**Logo

Description automatically generated CANADIAN INSTITUTE OF TECHNOLOGY**

**FACULTY OF ENGINEERING**

**DEPARTMENT OF COMPUTER ENGINEERING & IT**

**DATABASE DESIGN & ADMINISTRATION**

**FINAL PROJECT**

MASTER FALL 1ST YEAR

**RESTAURANT MANAGEMENT DATABASE MODEL, IMPLEMENTATION, AND WEB APP INTERFACE**

**WORKED BY** PAOLA MALLKAS

**SUPERVISED BY** DR. ADRIOLA FAQOLLI

FEBRUARY 23, 2023

**TABLE OF CONTENTS**

[**CHAPTER 1** INTRODUCTION 3](#_Toc128001250)

[**1.1 INTRODUCTION TO RESTAURANT MANAGEMENT SYSTEM** 3](#_Toc128001251)

[**1.2 OBJECTIVES OE STUDY** 4](#_Toc128001252)

[**1.3 THE FUNCTIONS THAT THE SYSTEM WILL PERFORM** 5](#_Toc128001253)

[**1.4 END USERS OF THE DATABASE** 5](#_Toc128001254)

[**1.5 TECHNICAL DESCRIPTION** 5](#_Toc128001255)

[**CHAPTER 2** SYSTEM DESIGN 6](#_Toc128001256)

[**2.1 BUSINESS RULES** 6](#_Toc128001257)

[**2.2 ENTITY RELATIONSHIP DESIGN** 6](#_Toc128001258)

[**2.2.1** DESCRIPTION OF ENTITIES 6](#_Toc128001259)

[**2.2.2** DEFINING THE STRONG AND WEAK ENTITIES. 7](#_Toc128001260)

[**2.2.3** ENTITY RELATIONSHIPS | CARDINALITY & ORDINALITY 8](#_Toc128001261)

[**2.3 DATABASE DESIGNING** 10](#_Toc128001262)

[**2.3.1 CHEN’S NOTATION** 10](#_Toc128001263)

[**2.4 DEFINE THE TABELS** 11](#_Toc128001264)

[**2.5 NORMALIZATION** 15](#_Toc128001265)

[**2.6** **CROW’S FOOT NOTATION** 20](#_Toc128001266)

[**CHAPTER 3** IMPEMENTATION PHASE 21](#_Toc128001267)

[**3.1 IMPLEMENTING TABLES IN SQL SERVER MANAGEMENT STUDIO** 21](#_Toc128001268)

[**3.2 QUERIES** 25](#_Toc128001269)

[**3.3 VIEWS** 31](#_Toc128001270)

[**3.4 STORED PROCEDURES** 32](#_Toc128001271)

[**3.5 TRIGGERS** 34](#_Toc128001272)

**PROJECT RESPONSIBILITIES**

We really enjoyed working on this database project. It was very challenging, but at the same time very interesting, putting all our knowledge into a well – structed project. The project scope first made us feel as though we couldn't complete it, but after we chose the topic “Restaurant Management System”, defining the entities and the business rules, it started to become easier. We put a lot of time into this project, and we hope we have done a good database design and implementation. This project sharpened our knowledge and made us realize how interesting Database Design and SQL are. We are really thankful to Professor Adriola for her excellent instruction and for assigning us this project.

**The project's task division is described below:**

1. Business Rules – all together
2. Description of entities, Defining the strong and weak entities – Desara
3. Entity Relationships | Cardinality & Ordinality – Reard
4. Chen’s Notation - Reard
5. Define the Tables – Paola
6. Normalization – Desara
7. Crow’s Foot Notation – Paola
8. Implementation in SQL – Paola
9. Queries – Paola
10. Views – Desara
11. Stored Procedures – Reard
12. Triggers – Paola

# INTRODUCTION

## **INTRODUCTION TO RESTAURANT MANAGEMENT SYSTEM**

Running a restaurant is difficult enough as it is, so why not make the day-to-day processes easier by having a system that will help ease the workload for you? There are so many day-to-day processes that restaurants have to deal with. These can range from scheduling employees, managing HR, monitoring employee attendance to preparing for payroll and keeping records of transactions and databases. In the current marketplace, there is a great value for food, restaurants, and its management. There is day by day increment in the number of restaurants and food places that are emerging today. It can be considered rapid growth in the field of business and food restaurants and their management system. The management system applied for every restaurant is different from the other one. Some restaurants may be bigger while others may be smaller, but every restaurant requires a management system, and this is termed as Restaurant Management System.

The proposed web base system helps when dealing with the main operations of the restaurant and a lot of other facilities. By using web base restaurant management system external users can easily get connected with the restaurant to make their inquiries, table reservations with advanced facilities such as special packages, table types, or other relevant services, and many more internal users get much freedom with web base restaurant management system.

It allows the users to perform their assigned tasks and customers to make reservations about their preferred areas. This database contains information about the employees, the customers, the tables, and all the services that are offered. It can be used by employees in a restaurant to handle the clients, their orders and can help them easily find free tables.

## **OBJECTIVES OF THE STUDY**

This project focuses mainly on the computerization of restaurant management consequent upon numerous problems faced by the manual handling of restaurant information. The objective of this system is to improve efficiency, reduce the overall manual workload of the restaurant and provide suitable technical and profitable solutions for current problems such as time consumption, less accuracy in heavy calculations, and wastage of resources.

Given below are the points that mention what are the objectives achieved by the project:

* Provide a system that helps to check the restaurant availability and do the relevant reservations efficiently and replace the reservation book. Online reservations increase the reservation process quality and make reservations efficiently.
* Automate all the possible work being done manually in the restaurant. The ability to check the current situation of the restaurant with less effort and less time with high efficiency.
* Ability to generate necessary reports quickly and easily for management decision-making. To facilitate customers to make reservation services easily.
* To provide an easy way to automate information about the day-to-day activities of the restaurant such as the record of attendance, and computing of bills, as well as an online facility for checking the availability of tables and booking the tables.

## **THE FUNCTIONS THAT THE SYSTEM WILL PERFORM**

The following proposed system aims to develop a web application for facilitating the reservation process at the restaurant.

* A customer can check the tables and reserve them, can request a table change or can request an area change.
* The manager can assign, remove, and change tasks assigned to supervisors and employees.
* Each supervisor and employee can see their tasks and change the status of the tasks as ‘Completed’ or ‘On-going’.

## **END USERS OF THE DATABASE**

End users are those people whose jobs require access to the database for querying, updating, and generating reports.

|  |  |
| --- | --- |
| CEO | Head waiter |
| Administrator | Receptionist |
| Managers | Sommelier |
| Executive chef | Bar staff |
| Kitchen manager | Waiters |
| Cooks | Bartender |
| Assistant cooks | Hostess |
| Sous-Chef | Cleaning team |

## **TECHNICAL DESCRIPTION**

As the database tool for designing the database, Microsoft SQL Server was used.

Microsoft SQL Server - Express is used to construct the database and perform queries.

Additionally, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is used to program the interface.

# SYSTEM DESIGN

## **BUSINESS RULES**

* A customer who chooses the restaurant, may make one or more reservations.
* Each restaurant area may be managed by a supervisor.
* An employee may be assigned to work at a single restaurant area.
* Each area can have many employees.
* One restaurant area can have many jobs.
* A table can have zero or many reservations.
* One or more orders can be made from a single table.
* A service can have zero or many orders.

*(For example, at the restaurant there may have been a lot of orders or none).*

* An order\_status can be assigned to many orders.

*(An order status may be “Draft”, “Ordered”, “Preparing”, “Prepared”, “Delivering”, “Taken Over”, “Cancelled”)*

* One employee can have zero or many tasks.
* A task status can be assigned to many tasks.
* An employee has only one Login account.
* One table type can be assigned to many tables.

*(For example, a lot of tables may be of the type of Classic Booth)*

* A table status can be assigned to many tables.

*(Tables may be available or reserved).*

* A country has many cities, and a city can be assigned to many customers.
* A city can also be assigned to many employees.

## **ENTITY RELATIONSHIP DESIGN**

### DESCRIPTION OF ENTITIES

* + - **Customer** - represents every client of the restaurant.
    - **Table** - represents all the tables of the restaurant.
    - **Table\_type**- describes the type of the table.
    - **Table\_status** - defines if the table is available or reserved.
    - **Reservation** - contains data about the bookings made for the tables.
    - **Employee** - represents all the employees of the restaurant.
    - **Login** - keeps a record of every employee who accesses the system.
    - **Task** - determines the duties that each employee is responsible for performing.
    - **Task\_status** - It may be “Completed” or “On Going”
    - **Employee\_Task** - connects Employee to Task.
    - **Job** - describes the job positions of the restaurant staff.
    - **Order** - represents the restaurant orders that are placed.
    - **Order\_Status** – The status that the order has. For example, “Prepared”.
    - **Service** - describes the services that the restaurant offers.
    - **Area** - identifies all the workstations where an employee can work.
    - **Country** - details the nationality of an employee or customer.
    - **City** - identifies the city an employee or customer is from.

### DEFINING THE STRONG AND WEAK ENTITIES.

### 

|  |  |
| --- | --- |
| **ENTITY** | **TYPE** |
| **Customer** | Strong |
| **Table** | Strong |
| **Table\_type** | Strong |
| **Table\_status** | Strong |
| **Reservation** | Strong |
| **Employee** | Strong |
| **Login** | Strong |
| **Task** | Strong |
| **Task\_Status** | Strong |
| **Employee\_Task** | Weak |
| **Job** | Strong |
| **Order** | Strong |
| **Order\_Status** | Strong |
| **Service** | Strong |
| **Area** | Strong |
| **Country** | Strong |
| **City** | Strong |

### ENTITY RELATIONSHIPS | CARDINALITY & ORDINALITY

* **CITY - CUSTOMER**: **[ ONE TO MANY ]**

A **city** can be added to the data of many customers, but a single **customer** only has a single city of origin.

* **TABLE\_TYPE – TABLE:** **[ ONE TO MANY ]**

A **Table\_Type** defines many tables, but one **Table** can have only one Table\_Type

* **TABLE\_STATUS – TABLE: [ ONE TO MANY ]**

One **Table** can have only one Table\_Status, but a **Table\_Status** can belong to many tables.

* **CUSTOMER – RESERVATION: [ ONE TO MANY ]**

A **customer** may book numerous reservations., but a **reservation** belongs to one and only one customer.

* **TABLE – RESERVATION: [ ONE TO MANY ]**

A **table** can have zero or many reservations, but a **reservation** belongs to only one table.

* **TABLE – ORDER: [ ONE TO MANY ]**

Zero or many orders can be made from one **table**, but an **order** belongs to only one table.

* **SERVICE – ORDER: [ ONE TO MANY ]**

A service offers zero or many orders, but one order belongs to only one service.

* **ORDER\_STATUS – ORDER: [ ONE TO MANY ]**

One **Order** can have only one Order\_Status, but a **Order\_Status** can belong to many orders.

* **CITY – EMPLOYEE: [ ONE TO MANY ]**

An employee comes from only one city, but a city may have many employees at the restaurant.

* **JOB – AREA: [ ONE TO MANY ]**

A job belongs to one and only one area, but one area may have many job positions.

* **EMPLOYEE – AREA: [ ONE TO MANY ]**

One employee is assigned to one and only one area, and an area has many employees.

* **COUNTRY – CITY: [ ONE TO MANY ]**

A county has one or more cities, but a city belongs to only one country.

* **EMPLOYEE – TASK: [ MANY TO MANY ]**

An **employee** can perform one or many tasks, and one **task** can be performed by zero or many employees.

*(Since this is a many-to-many relationship, a new entity called* ***Employee\_Task*** *is created)*

**EMPLOYEE – EMPLOYEE\_TASK:** **[ ONE TO MANY ]**

One **employee** belongs to many Employee\_Task, but one **Employee\_Task** is done by one employee.

**TASK – EMPLOYEE\_TASK: [ ONE TO MANY ]**

An **Employee\_Task** determines only one task, but a **task** can have zero or many Emplotee\_Task

* **TASK – TASK STATUS: [ ONE TO MANY ]**

A **task** has one and only one Task\_status, but a **Task\_status** can be assigned to one or many tasks.

* **EMPLOYEE – LOGIN: [ ONE TO ONE]**

An **employee** has only one Login account and a **Login account** belongs to only one employee.

## **DATABASE DESIGNING**

Database design is the process of producing a detailed data model of a database. The data in the database must be consistent and complete to get accurate results. The minimal redundancy will increase the performance of the operations handled by the database. To avoid data redundancy, the database diagram shown below is normalized to the third normal form.

### **CHEN’S NOTATION**

We started with designing the structure of the database. We drew the ER diagram on a piece of paper using a simple Chen’s Notation, placing all the entities in rectangles and using diamonds and lines to represent the relationship between entities. We tried to keep the ERD in a form where we can reduce the data redundancy and tried to make it accessible in an easy and simple way. Then, we used <https://app.diagrams.net/> to create the ER diagram which is shown below:

The link for the ERD is <file:///C:/Users/paola/Downloads/ERD%20PROJECTT%20(1).html>

Diagram

Description automatically generated

## **DEFINE THE TABELS**

We designed each table with columns and attributes respectively. The designed tables are shown below:

A screenshot of a computer

Description automatically generatedThe **CUSTOMER** TABLE has data about the customers that come to the restaurant. The Primary Key of this table is Customer\_ID.

There is a Foreign Key in this table, Customer\_City\_ID, that connects the Customer table and City table in a one-to-many relationship.

**The table has the following attributes:**

Customer\_Address which is a Composite Attribute

Customer\_Phone which is a Multivalued Attribute

Customer\_Email which is a Multivalued Attribute

A screenshot of a computer

Description automatically generated

The **TABLE** of Restaurant Table has data about the tables of the restaurant. It has one primary key, Table\_ID.

This table has 2 foreign keys, Table\_Status which has a one-to-many relationship with the Table\_Status table, and Table\_Type which has a one-to-many relationship with the Table\_Type table.

A screenshot of a computer

Description automatically generated

**TABLE\_TYPE** Table has data aboutthe type of tables.

It may be a Classic Booth Table, High Top Tables & Stools, Sofas & Coffee Table-Styled Seating. It has a primary key Table\_Type\_ID.

A screenshot of a computer

Description automatically generated

The **TABLE\_STATUS** Table defines if the table is free or reserved. Its Primary Key is Table\_Status\_ID.

A screenshot of a computer

Description automatically generated

The **RESERVATION** Table contains data about the reservations made for the table. The Primary Key for this table is Reservation\_ID.

The table has the following foreign keys:

* Reservation\_Customer\_ID which has a one-to-many relationship with the Customer table.
* Reservation\_Table\_ID which has a one-to-many relationship with the Table table.

A screenshot of a computer

Description automatically generated

The **ORDER** Table has data about the orders that are made to the restaurant. Its Primary key is Order\_ID.

The table has the following Foreign Keys:

* Order\_Service\_ID which has a one – to – many relationships with the Service Table.
* Order\_Table\_ID which has a one-to-many relationship with the Table table.

A screenshot of a computer

Description automatically generated

The **SERVICE** table contains data about the services used by the Customers. Its Primary Key is Service\_ID.

Graphical user interface, application, table, Excel

Description automatically generated

The **EMPLOYEE** table consists of data related to the employees. The primary key is Employee\_ID.

There are two foreign keys:

* Employee\_City denotes a one-to-many relationship with the City table.
* Employee\_Login\_ID denotes a one-to-one relationship with the Login table.

Graphical user interface, application, table, Excel

Description automatically generated

The **JOB** table has data about the job positions in the restaurant. Its Primary Key is Job\_ID.

It has a Foreign Key:

Job\_Area\_ID denotes a one–to–many relationship with the Area table.

A screenshot of a computer

Description automatically generated

The **AREA** table has data about all the workstations where an employee can work. Its Primary Key is Area\_ID and it has a Foreign Key: Area\_Employee\_ID denotes a one-to-many relationship with the Employee table.

Graphical user interface, application

Description automatically generatedThe **COUNTRY** Table stores data about the country an employee or a Customer comes from. Its Primary Key is Country\_ID.

Graphical user interface, application

Description automatically generated

The **CITY** Table stores data about the city an employee or a Customer comes from. Its Primary Key is City\_ID.

It has a Foreign Key:

City\_Country\_ID denotes a one-to-one relationship with the Country Table.

Graphical user interface, application, table, Excel

Description automatically generated

The **TASK** Table stores data about the tasks that should be performed by every employee. Its Primary Key is Task\_ID.

It has a Foreign Key: Task\_Status\_ID denotes a one-to-many relationship with the Task Status Table.

Graphical user interface, application, table, Excel

Description automatically generated

The **TASK STATUS** Table stores data if a task is “Completed” or “On Going”. Its Primary Key is Task\_Status\_ID.

A screenshot of a computer

Description automatically generated

The **LOGIN** table keeps track of all the employees that Log In to the system. Its Primary Key is Login\_ID.

## **NORMALIZATION**

Database normalization is the process of restructuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. Generally, if a database is normalized until the third normal form, then it is considered to be normalized. We tried to normalize the database until the third normal form. In this database, there are approximately 16 base tables. Each base table is in 3NF and has the following information associated with it: Primary Keys, Foreign Keys, Attributes, and Dependencies.

**CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| **CUSTOMER\_ID** | CUSTOMER\_FNAME | CUSTOMER\_LNAME | CUSTOMER\_ADDRESS |

|  |  |  |
| --- | --- | --- |
| CUSTOMER\_CITY\_ID | CUSTOMER\_PHONE | CUSTOMER\_EMAIL |

CUSTOMER ( **CUSTOMER\_ID**, CUSTOMER\_FNAME, CUSTOMER\_LNAME, CUSTOMER\_ADDRESS, CUSTOMER\_CITY\_ID, CUSTOMER\_PHONE, CUSTOMER\_EMAIL)

**TABLE**

|  |  |  |
| --- | --- | --- |
| **TABLE\_ID** | TABLE\_TYPE | TABLE\_STATUS |

TABLE (**TABLE\_ID**, TABLE\_TYPE, TABLE\_STATUS)

**TABLE TYPE**

|  |  |
| --- | --- |
| **TABLE\_TYPE\_ID** | TABLE\_TYPE\_NAME |

TABLE TYPE (**TABLE\_TYPE\_ID**, TABLE\_TYPE\_NAME)

**TABLE STATUS**

|  |  |
| --- | --- |
| **TABLE\_STATUS\_ID** | TABLE\_STATUS\_NAME |

TABLE STATUS (**TABLE\_STATUS\_ID**, TABLE\_STATUS\_NAME)

**RESERVATION**

|  |  |  |
| --- | --- | --- |
| **RESERVATION\_ID** | RESERVATION\_CUSTOMER\_ID | RESERVATION\_TABLE\_ID |

|  |  |
| --- | --- |
| RESERVATION\_DATE | TOTAL\_TABLES\_RESERVED |

RESERVATION (**RESERVATION\_ID**, RESERVATION\_CUSTOMER\_ID, RESERVATION\_TABLE\_ID, RESERVATION\_DATE, TOTAL\_TABLES\_RESERVED)

**EMPLOYEE**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | EMPLOYEE\_FNAME | EMPLOYEE\_LNAME | EMPLOYEE\_ADDRESS |

|  |  |  |
| --- | --- | --- |
| EMPLOYEE\_CITY\_ID | EMPLOYEE\_PHONE | EMPLOYEE\_LOGIN\_ID |

EMPOLOYEE (**EMPLOYEE\_ID**, EMPLOYEE\_FNAME, EMPLOYEE\_LNAME, EMPLOYEE\_ADDRESS, EMPLOYEE\_CITY\_ID, EMPLOYEE\_PHONE, EMPLOYEE\_LOGIN\_ID)

**LOGIN**

|  |  |  |
| --- | --- | --- |
| **LOGIN\_ID** | LOGIN\_EMAIL | LOGIN\_PASSWORD |

LOGIN (**LOGIN\_ID**, LOGIN\_EMAIL, LOGIN\_PASSWORD)

**TASK**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TASK\_ID** | TASK\_NAME | TASK\_BEGIN | TASK\_END | TASK\_STATUS\_ID |

TASK (**TASK\_ID**, TASK\_NAME, TASK\_BEGIN, TASK\_END, TASK\_STATUS\_ID)

**TASK STATUS**

|  |  |
| --- | --- |
| **TASK\_STATUS\_ID** | TASK\_STATUS\_NAME |

TASK STATUS (**TASK\_STATUS\_ID**, TASK\_STATUS\_NAME)

**JOB**

|  |  |  |  |
| --- | --- | --- | --- |
| **JOB\_ID** | JOB\_NAME | JOB\_WAGE | JOB\_AREA\_ID |

JOB (**JOB\_ID**, JOB\_NAME, JOB\_WAGE, JOB\_AREA\_ID)

**ORDER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ORDER\_ID** | TABLE\_ID | SERVICE\_ID | QUANTITY | ORDER\_DATE |

|  |  |
| --- | --- |
| ORDER\_PRICE | ORDER\_STATUS |

ORDER (**ORDER\_ID**, TABLE\_ID, SERVICE\_ID, QUANTITY, ORDER\_DATE, ORDER\_PRICE, ORDER\_STATUS)

**ORDER\_STATUS**

|  |  |
| --- | --- |
| **ORDER\_STATUS\_ID** | ORDER\_STATUS\_NAME |

ORDER STATUS (**ORDER\_STATUS\_ID**, ORDER\_STATUS\_NAME)

**SERVICE**

|  |  |  |
| --- | --- | --- |
| **SERVICE\_ID** | SERVICE\_NAME | AREA\_ID |

SERVICE (**SERVICE\_ID**, SERVICE\_NAME, AREA\_ID)

**AREA**

|  |  |  |
| --- | --- | --- |
| **AREA\_ID** | AREA\_NAME | AREA\_EMPLOYEE\_ID |

AREA (**AREA\_ID**, AREA\_NAME, AREA\_EMPLOYEE\_ID)

**COUNTRY**

|  |  |
| --- | --- |
| **COUNTRY\_ID** | COUNTRY\_NAME |

COUNTRY (**COUNTRY\_ID**, COUNTRY\_NAME)

**CITY**

|  |  |  |
| --- | --- | --- |
| **CITY\_ID** | CITY\_NAME | CITY\_COUNTRY\_ID |

CITY (**CITY\_ID**, CITY\_NAME, CITY\_COUNTRY\_ID)

## **CROW’S FOOT NOTATION**

# IMPEMENTATION PHASE

## **IMPLEMENTING TABLES IN SQL SERVER MANAGEMENT STUDIO**

CREATE DATABASE [RMS]

GO

USE [RMS]

GO

CREATE TABLE [dbo].[Table\_type](

[table\_type\_ID] [int] IDENTITY(1,1) NOT NULL,

[table\_type\_name] [varchar](30) NOT NULL,

CONSTRAINT PK\_TableType PRIMARY KEY (table\_type\_ID)

)

CREATE TABLE [dbo].[Table\_status](

[table\_status\_ID] [int] IDENTITY(1,1) NOT NULL,

[table\_status\_name] [varchar](30) NOT NULL,

CONSTRAINT PK\_TableStatus PRIMARY KEY (table\_status\_ID)

)

CREATE TABLE [dbo].[TableofRestaurant](

[table\_ID] [int] IDENTITY(1,1) NOT NULL,

[table\_type] [int],

[table\_status] [int],

[table\_capacity] [int],

CONSTRAINT PK\_TableofRestaurant PRIMARY KEY (table\_ID),

CONSTRAINT FK\_TableofRestaurant\_TableType FOREIGN KEY (table\_type)

REFERENCES [Table\_type] (table\_type\_ID),

CONSTRAINT FK\_TableofRestaurant\_TableStatus FOREIGN KEY (table\_status)

REFERENCES [Table\_status] (table\_status\_ID)

)

CREATE TABLE [dbo].[Country](

[country\_ID] [int] IDENTITY(1,1) NOT NULL,

[country\_name] [varchar](150) NOT NULL,

CONSTRAINT PK\_Country PRIMARY KEY (country\_ID)

)

CREATE TABLE [dbo].[City](

[city\_ID] [int] IDENTITY(1,1) NOT NULL,

[city\_name] [varchar](30) NOT NULL,

[city\_country\_ID] [int],

CONSTRAINT PK\_City PRIMARY KEY (city\_ID),

CONSTRAINT FK\_City\_Country FOREIGN KEY (city\_country\_ID)

REFERENCES [Country] (country\_ID),

)

CREATE TABLE [dbo].[Customer](

[customer\_ID] [int] IDENTITY(1,1) NOT NULL,

[customer\_fname] [varchar](30),

[customer\_lname] [varchar](30),

[customer\_add] [varchar](30),

[customer\_city\_ID] [int],

[customer\_phone] [varchar](30),

[customer\_email] [varchar](30),

CONSTRAINT PK\_Customer PRIMARY KEY (customer\_ID),

CONSTRAINT FK\_Customer\_City FOREIGN KEY (customer\_city\_ID)

REFERENCES [City] (city\_ID),

)

CREATE TABLE [dbo].[Reservation](

[reservation\_ID] [int] IDENTITY(1,1) NOT NULL,

[reservation\_customer\_ID] [int],

[reservation\_table\_ID] [int],

[reservation\_date] [date],

[total\_tables\_reserved] [int],

CONSTRAINT PK\_Reservation PRIMARY KEY (reservation\_ID),

CONSTRAINT FK\_Reservation\_Table FOREIGN KEY (reservation\_table\_ID)

REFERENCES [TableofRestaurant] (table\_ID),

CONSTRAINT FK\_Reservation\_Customer FOREIGN KEY (reservation\_customer\_ID) REFERENCES [Customer] (customer\_ID),

)

CREATE TABLE [dbo].[Login](

[login\_ID] [int] IDENTITY(1,1) NOT NULL,

[login\_email] [varchar] NOT NULL,

[login\_password] [varchar] NOT NULL,

CONSTRAINT PK\_Login PRIMARY KEY (login\_ID),

CONSTRAINT Login\_Email\_Unqiue UNIQUE (login\_email)

)

CREATE TABLE [dbo].[Employee](

[employee\_ID] [int] IDENTITY(1,1) NOT NULL,

[employee\_fname] [varchar](100),

[employee\_lname] [varchar](100),

[employee\_add] [varchar](100),

[employee\_city\_ID] [int],

[employee\_phone] [varchar](30),

[employee\_login\_id] [int],

[employee\_job\_id] [int],

CONSTRAINT PK\_Employee PRIMARY KEY (employee\_ID),

CONSTRAINT FK\_Employee\_Login FOREIGN KEY (employee\_login\_ID)

REFERENCES [Login] (login\_ID),

CONSTRAINT FK\_Employee\_City FOREIGN KEY (employee\_city\_ID)

REFERENCES [City] (City\_ID),

)

CREATE TABLE [dbo].[Area](

[area\_ID] [int] IDENTITY(1,1) NOT NULL,

[area\_name] [varchar] ,

[area\_employee\_id] [int],

CONSTRAINT PK\_Area PRIMARY KEY (area\_ID),

CONSTRAINT FK\_Area\_Employee FOREIGN KEY (area\_employee\_id)

REFERENCES [Employee] (employee\_ID),

)

CREATE TABLE [dbo].[Job](

[job\_ID] [int] IDENTITY(1,1) NOT NULL,

[job\_name] [varchar] ,

[job\_wage] [decimal](10,2),

[job\_area\_id] [int],

CONSTRAINT PK\_Job PRIMARY KEY (job\_ID),

CONSTRAINT FK\_Job\_Area FOREIGN KEY (job\_area\_ID)

REFERENCES [Area] (area\_ID),

)

CREATE TABLE [dbo].[Service](

[service\_ID] [int] IDENTITY(1,1) NOT NULL,

[service\_name] [varchar](30),

CONSTRAINT PK\_Service PRIMARY KEY (service\_ID),

)

CREATE TABLE [dbo].[Order\_status](

[order\_status\_ID] [int] IDENTITY(1,1) NOT NULL,

[order\_status\_name] [varchar](30) NOT NULL,

CONSTRAINT PK\_OrderStatus PRIMARY KEY (order\_status\_ID)

)

CREATE TABLE [dbo].[Order1](

[order\_ID] [int] IDENTITY(1,1) NOT NULL,

[table\_ID] [int],

[service\_ID] [int] ,

[quantity] [int] ,

[order\_date] [date],

[order\_price] [decimal](10,2),

[order\_order\_status\_ID] [int],

CONSTRAINT PK\_Order1 PRIMARY KEY (order\_ID),

CONSTRAINT FK\_Order1\_Table FOREIGN KEY (table\_ID)

REFERENCES [TableofRestaurant] (table\_ID),

CONSTRAINT FK\_Order1\_Service FOREIGN KEY (service\_ID) REFERENCES [Service] (service\_ID),

CONSTRAINT FK\_Order1\_OrderStatus FOREIGN KEY (order\_order\_status\_ID) REFERENCES [Order\_status] (order\_status\_ID),

)

CREATE TABLE [dbo].[Task\_status](

[task\_status\_ID] [int] IDENTITY(1,1) NOT NULL,

[task\_status\_name] [varchar](30) NOT NULL,

CONSTRAINT PK\_TaskStatus PRIMARY KEY (task\_status\_ID)

)

CREATE TABLE [dbo].[Task](

[task\_ID] [int] IDENTITY(1,1) NOT NULL,

[task\_name] [varchar](30) NOT NULL,

[task\_begin] [date],

[task\_end] [date],

[task\_task\_status\_ID] [int],

CONSTRAINT PK\_Task PRIMARY KEY (task\_ID),

CONSTRAINT FK\_Task\_TaskStatus FOREIGN KEY (task\_task\_status\_ID) REFERENCES [Task\_status] (task\_status\_ID),

)

CREATE TABLE [dbo].[Employee\_Task](

[employee\_ID] [int] NOT NULL,

[task\_ID] [int] NOT NULL,

CONSTRAINT PK\_Emp\_Task PRIMARY KEY (employee\_ID, task\_ID),

CONSTRAINT FK\_Employee FOREIGN KEY (employee\_ID)

REFERENCES Employee(employee\_ID),

CONSTRAINT FK\_Task\_Rel FOREIGN KEY (task\_ID)

REFERENCES Task(task\_ID)

);

## **QUERIES**

1. QUERY 1

**-- How many city customers have made reservations for a particular month?**

**-- IN Subquery**

SELECT customer\_fname, customer\_lname,customer\_phone

FROM Customer

WHERE customer\_ID IN

( SELECT distinct customer\_city\_ID -- get city customers

FROM Reservation

WHERE MONTH(reservation\_date) = 8); -- reservations for the month of August

1. QUERY 2

**-- How many reservations has a customer made in one year?**

**-- Aggregate Function**

SELECT COUNT (\*) AS 'Total Number of Reservations' -- count of total reservations.

FROM Reservation

WHERE YEAR(reservation\_date) = 2023 AND reservation\_customer\_ID = 1; -- bookings in Year 2023 by customer with id 1

1. QUERY 3

**-- How many tables are reserved on a given date?**

SELECT SUM(total\_tables\_reserved) AS 'Total Tables Reserved' -- sum of total tables

FROM Reservation

WHERE reservation\_date LIKE '2023-16-02%'; -- for date 16th February,2023;

1. QUERY 4

**-- Information of employee with login id = 1**

SELECT \*

FROM Employee

WHERE Employee.employee\_ID = 1;

1. QUERY 5

**-- All employees from Albania, whose name starts with A**

SELECT Employee.employee\_ID, Employee.employee\_lname, Employee.employee\_fname, Country.country\_name

FROM Employee

INNER JOIN City ON Employee.employee\_city\_ID = City.city\_ID

INNER JOIN Country ON City.city\_country\_ID = Country.country\_ID

WHERE country\_name = 'Albania'

AND Employee.employee\_fname LIKE 'A%'

OR Employee.employee\_lname LIKE 'A%';

1. QUERY 6

**--All employees from Tirana ordered by full name**

SELECT Employee.employee\_ID, Employee.employee\_fname, Employee.employee\_lname, Employee.employee\_phone, City.city\_name

FROM Employee

JOIN City ON Employee.employee\_city\_ID = City.city\_ID

WHERE City.city\_name = 'Tirane'

ORDER BY Employee.employee\_fname, Employee.employee\_lname

1. QUERY 7

**--Information for employees with wages greater than 500**

SELECT Employee.employee\_ID, Employee.employee\_fname, Employee.employee\_lname, Job.job\_name, Job.job\_wage

FROM Employee,Area, Job

WHERE Employee.employee\_ID = Area.area\_employee\_id

AND Area.area\_ID= Job.job\_area\_id

AND Job.job\_wage > 500

1. QUERY 20

**--Display the current salary for each employee in the restaurant**

SELECT employee\_ID, employee\_fname, employee\_lname, job\_name, job\_wage

FROM Employee, Area, Job

WHERE Employee.employee\_ID = Area.area\_employee\_id

AND Area.area\_ID = Job.job\_area\_id

ORDER BY job\_wage DESC;

1. QUERY 8

**-- All Customers from Tirana ordered by full name**

SELECT Customer.customer\_ID, Customer.customer\_fname, Customer.customer\_lname, City.city\_name

FROM Customer

JOIN City ON Customer.customer\_city\_ID = City.city\_ID

WHERE City.city\_name = 'Tirana'

ORDER BY Customer.customer\_fname, Customer.customer\_lname;

1. QUERY 9

**--All customers with a reservation in January 2023 ordered by customer’s full name**

SELECT Customer.customer\_ID, Customer.customer\_fname, Customer.customer\_lname, Reservation.reservation\_date

FROM Customer, Reservation

WHERE Customer.customer\_ID = Reservation.reservation\_customer\_ID

AND Month (Reservation.reservation\_date) = 1

ORDER BY Customer.customer\_fname, Customer.customer\_lname;

1. QUERY 10

**--All reservations on 2023 group by month**

SELECT COUNT (Reservation.reservation\_ID) AS Number\_of\_reservations,

MONTH ( Reservation.reservation\_date) AS Month\_Number

FROM Reservation

WHERE Year (Reservation.reservation\_date) = 2023

GROUP BY Month (Reservation.reservation\_date);

1. QUERY 11

**--All customers that have reserved a classic booth table**

SELECT Customer.customer\_ID, Customer.customer\_fname, Customer.customer\_lname, Table\_type.table\_type\_name

FROM Customer, Reservation, TableofRestaurant, Table\_type

WHERE Customer.customer\_ID = Reservation.reservation\_customer\_ID

AND Reservation.reservation\_table\_ID = TableofRestaurant.table\_ID

AND TableofRestaurant.table\_type = Table\_type.table\_type\_ID

AND Table\_type.table\_type\_name like '%classic booth%'

1. QUERY 12

**--Number of CUSTOMERS that have reserved a Classic Booth Table or High Top Tables.**

**--Group by table type**

SELECT COUNT (Customer.customer\_ID) AS Number\_of\_customers, Table\_type.table\_type\_name

FROM Customer, Reservation, TableofRestaurant, Table\_type

WHERE Customer.customer\_ID = Reservation.reservation\_customer\_ID

AND Reservation.reservation\_table\_ID = TableofRestaurant.table\_ID

AND TableofRestaurant.table\_type = Table\_type.table\_type\_ID

AND (Table\_type.table\_type\_name like '%Classic Booth%'

OR Table\_type.table\_type\_name like '%High top table%')

GROUP BY Table\_type.table\_type\_name

1. QUERY 13

**--Display a report of the restaurant on 2023-02-19**

SELECT Customer.customer\_ID, Customer.customer\_fname, Customer.customer\_lname,

Reservation.reservation\_ID, Reservation.reservation\_date,

TableofRestaurant.table\_ID,TableofRestaurant.table\_status

FROM Customer, Reservation, TableofRestaurant

WHERE Customer.customer\_ID = Reservation.reservation\_customer\_ID

AND Reservation.reservation\_table\_ID = TableofRestaurant.table\_ID

AND Reservation.reservation\_date = '2023-02-19';

1. QUERY 14

**/\*For each city that has at least one customer that has a reservation in our restaurant, display the total number of customers that fulfill these criteria.\*/**

SELECT City.city\_name, COUNT (Customer.customer\_ID) AS Total\_Number

FROM City

JOIN Customer ON City.city\_ID = Customer.customer\_city\_ID

JOIN Reservation ON Customer.customer\_ID = Reservation.reservation\_customer\_ID

GROUP BY City.city\_name

HAVING COUNT (Reservation.reservation\_customer\_ID) > 1

1. QUERY 15

**-- All orders that have more than one service requested**

SELECT Order1.order\_ID, COUNT(Service.service\_ID) AS Toral\_Number\_of\_Services

FROM Order1, Service

WHERE Order1.service\_ID = Service.service\_ID

GROUP BY Order1.order\_ID

HAVING COUNT(Service.service\_ID) > 1;

1. QUERY 16

**--For each city employee display the max and min wage**

SELECT City.city\_name,

MAX(Job.job\_wage) AS max\_wage, MIN(Job.job\_wage) AS min\_wage

FROM City

JOIN Employee ON City.city\_ID = Employee.employee\_city\_ID

JOIN Area ON Employee.employee\_ID = Area.area\_employee\_id

JOIN Job ON Area.area\_ID = Job.job\_area\_id

GROUP BY City.city\_name;

1. QUERY 17

**-- Display all the employees that have completed today's tasks**

SELECT Employee.employee\_fname, Employee.employee\_lname, Task.task\_name

FROM Employee, Employee\_Task,Task,Task\_status

WHERE Employee.employee\_ID = Employee\_Task.employee\_ID

AND Employee\_Task.task\_ID = Task.task\_ID

AND Task.task\_task\_status\_ID = Task\_status.task\_status\_ID

AND Task\_status.task\_status\_name = 'Completed'

AND Task.task\_begin = GETDATE ();

1. QUERY 18

**/\*Show all employee information for employees, whose name starts with J using subquery\*/**

SELECT \*

FROM Employee

WHERE employee\_ID in (SELECT employee\_ID

FROM Employee

WHERE employee\_fname like 'J%'

);

1. QUERY 19

**-- Display a report of the restaurant for the current date using subquery**

SELECT Customer.customer\_ID, Customer.customer\_fname, Customer.customer\_lname,

Reservation.reservation\_ID, Reservation.reservation\_date

FROM Customer, Reservation

WHERE Customer.customer\_ID = (SELECT Reservation.reservation\_customer\_ID

FROM Reservation

WHERE Reservation.reservation\_date = '2023-19-02'

AND Customer.customer\_ID = Reservation.reservation\_customer\_ID

)

## **VIEWS**

1. VIEW 1

**-- create a view named restaurant\_customers to get details of the customers**

CREATE VIEW restaurant\_customers AS

SELECT Customer.customer\_fname AS 'First Name', Customer.customer\_lname AS 'Last Name',

Customer.customer\_email AS 'Email Address', Customer.customer\_phone AS 'Contact Number'

FROM Customer

JOIN City ON City.city\_ID = Customer.customer\_city\_ID

WHERE Customer.customer\_ID IN

(SELECT DISTINCT customer\_ID

FROM Reservation)

1. VIEW 2

**-- create a view named restaurant\_employees to get details of all the employees**

CREATE VIEW restaurant\_employees AS

SELECT employee\_fname AS 'First Name', employee\_lname AS 'Last Name', employee\_phone AS 'Contact Number'

FROM Employee

JOIN Area

ON Area.area\_employee\_id = Employee.employee\_ID;

## **STORED PROCEDURES**

1. STORED PROCEDURE - Example 1

**--A stored procedure that makes the reservation of a table**

CREATE PROCEDURE ReserveTable (@customer\_id AS INT,@table\_id AS INT, @reservation\_date as DATE )

AS

BEGIN

SELECT table\_ID FROM TableofRestaurant WHERE table\_ID = @table\_id

AND table\_status = (SELECT table\_status\_ID FROM Table\_status WHERE table\_status\_name = 'Avaiable');

IF @@ROWCOUNT = 1

INSERT INTO [dbo].[Reservation]

([reservation\_customer\_ID]

,[reservation\_table\_ID]

,[reservation\_date]

)

VALUES

(@customer\_id

,@table\_id

,GETDATE()

)

END

GO

1. STORED PROCEDURE – Example 2

**-- Remove Table Reservation**

CREATE PROCEDURE RemoveReservation (@reservation\_id AS INT)

AS

BEGIN

SELECT table\_ID FROM TableofRestaurant

WHERE table\_ID = (SELECT reservation\_table\_ID

FROM Reservation

WHERE reservation\_ID = @reservation\_id)

AND table\_status = (SELECT table\_status\_ID

FROM Table\_status WHERE table\_status\_name = 'Reserved');

IF @@ROWCOUNT = 1

BEGIN

UPDATE TableofRestaurant

SET table\_status = (SELECT table\_status\_ID FROM Table\_status WHERE table\_status\_name = 'Reserved');

END

DELETE FROM [dbo].[Reservation] WHERE reservation\_ID = @reservation\_id ;

END

GO

## **TRIGGERS**

1. TRIGGER – Example 1

**-- A trigger that notifies the system administrator when someone tries to insert or update the job table**

USE RMS

GO

CREATE TRIGGER Notify\_me

ON Job

AFTER INSERT, UPDATE

AS RAISERROR ('Notify system administrator', 16, 10)

GO

1. TRIGGER – Example 2

**-- Trigger that inserts information into employee table**

CREATE TRIGGER employee\_information

ON Employee

AFTER INSERT

AS

BEGIN

DECLARE

@EMP\_id int ,

@EMP\_first\_name varchar(150),

@EMP\_last\_name varchar(150),

@EMP\_address varchar(150),

@EMP\_phone\_number varchar(150)

SELECT

@EMP\_id = inserted.employee\_ID,

@EMP\_first\_name = inserted.employee\_fname,

@EMP\_last\_name = inserted.employee\_lname,

@EMP\_address = inserted.employee\_add,

@EMP\_phone\_number = inserted.employee\_phone

FROM

inserted

--Check if these is the employee\_ID. If there is the employee\_ID, do not insert.

IF EXISTS (SELECT \*

FROM Employee

WHERE [employee\_ID]= @EMP\_id)

PRINT('This ID is already in the table')

ELSE

-- If there is not the employee\_ID, do an INSERT

INSERT INTO Employee (

employee\_ID, employee\_fname, employee\_lname,employee\_add, employee\_phone)

VALUES(@EMP\_id, @EMP\_first\_name, @EMP\_last\_name,@EMP\_address,@EMP\_phone\_number)

END

GO

1. TRIGGER – Example 3

**-- The following code shows how LOGON Trigger rejects attempts to logon for "user1",**

**--if they are initiated outside business hours 9:00-21:00.**

-- 1. Create a SQL user called "user1" on the server.

USE RMS

GO

--Create the login on the servel called "user1"

CREATE LOGIN [user1] WITH PASSWORD= N'Aa123456&'

,DEFAULT\_DATABASE=[RMS]

,DEFAULT\_LANGUAGE=[us\_english]

,CHECK\_EXPIRATION=OFF

,CHECK\_POLICY=OFF

GO

--2. Create logon trigger called “limit\_hours\_of\_use\_triggerr”,

--which only permits login attempts for “user1″ during business hours i.e. between 9:00 and 21:00 hours.

CREATE TRIGGER [limit\_hours\_of\_use\_trigger]

ON ALL SERVER

FOR LOGON

AS

BEGIN

DECLARE @ErrorText [varchar](128)

SET @ErrorText = 'Cannot allow login to "user1" outside of normal business hours. '

SET @ErrorText = @ErrorText + 'Please try again between business hours 9:00 and 21:00.'

IF ORIGINAL\_LOGIN() = 'user1' AND

(DATEPART(HOUR, GETDATE()) < 9 OR DATEPART (HOUR, GETDATE()) > 21)

BEGIN

PRINT @ErrorText

ROLLBACK;

END

END;

GO

ENABLE TRIGGER [limit\_hours\_of\_use\_trigger] ON ALL SERVER

GO