|  |
| --- |
| Foodie Express |
| Team 3  Angela Ozelski  Vu Ton  BRIAN PALMER  ist 659  fALL 2023  sECTION m401 |

# Summary / goal

Introducing FoodieExpress, the go-to solution for a delightful dining experience. Powered by a robust SQL database, our food delivery app seamlessly connects customers to a diverse array of restaurants, ensuring an extensive menu at your fingertips. The goal of the design of this application was to connect users with their favorite restaurants by offering a simple interface and delivery options.

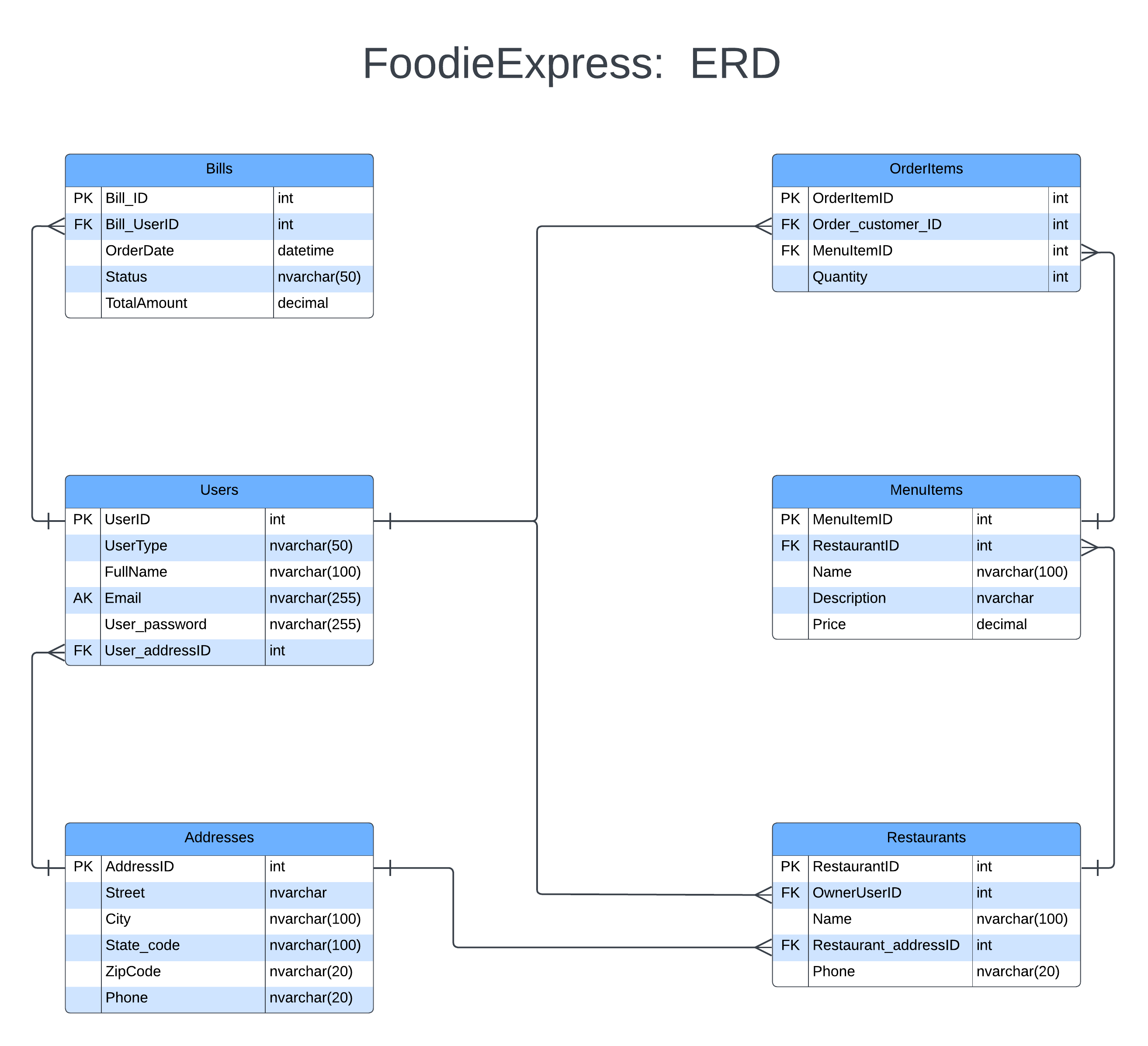
Our user-friendly interface allows customers to effortlessly explore a myriad of culinary options, placing orders with just a few taps. For restaurant partners, FoodieExpress offers increased visibility and customer engagement, fostering a mutually beneficial relationship for both restaurants and hungry customers.

# business Rules

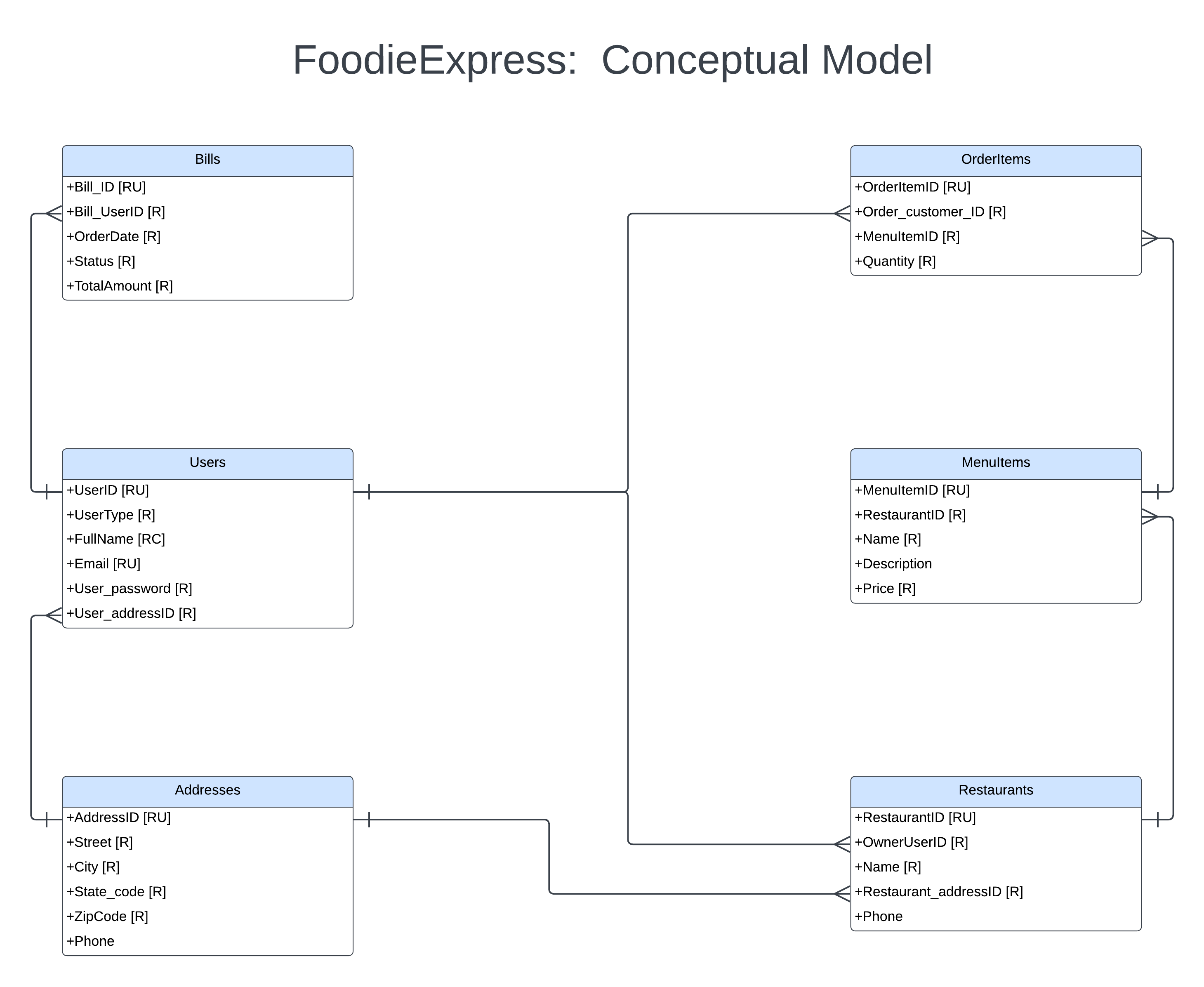
What are the business rules we identified?

* **Users:**
* Users must provide valid and unique information, including a full name, userID, email, and password.
* User profiles should include relevant details such as contact information and preferences.
* UserID is a Primary Key
* Email is an Alternative Key
* User\_addressID is a Foreign Key
* **Order Items:**
* Each order must be associated with a specific user, ensuring traceability.
* Order items should contain details like quantity, order item ID, and menu item ID.
* OrderItemID is a Primary Key
* MenuItemID is a Foreign Key
* **Addresses:**
* Restaurants should have valid and unique addresses, including city, street, state, and postal code.
* Delivery addresses for users must be complete and accurate to facilitate successful delivery.
* AddressID is a Primary Key
* **Menu Items:**
* Each menu item should be linked to a specific restaurant.
* Menu items must include relevant details like name, description, and price.
* Availability status should be tracked to reflect whether an item is currently offered.
* MenuItemID is a Primary Key
* RestaurantID is the Foreign Key
* **Restaurants:**
* Restaurants must have unique identifiers which includes RestaurantID, OwnerUserID, and Restaurant\_addressID.
* Relevant details such as name of the restaurant and phone number is included in this table
* Operational status (open or closed) should be maintained and updated as needed.
* RestaurantID is a Primary Key
* OwneryUserID and Restaurant\_addressID are Foreign Keys
* **Bills:**
  + Bills contains the order date, status of the order, total amount of the order, BillID which is unique for each order, and Bill\_UserID which is the individual who placed the order.
  + Bill\_ID is a Primary Key
  + Bill\_UserID is a Foreign Key

# concept Models



# concept Models



# Entities/attributes

|  |  |  |
| --- | --- | --- |
| **ENTITIES** | **ATTRIBUTES** | **TYPE** |
| Addresses | AddressID | int (auto increment) |
| Street | nvarchar(max) |
| City | nvarchar(100) |
| State\_code | nvarchar(100) |
| ZipCode | nvarchar(20) |
| Phone | nvarchar(20) |
| Bills | Bill\_ID | int (auto increment) |
| Bill\_UserID | int |
| OrderDate | datetime |
| Status | nvarchar(50) |
| TotalAmount | decimal(10,2) |
| MenuItems | MenuItemID | int (auto increment) |
| RestaurantID | int |
| Name | nvarchar(100) |
| Description | nvarchar(max) |
| Price | decimal(10,2) |
| OrderItems | OrderItemID | int (auto increment) |
| Order\_customer\_ID | int |
| MenuItemID | int |
| Quantity | int |
| Restaurants | RestaurantID | int (auto increment) |
| OwnerUserID | int |
| Name | nvarchar(100) |
| Restaurant\_addressID | int |
| Phone | nvarchar(20) |
| Users | UserID | int (auto increment) |
| UserType | nvarchar(50) |
| FullName | nvarchar(100) |
| Email | nvarchar(255) |
| User\_password | nvarchar(255) |
| User\_addressID | int |

# Relationships (fully ATTRIBUTED)

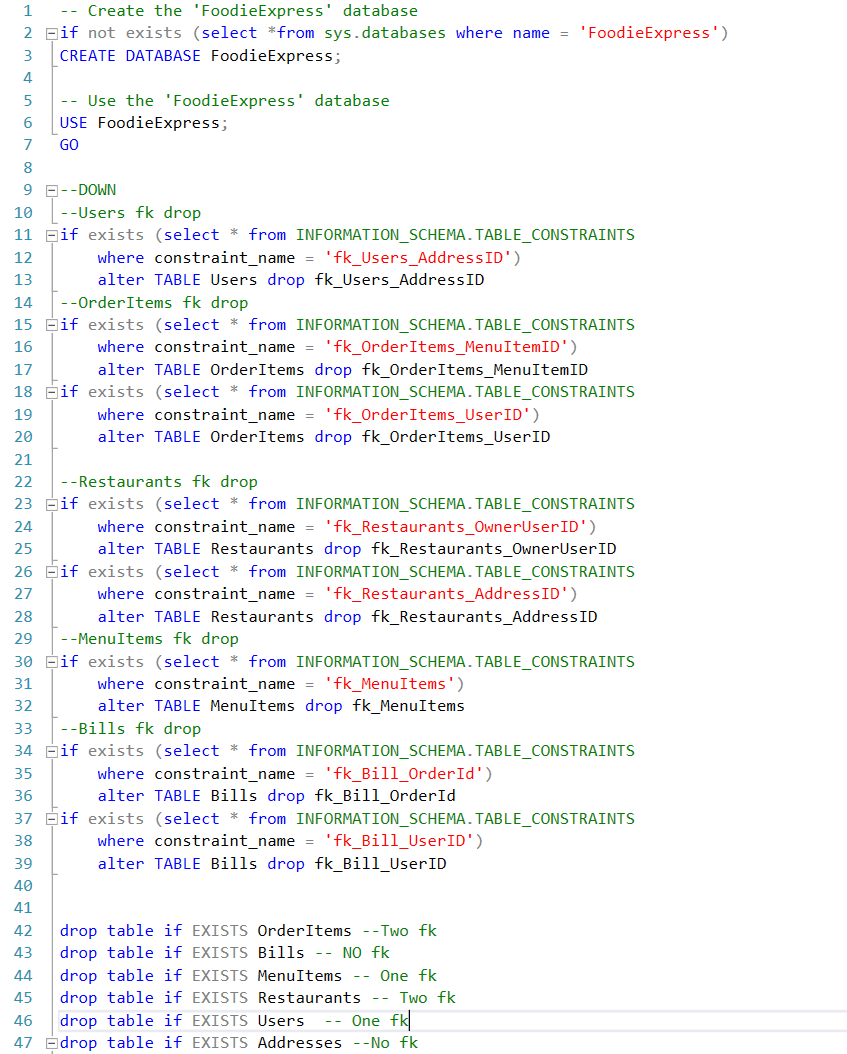
|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTES** | **TYPE** | **CONSTRAINTS** |
| Addresses | AddressID | int | PRIMARY KEY, IDENTITY |
| Street | nvarchar(max) | NOT NULL |
| City | nvarchar(100) | NOT NULL |
| State\_code | nvarchar(100) | NOT NULL |
| ZipCode | nvarchar(20) | NOT NULL |
| Phone | nvarchar(20) |  |
| Bills | Bill\_ID | int | PRIMARY KEY, IDENTITY |
| Bill\_UserID | int | FOREIGN KEY REFERENCES Users(UserID) |
| OrderDate | datetime | DEFAULT GETDATE() |
| Status | nvarchar(50) | NOT NULL |
| TotalAmount | decimal(10,2) | NOT NULL |
| MenuItems | MenuItemID | int | PRIMARY KEY, IDENTITY |
| RestaurantID | int | FOREIGN KEY REFERENCES Restaurants(RestaurantID) |
| Name | nvarchar(100) | NOT NULL |
| Description | nvarchar(max) |  |
| Price | decimal(10,2) | NOT NULL |
| OrderItems | OrderItemID | int | PRIMARY KEY, IDENTITY |
| Order\_customer\_ID | int | FOREIGN KEY REFERENCES Users(UserID) |
| MenuItemID | int | FOREIGN KEY REFERENCES MenuItems(MenuItemID) |
| Quantity | int | NOT NULL |
| Restaurants | RestaurantID | int | PRIMARY KEY, IDENTITY |
| OwnerUserID | int | FOREIGN KEY REFERENCES Users(UserID) |
| Name | nvarchar(100) | NOT NULL |
| Restaurant\_addressID | int | FOREIGN KEY REFERENCES Addresses(AddressID) |
| Phone | nvarchar(20) |  |
| Users | UserID | int | PRIMARY KEY, IDENTITY |
| UserType | nvarchar(50) | NOT NULL |
| FullName | nvarchar(100) | NOT NULL |
| Email | nvarchar(255) | UNIQUE, NOT NULL |
| User\_password | nvarchar(255) | NOT NULL |
| User\_addressID | int | FOREIGN KEY REFERENCES Addresses(AddressID) |

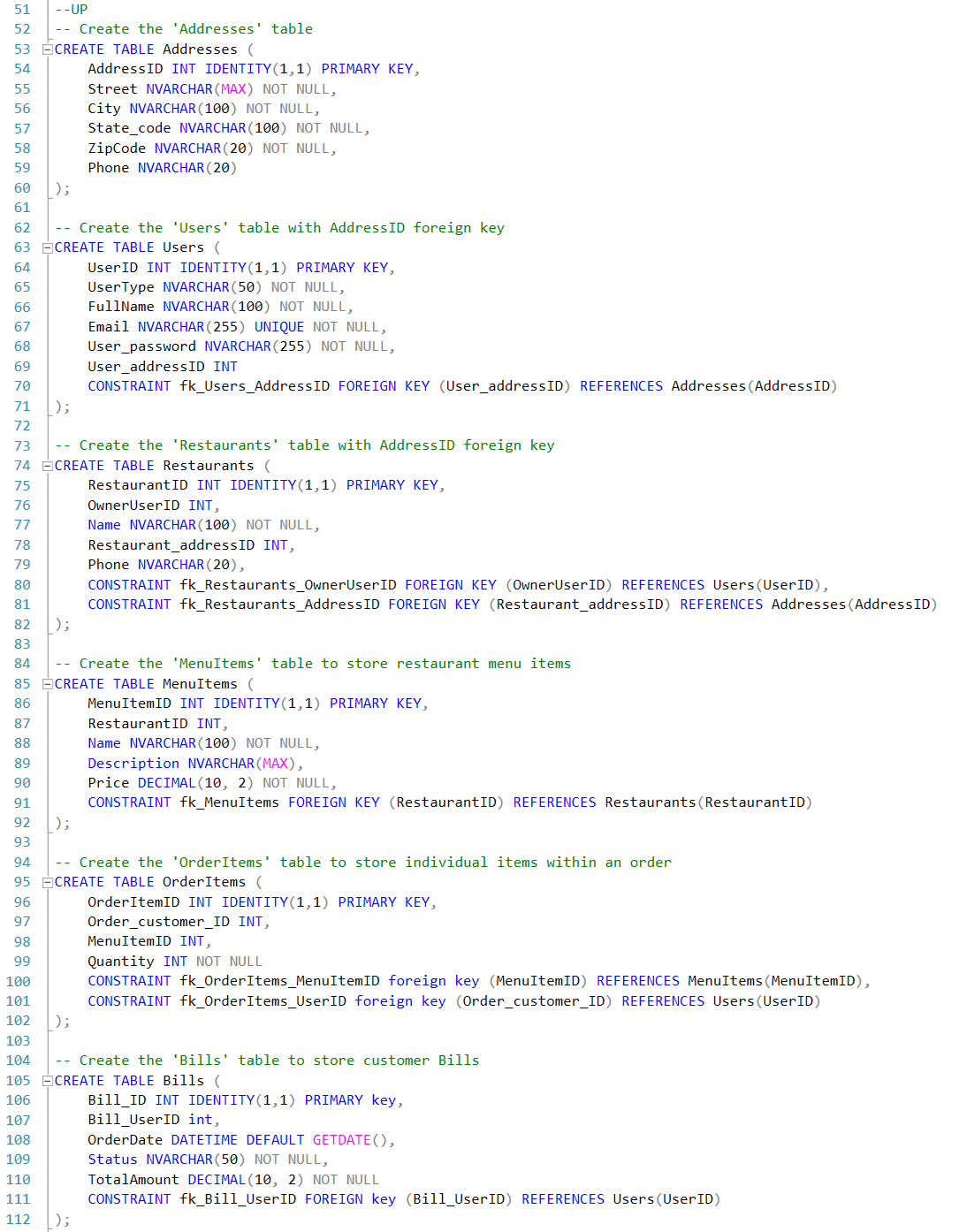
# Mockups/Dashboards

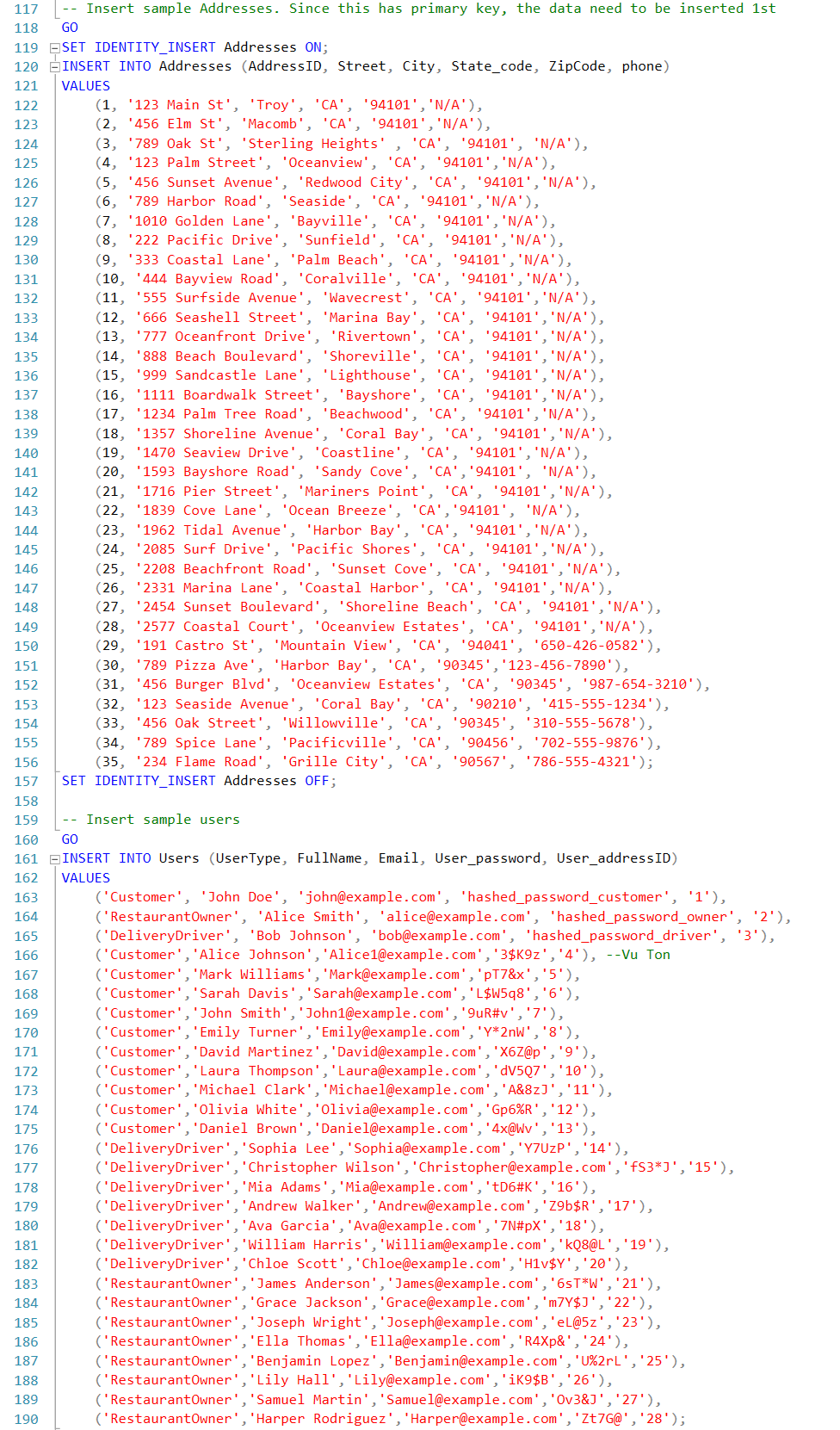
|  |  |
| --- | --- |
|  |  |
|  |  |

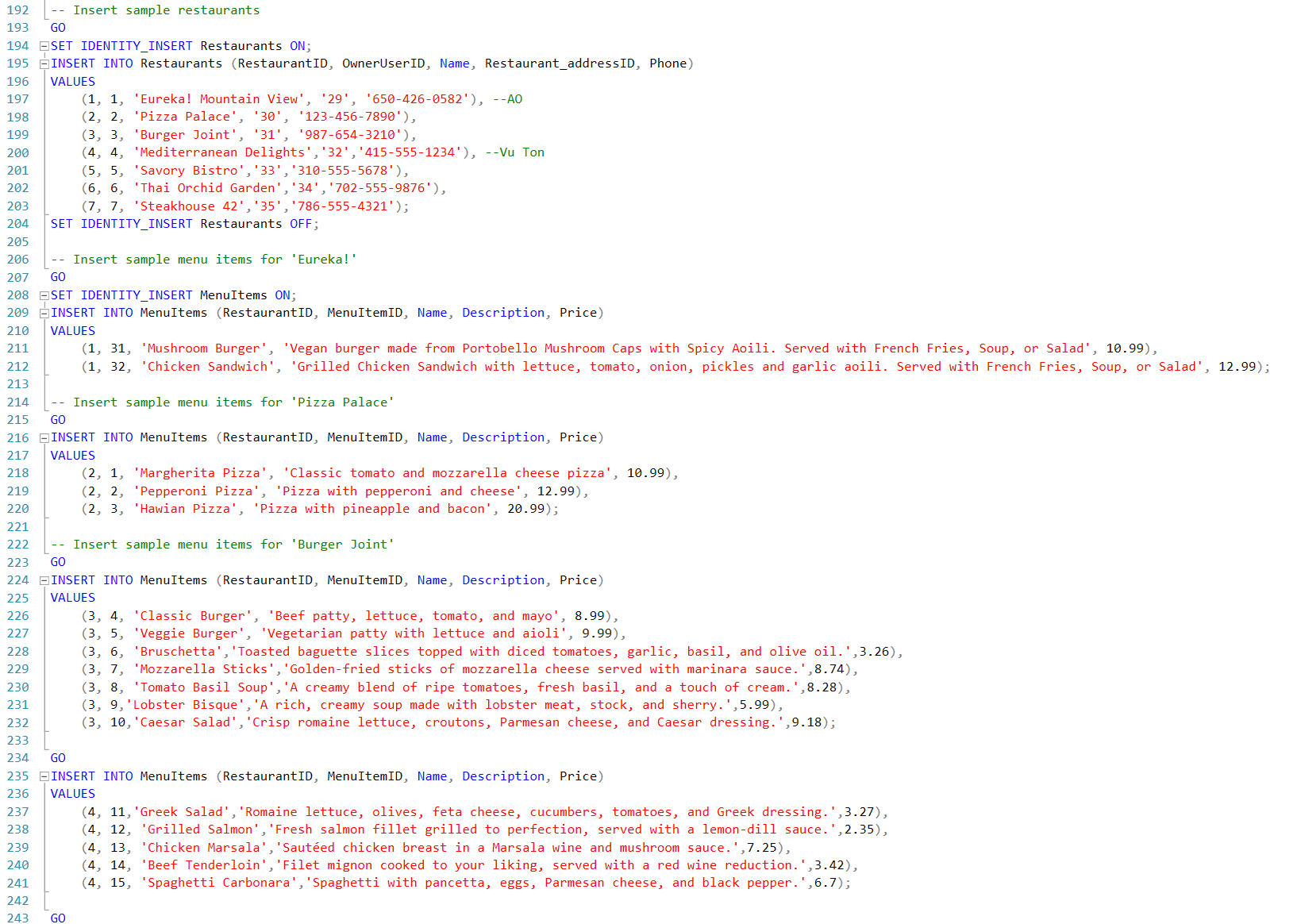
|  |  |
| --- | --- |
|  |  |
|  | |

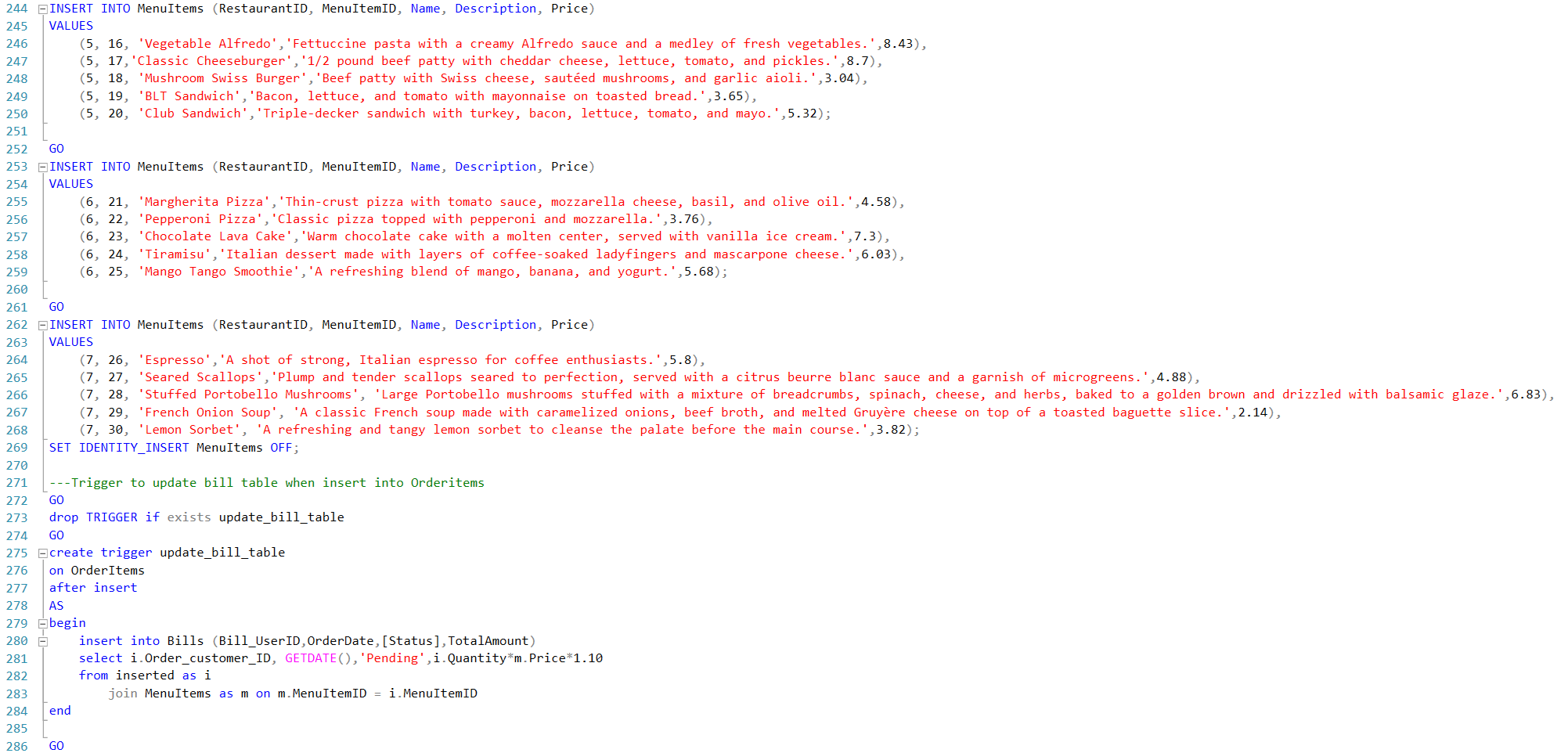
# SQL Code

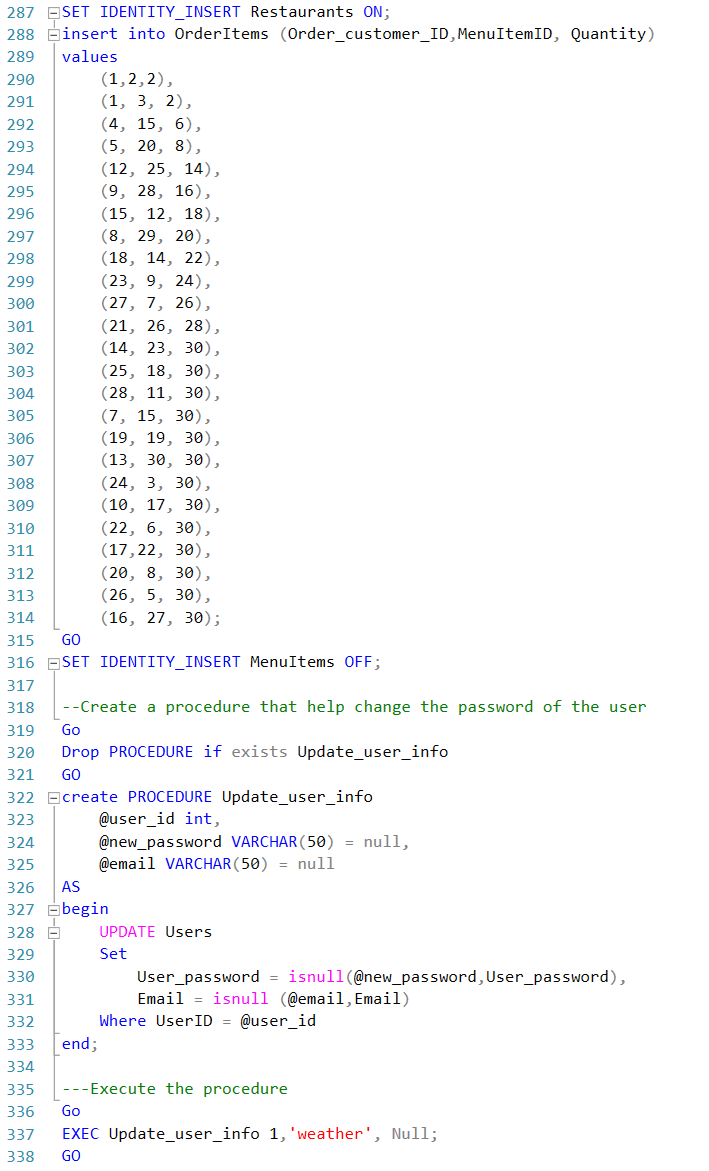


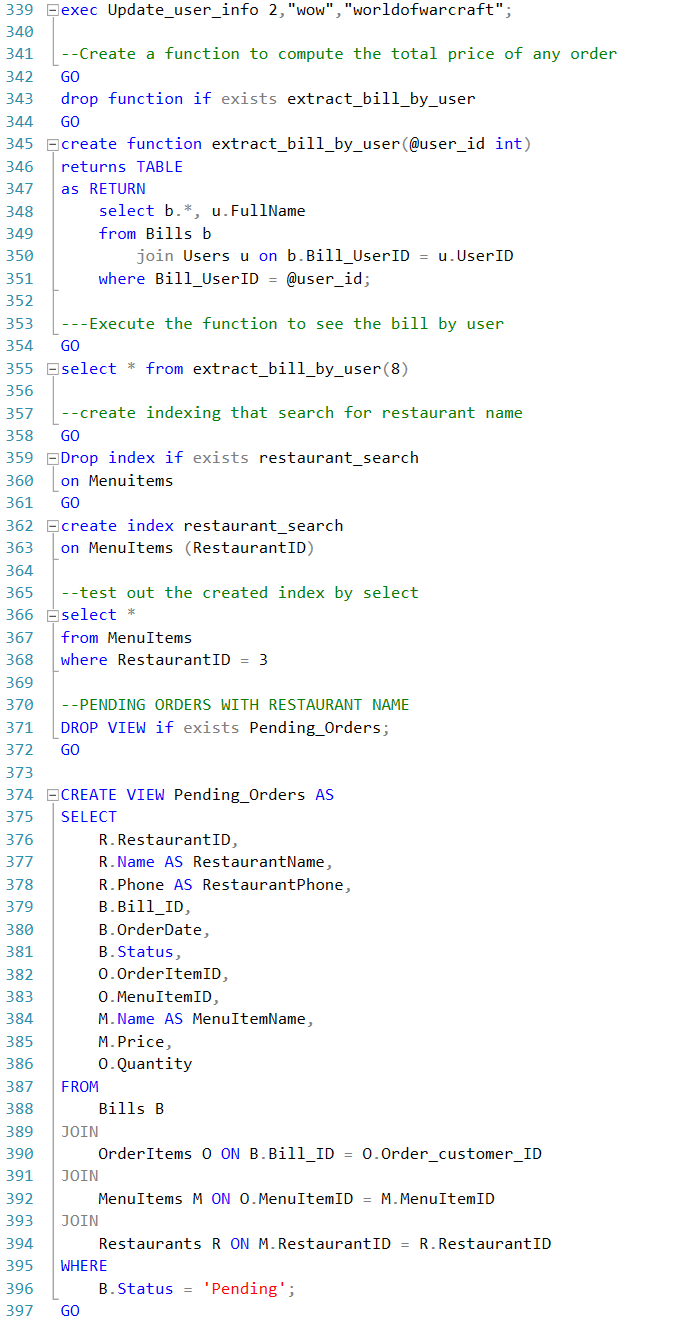


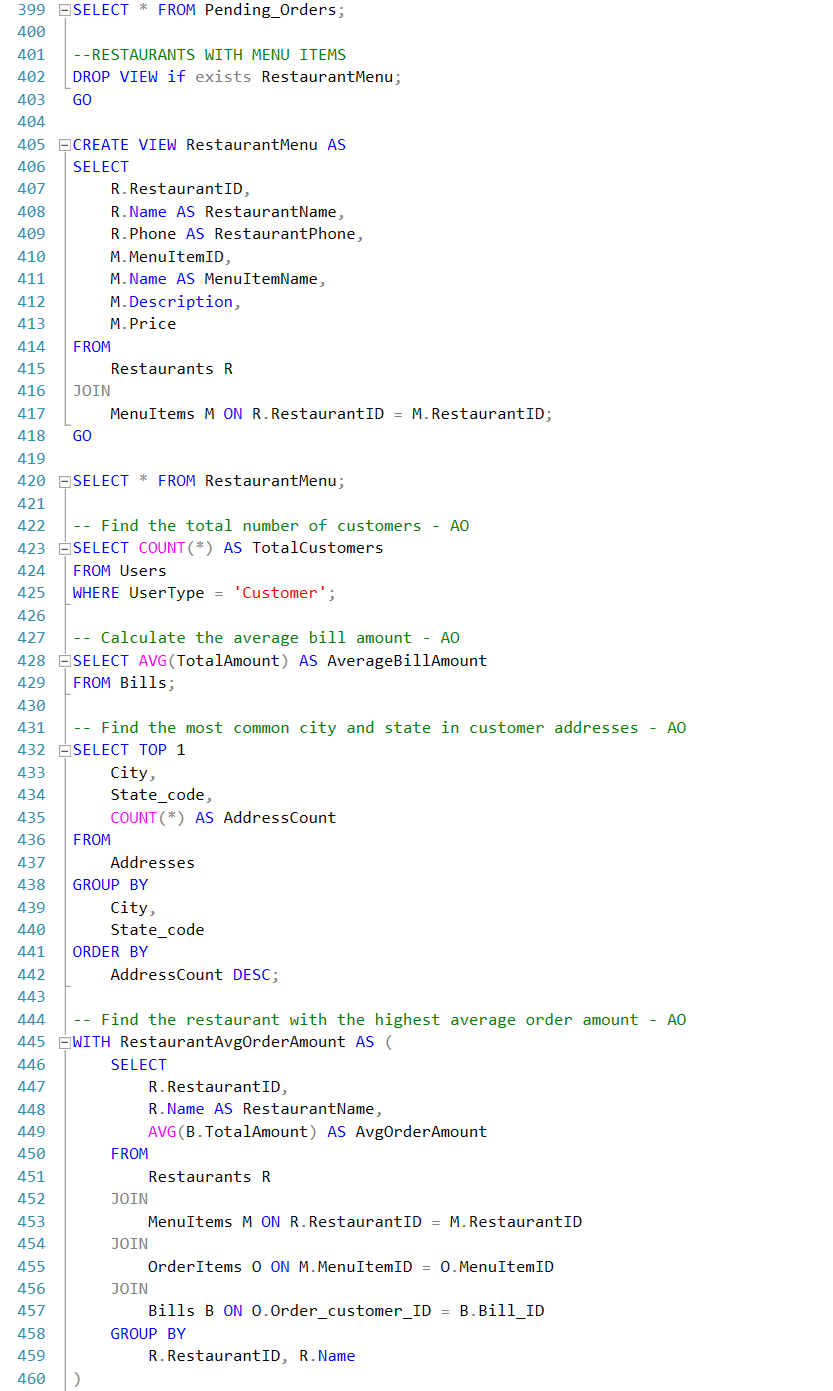


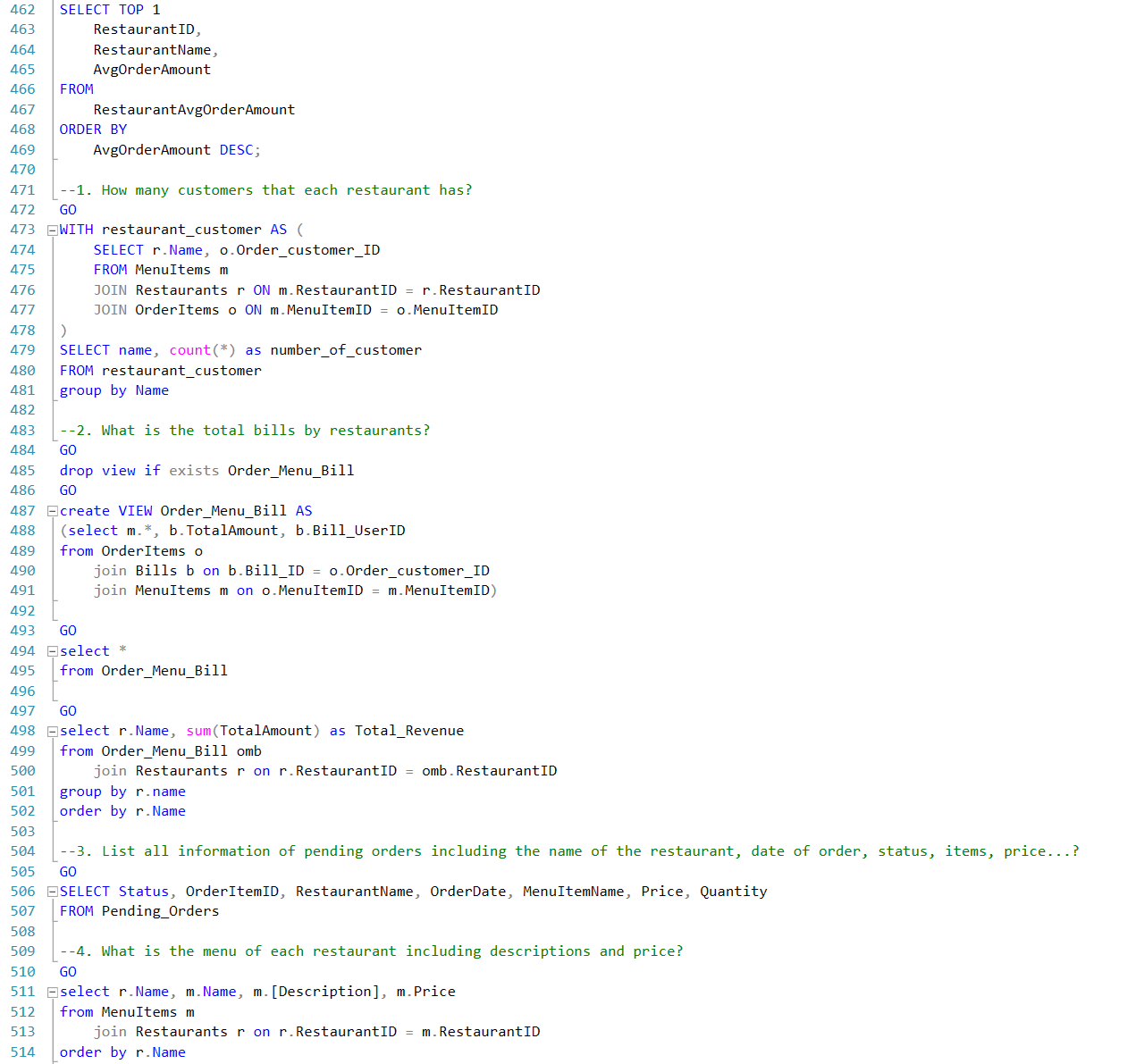












# Data Questions and answers

1. How many customers that each restaurant has?

|  |  |
| --- | --- |
| GO  WITH restaurant\_customer AS (  SELECT r.Name, o.Order\_customer\_ID  FROM MenuItems m  JOIN Restaurants r ON m.RestaurantID = r.RestaurantID  JOIN OrderItems o ON m.MenuItemID = o.MenuItemID )  SELECT name, count(\*) as number\_of\_customer  FROM restaurant\_customer  group by Name |  |

2. What is the total bills by restaurants?

|  |  |
| --- | --- |
| GO  drop view if exists Order\_Menu\_Bill  GO  create VIEW Order\_Menu\_Bill AS  (select m.\*, b.TotalAmount, b.Bill\_UserID  from OrderItems o  join Bills b on b.Bill\_ID = o.Order\_customer\_ID  join MenuItems m on o.MenuItemID = m.MenuItemID)  GO  select r.Name, sum(TotalAmount) as Total\_Revenue  from Order\_Menu\_Bill omb  join Restaurants r on r.RestaurantID = omb.RestaurantID  group by r.name  order by r.Name |  |

3. List all information of pending orders including the name of the restaurant, date of order, status, items, price...?

|  |  |
| --- | --- |
| DROP VIEW if exists RestaurantMenu;  GO  CREATE VIEW RestaurantMenu AS  SELECT  R.RestaurantID,  R.Name AS RestaurantName,  R.Phone AS RestaurantPhone,  M.MenuItemID,  M.Name AS MenuItemName,  M.Description,  M.Price  FROM  Restaurants R  JOIN  MenuItems M ON R.RestaurantID = M.RestaurantID;  GO  SELECT Status, OrderItemID, RestaurantName, OrderDate, MenuItemName, Price, Quantity  FROM Pending\_Orders |  |

4. What is the menu of each restaurant including descriptions and price?

|  |  |
| --- | --- |
| GO  select r.Name, m.Name, m.[Description], m.Price  from MenuItems m  join Restaurants r on r.RestaurantID = m.RestaurantID  order by r.Name |  |

5. Find the total number of customers

|  |  |
| --- | --- |
|  |  |

6. Calculate the average bill amount

|  |  |
| --- | --- |
|  |  |

7. Find the most common city and state in customer addresses

|  |  |
| --- | --- |
|  |  |

8. Find the restaurant with the highest average order amount

|  |  |
| --- | --- |
|  |  |

# summary/conclusion

The FoodieExpress database is a comprehensive solution designed to manage the operations of a food delivery service. This relational database system encompasses various entities, relationships, and functionalities crucial to the seamless functioning of a food delivery ecosystem.

**Key Components:**

* **Entities:**
  + **Addresses:** Stores information about customer addresses.
  + **Users:** Manages user details with a focus on different user types (Customers, Restaurant Owners).
  + **Restaurants:** Captures restaurant details including the name and contact information as well as a unique Restaurant ID.
  + **MenuItems:** Represents the menu items offered by restaurants which each menu item having a unique ID.
  + **OrderItems:** Tracks individual items within an order and their quantities.
  + **Bills:** Stores information about customer bills, including order status and total amount.
* **Relationships:**
  + Establishes relationships between entities such as Users and Addresses, Restaurants and MenuItems, OrderItems and Bills, etc.
* **Functionality Highlights:**
  + **Sample Data:** Provides sample data for Users, Restaurants, MenuItems, and more.
  + **Triggers:** Implements a trigger to automatically update the Bills table when new OrderItems are inserted.
  + **Stored Procedures:** Includes a stored procedure for updating user information, allowing for password changes and email updates.
  + **Functions:** Implements a function to extract bills by user, facilitating easy access to individual customer billing information.
  + **Indexing:** Demonstrates the use of indexing for optimized search queries, exemplified by a search for restaurant names.
  + **Views:** Creates views for pending orders and restaurant menus, offering insights into operational aspects.
* **Conclusion:** The FoodieExpress database is a foundational structure for managing various aspects of a food delivery service. Its extensibility allows for future enhancements and scalability, making it a robust solution for businesses in the food industry.

# Reflection

**Initial Impressions**

As a team embarking on the FoodieExpress project, our objective was to create an advanced and user-friendly food ordering database for an online delivery service. This project presented us with an opportunity to blend our diverse skills in a challenging yet exciting venture.

**Our Learning Curve**

Throughout the project, we deepened our understanding of complex SQL queries, transaction management, and database optimization techniques. Designing for handling real-time data and integrating different aspects of the food ordering process provided us with practical insights into database management.

Below were some challenges that we faced and overcame:

* Importing data to the system: we needed to generate a sizeable amount of data into our database. To overcome this, we split the work among the team and utilized the help from ChatGPT to generate some of the data.
* Designing the database system: we started off the project by creating tables that would be in our database before creating a logical model and an entity relational model (ERD). This caused some issues with inconsistency among tables and foreign key establishment. We overcame this by thoroughly analyzing our logical model and ERD; we then modified the ‘Bills’ table to better sync data with other tables using a ‘trigger’.
* Coding on SQL: we had some tough time debugging our codes multiple times. To overcome this, we worked together and reviewed the lines of code every time we met.. Each time we saved the code in different versions via OneDrive so that we could always track back to prior versions.

**Future Direction**

The completed FoodieExpress database project was a journey of technical growth, problem-solving, and collaborative learning. It underscored the importance of a team's adaptability and continuous learning in database management techniques. It was an experience that went beyond mere database building; it involved understanding and addressing real-world challenges in the online food delivery space. We look forward to carrying these learnings forward into our future projects.

Below are future enhancements that would have been added to our application:

* User Authentication: Implement a robust user authentication system for secure user access.
* Advanced Analytics: Integrate analytics for business intelligence, aiding in decision-making processes.
* Real-time Order Tracking: Enhance customer experience with real-time order tracking features.

# team log

Friday, 10/20 - discussed project ideas and chose FoodieExpress

Friday, 10/27 - implemented tables and inserted data

Friday, 11/10 - finalized tables and data

Friday, 11/17 - added common table expressions

Friday, 12/01 - created stored procedure, trigger, views added

Friday, 12/07 - Indexing added, ERD completed, Wireframing completed, Finalized SQL coding

Friday, 12/14 - finalized Exec Summary and PPT

Monday, 12/18 - reviewed SQL coding, Exec Summary, and PPT and collectively turned in project