

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

<u>Version</u>	<u>Description</u>
1.0	First released draft
2.0	<ol style="list-style-type: none"> 1. Mocca ERD 2. Mocca EERD 3. Mocca Relational Schema 4. Mocca Data Dictionary Header 5. Mocca Data Dictionary
3.0	<p>Summary of changes:</p> <ol style="list-style-type: none"> 1. Updated ERD 2. Updated EERD 3. Updates RS 1. Updated Mocca Data Dictionary <ol style="list-style-type: none"> 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

	<ul style="list-style-type: none"> k. Created check for QtyProduct in Product Supply. Qty Product >=0 1. Created check for Type in Supplier Type Table. Type like [C] OR [R] 2. Rules: 3. Defaults: 3. SPROCS 4. Table Views
4.0	<p>Summary of changes:</p> <ul style="list-style-type: none"> 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 Café

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 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

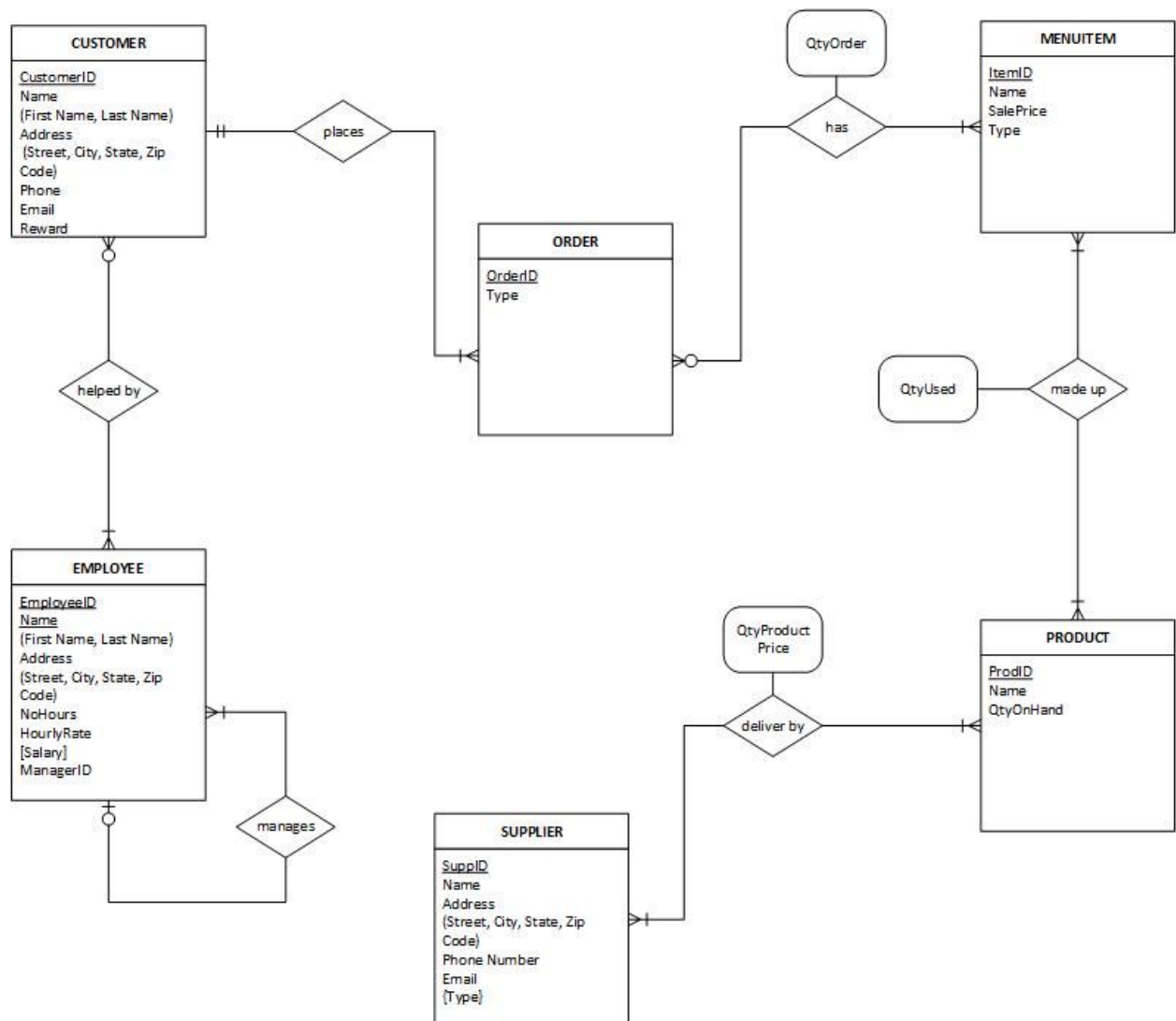
- PRODUCT

ProdID

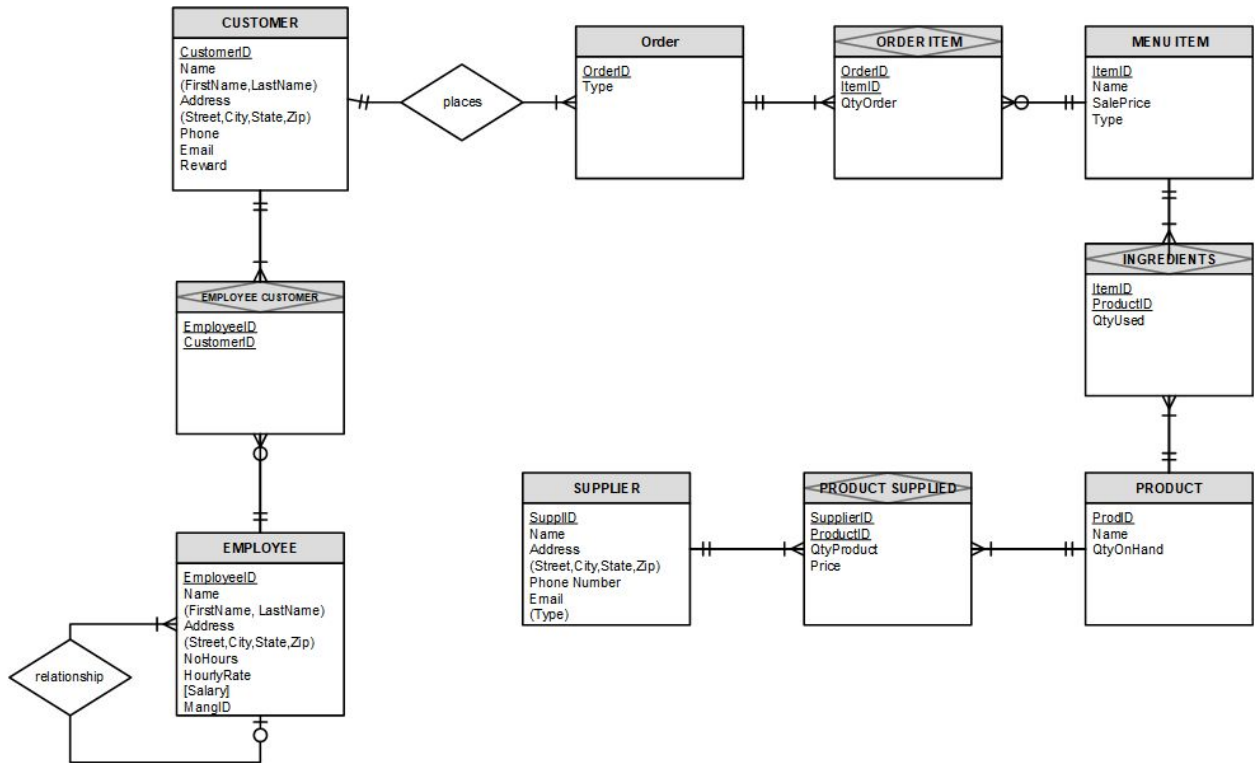
Name

QtyOnHand

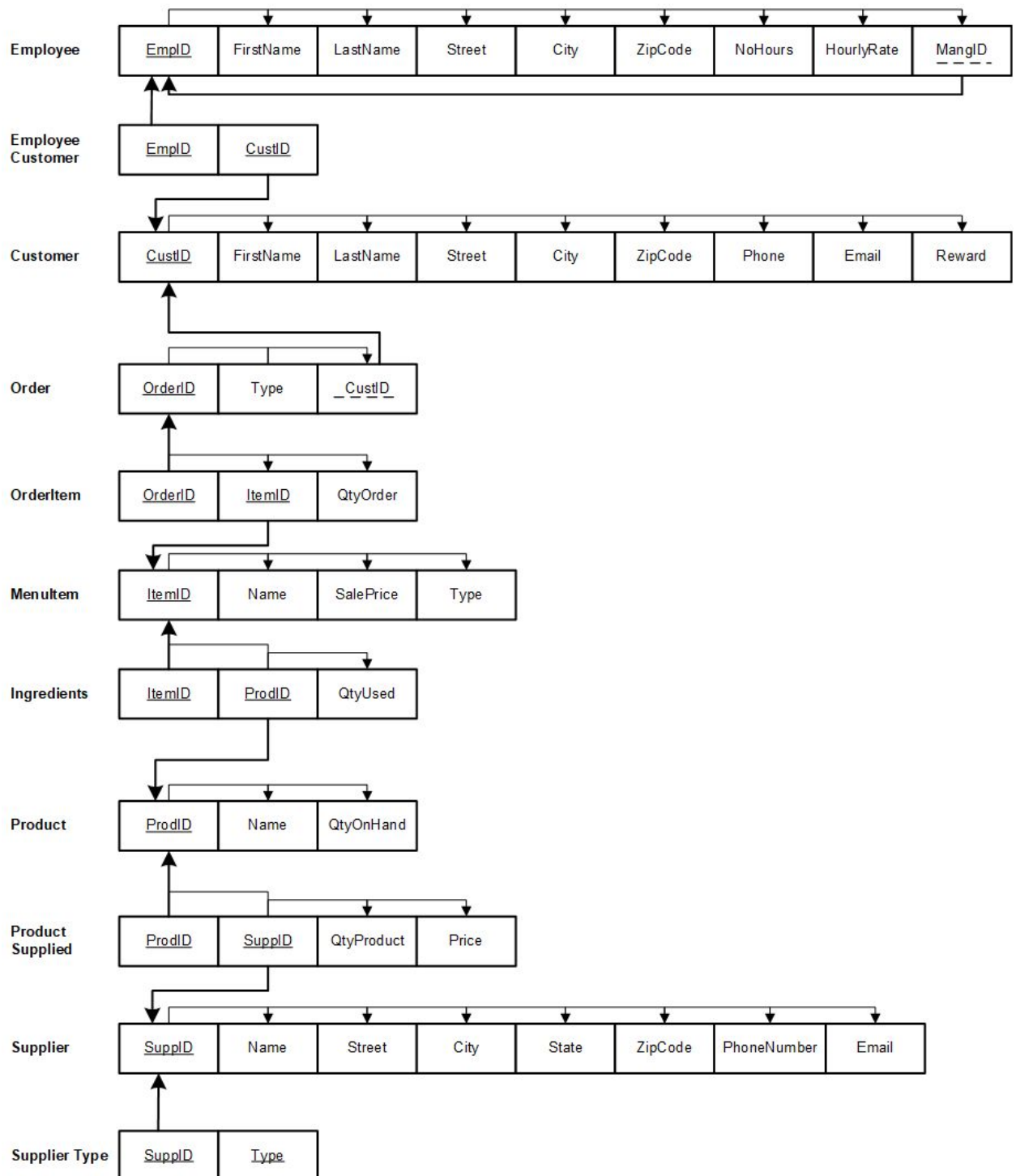
ERD



EERD



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	Varcha r	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	Uniqu e	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	Identit y	Uniqu e	Defaul t	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**

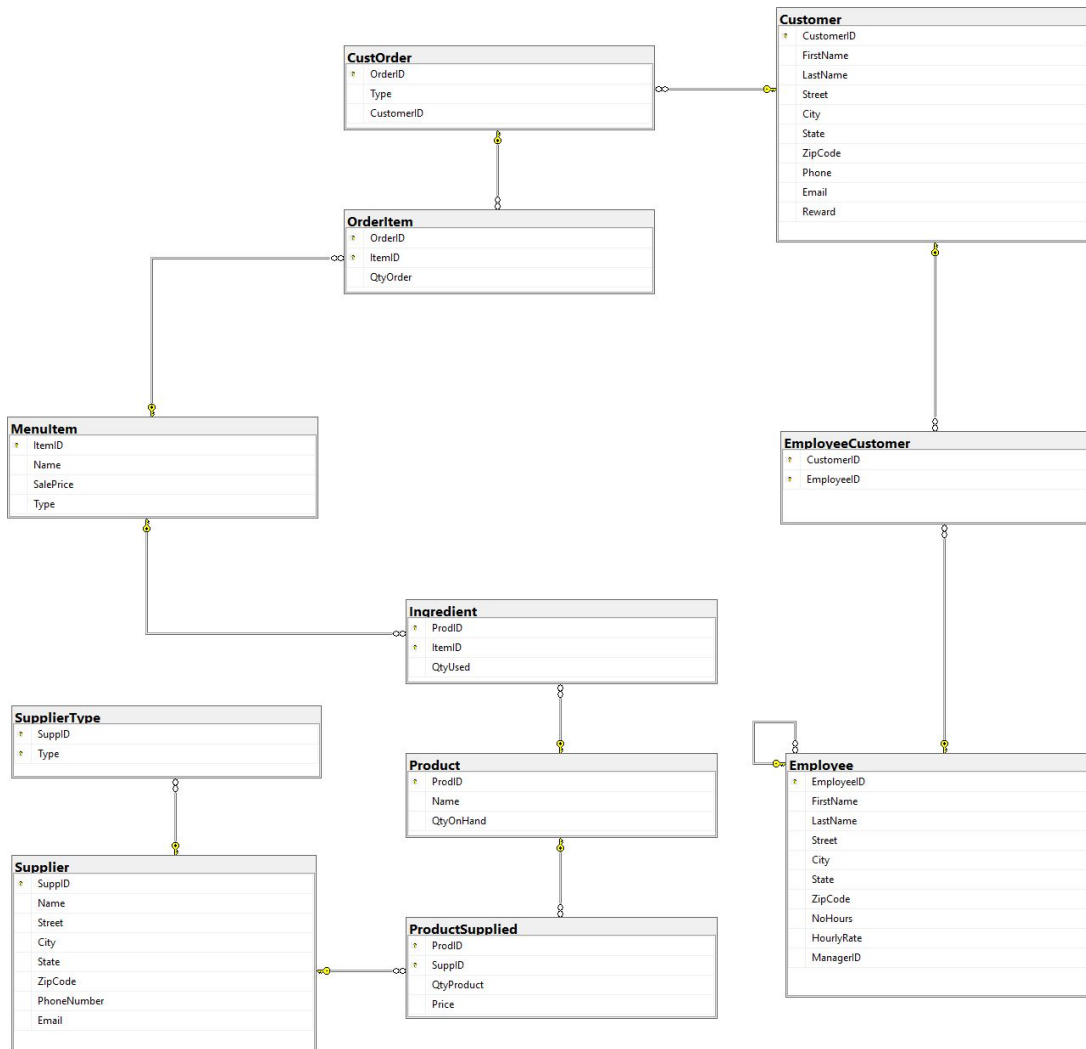
Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
SuppD	PK ; Unique sequential supplier ID number	int		Y						Y
Name	Name of the Supplier	varchar	20							

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**

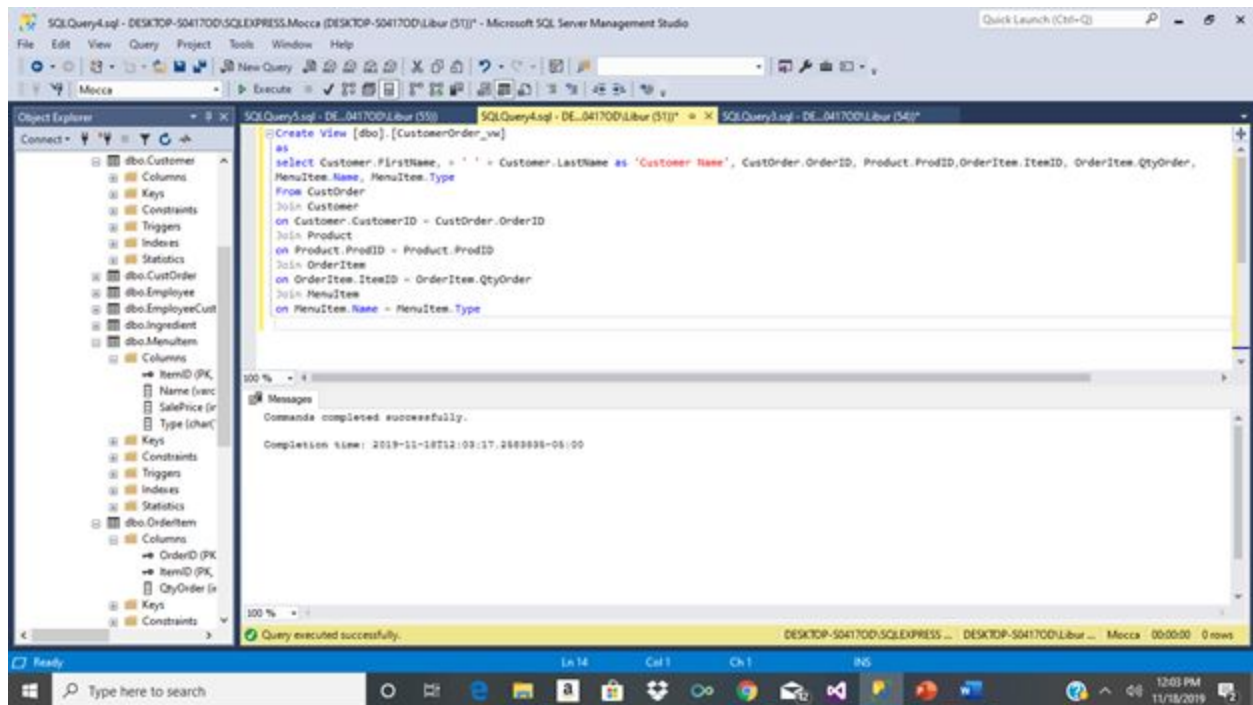
Column Name	Description	Data Type	Size	Identity	Unique	Default	Rule	Check	Allow Nulls	Index
SuppID	FK:PK; Foreign key to Supplier table	int		Y						Y
Type	Supplier Type	char	1					[Type] = (‘C’) OR [Type] = (‘R’)		

Database Diagram



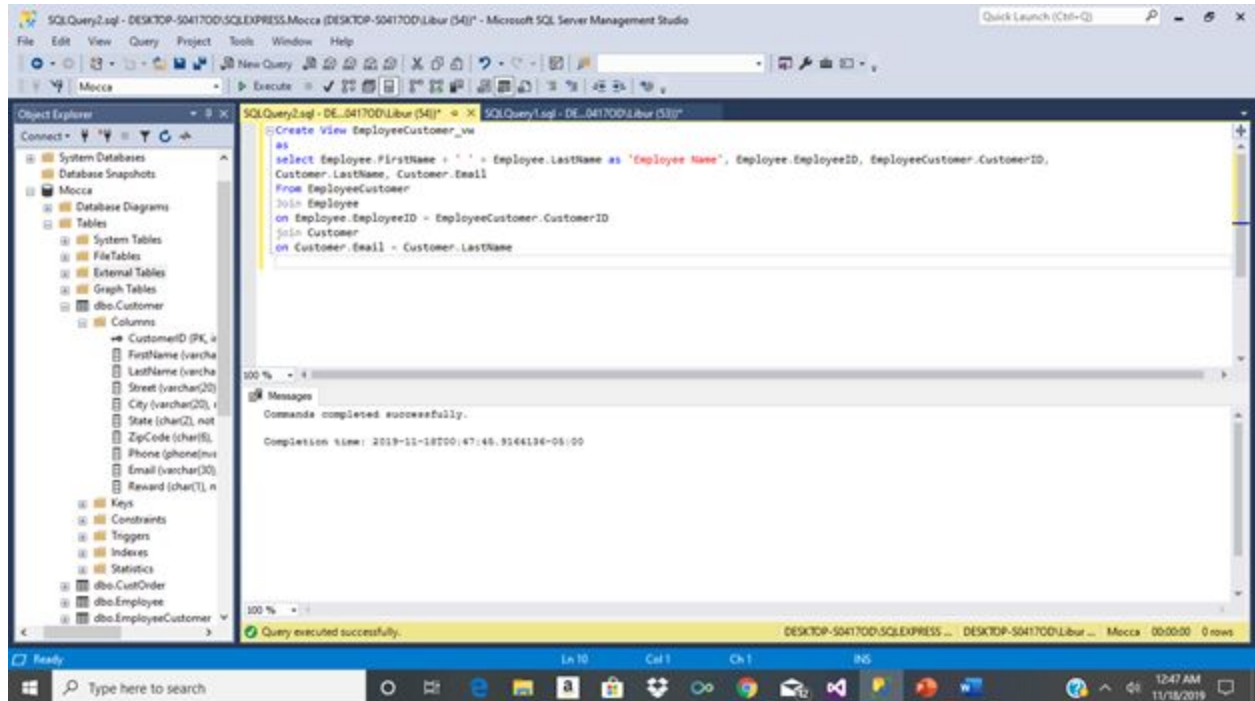
Customer Order View Description

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



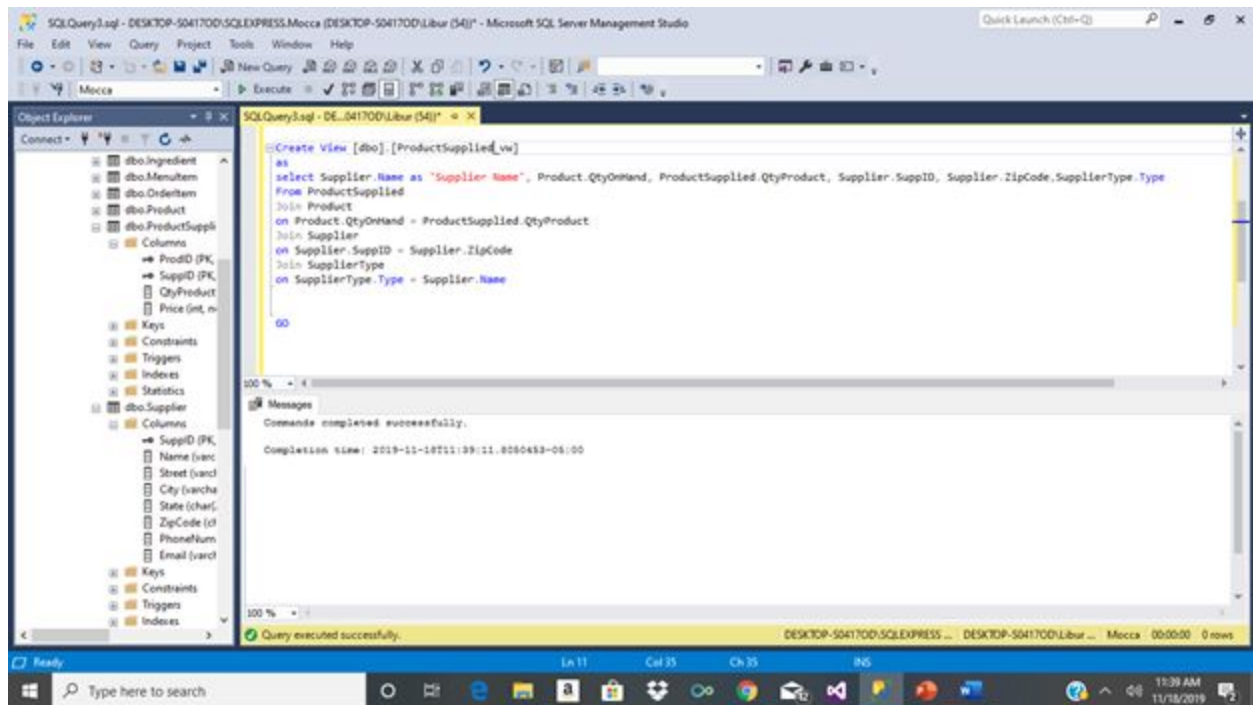
Employee Customer Relationship View Description

The Employee-Customer view joins the Employee, EmployeeCustomer, and Customer tables on their primary key, and foreign 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.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'Mocca', including tables like 'Customer', 'Employee', and 'EmployeeCustomer'. The central pane shows the SQL script for creating the 'CustomerEmployee' stored procedure. The script defines parameters for customer first and last names and includes a complex query joining 'Customer', 'Employee', and 'EmployeeCustomer' tables. The query filters for employees with a supervisor (ManagerID is not null) and matches the provided customer names. It calculates total compensation as the product of hourly rate and number of hours, and counts the number of orders taken. The results pane at the bottom shows the output of the procedure, which is a table with 6 columns: Customer No, Customer Full Name, Employee No, Employee Full Name, Total Compensation, and Number of Orders Taken. The results show 11 rows of data for various customers and employees.

```
1 create proc CustomerEmployee @FirstName varchar(20), @LastName varchar(20)
2 as
3 begin
4 select custorder.CustomerID as 'Customer No', customer.FirstName + ' ' + customer.LastName as 'Customer Full Name',
5 EmployeeCustomer.EmployeeID as 'Employee No', employee.FirstName + ' ' + employee.LastName as 'Employee Full Name',
6 (employee.HourlyRate * Employee.NoHours) as 'Total Compensation', count(custorder.orderid) as 'Number of Orders Taken'
7 from EmployeeCustomer
8 join employee on employee.EmployeeID = EmployeeCustomer.EmployeeID
9 join customer on Customer.CustomerID = EmployeeCustomer.CustomerID
10 join CustOrder on CustOrder.CustomerID = Customer.CustomerID
11 where Employee.ManagerID is not null and customer.FirstName = @FirstName and customer.LastName = @LastName
12 group by customer.customerID, custorder.CustomerID, customer.FirstName, customer.LastName, EmployeeCustomer.EmployeeID,
13 employee.FirstName, Employee.LastName, employee.HourlyRate, Employee.NoHours
14 order by customer.customerID
15 end
16
17 exec CustomerEmployee @FirstName='Sam', @LastName='Brown'
```

Customer No	Customer Full Name	Employee No	Employee Full Name	Total Compensation	Number of Orders Taken	
1	100	Carla Vander	101	Richard Chadler	200	3
2	101	Anna Smith	101	Richard Chadler	200	1
3	102	Anna Martinez	100	William Connolly	400	1
4	103	Luisa Pleysher	102	Natalie Cozy	600	2
5	104	Ronald Hernandez	101	Richard Chadler	200	1
6	105	Sam Brown	111	Aaron Wilde	200	4
7	118	Mabel Brubaker	101	Richard Chadler	200	2
8	119	Matthew Salley	111	Aaron Wilde	200	2
9	121	Julieann Novakowski	111	Aaron Wilde	200	2
10	122	Nathan Pineda	113	Mary Menard	240	3
11	123	Luke Durgin	112	Gladys Sain	480	1

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

This screenshot shows the same SQL Server Enterprise Manager interface as the previous one, but with the execution of the 'CustomerEmployee' stored procedure. The SQL script in the central pane is identical, but the execution command at the bottom is highlighted: 'exec CustomerEmployee @FirstName='Sam', @LastName='Brown''. The results pane at the bottom now displays only one row of data, corresponding to the specified parameters: Customer No 105, Customer Full Name Sam Brown, Employee No 111, Employee Full Name Aaron Wilde, Total Compensation 200, and Number of Orders Taken 4.

```
1 create proc CustomerEmployee @FirstName varchar(20), @LastName varchar(20)
2 as
3 begin
4 select custorder.CustomerID as 'Customer No', customer.FirstName + ' ' + customer.LastName as 'Customer Full Name',
5 EmployeeCustomer.EmployeeID as 'Employee No', employee.FirstName + ' ' + employee.LastName as 'Employee Full Name',
6 (employee.HourlyRate * Employee.NoHours) as 'Total Compensation', count(custorder.orderid) as 'Number of Orders Taken'
7 from EmployeeCustomer
8 join employee on employee.EmployeeID = EmployeeCustomer.EmployeeID
9 join customer on Customer.CustomerID = EmployeeCustomer.CustomerID
10 join CustOrder on CustOrder.CustomerID = Customer.CustomerID
11 where Employee.ManagerID is not null and customer.FirstName = @FirstName and customer.LastName = @LastName
12 group by customer.customerID, custorder.CustomerID, customer.FirstName, customer.LastName, EmployeeCustomer.EmployeeID,
13 employee.FirstName, Employee.LastName, employee.HourlyRate, Employee.NoHours
14 order by customer.customerID
15 end
16
17 exec CustomerEmployee @FirstName='Sam', @LastName='Brown'
```

Customer No	Customer Full Name	Employee No	Employee Full Name	Total Compensation	Number of Orders Taken	
1	105	Sam Brown	111	Aaron Wilde	200	4

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.

```
1 create proc PreferredCustomers @CustomerID int
2 as
3 begin
4 select CustOrder.OrderID as 'Order No', CustOrder.Type as 'Order Type', sum(OrderItem.QtyOrder)
5 as 'Total Quantity Ordered', (sum(OrderItem.QtyOrder*Menuitem.SalePrice)) as 'Total Price',
6 customer.FirstName + ' ' + customer.LastName as 'Customer Full Name', customer.Reward as 'Reward'
7 from CustOrder
8 join OrderItem on CustOrder.OrderID = OrderItem.OrderID
9 join MenuItem on MenuItem.ItemID = OrderItem.ItemID
10 join Customer on CustOrder.CustomerID = customer.CustomerID
11 where customer.State = 'FL' and CustOrder.CustomerID = @CustomerID
12 group by CustOrder.OrderID, CustOrder.Type, customer.LastName, customer.FirstName, customer.Reward
13 order by CustOrder.OrderID
14 end
15
16 exec PreferredCustomers @CustomerID='101'
```

Order No	Order Type	Total Quantity Ordered	Total Price	Customer Full Name	Reward
1	Online	3	9	Carla Vander	N
2	101	1	5	Alina Smith	Y
3	102	3	12	Anna Martinez	N
4	104	3	11	Luisa Playsher	N
5	105	1	5	Luisa Playsher	N
6	106	4	10	Ronald Hernandez	Y
7	107	2	6	Sam Brown	N
8	108	2	4	Sam Brown	N
9	109	2	5	Sam Brown	N
10	110	1	5	Mabel Brubaker	N
11	111	4	10	Mabel Brubaker	N
12	112	2	6	Mathew Salfey	Y
13	116	1	3	Julieann Nowakowski	N
14	117	2	8	Julieann Nowakowski	N
15	119	1	2	Sam Brown	N
16	122	3	12	Mathew Salfey	Y
17	123	4	10	Carla Vander	N
18	124	2	10	Luke Dugan	N

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

```
1 create proc PreferredCustomers @CustomerID int
2 as
3 begin
4 select CustOrder.OrderID as 'Order No', CustOrder.Type as 'Order Type', sum(OrderItem.QtyOrder)
5 as 'Total Quantity Ordered', (sum(OrderItem.QtyOrder*Menuitem.SalePrice)) as 'Total Price',
6 customer.FirstName + ' ' + customer.LastName as 'Customer Full Name', customer.Reward as 'Reward'
7 from CustOrder
8 join OrderItem on CustOrder.OrderID = OrderItem.OrderID
9 join MenuItem on MenuItem.ItemID = OrderItem.ItemID
10 join Customer on CustOrder.CustomerID = customer.CustomerID
11 where customer.State = 'FL' and CustOrder.CustomerID = @CustomerID
12 group by CustOrder.OrderID, CustOrder.Type, customer.LastName, customer.FirstName, customer.Reward
13 order by CustOrder.OrderID
14 end
15
16 exec PreferredCustomers @CustomerID='101'
```

Order No	Order Type	Total Quantity Ordered	Total Price	Customer Full Name	Reward
1	101	1	5	Alina Smith	Y

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.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure, including tables, views, and stored procedures. The central pane shows the SQL query editor with the following code:

```
1 create proc LowStock @ItemID int
2 as
3 begin
4 select menuitem.itemid, MenuItem.Name, Ingredient.ProdID, Product.Name, Ingredient.QtyUsed, product.QtyOnHand,
5 ProductSupplied.SuppID, Supplier.Name, ProductSupplied.Price
6 from ingredient
7 join MenuItem on MenuItem.ItemID = Ingredient.ItemID
8 join product on product.ProdID = Ingredient.ProdID
9 join ProductSupplied on ProductSupplied.ProdID = Ingredient.ProdID
10 join Supplier on supplier.SuppID = ProductSupplied.SuppID
11 where product.QtyOnHand < 15 and ingredient.ItemID = @ItemID
12 order by menuitem.itemid
13 end
14
15 exec LowStock @Itemid = '100'
```

The bottom pane shows the results of the execution, displaying a table with the following data:

ItemID	Name	ProdID	Name	QtyUsed	QtyOnHand	SuppID	Name	Price	
1	100	Caprese Sandwich	105	Tomato	1	10	102	Cheesy	4
2	100	Caprese Sandwich	103	Bread	2	10	102	Cheesy	3
3	100	Caprese Sandwich	109	Lettuce	2	10	113	Tevanna	4
4	101	Latte	100	Coffee	2	10	100	Mr.Coffee	4
5	102	American Coffee	100	Coffee	2	10	100	Mr.Coffee	4
6	103	Grilled cheese	102	Cheese	2	8	102	Cheesy	2
7	103	Grilled cheese	103	Bread	1	10	102	Cheesy	3
8	104	Mocha	100	Coffee	1	10	100	Mr.Coffee	4
9	104	Mocha	110	Sugar	1	7	100	Mr.Coffee	2
10	105	Chai Tea	111	Tapoca	1	5	114	SuppliesCo	1
11	105	Chai Tea	101	Black Tea	1	5	101	BeautyTea	3
12	105	Cheese danish	102	Cheese	2	8	102	Cheesy	2
13	106	Cheese danish	108	Flour	2	5	115	Coffeland	2
14	106	Cheese danish	110	Sugar	1	7	100	Mr.Coffee	2
15	107	Bobo Tea	111	Tapoca	1	5	114	SuppliesCo	1
16	107	Bobo Tea	101	Black Tea	1	5	101	BeautyTea	3
17	108	Almond croissant	108	Flour	2	5	115	Coffeland	2
18	108	Almond croissant	110	Sugar	1	7	100	Mr.Coffee	2
19	109	Espresso	100	Coffee	1	10	100	Mr.Coffee	4
20	110	Chicken panini	103	Bread	2	10	102	Cheesy	3
21	110	Chicken panini	109	Lettuce	1	10	113	Tevanna	4
22	110	Chicken panini	112	Chicken	2	7	116	Foodie	3
23	111	Apple foldover	110	Sugar	2	7	100	Mr.Coffee	2
24	111	Apple foldover	108	Flour	1	5	115	Coffeland	2
25	112	Cheesecake	108	Flour	1	5	115	Coffeland	2
26	112	Cheesecake	102	Cheese	3	8	102	Cheesy	2
27	112	Cheesecake	110	Sugar	2	7	100	Mr.Coffee	2
28	113	Iced coffee	100	Coffee	2	10	100	Mr.Coffee	4

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

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure. The central pane shows the SQL query editor with the following code:

```
1 create proc LowStock @ItemID int
2 as
3 begin
4 select menuitem.itemid, MenuItem.Name, Ingredient.ProdID, Product.Name, Ingredient.QtyUsed, product.QtyOnHand,
5 ProductSupplied.SuppID, Supplier.Name, ProductSupplied.Price
6 from ingredient
7 join MenuItem on MenuItem.ItemID = Ingredient.ItemID
8 join product on product.ProdID = Ingredient.ProdID
9 join ProductSupplied on ProductSupplied.ProdID = Ingredient.ProdID
10 join Supplier on supplier.SuppID = ProductSupplied.SuppID
11 where product.QtyOnHand < 15 and ingredient.ItemID = @ItemID
12 order by menuitem.itemid
13 end
14
15 exec LowStock @Itemid = '100'
```

The bottom pane shows the results of the execution, displaying a table with the following data:

ItemID	Name	ProdID	Name	QtyUsed	QtyOnHand	SuppID	Name	Price	
1	100	Caprese Sandwich	103	Bread	2	10	102	Cheesy	3
2	100	Caprese Sandwich	105	Tomato	1	10	102	Cheesy	4
3	100	Caprese Sandwich	109	Lettuce	2	10	113	Tevanna	4

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'