**Project: Jummie Burgers**

**Felipe Bedoya**

**Company Overview and Description:**

Welcome to “Jummie Burgers”. Our cozy restaurant invites you to embark on a delightful culinary adventure. Our menu, thoughtfully curated to provoke your taste buds, features a star attraction: the Jummie Classic. Crafted with the finest ingredients and bursting with flavor, each bite of our Jummie Burger promises a satisfying experience. Whether you’re a burger enthusiast or simply seeking a hearty meal, our Jummie Burger won’t disappoint. Accompanied by our signature sides and refreshing beverages, a visit to Jummie Burgers guarantees an unforgettable dining journey.

But that’s not all! Our diverse menu caters to every palate. From savory appetizers to indulgent desserts, our skilled chefs prepare each hamburger with passion and precision. Whether you’re enjoying a casual lunch with friends or a special dinner with loved ones, Jummie Burgers invites you to savor excellence in every bite.

Beyond the food, we’re committed to creating memorable moments for our guests. Our attentive staff ensures that your dining experience is exceptional, whether you’re celebrating a special occasion or simply unwinding after a long day. So, step into our welcoming atmosphere, where passion for food meets unparalleled hospitality. At Jummie Burgers, every visit is a journey to culinary paradise.

In addition to our commitment to culinary excellence and impeccable service, Jummie Burgers is proud to prioritize sustainability and community involvement. We source our ingredients locally whenever possible, supporting local farmers and businesses while reducing our carbon footprint. Through initiatives such as food donation programs and environmentally conscious practices, we strive to give back to the community that has embraced us. At Jummie Burgers, we believe in not only satisfying appetites but also making a positive impact, one delicious meal at a time. Join us in our mission to savor good food, foster connections, and contribute to a brighter future for all.

**Menu**







**Receipt Sample**

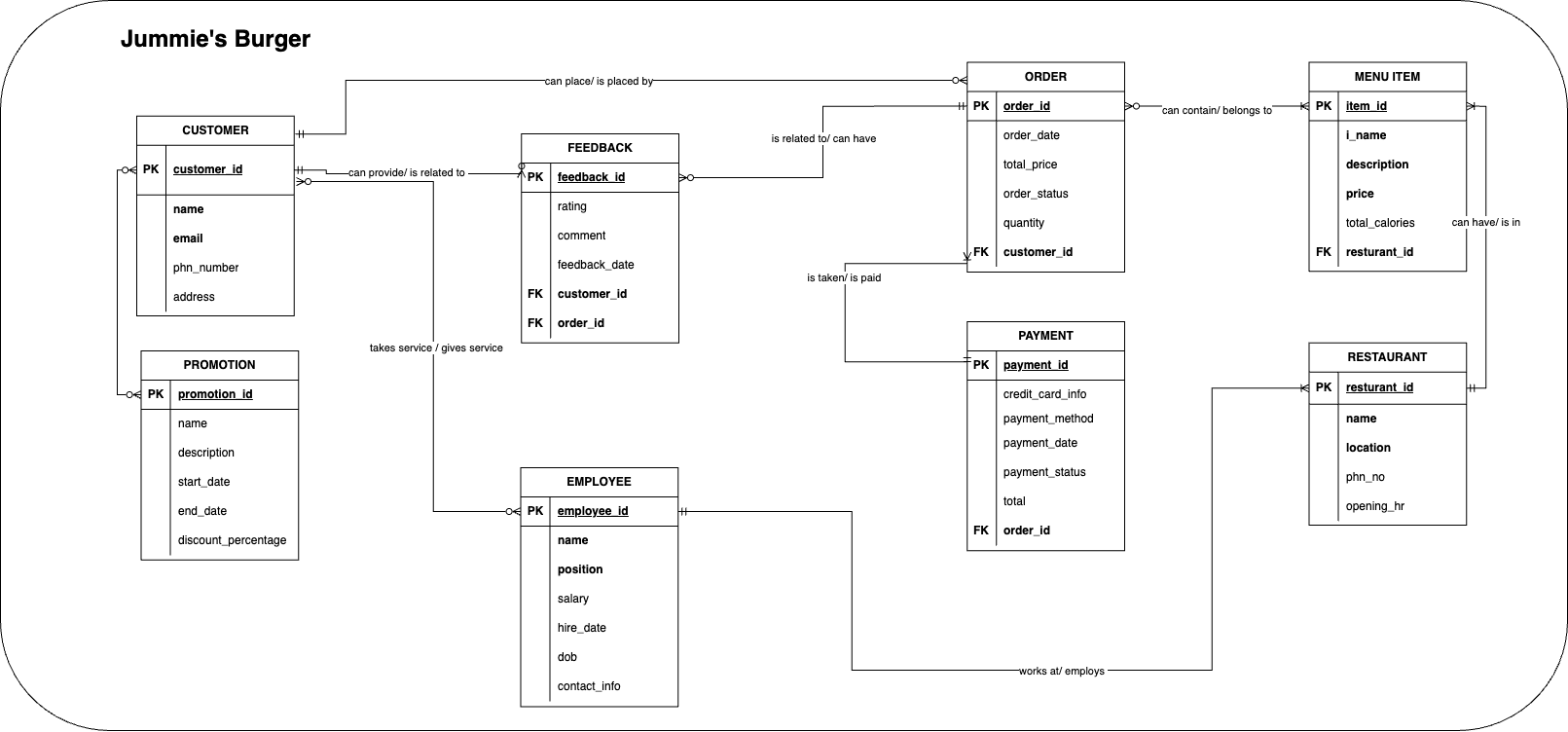
A screenshot of a menu

Description automatically generated

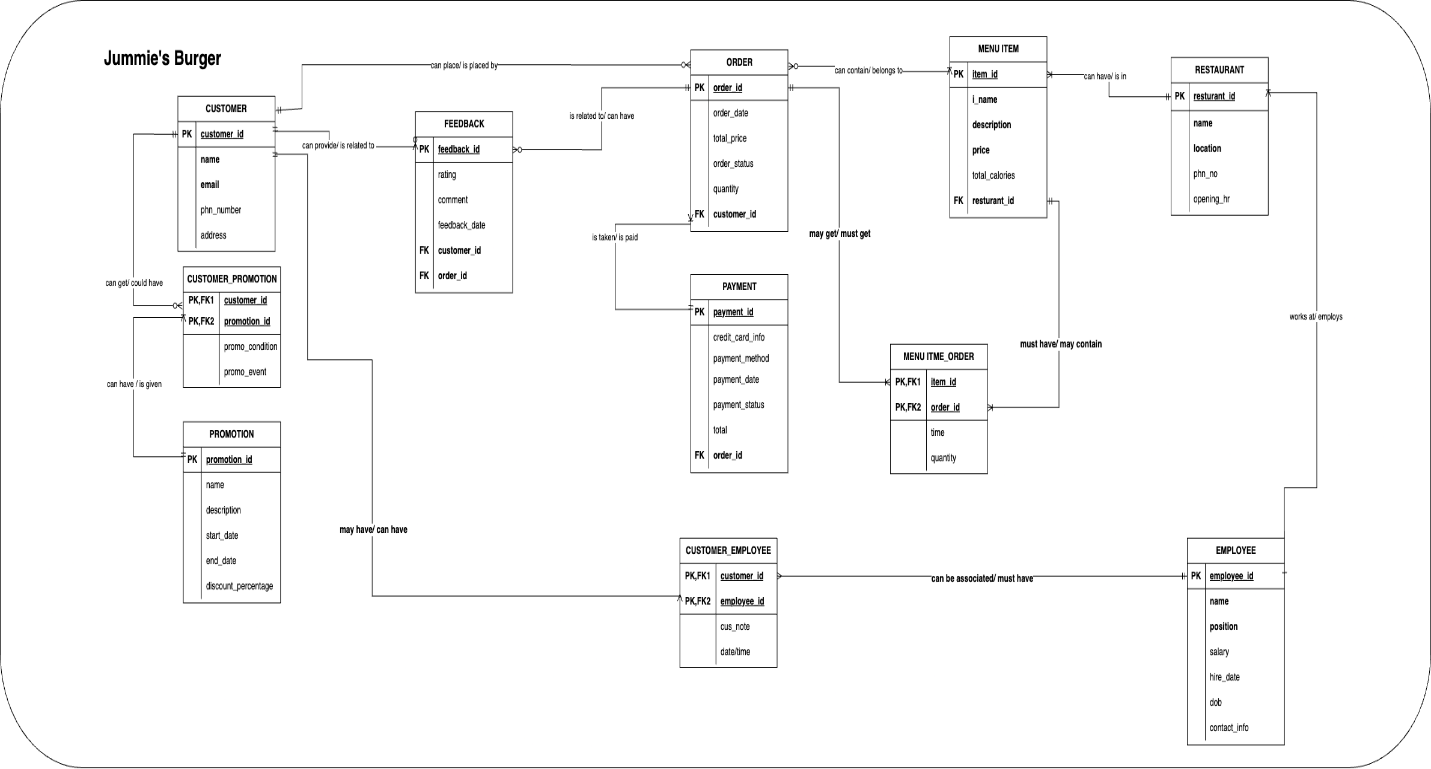
**Satisfaction Survey**



**Entity Relationship Diagram - Many to many relationship (Unresolved)**

****

**Entity Relationship Diagram - Many to many relationship (Resolved)**

****

**ERD Language**

RESTAURANT - MENU ITEM:

- each RESTAURANT may have one or more MENU ITEMs

- each MENU ITEM must be in one and only one RESTAURANT

RESTAURANT - EMPLOYEE:

- each RESTAURANT may have one or more EMPLOYEEs

- each EMPLOYEE must work in one and only one restaurant

ORDER - MENU ITEM:

- each MENU ITEM may get one or more ORDERs

- each ORDER may contain zero, one or many MENU ITEMs

CUSTOMER - ORDER:

- each CUSTOMER may place zero, one or more ORDERs

- each ORDER may be placed by one and only one CUSTOMER

ORDER - PAYMENT:

- each ORDER may have one and only one PAYMENT

- each PAYMENT may have one or more ORDER

CUSTOMER - FEEDBACK:

- each CUSTOMER may give zero, one or more FEEDBACK

- each FEEDBACK must be given by one and only one CUSTOMER

FEEDBACK - ORDER:

- each FEEDBACK may be related to one and only one ORDER

- each ORDER may have zero, one or more FEEDBACKs

CUSTOMER - CUSTOMER\_PROMOTION - PROMOTION:

- each CUSTOMER can get zero, one or more CUSTOMER\_PROMOTIONs

- each CUSTOMER\_PROMOTION could be given to one and only one CUSTOMER

- each CUSTOMER\_PROMOTION can have one and only one PROMOTION

- each PROMOTION can be given to one or more CUSTOMER\_PROMOTION

CUSTOMER - CUSTOMER\_EMPLOYEE - EMPLOYEE:

- each CUSTOMER may have one and only one CUSTOMER\_EMPLOYEE

- each CUSTOMER\_EMPLOYEE can have one or many CUSTOMERs

- each CUSTOMER\_EMPLOYEE can be associated with one or many EMPLOYEEs

- each EMPLOYEE must have one and only one CUSTOMER\_EMPLOYEE

MENU ITEM - MENU ITME\_ORDER - ORDER:

- each MENU ITEM must have one and only one MENU ITEM\_ORDER

- each MENU ITEM\_ORDER may contain one or many MENU ITEMs

- each MENU ITEM\_ORDER may get one or many ORDERs

- each ORDER must get one and only one MENU ITEM\_ORDER

**Business Rules**

**-** Each menu item has a unique ID, so it cannot have duplicate names or identifiers.

- Each order must have at least one menu item and cannot be modified after they are placed.

- Each customer gets a unique id for the order they have made, and they must have unique phone number and email address

- The customers can place multiple orders

- Each employee must have a unique ID and phone number

- the opening hours of the restaurants may be same or different.

- the promotions can be applied to specific menu items and they must have a valid start and end dates

- the promotion can have different conditions in regard to the customers

- the payment can be done via different methods like cash or credit/debit cards

- the customers have the choice to give feedback on the customer service and the order that they have made

**Relational Data Model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Restaurant | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | RestaurantID | Integer | 5 |
|  | \* | Name | Varchar | 50 |
|  | \* | Location | Varchar | 50 |
|  | 0 | PhoneNumber | Varchar | 10 |
|  | 0 | OpeningHours | Time | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | CustomerID | Integer | 5 |
|  | \* | Name | Varchar | 50 |
|  | \* | Email | Varchar | 50 |
|  | 0 | PhoneNumber | Varchar | 10 |
|  | 0 | Address | Varchar | 10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | OrderID | Integer | 5 |
|  | 0 | OrderDate | Date | 10 |
|  | 0 | Quantity | Integer | 5 |
|  | 0 | TotalPrice | Decimal | (10,2) |
| FK | \* | CustomerID | Integer | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Payment | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | PaymentID | Integer | 10 |
|  | \* | CreditCardInformation | Varchar | 20 |
|  | \* | PaymentMethod | Varchar | 50 |
|  | 0 | PaymentDate | Date | 10 |
|  | 0 | PaymentStatus | Varchar | 50 |
| FK | 0 | OrderID | Integer | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MenuItem | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | ItemID | Integer | 5 |
|  | \* | ItemName | Varchar | 50 |
|  | \* | Description | Text | 50 |
|  | 0 | Price | Decimal | (10,2) |
|  | 0 | TotalCalories | Integer | 5 |
| FK | 0 | RestaurantID | Integer | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MenuItemOrder | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | MenuItemOrderID | Integer | 5 |
| FK | \* | ItemID | Integer | 5 |
| FK | \* | OrderID | Integer | 5 |
|  | 0 | Quantity | Integer | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Employee | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | EmployeeID | Integer | 5 |
|  | \* | Name | Varchar | 50 |
|  | \* | Position | Varchar | 50 |
|  | 0 | Salary | Decimal | (10,2) |
|  | 0 | HireDate | Date | 10 |
|  |  | DateOfBirth | Date | 10 |
|  |  | ContactInfo | Varchar | (50) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feedback | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | FeedbackID | Integer | 5 |
| FK | \* | CustomerID | Integer | 5 |
| FK | \* | OrderID | Integer | 5 |
|  | 0 | Rating | Integer | 5 |
|  | 0 | Comment | Text | 50 |
|  |  | FeedbackDate | Date | 10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Promotion | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | PromotionID | Integer | 5 |
|  |  | Name | Varchar | 50 |
|  |  | Description | Text | 50 |
|  | 0 | StartDate | Date | 10 |
|  | 0 | EndDate | Date | 10 |
|  |  | DiscountPercentage | Decimal | (5,2) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CustomerEmployee | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | CustomerEmployeeID | Integer | 5 |
|  |  | CustomerID | Integer | 5 |
|  |  | EmployeeID | Integer | 5 |
|  | 0 | CustomerNote | Varchar | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CustomerPromotion | | | | |
| Key Type | Optionality | Column Name | Data Type | Length |
| PK | \* | CustomerPromotionID | Integer | 5 |
|  |  | CustomerID | Integer | 5 |
|  |  | PromotionID | Integer | 5 |
|  | 0 | PromotionCondition | Varchar | 50 |
|  | 0 | PromoEvent | Varchar | 50 |

**Tables with dummy data**

Restaurant



Customer



Order



Payment



MenuItem



MenuItemOrder



Employee



Feedback



Promotion



CustomerEmployee



CustomerPromotion



**SQL queries for Menu and Receipt**

CREATE TABLE Burgers (

BurgerName VARCHAR(50),

Calories INT,

Carbs INT,

TotalFat INT,

Sodium INT

);

INSERT INTO Burgers (BurgerName, Calories, Carbs, TotalFat, Sodium) VALUES ('Jummie Classic', 840, 38, 42, 420);

INSERT INTO Burgers (BurgerName, Calories, Carbs, TotalFat, Sodium) VALUES ('Jummie Cheese', 985, 43, 56, 1060);

INSERT INTO Burgers (BurgerName, Calories, Carbs, TotalFat, Sodium) VALUES ('Jummie Bacon', 925, 40, 51, 670);

INSERT INTO Burgers (BurgerName, Calories, Carbs, TotalFat, Sodium) VALUES ('Jummie Little', 540, 40, 27, 375);

CREATE TABLE Fries (

FriesName VARCHAR(50),

Calories INT,

Carbs INT,

TotalFat INT,

Sodium INT

);

INSERT INTO Fries (FriesName, Calories, Carbs, TotalFat, Sodium) VALUES ('Jummie Fries', 750, 132, 42, 963);

INSERT INTO Fries (FriesName, Calories, Carbs, TotalFat, Sodium) VALUES ('Jummie Cajun', 750, 135, 43, 963);

CREATE TABLE Beverages (

BeverageName VARCHAR(50),

Calories INT,

Carbs INT,

TotalFat INT,

Sodium INT

);

INSERT INTO Beverages (BeverageName, Calories, Carbs, TotalFat, Sodium) VALUES ('Coke', 361, 99, 0, 86);

INSERT INTO Beverages (BeverageName, Calories, Carbs, TotalFat, Sodium) VALUES ('Tea', 281, 76, 0, 86);

CREATE TABLE Desserts (

DessertName VARCHAR(50),

Calories INT,

Carbs INT,

TotalFat INT,

Sodium INT

);

INSERT INTO Desserts (DessertName, Calories, Carbs, TotalFat, Sodium) VALUES ('Ice Cream', 242, 41, 0, 155);

INSERT INTO Desserts (DessertName, Calories, Carbs, TotalFat, Sodium) VALUES ('Apple Pie', 271, 33, 15, 180);

A screenshot of a computer

Description automatically generated

CREATE TABLE Receipt (

OrderID VARCHAR(50),

RestaurantID VARCHAR(50),

ProductID INT,

MenuItem VARCHAR(50),

Quantity INT,

Price DECIMAL(5,2)

);

INSERT INTO Receipt (OrderID, RestaurantID, ProductID, MenuItem, Quantity, Price) VALUES ('09101', '3942', 8, 'Jummie Classic', 1, 4.99);

INSERT INTO Receipt (OrderID, RestaurantID, ProductID, MenuItem, Quantity, Price) VALUES ('09101', '3942', 7, 'Jummie Cheese', 3, 6.50);

INSERT INTO Receipt (OrderID, RestaurantID, ProductID, MenuItem, Quantity, Price) VALUES ('09101', '3942', 9, 'Jummie Bacon', 1, 8.12);

INSERT INTO Receipt (OrderID, RestaurantID, ProductID, MenuItem, Quantity, Price) VALUES ('09101', '3942', 2, 'Jummie Fries', 4, 2.00);

INSERT INTO Receipt (OrderID, RestaurantID, ProductID, MenuItem, Quantity, Price) VALUES ('09101', '3942', 5, 'Coke', 2, 1.75);

SELECT

SUM(Price \* Quantity) AS SubTotal,

SUM(Price \* Quantity) \* 0.13 AS Tax,

SUM(Price \* Quantity) + (SUM(Price \* Quantity) \* 0.13) AS Total

FROM Receipt

WHERE OrderID = '09101' AND RestaurantID = '3942';

A screenshot of a computer

Description automatically generated

**DDL queries for tables and Alter commands**

CREATE TABLE Restaurant (

RestaurantID INT NOT NULL,

Name VARCHAR(50),

Location VARCHAR(50),

PhoneNumber VARCHAR(20) NOT NULL,

OpeningHours TIME

);

ALTER TABLE Restaurant

ADD CONSTRAINT pk\_restaurant\_id PRIMARY KEY (RestaurantID);

CREATE TABLE Customer (

CustomerID INT NOT NULL,

Name VARCHAR(50),

Email VARCHAR(50),

PhoneNumber VARCHAR(20),

Address VARCHAR(50)

);

ALTER TABLE Customer

ADD CONSTRAINT pk\_CustomerID PRIMARY KEY (CustomerID);

CREATE TABLE [Order] (

OrderID INT NOT NULL,

OrderDate DATE,

Quantity INT,

TotalPrice DECIMAL(10, 2),

CustomerID INT NOT NULL

);

ALTER TABLE [Order]

ADD CONSTRAINT pk\_order\_id PRIMARY KEY (OrderID);

ALTER TABLE [Order]

ADD CONSTRAINT FK\_Order\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

CREATE TABLE Payment (

PaymentID INT NOT NULL,

CreditCardInformation VARCHAR(20),

PaymentMethod VARCHAR(50),

PaymentDate DATE,

PaymentStatus VARCHAR(50),

Total INT,

OrderID INT NOT NULL

);

ALTER TABLE Payment

ADD CONSTRAINT pk\_payment\_id PRIMARY KEY (PaymentID);

ALTER TABLE Payment

ADD CONSTRAINT FK\_Payment\_Order FOREIGN KEY (OrderID) REFERENCES [Order](OrderID);

CREATE TABLE MenuItem (

ItemID INT NOT NULL,

ItemName VARCHAR(50),

Description TEXT,

Price DECIMAL(10, 2),

TotalCalories INT,

RestaurantId INT NOT NULL

);

ALTER TABLE MenuItem

ADD CONSTRAINT pk\_item\_id PRIMARY KEY (ItemID);

ALTER TABLE MenuItem

ADD CONSTRAINT FK\_MenuItem\_Restaurant FOREIGN KEY (RestaurantId) REFERENCES Restaurant(RestaurantID);

CREATE TABLE MenuItemOrder (

MenuItemOrderId INT NOT NULL,

ItemID INT NOT NULL,

OrderID INT NOT NULL,

Quantity INT

);

ALTER TABLE MenuItemOrder

ADD CONSTRAINT pk\_menu\_item\_order\_id PRIMARY KEY (MenuItemOrderId);

ALTER TABLE MenuItemOrder

ADD CONSTRAINT FK\_MenuItemOrder\_Item FOREIGN KEY (ItemID) REFERENCES MenuItem(ItemID);

ALTER TABLE MenuItemOrder

ADD CONSTRAINT FK\_MenuItemOrder\_Order FOREIGN KEY (OrderID) REFERENCES [Order](OrderID);

CREATE TABLE Employee (

EmployeeID INT NOT NULL,

Name VARCHAR(50),

Position VARCHAR(50),

Salary DECIMAL(10, 2),

HireDate DATE,

DateOfBirth DATE,

ContactInfo VARCHAR(50)

);

ALTER TABLE Employee

ADD CONSTRAINT pk\_employee\_id PRIMARY KEY (EmployeeID);

CREATE TABLE Feedback (

FeedbackID INT NOT NULL,

CustomerID INT NOT NULL,

OrderID INT NOT NULL,

Rating INT,

Comment TEXT,

FeedbackDate DATE

);

ALTER TABLE Feedback

ADD CONSTRAINT pk\_feedback\_id PRIMARY KEY (FeedbackID);

ALTER TABLE Feedback

ADD CONSTRAINT FK\_Feedback\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

ALTER TABLE Feedback

ADD CONSTRAINT FK\_Feedback\_Order FOREIGN KEY (OrderID) REFERENCES [Order](OrderID);

CREATE TABLE Promotion (

PromotionID INT NOT NULL,

Name VARCHAR(50),

Description TEXT,

StartDate DATE,

EndDate DATE,

DiscountPercentage DECIMAL(5, 2)

);

ALTER TABLE Promotion

ADD CONSTRAINT pk\_promotion\_id PRIMARY KEY (PromotionID);

CREATE TABLE CustomerEmployee (

CustomerEmployeeId INT NOT NULL,

CustomerID INT NOT NULL,

EmployeeID INT NOT NULL,

CustomerNote VARCHAR(50),

[Date] DATE

);

ALTER TABLE CustomerEmployee

ADD CONSTRAINT pk\_customer\_employee\_id PRIMARY KEY (CustomerEmployeeId);

ALTER TABLE CustomerEmployee

ADD CONSTRAINT FK\_CustomerEmployee\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

ALTER TABLE CustomerEmployee

ADD CONSTRAINT FK\_CustomerEmployee\_Employee FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID);

CREATE TABLE CustomerPromotion (

CustomerPromotionId INT NOT NULL,

CustomerID INT NOT NULL,

PromotionID INT NOT NULL,

PromotionCondition VARCHAR(50),

PromoEvent VARCHAR(50)

);

ALTER TABLE CustomerPromotion

ADD CONSTRAINT pk\_customer\_promotion\_id PRIMARY KEY (CustomerPromotionId);

ALTER TABLE CustomerPromotion

ADD CONSTRAINT FK\_CustomerPromotion\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

ALTER TABLE CustomerPromotion

ADD CONSTRAINT FK\_CustomerPromotion\_Promotion FOREIGN KEY (PromotionID) REFERENCES Promotion(PromotionID);

INSERTING VALUES ON TABLES:

Restaurant

A screenshot of a computer

Description automatically generated

Customer

A screenshot of a computer

Description automatically generated

Order  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedPayment

MenuItem

A screenshot of a credit card

Description automatically generated

MenuItemOrder

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedEmployee

Feedback

A screenshot of a computer

Description automatically generated

Promotion

A screenshot of a computer

Description automatically generated

CustomerEmployee

A screenshot of a computer

Description automatically generated

CustomerPromotion

A screenshot of a computer

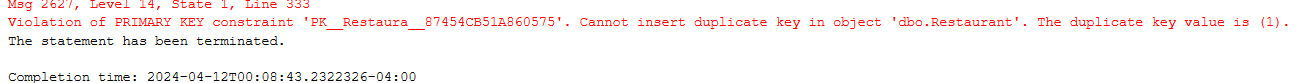
Description automatically generated

**Constraints Check**

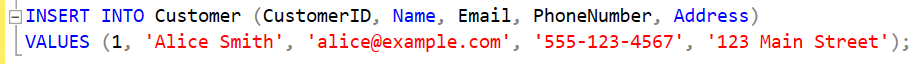
**Primary Key Constraint (pk\_restaurant\_id):**

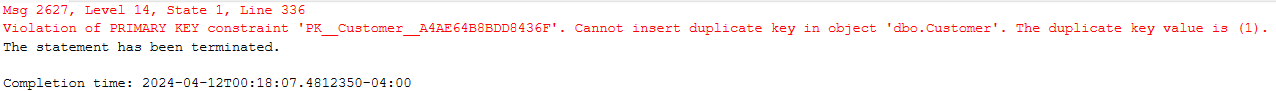
restaurant with an existing RestaurantID:





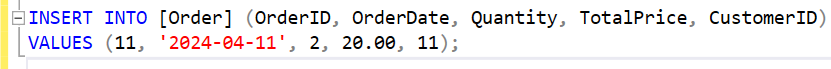
**Primary Key Constraint (pk\_CustomerID):**

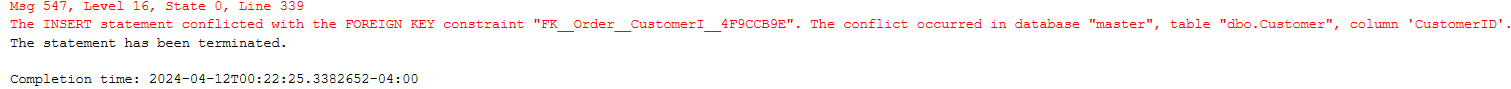
* customer with an existing CustomerId  
  



**Foreign Key Constraint (FK\_Order\_Customer):**

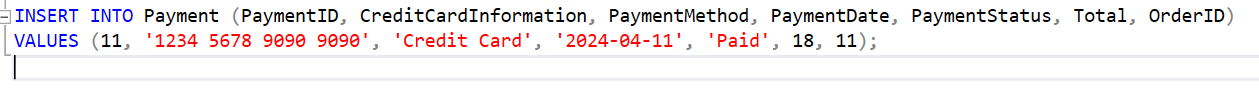
* order with a non-existing CustomerID:

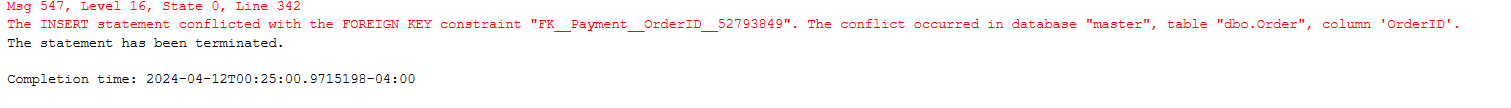




**Foreign Key Constraint (FK\_Payment\_Order):**

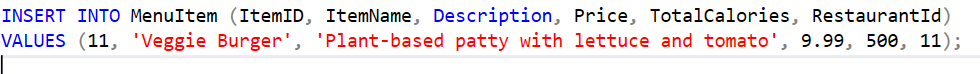
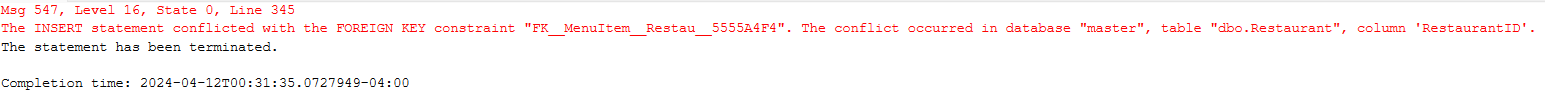
* payment with a non-existing OrderID:



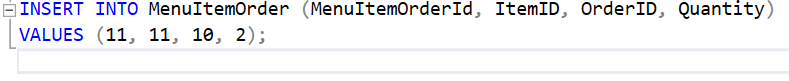


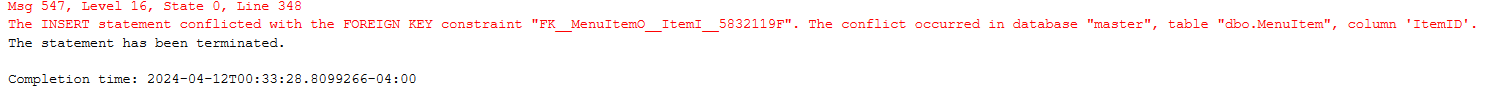
**Foreign Key Constraint (FK\_MenuItem\_Restaurant):**

* menu item with a non-existing RestaurantID:

  
**Foreign Key Constraint (FK\_MenuItemOrder\_Item):**

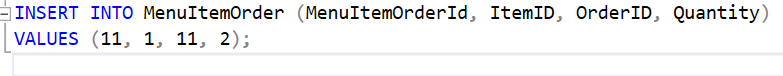
* menu item order with a non-existing ItemID:

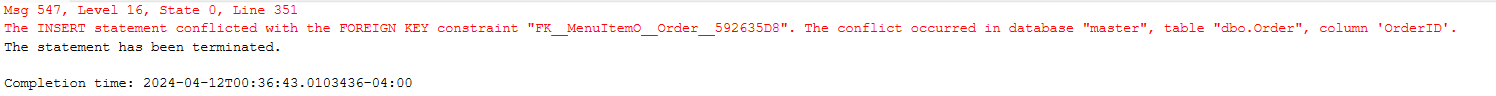




**Foreign Key Constraint (FK\_MenuItemOrder\_Order):**

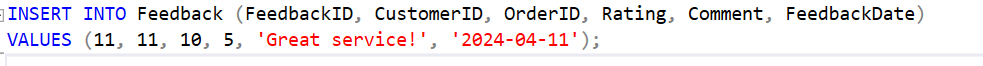
* menu item order with a non-existing OrderID:

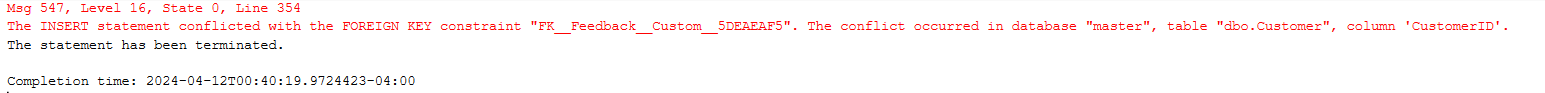




**Foreign Key Constraint (FK\_Feedback\_Customer):**

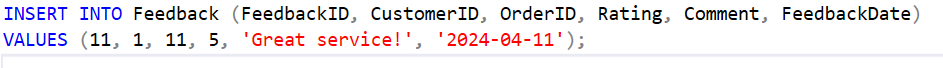
* feedback with a non-existing CustomerID:

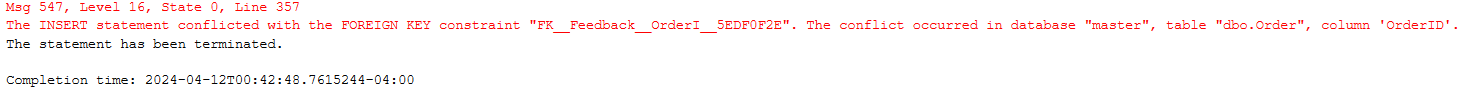




**Foreign Key Constraint (FK\_Feedback\_Order):**

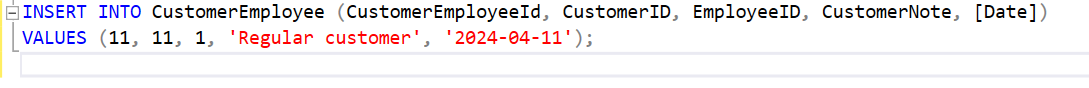
* feedback with a non-existing OrderID:

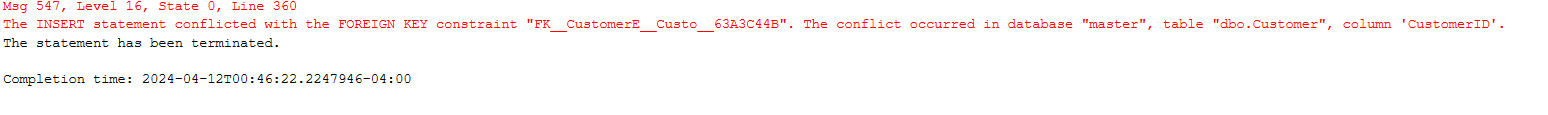




**Foreign Key Constraint (FK\_CustomerEmployee\_Customer):**

* customer-employee relationship with a non-existing CustomerID:





**SQL Queries with different commands**

1. Inner Join with Aggregation and Ordering:

A screenshot of a computer

Description automatically generated

2. Left Join with Subquery and Filtering:

A screenshot of a computer

Description automatically generated

3. Subquery with EXISTS

A screenshot of a computer

Description automatically generated

4. Create a view  
A screenshot of a computer

Description automatically generated

5. Using Subquery in WHERE Clause

A screenshot of a computer

Description automatically generated

6. Aggregate function with Grouping

A screenshot of a menu

Description automatically generated

7. Using Subquery in SELECT Clause

A screenshot of a computer

Description automatically generated

8. Count orders

A screenshot of a computer

Description automatically generated

9. Max calories on the menu

A screenshot of a computer

Description automatically generated

10. Rounding up the price on the menu

A screenshot of a computer

Description automatically generated

**REVIEWED DDL queries for tables and Alter commands**

CREATE TABLE Restaurant (

RestaurantID INT NOT NULL,

Name VARCHAR(50),

Location VARCHAR(50),

PhoneNumber VARCHAR(20) NOT NULL,

OpeningHours TIME,

CONSTRAINT chk\_phone\_number CHECK (PhoneNumber <> '')

);

ALTER TABLE Restaurant

ADD CONSTRAINT pk\_restaurant\_id PRIMARY KEY (RestaurantID);

CREATE TABLE Customer (

CustomerID INT NOT NULL,

Name VARCHAR(50),

Email VARCHAR(50),

PhoneNumber VARCHAR(20),

Address VARCHAR(50)

);

ALTER TABLE Customer

ADD CONSTRAINT pk\_CustomerID PRIMARY KEY (CustomerID);

CREATE TABLE [Order] (

OrderID INT NOT NULL,

OrderDate DATE,

Quantity INT,

TotalPrice DECIMAL(10, 2),

CustomerID INT NOT NULL,

CONSTRAINT chk\_total\_price CHECK (TotalPrice >= 0),

CONSTRAINT chk\_quantity CHECK (Quantity > 0)

);

ALTER TABLE [Order]

ADD CONSTRAINT pk\_order\_id PRIMARY KEY (OrderID);

ALTER TABLE [Order]

ADD CONSTRAINT FK\_Order\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

CREATE TABLE Payment (

PaymentID INT NOT NULL,

CreditCardInformation VARCHAR(20),

PaymentMethod VARCHAR(50),

PaymentDate DATE,

PaymentStatus VARCHAR(50),

Total INT,

OrderID INT NOT NULL,

CONSTRAINT chk\_total CHECK (Total >= 0)

);

ALTER TABLE Payment

ADD CONSTRAINT pk\_payment\_id PRIMARY KEY (PaymentID);

ALTER TABLE Payment

ADD CONSTRAINT FK\_Payment\_Order FOREIGN KEY (OrderID) REFERENCES [Order](OrderID);

CREATE TABLE MenuItem (

ItemID INT NOT NULL,

ItemName VARCHAR(50),

Description TEXT,

Price DECIMAL(10, 2),

TotalCalories INT,

RestaurantID INT NOT NULL,

CONSTRAINT chk\_price CHECK (Price >= 0),

CONSTRAINT chk\_total\_calories CHECK (TotalCalories >= 0)

);

ALTER TABLE MenuItem

ADD CONSTRAINT pk\_item\_id PRIMARY KEY (ItemID);

ALTER TABLE MenuItem

ADD CONSTRAINT FK\_MenuItem\_Restaurant FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID);

CREATE TABLE MenuItemOrder (

MenuItemOrderId INT NOT NULL,

ItemID INT NOT NULL,

OrderID INT NOT NULL,

Quantity INT,

CONSTRAINT chk\_quantity CHECK (Quantity > 0)

);

ALTER TABLE MenuItemOrder

ADD CONSTRAINT pk\_menu\_item\_order\_id PRIMARY KEY (MenuItemOrderId);

ALTER TABLE MenuItemOrder

ADD CONSTRAINT FK\_MenuItemOrder\_Item FOREIGN KEY (ItemID) REFERENCES MenuItem(ItemID);

ALTER TABLE MenuItemOrder

ADD CONSTRAINT FK\_MenuItemOrder\_Order FOREIGN KEY (OrderID) REFERENCES [Order](OrderID);

CREATE TABLE Employee (

EmployeeID INT NOT NULL,

Name VARCHAR(50),

Position VARCHAR(50),

Salary DECIMAL(10, 2),

HireDate DATE,

DateOfBirth DATE,

ContactInfo VARCHAR(50),

CONSTRAINT chk\_salary CHECK (Salary >= 0)

);

ALTER TABLE Employee

ADD CONSTRAINT pk\_employee\_id PRIMARY KEY (EmployeeID);

CREATE TABLE Feedback (

FeedbackID INT NOT NULL,

CustomerID INT NOT NULL,

OrderID INT NOT NULL,

Rating INT,

Comment TEXT,

FeedbackDate DATE,

CONSTRAINT chk\_rating CHECK (Rating BETWEEN 1 AND 5)

);

ALTER TABLE Feedback

ADD CONSTRAINT pk\_feedback\_id PRIMARY KEY (FeedbackID);

ALTER TABLE Feedback

ADD CONSTRAINT FK\_Feedback\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

ALTER TABLE Feedback

ADD CONSTRAINT FK\_Feedback\_Order FOREIGN KEY (OrderID) REFERENCES [Order](OrderID);

CREATE TABLE Promotion (

PromotionID INT NOT NULL,

Name VARCHAR(50),

Description TEXT,

StartDate DATE,

EndDate DATE,

DiscountPercentage DECIMAL(5, 2),

CONSTRAINT chk\_discount\_percentage CHECK (DiscountPercentage BETWEEN 0 AND 100),

CONSTRAINT chk\_dates CHECK (EndDate > StartDate)

);

ALTER TABLE Promotion

ADD CONSTRAINT pk\_promotion\_id PRIMARY KEY (PromotionID);

CREATE TABLE CustomerEmployee (

CustomerEmployeeID INT NOT NULL,

CustomerID INT NOT NULL,

EmployeeID INT NOT NULL,

CustomerNote VARCHAR(50),

[Date] DATE

);

ALTER TABLE CustomerEmployee

ADD CONSTRAINT pk\_customer\_employee\_id PRIMARY KEY (CustomerEmployeeID);

ALTER TABLE CustomerEmployee

ADD CONSTRAINT FK\_CustomerEmployee\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

ALTER TABLE CustomerEmployee

ADD CONSTRAINT FK\_CustomerEmployee\_Employee FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID);

CREATE TABLE CustomerPromotion (

CustomerPromotionID INT NOT NULL,

CustomerID INT NOT NULL,

PromotionID INT NOT NULL,

PromotionCondition VARCHAR(50),

PromoEvent VARCHAR(50)

);

ALTER TABLE CustomerPromotion

ADD CONSTRAINT pk\_customer\_promotion\_id PRIMARY KEY (CustomerPromotionID);

ALTER TABLE CustomerPromotion

ADD CONSTRAINT FK\_CustomerPromotion\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

ALTER TABLE CustomerPromotion

ADD CONSTRAINT FK\_CustomerPromotion\_Promotion FOREIGN KEY (PromotionID) REFERENCES Promotion(PromotionID);