ProdID	**PK**; Unique sequential product ID number	int		Y						Y
Name	Product Name	varchar	20							
QtyOnHand	Amount of product	int						>=0		

Table: **Product Supplied**

Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
ProdID	**CPK:FK;** Unique sequential employee ID number	int								Y
SuppID	**CPK:FK:** First name of the employee	int								Y
QtyProduct	Amount of product supplied	int						>=0	Y	
Price	Product Price	int								

Table: **Supplier**

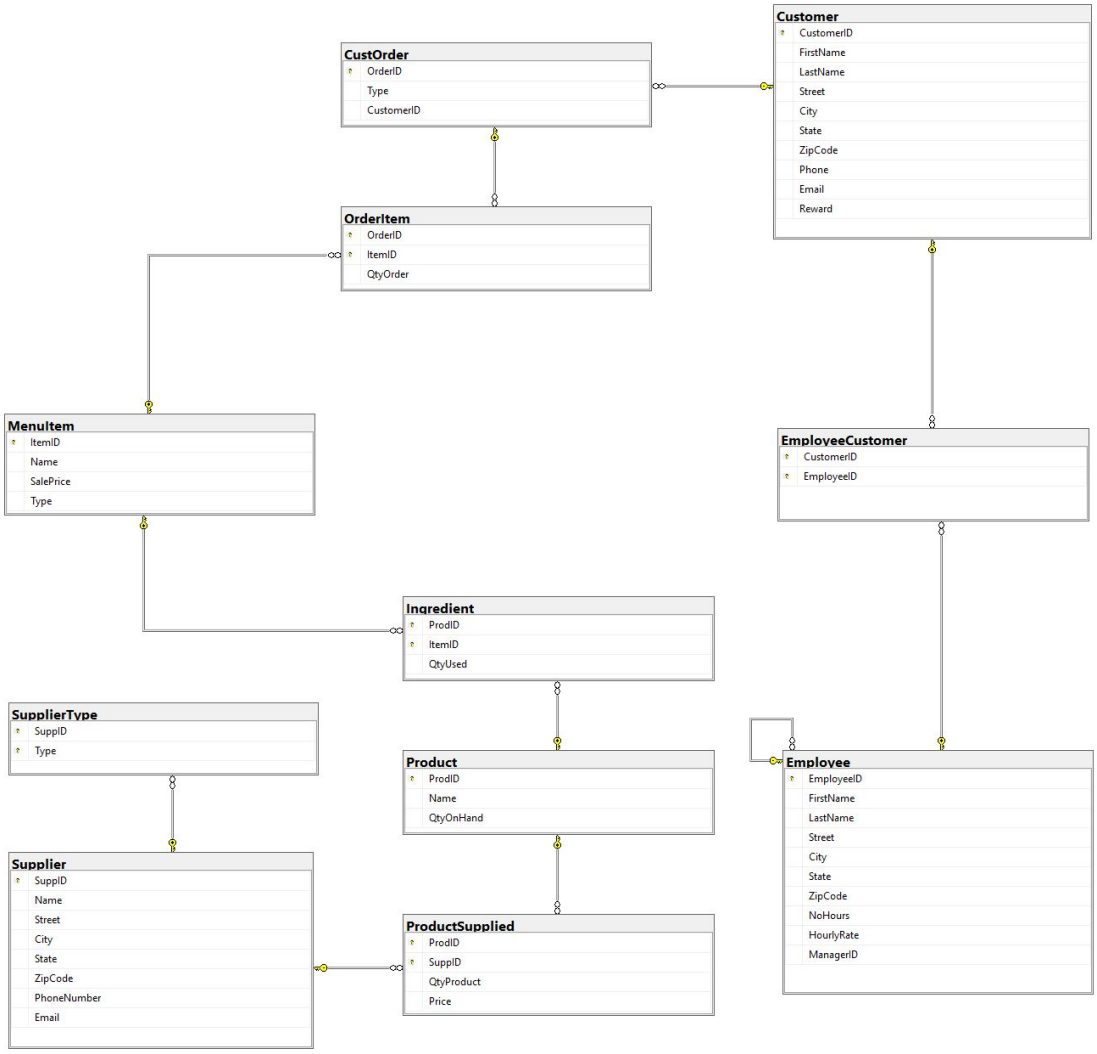
Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
SuppD	**PK;** Unique sequential supplier ID number	int		Y						Y
Name	Name of the Supplier	varchar	20							

Street	Street address of supplier	varchar	20							
City	City of the supplier	varchar	20							
State	State of the supplier	char	2					LIKE '[A-Z][A-Z]'		
ZipCode	Zip code of the supplier	char	5					Like '[0-9][0-9][0-9][0-9][0-9]'		
PhoneNumber	Phone number of the supplier	phone	10							Y
Email	Email of supplier	varchar	20							Y

Table: **Supplier Type**

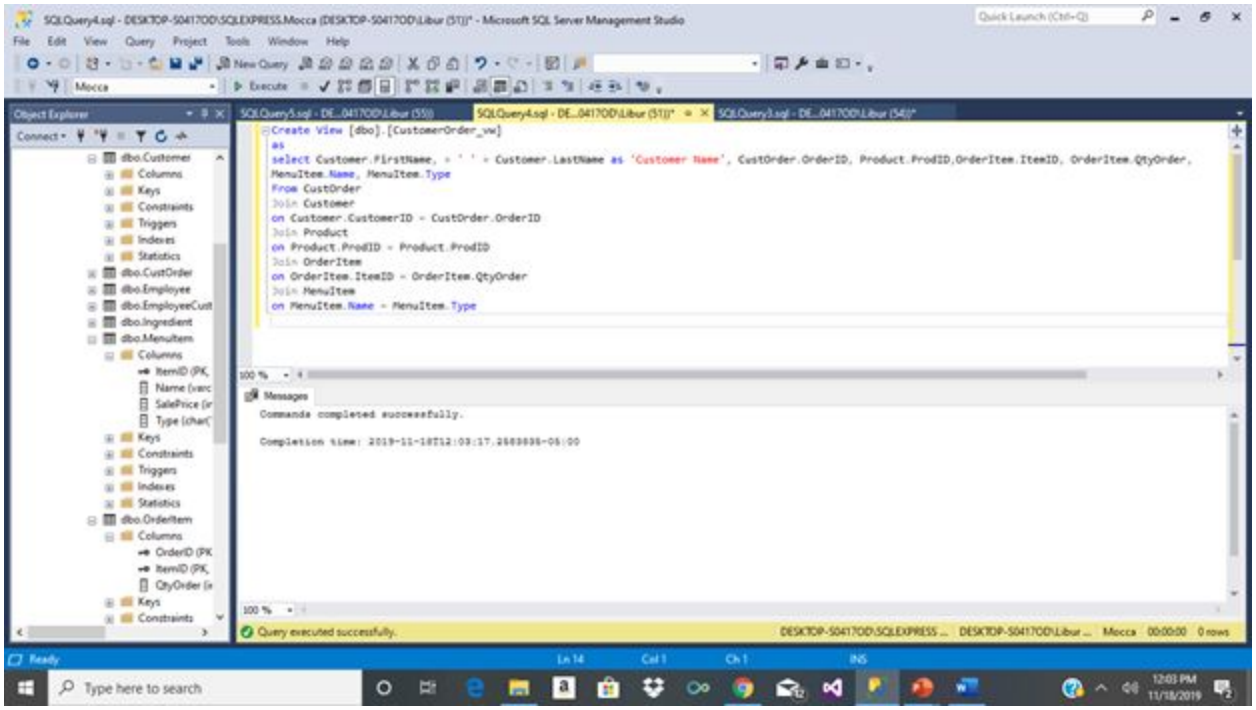
Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
SuppID	**FK:PK;** Foreign key to Supplier table	int		Y						Y
Type	Supplier Type	char	1					[Type] = ('C') OR [Type] = ('R')		

Database Diagram



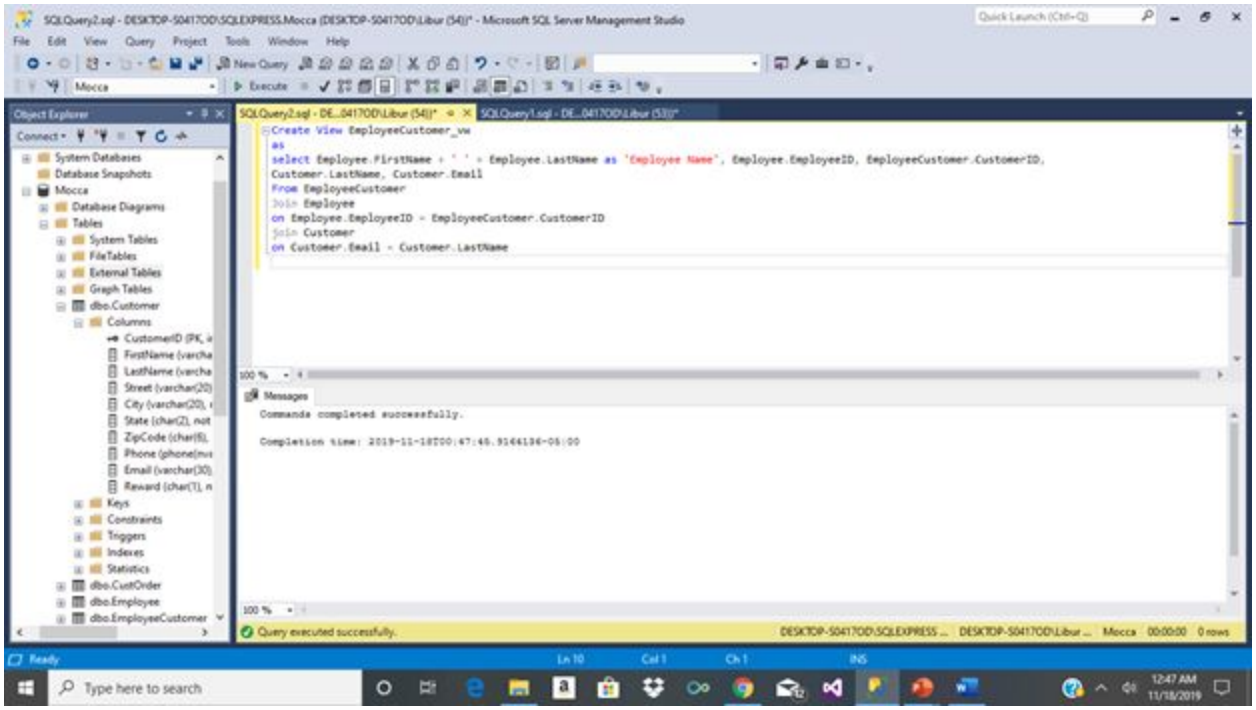
Customer Order View Description

The Customer Order view joins the Customer, CustomerOrder, Menuitem, OrderItem, and Product tables on their primary key, and foregin key relationship. The columns that are used are: Customer First Name, Customer Last Name, QtyOrder, MenuItem Name, MenuItem Type, OrderID, and ProductID.



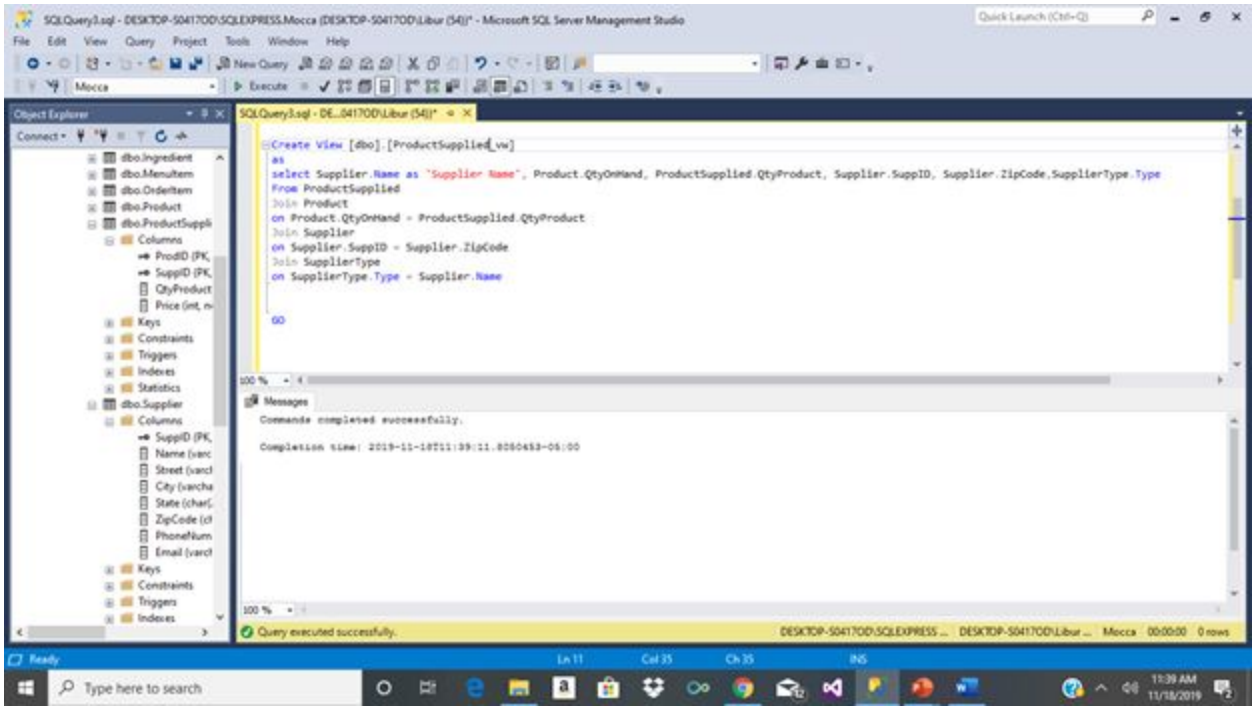
Employee Customer Relationship View Description

The Employee-Customer view joins the Employee, EmployeeCustomer, and Customer tables on their primary key, and foregin key relationship. The columns that are used are: Employee First Name, Employee Last Name, EmployeeID, CustomerID, Customer Last Name, Customer Email.



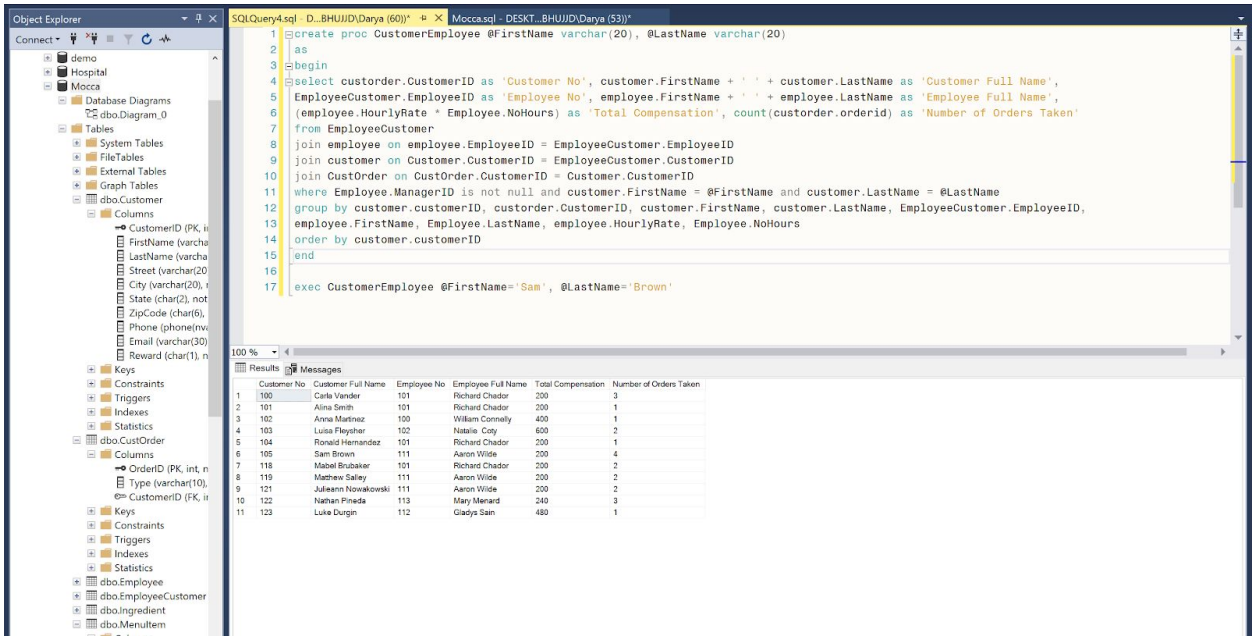
Products Supplied View Description

The Product Supplied Order view joins the Supplier, ProductSupplied, CustomerOrder, SupplierType, and Product tables on their primary key and their attributes. The columns that are used are: Supplier Name, Product QtyOnHand, ProductSupplied by Qty Product, SupplierType, SupplierName, Supplier ZipCode, and SupplierID.

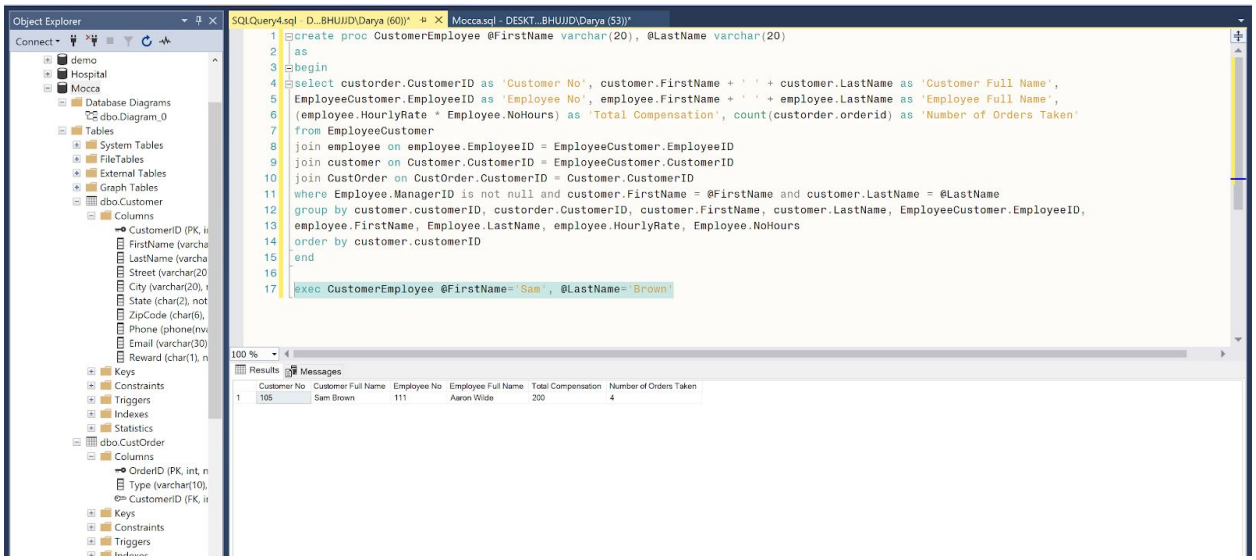


Stored Procedure 1: CustomerEmployee

This procedure demonstrates a list of IDs and full names of customers, full names and IDs of employees who served them, total compensation calculated as hourly rate * number of hours, and total number of orders taken from a given customer by a given employee. Employees are limited to those who have a supervisor (ManagerID is not null). This can be helpful in comparing employee workload and compensation by customer.

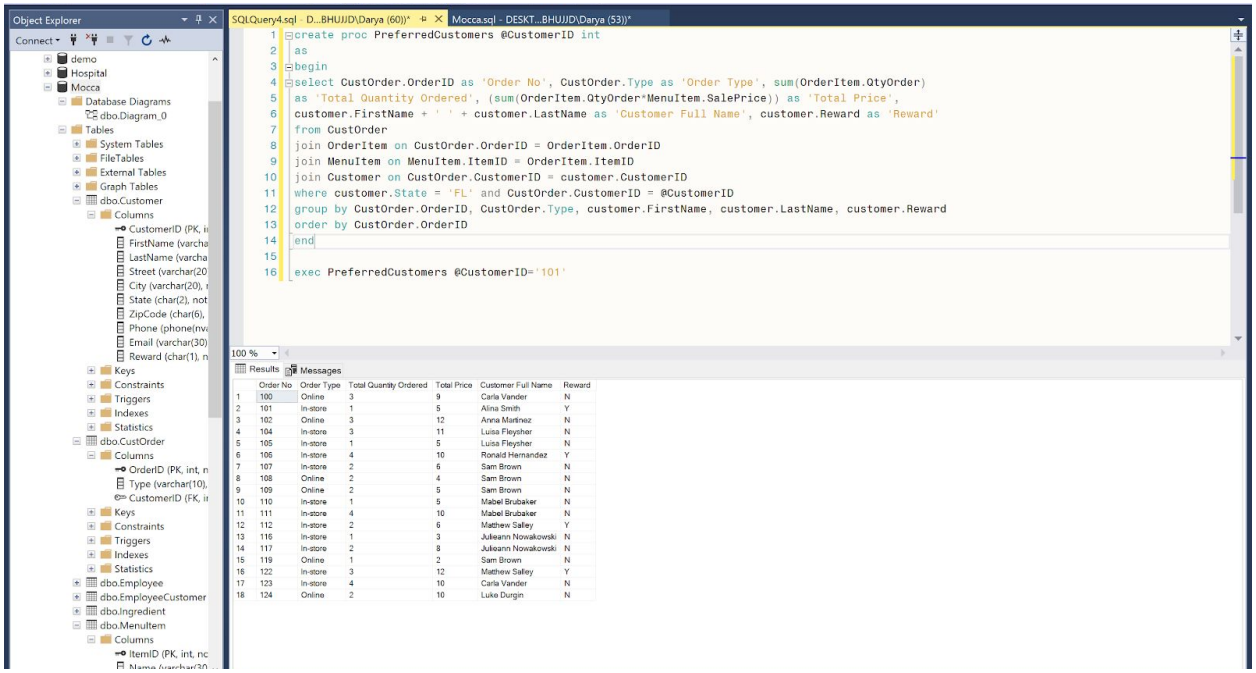


Below is the execution of this SPROC with a @parameter of customer first name and last name.

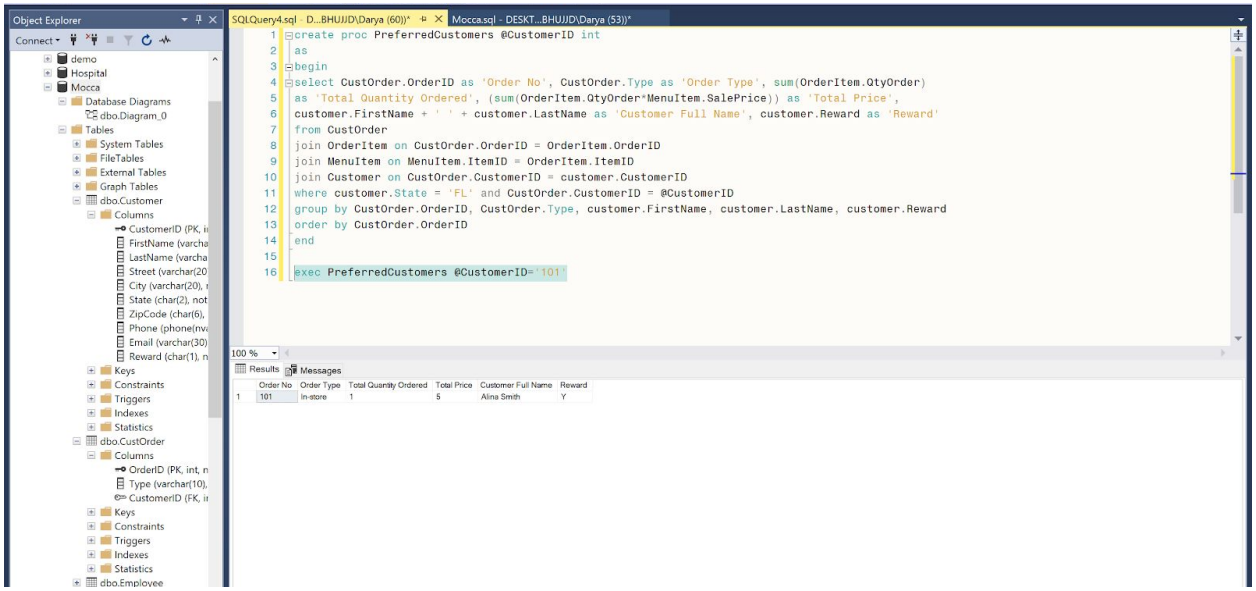


Stored Procedure 2: PreferredCustomers

This procedure combines given customers' order numbers, order types, calculates total amount of menu items ordered, calculates total price of the order, and prints customer's full name and whether he is a reward member. Customers are limited to those who are from Florida. This table can be used in selecting preferred customers - those who spend a lot and live nearby, to offer them to join rewards program.

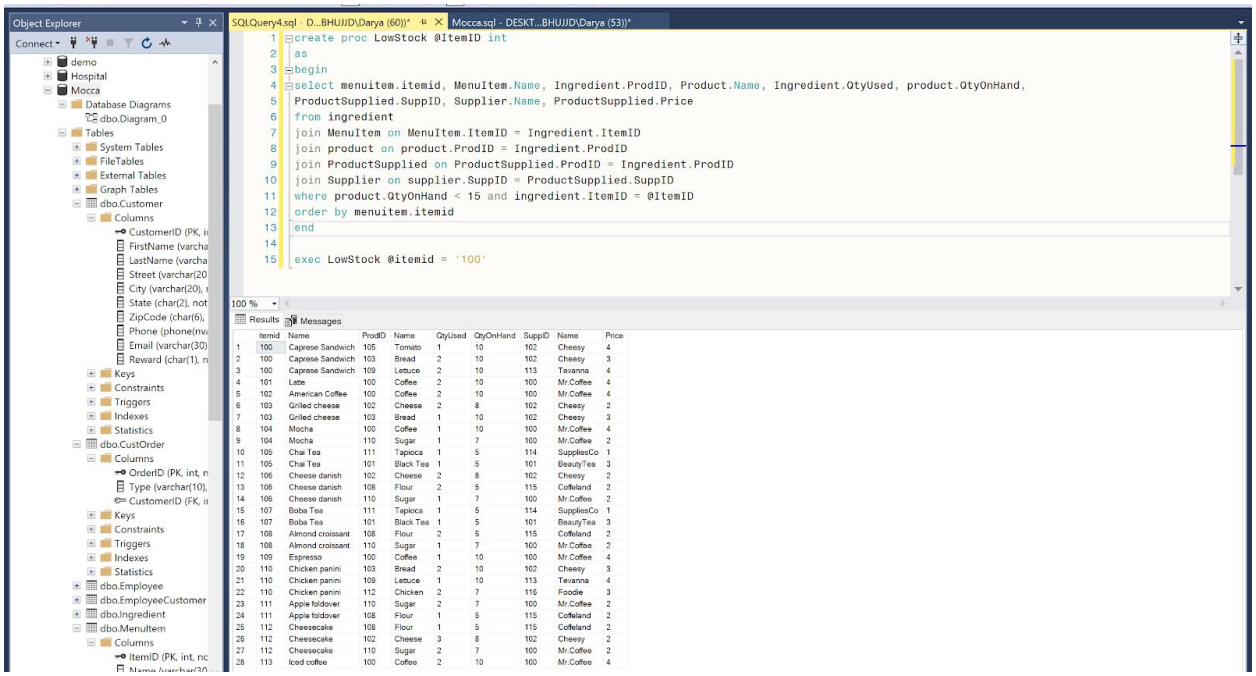


Below is the execution of this SPROC with a @parameter of customer ID.

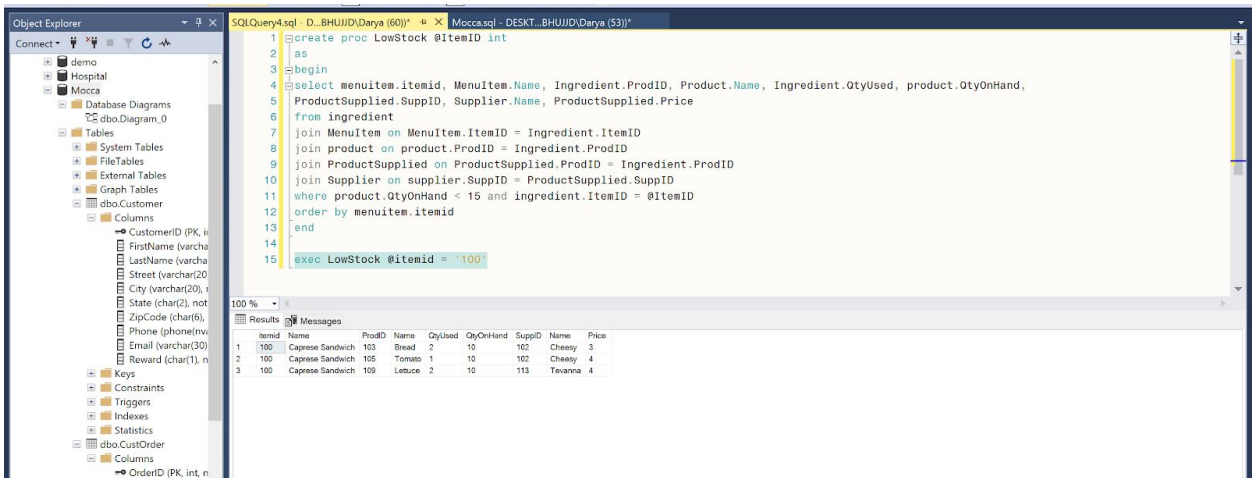


Stored Procedure 3: LowStock

This procedure is made for inventory control. It lists menu item IDs, their names, ingredients they are made of (IDs and names), quantity of ingredients used for the recipe of those menu items, remaining stock quantity of those ingredients, suppliers that deliver those ingredients (IDs and names), and ingredient price. The table is limited to those ingredients that have stock lower than 15. It will help to identify whether there are enough products for a particular menu item and who to order them from and for what price.



Below is the execution of this SPROC with a @parameter of item ID.



User Acceptance Test

Which order(s) have a latte and how many were order?

```
Select MenuItem.Name, MenuItem.ItemID, OrderItem.OrderID, OrderItem.QtyOrder
From MenuItem
Join OrderItem
on MenuItem.ItemID=OrderItem.ItemID
where
MenuItem.Name = 'Latte'
```

Which customer(s) are enrolled in the Rewards Program?

```
Select Customer.FirstName + ' ' + Customer.LastName as 'Customer Name', Customer.Reward
FROM Customer
Where Reward = 'Y'
```

Which products are used to make Caprese Sandwich? How much of each product?

```
Select Product.ProdID,Product.Name, Ingredient.QtyUsed, MenuItem.ItemID, MenuItem.Name
FROM Product
join Ingredient
on Product.ProdID=Ingredient.ProdID
join MenuItem
on Ingredient.ItemID=MenuItem.ItemID
where MenuItem.Name= 'Caprese Sandwich'
```