due date: 10.01.2023



**Beykoz University**

**Department of “Computer Engineering”**

**“Database Systems - 60612MEEOZ-CME0075”**

* **Restaurant Management System -**

**- Project Phase II -**

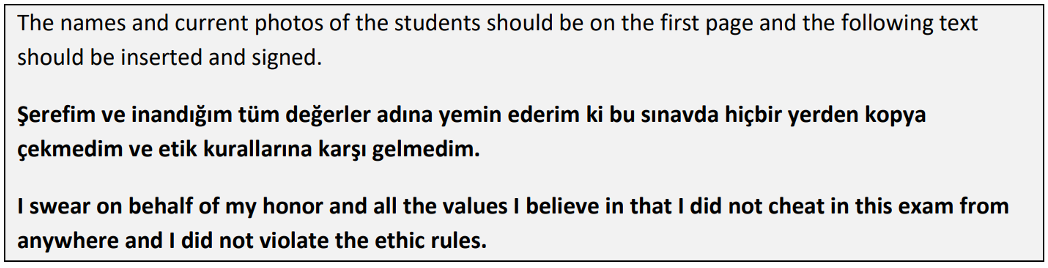
**Lecturer: Dr. Selçuk KIRAN**

***Project Owner:***

***Leyla Abdullayeva -***

***1904010038***

sign:



**TABLE OF CONTENTS**

[**PHASE I**](#_a5q79ycwto94) **2**

[**1.Brief Description of the System**](#_z3e3ik4k52i6) 2

[1.1 Scope Definition](#_dg5ss7vvta0n) 3

[1.2. Determining System Requirements](#_j1we9jg6kynl) 3

[1.3 Potential Users of the System](#_hn6unkvxyx4c) 4

[**2.Design of Database System -**](#_uoec2pilsc5z) 5

[2.1 ERD Schema](#_cixg79enia3p) 5

[2.2 Database ER Diagram from MS SQL - Management Studio](#_rbti41kle3ty) 6

[2.2 Database ER Diagram from MS SQL - Version 2](#_72wg9cji6lmb) 7

[2.3 Database Dictionary](#_fx527m4o3ghh) 8

# **PHASE I**

# **1.Brief Description of the System**

A Restaurant Management Database System is a computerized system that is designed to help manage the various aspects of a restaurant, including menu management, employee scheduling, and inventory tracking. The system is typically implemented using a database management system such as Microsoft SQL Server.

This study includes a new database application that will help the restaurant manager to manage the restaurant more effectively and efficiently by computerizing meal ordering, billing and inventory control. This is the main purpose of this Project.

The database infrastructure of the program is MS SQL.

The goals of our system are:

1. Improved efficiency in managing and updating the restaurant menu.
2. Streamlined employee scheduling and task assignments.
3. Enhanced tracking and management of inventory.
4. Improved financial reporting and analysis.
5. Enhanced customer relationship management.
6. Increased ability to measure and analyze key performance indicators.
7. Improved communication and collaboration among staff.
8. Enhanced security and data protection.
9. Increased overall profitability of the restaurant.

## **1.1 Scope Definition**

The scope of the Restaurant Management Database System project is to develop a comprehensive computerized system that automates various aspects of restaurant operations, including billing, employee management, and record-keeping. The system will be implemented using a database management system such as Microsoft SQL Server, and will be designed to be used by employees at all levels of the organization. The main goals of the project are to improve efficiency, streamline processes, and increase profitability by providing real-time data analysis and decision-making tools. The system will be designed to be user-friendly, with different levels of access granted to employees based on their roles and responsibilities. This will ensure that the privacy and security of sensitive records are protected. In addition to benefiting employees, the system is also expected to provide a better experience for customers by enabling faster, more accurate service. The database will be the central repository for all restaurant-related data, and will be used to track and manage inventory, employee schedules, and other key aspects of restaurant management. The database will be maintained throughout the project to ensure the accuracy and integrity of the data it contains.

### **1.2. Determining System Requirements**

1. Users (both restaurant managers, waiters, chefs) can login to the system and will be able to do different operations.
2. Restaurant manager (owner) can modify (change) its own data.
3. The system can verify the data before a transaction.
4. Restaurant manager should be able to update information about his/her restaurant.
5. Restaurant manager should be able to view weekly sales for his/her (own) outlet.
6. All the users including (restaurant manager, chef, waiter, customer) can check menu / food data by clicking on a certain menu's main page in a web application (or just page).
7. The system should have a login feature for different users, including restaurant managers, waiters, and chefs, to access and perform various actions.
8. Restaurant managers should have the ability to edit their own information.
9. The system should have the ability to verify data before transactions.
10. Restaurant managers should be able to update information about their restaurant.
11. Restaurant managers should be able to view weekly sales for their specific outlet.
12. Users, including restaurant managers, chefs, waiters, and customers, should be able to view menu and food information through a web application or page.
13. The system should be able to handle large amounts of data efficiently.
14. The system should have secure access controls to protect sensitive information.
15. The system should have backup and recovery capabilities in case of data loss.
16. The system should be scalable to allow for potential expansion in the future.

#### **1.3 Potential Users of the System**

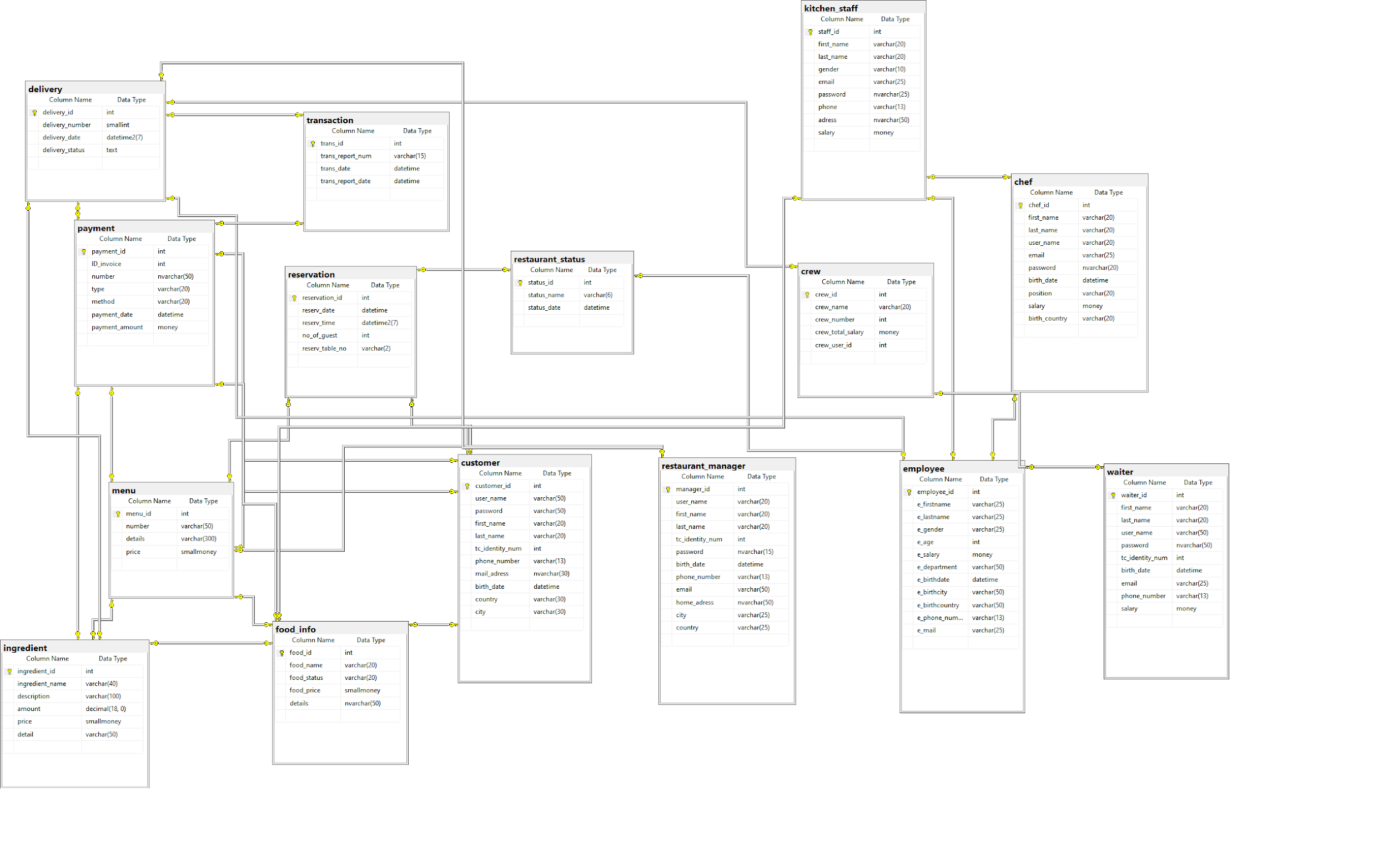
After the database design, user screen designs with different authorizations to use the system were made. Database tables were created to meet the needs of three different types of users.

* ***Restaurant Manager*** – Restaurant manager will receive statistical reports and will be able to follow business operations. These users will have the highest level of access to the system and will be responsible for managing and updating information about their restaurant, such as menu items and pricing. They will also be able to view sales data and make decisions based on this information.
* ***Customer*** – While customers may not have direct access to the system, they will still be considered users as they will be able to view menu and pricing information through a web application or page. This will allow them to make informed decisions about their orders. Customers can order the food by using this software from computers & also mobile smartphones.
* ***Employees (Waiter,Chef, Staff)*** - These users may include any other staff members at the restaurant, such as dishwashers or bussers, who may not have the same level of access as the other user groups. Their access and responsibilities will depend on their specific roles within the restaurant.
* ***Waiters -*** These users will be responsible for inputting orders into the system and updating the status of orders as they are prepared and served to customers. They will also be able to access menu and pricing information as needed.
* ***Chefs -*** These users will be able to view orders as they are inputted into the system and mark them as completed when they are finished preparing the food. They will also have access to menu and ingredient information to assist with food preparation.

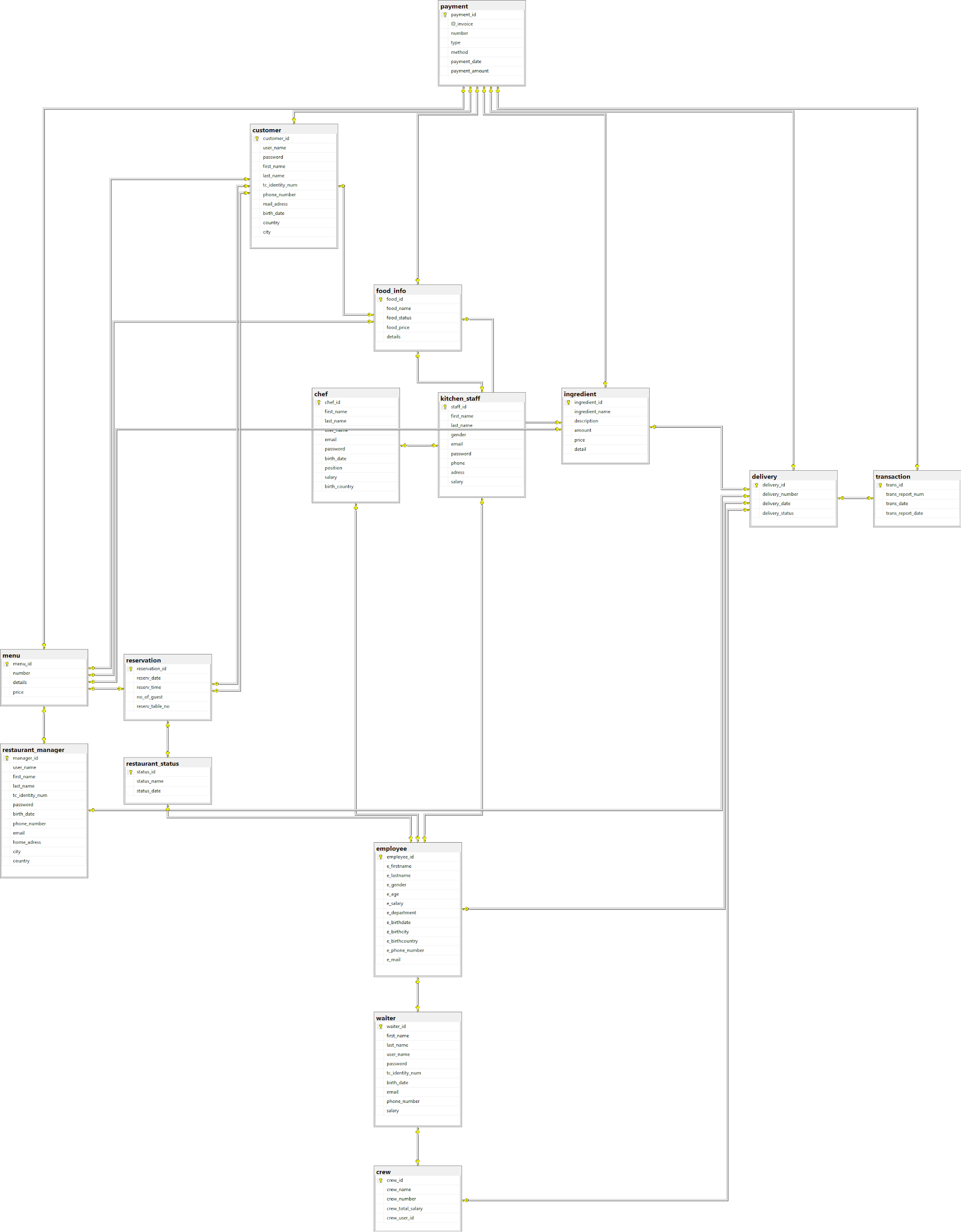
# **2.Design of Database System -**

###### **2.1 ERD Schema**

###### **2.2 Database ER Diagram from MS SQL - Management Studio**



###### **2.2 Database ER Diagram from MS SQL - Version 2**



###### **2.3 Database Dictionary - Part I**



**Part II**



###### **General view - Data Dictionary**

# **PHASE II**

# **MS SQL - DDL (CREATE, DROP, ALTER TABLE, INDEX)**

## 1. CREATE TABLE Clauses for Each Table (Create Tables.sql file)

| **/\*\*\*\*\*\* Object: Table [dbo].[chef] Script Date: 7.01.2023 23:39:37 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[chef](**  **[chef\_id] [int] NOT NULL,**  **[first\_name] [varchar](20) NOT NULL,**  **[last\_name] [varchar](20) NOT NULL,**  **[user\_name] [varchar](20) NULL,**  **[email] [varchar](25) NOT NULL,**  **[password] [nvarchar](20) NULL,**  **[birth\_date] [datetime] NOT NULL,**  **[position] [varchar](20) NULL,**  **[salary] [money] NULL,**  **[birth\_country] [varchar](20) NULL,**  **CONSTRAINT [PK\_chef] PRIMARY KEY CLUSTERED**  **(**  **[chef\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO**  **—----**  **/\*\*\*\*\*\* Object: Table [dbo].[crew] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[crew](**  **[crew\_id] [int] NOT NULL,**  **[crew\_name] [varchar](20) NULL,**  **[crew\_number] [int] NOT NULL,**  **[crew\_total\_salary] [money] NULL,**  **[crew\_user\_id] [int] NOT NULL,**  **CONSTRAINT [PK\_crew] PRIMARY KEY CLUSTERED**  **(**  **[crew\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO**  **/\*\*\*\*\*\* Object: Table [dbo].[delivery] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[delivery](**  **[delivery\_id] [int] NOT NULL,**  **[delivery\_number] [smallint] NULL,**  **[delivery\_date] [datetime2](7) NOT NULL,**  **[delivery\_status] [text] NULL,**  **CONSTRAINT [PK\_delivery] PRIMARY KEY CLUSTERED**  **(**  **[delivery\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]**  **GO**  **—---**  **/\*\*\*\*\*\* Object: Table [dbo].[employee] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[employee](**  **[employee\_id] [int] NOT NULL,**  **[e\_firstname] [varchar](25) NOT NULL,**  **[e\_lastname] [varchar](25) NOT NULL,**  **[e\_gender] [varchar](25) NULL,**  **[e\_age] [int] NULL,**  **[e\_salary] [money] NOT NULL,**  **[e\_department] [varchar](50) NOT NULL,**  **[e\_birthdate] [datetime] NULL,**  **[e\_birthcity] [varchar](50) NULL,**  **[e\_birthcountry] [varchar](50) NULL,**  **[e\_phone\_number] [varchar](13) NULL,**  **[e\_mail] [varchar](25) NULL,**  **CONSTRAINT [PK\_employee] PRIMARY KEY CLUSTERED**  **(**  **[employee\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| --- |

| **/\*\*\*\*\*\* Object: Table [dbo].[food\_info] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[food\_info](**  **[food\_id] [int] NOT NULL,**  **[food\_name] [varchar](20) NULL,**  **[food\_status] [varchar](20) NULL,**  **[food\_price] [smallmoney] NULL,**  **[details] [nvarchar](50) NOT NULL,**  **CONSTRAINT [PK\_food\_info] PRIMARY KEY CLUSTERED**  **(**  **[food\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| --- |
| **/\*\*\*\*\*\* Object: Table [dbo].[ingredient] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[ingredient](**  **[ingredient\_id] [int] NOT NULL,**  **[ingredient\_name] [varchar](40) NULL,**  **[description] [varchar](100) NULL,**  **[amount] [decimal](18, 0) NOT NULL,**  **[price] [smallmoney] NOT NULL,**  **[detail] [varchar](50) NULL,**  **CONSTRAINT [PK\_ingredient] PRIMARY KEY CLUSTERED**  **(**  **[ingredient\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |

| **/\*\*\*\*\*\* Object: Table [dbo].[kitchen\_staff] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[kitchen\_staff](**  **[staff\_id] [int] NOT NULL,**  **[first\_name] [varchar](20) NOT NULL,**  **[last\_name] [varchar](20) NOT NULL,**  **[gender] [varchar](10) NULL,**  **[email] [varchar](25) NULL,**  **[password] [nvarchar](25) NULL,**  **[phone] [varchar](13) NULL,**  **[adress] [nvarchar](50) NULL,**  **[salary] [money] NOT NULL,**  **CONSTRAINT [PK\_kitchen\_staff] PRIMARY KEY CLUSTERED**  **(**  **[staff\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| --- |
| **/\*\*\*\*\*\* Object: Table [dbo].[menu] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[menu](**  **[menu\_id] [int] NOT NULL,**  **[number] [varchar](50) NULL,**  **[details] [varchar](300) NULL,**  **[price] [smallmoney] NOT NULL,**  **CONSTRAINT [PK\_menu] PRIMARY KEY CLUSTERED**  **(**  **[menu\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| **/\*\*\*\*\*\* Object: Table [dbo].[payment] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[payment](**  **[payment\_id] [int] NOT NULL,**  **[ID\_invoice] [int] NULL,**  **[number] [nvarchar](50) NULL,**  **[type] [varchar](20) NULL,**  **[method] [varchar](20) NULL,**  **[payment\_date] [datetime] NOT NULL,**  **[payment\_amount] [money] NOT NULL,**  **CONSTRAINT [PK\_payment] PRIMARY KEY CLUSTERED**  **(**  **[payment\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| **/\*\*\*\*\*\* Object: Table [dbo].[reservation] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[reservation](**  **[reservation\_id] [int] NOT NULL,**  **[reserv\_date] [datetime] NULL,**  **[reserv\_time] [datetime2](7) NOT NULL,**  **[no\_of\_guest] [int] NOT NULL,**  **[reserv\_table\_no] [varchar](2) NULL,**  **CONSTRAINT [PK\_reservation] PRIMARY KEY CLUSTERED**  **(**  **[reservation\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO**  **/\*\*\*\*\*\* Object: Table [dbo].[restaurant\_manager] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[restaurant\_manager](**  **[manager\_id] [int] NOT NULL,**  **[user\_name] [varchar](20) NOT NULL,**  **[first\_name] [varchar](20) NOT NULL,**  **[last\_name] [varchar](20) NOT NULL,**  **[tc\_identity\_num] [int] NULL,**  **[password] [nvarchar](15) NOT NULL,**  **[birth\_date] [datetime] NULL,**  **[phone\_number] [varchar](13) NULL,**  **[email] [varchar](50) NOT NULL,**  **[home\_adress] [nvarchar](50) NULL,**  **[city] [varchar](25) NULL,**  **[country] [varchar](25) NULL,**  **CONSTRAINT [PK\_restaurant\_manager] PRIMARY KEY CLUSTERED**  **(**  **[manager\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| **/\*\*\*\*\*\* Object: Table [dbo].[restaurant\_status] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[restaurant\_status](**  **[status\_id] [int] NOT NULL,**  **[status\_name] [varchar](6) NOT NULL,**  **[status\_date] [datetime] NULL,**  **CONSTRAINT [PK\_restaurant\_status] PRIMARY KEY CLUSTERED**  **(**  **[status\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| **/\*\*\*\*\*\* Object: Table [dbo].[transaction] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[transaction](**  **[trans\_id] [int] NOT NULL,**  **[trans\_report\_num] [varchar](15) NULL,**  **[trans\_date] [datetime] NULL,**  **[trans\_report\_date] [datetime] NOT NULL,**  **CONSTRAINT [PK\_transaction] PRIMARY KEY CLUSTERED**  **(**  **[trans\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| **/\*\*\*\*\*\* Object: Table [dbo].[waiter] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[waiter](**  **[waiter\_id] [int] NOT NULL,**  **[first\_name] [varchar](20) NULL,**  **[last\_name] [varchar](20) NULL,**  **[user\_name] [varchar](50) NOT NULL,**  **[password] [nvarchar](50) NOT NULL,**  **[tc\_identity\_num] [int] NULL,**  **[birth\_date] [datetime] NULL,**  **[email] [varchar](25) NOT NULL,**  **[phone\_number] [varchar](13) NULL,**  **[salary] [money] NULL,**  **CONSTRAINT [PK\_waiter] PRIMARY KEY CLUSTERED**  **(**  **[waiter\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |
| **/\*\*\*\*\*\* Object: Table [dbo].[customer] Script Date: 7.01.2023 23:39:38 \*\*\*\*\*\*/**  **SET ANSI\_NULLS ON**  **GO**  **SET QUOTED\_IDENTIFIER ON**  **GO**  **CREATE TABLE [dbo].[customer](**  **[customer\_id] [int] NOT NULL,**  **[user\_name] [varchar](50) NULL,**  **[password] [varchar](50) NOT NULL,**  **[first\_name] [varchar](20) NOT NULL,**  **[last\_name] [varchar](20) NOT NULL,**  **[tc\_identity\_num] [int] NULL,**  **[phone\_number] [varchar](13) NULL,**  **[mail\_adress] [nvarchar](30) NOT NULL,**  **[birth\_date] [datetime] NULL,**  **[country] [varchar](30) NULL,**  **[city] [varchar](30) NULL,**  **CONSTRAINT [PK\_customer] PRIMARY KEY CLUSTERED**  **(**  **[customer\_id] ASC**  **)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]**  **) ON [PRIMARY]**  **GO** |

## 2. ALTER TABLE Clauses (Alter\_Table\_Statements.sql)

| * 1. Add a new column called gender to customer table:   **-- Before adding a new column the chef table,** |
| --- |
| **ALTER TABLE dbo.chef**  **ADD gender bit**  **-- After adding a new column the chef table,** |

| * 2. Drop “city” column from customer table:   **-- Before dropping the column,** |
| --- |
| **-- After dropping the “city” column,** |

| * 3. Set food\_info’s details from nvarchar(300) to nvarchar(250)   **-- food\_info table before,** |
| --- |
| **ALTER TABLE dbo.food\_info**  **ALTER COLUMN details nvarchar(250)**  **-- food\_info table after,** |

| * 4. Make “mail\_adress” unique in the customer table.   **ALTER TABLE customer**  **ADD UNIQUE (mail\_adress)** |
| --- |
| * 5. Add a check constraint to the “food\_price” to not to be below 0.1.   **ALTER TABLE dbo.food\_info**  **ADD CHECK (food\_price>0.1)** |

## 3. 2 DROP TABLE Clauses (Drop\_Statements.sql)

| * 1. Delete the birth\_date column from the “customer” table.   **ALTER TABLE customer DROP COLUMN birth\_date** |
| --- |
| * 2. Delete the “gender” column from the “chef” table.   **ALTER TABLE dbo.chef DROP COLUMN gender** |

## 4. 5 INDICES (Index\_Statements.sql)

| * 1. Create an index named ‘ix\_cname’ on the ‘first\_name’ and ‘last\_name’ columns in the customer table.   **CREATE INDEX ix\_cname**  **ON customer (first\_name, last\_name)** |
| --- |
| * 2. Create an index named ‘ix\_cname’ on the ‘user\_name’ column in the “chef” table.   **CREATE UNIQUE INDEX ix\_cname**  **ON chef (user\_name)** |
| * 3. Create an index named ‘ix\_fname’ on the ‘food\_name’ column on the “food\_info” table.   **CREATE UNIQUE INDEX ix\_fname**  **ON food\_info (food\_name)** |
| * 4. Create an index named ‘ix\_crname’ on the “crew\_number” column on the “crew” table.   **CREATE UNIQUE INDEX ix\_crname**  **ON crew (crew\_number)** |
| * 5. Create an index named ‘ix\_iname’ on the “ingredient\_name” column on the “ingredient” table.   **CREATE UNIQUE INDEX ix\_iname**  **ON ingredient (ingredient\_name)** |

# **DML (INSERT, UPDATE, DELETE COMMANDS)**

## 5 INSERT commands for tables (Insert\_Initial\_Values.sql file)

| **INSERT INTO dbo.chef**  **(chef\_id, first\_name, last\_name, user\_name, email, password, birth\_date, position, salary, birth\_country)**  **VALUES**  **('445566778', 'Alice', 'Smith', 'asmith1', 'asmith1@gmail.com', 'password4', '04-30-1995', 'commis', '2500', 'Canada')**    **INSERT INTO dbo.chef**  **(chef\_id, first\_name, last\_name, user\_name, email, password, birth\_date, position, salary, birth\_country)**  **VALUES**  **('556677889', 'James', 'Bond', 'jbond1', 'jbond1@gmail.com', 'password5', '05-19-2000', 'chef', '4000', 'UK')**    **INSERT INTO dbo.chef**  **(chef\_id, first\_name, last\_name, user\_name, email, password, birth\_date, position, salary, birth\_country)**  **VALUES**  **('667788901', 'Emma', 'Watson', 'ewatson1', 'ewatson1@gmail.com', 'password6', '06-26-2005', 'sous chef', '3500', 'UK')**    **INSERT INTO dbo.chef**  **(chef\_id, first\_name, last\_name, user\_name, email, password, birth\_date, position, salary, birth\_country)**  **VALUES**  **('334455667', 'Bob', 'Smith', 'bsmith1', 'bsmith1@gmail.com', 'password3', '03-22-1990', 'chef de partie', '3000', 'Canada')**    **INSERT INTO dbo.chef**  **(chef\_id, first\_name, last\_name, user\_name, email, password, birth\_date, position, salary, birth\_country)**  **VALUES**  **('22336677', 'Emin', 'Mammadov', 'eminchik085', 'eminmmadov0202@gmail.com', 'eminchik085', '02-02-2002', 'lead chef', '10000', 'Azerbaijan')**  **INSERT INTO dbo.chef**  **(chef\_id, first\_name, last\_name, user\_name, email, password, birth\_date, position, salary, birth\_country)**  **—------------------------------------------**  **INSERT INTO dbo.customer (customer\_id, user\_name, password, first\_name, last\_name, tc\_identity\_num, phone\_number, mail\_adress, birth\_date, country, city)**  **VALUES**  **(1, 'cust1', 'pass1', 'Nazrin', 'Huseynli', '997896543', '123-456-7890', 'hnazrin26@gmail.com', '02-26-2000', 'Azerbaijan', 'Baku')**  **INSERT INTO dbo.customer (customer\_id, user\_name, password, first\_name, last\_name, tc\_identity\_num, phone\_number, mail\_adress, birth\_date, country, city)**  **VALUES**  **(2, 'cust2', 'pass2', 'Alex', 'Parker', '123456789', '234-567-8901', 'aparker1@gmail.com', '03-17-1995', 'USA', 'New York')**    **INSERT INTO dbo.customer (customer\_id, user\_name, password, first\_name, last\_name, tc\_identity\_num, phone\_number, mail\_adress, birth\_date, country, city)**  **VALUES**  **(3, 'cust3', 'pass3', 'Samantha', 'James', '234567890', '345-678-9012', 'sjames1@gmail.com', '04-08-1990', 'Canada', 'Toronto')**  **INSERT INTO dbo.customer (customer\_id, user\_name, password, first\_name, last\_name, tc\_identity\_num, phone\_number, mail\_adress, birth\_date, country, city)**  **VALUES**  **(4, 'cust4', 'pass4', 'Michael', 'Brown', '345678901', '456-789-0123', 'mbrown1@gmail.com', '05-30-1985', 'UK', 'London')**  **INSERT INTO dbo.customer (customer\_id, user\_name, password, first\_name, last\_name, tc\_identity\_num, phone\_number, mail\_adress, birth\_date, country, city)**  **VALUES**  **(5, 'cust5', 'pass5', 'Emily', 'Smith', '456789012', '567-890-1234', 'esmith1@gmail.com', '06-20-1980', 'Australia', 'Sydney')**  **INSERT INTO dbo.customer (customer\_id, user\_name, password, first\_name, last\_name, tc\_identity\_num, phone\_number, mail\_adress, birth\_date, country, city).** | **INSERT INTO RESTAURANT\_MANAGEMENT\_SYSTEM.dbo.employee (employee\_id, e\_firstname, e\_lastname, e\_gender, e\_age, e\_salary, e\_department, e\_birthdate, e\_birthcity, e\_birthcountry, e\_phone\_number, e\_mail)**  **VALUES**  **(1, 'Nazrin', 'Huseynli', 'M', '23', '3500', 'Front of House', '02-26-2000', 'Baku', 'Azerbaijan', '123-456-7890', 'hnazrin26@gmail.com')**    **INSERT INTO RESTAURANT\_MANAGEMENT\_SYSTEM.dbo.employee (employee\_id, e\_firstname, e\_lastname, e\_gender, e\_age, e\_salary, e\_department, e\_birthdate, e\_birthcity, e\_birthcountry, e\_phone\_number, e\_mail)**  **VALUES**  **(2, 'Alex', 'Parker', 'M', '25', '3000', 'Back of House', '03-17-1995', 'New York', 'USA', '234-567-8901', 'aparker1@gmail.com')**    **INSERT INTO RESTAURANT\_MANAGEMENT\_SYSTEM.dbo.employee (employee\_id, e\_firstname, e\_lastname, e\_gender, e\_age, e\_salary, e\_department, e\_birthdate, e\_birthcity, e\_birthcountry, e\_phone\_number, e\_mail)**  **VALUES**  **(3, 'Samantha', 'James', 'F', '30', '2500', 'Bakery', '04-08-1990', 'Toronto', 'Canada', '345-678-9012', 'sjames1@gmail.com')**  **INSERT INTO RESTAURANT\_MANAGEMENT\_SYSTEM.dbo.employee (employee\_id, e\_firstname, e\_lastname, e\_gender, e\_age, e\_salary, e\_department, e\_birthdate, e\_birthcity, e\_birthcountry, e\_phone\_number, e\_mail)**  **VALUES**  **(4, 'Michael', 'Brown', 'M', '35', '2000', 'Bar', '05-30-1985', 'London', 'UK', '456-789-0123', 'mbrown1@gmail.com')**  **INSERT INTO RESTAURANT\_MANAGEMENT\_SYSTEM.dbo.employee (employee\_id, e\_firstname, e\_lastname, e\_gender, e\_age, e\_salary, e\_department, e\_birthdate, e\_birthcity, e\_birthcountry, e\_phone\_number, e\_mail)**  **VALUES**  **(5, 'Emily', 'Smith', 'F', '40', '3500', 'Management', '06-20-1980', 'Sydney', 'Australia', '567-890-1234', 'esmith1@gmail.com')**  **INSERT INTO RESTAURANT\_MANAGEMENT\_SYSTEM.dbo.employee (employee\_id, e\_firstname, e\_lastname, e\_gender, e\_age, e\_salary, e\_department, e\_birthdate, e\_birthcity, e\_birthcountry, e\_phone\_number, e\_mail)**  **—------------------------------------------**  **INSERT INTO dbo.food\_info (food\_id, food\_name, food\_status, food\_price, details)**  **VALUES**  **(1, 'Pizza Margherita', 'available', 10, 'A classic pizza with tomato sauce, mozzarella cheese, and fresh basil')**  **INSERT INTO dbo.food\_info (food\_id, food\_name, food\_status, food\_price, details)**  **VALUES**  **(2, 'Spaghetti Bolognese', 'available', 12, 'Spaghetti noodles with a rich meat sauce made with ground beef, onions, and tomatoes')**  **INSERT INTO dbo.food\_info (food\_id, food\_name, food\_status, food\_price, details)**  **VALUES**  **(3, 'Lasagna', 'available', 15, 'Layers of pasta, ground beef, and cheese, baked in a rich tomato sauce')**  **INSERT INTO dbo.food\_info (food\_id, food\_name, food\_status, food\_price, details)**  **VALUES**  **(4, 'Fish & Chips', 'available', 14, 'Battered and fried cod served with fries')**  **INSERT INTO dbo.food\_info (food\_id, food\_name, food\_status, food\_price, details)**  **VALUES**  **(5, 'Roast Beef', 'available', 18, 'Tender roast beef served with roast potatoes and vegetables')**  **INSERT INTO dbo.food\_info (food\_id, food\_name, food\_status, food\_price, details).** |
| --- | --- |

## 5 UPDATE commands for tables (Update Statements.sql)

1. Restaurant Should edit an order’s status as ‘On the way’, ‘preparing’, ‘ready’.

| **-- A restaurant wants to change an order whose order id is 1**  **-- The order type of the order(status\_name) before change,**  **​​SELECT status\_name, status\_date FROM restaurant\_status**  **INNER JOIN food\_info**  **ON status\_id = food\_id**  **WHERE status\_id = 1**    **-- Update order's status to “preparing”.**  **UPDATE restaurant\_status**  **SET status\_name = ('preparing')**  **WHERE status\_name = ('available')**  **-- Printed again** |
| --- |

2. Restaurants can change the prices of food items in “menu”.

| **- - First, let’s print the current price of the number 1 menu.**    **SELECT menu.menu\_id, menu.number, menu.price**  **FROM menu**  **INNER JOIN order\_menu**  **ON menu.menu\_id = order\_menu.menu\_id**  **WHERE menu.menu\_id = 2;**    **--Update price of menu 1 to 20.00.**  **UPDATE menu**  **SET price = 20.00**  **WHERE menu\_id = 2**  **-- Print again** |
| --- |

1. Restaurant suddenly wants to increase all the foods’ prices on the menu by 20%.

| **-- Food's prices before update**    **-- Increase all the menu's prices by 20%**  **UPDATE food**  **SET price = price \* 1.1**  **WHERE restaurant\_id = 3**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 1**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 2**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 3**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 4**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 5**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 6**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 7**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 8**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 9**  **UPDATE menu**  **SET price = price \* 2.2**  **WHERE menu\_id = 10**  **-- Food's prices after the update** |
| --- |

1. If a customer’s phone number doesn’t start with ‘+90’, add ‘+90’ to customer’s phone number.

| **-- Find whose phone number doesn't start with +90**  **SELECT id, first\_name, last\_name, phone\_number FROM customer**  **WHERE phone\_number NOT LIKE '+90%'**    **-- Update phone numbers which don't start with +90**  **UPDATE customer**  **SET phone\_number = '+90' + phone\_number**  **WHERE phone\_number NOT LIKE '+90%'**    **-- After the update,** |
| --- |

1. If an employee wanted to update the “birth\_country” for their web application account. (data info)

| **-Before the update of the employee “e\_birthcity” column in the table “employee”.**    **- - After the update of the employee “e\_birthcity” column in the table “employee”.**  **UPDATE dbo.employee**  **SET e\_birthcity = 'Gandja'**  **WHERE employee\_id = 1;** |
| --- |

## 3. 2 DELETE commands for tables ((Delete\_Statements.sql)

| 1. Delete an ingredient from “ingredients” table.   **- - Before deleting any data**    **DELETE FROM dbo.ingredient**  **WHERE ingredient\_id = 3**  **-- After deleting ingredient 3 which is “Butter” and ingredient id=3** |
| --- |
| 1. Delete a crew data from the “crew” table.   **- - Before deleting any data**    **– After deleting crew\_id=1**  **DELETE FROM dbo.crew**  **WHERE crew\_id = 1** |

# **DQL (25 SQL SELECT COMMANDS & OUTPUTS)**

All clauses are in the Queries.sql file.

| 1. **--1- ordering according to names**   **SELECT customer\_id, user\_name, last\_name FROM dbo.customer ORDER BY user\_name ASC**     1. -**-2- left join from employee crew table (employee\_id values)**   **SELECT employee.employee\_id, employee.e\_department, employee.e\_lastname FROM employee LEFT JOIN crew ON crew.crew\_id = employee.employee\_id**     1. **--3- full join to see food\_info's status.**   **SELECT \* FROM food\_info FULL OUTER JOIN restaurant\_status ON restaurant\_status.status\_name = restaurant\_status.status\_name**   1. **--4-**   **SELECT ingredient.ingredient\_id, ingredient.ingredient\_name, ingredient.description, ingredient.amount, ingredient.price, ingredient.detail, customer.first\_name, customer.last\_name**  **FROM ingredient INNER JOIN dbo.customer ON ingredient.ingredient\_id = customer.customer\_id;**   1. **SELECT ingredient.ingredient\_id, ingredient.ingredient\_name, ingredient.description, ingredient.amount, ingredient.price, ingredient.detail, food\_info.food\_name, food\_info.food\_name**   **FROM ingredient FULL OUTER JOIN food\_info ON ingredient.ingredient\_id = food\_info.food\_id;**   1. **--7- ordering employees according to their birthdays**   **SELECT employee\_id, e\_firstname, e\_lastname, e\_birthdate FROM employee ORDER BY e\_birthdate desc**     1. **--8- grouping currently customers according to their country (how many customers is from USA?)**   **SELECT customer.country, COUNT(\*) AS 'Number of Currently USA residents' FROM customer WHERE country = 'USA' GROUP BY country**     1. **--8- full join to see which types of food is available**   **SELECT \* FROM food\_info FULL OUTER JOIN restaurant\_status ON food\_info.food\_id = food\_info.food\_id;**   1. **--9-- right join to see the list of employee departments with the crew names together**   **SELECT \* FROM employee RIGHT JOIN crew ON crew.crew\_name = employee.e\_department**   1. **--10-- join to see customer's email (id and score come from different tables)**   **SELECT customer.mail\_adress, first\_name FROM customer JOIN restaurant\_status ON customer.mail\_adress = customer.first\_name**   . 5 VİEW commands for tables (Delete\_Statements.sql) |
| --- |