

DBMS - Mini Project
Restaurant Table
Booking System

Submitted by-
Name- Arushi Pandey
SRN- PES1UG20CS077
V Semester
Section - B

Short Description and Scope of the Project

Restaurant reservation systems **help manage the constant influx of reservations and customers** – allowing the customers to book their tables remotely so managers can schedule resources according to the number of bookings

This project consists of UI in which customers can –

- View the vacant tables in a restaurant.
- See reviews of the selected restaurant which will help the customer in deciding which restaurant to book a table in.
- They can make reservations in a particular restaurant for a specific date and time.
- They can view, edit and delete the reservations as well.
- View vacant tables in a particular area(For example Koramangala).
- Find the names of customers who have made atleast one reservation.

Scope of the project-

- For customers to get effective deals
- For customers to go to likeable place
- Reservations of later dining times
- Reducing wait times
- Providing options nearby
- Promoting and advertising their restaurants to get like minded or repeat buisness

Note-

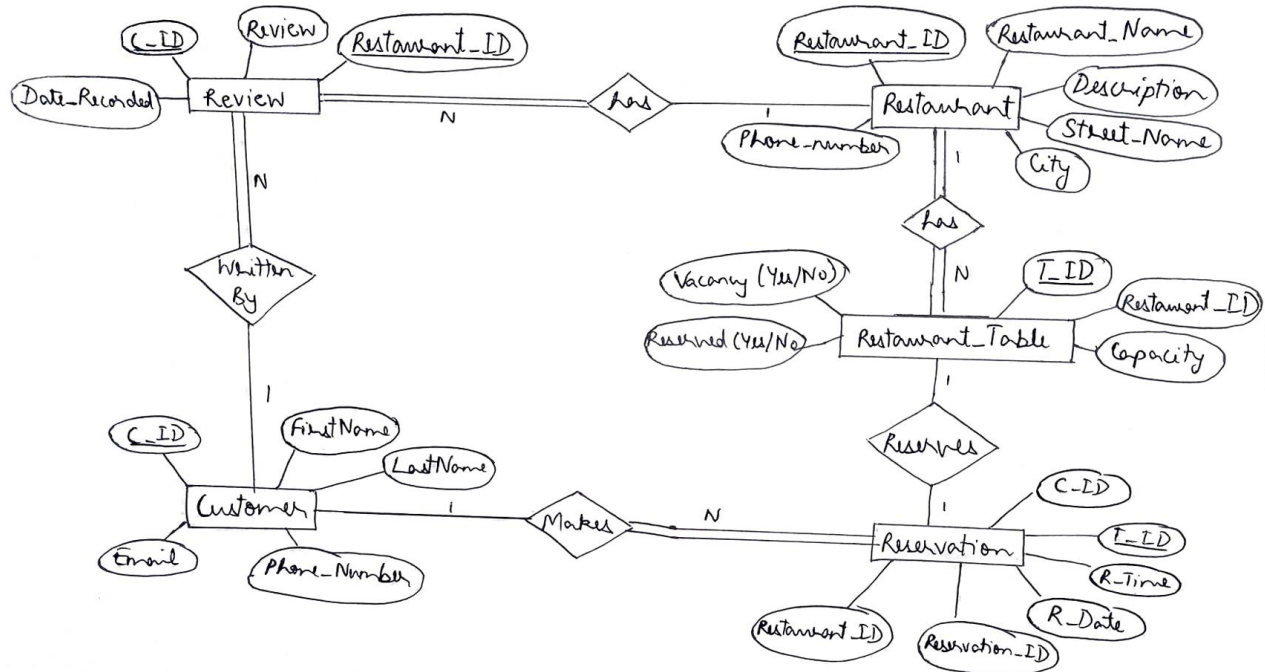
(CRUD operations have been implemented for all 5 tables.

Also we can view data for all 5 tables.

Join,procedure, trigger, set operations and aggregate operations have also been implemented in frontend.)

ER Diagram

RESTAURANT TABLE BOOKING SYSTEM



Relational Schema

Review

<u>C_ID</u>	<u>Restaurant_ID</u>	Review	Date_Recorded
-------------	----------------------	--------	---------------

Reservation

<u>Reservation_ID</u>	<u>T_ID</u>	C_ID	Restaurant_ID	R_Date	R_Time
-----------------------	-------------	------	---------------	--------	--------

Restaurant_Table

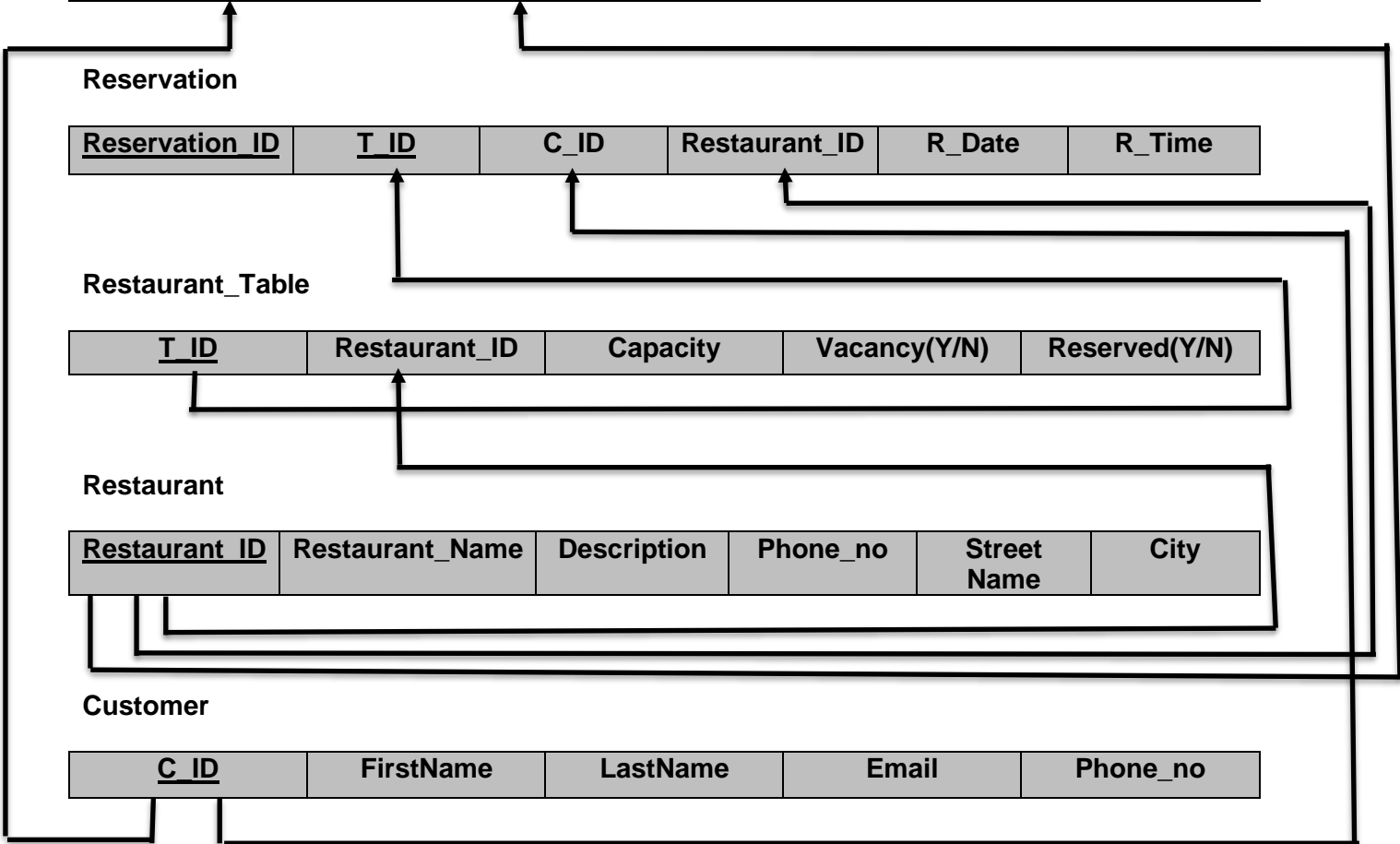
<u>T_ID</u>	Restaurant_ID	Capacity	Vacancy(Y/N)	Reserved(Y/N)
-------------	---------------	----------	--------------	---------------

Restaurant

<u>Restaurant_ID</u>	Restaurant_Name	Description	Phone_no	Street Name	City
----------------------	-----------------	-------------	----------	-------------	------

Customer

<u>C_ID</u>	FirstName	LastName	Email	Phone_no
-------------	-----------	----------	-------	----------



DDL statements - Building the database

Creating tables-

```
Command Prompt - mysql -u root
MariaDB [(none)]> CREATE DATABASE restaurant_table_booking_system;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> USE restaurant_table_booking_system;
Database changed
MariaDB [restaurant_table_booking_system]> CREATE TABLE Customer(
  -> C_ID int PRIMARY KEY,
  -> FirstName varchar(255),
  -> LastName varchar(255),
  -> Email_ID varchar(255),
  -> Phone_number bigint
  -> );
Query OK, 0 rows affected (0.047 sec)

MariaDB [restaurant_table_booking_system]> CREATE TABLE Restaurant(
  -> Restaurant_ID int PRIMARY KEY,
  -> Restaurant_Name varchar(255),
  -> Description varchar(255),
  -> Phone_number bigint,
  -> Street_name varchar(255),
  -> City varchar(255)
  -> );
Query OK, 0 rows affected (0.036 sec)

MariaDB [restaurant_table_booking_system]>
```

```
Command Prompt - mysql -u root
MariaDB [restaurant_table_booking_system]> CREATE TABLE Restaurant_Table(
  -> T_ID int,
  -> Restaurant_ID int,
  -> Capacity int,
  -> Vacancy varchar(5),
  -> Reserved varchar(5),
  -> PRIMARY KEY(T_ID),
  -> FOREIGN KEY (Restaurant_ID) REFERENCES Restaurant(Restaurant_ID) ON DELETE CASCADE
  -> );
Query OK, 0 rows affected (0.047 sec)

MariaDB [restaurant_table_booking_system]> CREATE TABLE reservation(
  -> Reservation_ID int NOT NULL UNIQUE,
  -> T_ID int NOT NULL UNIQUE,
  -> C_ID int,
  -> Restaurant_ID int,
  -> R_Date Date,
  -> R_Time Time,
  -> PRIMARY KEY(Reservation_ID,T_ID),
  -> FOREIGN KEY (Restaurant_ID) REFERENCES Restaurant(Restaurant_ID) ON DELETE CASCADE,
  -> FOREIGN KEY (C_ID) REFERENCES Customer(C_ID) ON DELETE CASCADE,
  -> FOREIGN KEY (T_ID) REFERENCES Restaurant_Table(T_ID) ON DELETE CASCADE
  -> );
Query OK, 0 rows affected (0.067 sec)

MariaDB [restaurant_table_booking_system]>
```

```
Command Prompt - mysql -u root
MariaDB [restaurant_table_booking_system]> CREATE TABLE Review(
  -> C_ID int,
  -> Restaurant_ID int,
  -> Review varchar(255),
  -> Date_Recorded Date,
  -> PRIMARY KEY(C_ID,Restaurant_ID),
  -> FOREIGN KEY (C_ID) REFERENCES Customer(C_ID) ON DELETE CASCADE,
  -> FOREIGN KEY (Restaurant_ID) REFERENCES Restaurant(Restaurant_ID) ON DELETE CASCADE
  -> );
Query OK, 0 rows affected (0.065 sec)

MariaDB [restaurant_table_booking_system]>
```

Populating the Database

```
Command Prompt - mysql -u root

MariaDB [restaurant_table_booking_system]> LOAD DATA INFILE "Customers.csv" INTO TABLE Customer
  -> COLUMNS TERMINATED BY ','
  -> OPTIONALLY ENCLOSED BY '"'
  -> ESCAPED BY ''
  -> LINES TERMINATED BY '\n'
  -> IGNORE 1 LINES;
Query OK, 7 rows affected, 7 warnings (0.010 sec)
Records: 7 Deleted: 0 Skipped: 0 Warnings: 7

MariaDB [restaurant_table_booking_system]> LOAD DATA INFILE "Restaurant.csv" INTO TABLE Restaurant
  -> COLUMNS TERMINATED BY ','
  -> OPTIONALLY ENCLOSED BY '"'
  -> ESCAPED BY ''
  -> LINES TERMINATED BY '\n'
  -> IGNORE 1 LINES;
Query OK, 7 rows affected (0.012 sec)
Records: 7 Deleted: 0 Skipped: 0 Warnings: 0

MariaDB [restaurant_table_booking_system]>
MariaDB [restaurant_table_booking_system]> LOAD DATA INFILE "Table.csv" INTO TABLE restaurant_table
  -> COLUMNS TERMINATED BY ','
  -> OPTIONALLY ENCLOSED BY '"'
  -> ESCAPED BY ''
  -> LINES TERMINATED BY '\n'
  -> IGNORE 1 LINES;
Query OK, 10 rows affected (0.011 sec)
Records: 10 Deleted: 0 Skipped: 0 Warnings: 0

MariaDB [restaurant_table_booking_system]>
```

```
Command Prompt - mysql -u root

MariaDB [restaurant_table_booking_system]> LOAD DATA INFILE "Reviews.csv" INTO TABLE Review
  -> COLUMNS TERMINATED BY ','
  -> OPTIONALLY ENCLOSED BY '"'
  -> ESCAPED BY ''
  -> LINES TERMINATED BY '\n'
  -> IGNORE 1 LINES;
Query OK, 7 rows affected (0.026 sec)
Records: 7 Deleted: 0 Skipped: 0 Warnings: 0

MariaDB [restaurant_table_booking_system]> _
```

Join Queries

Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

Find the names of customers who have made atleast 1 reservation:

SQL Query execution-

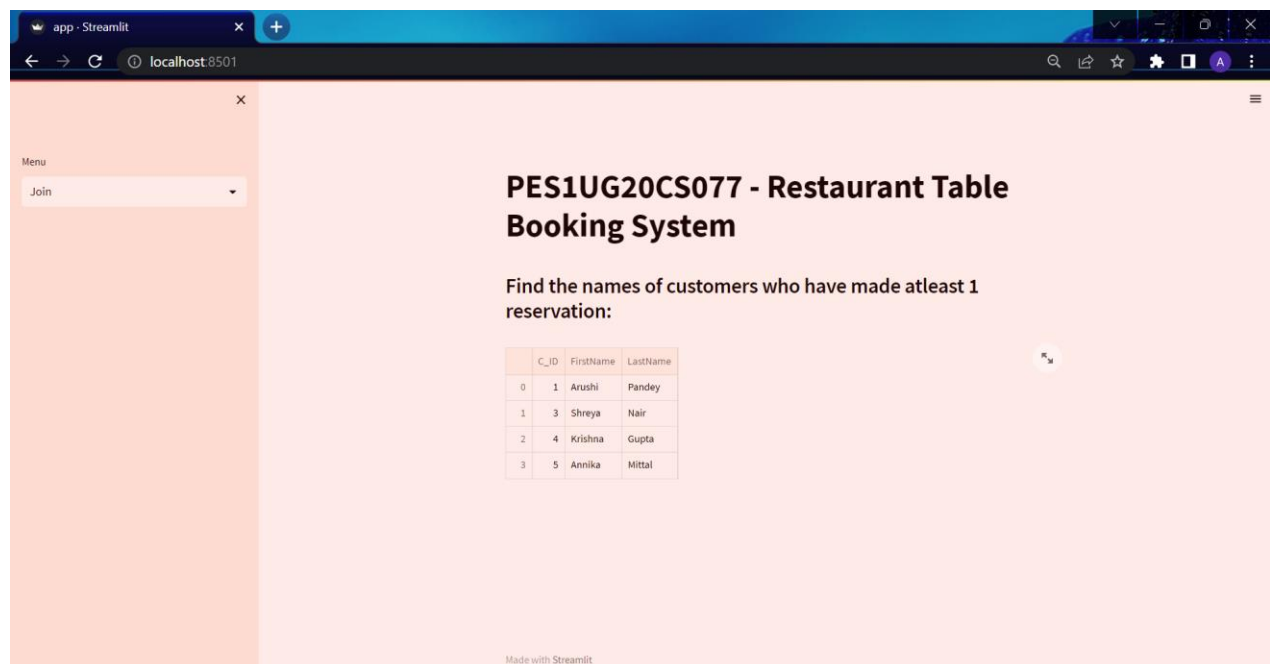
```
MariaDB [(none)]> use restaurant_table_booking_system;
Database changed
MariaDB [restaurant_table_booking_system]> SELECT C.C_ID,C.FirstName,C.LastName FROM Customer C INNER JOIN Reservation R
ON C.C_ID=R.C_ID;
+-----+-----+-----+
| C_ID | FirstName | LastName |
+-----+-----+-----+
| 1 | Arushi | Pandey |
| 3 | Shreya | Nair |
| 4 | Krishna | Gupta |
| 5 | Annika | Mittal |
+-----+-----+-----+
4 rows in set (0.030 sec)

MariaDB [restaurant_table_booking_system]> |
```

Python code-

```
def join2():
    c.execute('SELECT C.C_ID,C.FirstName,C.LastName FROM
Customer C INNER JOIN Reservation R ON C.C_ID=R.C_ID')
    data=c.fetchall()
    return data
```

UI Screenshot



Aggregate Functions

Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

Query to find the number of vacant tables

SQL Query Execution-

```
MariaDB [restaurant_table_booking_system]> SELECT COUNT(T_ID) FROM Restaurant_table WHERE vacancy="Yes";
+-----+
| COUNT(T_ID) |
+-----+
|          5 |
+-----+
1 row in set (0.010 sec)

MariaDB [restaurant_table_booking_system]> |
```

Python code

```
def agg_cnt2():
    c.execute('SELECT COUNT(T_ID) FROM Restaurant_table WHERE vacancy="Yes"')
    data=c.fetchall()
    return data
```

View the number of vacant tables:

Number of Vacant Tables:

Number of Vacant Tables:	
0	5

Set Operations

Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

Find the restaurants who have vacant tables and are present in Koramangala-

SQL Query Execution-

```
MariaDB [restaurant_table_booking_system]> SELECT * FROM Restaurant WHERE Restaurant_ID IN (SELECT Restaurant_ID from Restaurant WHERE Street_Name="Koramangala" INTERSECT SELECT Restaurant_ID from Restaurant_Table WHERE Vacancy="Yes");
```

Restaurant_ID	Restaurant_Name	Description	Phone_number	Street_name	City
3	Mainland China	Chinese	7892279326	Koramangala	Bengaluru
4	Barbeque Nation	Buffet	9557522075	Koramangala	Bengaluru

2 rows in set (0.004 sec)

```
MariaDB [restaurant_table_booking_system]> |
```

Python code-

```
def set_oper2():  
    c.execute('SELECT * FROM Restaurant WHERE Restaurant_ID IN  
(SELECT Restaurant_ID from Restaurant WHERE  
Street_Name="Koramangala" INTERSECT SELECT Restaurant_ID from  
Restaurant_Table WHERE Vacancy="Yes")')  
    data=c.fetchall()  
    return data
```

UI Screenshot-

Find the restaurants who have vacant tables and are present in Koramangala:

View Restaurants Details						
	Restaurant_ID	Restaurant_Name	Description	Phone_number	Street_name	City
0	3	Mainland China	Chinese	7892279326	Koramangala	Bengaluru
1	4	Barbeque Nation	Buffet	9557522075	Koramangala	Bengaluru

Functions and Procedures

Create a Function and Procedure. State the objective of the function / Procedure. Run and display the results.

Write a procedure to display the reviews given the name of a particular restaurant.

```
Command Prompt - mysql -u root

MariaDB [restaurant_table_booking_system]> DELIMITER &&
MariaDB [restaurant_table_booking_system]> CREATE PROCEDURE get_reviews (IN rest_name VARCHAR(255))
-> BEGIN
-> SELECT r1.C_ID , r1.Restaurant_ID , r1.Review,r1.Date_Recorded , r2.Restaurant_Name
-> FROM Review r1 INNER JOIN Restaurant r2 ON r1.restaurant_id=r2.restaurant_id
-> WHERE Restaurant_Name=rest_name;
-> END &&
Query OK, 0 rows affected (0.019 sec)

MariaDB [restaurant_table_booking_system]> DELIMITER ;
MariaDB [restaurant_table_booking_system]>
```

Python code-

```
def procedure2(rname):
    c.callproc('get_reviews',[rname])
    df=pd.DataFrame()
    for result in c.stored_results():
        temp_df=pd.DataFrame(result.fetchall())
        df=df.append(temp_df)
    return df

def procedure():
    list_of_r = [i[0] for i in view_only_restaurant_names()]
    rname = st.selectbox("Restaurant name", list_of_r)
    if st.button("View reviews"):
        df=procedure2(rname)
        df.rename(columns =
{0:"C_ID",1:"Restaurant_ID",2:"Review",3:"Date_Recorded",4:"Res
taurant_Name"}, inplace = True)
        #df2=pd.DataFrame(df,columns=[ "C_ID" , "Restaurant_ID"
, "Review","Date_Recorded" , "Restaurant_Name"])
        with st.expander("View all reviews for
{}".format(rname)):
            st.dataframe(df)
```

Select the restaurant name for which you want to view the reviews

Procedure that filters reviews based on restaurant names:

Restaurant name

Taaza Thindi

Taaza Thindi

Pakshaala

Mainland China

Barbeque Nation

Jalsa

Onesta

Via Milano

Press the “View Reviews” button to see the reviews for that particular restaurant

Procedure that filters reviews based on restaurant names:

Restaurant name

Taaza Thindi

View reviews

View all reviews for Taaza Thindi

	C_ID	Restaurant_ID	Review
0	1	1	Amazing food at low cost!
1	3	1	Top class hygienic vegetarian restaurant in the heart of city, but we see the service i

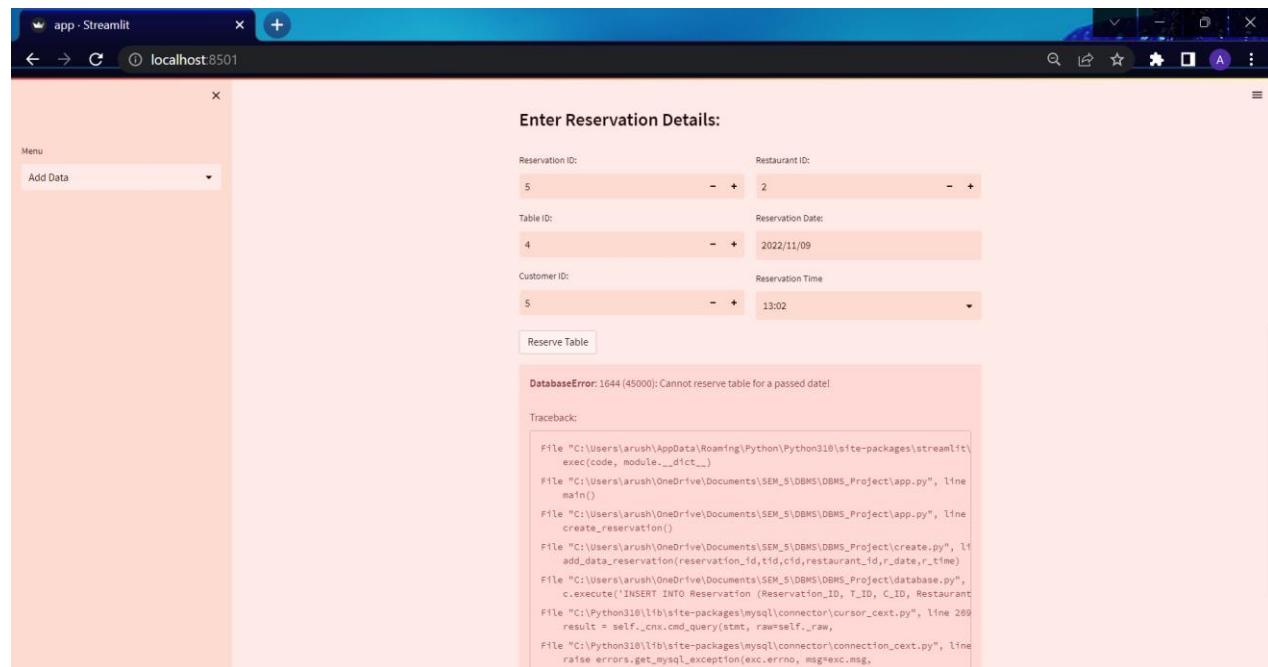
Triggers and Cursors

Create a Trigger and a Cursor. State the objective. Run and display the results.

Create a trigger to check if the reservation that is being made is not of a passed date.

```
Command Prompt - mysql -u root
MariaDB [restaurant_table_booking_system]> DELIMITER $$
MariaDB [restaurant_table_booking_system]> CREATE TRIGGER check_date
  -> BEFORE INSERT
  -> ON reservation FOR EACH ROW
  -> BEGIN
  -> DECLARE error_msg VARCHAR(255);
  -> SET error_msg = ('Cannot reserve table for a passed date!');
  -> IF (NEW.R_Date < CURDATE()) THEN
  -> SIGNAL SQLSTATE '45000'
  -> SET MESSAGE_TEXT = error_msg;
  -> END IF;
  -> END $$
Query OK, 0 rows affected (0.021 sec)
MariaDB [restaurant_table_booking_system]> DELIMITER ;
```

Trigger invoked when making reservation for a passed date



Trigger for checking if phone number is a 10 digit number

```
DELIMITER $$
CREATE TRIGGER check_phone
BEFORE INSERT
ON customer FOR EACH ROW
BEGIN
DECLARE error_msg VARCHAR(255);
SET error_msg = ('Please enter valid phone number!');
IF (NEW.Phone_Number < 1000000000) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = error_msg;
END IF;
END $$
DELIMITER ;
```

UI Screenshot-

The screenshot displays a web application interface for a restaurant table booking system. The browser's address bar shows 'localhost:8501'. The application has a sidebar menu with an 'Add Data' option. The main content area is titled 'PES1UG20CS077 - Restaurant Table Booking System'. It features a 'Choose Table' dropdown menu currently set to 'Customer'. Below this, there is a section titled 'Enter Customer Details:'. This section contains several input fields: 'Customer ID' with the value '8', 'Email ID' with 'ramkumar@gmail.com', 'First Name' with 'Ram', and 'Last Name' with 'Kumar'. The 'Phone number' field contains '9875594' and is accompanied by a validation error message that reads 'Value must be greater than or equal to 1000000000.' An 'Add Customer' button is positioned at the bottom of the form. The footer of the application indicates it was 'Made with Streamlit'.

Cursor created in database.py file to execute various SQL queries and fetch results.

```
import mysql.connector
import pandas as pd
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="restaurant_table_booking_system"
)
c = mydb.cursor()

def create_table():
    c.execute('CREATE TABLE IF NOT EXISTS Customer (C_ID int PRIMARY KEY,FirstName varchar(255),LastName varchar(255),Email_ID varchar(255))')
    #c.execute('Use restaurant_table_booking_system')

def add_data_cust(c_id, firstname, lastname, email, phone):
    c.execute('INSERT INTO Customer (C_ID, FirstName, LastName, Email_ID, Phone_number) VALUES (%s,%s,%s,%s,%s)',
              (c_id, firstname, lastname, email, phone))
    mydb.commit()

def add_data_rest(restaurant_id, rname, descrip, phone,street_name,city):
    c.execute('INSERT INTO Restaurant (Restaurant_ID, Restaurant_Name, Description, Phone_number, Street_name, City) VALUES (%s,%s,%s,%s,%s,%s)',
              (restaurant_id, rname, descrip, phone,street_name,city))
    mydb.commit()

def add_data_rev(c_id,restaurant_id,review,date):
    c.execute('INSERT INTO Review (C_ID, Restaurant_ID, Review , Date_Recorded) VALUES (%s,%s,%s,%s)',
              (c_id, restaurant_id, review, date))
    mydb.commit()
```

Ln 211, Col 76 (60 selected) — Spaces: 4 — UTF-8 — GBK — Python — @ 6

Developing a Frontend

The frontend should support

1. Addition, Modification and Deletion of records from any chosen table
2. There should be an window to accept and run any SQL statement and display the result

Addition Operation-

Adding a new customer

Menu
Add Data

PES1UG20CS077 - Restaurant Table Booking System

Choose Table
Customer

Enter Customer Details:

Customer ID: 8 Email ID: ramkumar@gmail.com

First Name: Ram Phone number: 9875594597

Last Name: Kumar

Add Customer

Successfully added Customer: Ram

Modification/Update

Menu
Edit Data

PES1UG20CS077 - Restaurant Table Booking System

Choose Table
Customer

Edit Customer Details:

Current customers

	C_ID	FirstName	LastName	EmailID	Phone_number
0	1	Arushi	Pandey	arushigri@gmail.com	7349750009
1	2	Anushka	Jalori	anushkajalori@gmail.com	9845183703
2	3	Shreya	Nair	shreyanair@yahoo.com	9686337517
3	4	Krishna	Gupta	krishnagupta@pesu.pes.edu	7019210710
4	5	Annika	Mittal	reachannika@gmail.com	9591785522
5	6	Sanjeev	Kapoor	sanjeev@gmail.com	9879695934
6	7	Ajay	Chopra	ajay@yahoo.com	9845183703
7	8	Ram	Kumar	ramkumar@gmail.com	9875594597

Customer ID to Edit
8

Choose customer ID to edit customer details

The screenshot shows a Streamlit web application running on localhost:8501. On the left, there is a sidebar menu with a button labeled "Edit Data". The main area displays a table of customer information:

2	3	Shreya	Nair	shreyanair@yahoo.com	9686337517
3	4	Krishna	Gupta	krishnagupta@pesu.pes.edu	7019210710
4	5	Annika	Mittal	reachannika@gmail.com	9591785522
5	6	Sanjeev	Kapoor	sanjeev@gmail.com	9879695934
6	7	Ajay	Chopra	ajay@yahoo.com	9845183703
7	8	Ram	Kumar	ramkumar@gmail.com	9875594597

Below the table, there is a section titled "Customer ID to Edit" with a dropdown menu showing the value "8". Below the dropdown is a list of numbers 1 through 7. At the bottom of this section is a button labeled "Update Customer".

Press Update button to update the customer

The screenshot shows the same Streamlit web application after the "Update Customer" button was pressed. The "Customer ID to Edit" dropdown still shows "8". Below it, the details for customer ID 8 are displayed:

C_ID: 8 email: reachram@gmail.com

First Name: Ram phone: 9875594597

lastname: Kumar

Below these details is a button labeled "Update Customer". A green message box below the button says "Successfully updated: Ram to :-Ram".

At the bottom, there is a section titled "Updated data" with a table showing the updated customer data:

	C_ID	FirstName	LastName	EmailID	Phone_number
0	1	Arushi	Pandey	arushig88@gmail.com	7349750009
1	2	Anushka	Jalori	anshkajalori@gmail.com	9845183703
2	3	Shreya	Nair	shreyanair@yahoo.com	9686337517
3	4	Krishna	Gupta	krishnagupta@pesu.pes.edu	7019210710
4	5	Annika	Mittal	reachannika@gmail.com	9591785522
5	6	Sanjeev	Kapoor	sanjeev@gmail.com	9879695934
6	7	Ajay	Chopra	ajay@yahoo.com	9845183703
7	8	Ram	Kumar	reachram@gmail.com	9875594597

Similarly, adding data in reservation table-

Menu

Add Data

Find the restaurants who have vacant tables and are present in Koramangala:

View Restaurants Details

Enter Reservation Details:

Reservation ID:

2

Restaurant ID:

3

Table ID:

5

Reservation Date:

2022/12/03

Customer ID:

4

Reservation Time:

11:46

Reserve Table

Successfully added Reservation: 2

View Tables-

Menu

View Data

PES1UG20CS077 - Restaurant Table Booking System

View Data:

View all Customers Details

	C_ID	FirstName	LastName	EmailID	Phone_number
0	1	Arushi	Pandey	arushigr88@gmail.com	7349750009
1	2	Anushka	Jalori	anshkajalori@gmail.com	9845183703
2	3	Shreya	Nair	shreyanair@yahoo.com	9686337517
3	4	Krishna	Gupta	krishnagupta@pesu.pes.edu	7019210710
4	5	Annika	Mittal	reachannika@gmail.com	9591785522
5	6	Sanjeev	Kapoor	sanjeev@gmail.com	9879695934
6	7	Ajay	Chopra	ajay@yahoo.com	9845183703

View all Restaurants Details

View all Review Details

View all restaurant table Details

Menu

View Data

View all Restaurants Details

	Restaurant_ID	Restaurant_Name	Description	Phone_number	Street_name	City
0	1	Taaza Thindi	South Indian	9880704650	Banshankari	Bengaluru
1	2	Pakshala	Multi Cuisine	8904809459	Jayanagar	Bengaluru
2	3	Mainland China	Chinese	7892279326	Koramangala	Bengaluru
3	4	Barbeque Nation	Buffet	9557522075	Koramangala	Bengaluru
4	5	Jalsa	Multi Cuisine	8971351907	Banshankari	Bengaluru
5	6	Onesta	Italian	6362283640	Indiranagar	Bengaluru
6	7	Via Milano	Italian	8979695979	Koramangala	Bengaluru

View all restaurant table Details

View all Reservation Details

View all Review Details

	C_ID	Restaurant_ID	Review
0	1	1	Amazing food at low cost!
1	1	4	I'm a regular visitor to Barbeque Nation. Across outlets in Bangalore. The vegetarian
2	2	3	The best chinese i have ever had! Panner machurian is a must have
3	2	6	Ordered pizza,. Crust was thin,. Hygiene was good too,. Delivery was too fast,. Over
4	3	1	Top class hygienic vegetarian restaurant in the heart of city, but we see the service i
5	3	3	Wait time was too much,. Food was also not that great
6	4	6	Unlimited pizza offers 50% of overall in the bill. Service was excellent

Menu

View Data

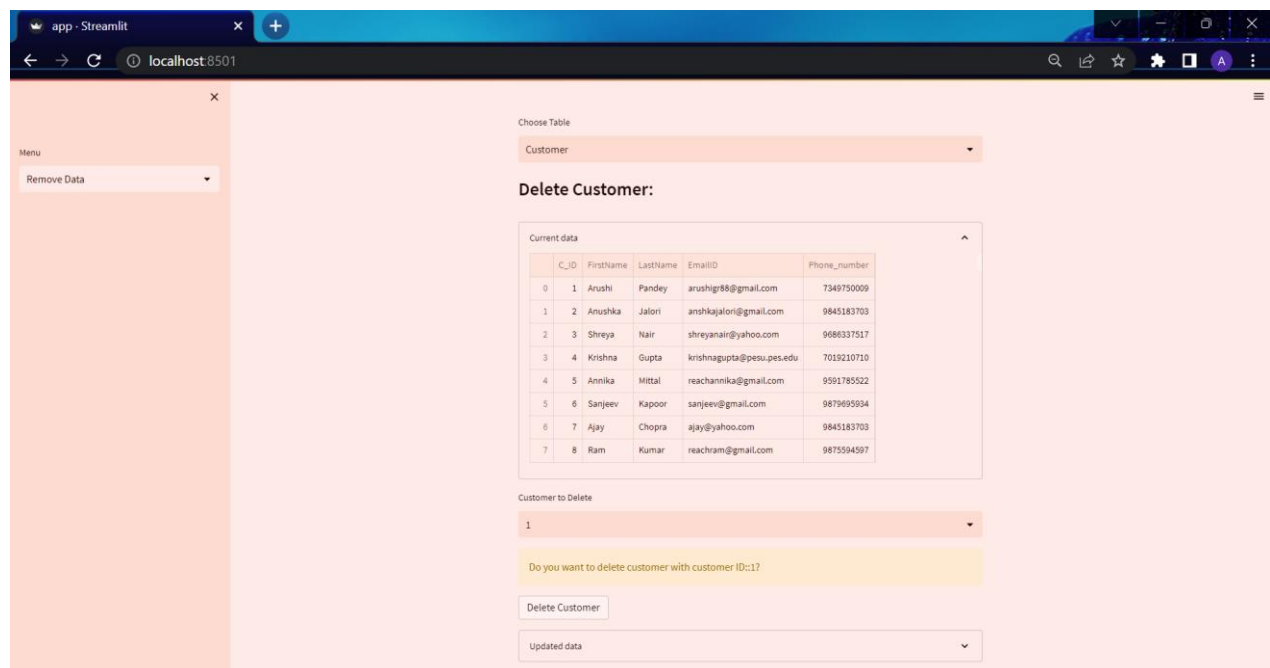
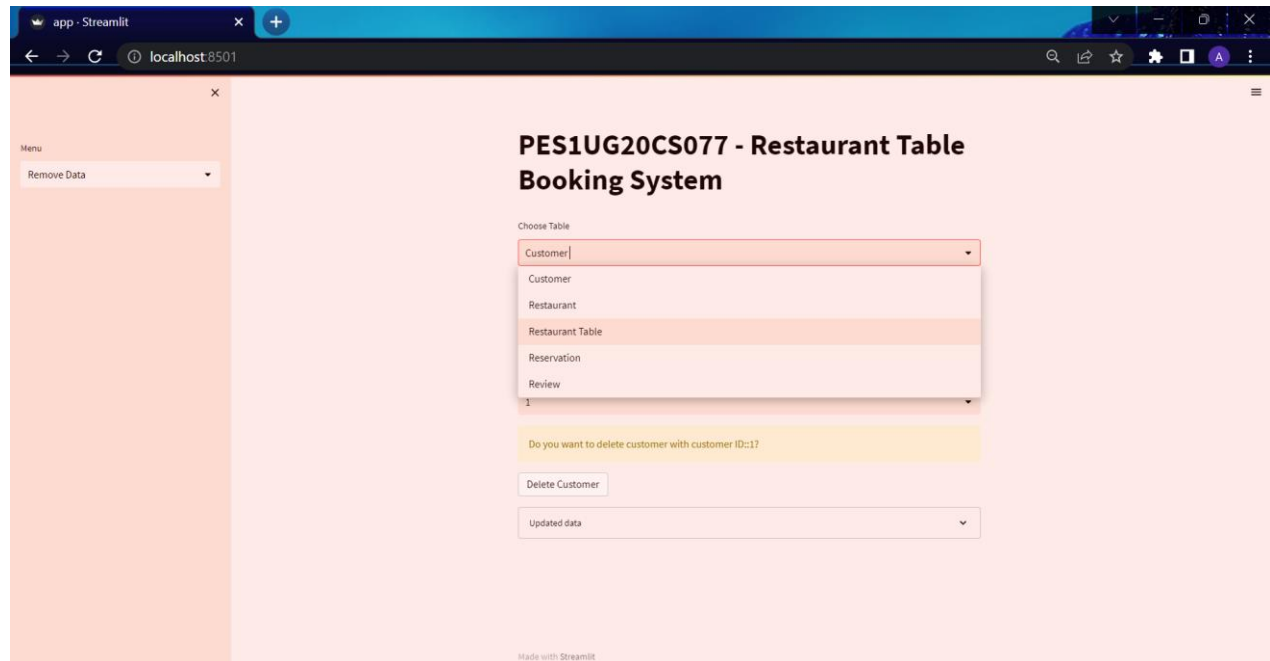
View all restaurant table Details

	T_ID	Restaurant_ID	Capacity	Vacancy	Reserved
0	1	1	4	Yes	No
1	2	1	2	No	Yes
2	3	2	4	Yes	No
3	4	2	2	Yes	No
4	5	3	6	No	Yes
5	6	3	2	Yes	No
6	7	4	4	No	Yes
7	8	4	4	Yes	No
8	9	5	8	Yes	No
9	10	5	4	No	No

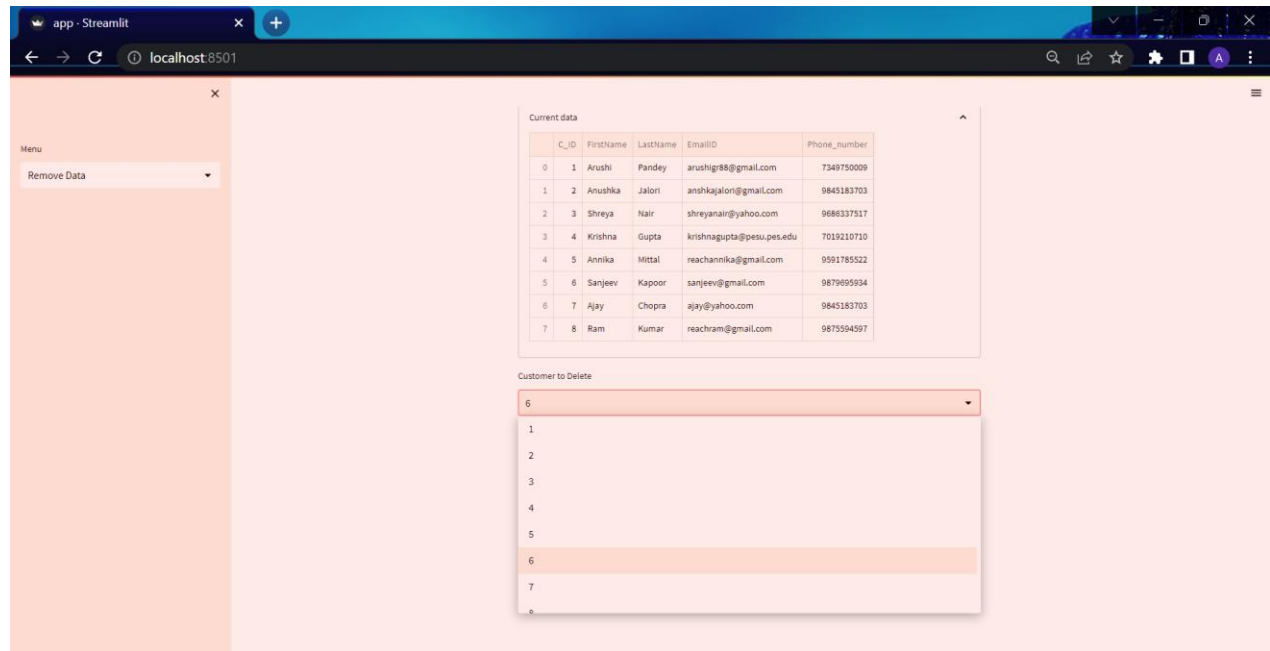
View all Reservation Details

	Reservation_ID	T_ID	C_ID	Restaurant_ID	R_Date	R_Time
0	1	2	1	1	2022-11-25	20:15:00
1	2	5	4	3	2022-12-03	11:46:00
2	3	7	5	4	2022-12-03	11:48:00

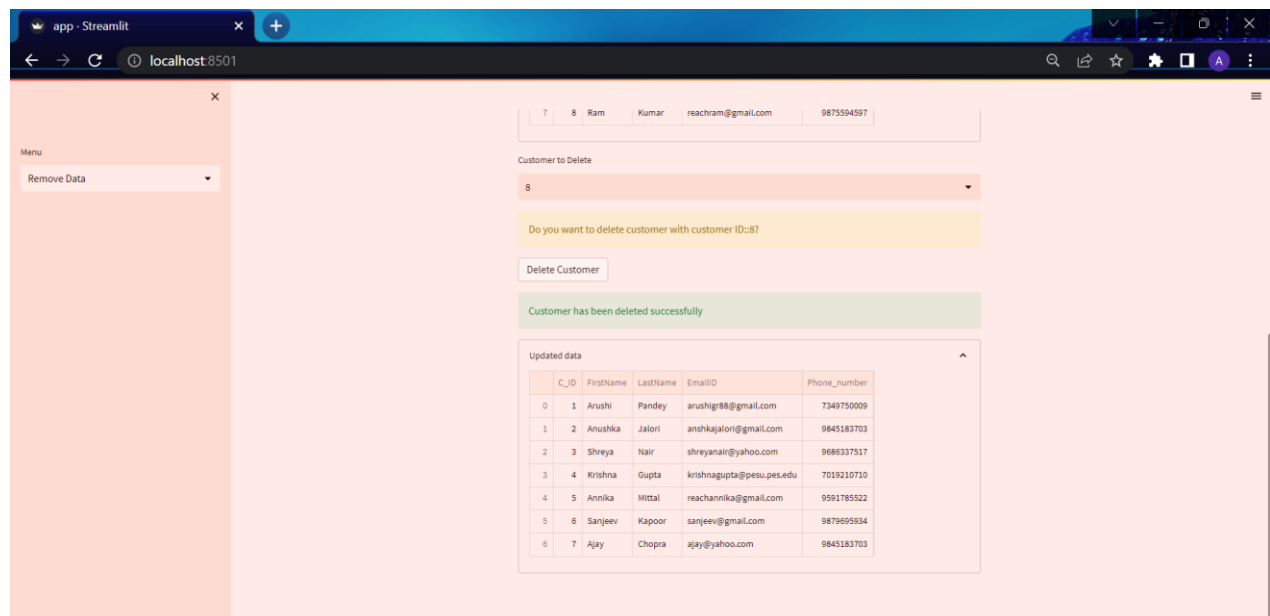
Delete-
Select table to delete from-



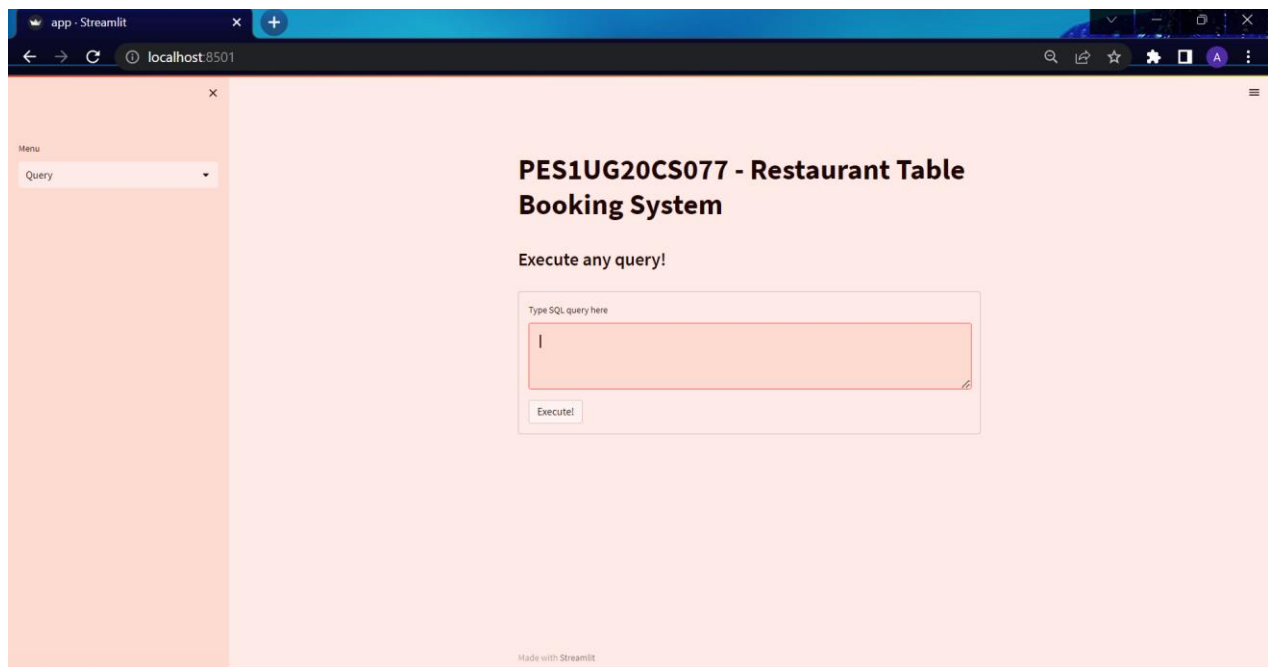
Select customer ID to delete-



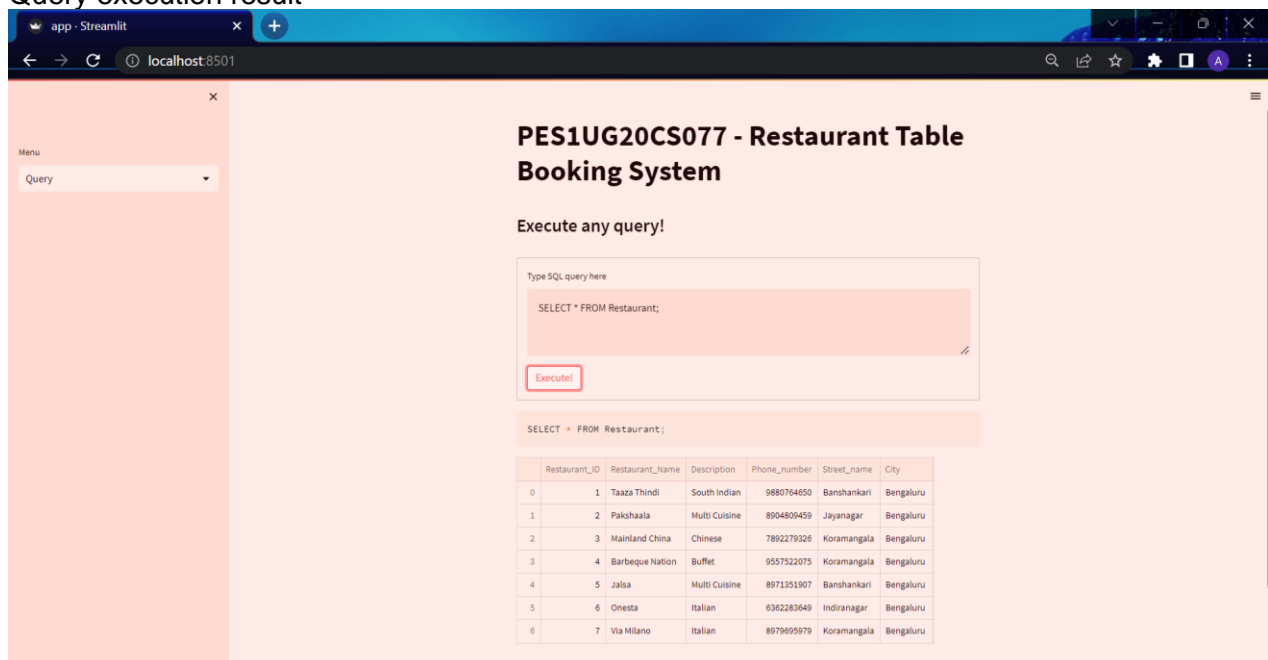
Click on delete button to delete the customer



Query in UI



Query execution result



CRUD operations have been implemented for all 5 tables.

Also we can view data for all 5 tables.

Join,procedure, trigger, set operations and aggregate operations have also been implemented in frontend.