SkyLink: A Comprehensive Airline Operations and Management Solution

A PROJECT REPORT

Submitted by

Manas Sharma [Reg No:RA2211056010132]

Lalith Krishan [Reg No:RA2211056010095]

Heer Mehta [Reg No:RA2211056010122]

Under the Guidance of

Dr. Shobana J

Assistant Professor, Department of Data Science and Business Systems

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register No. RA2211056010095, RA2211056010132, RA2211056010122 Certified to be the bonafide work done by **V. Lalith Krishna, Manas Sharma, Heer Mehta** of II year/IV sem B.Tech Degree Course in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

Date: 3-05-24

Faculty in Charge
Dr. Shobana J
Designation
Department name
SRMSIT -KTR

HEAD OF THE DEPARTMENT
Dr. M Lakshmi
Designation
Department name
SRMIST - KTR

ABSTRACT

Effective management of airline data is essential for the smooth operation of the aviation industry. However, traditional relational databases often struggle to handle the complex relationships inherent in airline management. This study explores database management systems, with a particular focus on evaluating their suitability for airline data management. Through a structured approach, a database using MySQL Workbench is designed and populated with diverse airline entities. The study evaluates the system's readiness for querying and analysis, with a particular emphasis on metrics such as query performance. Comparative benchmarking against alternative database solutions provides insights into scalability and performance in managing airline data. The project contributes to understanding the challenges and opportunities associated with leveraging database management systems for effective airline data management.

TABLE OF CONTENTS

Chapter – 1

SkyLink: A Comprehensive Airline Operations and Management Solution

Problem Definition:

SkyLink is an innovative Airlines Database Management System tailored for the aviation industry. It streamlines operations by managing flight schedules, reservations, and ensuring compliance with regulations. The system enhances passenger experience through personalized services and upholds rigorous safety protocols. By automating routine tasks, SkyLink increases operational efficiency and supports strategic decision-making. Its adaptable infrastructure is designed to meet the evolving demands of airline management, fostering growth, and maintaining a competitive edge in the market. SkyLink's intuitive interface and automated processes reduce manual errors and enhance the overall efficiency of airline operations. It's a comprehensive solution that not only simplifies management tasks but also contributes to the sustainability of the aviation ecosystem.

Functionality and Modules:

1. Flight Management Module:

- Manages flight scheduling, routing, and status tracking.
- Stores flight information such as departure, arrival, and date.
- Connects flights to specific aircraft types through the Flight table's foreign key reference to Airplane_type.

2. Reservation Management Module:

- Facilitates passenger reservation processes.
- Stores passenger details including name, address, age, and contact information.
- Associates reservations with specific flights through the Flight_ID foreign key reference.

3. Aircraft Management Module:

- Centralizes information regarding aircraft types.
- Stores details such as capacity, weight, and manufacturer.
- Linked to Flight table through the foreign key reference to A_ID.

4. Fare Management Module:

- Manages fare charges for flights.
- Records fare details like charge amount and description.
- Linked to flights through the Flight_ID foreign key reference.

5. Passenger Management Module:

- Manages passenger information.
- Stores passenger profiles and demographic data.
- Associated with specific flights via the Flight_ID foreign key reference.

6. Country and Airport Management Module:

- Manages country and airport information.
- Stores details such as country codes, names, airport codes, names, cities, and states.
- Foreign key relationships ensure data integrity between countries, airports, and flights.

7. Employee Management Module:

- Manages employee details such as name, address, age, email, and contact information.
- Associates employees with specific airports through the Air_code foreign key reference.

8. Transaction Management Module:

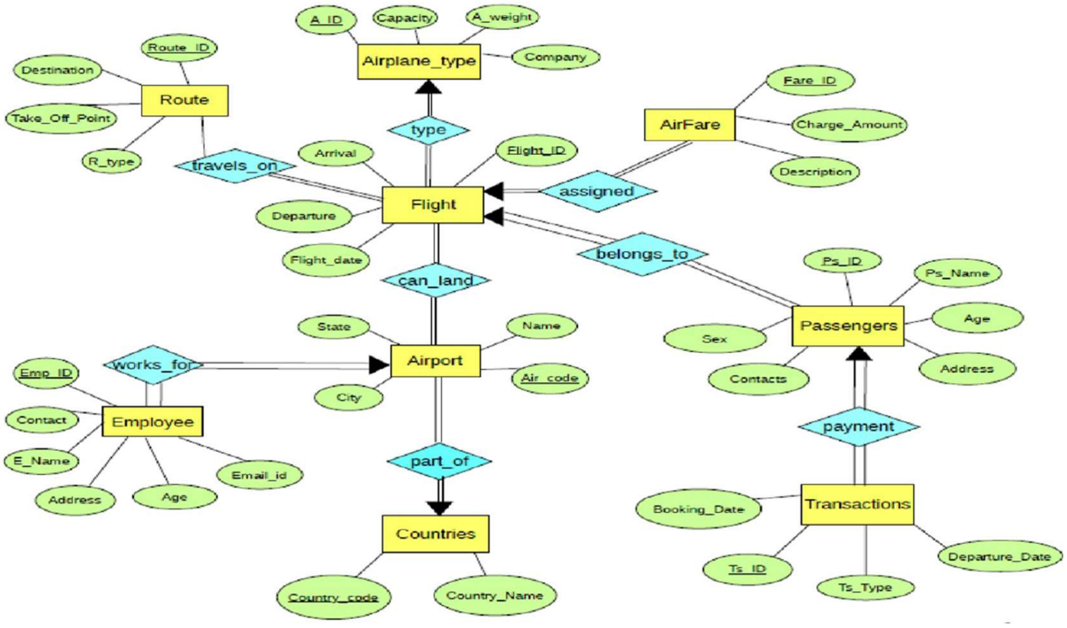
- Tracks various transactions related to flights.
- Records booking and departure dates, transaction types, employee and passenger IDs, flight IDs, and charge amounts.
- Maintains integrity through foreign key relationships with employees, passengers, flights, and fare charges.

9. Route Management Module:

- Manages flight routes.
- Stores details such as route ID, take-off points, destinations, and route types.
- Linked to flights through the Travels_on table's foreign key reference to Flight_ID.

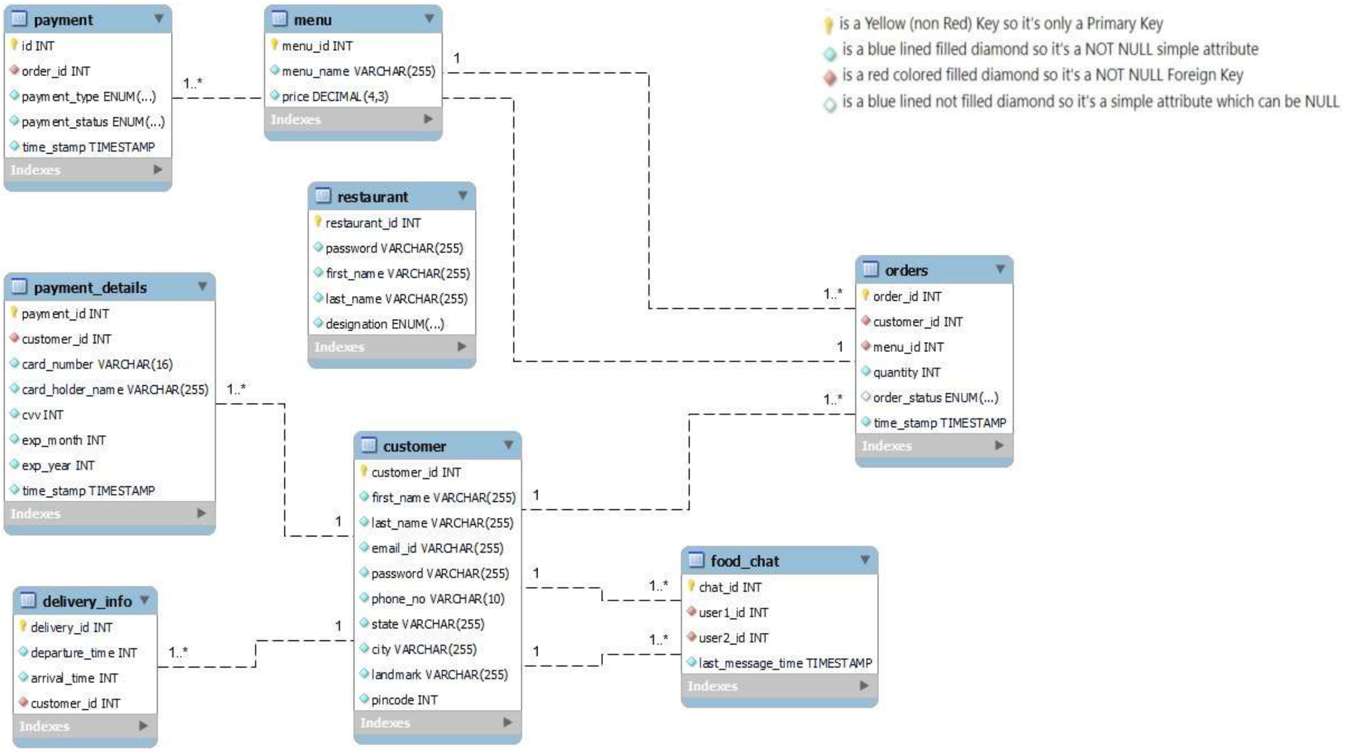
SkyLink is designed to revolutionize airline operations by providing a robust, integrated solution for managing critical aspects of the business. With its modular architecture and advanced functionality, SkyLink offers scalability, flexibility, and efficiency, enabling airlines to enhance service delivery and maintain competitiveness in the aviation industry.

ER diagram: -


A_ID
Capacity
A_weight
Airplane_type
Company
Route_ID
Destination
Route
Take_Off_Point
R_type
type
Fare_ID
AirFare
Charge_Amount
Description
Arrival
Flight_ID
travels_on
assigned
Flight
Departure
Flight_date
belongs_to
can_land
Ps_ID
Ps_Name
Passengers
Age
Sex
Address
Contacts
State
Name
Airport
Air_code
City
works_for
Emp_ID
Contact
Employee
E_Name
Address
Age
Email_id
part_of
payment
Countries
Transactions
Booking_Date
Departure_Date
Ts_ID
Ts_Type
Country_code
Country_Name

Chapter 2

Schema: -



🔑 is a Yellow (non Red) Key so it's only a Primary Key

◆ is a blue lined filled diamond so it's a NOT NULL simple attribute

◆ is a red colored filled diamond so it's a NOT NULL Foreign Key

◇ is a blue lined not filled diamond so it's a simple attribute which can be NULL

Creation of Database Tables for the project

```
CREATE DATABASE dbms;

CREATE TABLE Airplane_type(

  A_ID INT,

  Capacity INT,

  A_weight INT,

  Company VARCHAR(15),

  PRIMARY KEY(A_ID)

);

CREATE TABLE Route(

   Route_ID INT,
   Take_Off_point VARCHAR(15),
   Destination VARCHAR(15),
   R_type VARCHAR(15),
   PRIMARY KEY(Route_ID)
);

CREATE TABLE Flight(
   Flight_ID VARCHAR(15),
   Departure VARCHAR(30),
   Arrival VARCHAR(30),
   Flight_date DATE,
   A_ID INT,
   PRIMARY KEY(Flight_ID),
   FOREIGN KEY (A_ID) REFERENCES Airplane_type(A_ID)
);

CREATE TABLE AirFare(
   Fare_ID INT,
   Charge_Amount INT,
   Description VARCHAR(25),
   Flight_ID VARCHAR(15),
   PRIMARY KEY(Fare_ID),
   FOREIGN KEY (Flight_ID) REFERENCES Flight(Flight_ID)
);

CREATE TABLE Passengers(
   Ps_ID INT,
   Ps_Name VARCHAR(20),
   Address VARCHAR(50),
   Age INT,
   Sex VARCHAR(1),
   Contacts VARCHAR(10),
   Flight_ID VARCHAR(15),
   PRIMARY KEY(Ps_ID),
   FOREIGN KEY (Flight_ID) REFERENCES Flight(Flight_ID)
);

CREATE TABLE Countries(
   Country_code INT,
   Country_Name VARCHAR(20),
   PRIMARY KEY(Country_code)
);


CREATE TABLE Airport(
   Air_code VARCHAR(10),
   Air_Name VARCHAR(50),
   City VARCHAR(20),
   State VARCHAR(20),
   Country_code INT,
   PRIMARY KEY(Air_code),
   FOREIGN KEY (Country_code) REFERENCES Countries(Country_code)
);

CREATE TABLE Employees(
   Emp_ID INT,
   E_Name VARCHAR(20),
   Address VARCHAR(50),
   Age INT,
   Email_ID VARCHAR(20),
   Contact VARCHAR(20),
   Air_code VARCHAR(10),
   PRIMARY KEY(Emp_ID),
   FOREIGN KEY (Air_code) REFERENCES Airport(Air_code)
);
CREATE TABLE Can_Land(
   Air_code VARCHAR(10),
   Flight_ID VARCHAR(15),
   PRIMARY KEY(Air_code,Flight_ID),
   FOREIGN KEY(Air_code) REFERENCES Airport(Air_code),
   FOREIGN KEY(Flight_ID) REFERENCES Flight(Flight_ID)
);
CREATE TABLE Transactions(
   TS_ID INT,
   Booking_Date DATE,
   Departure_Date DATE,
   TS_Type VARCHAR(20),
   Emp_ID INT,
   Ps_ID INT,
   Flight_ID VARCHAR(15),
   Charge_Amount INT,
   PRIMARY KEY(TS_ID),
   FOREIGN KEY (Emp_ID) REFERENCES Employees(Emp_ID) ,
   FOREIGN KEY (Ps_ID) REFERENCES Passengers(Ps_ID),
   FOREIGN KEY (Flight_ID) REFERENCES Flight(Flight_ID),
   FOREIGN KEY (Charge_Amount) REFERENCES AirFare(Fare_ID)
);
CREATE TABLE Travels_on(
   Route_ID INT,
   Flight_ID VARCHAR(15),
   PRIMARY KEY(Route_ID,Flight_ID),
   FOREIGN KEY(Route_ID) REFERENCES Route(Route_ID),
   FOREIGN KEY(Flight_ID) REFERENCES Flight(Flight_ID)
);
CREATE TABLE Airplane_type(
   A_ID INT,
   Capacity INT,
   A_weight INT,
   Company VARCHAR(15),
   PRIMARY KEY(A_ID));
```

Chapter 3

1. List all flights with their departure and arrival times, along with the airplane type and company:

sql

```
SELECT f.Flight_ID, f.Departure, f.Arrival, a.Capacity, a.Company
FROM Flight f
JOIN Airplane_type a ON f.A_ID = a.A_ID;
```

2. Calculate the total charge amount for each flight including the fare and the total number of passengers:

sql

```
SELECT f.Flight_ID, SUM(af.Charge_Amount) AS Total_Charge, COUNT(p.Ps_ID) AS Total_Passengers
FROM Flight f
LEFT JOIN AirFare af ON f.Flight_ID = af.Flight_ID
LEFT JOIN Passengers p ON f.Flight_ID = p.Flight_ID
GROUP BY f.Flight_ID;
```

3. Create a view to show the details of flights and their corresponding departure and arrival airport names:

sql

```
CREATE VIEW Flight_Details AS
SELECT f.Flight_ID, f.Departure, f.Arrival, a1.Air_Name AS Departure_Airport, a2.Air_Name
AS Arrival_Airport
FROM Flight f
JOIN Airport a1 ON f.Departure = a1.Air_code
JOIN Airport a2 ON f.Arrival = a2.Air_code;
```

4. Create a trigger to update the departure date in Transactions table when a flight's departure date is updated:

sql

```
CREATE TRIGGER Update_Transactions
AFTER UPDATE ON Flight
FOR EACH ROW
BEGIN
    UPDATE Transactions
    SET Departure_Date = NEW.Departure
    WHERE Flight_ID = NEW.Flight_ID;
END;
```

5. Calculate the total weight of passengers on each flight (assuming each passenger has a weight):

sql

```
SELECT f.Flight_ID, SUM(a.A_weight) AS Total_Passenger_Weight
FROM Flight f
JOIN Passengers p ON f.Flight_ID = p.Flight_ID
JOIN Airplane_type a ON f.A_ID = a.A_ID
GROUP BY f.Flight_ID;
```

6. Use a cursor to fetch details of passengers on a specific flight:

sql

```
DECLARE cur_passengers CURSOR FOR
SELECT Ps_ID, Ps_Name, Age, Sex
FROM Passengers
WHERE Flight_ID = 'your_flight_id';
OPEN cur_passengers
FETCH NEXT FROM cur_passengers INTO @Ps_ID, @Ps_Name, @Age, @Sex;
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Process the fetched data as needed
    PRINT 'Passenger ID: ' + CAST(@Ps_ID AS VARCHAR(10)) + ', Name: ' + @Ps_Name + ', Age: ' + CAST(@Age AS VARCHAR(3)) + ', Sex: ' + @Sex;

    FETCH NEXT FROM cur_passengers INTO @Ps_ID, @Ps_Name, @Age, @Sex;
END;
CLOSE cur_passengers;
DEALLOCATE cur_passengers;
```

Chapter 4

Pitfalls:

1. Redundancy: The schema includes repeated creation of the `Airplane_type` table. Redundant tables can lead to data inconsistency and maintenance issues.

2. Data Integrity: There are no constraints like `NOT NULL` or `CHECK` constraints defined in several tables. This could lead to the insertion of invalid or incomplete data.

3. Normalization: The schema appears to be in the initial stages and lacks normalization. Redundant data and update anomalies might arise.

Dependencies:

1. Entity Relationships: There are clear entity relationships such as flights being associated with airplanes, passengers, routes, and fares.

2. Foreign Keys: Foreign key constraints define dependencies between tables, ensuring referential integrity.

Normalization:

Normalization is a process of organizing the attributes and tables of a relational database to minimize redundancy and dependency.

1. First Normal Form (1NF): Ensure that each column contains atomic values. It seems the schema already satisfies this.

2. Second Normal Form (2NF): Make sure that non-key attributes are fully functional dependent on the primary key. Splitting the `Passengers` table might be necessary to avoid partial dependencies.

3. Third Normal Form (3NF): Remove transitive dependencies. For instance, the `Flight` table could be split into `Flight` and `Flight_Details` to separate flight information from departure and arrival details.

4. Fourth Normal Form (4NF): Address multi-valued dependencies if present. Evaluate if any tables exhibit multi-valued dependencies that need to be resolved.

5. Fifth Normal Form (5NF): Further normalization to address join dependencies if necessary.

Applying these normalization steps can improve data integrity, reduce redundancy, and enhance the overall efficiency of the database schema.

Chapter 5

Implementing concurrency control and recovery mechanisms is crucial for ensuring data consistency and system reliability in a database management system. Here are some approaches to implementing these mechanisms:

Concurrency Control:

1. Lock-Based Concurrency Control:

- Implement locking mechanisms such as exclusive locks (X-lock) and shared locks (S-lock) to control access to database resources.
- Use lock manager to grant or deny lock requests based on the concurrency control protocol (e.g., two-phase locking).
- Ensure proper lock granularity (row-level, page-level, table-level) based on the application requirements and performance considerations.

2. Timestamp-Based Concurrency Control:

- Assign unique timestamps to transactions to determine their relative order.
- Use timestamps to enforce serializability by allowing transactions with higher timestamps to proceed while blocking conflicting transactions with lower timestamps.
- Implement mechanisms to handle timestamp conflicts, such as aborting or rolling back transactions.

3. Optimistic Concurrency Control:

- Allow transactions to execute without acquiring locks upfront.

- Perform validation checks at the end of transactions to detect conflicts.
- Rollback transactions that violate consistency constraints and retry them with updated data.

Recovery Mechanisms:

1. Write-Ahead Logging (WAL):
   - Use a write-ahead logging protocol to ensure that log records are written to stable storage before corresponding database modifications.
   - Implement a log manager to manage log records and maintain a consistent log sequence.

2. Checkpointing:
   - Periodically perform checkpoints to write dirty pages from memory to disk and update the checkpoint record in the log.
   - Implement algorithms such as fuzzy checkpointing to minimize the impact on transaction processing.

3. Transaction Undo/Redo:
   - Maintain undo and redo logs to support transaction recovery.
   - During transaction execution, log undo information for rollback purposes and redo information for recovery after system failures.

4. Shadow Paging:

- Use shadow paging technique to maintain a shadow copy of the database.
- Update pages in the shadow copy and maintain a mapping table to track the current version of each page.
- Ensure atomicity of updates by committing changes atomically at the page level.

5. Checkpoint Restart:

- After a system failure, restart the database system from the last checkpoint to reduce recovery time.
- Use checkpoint information and transaction logs to restore the database to a consistent state.

6. Recovery Manager:

- Implement a recovery manager responsible for coordinating recovery operations, including redo, undo, and log-based recovery.
- Ensure that recovery manager adheres to ACID properties (Atomicity, Consistency, Isolation, Durability) to maintain data integrity.

By implementing these concurrency control and recovery mechanisms, database systems can provide robust support for concurrent transactions and recover from failures effectively, ensuring data consistency and system reliability.

Code

```
-- Create table for transactions with timestamp
CREATE TABLE Transactions (
    Transaction_ID INT PRIMARY KEY,
    Timestamp TIMESTAMP,
    Status VARCHAR(10) -- Can be 'Active', 'Aborted', or 'Committed'
);

-- Create table for locking
CREATE TABLE Locks (
    Lock_ID INT PRIMARY KEY,
    Transaction_ID INT,
    Resource_ID INT, -- Resource being locked
    Lock_Type VARCHAR(10), -- Exclusive or Shared lock
    FOREIGN KEY (Transaction_ID) REFERENCES Transactions(Transaction_ID)
);

-- Create table for logging
CREATE TABLE Log (
    Log_ID INT PRIMARY KEY,
    Timestamp TIMESTAMP,
    Transaction_ID INT,
    Operation VARCHAR(10), -- 'Write', 'Commit', 'Abort'
    Resource_ID INT, -- Resource being modified
    Old_Value VARCHAR(255), -- Value before modification
    New_Value VARCHAR(255) -- Value after modification
);

-- Function to acquire lock
CREATE FUNCTION AcquireLock(transaction_id INT, resource_id INT, lock_type VARCHAR(10))
RETURNS BOOLEAN
BEGIN
    DECLARE conflict INT;
    SET conflict = 0;

    -- Check for conflicts
    IF lock_type = 'Exclusive' THEN
        SELECT COUNT(*) INTO conflict
        FROM Locks
        WHERE Resource_ID = resource_id AND Lock_Type = 'Exclusive';
    ELSE
        SELECT COUNT(*) INTO conflict
        FROM Locks
        WHERE Resource_ID = resource_id AND Lock_Type = 'Exclusive';
    END IF;
    -- If no conflict, acquire lock
    IF conflict = 0 THEN
        INSERT INTO Locks (Transaction_ID, Resource_ID, Lock_Type)
        VALUES (transaction_id, resource_id, lock_type);
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;

-- Function to release lock
CREATE FUNCTION ReleaseLock(transaction_id INT, resource_id INT)
RETURNS BOOLEAN
BEGIN
    DELETE FROM Locks
    WHERE Transaction_ID = transaction_id AND Resource_ID = resource_id;
    RETURN TRUE;
END;

-- Procedure for transaction commit
CREATE PROCEDURE CommitTransaction(transaction_id INT)
BEGIN
```

```
    -- Update transaction status
    UPDATE Transactions SET Status = 'Committed' WHERE Transaction_ID =
transaction_id;

    -- Log commit operation
    INSERT INTO Log (Timestamp, Transaction_ID, Operation)
    VALUES (NOW(), transaction_id, 'Commit');

    -- Release all locks held by the transaction
    DELETE FROM Locks WHERE Transaction_ID = transaction_id;
END;

-- Procedure for transaction abort
CREATE PROCEDURE AbortTransaction(transaction_id INT)
BEGIN
    -- Update transaction status
    UPDATE Transactions SET Status = 'Aborted' WHERE Transaction_ID =
transaction_id;

    -- Log abort operation
    INSERT INTO Log (Timestamp, Transaction_ID, Operation)
    VALUES (NOW(), transaction_id, 'Abort');
    -- Rollback changes using log records
    DECLARE done BOOLEAN DEFAULT FALSE;
    DECLARE cur CURSOR FOR
        SELECT Resource_ID, Old_Value
        FROM Log
        WHERE Transaction_ID = transaction_id AND Operation = 'Write';
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO resource_id, old_value;
        IF done THEN
            LEAVE read_loop;
        END IF;
        -- Apply rollback logic here
        -- Example: UPDATE Resource SET Value = old_value WHERE ID = resource_id;
    END LOOP;
    CLOSE cur;

    -- Release all locks held by the transaction
    DELETE FROM Locks WHERE Transaction_ID = transaction_id;
END;
```

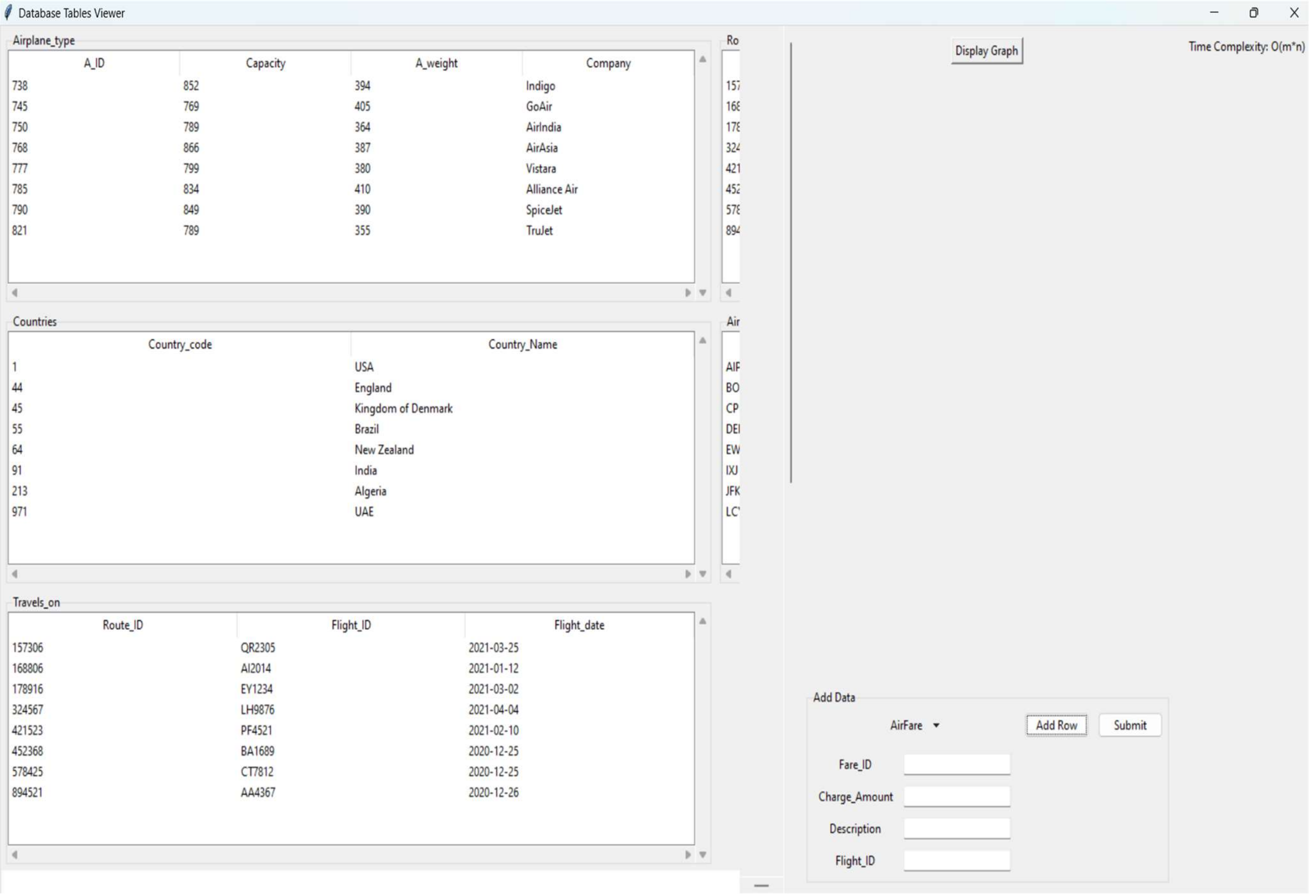
Chapter 6

CODE: -

```
import tkinter as tk

from tkinter import ttk

from sqlalchemy import create_engine

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


class ScrollableFrame(ttk.Frame):

    def __init__(self, container, *args, **kwargs):

        super().__init__(container, *args, **kwargs)

        self.canvas = tk.Canvas(self)

        v_scrollbar = ttk.Scrollbar(self, orient="vertical", command=self.canvas.yview)

        h_scrollbar = ttk.Scrollbar(self, orient="horizontal", command=self.canvas.xview)

        self.scrollable_frame = ttk.Frame(self.canvas)

                                                self.scrollable_frame.bind("<Configure>",              lambda              e:
self.canvas.configure(scrollregion=self.canvas.bbox("all")))

        self.canvas.create_window((0, 0), window=self.scrollable_frame, anchor="nw")

        self.canvas.configure(yscrollcommand=v_scrollbar.set)

        self.canvas.configure(xscrollcommand=h_scrollbar.set)

        self.canvas.pack(side="left", fill="both", expand=True)

        v_scrollbar.pack(side="right", fill="y")

        h_scrollbar.pack(side="bottom", fill="x")


def fetch_and_display_all_tables(root):
    connection_string = "mysql+mysqlconnector://root:1234@localhost/dbms"
    engine = create_engine(connection_string)
    tables = ['Airplane_type', 'Route', 'Flight', 'AirFare', 'Passengers', 'Countries', 'Airport', 'Employees', 'Can_Land', 'Transactions', 'Travels_on']
    columns = 5

    for i, table_name in enumerate(tables):
        query = f"SELECT * FROM {table_name}"
        df = pd.read_sql(query, engine)
        table_frame = ttk.LabelFrame(root.scrollable_frame, text=table_name)
        table_frame.grid(row=i // columns, column=i % columns, padx=5, pady=5, sticky="nsew")
        tree = ttk.Treeview(table_frame)
        tree["columns"] = df.columns.tolist()
        tree["show"] = "headings"
        for column in df.columns.tolist():
            tree.heading(column, text=column)
        for index, row in df.iterrows():
            tree.insert("", "end", values=row.tolist())
        scrollbar_y = ttk.Scrollbar(table_frame, orient="vertical", command=tree.yview)
        scrollbar_y.pack(side="right", fill="y")
        tree.configure(yscrollcommand=scrollbar_y.set)
        scrollbar_x = ttk.Scrollbar(table_frame, orient="horizontal", command=tree.xview)
        scrollbar_x.pack(side="bottom", fill="x")
        tree.configure(xscrollcommand=scrollbar_x.set)
        tree.pack(expand=True, fill="both")

def display_graph():
    connection_string = "mysql+mysqlconnector://root:1234@localhost/dbms"
    engine = create_engine(connection_string)
    tables = ['Airplane_type', 'Route', 'Flight', 'AirFare', 'Passengers', 'Countries', 'Airport', 'Employees', 'Can_Land', 'Transactions', 'Travels_on']
    table_counts = []
    for table_name in tables:
        query = f"SELECT * FROM {table_name}"
        df = pd.read_sql(query, engine)
        table_counts.append(len(df))
    plt.figure(figsize=(14, 8))
    sns.barplot(x=tables, y=table_counts)
    plt.title('Number of rows in each table')
    plt.xticks(rotation=90)
    canvas = FigureCanvasTkAgg(plt.gcf(), master=scrollable_frame.scrollable_frame)
    canvas.draw()
    plot_row = len(tables) // 5 + 1
    canvas.get_tk_widget().grid(row=plot_row, column=0, columnspan=5, sticky="nsew")

root = tk.Tk()
root.title("Database Tables Viewer")

scrollable_frame = ScrollableFrame(root)
scrollable_frame.pack(expand=True, fill="both", side="left")

info_frame = tk.Frame(root, width=200)
info_frame.pack(side="right", fill="y", expand=False)

time_complexity_label = tk.Label(info_frame, text="Time Complexity: O(m*n)", justify="left")
time_complexity_label.pack(pady=10, padx=10)
```

```
btn_display_graph = tk.Button(root, text="Display Graph", command=display_graph)
btn_display_graph.pack(side=tk.TOP, pady=12)
fetch_and_display_all_tables(scrollable_frame)

root.mainloop()
INSERT INTO Airplane_type VALUES (738, 853, 394, 'Indigo');
INSERT INTO Airplane_type VALUES (777, 800, 380, 'Vistara');
INSERT INTO Airplane_type VALUES (750, 790, 364, 'AirIndia');
INSERT INTO Airplane_type VALUES (790, 850, 390, 'SpiceJet');
INSERT INTO Airplane_type VALUES (745, 770, 405, 'GoAir');
INSERT INTO Airplane_type VALUES (768, 867, 387, 'AirAsia');
INSERT INTO Airplane_type VALUES (821, 790, 355, 'TruJet');
INSERT INTO Airplane_type VALUES (785, 835, 410, 'Alliance Air');

INSERT INTO Route VALUES (168806, 'London', 'Delhi', 'Direct');
INSERT INTO Route VALUES (157306, 'NewJersey', 'Mumbai', '2Hr Break');
INSERT INTO Route VALUES (178916, 'Washington', 'Jodhpur', '3Hr Break');
INSERT INTO Route VALUES (324567, 'Chennai', 'Denmark', 'Direct');
INSERT INTO Route VALUES (452368, 'Chandigard', 'NewYork', '3Hr Break');
INSERT INTO Route VALUES (894521, 'Daman', 'Delhi', 'Direct');
INSERT INTO Route VALUES (578425, 'Beijing', 'Punjab', 'Direct');
INSERT INTO Route VALUES (421523, 'Hyderabad', 'Jammu & Kashmir', 'Direct');

INSERT INTO Flight VALUES ('AI2014', '2021-01-12 08:45am', '2021-01-12 10:25pm', '2021-01-12', 738);
INSERT INTO Flight VALUES ('QR2305', '2020-12-26 12:05pm', '2020-12-27 12:25pm', '2020-12-26', 777);
INSERT INTO Flight VALUES ('EY1234', '2021-02-10 05:00am', '2021-02-10 10:30pm', '2021-02-10',
750);
INSERT INTO Flight VALUES ('LH9876', '2021-02-25 10:15am', '2021-02-25 11:00pm', '2021-02-25', 790);
INSERT INTO Flight VALUES ('BA1689', '2021-03-02 2:15am', '2021-03-02 10:00pm', '2021-03-02', 745);
INSERT INTO Flight VALUES ('AA4367', '2021-03-25 12:05am', '2021-03-25 02:15am', '2021-03-25', 768);
INSERT INTO Flight VALUES ('CT7812', '2021-04-04 2:15pm', '2021-04-04 8:00pm', '2021-04-04', 821);
INSERT INTO Flight VALUES ('PF4521', '2020-12-25 5:00pm', '2020-12-25 10:30pm', '2020-12-25', 785);

INSERT INTO AirFare VALUES (1, 27341, 'Standard Single', 'AI2014');
INSERT INTO AirFare VALUES (4, 34837, 'Standard Return', 'QR2305');
INSERT INTO AirFare VALUES (2, 42176, 'Key Fare Single', 'EY1234');
INSERT INTO AirFare VALUES (3, 27373, 'Business Return', 'LH9876');
INSERT INTO AirFare VALUES (6, 44592, 'Advanced Purchase', 'BA1689');
INSERT INTO AirFare VALUES (5, 8777, 'Superpex Return', 'AA4367');
INSERT INTO AirFare VALUES (7, 9578, 'Standard Return', 'CT7812');
INSERT INTO AirFare VALUES (8, 4459, 'Superpex Return', 'PF4521');

INSERT INTO Passengers VALUES (1, 'Steve Smith', '2230 Northside,Apt 11,London', 30, 'M', '8080367290', 'AI2014');
INSERT INTO Passengers VALUES (2, 'Ankita Ahir', '3456 Vikas Apts,Apt 102,New Jersey', 26, 'F', '8080367280', 'QR2305');
INSERT INTO Passengers VALUES (4, 'Akhilesh Joshi', '345 Chatam courts,Apt 678,Chennai', 29, 'M', '9080369290', 'EY1234');
INSERT INTO Passengers VALUES (3, 'Khyati Mishra', '7820 Mccallum courts,Apt 234,Washington', 30, 'F', '8082267280', 'LH9876');
INSERT INTO Passengers VALUES (5, 'Rom Solanki', '1234 Baker Apts,Apt 208,Chandigard', 60, 'M', '9004568903', 'EY1234');
INSERT INTO Passengers VALUES (6, 'Lakshmi Sharma', '1110 Fir hills,Apt 90,Daman', 30, 'F', '7666190505', 'AA4367');
INSERT INTO Passengers VALUES (8, 'Manan Lakhani', '7720 Mccallum Blvd,Apt 77,Beijing', 45, 'M', '8124579635', 'CT7812');
INSERT INTO Passengers VALUES (7, 'Ria Gupta', 'B-402,Aditya Apt,Hyderabad', 34, 'F', '9819414036', 'EY1234');

INSERT INTO Countries VALUES (+44, 'England');
INSERT INTO Countries VALUES (+1, 'USA');
INSERT INTO Countries VALUES (+91, 'India');
INSERT INTO Countries VALUES (+45, 'Kingdom of Denmark');
INSERT INTO Countries VALUES (+64, 'New Zealand');
INSERT INTO Countries VALUES (+971, 'UAE');
INSERT INTO Countries VALUES (+213, 'Algeria');
INSERT INTO Countries VALUES (+55, 'Brazil');

INSERT INTO Airport VALUES ('DEL', 'Indira Gandhi International Airport', 'Delhi', 'UP', +91);
INSERT INTO Airport VALUES ('BOM', 'Chhatrapati Shivaji Maharaj International Airport', 'Mumbai', 'Maharashtra', +91);
INSERT INTO Airport VALUES ('LCY', 'London City Airport', 'Newham', 'London', +44);
INSERT INTO Airport VALUES ('EWR', 'Newark Liberty International Airport', 'Newark', 'New Jersey', +1);
INSERT INTO Airport VALUES ('JFK', 'John F.Kennnedy International Airport', 'New York City', 'New York', +1);
INSERT INTO Airport VALUES ('CPH', 'Copenhagen Airport', 'Copenhagen', 'Denmark', +45);
INSERT INTO Airport VALUES ('AIP', 'Adampur Airport', 'Jalandhar', 'Punjab', +91);
INSERT INTO Airport VALUES ('IXJ', 'Satwari Airport', 'Jammu', 'Jammu & Kashmir', +91);
INSERT INTO Employees VALUES (1234, 'Rekha Tiwary', '202-Meeta Apt,Yogi Nagar,Mumbai', 30, 'rekha1234@gmail.com', '+918530324018', 'DEL');
INSERT INTO Employees VALUES (3246, 'John Dsouza', '302-Fountain Apt,ElizaBeth Street, Newham', 26, 'john2346@gmail.com', '+447911123456', 'BOM');
INSERT INTO Employees VALUES (9321, 'Sanjay Rathod', '62-Patwa Apt,Pradeep Nagar, Delhi', 36, 'sanjay78@gmail.com', '+917504681201', 'LCY');
INSERT INTO Employees VALUES (8512, 'Hafsa Iqmar', '1023-Prajwal Apt,Newark', 41, 'hafsa964@gmail.com', '6465554468', 'EWR');
INSERT INTO Employees VALUES (7512, 'Akshay Sharma', 'Akshay Villa,Queens Street,Copenhagen', 20, 'akshay27@gmail.com', '+45886443210', 'JFK');
INSERT INTO Employees VALUES (5123, 'Lara Jen', '28-Mark road,Victoria street,New York City', 31, 'jenlara4@gmail.com', '+448000751234', 'CPH');
INSERT INTO Employees VALUES (2458, 'Johny Paul', '45-Balaji Apt,Ajit Nagar,Jalandar', 32, 'johnypaul8@gmail.com', '+919785425154', 'AIP');
INSERT INTO Employees VALUES (4521, 'Nidhi Maroliya', '6-Matruchaya Apt,Park Road, Jammu', 31, 'nidhi785@gmail.com', '+918211954901', 'IXJ');

INSERT INTO Can_Land VALUES ('DEL', 'AI2014');
INSERT INTO Can_Land VALUES ('BOM', 'QR2305');
INSERT INTO Can_Land VALUES ('LCY', 'EY1234');
INSERT INTO Can_Land VALUES ('EWR', 'LH9876');
INSERT INTO Can_Land VALUES ('JFK', 'BA1689');
INSERT INTO Can_Land VALUES ('CPH', 'AA4367');
INSERT INTO Can_Land VALUES ('AIP', 'CT7812');
INSERT INTO Can_Land VALUES ('IXJ', 'PF4521');

INSERT INTO Transactions VALUES (12345678, '2021-02-21', '2021-02-22', 'Google Pay', 1234, 1, 'AI2014', 27341);
INSERT INTO Transactions VALUES (45612789, '2021-01-12', '2021-01-14', 'Credit Card', 3246, 2, 'QR2305', 34837);
INSERT INTO Transactions VALUES (56987123, '2020-12-05', '2020-12-02', 'Paytm', 9321, 4, 'EY1234', 42176);
INSERT INTO Transactions VALUES (45321879, '2021-03-15', '2021-03-16', 'PhonePe', 8512, 3, 'LH9876', 27373);
INSERT INTO Transactions VALUES (75145863, '2021-04-22', '2021-04-25', 'Paytm', 7512, 5, 'EY1234', 44592);
INSERT INTO Transactions VALUES (17892455, '2021-02-05', '2021-02-08', 'Paytm', 5123, 6, 'AA4367', 8777);
INSERT INTO Transactions VALUES (24517852, '2021-03-06', '2021-03-08', 'PhonePe', 2458, 8, 'CT7812', 9578);
INSERT INTO Transactions VALUES (32548525, '2021-01-20', '2021-01-25', 'Credit Card', 4521, 7, 'EY1234', 4459);

INSERT INTO Travels_on VALUES (168806, 'AI2014');
INSERT INTO Travels_on VALUES (157306, 'QR2305');
INSERT INTO Travels_on VALUES (178916, 'EY1234');
INSERT INTO Travels_on VALUES (324567, 'LH9876');
INSERT INTO Travels_on VALUES (452368, 'BA1689');
INSERT INTO Travels_on VALUES (894521, 'AA4367');
INSERT INTO Travels_on VALUES (578425, 'CT7812');
INSERT INTO Travels_on VALUES (421523, 'PF4521');
```

Chapter 7



Database Tables Viewer

Airplane_type

A_ID	Capacity	A_weight	Company
738	852	394	Indigo
745	769	405	GoAir
750	789	364	AirIndia
768	866	387	AirAsia
777	799	380	Vistara
785	834	410	Alliance Air
790	849	390	SpiceJet
821	789	355	TruJet

Ro

Ro
157
168
178
324
421
452
578
894

Countries

Country_code	Country_Name
1	USA
44	England
45	Kingdom of Denmark
55	Brazil
64	New Zealand
91	India
213	Algeria
971	UAE

Air

Air
AIF
BO
CP
DEI
EW
IXJ
JFK
LC'

Travels_on

Route_ID	Flight_ID	Flight_date
157306	QR2305	2021-03-25
168806	AI2014	2021-01-12
178916	EY1234	2021-03-02
324567	LH9876	2021-04-04
421523	PF4521	2021-02-10
452368	BA1689	2020-12-25
578425	CT7812	2020-12-25
894521	AA4367	2020-12-26

Display Graph

Time Complexity: O(m*n)

Add Data

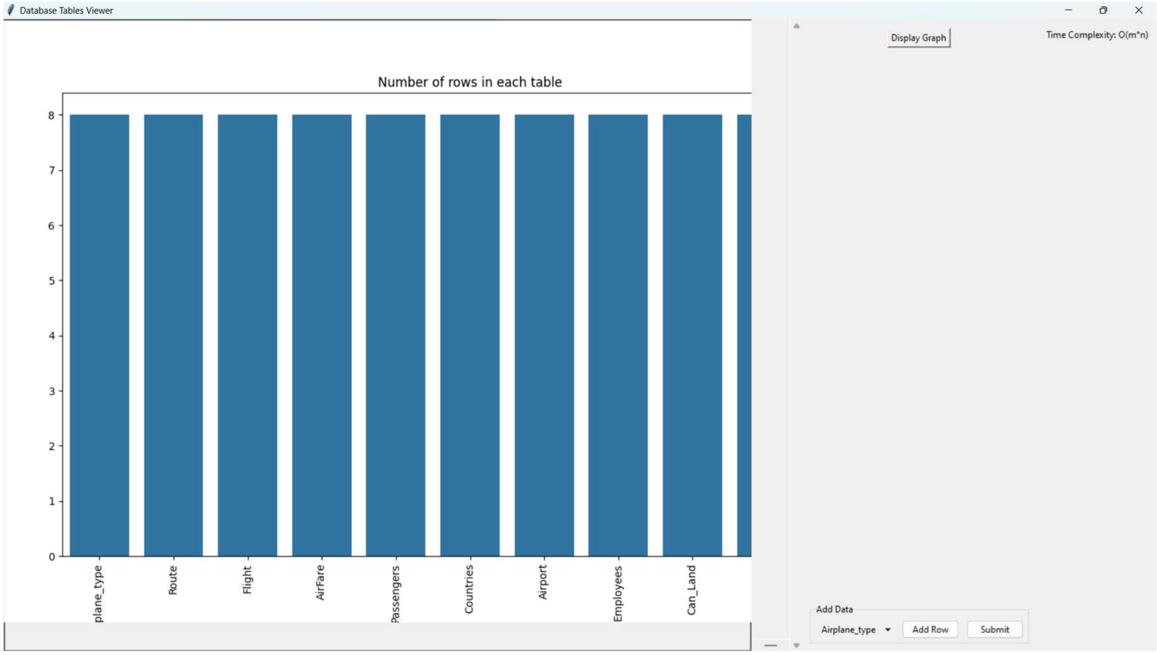
AirFare ▾ Add Row Submit

Fare_ID

Charge_Amount

Description

Flight_ID



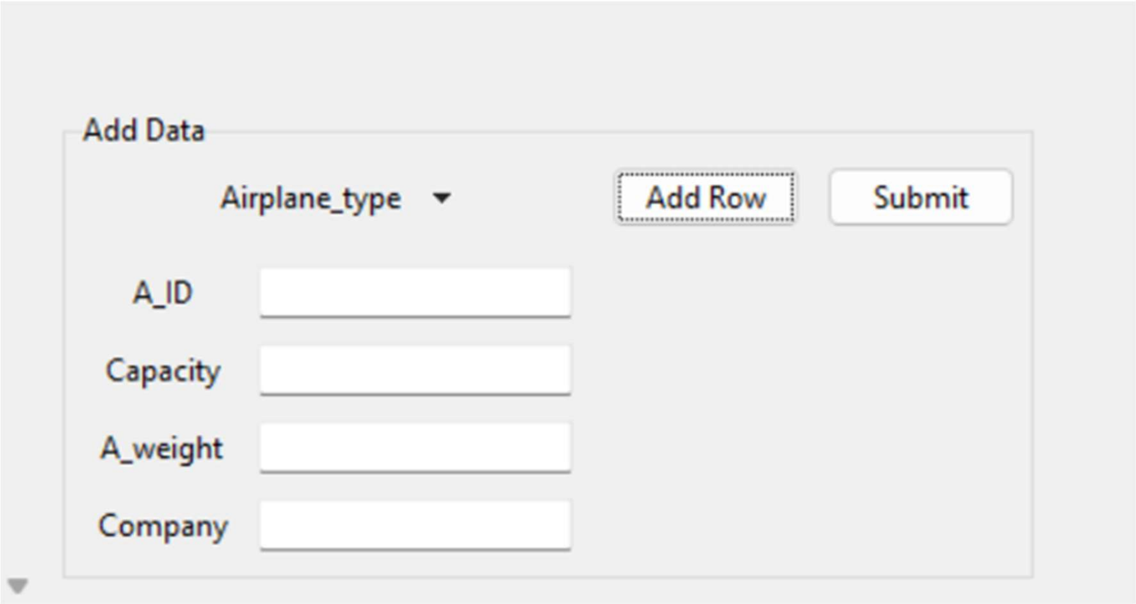
Database Tables Viewer

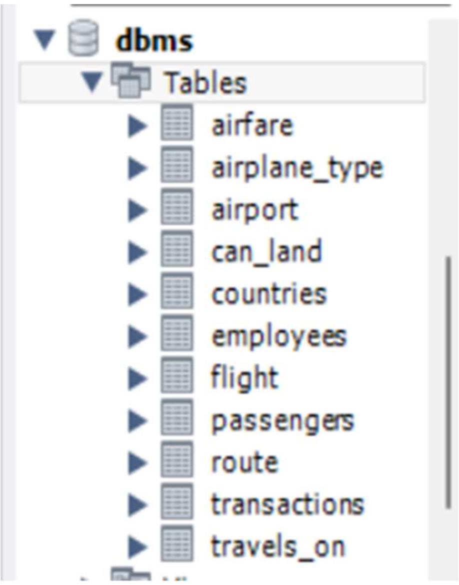
Display Graph

Time Complexity: O(m*n)

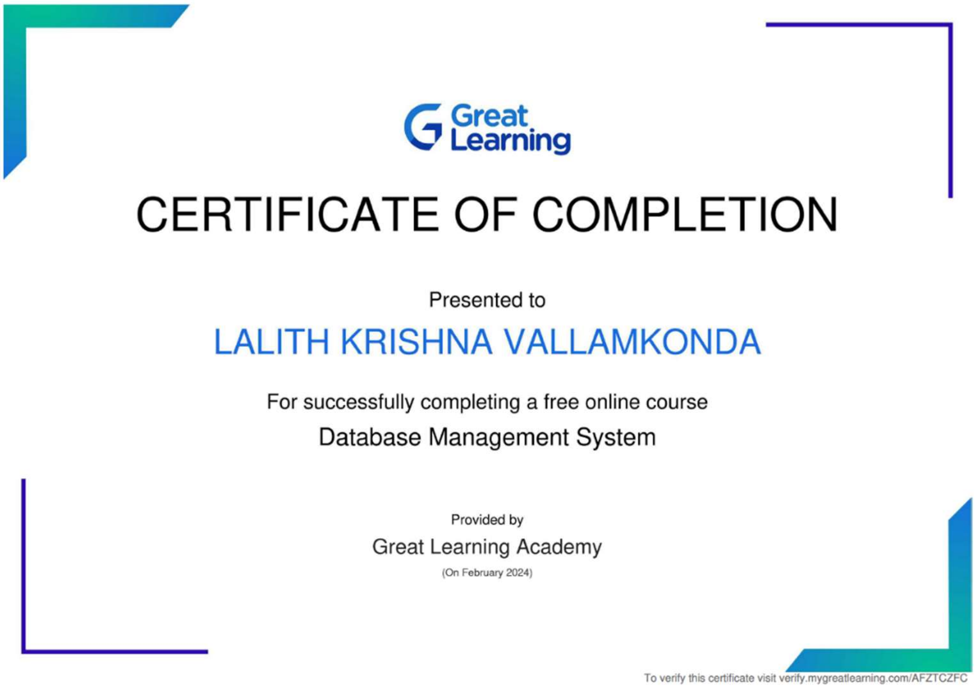
Add Data

Airplane_type ▾ Add Row Submit


Add Data
Airplane_type
Add Row
Submit
A_ID
Capacity
A_weight
Company


dbms
Tables
airfare
airplane_type
airport
can_land
countries
employees
flight
passengers
route
transactions
travels_on

Chapter 8



Great Learning

CERTIFICATE OF COMPLETION

Presented to

LALITH KRISHNA VALLAMKONDA

For successfully completing a free online course

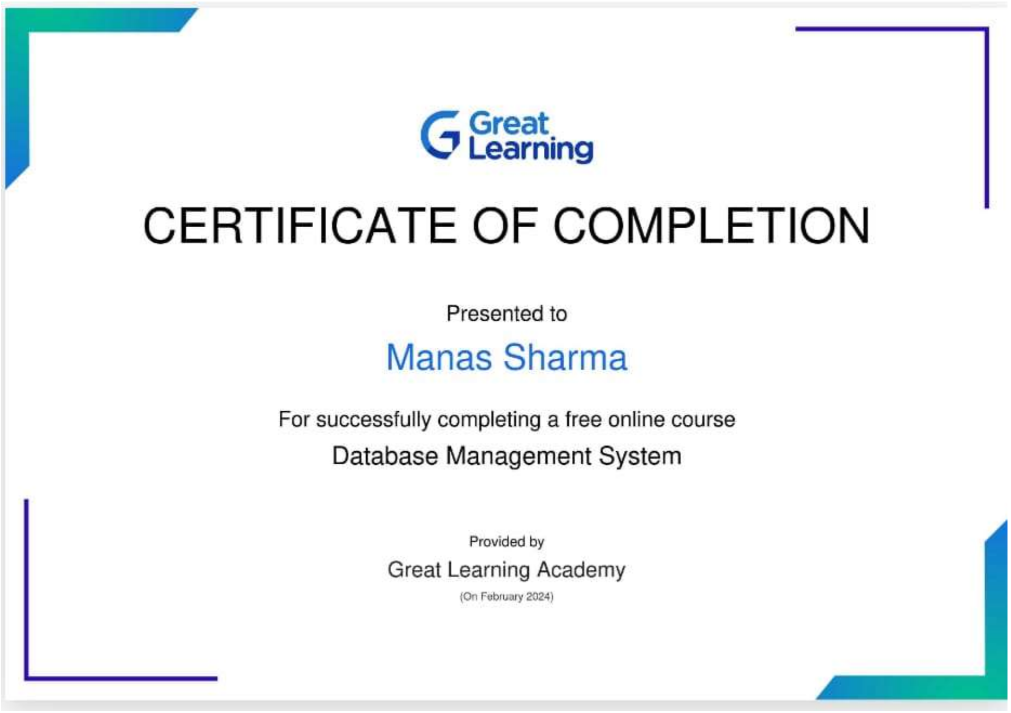
Database Management System

Provided by

Great Learning Academy

(On February 2024)

To verify this certificate visit verify.mygreatlearning.com/AFZTCZFC



Great Learning

CERTIFICATE OF COMPLETION

Presented to

Manas Sharma

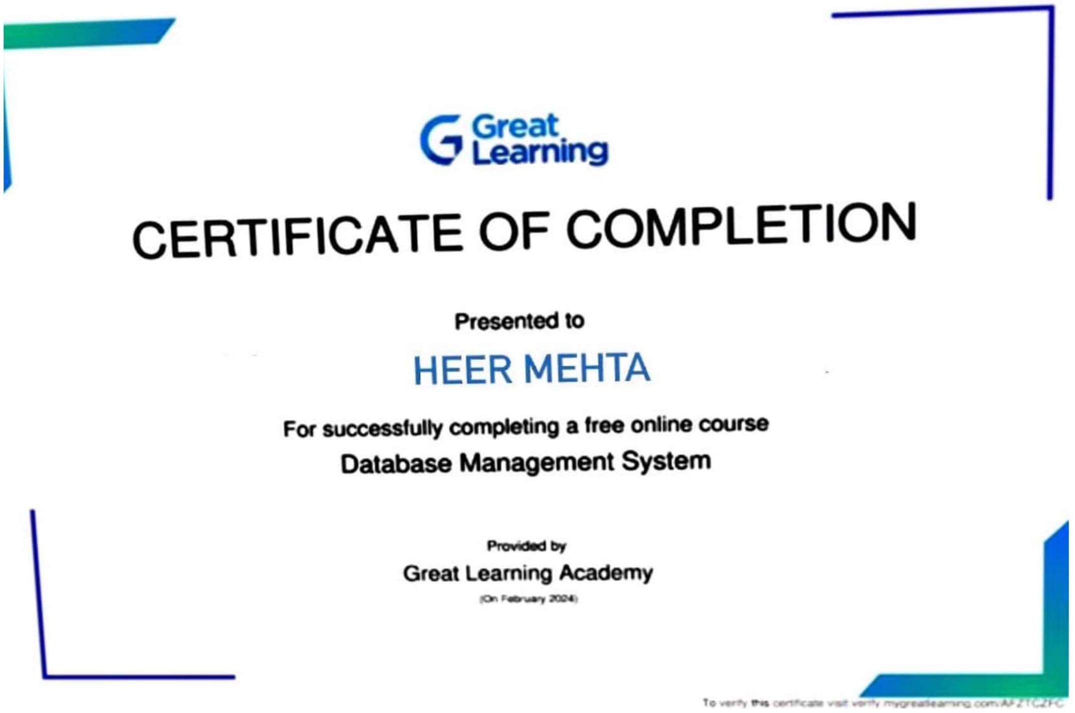
For successfully completing a free online course

Database Management System

Provided by

Great Learning Academy

(On February 2024)



Great Learning

CERTIFICATE OF COMPLETION

Presented to

HEER MEHTA

For successfully completing a free online course

Database Management System

Provided by

Great Learning Academy

(On February 2024)

To verify this certificate visit verify.mygreatlearning.com/AFZTCZFC