**Project Report On**

**City Taxi Call Center Database Management**

**System**



## Academic Year 2023-24

**Under the guidance of**

**Prof. Sachin Bhandari**

**Submitted By**

**Madhav Gupta E212 Om Amit Mishra E216 Yash Taneja E224**

Case Study: City Taxi Call Center Database Management System

# Problem Statement:

In the bustling urban environment of CityX, a taxi call center is striving to enhance its operations by implementing a Database Management System (DBMS). This beginner-level project aims to streamline the management of taxi bookings, drivers, and customer data.

# Background:

City Taxi Call Center currently relies on manual record-keeping and traditional filing systems, leading to inefficiencies, data inaccuracies, and delayed response times. To address these challenges, the decision was made to implement a DBMS.

# Objective:

The primary goal of this project is to design and implement a DBMS that can efficiently manage taxi bookings, driver information, and customer details. The system should ensure quick retrieval of data, easy updates, and enhanced overall performance.

# Motivation for Developing the Project:

We'd like to share the journey that led us to conceptualize and work on the City Taxi Call Center Database Management System project. It all started with a series of observations and experiences that highlighted significant challenges faced by both the taxi company and its customers.

Firstly, we noticed the frustration among customers due to long wait times when calling for a taxi. Many times, incorrect information or delays in dispatching contributed to this dissatisfaction. Similarly, the taxi drivers

often struggled with inefficient allocation of rides and lacked real-time updates on customer locations and demand patterns.

Driven by these real-world challenges, our team felt compelled to find a solution that would not only address the immediate issues but also lay the groundwork for a more efficient and customer-centric taxi service.

However, the journey wasn't without its hurdles. We encountered various challenges along the way, ranging from understanding the complexities of the taxi industry to designing a scalable and user-friendly database management system.

One of the major obstacles we faced was integrating real-time data updates into the system. Ensuring that both the call center operators and drivers had access to accurate and up-to-date information posed a significant technical challenge. Additionally, designing a user interface that was intuitive for both the call center staff and the drivers required careful consideration of usability principles and feedback from stakeholders.

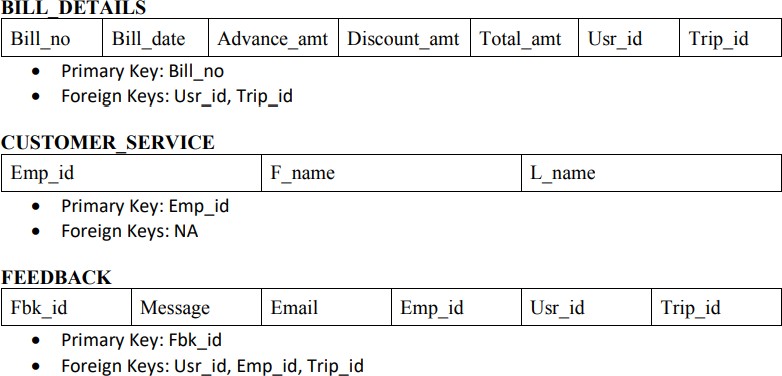
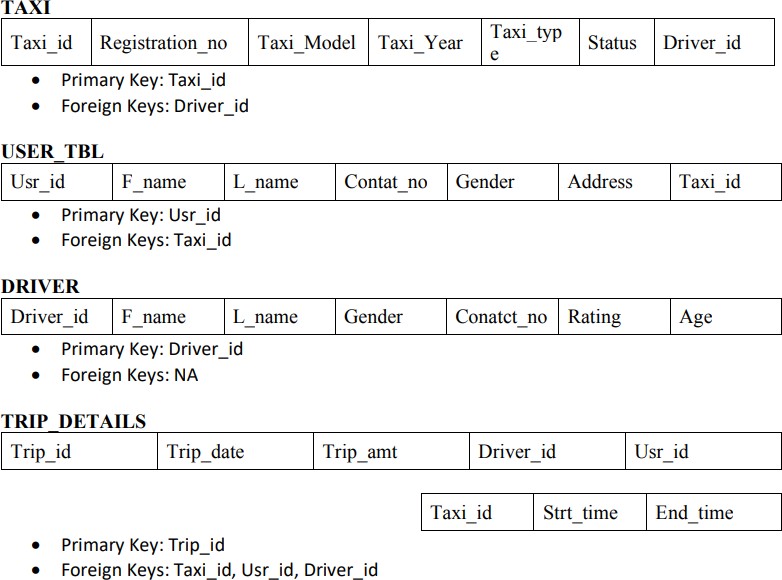
Moreover, navigating the regulatory landscape and ensuring compliance with data protection laws added another layer of complexity to the project. We had to carefully balance the need for data security and privacy with the requirements for seamless information sharing and communication within the system.

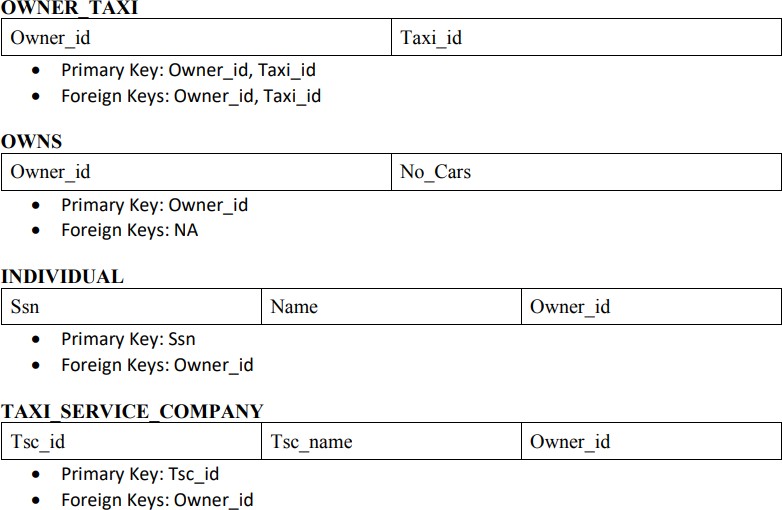
Despite these challenges, working on this project has been an invaluable learning experience for our team. We've gained a deeper understanding of database management principles, software development methodologies, and the nuances of the transportation industry.

In conclusion, our motivation for developing the City Taxi Call Center Database Management System stemmed from a genuine desire to improve the taxi service experience for both customers and drivers. Through perseverance, teamwork, and a commitment to excellence, we overcame

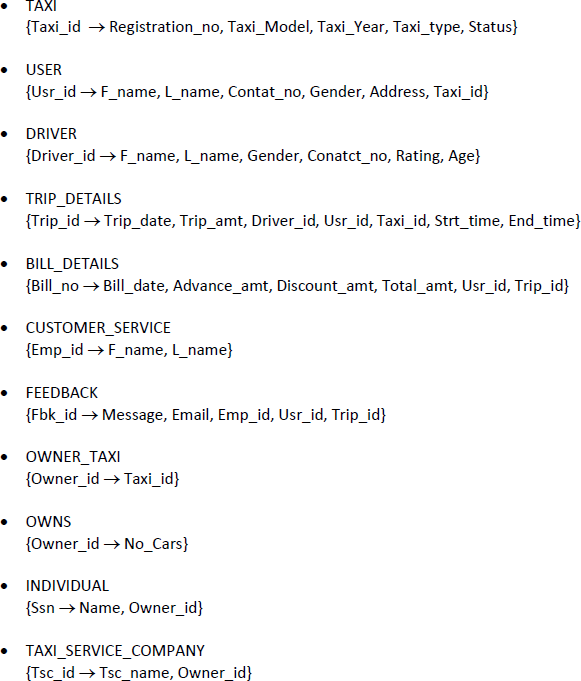
challenges and gained valuable insights that will undoubtedly shape our future endeavours in the field of software development.

# ER Diagram:

**Schema:**



# Normalization:



**CREATE And INSERT Queries:**

import sqlite3

db\_file = 'example.db'

conn = sqlite3.connect(db\_file) cursor = conn.cursor()

sql\_script = """

PRAGMA foreign\_keys = ON;

CREATE TABLE IF NOT EXISTS TAXI ( Taxi\_id INTEGER PRIMARY KEY,

Registration\_no TEXT UNIQUE, Model TEXT,

Taxi\_Year DATE, Taxi\_type TEXT, Taxi\_status TEXT, Driver\_id INTEGER,

FOREIGN KEY (Driver\_id) REFERENCES DRIVER(Driver\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS USER\_TBL ( Usr\_id INTEGER PRIMARY KEY,

F\_name TEXT, L\_name TEXT, Contat\_no INTEGER, Gender TEXT, USR\_Address TEXT, Taxi\_id INTEGER,

FOREIGN KEY (Taxi\_id) REFERENCES TAXI(Taxi\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS DRIVER ( Driver\_id INTEGER PRIMARY KEY,

F\_name TEXT, L\_name TEXT, Gender TEXT, Contact\_no TEXT, Rating INTEGER, Age INTEGER

);

CREATE TABLE IF NOT EXISTS TRIP\_DETAILS ( Trip\_id INTEGER PRIMARY KEY,

Trip\_date DATE, Trip\_amt DECIMAL(10,2),

Driver\_id INTEGER NOT NULL, Usr\_id INTEGER NOT NULL,

ACTION,

);

Taxi\_id INTEGER NOT NULL,

Strt\_time DATETIME, End\_time DATETIME, T\_Source VARCHAR(20), T\_Destination VARCHAR(20),

FOREIGN KEY (Driver\_id) REFERENCES DRIVER(Driver\_id) ON DELETE NO

FOREIGN KEY (Usr\_id) REFERENCES USER\_TBL(Usr\_id) ON DELETE NO ACTION, FOREIGN KEY (Taxi\_id) REFERENCES TAXI(Taxi\_id) ON DELETE NO ACTION

CREATE TABLE IF NOT EXISTS BILL\_DETAILS ( Bill\_no INTEGER PRIMARY KEY,

Bill\_date DATE, Advance\_amt DECIMAL(10,2), Discount\_amt DECIMAL(10,2), Total\_amt DECIMAL(10,2), Usr\_id INTEGER,

Trip\_id INTEGER,

FOREIGN KEY (Usr\_id) REFERENCES USER\_TBL(Usr\_id) ON DELETE CASCADE, FOREIGN KEY (Trip\_id) REFERENCES TRIP\_DETAILS(Trip\_id) ON DELETE NO

ACTION

);

CREATE TABLE IF NOT EXISTS CUSTOMER\_SERVICE ( Emp\_id INTEGER PRIMARY KEY,

F\_name TEXT, L\_name TEXT

);

CREATE TABLE IF NOT EXISTS FEEDBACK ( Fbk\_id INTEGER PRIMARY KEY,

Message TEXT, Email TEXT, Emp\_id INTEGER, Usr\_id INTEGER, Trip\_id INTEGER, F\_Type TEXT,

FOREIGN KEY (Emp\_id) REFERENCES CUSTOMER\_SERVICE(Emp\_id) ON DELETE CASCADE,

FOREIGN KEY (Usr\_id) REFERENCES USER\_TBL(Usr\_id) ON DELETE NO ACTION, FOREIGN KEY (Trip\_id) REFERENCES TRIP\_DETAILS(Trip\_id) ON DELETE

CASCADE

);

CREATE TABLE IF NOT EXISTS OWNS ( Owner\_id INTEGER PRIMARY KEY,

No\_Cars INTEGER

);

CREATE TABLE IF NOT EXISTS OWNER\_TAXI (

Owner\_id INTEGER, Taxi\_id INTEGER,

PRIMARY KEY (Owner\_id, Taxi\_id),

FOREIGN KEY (Owner\_id) REFERENCES OWNS(Owner\_id) ON DELETE CASCADE, FOREIGN KEY (Taxi\_id) REFERENCES TAXI(Taxi\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS INDIVIDUAL ( Ssn INTEGER PRIMARY KEY,

Name TEXT, Owner\_id INTEGER,

FOREIGN KEY (Owner\_id) REFERENCES OWNS(Owner\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS TAXI\_SERVICE\_COMPANY ( Tsc\_id INTEGER PRIMARY KEY,

Tsc\_name TEXT, Owner\_id INTEGER,

FOREIGN KEY (Owner\_id) REFERENCES OWNS(Owner\_id) ON DELETE CASCADE

);

CREATE TABLE TMP\_TRIP\_DETAILS AS SELECT \* FROM TRIP\_DETAILS;

DROP TABLE TRIP\_DETAILS;

CREATE TABLE TRIP\_DETAILS ( Trip\_id INTEGER PRIMARY KEY,

Trip\_date DATE, Trip\_amt DECIMAL(10,2),

Driver\_id INTEGER NOT NULL, Usr\_id INTEGER NOT NULL, Taxi\_id INTEGER NOT NULL,

Strt\_time DATETIME, End\_time DATETIME, T\_Source VARCHAR(20), T\_Destination VARCHAR(20),

FOREIGN KEY (Driver\_id) REFERENCES DRIVER(Driver\_id) ON DELETE NO

ACTION,

);

FOREIGN KEY (Usr\_id) REFERENCES USER\_TBL(Usr\_id) ON DELETE NO ACTION, FOREIGN KEY (Taxi\_id) REFERENCES TAXI(Taxi\_id) ON DELETE NO ACTION

INSERT INTO TRIP\_DETAILS SELECT \* FROM TMP\_TRIP\_DETAILS;

DROP TABLE TMP\_TRIP\_DETAILS;

"""

print("Table created successfully") print("Inserting Data into Tables")

*# TAXI Insertion*

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (1, 'GJ152004', 'BENZE', '2024-03-11', 'SUV', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (2, 'MH432010', 'MARUTI', '2024-03-12', 'Hatchback', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (3, 'DL678003', 'HYUNDAI', '2024-03-13', 'Sedan', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (4, 'KA987007', 'TATA', '2024-03-14', 'SUV', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (5, 'UP765005', 'MAHINDRA', '2024-03-15', 'SUV', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (6, 'TN876006', 'TOYOTA', '2024-03-16', 'Sedan', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (7, 'AP543001', 'HONDA', '2024-03-17', 'Hatchback', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (8, 'PB234008', 'FORD', '2024-03-18', 'SUV', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (9, 'HR654009', 'VOLKSWAGEN', '2024-03-19', 'Hatchback', 'Available', NULL)")

cursor.execute("INSERT OR IGNORE INTO TAXI (Taxi\_id, Registration\_no, Model, Taxi\_Year, Taxi\_type, Taxi\_status, Driver\_id) VALUES (10, 'MP876002', 'CHEVROLET', '2024-03-20', 'SUV', 'Available', NULL)")

*# USER\_TBL Insertion*

cursor.execute("INSERT OR IGNORE INTO USER\_TBL VALUES(1, 'Yash', 'Taneja', 7987845578, 'Male', 'Chhindwara', 1)")

cursor.execute("INSERT OR IGNORE INTO USER\_TBL VALUES(2, 'Riya', 'Sharma', 9090909090, 'Female', 'Delhi', 2)")

cursor.execute("INSERT OR IGNORE INTO USER\_TBL VALUES(3, 'Amit', 'Singh', 9876543210, 'Male', 'Mumbai', 3)")

cursor.execute("INSERT OR IGNORE INTO USER\_TBL VALUES(4, 'Priya', 'Patel', 8765432109, 'Female', 'Ahmedabad', 4)")

cursor.execute("INSERT OR IGNORE INTO USER\_TBL VALUES(5, 'Kunal', 'Gupta', 7654321098, 'Male', 'Bangalore', 5)")

*# TRIP\_DETAILS Insertion \*/*

cursor.execute("INSERT OR IGNORE INTO TRIP\_DETAILS VALUES(1, '2024-03-11', 150.45, 1, 1, 1, '2024-03-11 12:00:00', '2024-03-11 13:05:00', 'Chhindwara',

'Sendhwa')")

cursor.execute("INSERT OR IGNORE INTO TRIP\_DETAILS VALUES(2, '2024-03-13', 175.50, 2, 2, 2, '2024-03-13 12:00:00', '2024-03-13 13:05:00', 'Delhi',

'Mumbai')")

cursor.execute("INSERT OR IGNORE INTO TRIP\_DETAILS VALUES(3, '2024-03-14', 200.00, 3, 3, 3, '2024-03-14 12:00:00', '2024-03-14 13:05:00', 'Kolkata',

'Chennai')")

cursor.execute("INSERT OR IGNORE INTO TRIP\_DETAILS VALUES(4, '2024-03-15', 120.75, 4, 4, 4, '2024-03-15 12:00:00', '2024-03-15 13:05:00', 'Bangalore',

'Hyderabad')")

cursor.execute("INSERT OR IGNORE INTO TRIP\_DETAILS VALUES(5, '2024-03-16', 95.25, 5, 5, 5, '2024-03-16 12:00:00', '2024-03-16 13:05:00', 'Pune',

'Ahmedabad')")

*# BILL\_DETAILS Insertion \*/*

cursor.execute("INSERT OR IGNORE INTO BILL\_DETAILS VALUES(1, '2024-03-11', 100, 0, 50.45, 1, 1)")

cursor.execute("INSERT OR IGNORE INTO BILL\_DETAILS VALUES(2, '2024-03-13', 100, 0, 75.50, 2, 2)")

cursor.execute("INSERT OR IGNORE INTO BILL\_DETAILS VALUES(3, '2024-03-14', 100, 0, 200.00, 3, 3)")

cursor.execute("INSERT OR IGNORE INTO BILL\_DETAILS VALUES(4, '2024-03-15', 100, 0, 20.75, 4, 4)")

cursor.execute("INSERT OR IGNORE INTO BILL\_DETAILS VALUES(5, '2024-03-16', 0, 0, 95.25, 5, 5)")

*# Customer\_Service Insertion \*/*

cursor.execute("INSERT OR IGNORE INTO CUSTOMER\_SERVICE VALUES(1, 'Madhav',

'Patel')")

cursor.execute("INSERT OR IGNORE INTO CUSTOMER\_SERVICE VALUES(2, 'Aarav',

'Shah')")

cursor.execute("INSERT OR IGNORE INTO CUSTOMER\_SERVICE VALUES(3, 'Diya',

'Verma')")

cursor.execute("INSERT OR IGNORE INTO CUSTOMER\_SERVICE VALUES(4, 'Rohan',

'Singh')")

cursor.execute("INSERT OR IGNORE INTO CUSTOMER\_SERVICE VALUES(5, 'Neha',

'Gupta')")

*# FEEDBACK Insertion \*/*

cursor.execute("INSERT OR IGNORE INTO FEEDBACK VALUES(1, 'so good', 'abhi@gmail.com', 1, 1, 1, 'good')")

cursor.execute("INSERT OR IGNORE INTO FEEDBACK VALUES(2, 'bad experience', 'john@example.com', 2, 2, 2, 'bad')")

cursor.execute("INSERT OR IGNORE INTO FEEDBACK VALUES(3, 'could be better', 'jane@example.com', 3, 3, 3, 'bad')")

cursor.execute("INSERT OR IGNORE INTO FEEDBACK VALUES(4, 'excellent service', 'smith@example.com', 4, 4, 4, 'good')")

cursor.execute("INSERT OR IGNORE INTO FEEDBACK VALUES(5, 'friendly staff', 'emma@example.com', 5, 5, 5, 'good')")

*# owns Insertion \*/*

cursor.execute("INSERT OR IGNORE INTO OWNS VALUES(1, 1)") cursor.execute("INSERT OR IGNORE INTO OWNS VALUES(2, 2)") cursor.execute("INSERT OR IGNORE INTO OWNS VALUES(3, 3)") cursor.execute("INSERT OR IGNORE INTO OWNS VALUES(4, 4)") cursor.execute("INSERT OR IGNORE INTO OWNS VALUES(5, 5)")

*# OWNER\_TAXI Insertion \*/*

cursor.execute("INSERT OR IGNORE INTO OWNER\_TAXI VALUES (1, 1)") cursor.execute("INSERT OR IGNORE INTO OWNER\_TAXI VALUES (2, 2)") cursor.execute("INSERT OR IGNORE INTO OWNER\_TAXI VALUES (3, 3)") cursor.execute("INSERT OR IGNORE INTO OWNER\_TAXI VALUES (4, 4)") cursor.execute("INSERT OR IGNORE INTO OWNER\_TAXI VALUES (5, 5)")

*# INDIVIDUAL Insertion*

cursor.execute("INSERT OR IGNORE INTO INDIVIDUAL VALUES (123, 'abhi owner ind', 1)")

cursor.execute("INSERT OR IGNORE INTO INDIVIDUAL VALUES (456, 'john owner ind', 2)")

cursor.execute("INSERT OR IGNORE INTO INDIVIDUAL VALUES (789, 'harsh owner ind', 3)")

cursor.execute("INSERT OR IGNORE INTO INDIVIDUAL VALUES (101112, 'shan

owner ind', 4)")

cursor.execute("INSERT OR IGNORE INTO INDIVIDUAL VALUES (131415, 'ramesh

owner ind', 5)")

*# TAXI\_SERVICE\_COMPANY Insertion*

cursor.execute("INSERT OR IGNORE INTO TAXI\_SERVICE\_COMPANY VALUES (1,

'abhi taxi comp', 2)")

cursor.execute("INSERT OR IGNORE INTO TAXI\_SERVICE\_COMPANY VALUES (2, 'Raj

Taxi Services', 3)")

cursor.execute("INSERT OR IGNORE INTO TAXI\_SERVICE\_COMPANY VALUES (3,

'Mumbai Cabs Pvt Ltd', 4)")

cursor.execute("INSERT OR IGNORE INTO TAXI\_SERVICE\_COMPANY VALUES (4,

'Delhi Taxi Solutions', 5)")

cursor.execute("INSERT OR IGNORE INTO TAXI\_SERVICE\_COMPANY VALUES (5,

'Bangalore Taxi Corp', 5)")

print("All data inserted successfully")

/\* Agrregate Functions \*/

/\* Calculate the total number of taxi models available in the system. \*/ SELECT COUNT(DISTINCT Model) AS Total\_Taxi\_Models FROM TAXI;

/\* Calculate the total number of trips made by a specific driver \*/

SELECT DRIVER.Driver\_id, SUM(CASE WHEN TRIP\_DETAILS.Trip\_id IS NOT NULL THEN 1 ELSE 0

END) AS Total\_Trips FROM DRIVER

LEFT JOIN TRIP\_DETAILS ON DRIVER.Driver\_id = TRIP\_DETAILS.Driver\_id

GROUP BY DRIVER.Driver\_id;

/\* Calculate the average age of drivers in the system. \*/ SELECT AVG(Age) AS Average\_Driver\_Age FROM DRIVER;

/\* Find the minimum trip amount recorded in the system. \*/ SELECT MIN(Trip\_amt) AS Min\_Trip\_Amount FROM TRIP\_DETAILS;

/\* Find the maximum rating achieved by any driver in the system. \*/ SELECT MAX(Rating) AS Max\_Driver\_Rating FROM DRIVER;

/\* Calculate the total number of trips made by each taxi type. \*/

SELECT Taxi\_type, COUNT(Trip\_id) AS Total\_Trips FROM TAXI INNER JOIN TRIP\_DETAILS ON TAXI.Taxi\_id = TRIP\_DETAILS.Taxi\_id GROUP BY Taxi\_type;

/\* Find the taxi types with more than 10 trips recorded. \*/

SELECT Taxi\_type, COUNT(Trip\_id) AS Total\_Trips FROM TAXI INNER JOIN TRIP\_DETAILS

ON TAXI.Taxi\_id = TRIP\_DETAILS.Taxi\_id GROUP BY Taxi\_type HAVING COUNT(Trip\_id) > 10;

/\* List all unique taxi models available in the system. \*/ SELECT DISTINCT Model FROM TAXI;

/\* JOINS \*/

/\* Inner Join The taxi service company wants to retrieve a list of all trips along with their corresponding driver and user details. \*/

SELECT TD.Trip\_id, TD.Trip\_date, TD.Strt\_time, TD.End\_time, D.F\_name AS Driver\_FirstName, D.L\_name AS Driver\_LastName,

D.Gender AS Driver\_Gender, D.Conatct\_no AS Driver\_Contact,

U.F\_name AS User\_FirstName, U.L\_name AS User\_LastName, U.Gender AS User\_Gender, U.Contat\_no AS User\_Contact

FROM TRIP\_DETAILS TD

INNER JOIN DRIVER D ON TD.Driver\_id = D.Driver\_id INNER JOIN USER\_TBL U ON TD.Usr\_id = U.Usr\_id;

/\* Left Join The taxi service company wants to list all taxis along with their associated driver information.

Some taxis may not have assigned drivers yet. \*/

SELECT T.Taxi\_id, T.Registration\_no, T.Model, T.Taxi\_Year, T.Taxi\_type, T.Taxi\_status, D.F\_name AS Driver\_FirstName,

D.L\_name AS Driver\_LastName, D.Gender AS Driver\_Gender, D.Conatct\_no AS Driver\_Contact FROM TAXI T

LEFT JOIN DRIVER D ON T.Driver\_id = D.Driver\_id;

/\* Right Join The taxi service company wants to see a list of all drivers along with the details of the

trips they have completed, even if some drivers haven't completed any trips yet. \*/ SELECT D.Driver\_id, D.F\_name, D.L\_name, D.Gender, D.Conatct\_no, TD.Trip\_id, TD.Trip\_date, TD.Strt\_time, TD.End\_time

FROM DRIVER D

RIGHT JOIN TRIP\_DETAILS TD ON D.Driver\_id = TD.Driver\_id;

/\* Full Outer Join he taxi service company wants to generate a comprehensive report that lists

all taxis, drivers, and their associated trip details, regardless of whether each taxi has been

assigned a driver or has completed any trips yet. \*/

SELECT T.Taxi\_id, T.Registration\_no, Model, T.Taxi\_Year, T.Taxi\_type, T.Taxi\_status, D.Driver\_id, D.F\_name AS Driver\_FirstName, D.L\_name AS Driver\_LastName, D.Gender AS Driver\_Gender, D.Conatct\_no AS Driver\_Contact,

TD.Trip\_id, TD.Trip\_date, TD.Strt\_time, TD.End\_time FROM TAXI T

FULL OUTER JOIN DRIVER D ON T.Driver\_id = D.Driver\_id

FULL OUTER JOIN TRIP\_DETAILS TD ON T.Taxi\_id = TD.Taxi\_id;

/\* Set Operations \*/

--Query to Combine All Unique Driver Names from Two Tables:

SELECT F\_name AS Entity\_Name FROM DRIVER UNION

SELECT F\_name AS Entity\_Name FROM CUSTOMER\_SERVICE;

--Query to Find All Taxi Models Owned by Individual Owners but Not Owned by Taxi Service Companies:

SELECT Name AS Entity\_Name FROM INDIVIDUAL EXCEPT

SELECT Tsc\_name AS Entity\_Name FROM TAXI\_SERVICE\_COMPANY;

--It finds the intersection of taxi IDs owned by individual owners and taxi IDs in the general taxi pool.

SELECT Taxi\_id FROM TAXI INTERSECT

SELECT Taxi\_id FROM OWNER\_TAXI;

--Retrieve a list of all unique taxi IDs, including those owned by individual owners and

--those managed by the taxi service company, ensuring that each taxi ID is listed only once.

SELECT Taxi\_id FROM TAXI UNION

SELECT Taxi\_id FROM OWNER\_TAXI;

--Group by and having clause

--The taxi service company wants to generate a report showing the total number of trips made by

--each taxi, along with the average trip amount for taxis that have completed more than 10 trips.

SELECT T.Taxi\_id, COUNT(TD.Trip\_id) AS Total\_Trips, AVG(TD.Trip\_amt) AS Average\_Trip\_Amount

FROM TAXI T

LEFT JOIN TRIP\_DETAILS TD ON T.Taxi\_id = TD.Taxi\_id

GROUP BY T.Taxi\_id

HAVING COUNT(TD.Trip\_id) > 10;

--The taxi service company wants to find out the total number of trips completed by each driver,

--along with their average rating.

SELECT D.Driver\_id, D.F\_name, D.L\_name, COUNT(TD.Trip\_id) AS Total\_Trips, AVG(D.Rating) AS Average\_Rating

FROM DRIVER D

LEFT JOIN TRIP\_DETAILS TD ON D.Driver\_id = TD.Driver\_id GROUP BY D.Driver\_id, D.F\_name, D.L\_name;

--The taxi service company wants to identify users who have taken more than 5 trips and have spent a total amount

--greater than $500 on trips.

SELECT U.Usr\_id, U.F\_name, U.L\_name, COUNT(TD.Trip\_id) AS Total\_Trips, SUM(BD.Total\_amt) AS Total\_Amount\_Spent

FROM USER\_TBL U

LEFT JOIN TRIP\_DETAILS TD ON U.Usr\_id = TD.Usr\_id LEFT JOIN BILL\_DETAILS BD ON U.Usr\_id = BD.Usr\_id

GROUP BY U.Usr\_id, U.F\_name, U.L\_name

HAVING COUNT(TD.Trip\_id) > 5 AND SUM(BD.Total\_amt) > 500;

-- Simple Queries

--Retrieve the registration number, taxi model, and status of taxis driven by female drivers under the age of 30.

SELECT T.Registration\_no, T.Model, T.Taxi\_status FROM TAXI T, DRIVER D

WHERE T.Driver\_id = D.Driver\_id AND D.Gender = 'Female' AND D.Age < 30;

--Retrieve the names and contact numbers of drivers who have received feedback messages containing the word "excellent".

SELECT D.F\_name, D.L\_name, D.Conatct\_no FROM DRIVER D, FEEDBACK F

WHERE D.Driver\_id = F.Emp\_id AND F.Message LIKE '%excellent%';

--Retrieve the names and addresses of users who have taken trips in taxis registered in 2022.

SELECT U.F\_name, U.L\_name, U.USR\_Address FROM USER\_TBL U, TAXI T, TRIP\_DETAILS TD

WHERE U.Usr\_id = TD.Usr\_id AND TD.Taxi\_id = T.Taxi\_id AND T.Taxi\_Year = '2022';

--Retrieve the owner names and the total number of taxis they own, excluding owners who do not have any taxis.

SELECT I.Name AS Owner\_Name, O.No\_Cars AS Total\_Taxis\_Owned FROM INDIVIDUAL I, OWNS O

WHERE I.Owner\_id = O.Owner\_id AND O.No\_Cars > 0;

--To retrieve all feedback provided by users and sort them based on the feedback, bad to good

SELECT F.Fbk\_id, F.Message, F.Email, CASE

WHEN F.Message LIKE '%poor%' THEN 'Poor' WHEN F.Message LIKE '%bad%' THEN 'Bad'

WHEN F.Message LIKE '%average%' THEN 'Average' WHEN F.Message LIKE '%better%' THEN 'Average'

WHEN F.Message LIKE '%good%' THEN 'Good'

WHEN F.Message LIKE '%excellent%' THEN 'Excellent' WHEN F.Message LIKE '%best%' THEN 'Excellent'

ELSE 'Unknown' END AS Feedback\_Rating

FROM FEEDBACK F

ORDER BY Feedback\_Rating DESC;

--Nested queries:

--Retrieve the names of users who have taken trips using taxis owned by individuals named "John".

SELECT DISTINCT U.F\_name, U.L\_name FROM USER\_TBL U

WHERE U.Usr\_id IN (

SELECT TD.Usr\_id FROM TRIP\_DETAILS TD

WHERE TD.Taxi\_id IN (

SELECT OT.Taxi\_id FROM OWNER\_TAXI OT

JOIN INDIVIDUAL I ON OT.Owner\_id = I.Owner\_id WHERE I.Name = 'John'

));

--Retrieve the total number of trips taken by users who have given feedback with a message containing the word "complaint".

SELECT COUNT(\*) FROM TRIP\_DETAILS

WHERE Usr\_id IN (

SELECT DISTINCT Usr\_id FROM FEEDBACK

WHERE Message LIKE '%complaint%'

);

--Retrieve the names and contact numbers of drivers who have received feedback from users residing in "Chhindwara".

SELECT DISTINCT D.F\_name, D.L\_name, D.Conatct\_no FROM DRIVER D

WHERE D.Driver\_id IN ( SELECT DISTINCT F.Emp\_id

FROM FEEDBACK F

JOIN USER\_TBL U ON F.Usr\_id = U.Usr\_id WHERE U.USR\_Address LIKE '%Chhindwara%'

);

--Retrieve the registration numbers of taxis driven by male drivers under the age of 25.

SELECT DISTINCT T.Registration\_no FROM TAXI T

WHERE T.Driver\_id IN ( SELECT DISTINCT Driver\_id FROM DRIVER

WHERE Gender = 'Male' AND Age < 25

);

--UPDATE QUERIES

--Update Driver's Contact Number UPDATE DRIVER

SET Conatct\_no = 'New\_Contact\_Number' WHERE Driver\_id = 'Driver\_ID';

--Change Taxi Model for a Specific Taxi UPDATE TAXI

SET Model = 'New\_Taxi\_Model' WHERE Taxi\_id = 'Taxi\_ID';

--Update Trip Amount for a Specific Trip UPDATE TRIP\_DETAILS

SET Trip\_amt = 'New\_Trip\_Amount' WHERE Trip\_id = 'Trip\_ID';

--Update User's Address UPDATE USER\_TBL

SET USR\_Address = 'New\_Address' WHERE Usr\_id = 'User\_ID';

--Change Taxi Status UPDATE TAXI

SET Taxi\_status = 'Unavailable' WHERE Taxi\_type = 'Taxi\_Type';

--DELETE QUERIES

--Delete a Specific Taxi DELETE FROM TAXI

WHERE Taxi\_id = 'Taxi\_ID';

--Remove a Driver Record DELETE FROM DRIVER

WHERE Driver\_id = 'Driver\_ID';

--Delete a User Record DELETE FROM USER\_TBL

WHERE Usr\_id = 'User\_ID';

--Cancel a Trip

DELETE FROM TRIP\_DETAILS

WHERE Trip\_id = 'Trip\_ID';

--Remove an Individual Owner DELETE FROM INDIVIDUAL

WHERE Owner\_id = 'Owner\_ID';

# Modules In The Project:

## User Interface Module (Tkinter):

This module utilizes the Tkinter library to create a graphical user interface (GUI) for the application. It presents users with buttons that execute aggregated functions when clicked, allowing them to interact with the system efficiently. Additionally, the module displays the contents of relevant database tables within the GUI, providing users with real-time information. Functions include aggregating data from various tables, processing user requests, and displaying results dynamically on the interface. This module plays a crucial role in facilitating user interaction and system navigation, enhancing the overall user experience.

## Database Connection Module (SQLite3):

This module is responsible for managing the SQLite database utilized by the application. It establishes connections to the SQLite database where various types of information such as customer details, taxi records, and bookings are stored. Within this module, tables are created to structure the database, defining the schema for organizing data efficiently. Using the SQLite3 module in

Python, the application performs operations such as creating tables, inserting rows of data, and applying constraints to ensure data integrity.

## Customer Management Module:

Allows the call center staff to manage customer information such as adding new customers, updating existing records, and searching for customer details based on specific criteria.

## Taxi Management Module:

Handles taxi-related functionalities such as adding new taxis to the system, updating their availability status, and retrieving information about available taxis based on location or other parameters.

## Booking Module:

Facilitates the process of making and managing bookings. It includes functions to schedule taxi rides for customers, assign available taxis to bookings, update booking status, and handle cancellations or modifications.

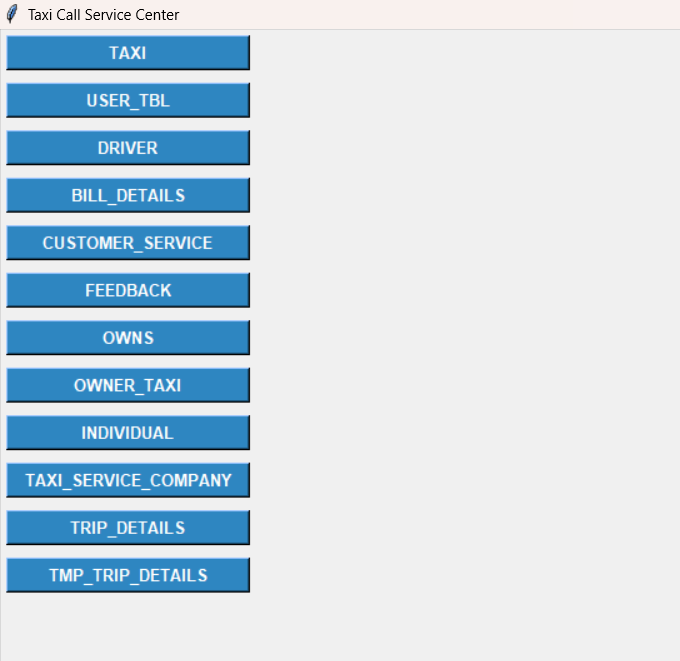
## Customer Service Module:

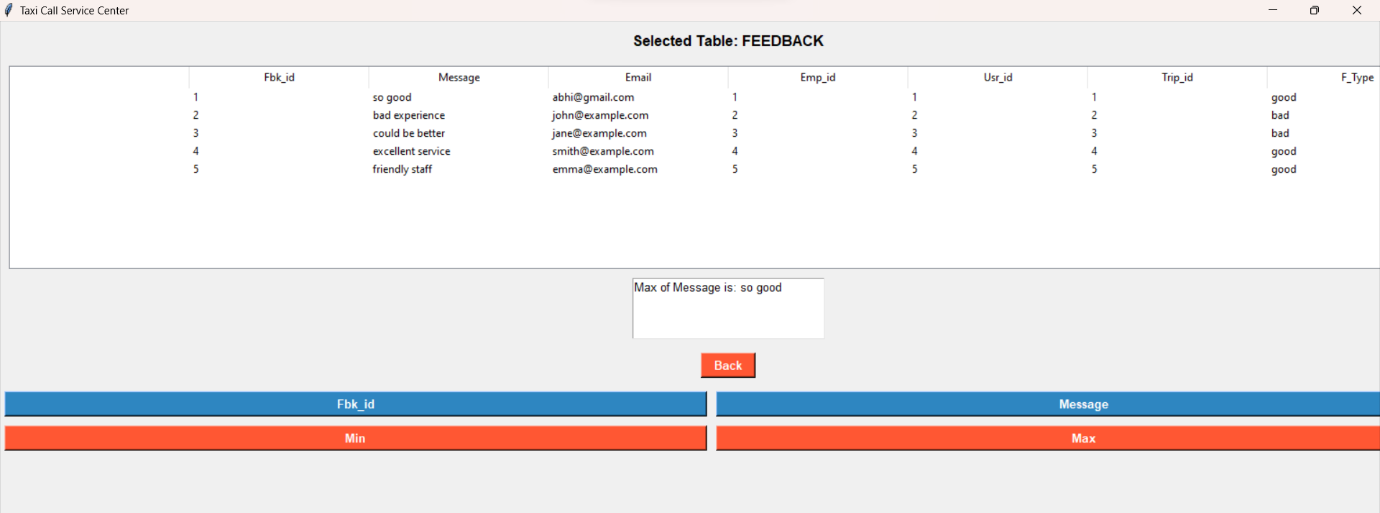
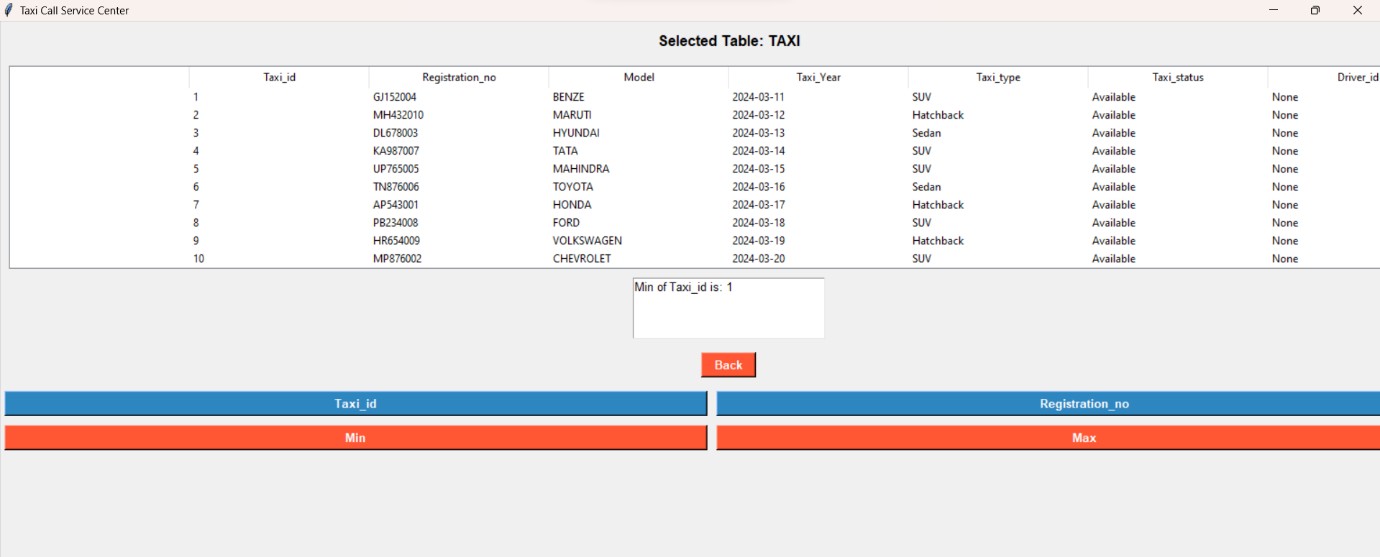
The Customer Service Module is dedicated to managing all aspects of customer interactions and service requests within the taxi call center system. It serves as a central hub for handling customer inquiries, bookings, and resolving issues efficiently.

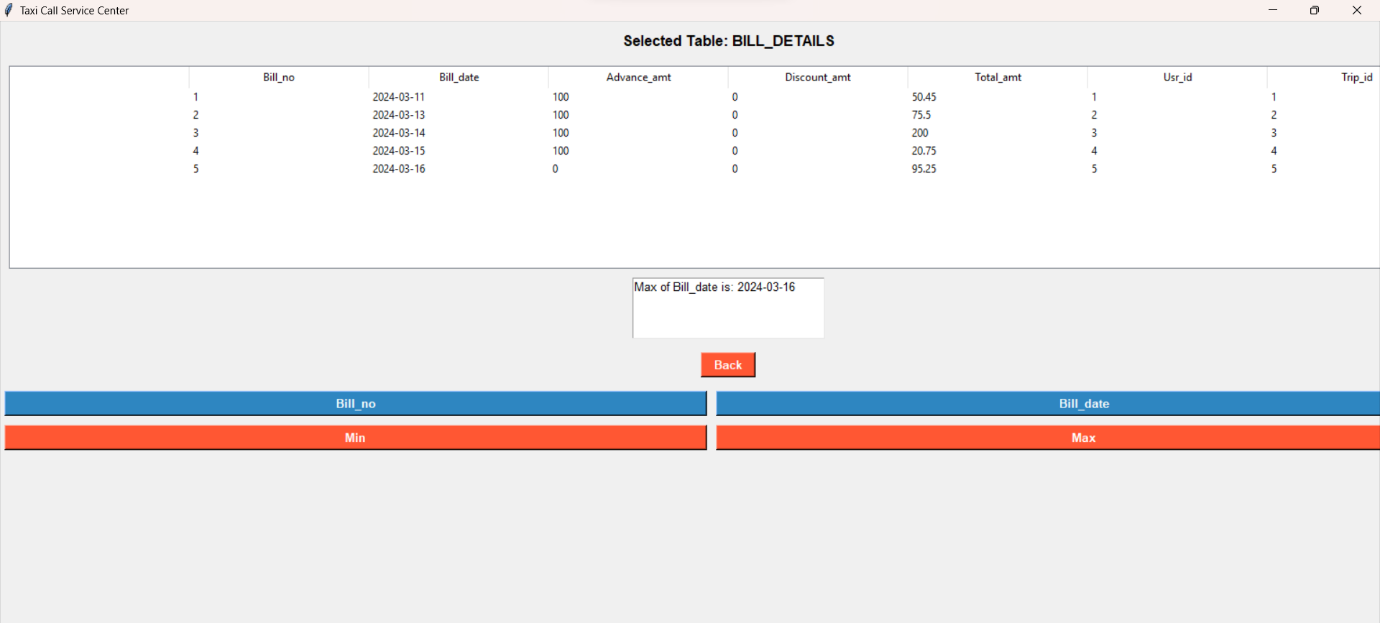
## Feedback Module:

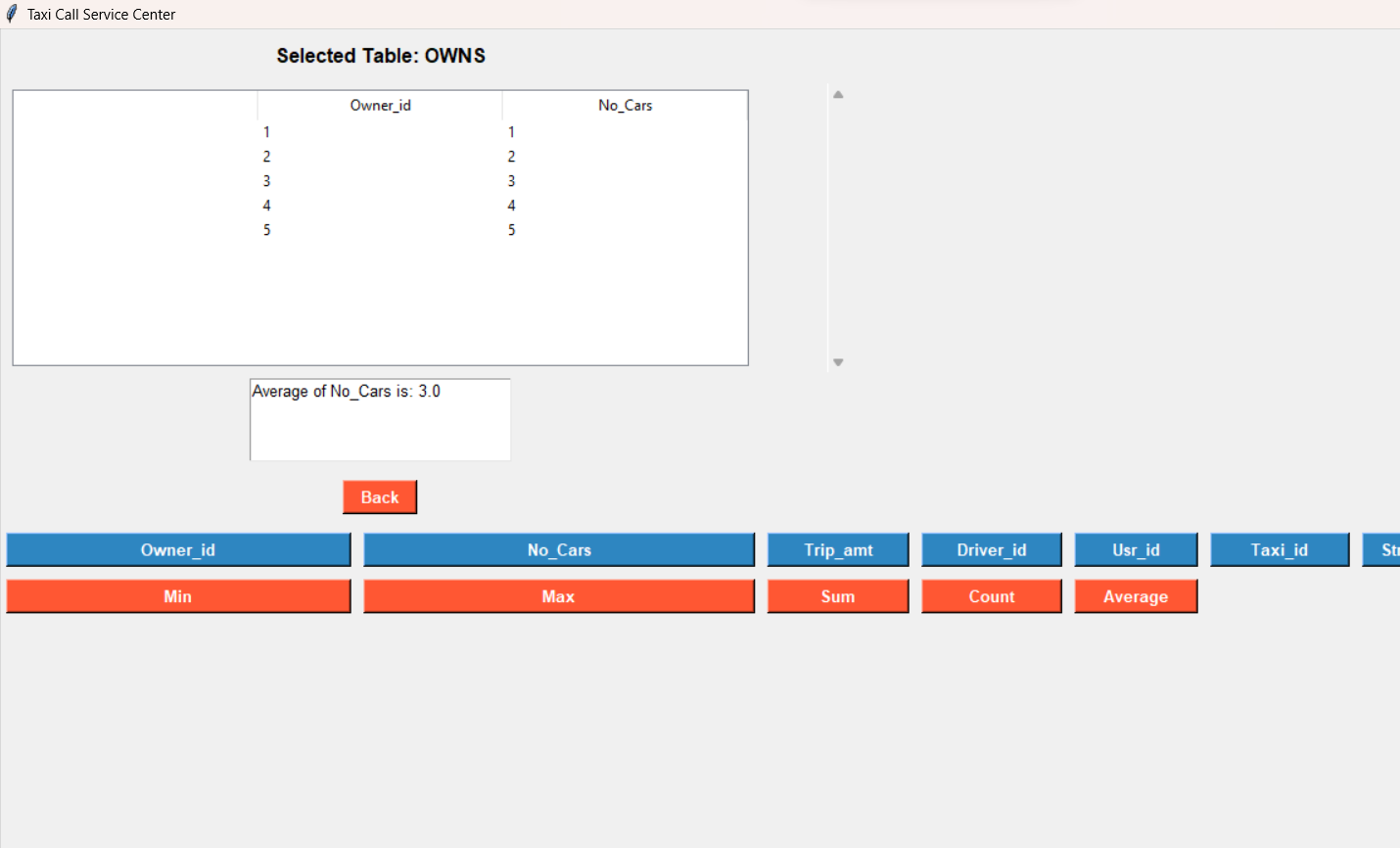
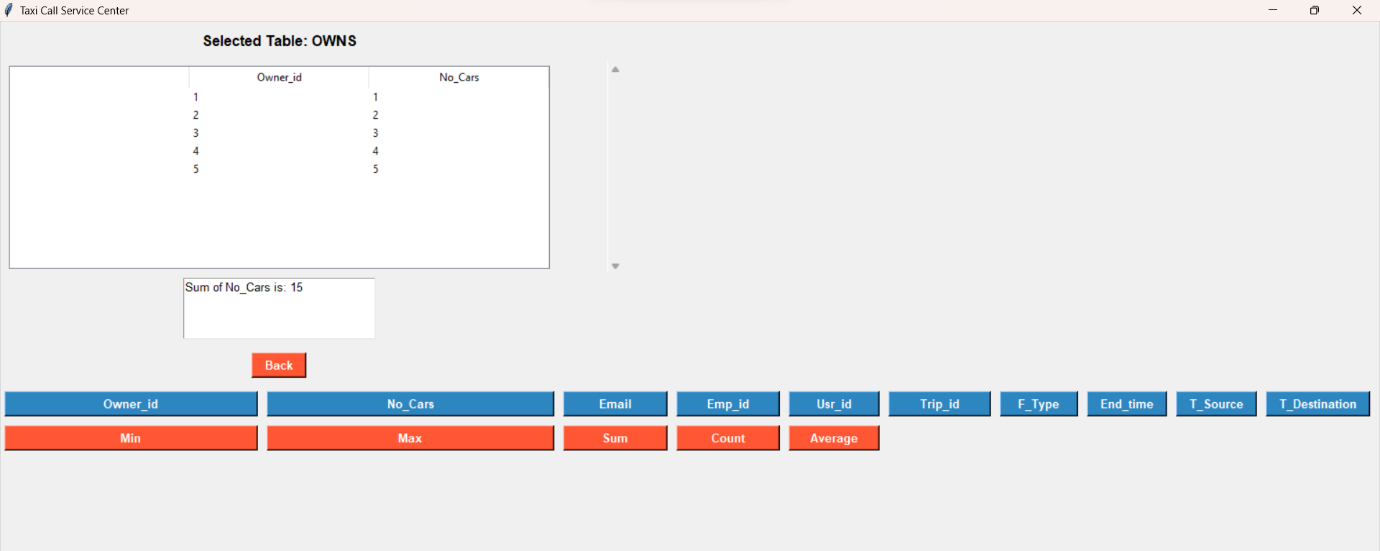
The Feedback Module is responsible for collecting, managing, and analyzing customer feedback to improve service quality and customer satisfaction. It provides a mechanism for customers to share their experiences, suggestions, and concerns with the taxi call center system.

# Implementation:









**Conclusion:**

The City Taxi Call Center DBMS project is a crucial step towards modernizing operations and improving overall efficiency. By embracing this technology, the call center aims to offer a seamless experience for both customers and drivers, ultimately establishing itself as a leader in the city's transportation services.