TABLE OF CONTENTS

Chapter 1 : Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for the project

The Problem: In this case, the problem involves developing a database system called "Cloud Wings Database Air Management System" for managing air travel operations. The system should facilitate booking flights (mountain, international, domestic), tour packages, and private jet services. It should also include features for customer feedback, social media integration, and secure payment processing.

Identification of Entities and Relationships:

a. Entities:

i. Airlines

ii. Flights

iii. Customers

iv. Tickets

v. Bills Payment

vi. Social Media Platform

b. Relationships:

i. Airlines operate Flights

ii. Customers book Tickets

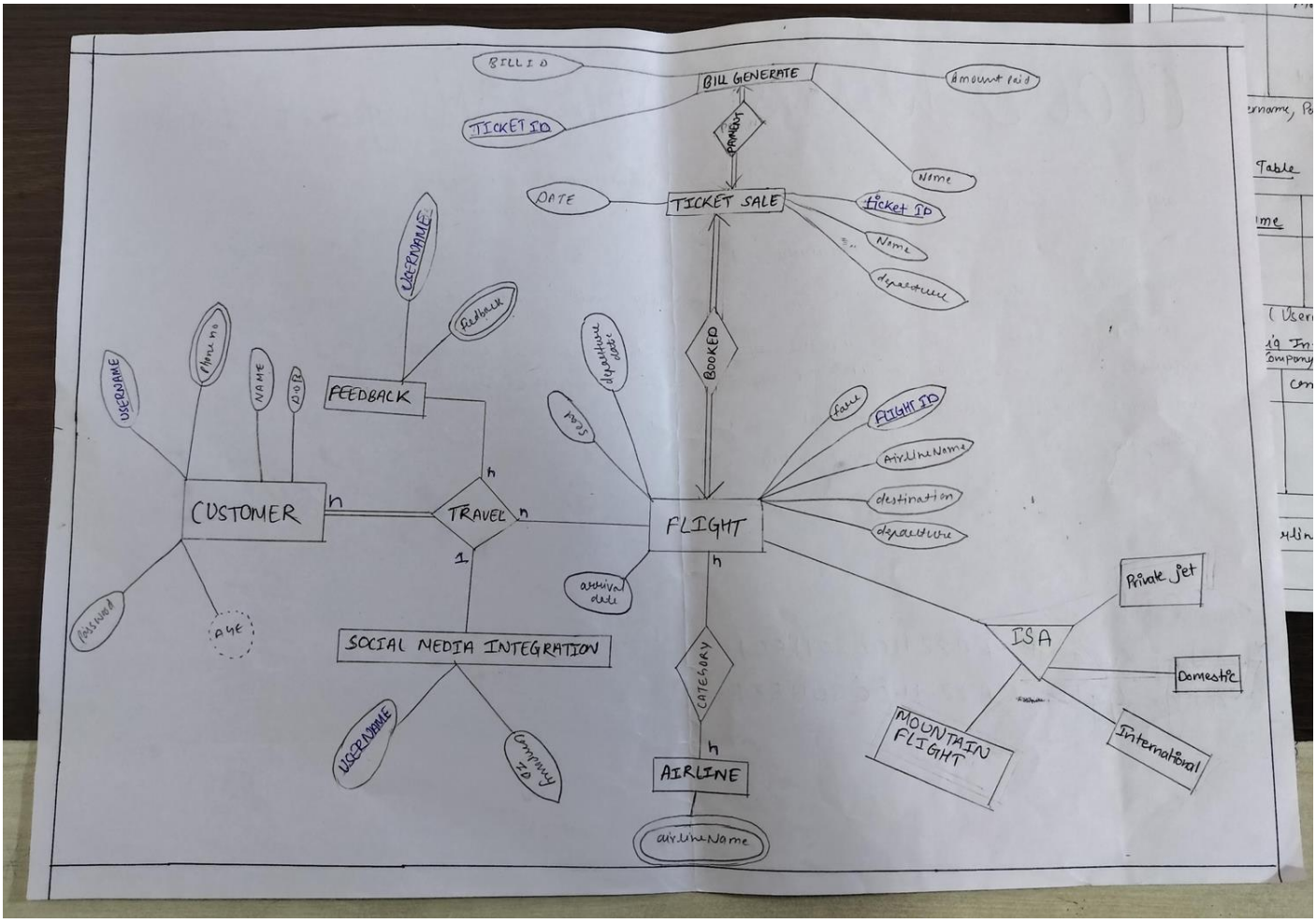
iii. Customers make Bills Payment

iv. Flights have Tickets

v. Understanding Customers provide Feedback

vi. Social Media Platform integrates with the system

Er diagram :



Chapter 2: Design of Relational Schemas, Creation of Database Tables for the project.

We have created a database named “Cloud_wings” for the project and it is likely stores data related to our airline operation, the table contain inside the databases are :

```
mysql> show tables;
+-----------------------+
| Tables_in_cloud_wings |
+-----------------------+
| admin                 |
| bill                  |
| bill_summary          |
| customer              |
| customer_bill_summary |
| customer_feedback     |
| feedback              |
| flight                |
| flight_airline_combo  |
| flight_archive        |
| flight_info           |
| phonenumber           |
| socialmedia           |
| ticket_sale           |
+-----------------------+
14 rows in set (0.05 sec)
```

1.ADMIN:

```
mysql> desc admin;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| A_id     | int         | NO   | PRI | NULL    | auto_increment |
| username | varchar(50) | NO   | UNI | NULL    |                |
| password | varchar(50) | NO   |     | NULL    |                |
| role     | varchar(20) | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.05 sec)
```

2.CUSTOMER:

```
mysql> desc customer;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| user_id       | int          | NO   | PRI | NULL    | auto_increment |
| username      | varchar(200) | NO   | UNI | NULL    |                |
| password      | varchar(200) | NO   |     | NULL    |                |
| name          | varchar(200) | NO   |     | NULL    |                |
| Date_of_Birth | date         | NO   |     | NULL    |                |
| address       | varchar(200) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.07 sec)
```

3.FEEDBACK:

```
mysql> desc feedback;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| description | varchar(100) | NO   |     | NULL    |                |
| username    | varchar(100) | NO   | MUL | NULL    |                |
| feedbackno  | int          | NO   | PRI | NULL    | auto_increment |
+-------------+--------------+------+-----+---------+----------------+
3 rows in set (0.05 sec)
```

4.FLIGHT:

```
mysql> desc flight;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| flight_id      | int          | NO   | PRI | NULL    | auto_increment |
| airlines       | varchar(100) | NO   |     | NULL    |                |
| departure      | varchar(100) | NO   |     | NULL    |                |
| destination    | varchar(100) | NO   |     | NULL    |                |
| departure_time | datetime     | YES  |     | NULL    |                |
| arrival_time   | datetime     | YES  |     | NULL    |                |
| price          | float        | NO   |     | NULL    |                |
| capacity       | int          | NO   |     | NULL    |                |
| flight_type    | varchar(200) | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

5.FLIGHT_AIRLINE_COMBO:

```
mysql> desc flight_airline_combo;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| flight_id | int          | NO   | PRI | NULL    |       |
| airlines  | varchar(100) | NO   | MUL | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

6.PHONE_NUMBER:

```
mysql> desc phonenumber;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| user_id      | int          | NO   | MUL | NULL    |       |
| phone_number | varchar(200) | NO   |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

7.SOCIAL MEDIA:

```
mysql> desc socialmedia;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| socialmedia_id | int         | NO   | PRI | NULL    | auto_increment |
| username       | varchar(20) | NO   | MUL | NULL    |                |
| airlines       | varchar(20) | NO   | MUL | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```

8.**BILL**:

```
+--------------------+---------------+------+-----+---------+----------------+
| Field              | Type          | Null | Key | Default | Extra          |
+--------------------+---------------+------+-----+---------+----------------+
| bill_id            | int           | NO   | PRI | NULL    | auto_increment |
| customer_id        | int           | YES  | MUL | NULL    |                |
| customer_name      | varchar(200)  | YES  |     | NULL    |                |
| flight_id          | int           | YES  | MUL | NULL    |                |
| flight_date        | datetime      | YES  |     | NULL    |                |
| departure_airline  | varchar(100)  | YES  |     | NULL    |                |
| departure_location | varchar(100)  | YES  |     | NULL    |                |
| arrival_location   | varchar(100)  | YES  |     | NULL    |                |
| amount_paid        | decimal(10,2) | YES  |     | NULL    |                |
| bill_date          | datetime      | YES  |     | NULL    |                |
+--------------------+---------------+------+-----+---------+----------------+
10 rows in set (2.81 sec)
```

9.TICKET_SALE:

```
mysql> desc ticket_sale;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| ticket_id          | int          | NO   | PRI | NULL    | auto_increment |
| customer_user_id   | int          | YES  | MUL | NULL    |                |
| customer_name      | varchar(200) | YES  |     | NULL    |                |
| departure_airline  | varchar(100) | YES  |     | NULL    |                |
| departure_location | varchar(100) | YES  |     | NULL    |                |
| departure_time     | datetime     | YES  |     | NULL    |                |
| flight_type        | varchar(200) | YES  |     | NULL    |                |
| amount_paid        | float        | YES  |     | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
8 rows in set (0.16 sec)
```

Chapter 3: Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors.

TRIGGERS:

BILL SUMMARY :

```
mysql> CREATE TRIGGER update_bill_summary AFTER INSERT ON bill
    -> FOR EACH ROW
    -> BEGIN
    ->   DECLARE summary_exists INT;
    ->
    ->   SELECT COUNT(*) INTO summary_exists FROM bill_summary;
    ->
    ->   IF summary_exists = 0 THEN
    ->     INSERT INTO bill_summary (total_sales) VALUES
(NEW.amount_paid);
    ->   ELSE
    ->     UPDATE bill_summary
    ->     SET total_sales = total_sales + NEW.amount_paid;
    ->   END IF;
    -> END;
    -> //
```

```
mysql> select * from bill;
+---------+-------------+---------------+-----------+---------------------+-------------------+--------------------+-------------------+-------------+---------------------+
| bill_id | customer_id | customer_name | flight_id | flight_date         | departure_airline | departure_location | arrival_location  | amount_paid | bill_date           |
+---------+-------------+---------------+-----------+---------------------+-------------------+--------------------+-------------------+-------------+---------------------+
|      54 |         104 | NULL          |       201 | 2024-04-01 00:00:00 | Cloud Wings       | New York           | Los Angeles       |      300.00 | 2024-03-31 00:00:00 |
|      55 |         105 | NULL          |       203 | 2024-04-10 00:00:00 | Emirates          | Dubai              | London            |      500.00 | 2024-03-31 00:00:00 |
|      56 |         104 | NULL          |       202 | 2024-04-01 00:00:00 | Cloud Wings       | New York           | Los Angeles       |      300.00 | 2024-03-31 00:00:00 |
+---------+-------------+---------------+-----------+---------------------+-------------------+--------------------+-------------------+-------------+---------------------+
3 rows in set (0.00 sec)

mysql> select * from bill_summary;
+------------+-------------+
| summary_id | total_sales |
+------------+-------------+
|          1 |     1100.00 |
+------------+-------------+
1 row in set (0.00 sec)
```

2.FLIGHT_SYSTEM:

```
mysql>
mysql> CREATE TRIGGER archive_flight_update
    -> AFTER UPDATE ON flight
    -> FOR EACH ROW
    -> BEGIN
    ->   IF OLD.flight_id IS NOT NULL THEN
    ->     INSERT INTO flight_archive (flight_id, airlines, departure, destination,
departure_time, arrival_time, price, capacity, flight_type)
    ->     VALUES (OLD.flight_id, OLD.airlines, OLD.departure, OLD.destination,
OLD.departure_time, OLD.arrival_time, OLD.price, OLD.capacity, OLD.flight_type);
    ->   END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.18 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> CREATE TABLE flight_archive (
    ->   archive_id INT PRIMARY KEY AUTO_INCREMENT,
    ->   flight_id INT NOT NULL,
    ->   airlines VARCHAR(100) NOT NULL,
    ->   departure VARCHAR(100) NOT NULL,
    ->   destination VARCHAR(100) NOT NULL,
    ->   departure_time DATETIME NOT NULL,
    ->   arrival_time DATETIME NOT NULL,
    ->   price DECIMAL(10,2) NOT NULL,
    ->   capacity INT NOT NULL,
    ->   flight_type VARCHAR(200) NOT NULL,
    ->   archived_at DATETIME DEFAULT CURRENT_TIMESTAMP,  -- Captures
timestamp when archived
    ->   FOREIGN KEY (flight_id) REFERENCES flight(flight_id)  -- References
original flight_id
    -> );
```

```
mysql> SELECT * FROM FLIGHT;
+-----------+-------------+-----------+-------------+---------------------+---------------------+-------+----------+---------------+
| flight_id | airlines    | departure | destination | departure_time      | arrival_time        | price | capacity | flight_type   |
+-----------+-------------+-----------+-------------+---------------------+---------------------+-------+----------+---------------+
|       201 | Cloud Wings | New York  | Los Angeles | 2024-04-01 08:00:00 | 2024-04-01 11:00:00 |   300 |      200 | Domestic      |
|       202 | Cloud Wings | New York  | Los Angeles | 2024-04-01 08:00:00 | 2024-04-01 11:00:00 |   300 |      200 | Domestic      |
|       203 | ktm         | Dubai     | London      | 2024-04-10 14:00:00 | 2024-04-11 18:00:00 |   500 |      150 | International |
+-----------+-------------+-----------+-------------+---------------------+---------------------+-------+----------+---------------+
3 rows in set (0.40 sec)

mysql> SELECT * FROM FLIGHT_ARCHIVE;
+------------+-----------+----------+-----------+-------------+---------------------+---------------------+--------+----------+---------------+---------------------+
| archive_id | flight_id | airlines | departure | destination | departure_time      | arrival_time        | price  | capacity | flight_type   | archived_at         |
+------------+-----------+----------+-----------+-------------+---------------------+---------------------+--------+----------+---------------+---------------------+
|          4 |       203 | Emirates | Dubai     | London      | 2024-04-10 14:00:00 | 2024-04-11 18:00:00 | 500.00 |      150 | International | 2024-03-31 04:24:33 |
+------------+-----------+----------+-----------+-------------+---------------------+---------------------+--------+----------+---------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

CURSORS:

1. **PHONE_NUMBER:**

```
mysql> call FindCustomersWithoutPhone();
+-----------------------------------------------+
| Message                                       |
+-----------------------------------------------+
| Customer John Doe has no phone number provided. |
+-----------------------------------------------+
1 row in set (0.00 sec)

+-----------------------------------------------+
| Message                                       |
+-----------------------------------------------+
| Customer Jane Smith has no phone number provided. |
+-----------------------------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

2. **TOTAL_SALES _OF_EACH_CUSTOME**

```
mysql> CALL CalculateCustomerTotalSales();
+-------------+---------------+-------------+
| customer_id | customer_name | total_sales |
+-------------+---------------+-------------+
|         104 | John Doe      |      300.00 |
|         105 | Jane Smith    |        NULL |
+-------------+---------------+-------------+
2 rows in set (1.00 sec)

Query OK, 0 rows affected (1.04 sec)
```

VIEWS AND JOINS:

1.FLIGHT_INFO:

```
mysql> CREATE VIEW flight_info AS
    -> SELECT f.flight_id, f.airlines, f.departure, f.destination,
    ->        f.departure_time, f.arrival_time, f.price, f.capacity, f.flight_type
    -> FROM flight f
    -> JOIN flight_airline_combo fac ON f.flight_id = fac.flight_id
    -> LEFT JOIN flight_archive fa ON f.flight_id = fa.flight_id;
```

```
mysql> desc flight_info;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| flight_id      | int          | NO   |     | 0       |       |
| airlines       | varchar(100) | NO   |     | NULL    |       |
| departure      | varchar(100) | NO   |     | NULL    |       |
| destination    | varchar(100) | NO   |     | NULL    |       |
| departure_time | datetime     | YES  |     | NULL    |       |
| arrival_time   | datetime     | YES  |     | NULL    |       |
| price          | float        | NO   |     | NULL    |       |
| capacity       | int          | NO   |     | NULL    |       |
| flight_type    | varchar(200) | NO   |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

2.CUSTOMER_FEEDBACK:

```
mysql> CREATE VIEW customer_feedback AS
    -> SELECT f.feedbackno, f.description, c.username AS customer_username
    -> FROM feedback f
    -> JOIN customer c ON f.username = c.username;
```

```
mysql> desc customer_feedback;
+-------------------+--------------+------+-----+---------+-------+
| Field             | Type         | Null | Key | Default | Extra |
+-------------------+--------------+------+-----+---------+-------+
| feedbackno        | int          | NO   |     | 0       |       |
| description       | varchar(100) | NO   |     | NULL    |       |
| customer_username | varchar(200) | NO   |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
3 rows in set (0.13 sec)
```

3.CUSTOMER_BILL_SUMMARY:

```
mysql> CREATE VIEW customer_bill_summary AS
    -> SELECT b.customer_id, c.name AS customer_name, b.bill_id, b.flight_id,
    ->        b.flight_date, b.amount_paid
    -> FROM bill b
    -> JOIN customer c ON b.customer_id = c.user_id;
Query OK, 0 rows affected (0.16 sec)
```

```
mysql> desc customer_bill_summary;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| customer_id   | int           | YES  |     | NULL    |       |
| customer_name | varchar(200)  | NO   |     | NULL    |       |
| bill_id       | int           | NO   |     | 0       |       |
| flight_id     | int           | YES  |     | NULL    |       |
| flight_date   | datetime      | YES  |     | NULL    |       |
| amount_paid   | decimal(10,2) | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
6 rows in set (0.27 sec)
```

CHAPTER 4: Analyzing the pitfalls, identifying the dependencies, and applying normalizations

1.NORMALIZATION OF FLIGHT TABLE :

In the flight table, we normalized the data by introducing a separate table for flight types (flight_type). This normalization was necessary to address the redundancy present in the flight_type column of the flight table.

Creating the table Flight_type

```
mysql> CREATE TABLE flight_type (
    ->     flight_type_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     type_name VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (1.34 sec)

mysql> ALTER TABLE flight
    -> ADD COLUMN flight_type_id INT,
    -> ADD CONSTRAINT fk_flight_type
    ->     FOREIGN KEY (flight_type_id)
    ->     REFERENCES flight_type(flight_type_id);
Query OK, 0 rows affected (1.14 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc flight;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| flight_id      | int          | NO   | PRI | NULL    | auto_increment |
| airlines       | varchar(100) | NO   |     | NULL    |                |
| departure      | varchar(100) | NO   |     | NULL    |                |
| destination    | varchar(100) | NO   |     | NULL    |                |
| departure_time | datetime     | YES  |     | NULL    |                |
| arrival_time   | datetime     | YES  |     | NULL    |                |
| price          | float        | NO   |     | NULL    |                |
| capacity       | int          | NO   |     | NULL    |                |
| flight_type_id | int          | YES  | MUL | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

mysql> desc flight_type;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| flight_type_id | int         | NO   | PRI | NULL    | auto_increment |
| type_name      | varchar(50) | NO   |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

In normalizing the flight table, we opted to create a separate flight_type table to address redundancy and enhance data integrity. Storing flight types directly within the flight table could lead to duplicated information if the same flight types are repeated across multiple flight records. By centralizing flight type information in its own table, we ensure that each type is stored only once, minimizing redundancy and conserving storage space. Additionally, the flight_type table allows us to enforce data integrity by using foreign key constraints to link flight records to their corresponding types. This ensures that only valid flight types can be associated with flights, preventing inconsistencies and maintaining data accuracy. Furthermore, separating flight types into their own table simplifies maintenance and updates, as changes to the list of types can be made centrally without affecting individual flight records.

2. Drop the customer_name in bill and ticket_sale table :

To normalize the Bill table and remove the dependency on customer_name, we need to remove the customer_name column

After the Dropping customer_name from bill

```
mysql> desc bill;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| bill_id            | int          | NO   | PRI | NULL    | auto_increment |
| customer_id        | int          | YES  | MUL | NULL    |                |
| flight_id          | int          | YES  | MUL | NULL    |                |
| flight_date        | datetime     | YES  |     | NULL    |                |
| departure_airline  | varchar(100) | YES  |     | NULL    |                |
| departure_location | varchar(100) | YES  |     | NULL    |                |
| arrival_location   | varchar(100) | YES  |     | NULL    |                |
| amount_paid        | float        | YES  |     | NULL    |                |
| bill_date          | datetime     | YES  |     | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.68 sec)
```

3. Drop Customer_name from ticket_sale

To normalize the Bill table and remove the dependency on customer_name, we need to remove the customer_name column

After dropping customer_name from ticket_sale

```
mysql> desc ticket_sale;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| ticket_id          | int          | NO   | PRI | NULL    | auto_increment |
| customer_user_id   | int          | YES  |     | NULL    |                |
| departure_airline  | varchar(100) | YES  |     | NULL    |                |
| departure_location | varchar(100) | YES  |     | NULL    |                |
| departure_time     | datetime     | YES  |     | NULL    |                |
| flight_type        | varchar(50)  | YES  |     | NULL    |                |
| amount_paid        | float        | YES  |     | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
7 rows in set (0.14 sec)
```

4. Updating the Trigger table :

The trigger "archive_flight_update" and table "flight_archive" ensure data integrity by preserving old flight data before updates. They establish a dependency between archived records and their original flights via foreign key constraints, ensuring relational integrity. This proactive approach safeguards against data loss or corruption and maintains a reliable historical record of flight changes.

```
mysql> desc flight_archive;
+----------------+--------------+------+-----+-------------------+-------------------+
| Field          | Type         | Null | Key | Default           | Extra             |
+----------------+--------------+------+-----+-------------------+-------------------+
| archive_id     | int          | NO   | PRI | NULL              | auto_increment    |
| flight_id      | int          | NO   | MUL | NULL              |                   |
| airlines       | varchar(100) | NO   |     | NULL              |                   |
| departure      | varchar(100) | NO   |     | NULL              |                   |
| destination    | varchar(100) | NO   |     | NULL              |                   |
| departure_time | datetime     | YES  |     | NULL              |                   |
| arrival_time   | datetime     | YES  |     | NULL              |                   |
| price          | float        | NO   |     | NULL              |                   |
| capacity       | int          | NO   |     | NULL              |                   |
| flight_type_id | int          | YES  |     | NULL              |                   |
| archived_at    | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+----------------+--------------+------+-----+-------------------+-------------------+
11 rows in set (0.01 sec)
```

```
mysql> -- Set delimiter to define the trigger
mysql> DELIMITER //
mysql>
mysql> -- Create trigger to archive flight updates
mysql> CREATE TRIGGER archive_flight_update
    -> AFTER UPDATE ON flight
    -> FOR EACH ROW
    -> BEGIN
    ->   -- Check if old flight_id is not null
    ->   IF OLD.flight_id IS NOT NULL THEN
    ->     -- Insert old flight data into flight_archive table
    ->     INSERT INTO flight_archive (flight_id, airlines, departure, destination,
departure_time, arrival_time, price, capacity, flight_type_id)
    ->     VALUES (OLD.flight_id, OLD.airlines, OLD.departure, OLD.destination,
OLD.departure_time, OLD.arrival_time, OLD.price, OLD.capacity,
OLD.flight_type_id);
    ->   END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.18 sec)
```

CHAPTER 5:

Implementing concurrency control and recovery mechanisms is crucial for ensuring the consistency, durability, and reliability of a database system, especially in multi-user environments.

Here's an overview of how these mechanisms can be implemented

1. **Concurrency Control**:
   - **Lock-Based Concurrency Control**: Implement locking mechanisms to control access to data and prevent concurrent transactions from interfering with each other. Use locks such as shared locks and exclusive locks to control read and write operations on data items.
   - **Two-Phase Locking (2PL)**: Implement a 2PL protocol to ensure serializability of transactions. Acquire locks on data items before performing any operation and release them only after the transaction completes.
   - **Timestamp-Based Concurrency Control**: Assign timestamps to transactions and data items to determine their order of execution. Use techniques such as Timestamp Ordering Protocol (TO) or Thomas Write Rule to ensure serializability.
   - **Multiversion Concurrency Control (MVCC)**: Maintain multiple versions of data items to allow concurrent read and write operations without blocking. Use techniques like snapshot isolation or optimistic concurrency control to manage concurrent access.

2. **Recovery Mechanisms**:

- **Write-Ahead Logging (WAL)**: Implement a WAL mechanism to ensure durability by writing transaction changes to a log before modifying the database. During recovery, replay the log to restore the database to a consistent state.
- **Checkpointing**: Periodically create checkpoints to record the state of the database. During recovery, use checkpoints to reduce the amount of log records that need to be replayed.
- **Transaction Rollback and Commit**: Maintain transaction logs to track the changes made by transactions. During recovery, roll back incomplete transactions and redo committed transactions to restore the database to a consistent state.
- **Shadow Paging**: Implement shadow paging as a recovery mechanism where a shadow copy of the database is maintained. During recovery, switch to the shadow copy to restore the database to a consistent state.

3. **Implementation Considerations**:

- **Isolation Levels**: Implement different isolation levels such as Read Uncommitted, Read Committed, Repeatable Read, and Serializable to control the visibility of changes made by concurrent transactions.
- **Deadlock Detection and Resolution**: Implement mechanisms to detect and resolve deadlocks that may occur due to conflicting lock acquisitions by concurrent transactions.
- **Transaction Management**: Implement transaction management functionality to begin, commit, or rollback transactions and ensure atomicity, consistency,

isolation, and durability (ACID properties).

4. **Testing and Optimization**:
   - Test the concurrency control and recovery mechanisms thoroughly using various scenarios and workload patterns to ensure correctness and performance.
   - Optimize the implementation for efficiency, scalability, and fault tolerance to handle a large number of concurrent transactions and ensure quick recovery in case of failures.

By implementing robust concurrency control and recovery mechanisms, the database system can maintain data consistency, integrity, and availability even in the presence of concurrent transactions and system failures.

Chapter 6: Code for the project

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.Vector;

public class Main extends JFrame {
    private Connection connection;
    private JTable flightTable;
    private DefaultTableModel flightTableModel;
    private JButton bookButton, postButton, searchButton, registerButton;
    private JTextField destinationField, departureField, dateField, usernameField, passwordField,
nameField, dobField, addressField;
    private JTextArea chatTextArea;
    private JScrollPane chatScrollPane;
    private JPanel mainPanel, loginPanel;
    private boolean loggedIn = false;
    private int loggedInUserId;
    private String loggedInUsername;

    public Main() {
        setTitle("Airline Booking App");
        setSize(800, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        // Login Panel
```

```
        loginPanel = new JPanel(new GridLayout(7, 2));
        loginPanel.setBackground(new Color(173, 216, 230)); // Light Blue background
        loginPanel.add(new JLabel("Username:"));
        usernameField = new JTextField();
        loginPanel.add(usernameField);
        loginPanel.add(new JLabel("Password:"));
        passwordField = new JPasswordField();
        loginPanel.add(passwordField);
        loginPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        loginPanel.add(nameField);
        loginPanel.add(new JLabel("Date of Birth (YYYY-MM-DD):"));
        dobField = new JTextField();
        loginPanel.add(dobField);
        loginPanel.add(new JLabel("Address:"));
        addressField = new JTextField();
        loginPanel.add(addressField);
        JButton loginButton = new JButton("Login");
        loginPanel.add(loginButton);
        registerButton = new JButton("Register");
        loginPanel.add(registerButton);

        add(loginPanel);
        setVisible(true);

        // Connect to the database
        try {
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/cloud_airline", "root",
"ADISHMA");
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }

        // Action Listeners for login panel buttons
        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = passwordField.getText();
                if (validateLogin(username, password)) {
                    loggedIn = true;
                    loggedInUserId = getUserId(username);
                    loggedInUsername = username;
                    JOptionPane.showMessageDialog(Main.this, "Login successful!");
                    remove(loginPanel);
                    initializeMainPanel();
                } else {
                    JOptionPane.showMessageDialog(Main.this, "Invalid username or password. Please try
again.");
                }
            }
        });

        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = passwordField.getText();
```

```
            String name = nameField.getText();
            String dob = dobField.getText();
            String address = addressField.getText();
            if (!username.isEmpty() && !password.isEmpty() && !name.isEmpty() && !dob.isEmpty() &&
!address.isEmpty()) {
                if (registerUser(username, password, name, dob, address)) {
                    JOptionPane.showMessageDialog(Main.this, "Registration successful! Please login.");
                } else {
                    JOptionPane.showMessageDialog(Main.this, "Registration failed! Please try again.");
                }
            } else {
                JOptionPane.showMessageDialog(Main.this, "Please enter all fields.");
            }
        }
    });
}

private boolean validateLogin(String username, String password) {
    try {
        String query = "SELECT * FROM customer WHERE username = ? AND password = ?";
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setString(1, username);
        statement.setString(2, password);
        ResultSet resultSet = statement.executeQuery();
        return resultSet.next(); // Return true if a row is found
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
```

```
    }

    private int getUserId(String username) {
        try {
            String query = "SELECT user_id FROM customer WHERE username = ?";
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setString(1, username);
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                return resultSet.getInt("user_id");
            } else {
                return -1;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return -1;
        }
    }

    private boolean registerUser(String username, String password, String name, String dob, String
address) {
        try {
            String query = "INSERT INTO customer (username, password, name, Date_of_Birth, address)
VALUES (?, ?, ?, ?, ?)";
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setString(1, username);
            statement.setString(2, password);
            statement.setString(3, name);
            statement.setString(4, dob);
```

```
        statement.setString(5, address);
        int rowsInserted = statement.executeUpdate();
        return rowsInserted > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

private void initializeMainPanel() {
    // Main Panel
    mainPanel = new JPanel(new BorderLayout());
    mainPanel.setBackground(Color.WHITE); // White background

    // Flight Table
    flightTableModel = new DefaultTableModel();
    flightTableModel.addColumn("Flight ID");
    flightTableModel.addColumn("Airlines");
    flightTableModel.addColumn("Departure");
    flightTableModel.addColumn("Destination");
    flightTableModel.addColumn("Departure Time");
    flightTableModel.addColumn("Arrival Time");
    flightTableModel.addColumn("Price");
    flightTableModel.addColumn("Capacity");
    flightTableModel.addColumn("Flight Type"); // New column for flight type
    flightTable = new JTable(flightTableModel);
    JScrollPane scrollPane = new JScrollPane(flightTable);

    // Search Panel
```

```
JPanel searchPanel = new JPanel(new FlowLayout());
searchPanel.setBackground(new Color(240, 240, 240)); // Light Gray background
destinationField = new JTextField(10);
departureField = new JTextField(10);
dateField = new JTextField(10);
searchButton = new JButton("Search Flights");
searchPanel.add(new JLabel("Destination:"));
searchPanel.add(destinationField);
searchPanel.add(new JLabel("Departure:"));
searchPanel.add(departureField);
searchPanel.add(new JLabel("Date (YYYY-MM-DD):"));
searchPanel.add(dateField);
searchPanel.add(searchButton);

// Chat Panel
JPanel chatPanel = new JPanel(new BorderLayout());
chatPanel.setBackground(new Color(255, 250, 205)); // LemonChiffon background
chatTextArea = new JTextArea(10, 50);
chatScrollPane = new JScrollPane(chatTextArea);
chatPanel.add(new JLabel("Chat Box"), BorderLayout.NORTH);
chatPanel.add(chatScrollPane, BorderLayout.CENTER);

// Buttons Panel
JPanel buttonPanel = new JPanel(new FlowLayout());
buttonPanel.setBackground(new Color(240, 248, 255)); // AliceBlue background
bookButton = new JButton("Book Flight");
postButton = new JButton("Post Message");
buttonPanel.add(bookButton);
buttonPanel.add(postButton);
```

```
// Add components to main panel
mainPanel.add(searchPanel, BorderLayout.NORTH);
mainPanel.add(scrollPane, BorderLayout.CENTER);
mainPanel.add(buttonPanel, BorderLayout.SOUTH);
mainPanel.add(chatPanel, BorderLayout.SOUTH);

add(mainPanel);
setVisible(true);

// Action Listeners
searchButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String destination = destinationField.getText();
        String departure = departureField.getText();
        String date = dateField.getText();
        searchFlights(destination, departure, date);
    }
});

bookButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Handle flight booking
        int selectedRow = flightTable.getSelectedRow();
        if (selectedRow != -1) {
            int flightId = (int) flightTableModel.getValueAt(selectedRow, 0);
            double price = (double) flightTableModel.getValueAt(selectedRow, 6);
```

```
            String departure = (String) flightTableModel.getValueAt(selectedRow, 2);
            String destination = (String) flightTableModel.getValueAt(selectedRow, 3);
            double amountPaid = price; // You can calculate based on additional features
            generateBill(flightId, departure, destination, amountPaid);
        } else {
            JOptionPane.showMessageDialog(Main.this, "Please select a flight to book.");
        }
    }
});

postButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Handle posting message to chat
        String message = JOptionPane.showInputDialog(Main.this, "Enter your message:");
        if (message != null && !message.isEmpty()) {
            postMessage(message);
        }
    }
});

// Add a MouseListener to the flightTable
flightTable.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 1) { // Check if it's a single click
            JTable target = (JTable) e.getSource();
            int row = target.getSelectedRow();
            int column = target.getSelectedColumn();
```

```
            if (row != -1 && column != -1) {
                // Perform actions for generating bill and selling ticket
                int flightId = (int) flightTableModel.getValueAt(row, 0);
                double price = (double) flightTableModel.getValueAt(row, 6);
                String departure = (String) flightTableModel.getValueAt(row, 2);
                String destination = (String) flightTableModel.getValueAt(row, 3);
                double amountPaid = price; // You can calculate based on additional features
                generateBill(flightId, departure, destination, amountPaid);
            }
        }
    }
});
}

private void loadFlights(ResultSet resultSet) throws SQLException {
    flightTableModel.setRowCount(0); // Clear current table data
    while (resultSet.next()) {
        Vector<Object> row = new Vector<>();
        row.add(resultSet.getInt("flight_id"));
        row.add(resultSet.getString("airlines"));
        row.add(resultSet.getString("departure"));
        row.add(resultSet.getString("destination"));
        row.add(resultSet.getTimestamp("departure_time"));
        row.add(resultSet.getTimestamp("arrival_time"));
        row.add(resultSet.getDouble("price"));
        row.add(resultSet.getInt("capacity"));
        row.add(getFlightType(resultSet.getInt("flight_type_id"))); // Add flight type
        flightTableModel.addRow(row);
    }
```

```
    }

    private String getFlightType(int flightTypeId) {
        try {
            String query = "SELECT type_name FROM flight_type WHERE flight_type_id = ?";
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setInt(1, flightTypeId);
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                return resultSet.getString("type_name");
            } else {
                return "Unknown";
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return "Unknown";
        }
    }

    private void searchFlights(String destination, String departure, String date) {
        try {
            String query = "SELECT * FROM flight WHERE destination = ? AND departure = ? AND
DATE(departure_time) = ?";
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setString(1, destination);
            statement.setString(2, departure);
            statement.setString(3, date);
            ResultSet resultSet = statement.executeQuery();
```

```
            // Check if there are any flights found
            if (!resultSet.isBeforeFirst()) {
                // No flights found
                JOptionPane.showMessageDialog(this, "No flights found for the given criteria.");
            } else {
                // Flights found, load them into the table
                loadFlights(resultSet);
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "An error occurred while searching for flights.");
        }
    }

    private void generateBill(int flightId, String departure, String destination, double amountPaid) {
        try {
            // Insert bill details into the 'bill' table
            String insertBillQuery = "INSERT INTO bill (customer_id, flight_id, flight_date, departure_airline,
departure_location, arrival_location, amount_paid, bill_date) VALUES (?, ?, NOW(), ?, ?, ?, ?, NOW())";
            PreparedStatement billStatement = connection.prepareStatement(insertBillQuery,
Statement.RETURN_GENERATED_KEYS);
            billStatement.setInt(1, loggedInUserId);
            billStatement.setInt(2, flightId);
            billStatement.setString(3, "Airlines"); // You can set the airline name here
            billStatement.setString(4, departure);
            billStatement.setString(5, destination);
            billStatement.setDouble(6, amountPaid);
            int rowsInserted = billStatement.executeUpdate();
```

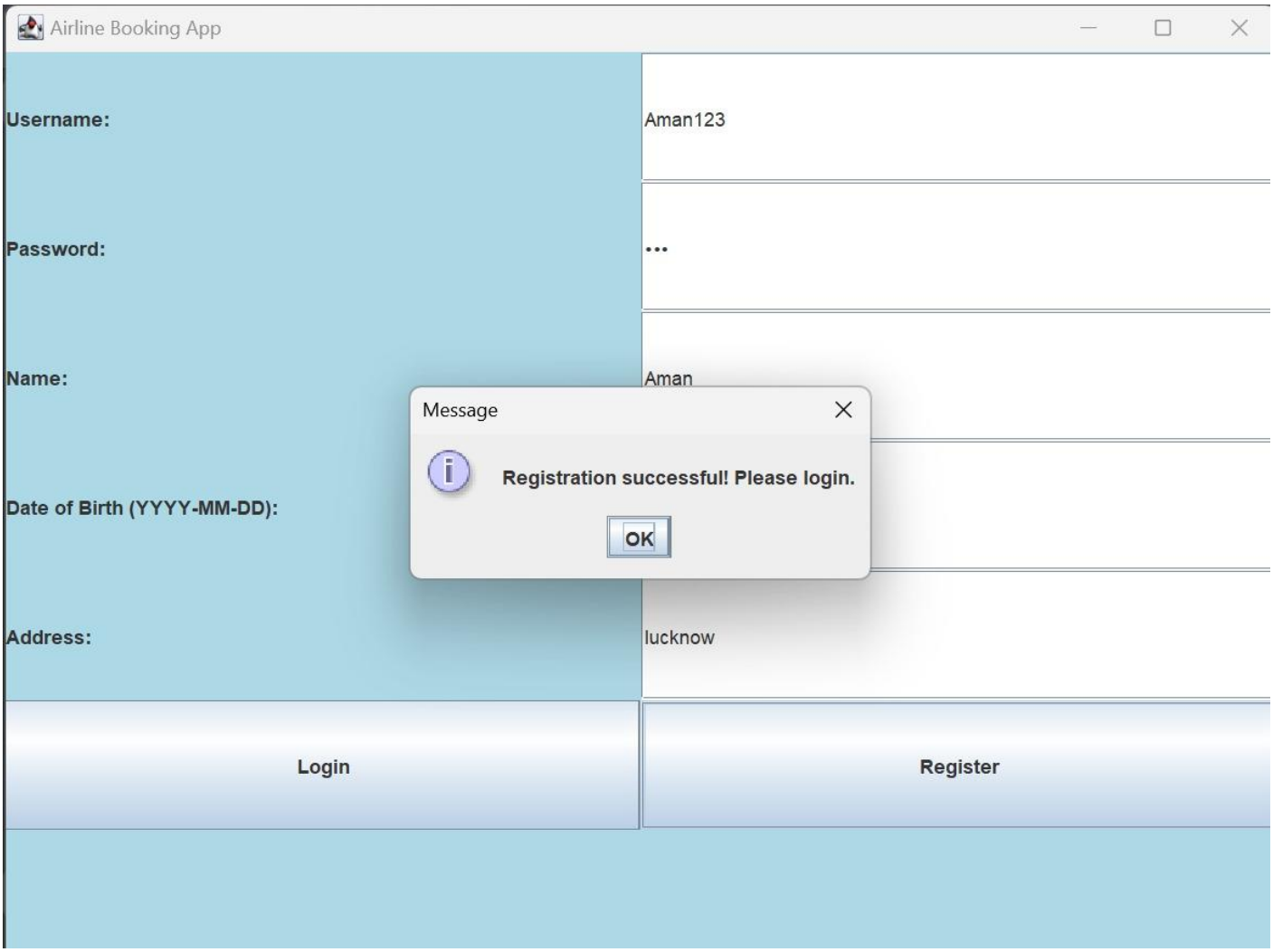
```
        // Get the generated bill ID
        int billId = -1;
        ResultSet generatedKeys = billStatement.getGeneratedKeys();
        if (generatedKeys.next()) {
            billId = generatedKeys.getInt(1);
        }

        // If the bill insertion is successful
        if (rowsInserted > 0 && billId != -1) {
            // Insert ticket sale details into the 'ticket_sale' table
            String insertTicketQuery = "INSERT INTO ticket_sale (customer_user_id, departure_airline,
departure_location, departure_time, flight_type, amount_paid) VALUES (?, ?, ?, NOW(), ?, ?)";
            PreparedStatement ticketStatement = connection.prepareStatement(insertTicketQuery);
            ticketStatement.setInt(1, loggedInUserId);
            ticketStatement.setString(2, "Airlines"); // You can set the airline name here
            ticketStatement.setString(3, departure);
            ticketStatement.setString(4, "Domestic"); // You can set the flight type here
            ticketStatement.setDouble(5, amountPaid);
            int ticketRowsInserted = ticketStatement.executeUpdate();

            // If ticket sale insertion is successful
            if (ticketRowsInserted > 0) {
                JOptionPane.showMessageDialog(this, "Flight booked!\nBill ID: " + billId +
                        "\nFrom: " + departure + "\nTo: " + destination + "\nAmount Paid: $" + amountPaid);
            } else {
                JOptionPane.showMessageDialog(this, "Failed to book the flight. Please try again.");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Failed to generate the bill. Please try again.");
```

```
        }
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "An error occurred while generating the bill.");
    }
}

private void postMessage(String message) {
    // Implement posting message to chat logic here
    // You can append the message to the chatTextArea and store it in the database
    // Also append the username before the message
    chatTextArea.append(loggedInUsername + ": " + message + "\n");
    storeFeedback(loggedInUserId, message); // Store the feedback in the database
}

private void storeFeedback(int userId, String message) {
    try {
        String query = "INSERT INTO feedback (user_id, message, username) VALUES (?, ?, ?)";
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setInt(1, userId);
        statement.setString(2, message);
        statement.setString(3, loggedInUsername); // Add the username
        int rowsInserted = statement.executeUpdate();
        if (rowsInserted > 0) {
            System.out.println("Feedback stored successfully.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
```

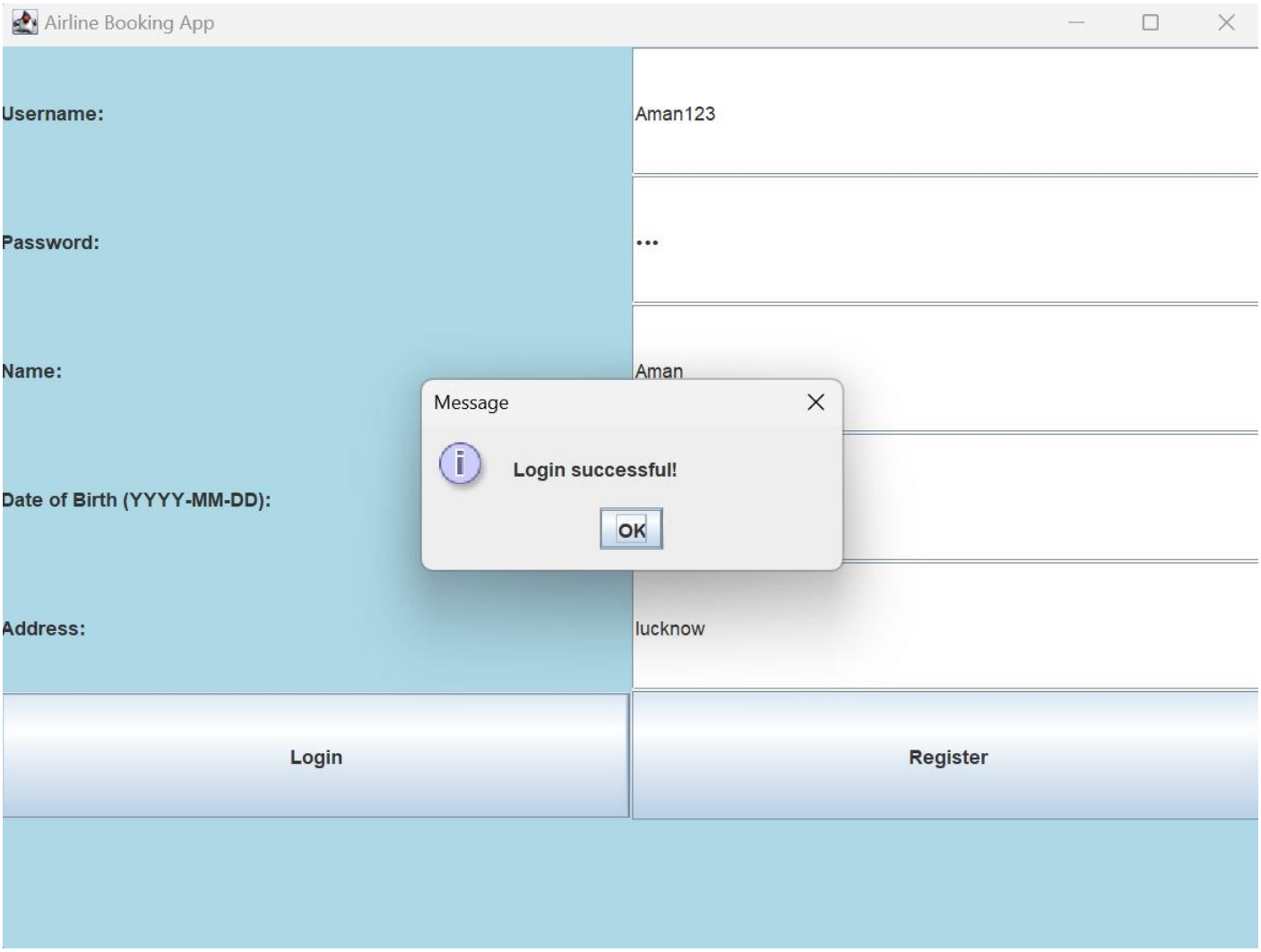
```
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Main();
            }
        });
    }
}
```

Chapter 7: Screenshot of the project Output

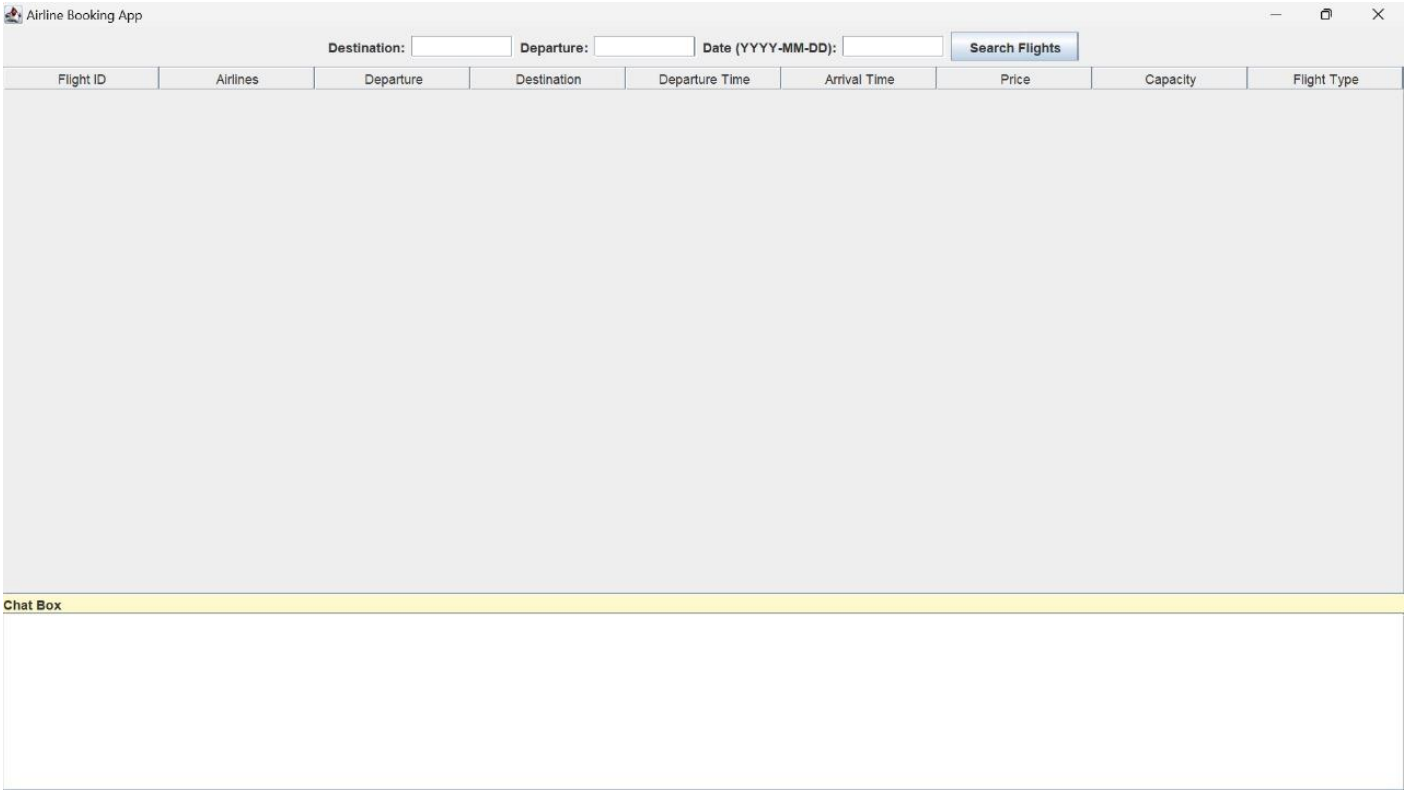
First step of registration:



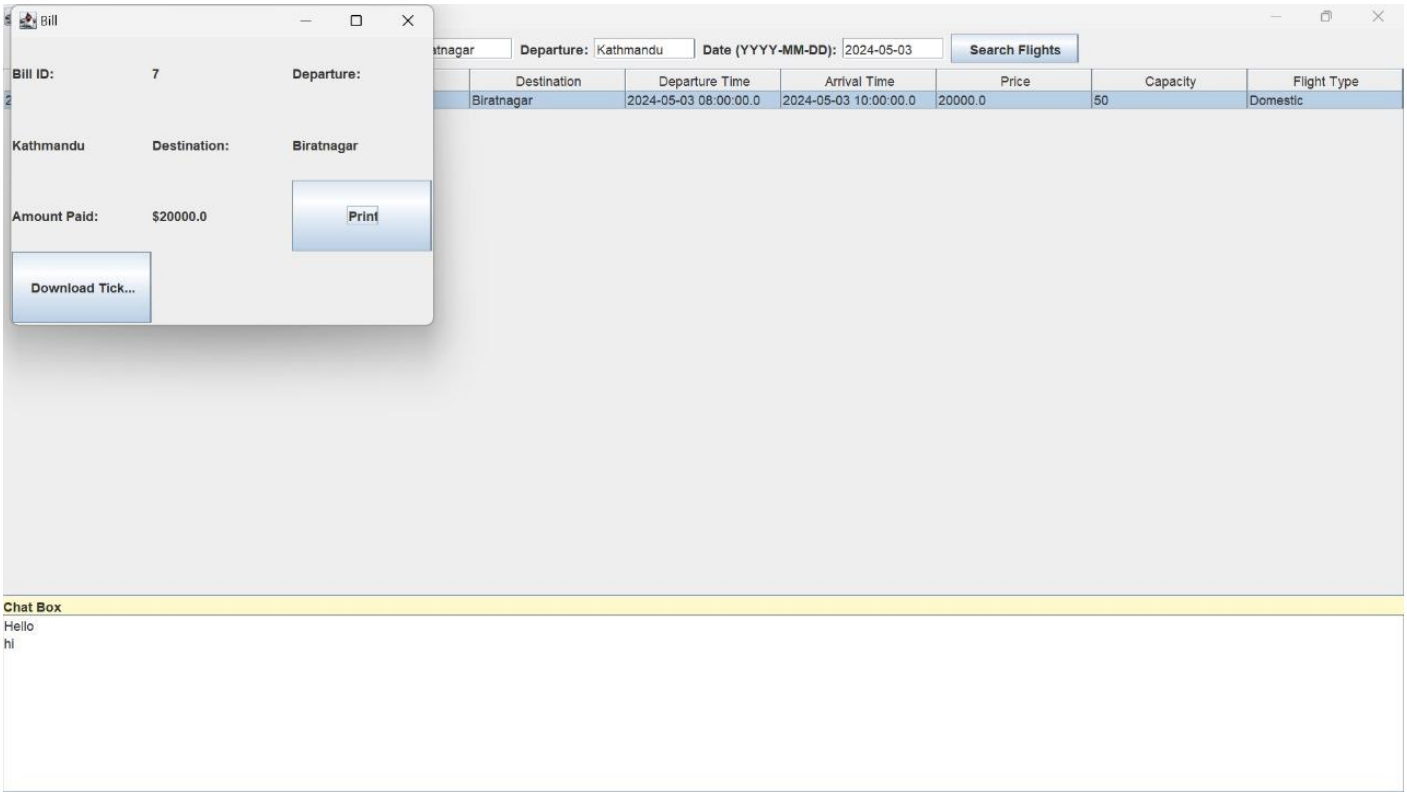
Login step:


Airline Booking App
Username:
Aman123
Password:
•••
Name:
Aman
Date of Birth (YYYY-MM-DD):
Address:
lucknow
Login
Register
Message
Login successful!
OK

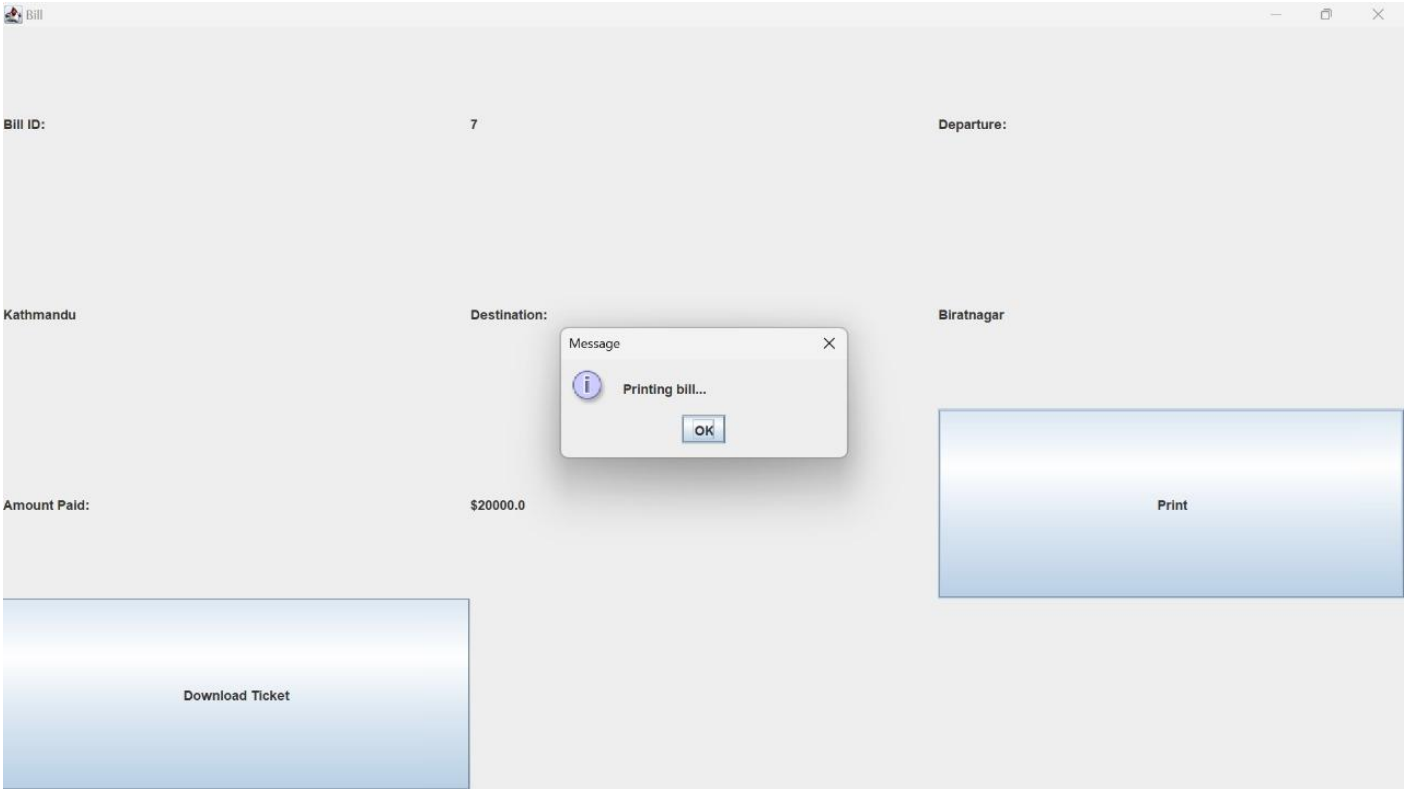
Ticket booking :



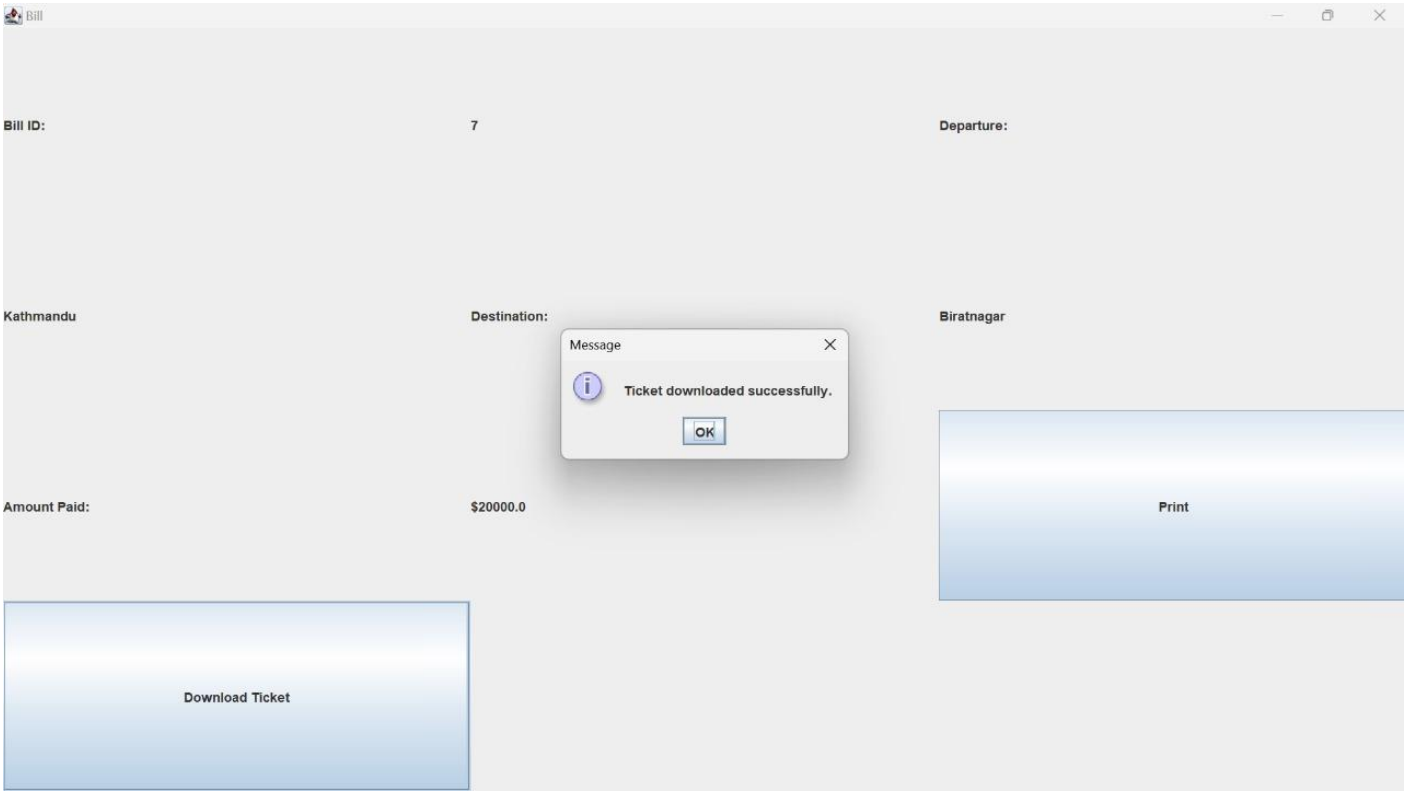
Ticket confirmation:



Printing bill :



Ticket downloading step:



User information in the database :

```
mysql> select * from customer;
+---------+----------+----------+----------+---------------+---------+
| user_id | username | password | name     | Date_of_Birth | address |
+---------+----------+----------+----------+---------------+---------+
|       1 | adc      | abc      | Aditya   | 2004-12-04    | abc     |
|       2 | asity    | dfg      | acy      | 2007-11-07    | fg      |
|       3 | yash     | 1234     | yashraj  | 2024-12-23    | chennai |
|       4 | abcd     | abcd     | prakash  | 2003-12-04    | chennai |
|       5 | Aman123  | gay      | Aman     | 2003-12-03    | lucknow |
+---------+----------+----------+----------+---------------+---------+
5 rows in set (0.07 sec)
```

Ticket booking information in database :

```
mysql> select *from flight;
+-----------+----------+-----------+-------------+--------------------+--------------------+-------+----------+----------------+
| flight_id | airlines | departure | destination | departure_time     | arrival_time       | price | capacity | flight_type_id |
+-----------+----------+-----------+-------------+--------------------+--------------------+-------+----------+----------------+
|         2 | BUDDHA   | Kathmandu | Biratnagar  | 2024-05-03 08:00:00 | 2024-05-03 10:00:00 | 20000 |       50 |              1 |
+-----------+----------+-----------+-------------+--------------------+--------------------+-------+----------+----------------+
1 row in set (0.04 sec)

mysql> |
```

Bill information in database:

```
mysql> select * from bill;
+---------+-------------+-----------+---------------------+------------------+-------------------+-----------------+-------------+---------------------+
| bill_id | customer_id | flight_id | flight_date         | departure_airline | departure_location | arrival_location | amount_paid | bill_date           |
+---------+-------------+-----------+---------------------+------------------+-------------------+-----------------+-------------+---------------------+
|       1 |           1 |         2 | 2024-05-03 07:39:41 | Airlines         | Kathmandu         | Biratnagar      |       20000 | 2024-05-03 07:39:41 |
|       2 |           1 |         2 | 2024-05-03 07:39:56 | Airlines         | Kathmandu         | Biratnagar      |       20000 | 2024-05-03 07:39:56 |
|       3 |           1 |         2 | 2024-05-03 07:40:25 | Airlines         | Kathmandu         | Biratnagar      |       20000 | 2024-05-03 07:40:25 |
|       4 |           1 |         2 | 2024-05-03 08:31:14 | Airlines         | Kathmandu         | Biratnagar      |       20000 | 2024-05-03 08:31:14 |
|       5 |           1 |         2 | 2024-05-03 08:54:44 | Airlines         | Kathmandu         | Biratnagar      |       20000 | 2024-05-03 08:54:44 |
|       6 |           3 |         2 | 2024-05-03 08:58:52 | Airlines         | Kathmandu         | Biratnagar      |       20000 | 2024-05-03 08:58:52 |
|       7 |           5 |         2 | 2024-05-03 09:16:18 | Airlines         | Kathmandu         | Biratnagar      |       20000 | 2024-05-03 09:16:18 |
+---------+-------------+-----------+---------------------+------------------+-------------------+-----------------+-------------+---------------------+
7 rows in set (0.00 sec)
```

Ticket conformation information in database:

```
mysql> select * from ticket_sale;
+-----------+------------------+-------------------+--------------------+---------------------+-------------+-------------+
| ticket_id | customer_user_id | departure_airline | departure_location | departure_time      | flight_type | amount_paid |
+-----------+------------------+-------------------+--------------------+---------------------+-------------+-------------+
|         1 |                1 | Airlines          | Kathmandu          | 2024-05-03 07:39:42 | Domestic    |       20000 |
|         2 |                1 | Airlines          | Kathmandu          | 2024-05-03 07:39:57 | Domestic    |       20000 |
|         3 |                1 | Airlines          | Kathmandu          | 2024-05-03 07:40:25 | Domestic    |       20000 |
|         4 |                1 | Airlines          | Kathmandu          | 2024-05-03 08:31:14 | Domestic    |       20000 |
|         5 |                1 | Airlines          | Kathmandu          | 2024-05-03 08:54:44 | Domestic    |       20000 |
|         6 |                3 | Airlines          | Kathmandu          | 2024-05-03 08:58:52 | Domestic    |       20000 |
|         7 |                5 | Airlines          | Kathmandu          | 2024-05-03 09:16:18 | Domestic    |       20000 |
+-----------+------------------+-------------------+--------------------+---------------------+-------------+-------------+
7 rows in set (0.00 sec)
```

Name :Aditya Gupta

CERTIFICATE OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

ADITYA GUPTA

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges

03 May 2024

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE BY SCALER