**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

A.Y. 2023 - 24

**Course: Database Management Systems**

***Project Report***

|  |  |  |
| --- | --- | --- |
| **Program** | **MBA(Tech.) AI** | |
| **Semester** | **IV** | |
| **Name of the Project** | **WanderDB – Travel Agency Management System** | |
| **Details of Project Members** | | |
| **Batch** | **Roll No.** | **Name** |
| **B1** | **R004** | **Abhipushp Maurya** |
| **R006** | **Amay Doshi** |
| **R007** | **Arya Gupte** |
| **Date of Submission: 01/04/2024** | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| **Roll No.** | **Name** | **Contribution** |
| **R004** | **Abhipushp Maurya** | **Table Definition, Data Entry, Queries, GUI,** |
| **R006** | **Amay Doshi** | **ER Diagram, Relational Model, Queries, GUI** |
| **R007** | **Arya Gupte** | **ER Diagram, Relational Model, Data Entry, Queries, GUI** |

***GitHub link of your project:*** [***https://github.com/abhipushp04/DBMS\_project\_travel***](https://github.com/abhipushp04/DBMS_project_travel)

**Project Report**

**WanderDB**

***Travel Agency Management System***

**by**

**Abhipushp Maurya - R004**

**Amay Doshi - R006**

**Arya Gupte - R007**

**Course: DBMS**

**AY: 2023-24**

Table of Contents

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline | 4 |
| **2** | Components of Database Design | 5 |
| **3** | Entity Relationship Diagram | 7 |
| **4** | Relational Model | 8 |
| **5** | Normalization | 8 |
| **6** | SQL Queries | 11 |
| **7** | Project Demonstration | 21 |
| **8** | Learning from the Project | 29 |
| **9** | Challenges you faced while doing the Project | 29 |
| **10** | Conclusion | 30 |

# ***I. Storyline***

In response to the pain points faced by travel agencies in today's competitive landscape, we propose the development and implementation of a comprehensive travel agency database solution: **WanderDB**. The problems of fragmented data management, complex tour package administration, internet competition, operational efficiency, flexibility and scalability, and consumer loyalty are all intended to be addressed by this solution. With the benefits of analytics, simplified file management, scalability, ease of use, and competitive efficiency, our database solution is customized to meet the unique requirements of travel agencies.

## **Principal attributes and advantages:**

1. **Capabilities for Analytics:**

* Problem: Travel agencies find it difficult to understand consumer behavior and industry trends, which makes it difficult for them to compete successfully and make wise decisions.
* Solution: Travel agencies will be able to obtain important insights into consumer behavior, market trends, and operational performance thanks to our database solution's powerful analytics capabilities. This will enable strategic planning and well-informed decision-making, assisting agencies in maintaining their competitiveness in the digital market.

1. **Streamlined Data Management:**

* Problem: Managing dispersed data across multiple platforms and systems causes errors, inefficiencies, and makes it harder to obtain important data.
* Solution: By providing a unified framework for fragmented files, our database solution will simplify data management and enhance efficiency, accuracy, and accessibility. This will improve overall operational effectiveness and lessen the agony of handling many data sources.

1. **Customizable Tour Package Management:**

* Problem: Designing and overseeing tour packages is a difficult and drawn-out procedure that frequently presents difficulties with profitability and customization.
* Solution: To solve the difficulty of complex tour package administration, our database solution will contain capabilities that make it easier to create and maintain bespoke tour packages. This will enable travel agencies to provide individualized experiences for their clients.

1. **Scalability and Operational Efficiency:**

* Problem: Travel agencies find it difficult to adjust to shifting market conditions and company needs due to inefficiencies in their operational procedures and a lack of scalability.
* Resolution: Our database solution's user-friendly interface is made to meet the needs of small travel firms by being simple to use and effective. The system's scalability further guarantees that it can develop alongside the agency, adjusting to shifting market conditions and business needs.

1. **Customer retention and competitive efficiency:**

* Problem: Travel agents have trouble keeping clients and encouraging loyalty in addition to having trouble competing with internet travel aggregators.
* Solution: Travel agencies will have the resources they need to successfully compete with online travel aggregators thanks to our database solution. Agencies can build enduring ties with their clientele by increasing customer retention and loyalty through the provision of tailored experiences and improved operational efficiency.

# ***II. Components of Database Design***

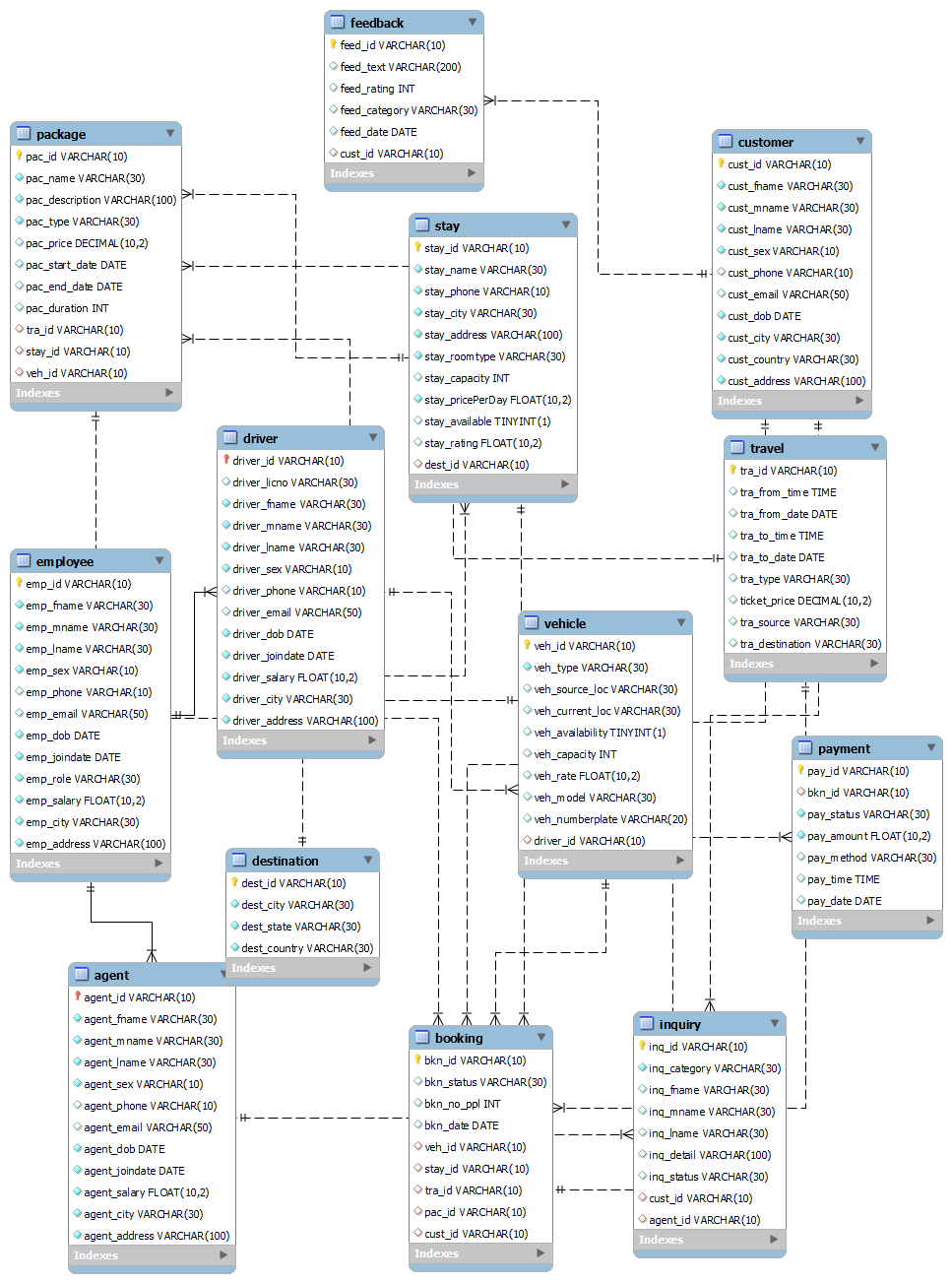
## **Entities -**

|  |  |  |
| --- | --- | --- |
| Table Name | Attributes | Description |
| Customer | ***cust\_id***, cust\_fname, cust\_mname, cust\_lname, cust\_sex, cust\_phone, cust\_email, cust\_dob, cust\_city, cust\_country, cust\_address | Contains details about customers |
| Employee | ***emp\_id***, emp\_fname, emp\_mname, emp\_lname, emp\_sex, emp\_phone, emp\_email, emp\_dob, emp\_joindate, emp\_role, emp\_salary, emp\_city, emp\_address | Contains details about employee |
| Destination | ***dest\_id***, dest\_city, dest\_state, dest\_country | Contains details about destination |
| Stay | ***stay\_id***, stay\_name, stay\_phone, stay\_city, stay\_address, stay\_roomtype, stay\_capacity, stay\_pricePerDay, stay\_available, stay\_rating, | Contains details about stays, hotels |
| Vehicle | ***veh\_id***, veh\_type, veh\_source\_loc, veh\_current\_loc, veh\_availability, veh\_capacity, veh\_rate, veh\_model, veh\_numberplate, | Contains details about vehicle |
| Inquiry | ***inq\_id***, inq\_category, inq\_fname, inq\_mname, inq\_lname, inq\_detail, inq\_status | Contains details about inquiry raised |
| Feedback | ***feed\_id***, feed\_text, feed\_rating, feed\_category, feed\_date | Contains details about feedback for different category |
| Travel | ***tra\_id***, tra\_from\_time, tra\_from\_date, tra\_to\_time, tra\_to\_date, tra\_type, tra\_source, tra\_destination, ticket\_price | Contains details about travel |
| Package | ***pac\_id***, pac\_name, pac\_description, pac\_type, pac\_price, pac\_start\_date, pac\_end\_date, pac\_duration | Contains details about package |
| Booking | ***bkn\_id***, bkn\_status, bkn\_no\_ppl, bkn\_date | Contains details about Booking tickets |
| Payment | ***pay\_id***, pay\_status, pay\_amount, pay\_method, pay\_time, pay\_date | Contains details about payment |

## **Relationships –**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Entity 1 | Entity 2 | Cardinality | Participation | Description |
| Employee-Agent | Employee | Agent | 1:1 | Partial : Total | IS-A |
| Employee-Driver | Employee | Driver | 1:1 | Partial : Total | IS-A |
| Stay-Destination | Stay | Destination | N:1 | Total : Partial |  |
| Driver-Vehicle | Driver | Vehicle | 1:N | Total : Total |  |
| Inquiry-Customer | Inquiry | Customer | N:1 | Total : Total | Weak Relation |
| Inquiry-Agent | Inquiry | Agent | 1:N | Total : Partial |  |
| Feedback-customer | Feedback | Customer | N:1 | Total : Partial | Weak Relation |
| Package-travel | Package | Travel | 1 : N | Partial : Partial |  |
| Package-stay | Package | Stay | 1 : N | Partial : Partial |  |
| Package-vehicle | Package | Vehicle | 1 : N | Partial : Partial |  |
| Booking-vehicle | Booking | Vehicle | N : 1 | Partial : Partial |  |
| Booking-stay | Booking | Stay | N : 1 | Partial : Partial |  |
| Booking-travel | Booking | Travel | N : 1 | Partial : Partial |  |
| Booking-package | Booking | Package | N : 1 | Partial : Partial |  |
| booking-customer | Booking | Customer | N : 1 | Total : Partial | Weak Relation |
| Booking-payment | Booking | Payment | 1 : 1 | Total : Total | Weak Relation |

# ***III. ER diagram***



# ***IV. Relational Model***

|  |  |
| --- | --- |
| Table Name | Attributes |
| Customer | ***cust\_id***, cust\_fname, cust\_mname, cust\_lname, cust\_sex, cust\_phone, cust\_email, cust\_dob, cust\_city, cust\_country, cust\_address |
| Employee | ***emp\_id***, emp\_fname, emp\_mname, emp\_lname, emp\_sex, emp\_phone, emp\_email, emp\_dob, emp\_joindate, emp\_role, emp\_salary, emp\_city, emp\_address |
| Destination | ***dest\_id***, dest\_city, dest\_state, dest\_country |
| Stay | ***stay\_id***, stay\_name, stay\_phone, stay\_city, stay\_address, stay\_roomtype, stay\_capacity, stay\_pricePerDay, stay\_available, stay\_rating, ***dest\_id\**** |
| Vehicle | ***veh\_id***, veh\_type, veh\_source\_loc, veh\_current\_loc, veh\_availability, veh\_capacity, veh\_rate, veh\_model, veh\_numberplate, ***driver\_id\**** |
| Inquiry | ***inq\_id***, inq\_category, inq\_fname, inq\_mname, inq\_lname, inq\_detail, inq\_status, ***cust\_id\*, agent\_id\**** |
| Feedback | ***feed\_id***, feed\_text, feed\_rating, feed\_category, feed\_date, ***cust\_id\**** |
| Travel | ***tra\_id***, tra\_from\_time, tra\_from\_date, tra\_to\_time, tra\_to\_date, tra\_type, tra\_source, tra\_destination, ticket\_price |
| Package | ***pac\_id***, pac\_name, pac\_description, pac\_type, pac\_price, pac\_start\_date, pac\_end\_date, pac\_duration, ***tra\_id\*, stay\_id\*, veh\_id\**** |
| Booking | ***bkn\_id***, bkn\_status, bkn\_no\_ppl, bkn\_date, ***veh\_id\*, stay\_id\*, tra\_id\*, pac\_id\*,cust\_id\**** |
| Payment | ***pay\_id***, pay\_status, pay\_amount, pay\_method, pay\_time, pay\_date |
| Driver | ***driver\_id*** *,* driver\_licno, driver\_fname, driver\_mname, driver\_lname, driver\_sex, driver\_phone, driver\_email, driver\_dob, driver\_joindate, driver\_salary, driver\_city, driver\_address, ***emp\_id\**** |
| Agent | ***agent\_id*** *,* agent\_fname, agent\_mname, agent\_lname, agent\_sex, agent\_phone, agent\_email, agent\_dob, agent\_joindate, agent\_salary, agent\_city, agent\_address, ***emp\_id*** |

# ***V. Normalization***

## ***Check 1NF -***

1. **Customer Table**:
   * The Customer table appears to be in 1NF. Each column contains atomic values, and there are no repeating groups.
2. **Employee, Agent, and Driver Tables**:
   * Similarly, these tables seem to be in 1NF. Each column contains atomic values, and there are no repeating groups.
3. **Destination Table**:
   * The Destination table is in 1NF. Each column contains atomic values, and there are no repeating groups.
4. **Stay Table**:
   * The Stay table is in 1NF. Each column contains atomic values, and there are no repeating groups.
5. **Vehicle Table**:
   * The Vehicle table is in 1NF. Each column contains atomic values, and there are no repeating groups.
6. **Inquiry Table**:
   * The Inquiry table is in 1NF. Each column contains atomic values, and there are no repeating groups.
7. **Travel Table**:
   * The Travel table is in 1NF. Each column contains atomic values, and there are no repeating groups.
8. **Feedback Table**:
   * The Feedback table is in 1NF. Each column contains atomic values, and there are no repeating groups.
9. **Package Table**:
   * The Package table is in 1NF. Each column contains atomic values, and there are no repeating groups.
10. **Booking Table**:
    * The Booking table is in 1NF. Each column contains atomic values, and there are no repeating groups.
11. **Payment Table**:
    * The Payment table is in 1NF. Each column contains atomic values, and there are no repeating groups.

## ***Check 2NF –***

1. **Customer Table**:
   * All attributes (cust\_fname, cust\_mname, cust\_lname, cust\_sex, cust\_phone, cust\_email, cust\_dob, cust\_city, cust\_country, cust\_address) seem to be functionally dependent on the primary key (cust\_id). This table satisfies 2NF.
2. **Employee, Agent, and Driver Tables**:
   * Similarly, all attributes in these tables seem to be functionally dependent on their respective primary keys (emp\_id, agent\_id, driver\_id). Thus, they satisfy 2NF.
3. **Destination Table**:
   * All attributes (dest\_city, dest\_state, dest\_country) appear to be functionally dependent on the primary key (dest\_id). This table satisfies 2NF.
4. **Stay Table**:
   * All attributes (stay\_name, stay\_phone, stay\_city, stay\_address, stay\_roomtype, stay\_capacity, stay\_pricePerDay, stay\_available, stay\_rating, dest\_id) seem to be functionally dependent on the primary key (stay\_id). This table satisfies 2NF.
5. **Vehicle Table**:
   * All attributes (veh\_type, veh\_source\_loc, veh\_current\_loc, veh\_availability, veh\_capacity, veh\_rate, veh\_model, veh\_numberplate, driver\_id) appear to be functionally dependent on the primary key (veh\_id). This table satisfies 2NF.
6. **Inquiry Table**:
   * All attributes (inq\_category, inq\_fname, inq\_mname, inq\_lname, inq\_detail, inq\_status, cust\_id, agent\_id) seem to be functionally dependent on the primary key (inq\_id). This table satisfies 2NF.
7. **Travel Table**:
   * All attributes (tra\_from\_time, tra\_from\_date, tra\_to\_time, tra\_to\_date, tra\_type, tra\_source, tra\_destination, ticket\_price) appear to be functionally dependent on the primary key (tra\_id). This table satisfies 2NF.
8. **Feedback Table**:
   * All attributes (feed\_text, feed\_rating, feed\_category, feed\_date, cust\_id) seem to be functionally dependent on the primary key (feed\_id). This table satisfies 2NF.
9. **Package Table**:
   * All attributes (pac\_name, pac\_description, pac\_type, pac\_price, pac\_start\_date, pac\_end\_date, pac\_duration, tra\_id, stay\_id, veh\_id) appear to be functionally dependent on the primary key (pac\_id). This table satisfies 2NF.
10. **Booking Table**:
    * All attributes (bkn\_status, bkn\_no\_ppl, bkn\_date, cust\_id, veh\_id, stay\_id, tra\_id, pac\_id) seem to be functionally dependent on the primary key (bkn\_id). This table satisfies 2NF.
11. **Payment Table**:
    * All attributes (pay\_status, pay\_amount, pay\_method, pay\_time, pay\_date, cust\_id) seem to be functionally dependent on the primary key (pay\_id). This table satisfies 2NF.

## ***Check 3NF –***

1. **Customer Table**:
   * All attributes seem to be directly dependent on the primary key (cust\_id). This table satisfies 3NF.
2. **Employee, Agent, and Driver Tables**:
   * Similar to the Customer table, all attributes appear to be directly dependent on their respective primary keys (emp\_id, agent\_id, driver\_id). These tables satisfy 3NF.
3. **Destination Table**:
   * All attributes seem to be directly dependent on the primary key (dest\_id). This table satisfies 3NF.
4. **Stay Table**:
   * All attributes seem to be directly dependent on the primary key (stay\_id). This table satisfies 3NF.
5. **Vehicle Table**:
   * All attributes appear to be directly dependent on the primary key (veh\_id). This table satisfies 3NF.
6. **Inquiry Table**:
   * All attributes appear to be directly dependent on the primary key (inq\_id). This table satisfies 3NF.
7. **Travel Table**:
   * All attributes appear to be directly dependent on the primary key (tra\_id). This table satisfies 3NF.
8. **Feedback Table**:
   * All attributes appear to be directly dependent on the primary key (feed\_id). This table satisfies 3NF.
9. **Package Table**:
   * All attributes appear to be directly dependent on the primary key (pac\_id). This table satisfies 3NF.
10. **Booking Table**:
    * All attributes seem to be directly dependent on the primary key (bkn\_id). This table satisfies 3NF.
11. **Payment Table**:
    * All attributes seem to be directly dependent on the primary key (pay\_id). This table satisfies 3NF.

## ***Check BC NF –***

1. **Customer Table**:
   * There are no non-trivial functional dependencies other than cust\_id → {all other attributes}, where cust\_id is the primary key. This table satisfies BCNF.
2. **Employee, Agent, and Driver Tables**:
   * Similarly, each of these tables has the primary key (emp\_id, agent\_id, driver\_id) determining all other attributes. They all satisfy BCNF.
3. **Destination Table**:
   * The primary key (dest\_id) determines all other attributes. This table satisfies BCNF.
4. **Stay Table**:
   * The primary key (stay\_id) determines all other attributes. This table satisfies BCNF.
5. **Vehicle Table**:
   * The primary key (veh\_id) determines all other attributes. This table satisfies BCNF.
6. **Inquiry Table**:
   * The primary key (inq\_id) determines all other attributes. This table satisfies BCNF.
7. **Travel Table**:
   * The primary key (tra\_id) determines all other attributes. This table satisfies BCNF.
8. **Feedback Table**:
   * The primary key (feed\_id) determines all other attributes. This table satisfies BCNF.
9. **Package Table**:
   * The primary key (pac\_id) determines all other attributes. This table satisfies BCNF.
10. **Booking Table**:
    * The primary key (bkn\_id) determines all other attributes. This table satisfies BCNF.
11. **Payment Table**:
    * The primary key (pay\_id) determines all other attributes. This table satisfies BCNF.

# ***VI. SQL Queries***

## ***Tables***

DROP DATABASE IF EXISTS 2000\_sql\_project;

CREATE DATABASE IF NOT EXISTS 2000\_sql\_project;

USE 2000\_sql\_project;

DROP TABLE IF EXISTS Customer;

CREATE TABLE IF NOT EXISTS Customer(

cust\_id varchar(10) PRIMARY KEY,

cust\_fname varchar(30) NOT NULL,

cust\_mname varchar(30) NOT NULL,

cust\_lname varchar(30) NOT NULL,

cust\_sex varchar(10) NOT NULL,

cust\_phone varchar(10),

cust\_email varchar(50),

cust\_dob date NOT NULL,

cust\_city varchar(30) NOT NULL,

cust\_country varchar(30) NOT NULL,

cust\_address varchar(100) NOT NULL

);

DROP TABLE IF EXISTS Employee;

CREATE TABLE IF NOT EXISTS Employee(

emp\_id varchar(10) PRIMARY KEY,

emp\_fname varchar(30) NOT NULL,

emp\_mname varchar(30) NOT NULL,

emp\_lname varchar(30) NOT NULL,

emp\_sex varchar(10) NOT NULL,

emp\_phone varchar(10),

emp\_email varchar(50),

emp\_dob date NOT NULL,

emp\_joindate date NOT NULL,

emp\_role varchar(30) NOT NULL,

emp\_salary float(10,2) NOT NULL,

emp\_city varchar(30) NOT NULL,

emp\_address varchar(100) NOT NULL

);

-- Create Agent table

DROP TABLE IF EXISTS Agent;

CREATE TABLE IF NOT EXISTS Agent (

agent\_id VARCHAR(10) PRIMARY KEY,

agent\_fname VARCHAR(30) NOT NULL,

agent\_mname VARCHAR(30) NOT NULL,

agent\_lname VARCHAR(30) NOT NULL,

agent\_sex VARCHAR(10) NOT NULL,

agent\_phone VARCHAR(10),

agent\_email VARCHAR(50),

agent\_dob DATE NOT NULL,

agent\_joindate DATE NOT NULL,

agent\_salary FLOAT(10,2) NOT NULL,

agent\_city VARCHAR(30) NOT NULL,

agent\_address VARCHAR(100) NOT NULL,

FOREIGN KEY (agent\_id) REFERENCES Employee(emp\_id)

);

-- Create Driver table

DROP TABLE IF EXISTS Driver;

CREATE TABLE IF NOT EXISTS Driver (

driver\_id VARCHAR(10) PRIMARY KEY,

driver\_licno VARCHAR(30),

driver\_fname VARCHAR(30) NOT NULL,

driver\_mname VARCHAR(30) NOT NULL,

driver\_lname VARCHAR(30) NOT NULL,

driver\_sex VARCHAR(10) NOT NULL,

driver\_phone VARCHAR(10),

driver\_email VARCHAR(50),

driver\_dob DATE NOT NULL,

driver\_joindate DATE NOT NULL,

driver\_salary FLOAT(10,2) NOT NULL,

driver\_city VARCHAR(30) NOT NULL,

driver\_address VARCHAR(100) NOT NULL,

FOREIGN KEY (driver\_id) REFERENCES Employee(emp\_id)

);

DROP TABLE IF EXISTS Destination;

CREATE TABLE IF NOT EXISTS Destination(

dest\_id varchar(10) PRIMARY KEY,

dest\_city varchar(30) NOT NULL,

dest\_state varchar(30) NOT NULL,

dest\_country varchar(30) NOT NULL

);

DROP TABLE IF EXISTS Stay;

CREATE TABLE IF NOT EXISTS Stay(

stay\_id varchar(10) PRIMARY KEY,

stay\_name varchar(30) NOT NULL,

stay\_phone VARCHAR(10) NOT NULL,

stay\_city varchar(30) NOT NULL,

stay\_address varchar(100) NOT NULL,

stay\_roomtype varchar(30) NOT NULL,

stay\_capacity int ,

stay\_pricePerDay float(10,2) NOT NULL,

stay\_available boolean,

stay\_rating float(10,2),

dest\_id varchar(10),

FOREIGN KEY (dest\_id) REFERENCES Destination(dest\_id)

);

DROP TABLE IF EXISTS Vehicle;

CREATE TABLE IF NOT EXISTS Vehicle(

veh\_id varchar(10) PRIMARY KEY,

veh\_type varchar(30) NOT NULL,

veh\_source\_loc varchar(30),

veh\_current\_loc varchar(30),

veh\_availability boolean,

veh\_capacity int ,

veh\_rate float(10, 2),

veh\_model varchar(30),

veh\_numberplate varchar(20),

driver\_id varchar(10),

FOREIGN KEY (driver\_id) REFERENCES Driver(driver\_id)

);

DROP TABLE IF EXISTS Inquiry;

CREATE TABLE IF NOT EXISTS Inquiry(

inq\_id varchar(10) PRIMARY KEY,

inq\_category varchar(30) NOT NULL,

inq\_fname varchar(30),

inq\_mname varchar(30),

inq\_lname varchar(30),

inq\_detail varchar(100),

inq\_status varchar(30),

cust\_id varchar(10),

agent\_id varchar(10),

FOREIGN KEY (cust\_id) REFERENCES Customer(cust\_id),

FOREIGN KEY (agent\_id) REFERENCES Agent(agent\_id)

);

DROP TABLE IF EXISTS Travel;

CREATE TABLE IF NOT EXISTS Travel(

tra\_id VARCHAR(10) PRIMARY KEY,

tra\_no\_ppl INT, -- per seat price

tra\_from\_time TIME,

tra\_from\_date DATE,

tra\_to\_time TIME,

tra\_to\_date DATE,

tra\_type VARCHAR(30),

tra\_source VARCHAR(30),

tra\_destination VARCHAR(30)

);

ALTER TABLE Travel

ADD COLUMN ticket\_price DECIMAL(10, 2) AFTER tra\_type,

DROP COLUMN tra\_no\_ppl;

DROP TABLE IF EXISTS Feedback;

CREATE TABLE IF NOT EXISTS Feedback(

feed\_id VARCHAR(10) PRIMARY KEY,

feed\_text VARCHAR(200),

feed\_rating INT,

feed\_category VARCHAR(30),

feed\_date DATE,

cust\_id VARCHAR(10),

FOREIGN KEY (cust\_id) REFERENCES Customer(cust\_id)

);

DROP TABLE IF EXISTS Package;

CREATE TABLE IF NOT EXISTS Package(

pac\_id varchar(10) PRIMARY KEY,

pac\_name varchar(30) NOT NULL,

pac\_description varchar(100) NOT NULL,

pac\_type varchar(30) NOT NULL, -- LIKE ADVENTURE, FAMILY, LIESURE, etc.

pac\_price decimal(10,2),

pac\_start\_date date,

pac\_end\_date date,

pac\_duration int,

tra\_id varchar(10),

stay\_id varchar(10),

veh\_id varchar(10),

FOREIGN KEY (tra\_id) REFERENCES Travel(tra\_id),

FOREIGN KEY (stay\_id) REFERENCES Stay(stay\_id),

FOREIGN KEY (veh\_id) REFERENCES Vehicle(veh\_id)

);

DROP TABLE IF EXISTS Booking;

CREATE TABLE IF NOT EXISTS Booking(

bkn\_id VARCHAR(10) PRIMARY KEY,

bkn\_status VARCHAR(30),

bkn\_no\_ppl INT,

bkn\_date DATE,

veh\_id VARCHAR(10),

stay\_id VARCHAR(10),

tra\_id VARCHAR(10),

pac\_id VARCHAR(10),

cust\_id VARCHAR(10),

FOREIGN KEY (veh\_id) REFERENCES Vehicle(veh\_id),

FOREIGN KEY (stay\_id) REFERENCES Stay(stay\_id),

FOREIGN KEY (tra\_id) REFERENCES Travel(tra\_id),

FOREIGN KEY (pac\_id) REFERENCES Package(pac\_id),

FOREIGN KEY (cust\_id) REFERENCES Customer(cust\_id)

);

DROP TABLE IF EXISTS Customer\_History;

CREATE TABLE IF NOT EXISTS Customer\_History(

his\_id VARCHAR(10) PRIMARY KEY,

his\_type VARCHAR(30),

his\_timestamp TIMESTAMP,

cust\_id VARCHAR(10),

FOREIGN KEY (cust\_id) REFERENCES Customer(cust\_id)

);

DROP TABLE IF EXISTS Payment;

CREATE TABLE IF NOT EXISTS Payment(

pay\_id VARCHAR(10) PRIMARY KEY,

pay\_status VARCHAR(30) NOT NULL,

pay\_amount FLOAT(10, 2) NOT NULL,

pay\_method VARCHAR(30),

pay\_time TIME,

pay\_date DATE

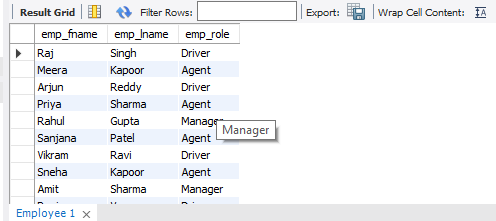
);

SHOW TABLES;

## ***Basic Quesries***

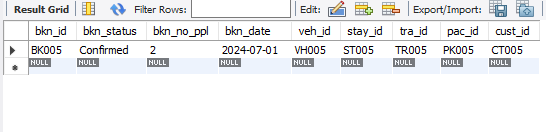
1. Retrieve the names and roles of all employees.

SELECT emp\_fname, emp\_lname, emp\_role FROM Employee;



1. Retrieve all bookings made by a specific customer.

SELECT \* FROM Booking WHERE cust\_id = 'CT005';

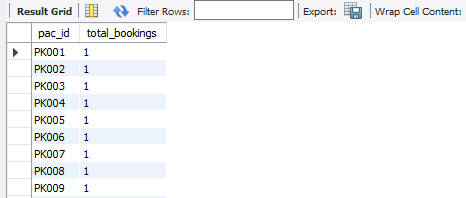


1. Retrieve the total number of bookings made for each travel package.

SELECT pac\_id, COUNT(\*) AS total\_bookings

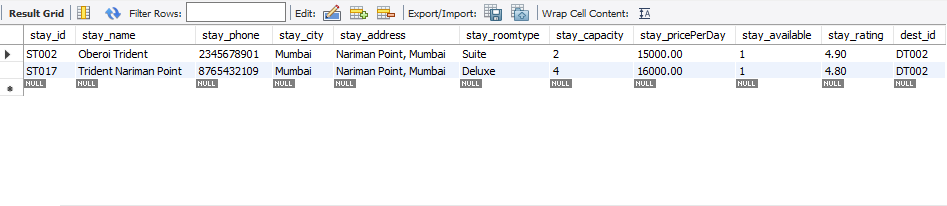
FROM Booking

GROUP BY pac\_id;



1. Retrieve the details of stays available in a specific city.

SELECT \* FROM Stay WHERE stay\_city = 'Mumbai';

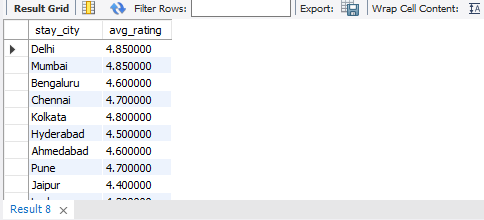


1. Retrieve the average rating of stays in each city

SELECT stay\_city, AVG(stay\_rating) AS avg\_rating

FROM Stay

GROUP BY stay\_city;

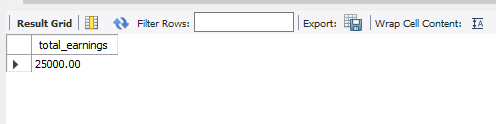


1. Retrieve the total amount earned from bookings made in a specific month

SELECT SUM(pay\_amount) AS total\_earnings

FROM Payment

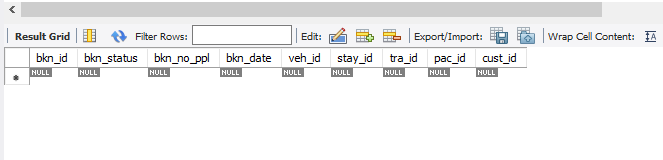
WHERE MONTH(pay\_date) = 01;



1. Retrieve the details of bookings made by customers who have not provided feedback.

SELECT \* FROM Booking

WHERE bkn\_id NOT IN (SELECT bkn\_id FROM Feedback);



1. Retrieve the top 5 most popular destinations based on the number of booking

SELECT dest\_id, COUNT(\*) AS total\_bookings

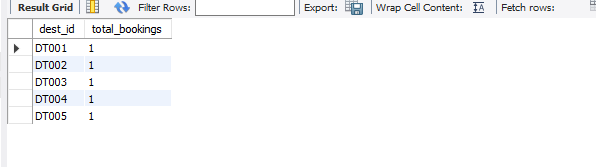
FROM Booking

JOIN Stay ON Booking.stay\_id = Stay.stay\_id

GROUP BY dest\_id

ORDER BY total\_bookings DESC

LIMIT 5;



1. Retrieve the details of customers who have made bookings for adventure packages and have a rating of more than 4 for their stays.

SELECT \*

FROM Customer

WHERE cust\_id IN (

SELECT DISTINCT Booking.cust\_id

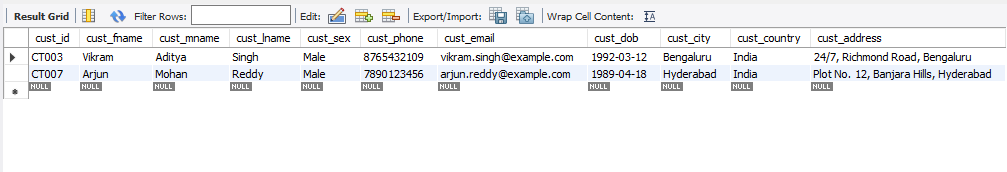
FROM Booking

JOIN Package ON Booking.pac\_id = Package.pac\_id

JOIN Stay ON Package.stay\_id = Stay.stay\_id

WHERE Package.pac\_type = 'ADVENTURE' AND Stay.stay\_rating > 4

);



1. Retrieve the details of stays where the price per day is less than the average price per day across all stays

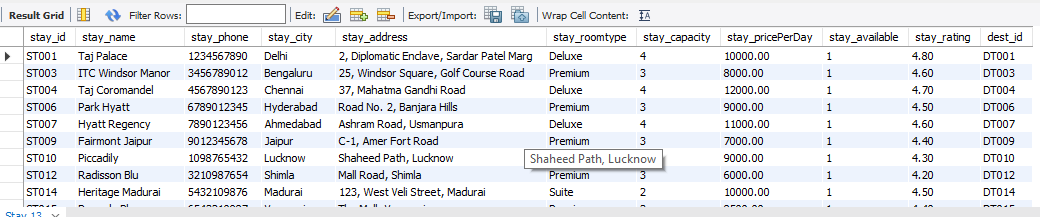
SELECT \*

FROM Stay

WHERE stay\_pricePerDay < (

SELECT AVG(stay\_pricePerDay) FROM Stay

);



1. Retrieve the details of packages that include stays with a rating higher than the average rating across all stays.

SELECT \*

FROM Package

WHERE stay\_id IN (

SELECT stay\_id

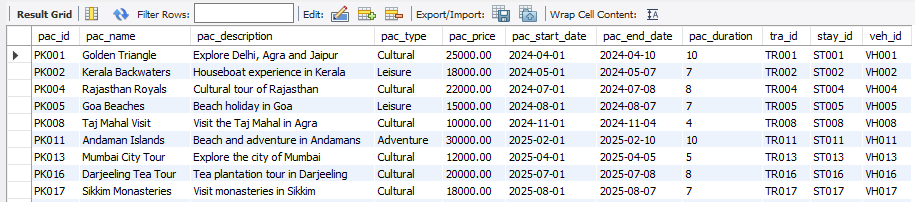
FROM Stay

WHERE stay\_rating > (

SELECT AVG(stay\_rating) FROM Stay

)

);



1. Retrieve the details of customers who have made the highest number of bookings.

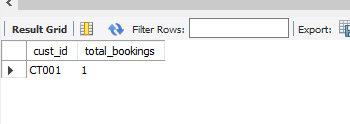
SELECT cust\_id, COUNT(\*) AS total\_bookings

FROM Booking

GROUP BY cust\_id

ORDER BY total\_bookings DESC

LIMIT 1;



1. Retrieve the details of stays that have not been booked by any customer

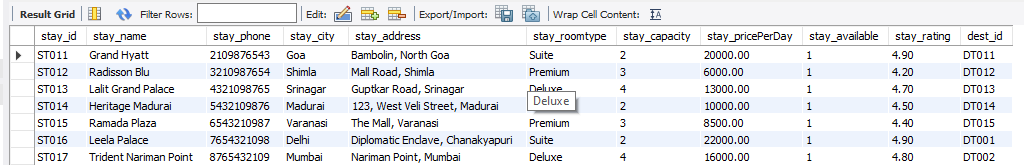
SELECT \*

FROM Stay

WHERE stay\_id NOT IN (

SELECT stay\_id FROM Booking

);



1. Retrieve the total number of bookings made for each destination, ordered by the number of bookings in descending order.

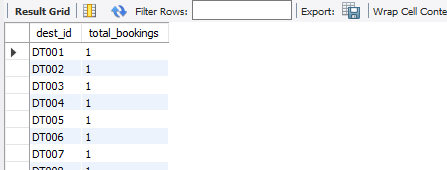
SELECT dest\_id, COUNT(\*) AS total\_bookings

FROM Booking

JOIN Stay ON Booking.stay\_id = Stay.stay\_id

GROUP BY dest\_id

ORDER BY total\_bookings DESC;

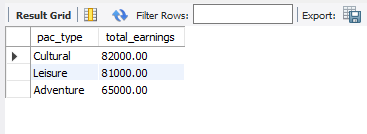


1. SELECT Package.pac\_type, SUM(Package.pac\_price) AS total\_earnings

FROM Package

JOIN Booking ON Package.pac\_id = Booking.pac\_id

GROUP BY Package.pac\_type;



1. Retrieve the details of packages that include stays with a room capacity greater than 3

SELECT \*

FROM Package

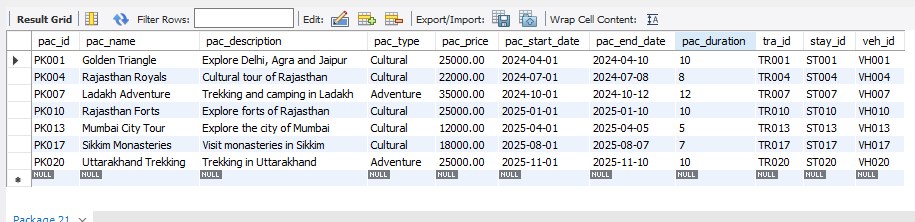
WHERE stay\_id IN (

SELECT stay\_id

FROM Stay

WHERE stay\_capacity > 3

);



1. Retrieve the details of stays that are fully booked (no available rooms)

SELECT \*

FROM Stay

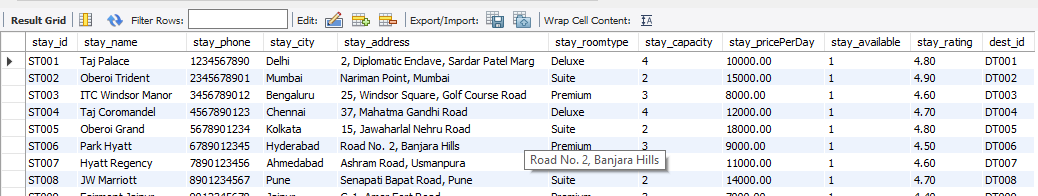
WHERE stay\_id NOT IN (

SELECT stay\_id

FROM Booking

WHERE bkn\_status = 'BOOKED'

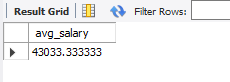
);



1. Retrieve the average salary of all employees.

SELECT AVG(emp\_salary) AS avg\_salary

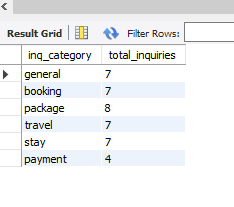
FROM Employee;



1. Retrieve the total number of inquiries received for each category.

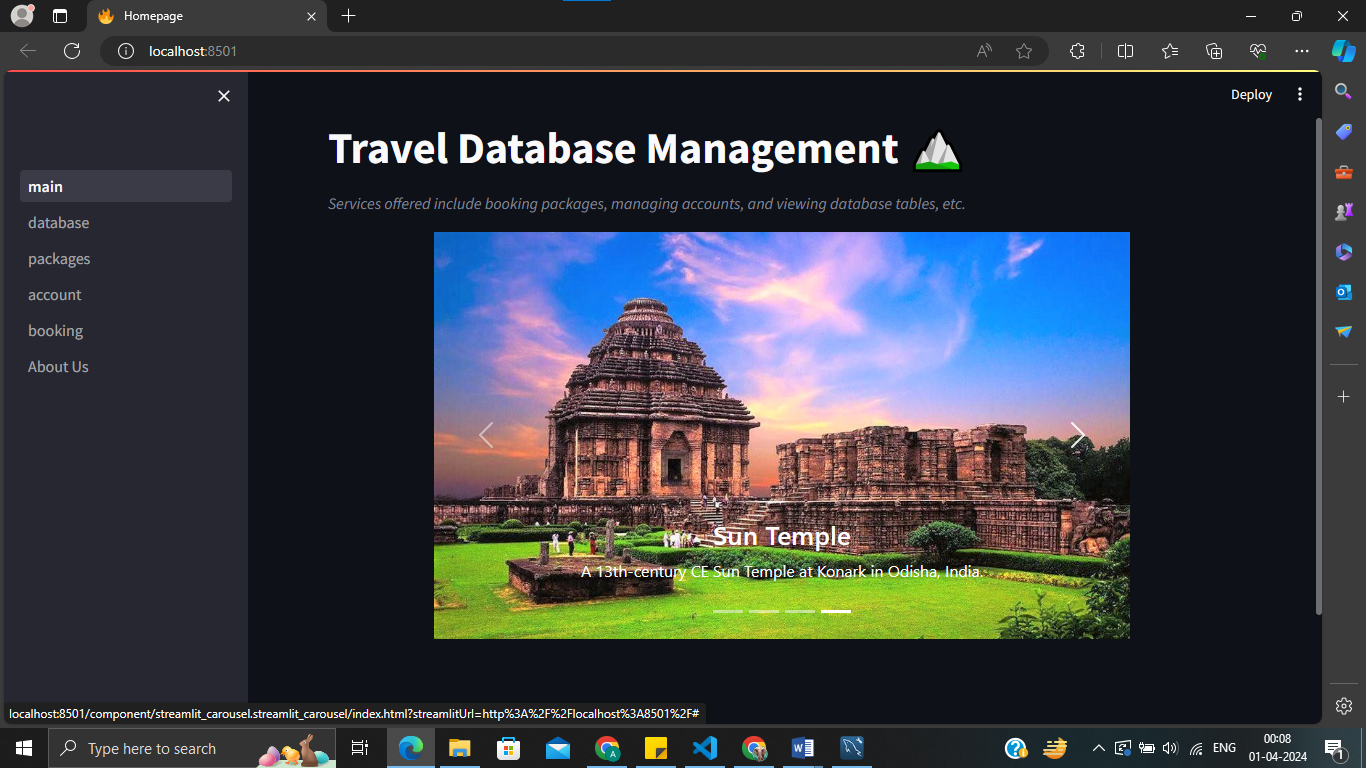
SELECT inq\_category, COUNT(\*) AS total\_inquiries FROM Inquiry

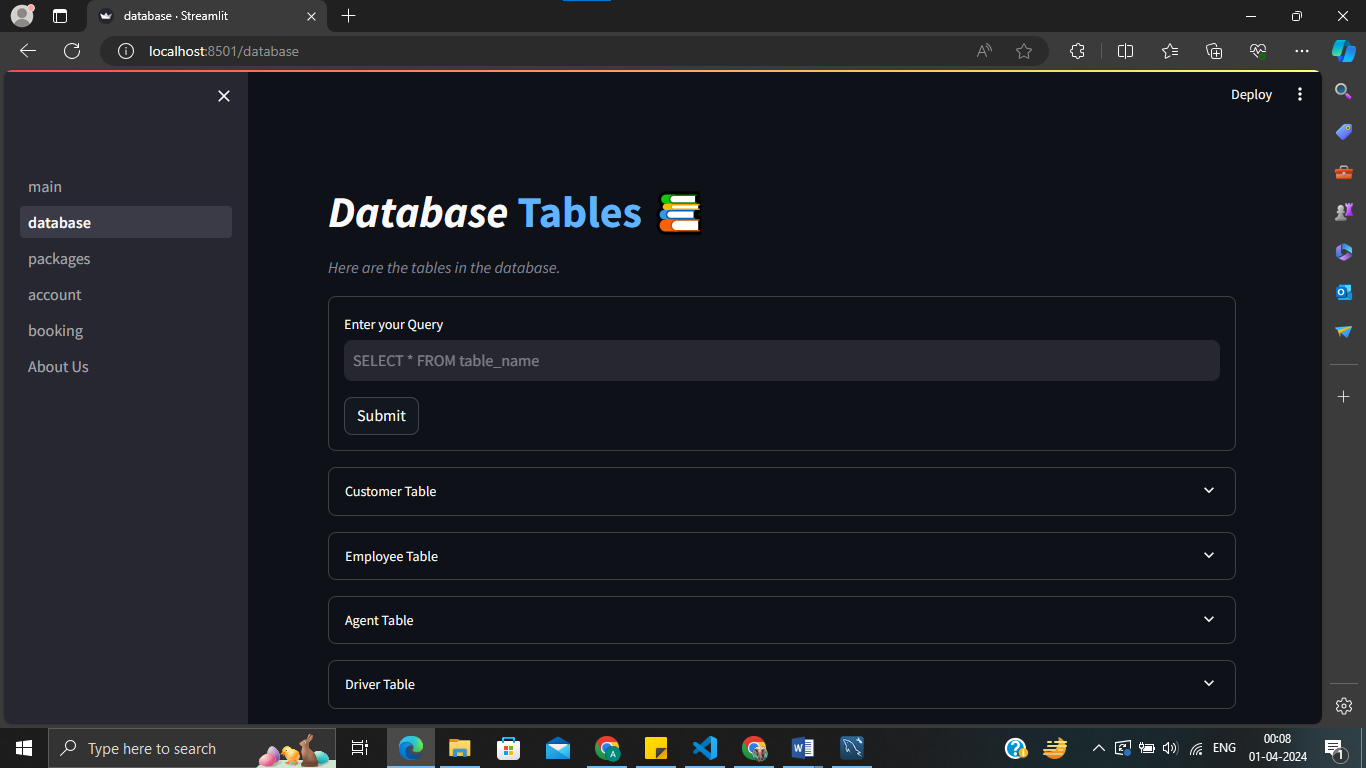
GROUP BY inq\_category;

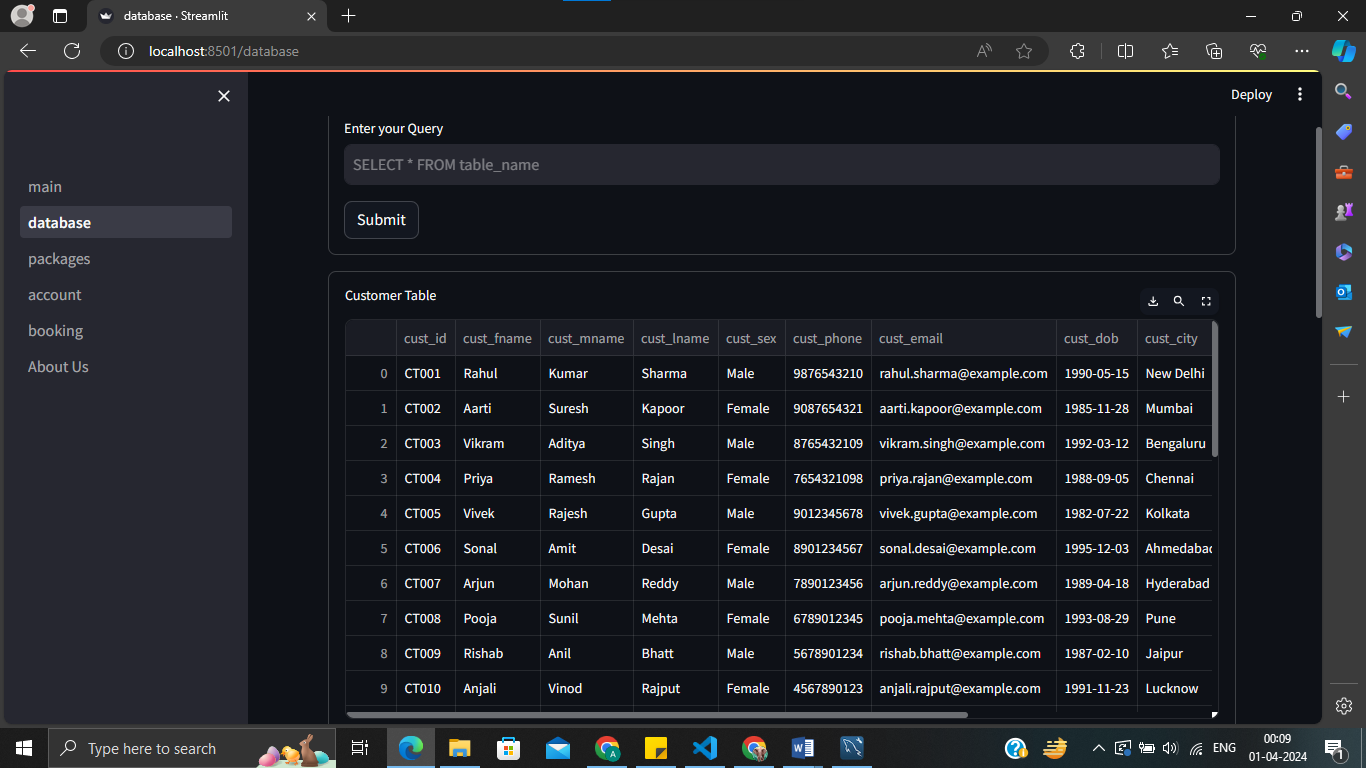


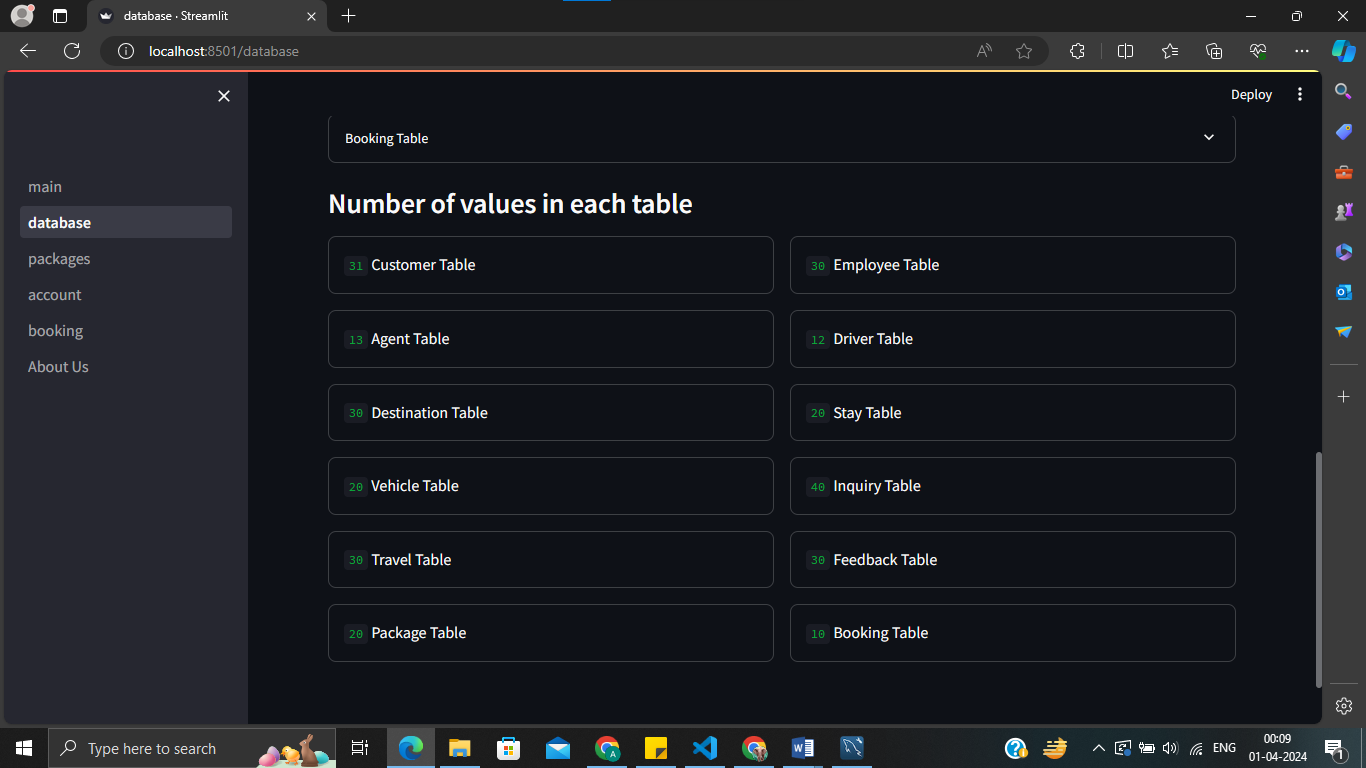
# ***VI. Project demonstration***

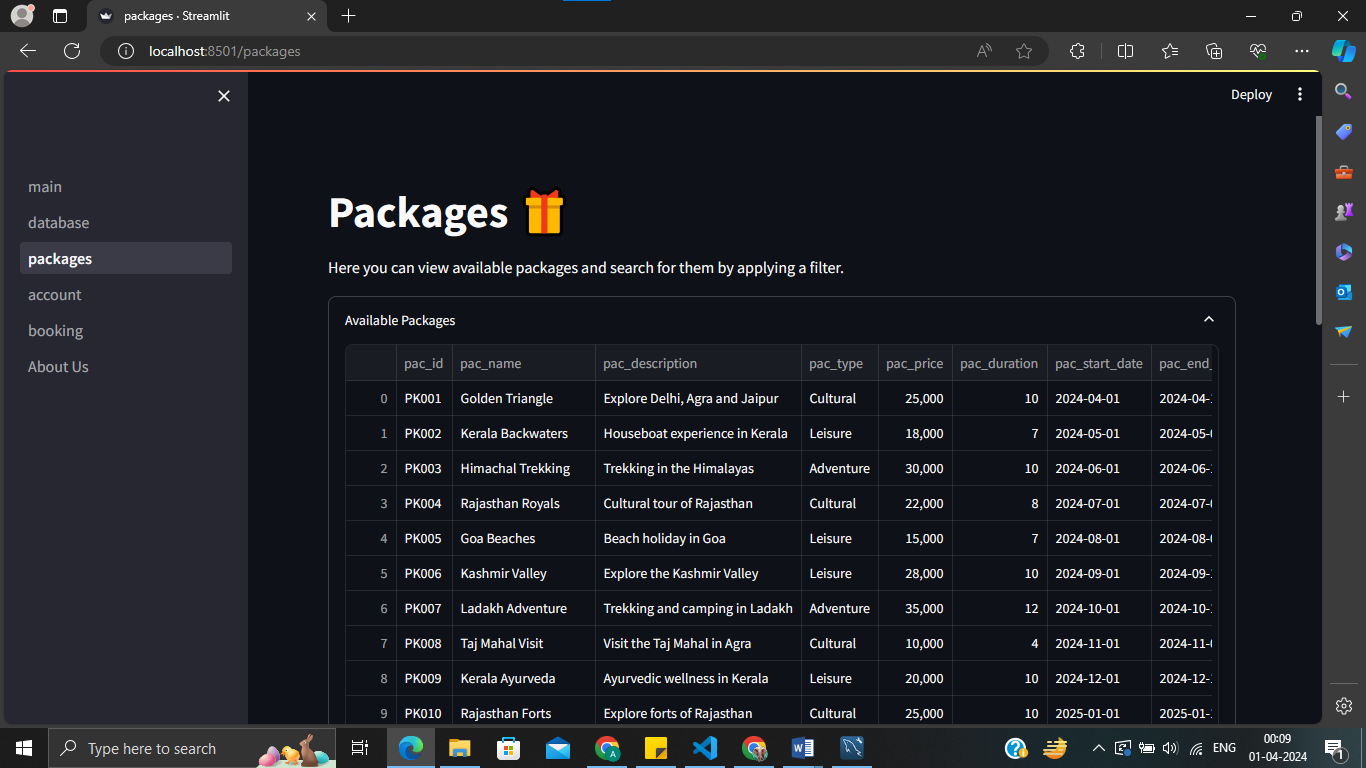
There are 6 files for this project run the main.py file in streamlit and then it will launch the webapp.

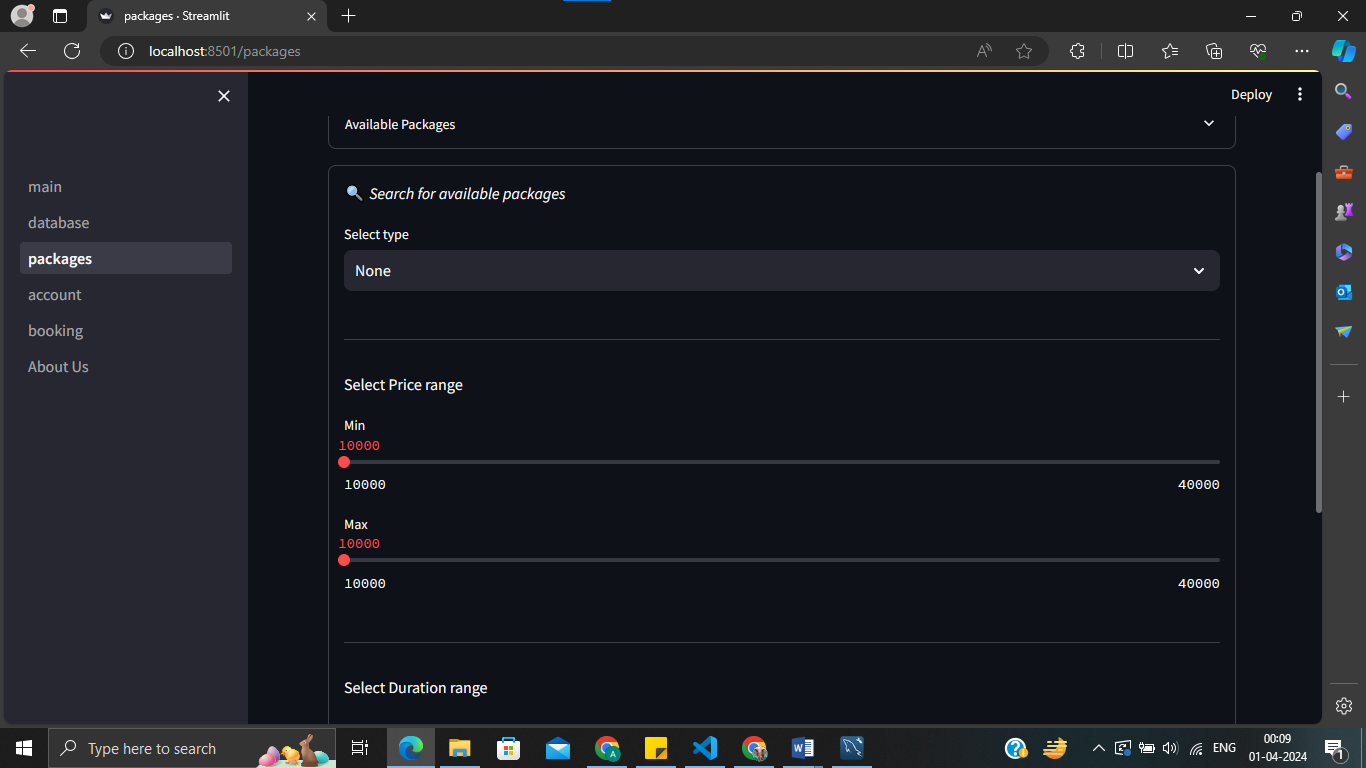


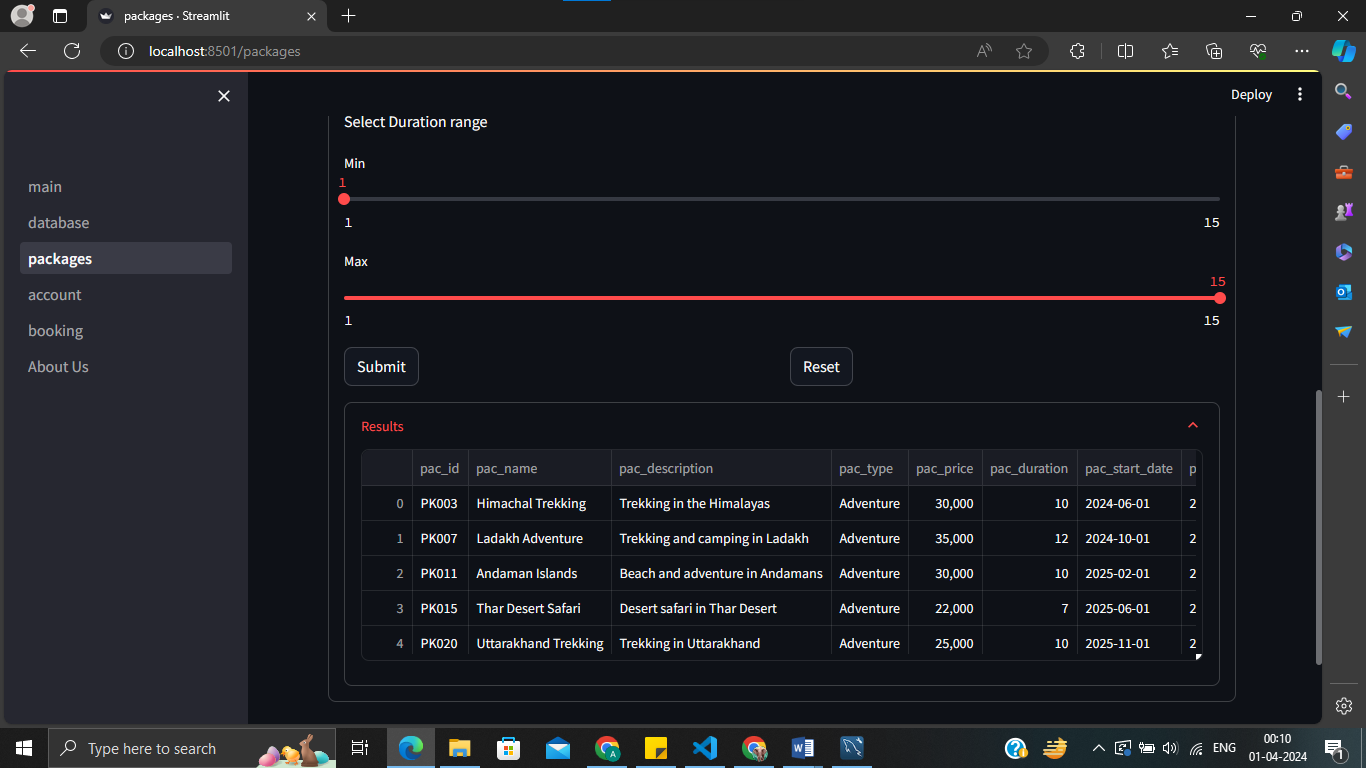


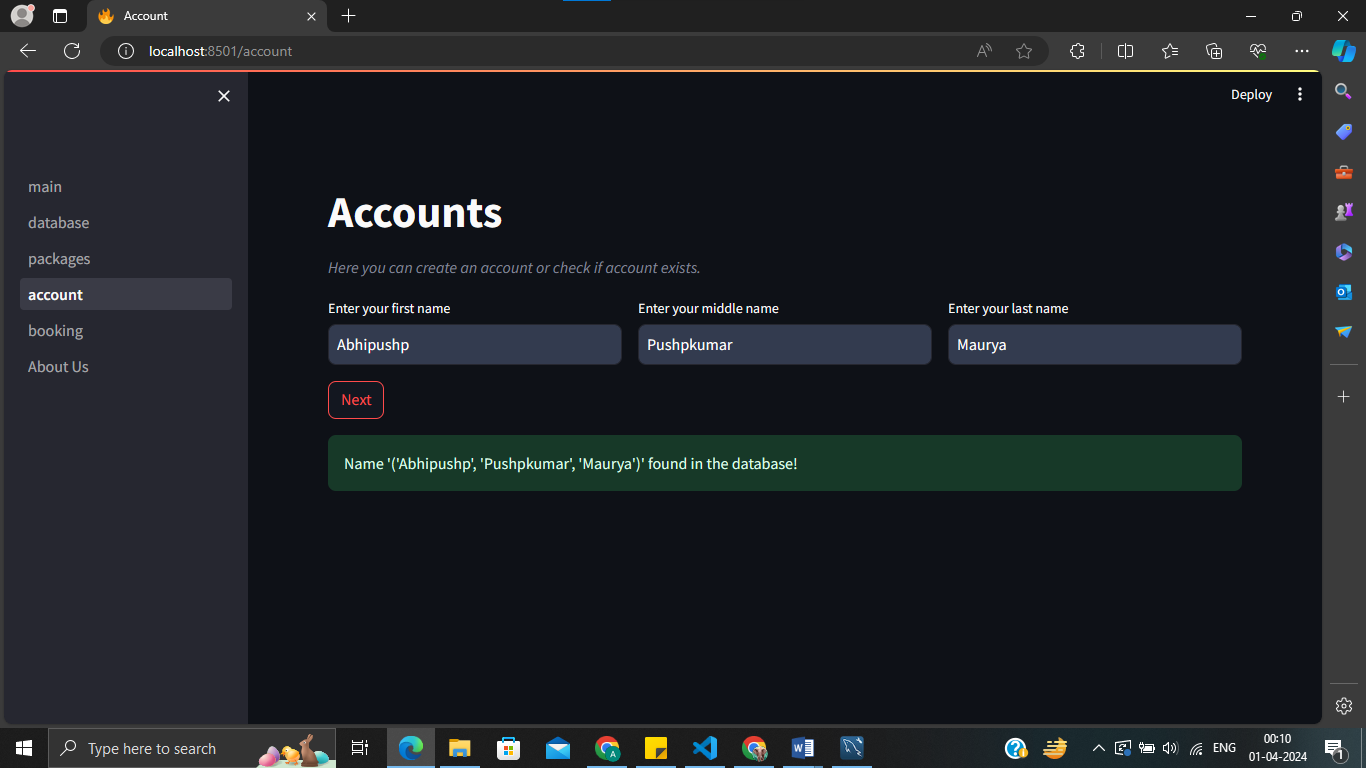


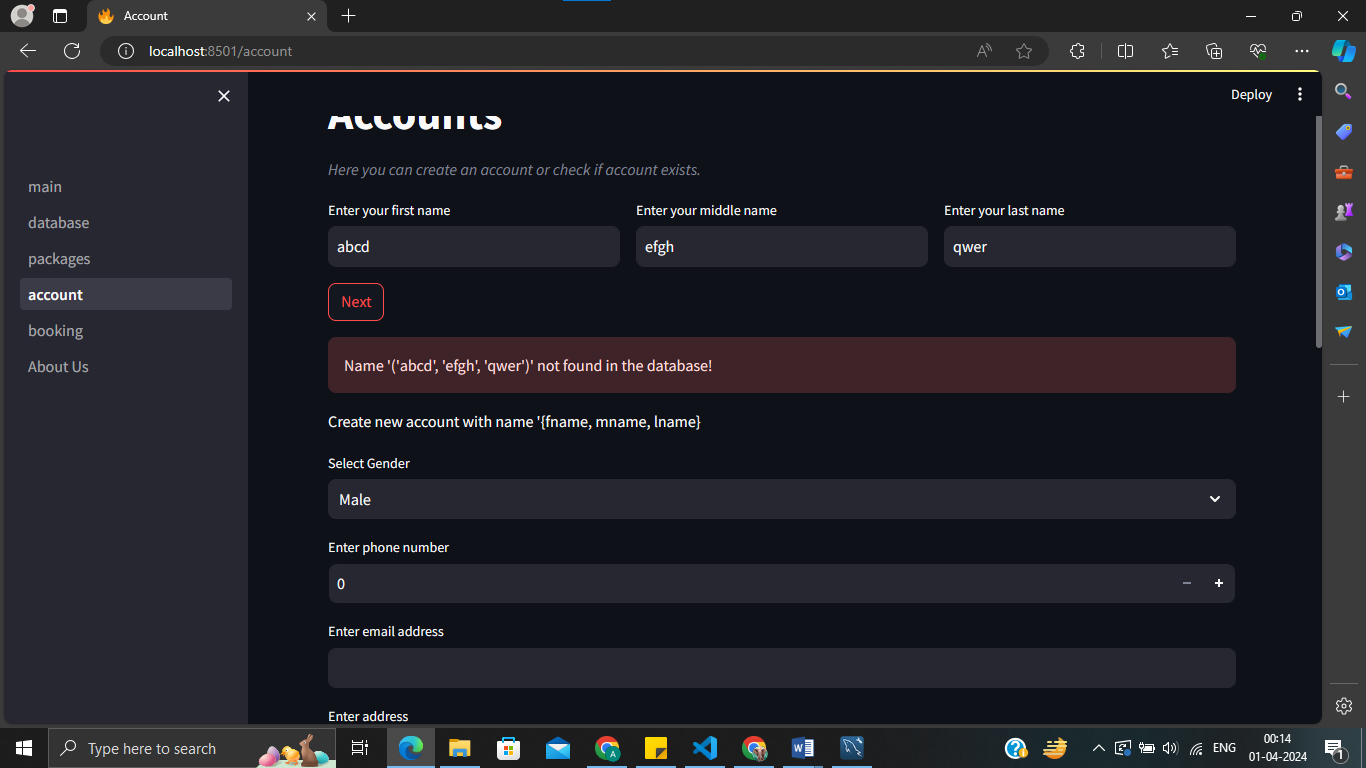


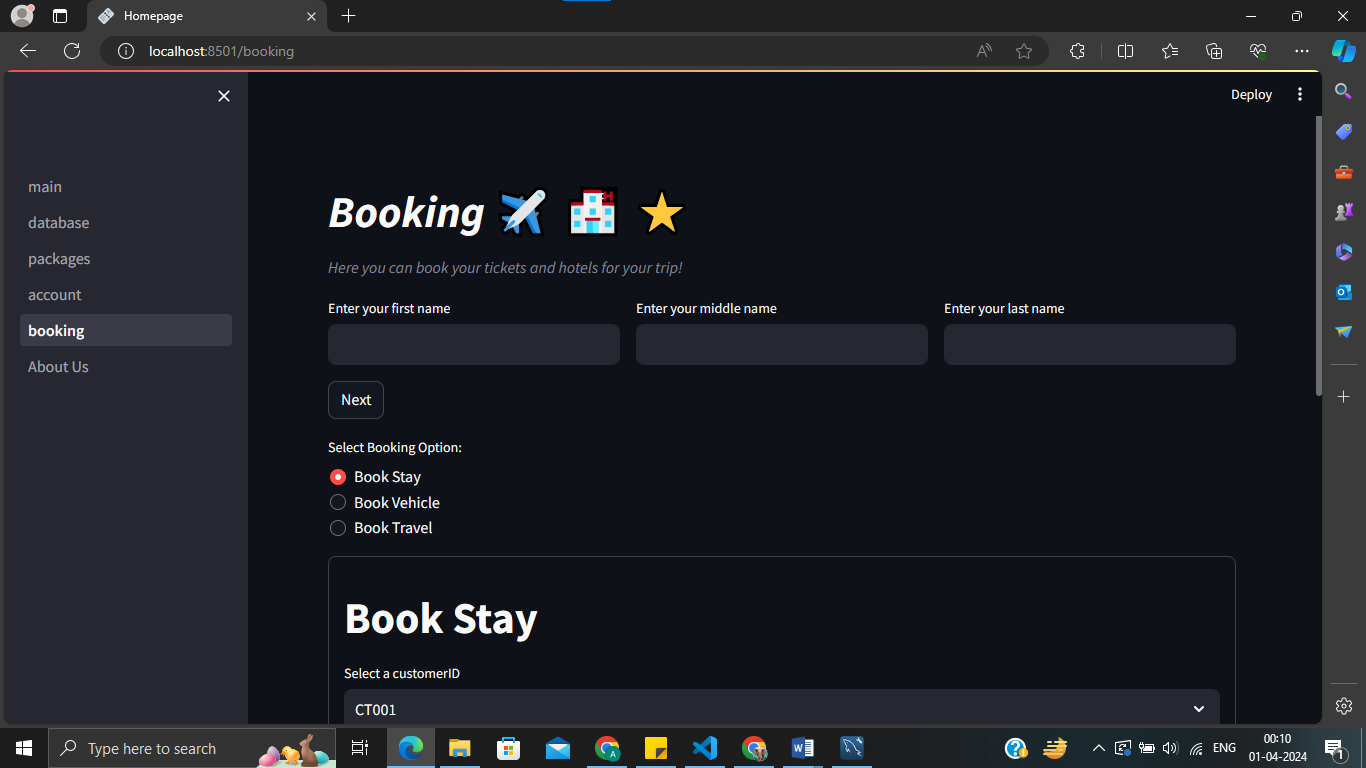


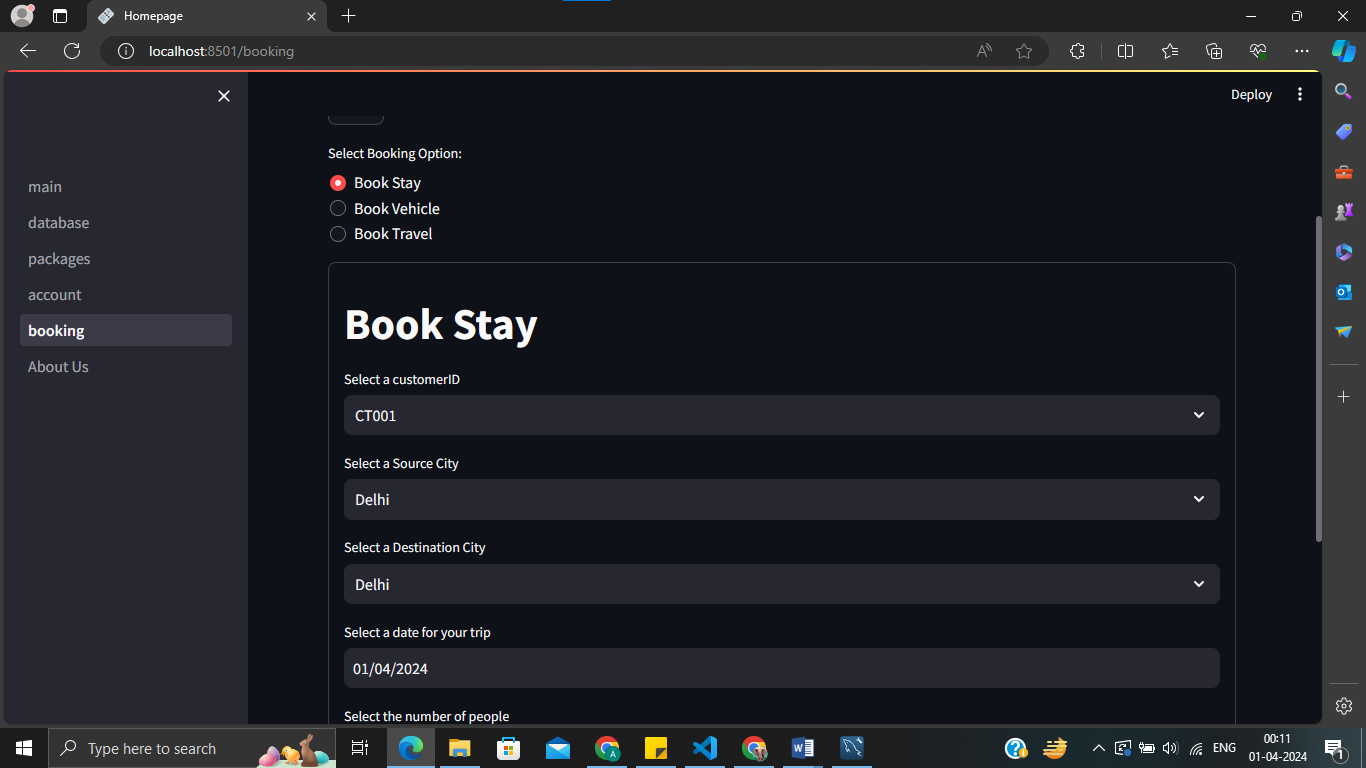


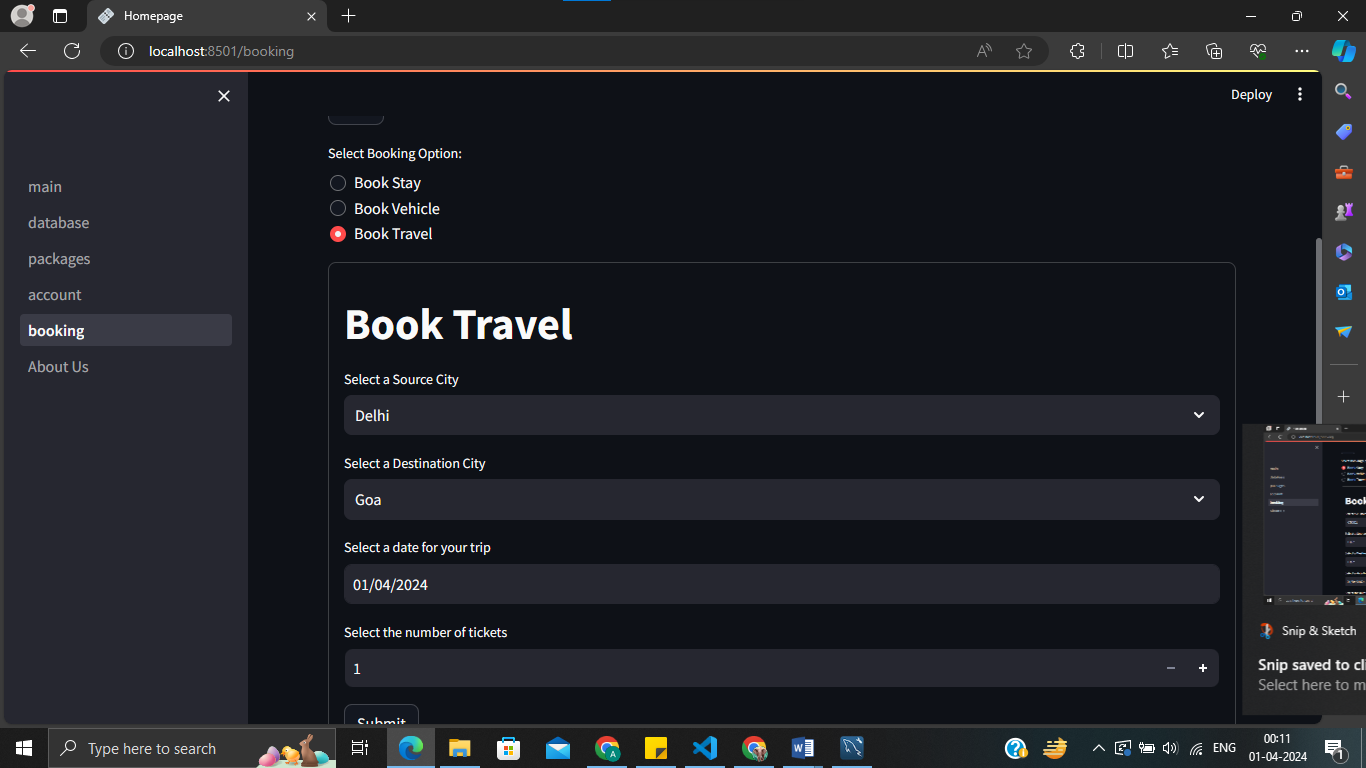


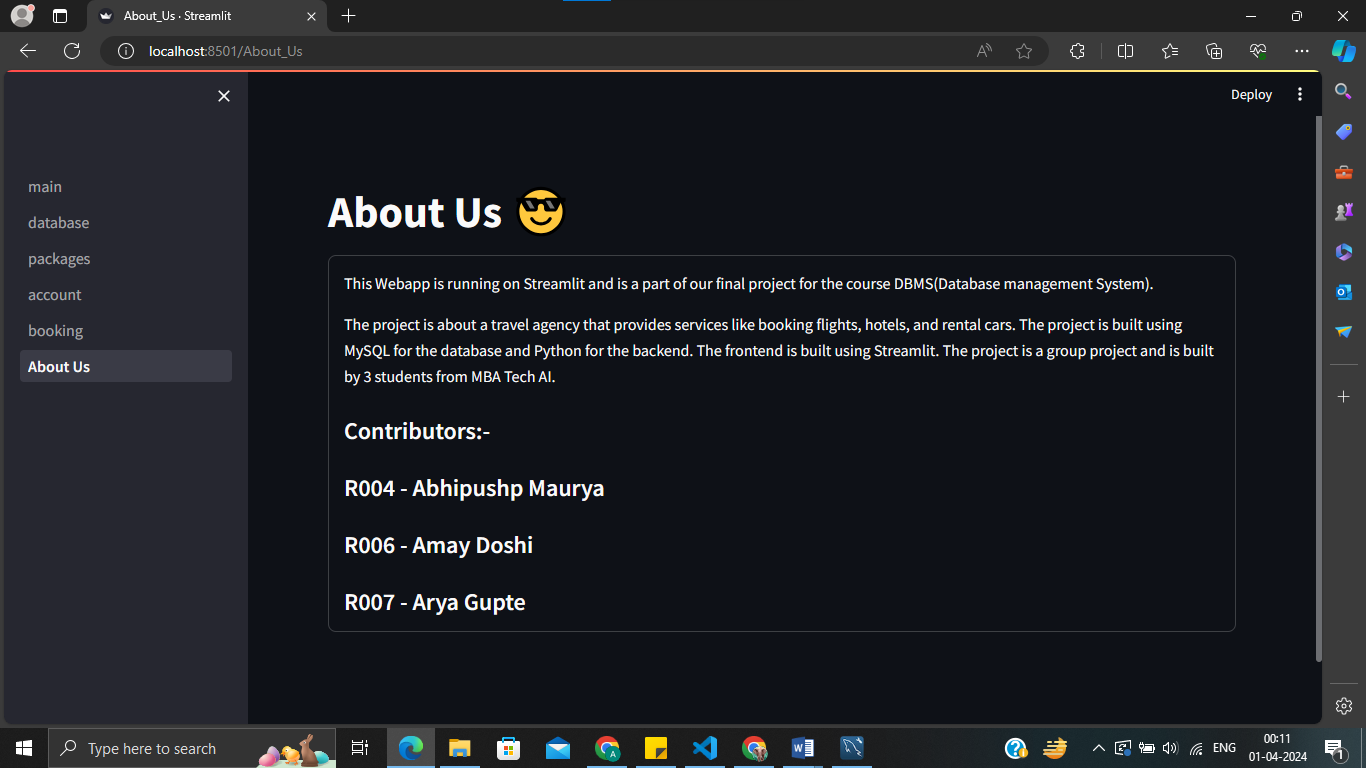












# ***VII. Self -Learning***

In my quest to improve the user experience of my travel database management system, I found Streamlit to be an invaluable tool. Streamlit is a Python framework designed to simplify the process of creating interactive web applications. Its ease of use and powerful capabilities make it an excellent choice for developers looking to build intuitive and visually appealing interfaces.

I started my Streamlit learning journey by looking through the online lessons and documentation. These publications gave me an excellent foundation by outlining the main ideas and characteristics of Streamlit. I gained knowledge on how Streamlit apps are put together, including how to manage user interactions, build layouts, and arrange code.

Streamlit's vast selection of widgets, which let programmers add interactive components like text inputs, dropdown menus, and sliders to their apps, is one of its most intriguing features. I learned more about how to use these widgets to make dynamic and responsive user interfaces by playing around with them.

In addition to widgets, Streamlit offers robust capabilities for displaying and visualizing data. I learned how to load data into my applications, manipulate it using Python libraries such as Pandas, and present it to users in various formats, including tables, charts, and maps. This allowed me to create informative and engaging interfaces that enable users to explore and analyze travel-related data with ease.

Streamlit's ease of use and adaptability have consistently impressed me along my learning process. I could create a basic prototype or a sophisticated application, and Streamlit gave me the freedom and resources I needed to make my ideas a reality. I was able to improve the user experience of my travel database management system and make it a more engaging and interactive platform for users to interact with by utilising Streamlit's capabilities.

Equipped with my acquired expertise, I skillfully combined Streamlit with my SQL database. Using Python modules such as mysql.connector, I created a database connection and ran SQL queries given by the user. Users of the Streamlit app might enter their questions in text fields or forms. The output, which may be tabular data or JSON-like, was dynamically displayed. This integration improved my project's overall functionality while streamlining the querying process.

As I conclude this self-learning phase, I recognize that there’s more to explore. I plan to delve deeper into Streamlit’s advanced features, theming options, and deployment strategies. Additionally, I appreciate how this experience broadened my toolkit as a data enthusiast. Streamlit’s simplicity and versatility have left me eager to tackle future projects with even greater confidence.

# ***VIII. Learning from the Project***

Our team gained valuable insights and lessons that enriched our learning experience:

1. **Hands-on Application:** Working on a real-world project allowed us to apply theoretical concepts and programming skills in a practical setting. This hands-on experience deepened our understanding of Python, SQL, and web development frameworks like Streamlit.
2. **Teamwork and Collaboration:** Collaborating on a team project strengthened our teamwork and communication skills. We learned to effectively communicate ideas, delegate tasks, and coordinate efforts to achieve common goals.
3. **Problem-solving:** Overcoming challenges encountered during the project enhanced our problem-solving abilities. We learned to approach problems systematically, identify root causes, and implement effective solutions through collaborative brainstorming and experimentation.
4. **Continuous Learning:** The project exposed us to new technologies and techniques, fostering a mindset of continuous learning. We embraced opportunities to explore Streamlit, SQL integration, and other tools, expanding our skill set and staying updated with industry trends.
5. **User-Centric Design:** Prioritizing user experience and feedback helped us refine the project and tailor features to meet user needs. Understanding user requirements and incorporating user-centric design principles were essential for creating a successful application.
6. **Project Management:** Managing tasks, timelines, and resources effectively was crucial for project success. We learned to prioritize tasks, adapt to changing requirements, and enhancing our project management skills.

All in all, the project was a worthwhile educational opportunity that gave me useful skills, approaches to solving problems, and understanding of software development procedures. It reaffirmed how crucial practical experience, clear communication, and ongoing development are to achieving professional objectives in the technology sector.

# ***IX. Challenges Faced***

we encountered several challenges that tested our collective problem-solving abilities and teamwork:

1. **Integration Complexity:** Integrating Streamlit with SQL database functionality posed initial challenges, requiring careful coordination and collaboration among team members to establish a secure and efficient connection, handle user inputs, and execute queries while ensuring data integrity and security.
2. **Handling User Inputs:** Implementing robust mechanisms to handle user inputs and validate queries.
3. **User Interface Design:** Designing an intuitive and user-friendly interface that effectively communicated the application's functionality and allowed users to interact seamlessly with the database required collaborative brainstorming sessions, feedback gathering, and iterative design improvements.
4. **Error Handling:** Implementing robust error handling mechanisms to gracefully handle unexpected errors and provide informative feedback to users involved collaborative efforts in identifying potential error scenarios, designing appropriate error messages, and testing error-handling functionality.
5. **Data Fetching:** most of the data was not in the form we needed it so had to make changes in the database to solve the redundancies.

# ***X. Conclusion***

To sum up, our team's creation of the travel database management system and its connection with SQL and Streamlit was a life-changing educational opportunity. Working together, addressing problems, and never stopping to learn, we overcome obstacles to produce a workable result.

We honed our programming abilities and gained insights into web development, database administration, and user interface design by applying theoretical knowledge to real-world settings. Together, we gave an example of the value of cooperation, clear communication, and common problem-solving techniques.

Our technical capabilities were improved by this project, but it also improved our teamwork, communication, and project management abilities. In the future, we can't wait to advance our knowledge, use our abilities on fresh initiatives, and contribute significantly to the technological industry. We have a strong foundation and a culture of growth, creativity, and cooperation, so we're ready for any difficulties or possibilities that lie ahead.