AIRLINE TICKET BOOKING MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

Jay Adithya RA2211003011013

M V Prahlad Karthik RA2211003011022

Under the Guidance of

Dr. M. KANDAN

Assistant Professor, Department of Computing Technologies

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

MAY 2024

SRM

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Certified to be the bonafide work done by **Jay Adithya (RA2211003011013) & M V Prahlad Karthik (RA2211003011022)** 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: 30th April, 2024

[illegible]

Faculty in Charge
Dr. M. KANDAN
Assistant Professor
Department of Computing Technologies
SRMIST - KTR

Department of Computing Technologies
Kattankulathur
603 203
SRM IST

M. Pushpalatha

HEAD OF THE DEPARTMENT
Dr. M. PUSHPALATHA
Professor
Department of Computing Technologies
SRMIST - KTR

ABSTRACT

The current landscape of airline ticket booking systems is fraught with inefficiencies and challenges that impede operational excellence and inhibit growth. This abstract outlines the key issues plaguing existing systems, including inconsistent data models, manual data entry processes, limited system integration, and inadequate reporting capabilities. These deficiencies result in errors, slower processing times, and hindered decision-making abilities. To address these challenges, there is a pressing need for a modernized airline ticket booking management system that emphasizes robust data management, streamlined processes, and enhanced reporting capabilities. Such a system would improve operational efficiency, facilitate scalability, and empower airlines to make informed decisions based on comprehensive insights into booking trends and customer behavior. By addressing these critical issues, airlines can enhance the booking experience for customers while optimizing internal processes.

TABLE OF CONTENTS

Problem Statement

The current airline booking system suffers from several critical issues that hinder efficiency, scalability, and decision-making. Firstly, the lack of a formal data model creates inconsistencies, redundancies, and difficulties in retrieving accurate information. Additionally, manual data entry and limited integration between systems lead to errors, slower processing, and information silos. Furthermore, complex manual procedures for booking, payments, and seat assignments are prone to errors and time-consuming, and the system struggles to accommodate growth, hindering expansion and profitability. Finally, limited reporting capabilities restrict insights into booking trends, customer behavior, and operational performance, leading to suboptimal decision-making and challenges in responding to disruptions. These combined issues highlight the urgent need for a modernized and robust data management system to address these challenges and improve the overall booking experience and operational efficiency.

Chapter-1

Problem Understanding, Identification of Entity and Relationships, Construction of DB using ER Model for the project

Existing System:

The existing system appears to be hindered by antiquated technology, lacking the scalability and flexibility needed for modern demands. Its monolithic architecture further restricts adaptation and integration with other vital systems. Additionally, the limited reporting tools fail to provide the comprehensive insights and real-time analysis necessary for informed decision-making. These combined limitations suggest a significant need for a revamped system built on modern technology, modular design, and advanced reporting capabilities.

1.3 Objective of the project:

1. **Efficient Data Management:** Implement the proposed ER diagram, creating a well-organized and normalized database. This structure streamlines data access, retrieval, and updates, ensuring efficient management and scalability as the system grows.
2. **Minimize Errors and Enhance Quality:** Automate data capture and validation processes. This significantly reduces manual errors and inconsistencies, leading to cleaner, more reliable data for analysis and decision-making.
3. **Security and Compliance:** Establish clear data ownership and access controls. This enhances data security by ensuring only authorized users can access and modify specific information, further strengthening compliance with regulations.
4. **Seamless Booking Experience:** Implement an integrated booking system for automated booking creation, payment processing, and seat allocation. This streamlines the entire booking process, minimizing manual interventions and offering a smooth experience for customers.
5. **Real-Time Availability and Conflict Resolution:** Develop real-time availability checks and conflict resolution mechanisms. This prevents double bookings and ensures customers access accurate information, leading to fewer booking errors and frustrations.
6. **Self-Service Convenience:** Offer self-service options through web, mobile app, and kiosks. This empowers customers to manage their bookings independently, reduces workload for agents, and provides greater convenience and flexibility

7. **Visualize Trends and Performance:** Design comprehensive reporting dashboards to visualize booking trends, customer behavior, and operational performance. This allows for easy identification of patterns, insights, and areas for improvement.
8. **Predictive Analytics and Resource Optimization:** Leverage data analytics tools to identify patterns, predict demand, and optimize resource allocation. This empowers proactive decision-making, leading to improved efficiency, cost savings, and better customer service.
9. **Real-Time Decision-Making:** Generate real-time reports and alerts for immediate response to disruptions and proactive decision-making. This allows airlines to quickly adapt to changing conditions, minimize disruptions, and improve overall operational efficiency.
10. **User-Friendly Interface:** Develop a user-friendly and intuitive interface for both customers and staff, simplifying booking processes and system navigation. This ensures user satisfaction and reduces training time for staff.
11. **Multiple Access Points:** Offer multiple access points through web, mobile app, and kiosks. This provides customers with flexibility and convenience, allowing them to manage bookings on their preferred platform.
12. **Personalized Recommendations:** Provide personalized recommendations and targeted offers based on customer preferences and booking history. This enhances customer engagement, increases satisfaction, and potentially leads to higher booking volumes.

2. ER Diagram:

2.1 Entity and Their Attributes

Client: Client ID (primary key), Username, Password, First Name, Middle Name, Last Name, Status.

Booking: Booking ID (primary key), Client ID (foreign key), Flight ID (foreign key), Booking Date, Payment ID (foreign key).

Flight: Flight ID (primary key), Airline ID (foreign key), Flight Number, Departure Date, Departure Time, Arrival Time, Destination, Class Selection.

Payment: Payment ID (primary key), Booking ID (foreign key), Amount, Payment Date.

Airline: Airline ID (primary key), Airline Name.

Seat: Seat ID (primary key), Booking ID (foreign key), Flight ID (foreign key).

Administrator: Username, Password.

2.2 Relationships between Entities:

Client: Has many: Bookings: One client can make multiple bookings. **Belongs to one: None:** Clients are not directly linked to other entities besides their bookings.

Booking: Belongs to one: Client: Each booking is associated with a single client. **Has one: Flight:** Each booking references a specific flight. **Has one: Payment:** Each booking has a single associated payment. **Has one: Seat:** Each booking secures a single seat on a designated flight.

Flight: Belongs to one: Airline: Each flight belongs to a single airline. **Has many: Bookings:** One flight can be booked multiple times. **Has one: Seat:** Each flight has multiple seats, but seats are linked to bookings, not directly to flights.

Payment: Belongs to one: Booking: Each payment is associated with a specific booking.

Airline: Has many: Flights: One airline operates multiple flights.

Seat: Belongs to one: Booking: Each seat is assigned to a single booking. **Belongs to one: Flight:** Each seat is located on a specific flight.

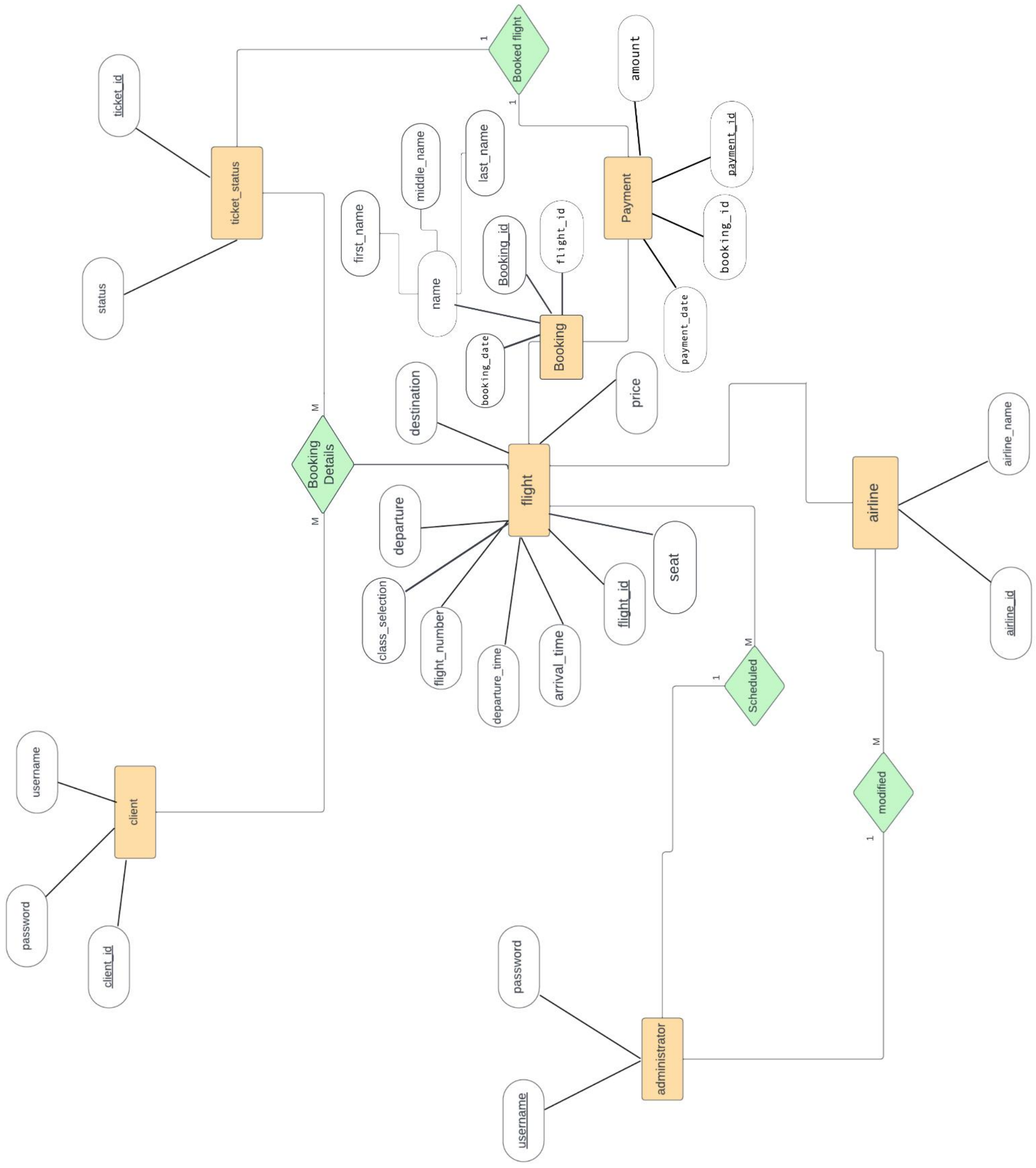


Fig 1.1 ER Diagram on Airline ticket booking management system

Chapter-2

Design of Relational Schemas, Creation of Database Tables for the project

. Relational Tables and Schema

3.1 Schema Diagram

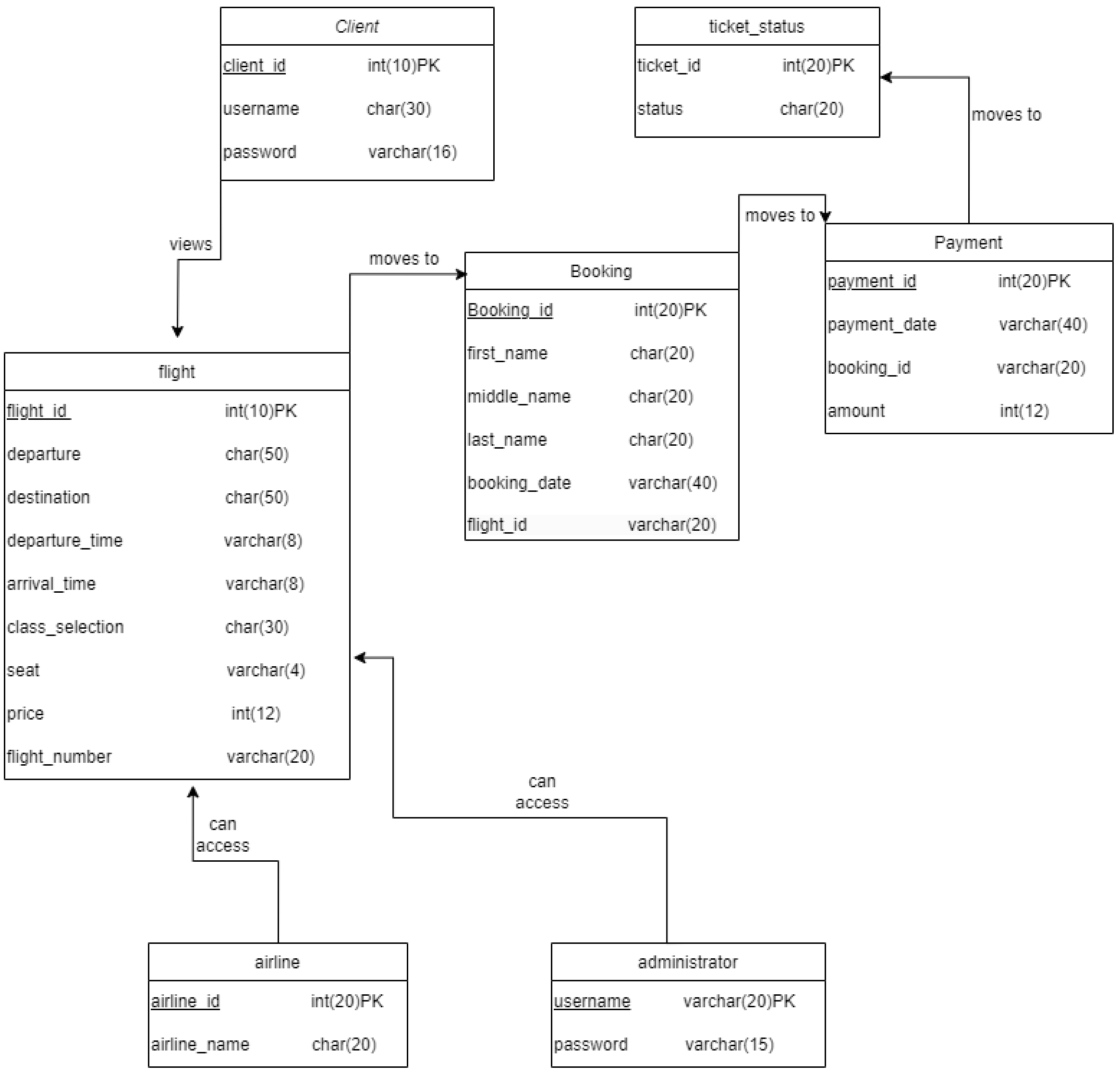


Figure 3.1 Schema Diagram for Airline ticket booking management system

Schema

Client(client_id, username, password)

flight(flight_id, departure, destination, Departure time, Arrival time, class_selection, seat, price, flight_number)

booking(booking_id, first_name, middle_name, last_name, booking_date, flight_id)
payment(payment_id, payment_date, booking_id, amount)

ticket_status(ticket_id, status)

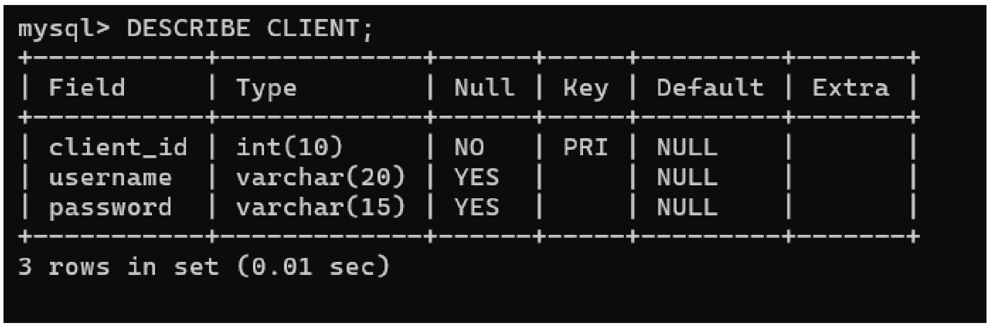
airline(airline_id, airline_name)

Administrator(username, password)

Relational Tables

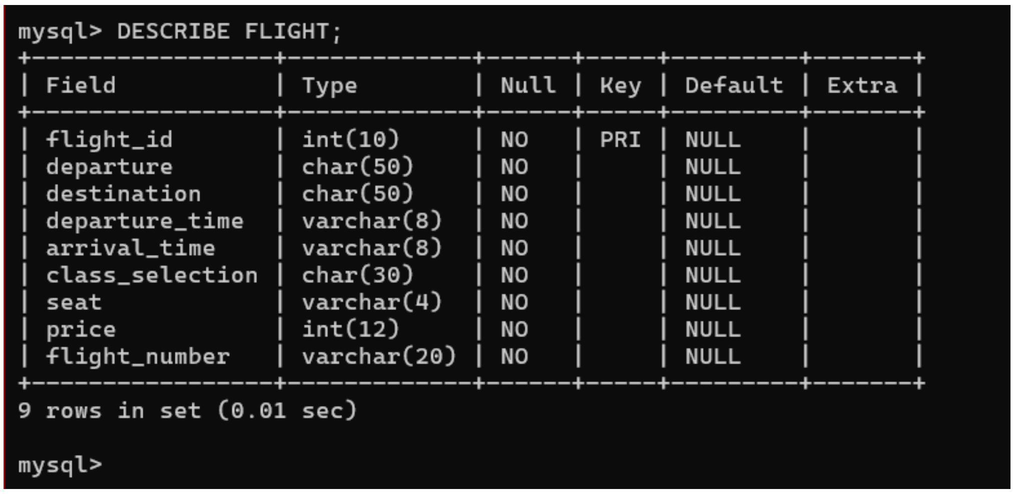
3.2.1 DDL Commands and Results

```
mysql> CREATE TABLE CLIENT(
-> client_id INT(10) PRIMARY KEY,
-> username VARCHAR(20),
-> password VARCHAR(15) );
```



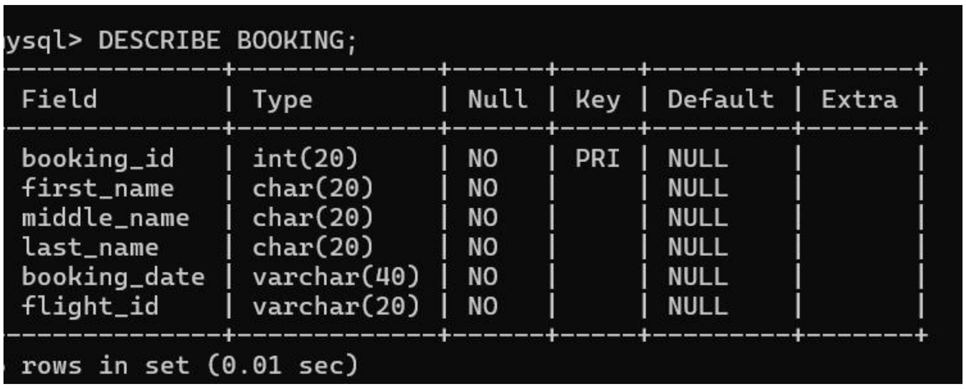
```
mysql> DESCRIBE CLIENT;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| client_id | int(10)     | NO   | PRI | NULL    |       |
| username  | varchar(20) | YES  |     | NULL    |       |
| password  | varchar(15) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

```
mysql> CREATE TABLE FLIGHT(
        -> flight_id INT(10) NOT NULL PRIMARY KEY,
        -> departure CHAR(50) NOT NULL,
        -> destination CHAR(50) NOT NULL,
        -> departure_time VARCHAR(8) NOT NULL,
        -> arrival_time VARCHAR(8) NOT NULL,
        -> class_selection CHAR(30) NOT NULL,
        -> seat VARCHAR(4) NOT NULL,
        -> price INT(12) NOT NULL,
        -> flight_number VARCHAR(20) NOT NULL );
```



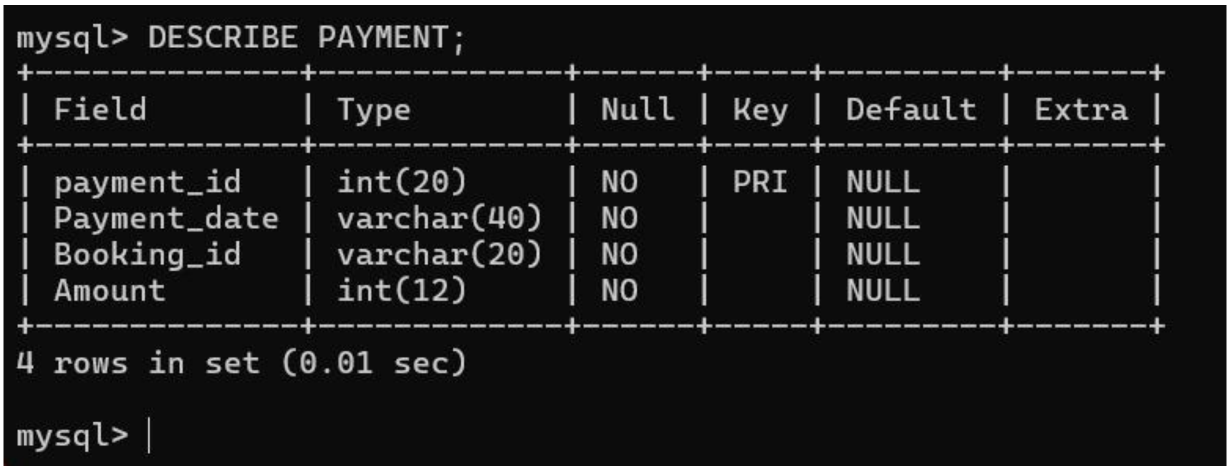
```
mysql> DESCRIBE FLIGHT;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| flight_id       | int(10)     | NO   | PRI | NULL    |       |
| departure       | char(50)    | NO   |     | NULL    |       |
| destination     | char(50)    | NO   |     | NULL    |       |
| departure_time  | varchar(8)  | NO   |     | NULL    |       |
| arrival_time    | varchar(8)  | NO   |     | NULL    |       |
| class_selection | char(30)    | NO   |     | NULL    |       |
| seat            | varchar(4)  | NO   |     | NULL    |       |
| price           | int(12)     | NO   |     | NULL    |       |
| flight_number   | varchar(20) | NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
9 rows in set (0.01 sec)

mysql>
```

```
mysql> CREATE TABLE BOOKING(
    -> booking_id INT(20) NOT NULL PRIMARY KEY,
    -> first_name CHAR(20) NOT NULL,
    -> middle_name CHAR(20) NOT NULL,
    -> last_name CHAR(20) NOT NULL,
   -> booking_date VARCHAR(40) NOT NULL,
   -> flight_id VARCHAR(20) NOT NULL );
```



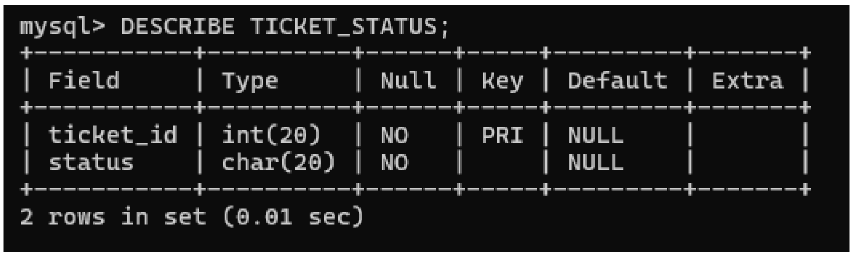
```
ysql> DESCRIBE BOOKING;
--------------+-------------+------+-----+---------+-------+
 Field        | Type        | Null | Key | Default | Extra |
--------------+-------------+------+-----+---------+-------+
 booking_id   | int(20)     | NO   | PRI | NULL    |       |
 first_name   | char(20)    | NO   |     | NULL    |       |
 middle_name  | char(20)    | NO   |     | NULL    |       |
 last_name    | char(20)    | NO   |     | NULL    |       |
 booking_date | varchar(40) | NO   |     | NULL    |       |
 flight_id    | varchar(20) | NO   |     | NULL    |       |
--------------+-------------+------+-----+---------+-------+
 rows in set (0.01 sec)
```

```
mysql> CREATE TABLE PAYMENT(
    -> payment_id INT(20) NOT NULL PRIMARY KEY,
    -> Payment_date VARCHAR(40) NOT NULL,
    -> Booking_id VARCHAR(20) NOT NULL,
    -> Amount INT(12) NOT NULL );
```



```
mysql> DESCRIBE PAYMENT;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| payment_id   | int(20)     | NO   | PRI | NULL    |       |
| Payment_date | varchar(40) | NO   |     | NULL    |       |
| Booking_id   | varchar(20) | NO   |     | NULL    |       |
| Amount       | int(12)     | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> |
```

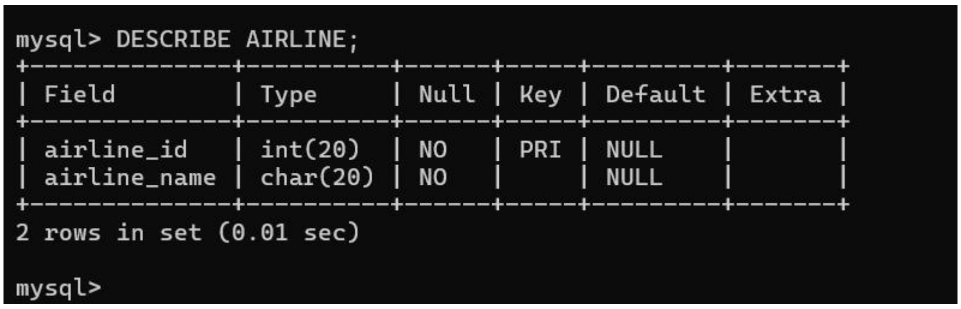
```
mysql> CREATE TABLE TICKET_STATUS(
    -> ticket_id INT(20) NOT NULL PRIMARY KEY,
    -> status CHAR(20) NOT NULL );
```



```
mysql> DESCRIBE TICKET_STATUS;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| ticket_id | int(20)  | NO   | PRI | NULL    |       |
| status    | char(20) | NO   |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> CREATE TABLE AIRLINE(
    -> airline_id INT(20) NOT NULL PRIMARY KEY,
```

-> airline_name CHAR(20) NOT NULL);


```
mysql> DESCRIBE AIRLINE;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| airline_id   | int(20)  | NO   | PRI | NULL    |       |
| airline_name | char(20) | NO   |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql>
```

mysql> CREATE TABLE ADMINISTRATOR(

-> Username VARCHAR(20) NOT NULL PRIMARY KEY,

-> Password VARCHAR(15) NOT NULL);

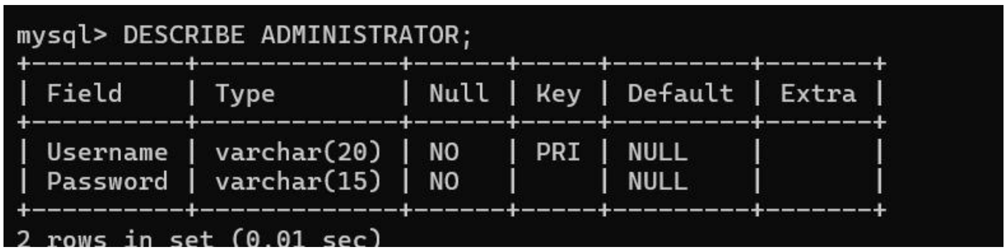

```
mysql> DESCRIBE ADMINISTRATOR;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Username | varchar(20) | NO   | PRI | NULL    |       |
| Password | varchar(15) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Figure 3.2 Create a Tables using DDL Commands for Airline ticket booking management system

3.2.1 DML Commands (INSERT) and Results

mysql> INSERT INTO CLIENT (client_id,username,password) VALUES

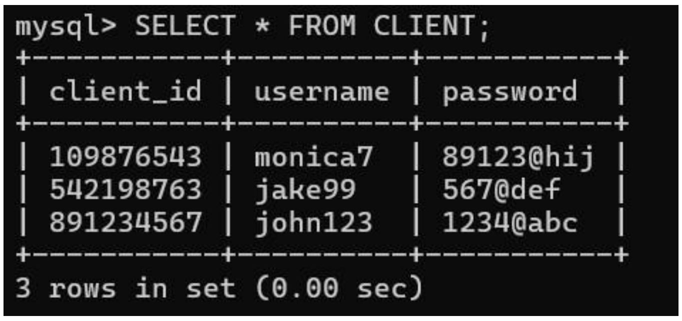
-> ('891234567','john123','1234@abc'),

-> ('542198763','jake99','567@def'),

-> ('109876543','monica7','89123@hij');

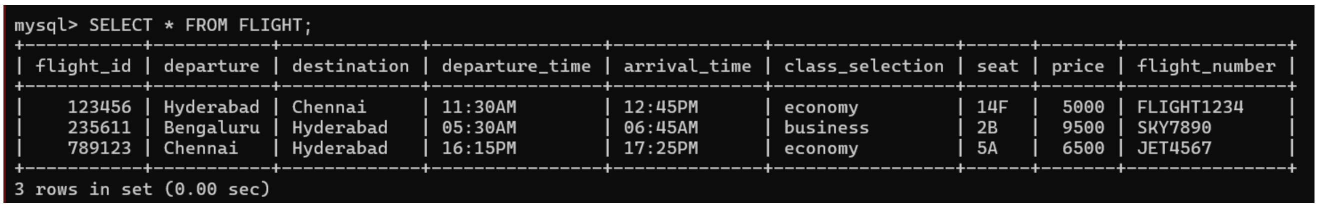
Query OK, 3 rows affected (0.02 sec)

Records: 3 Duplicates: 0 Warnings: 0


```
mysql> SELECT * FROM CLIENT;
+-----------+----------+-----------+
| client_id | username | password  |
+-----------+----------+-----------+
| 109876543 | monica7  | 89123@hij |
| 542198763 | jake99   | 567@def   |
| 891234567 | john123  | 1234@abc  |
+-----------+----------+-----------+
3 rows in set (0.00 sec)
```

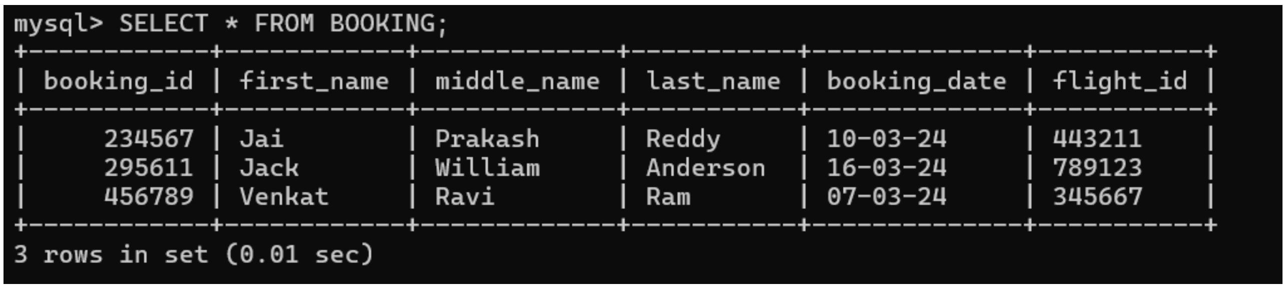
mysql> INSERT INTO FLIGHT (flight_id,departure,destination,departure_time, arrival_time,class_selection,seat,price,flight_number) VALUES

```
->('123456','Hyderabad','Chennai','11:30AM','12:45PM','economy','14F','5000','FLIGHT1234'),
->  ('789123','Chennai','Hyderabad','16:15PM','17:25PM','economy','5A','6500','JET4567'),
 -> ('235611','Bengaluru','Hyderabad','05:30AM','06:45AM','business','2B','9500','SKY7890');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```



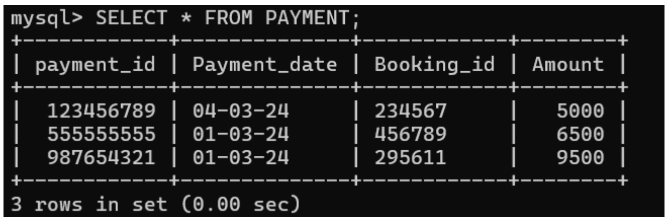
```
mysql> SELECT * FROM FLIGHT;
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
| flight_id | departure | destination | departure_time | arrival_time | class_selection | seat | price | flight_number |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
|    123456 | Hyderabad | Chennai     | 11:30AM        | 12:45PM      | economy         | 14F  |  5000 | FLIGHT1234    |
|    235611 | Bengaluru | Hyderabad   | 05:30AM        | 06:45AM      | business        | 2B   |  9500 | SKY7890       |
|    789123 | Chennai   | Hyderabad   | 16:15PM        | 17:25PM      | economy         | 5A   |  6500 | JET4567       |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
3 rows in set (0.00 sec)
```

```
mysql> INSERT INTO BOOKING
(booking_id,first_name,middle_name,last_name,booking_date,flight_id) VALUES
    -> ('456789','Venkat','Ravi','Ram','07-03-24','345667'),
    -> ('234567','Jai','Prakash','Reddy','10-03-24','443211'),
    -> ('295611','Jack','William','Anderson','16-03-24','789123');
Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0
```



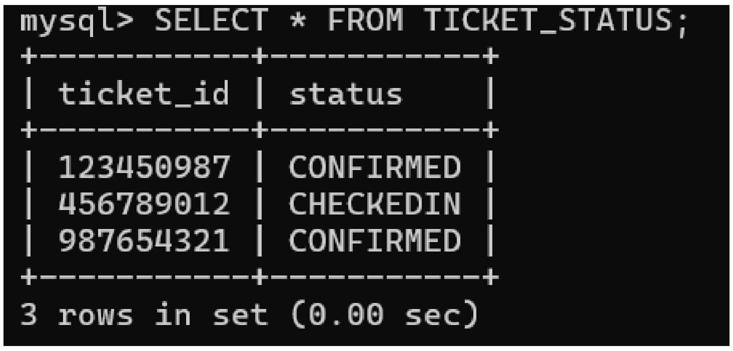
```
mysql> SELECT * FROM BOOKING;
+------------+------------+-------------+-----------+--------------+-----------+
| booking_id | first_name | middle_name | last_name | booking_date | flight_id |
+------------+------------+-------------+-----------+--------------+-----------+
|     234567 | Jai        | Prakash     | Reddy     | 10-03-24     | 443211    |
|     295611 | Jack       | William     | Anderson  | 16-03-24     | 789123    |
|     456789 | Venkat     | Ravi        | Ram       | 07-03-24     | 345667    |
+------------+------------+-------------+-----------+--------------+-----------+
3 rows in set (0.01 sec)
```

```
mysql>  INSERT INTO PAYMENT(payment_id,Payment_date,Booking_id,Amount) VALUES
    ->     ('123456789','04-03-24','234567','5000'),
    ->     ('987654321','01-03-24','295611','9500'),
    ->     ('555555555','01-03-24','456789','6500');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```



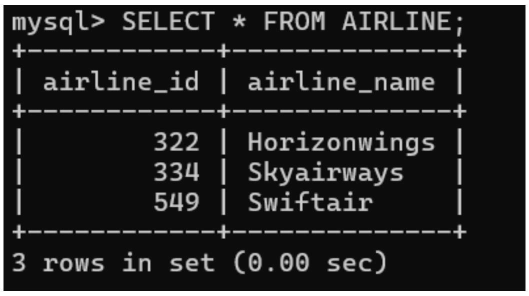
```
mysql> SELECT * FROM PAYMENT;
+------------+--------------+------------+--------+
| payment_id | Payment_date | Booking_id | Amount |
+------------+--------------+------------+--------+
|  123456789 | 04-03-24     | 234567     |   5000 |
|  555555555 | 01-03-24     | 456789     |   6500 |
|  987654321 | 01-03-24     | 295611     |   9500 |
+------------+--------------+------------+--------+
3 rows in set (0.00 sec)
```

```
mysql>  INSERT INTO TICKET_STATUS (ticket_id,status) VALUES
    ->     ('987654321','CONFIRMED'),
    ->     ('456789012','CHECKEDIN'),
    ->     ('123450987','CONFIRMED');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```



```
mysql> SELECT * FROM TICKET_STATUS;
+-----------+-----------+
| ticket_id | status    |
+-----------+-----------+
| 123450987 | CONFIRMED |
| 456789012 | CHECKEDIN |
| 987654321 | CONFIRMED |
+-----------+-----------+
3 rows in set (0.00 sec)
```

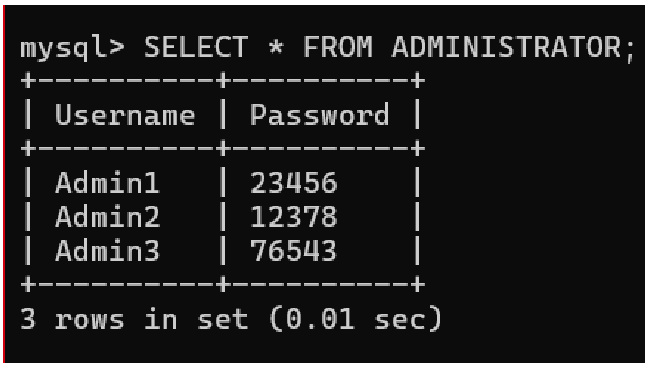
```
mysql>  INSERT INTO AIRLINE (airline_id,airline_name) VALUES
    ->     ('334','Skyairways'),
    ->     ('549','Swiftair'),
    ->     ('322','Horizonwings');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```



```
mysql> SELECT * FROM AIRLINE;
+------------+--------------+
| airline_id | airline_name |
+------------+--------------+
|        322 | Horizonwings |
|        334 | Skyairways   |
|        549 | Swiftair     |
+------------+--------------+
3 rows in set (0.00 sec)
```

```
mysql> INSERT INTO ADMINISTRATOR (Username,Password) VALUES
```

```
    -> ('Admin1','23456'),
    -> ('Admin2','12378'),
    -> ('Admin3','76543');
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0
```



```
mysql> SELECT * FROM ADMINISTRATOR;
+----------+----------+
| Username | Password |
+----------+----------+
| Admin1   | 23456    |
| Admin2   | 12378    |
| Admin3   | 76543    |
+----------+----------+
3 rows in set (0.01 sec)
```

Figure 3.3 Inserting values into Tables using DML Commands for Your Topic

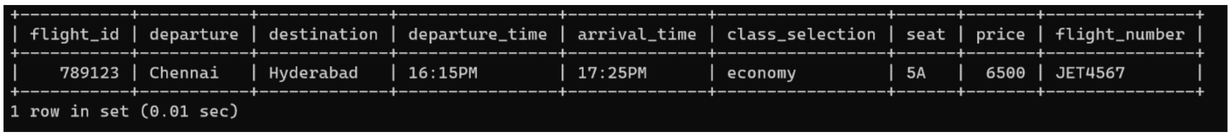
Chapter-3

Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors

Complex Queries :

- Sub Queries:
  1. Sub Query to find all flights that depart from a city that has a destination flight arriving after 5 PM.

```
SELECT f.*
FROM FLIGHT f
WHERE f.departure IN (
  SELECT departure
  FROM FLIGHT f2
  WHERE f2.arrival_time > '17:00:00'
);
```

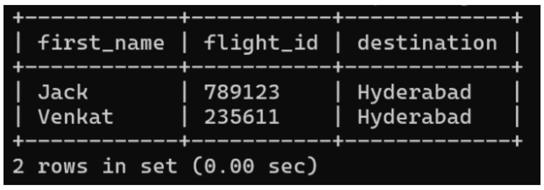


```
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
| flight_id | departure | destination | departure_time | arrival_time | class_selection | seat | price | flight_number |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
|    789123 | Chennai   | Hyderabad   | 16:15PM        | 17:25PM      | economy         | 5A   |  6500 | JET4567       |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
1 row in set (0.01 sec)
```

  2. Sub Query to find all clients who have booked flights to a specific destination city more than once.

```
SELECT b.first_name, b.flight_id, f.destination
FROM BOOKING b
JOIN FLIGHT f ON b.flight_id = f.flight_id  -- Join on flight_id
WHERE b.flight_id IN (  -- Flights going to Hyderabad
  SELECT flight_id
  FROM FLIGHT
  WHERE destination = 'Hyderabad'
)
GROUP BY b.first_name, b.flight_id, f.destination  -- Group by all three columns
```

ORDER BY b.first_name, b.flight_id; -- Optional ordering



```
+------------+-----------+-------------+
| first_name | flight_id | destination |
+------------+-----------+-------------+
| Jack       | 789123    | Hyderabad   |
| Venkat     | 235611    | Hyderabad   |
+------------+-----------+-------------+
2 rows in set (0.00 sec)
```

3. Sub Query for finding clients with bookings before a specific date to a specific destination

SELECT b.first_name, b.flight_id

FROM BOOKING b

WHERE b.flight_id IN (

  SELECT flight_id

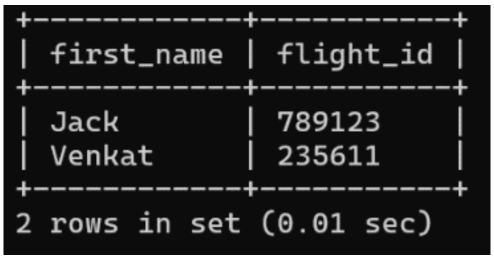
  FROM FLIGHT

  WHERE destination = 'Hyderabad'

)

AND b.booking_date < '2024-04-05'

GROUP BY b.first_name, b.flight_id;



```
+------------+-----------+
| first_name | flight_id |
+------------+-----------+
| Jack       | 789123    |
| Venkat     | 235611    |
+------------+-----------+
2 rows in set (0.01 sec)
```

4. Sub Query to find bookings with Maximum payment amount

SELECT b.*, p.Amount

FROM BOOKING b

JOIN PAYMENT p ON b.booking_id = p.Booking_id

WHERE b.booking_id IN (

    SELECT Booking_id

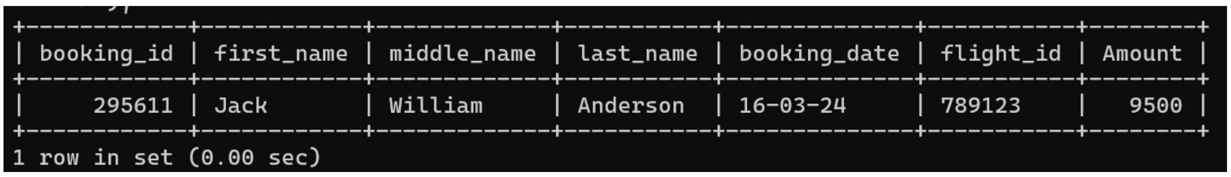
    FROM PAYMENT

    WHERE Amount = (

        SELECT MAX(Amount)

        FROM PAYMENT

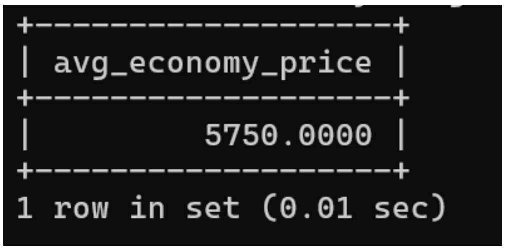
```
    )
 );
```



```
+------------+------------+-------------+-----------+--------------+-----------+--------+
| booking_id | first_name | middle_name | last_name | booking_date | flight_id | Amount |
+------------+------------+-------------+-----------+--------------+-----------+--------+
|     295611 | Jack       | William     | Anderson  | 16-03-24     | 789123    |   9500 |
+------------+------------+-------------+-----------+--------------+-----------+--------+
1 row in set (0.00 sec)
```

5. Sub Query to calculate average price of flights in economy class

```
SELECT AVG(price) AS avg_economy_price
FROM (
    SELECT price
    FROM FLIGHT
    WHERE class_selection = 'economy'
) AS economy_flights;
```



```
+-------------------+
| avg_economy_price |
+-------------------+
|         5750.0000 |
+-------------------+
1 row in set (0.01 sec)
```

- Constraints :

- **Query to give Primary Key Constraint for Booking ID in BOOKING table.**

ALTER TABLE BOOKING ADD CONSTRAINT PK_BOOKING_ID PRIMARY KEY (booking_id);

- **Query to give Foreign Key Constraint for flight ID in BOOKING table referencing flight ID in FLIGHT table.**

```
ALTER TABLE BOOKING ADD CONSTRAINT FK_FLIGHT_ID FOREIGN KEY (flight_id)
        REFERENCES FLIGHT(flight_id);
```

- **Query to have a unique Constraint on Username in the ADMINISTRATOR table.**

ALTER TABLE ADMINISTRATOR ADD CONSTRAINT UC_USERNAME UNIQUE (Username);

- **Query to have a Check Constraint on Price in the FLIGHT table to ensure it’s non-negetive.**

ALTER TABLE FLIGHT ADD CONSTRAINT CHK_PRICE CHECK (price >= 0);

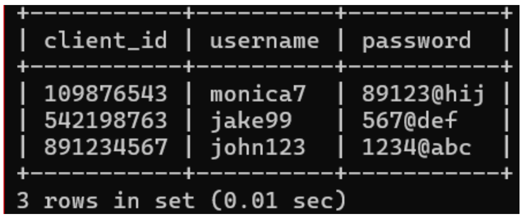
- **Query to have a Unique Constraint on client ID in the CLIENT table.**

ALTER TABLE CLIENT ADD CONSTRAINT UC_CLIENT_ID UNIQUE (client_id);

- Sets:

1. Query to see the clients who have made bookings.

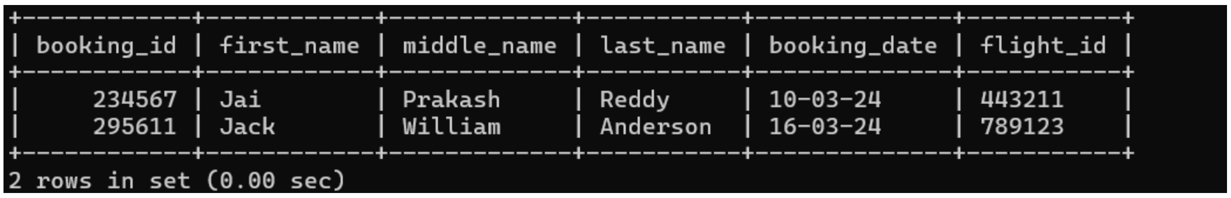
SELECT * FROM CLIENT WHERE client_id IN (SELECT DISTINCT client_id FROM BOOKING);



```
+-----------+----------+-----------+
| client_id | username | password  |
+-----------+----------+-----------+
| 109876543 | monica7  | 89123@hij |
| 542198763 | jake99   | 567@def   |
| 891234567 | john123  | 1234@abc  |
+-----------+----------+-----------+
3 rows in set (0.01 sec)
```

2. Query to find bookings made on or after March 1o, 2024.

SELECT * FROM BOOKING WHERE STR_TO_DATE(booking_date, '%d-%m-%y') >=

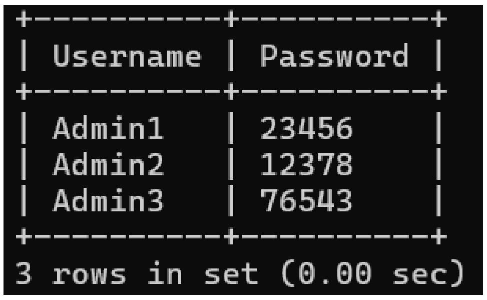


```
+------------+------------+-------------+-----------+--------------+-----------+
| booking_id | first_name | middle_name | last_name | booking_date | flight_id |
+------------+------------+-------------+-----------+--------------+-----------+
|     234567 | Jai        | Prakash     | Reddy     | 10-03-24     | 443211    |
|     295611 | Jack       | William     | Anderson  | 16-03-24     | 789123    |
+------------+------------+-------------+-----------+--------------+-----------+
2 rows in set (0.00 sec)
```

STR_TO_DATE('10-03-24', '%d-%m-%y');

3. Query to find the Admin usernames length greater than 5 characters.

SELECT * FROM ADMINISTRATOR WHERE LENGTH(Username) > 5;



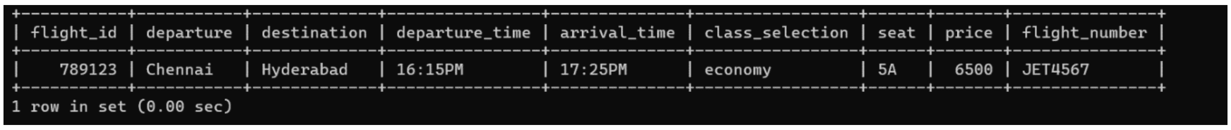
```
+----------+----------+
| Username | Password |
+----------+----------+
| Admin1   | 23456    |
| Admin2   | 12378    |
| Admin3   | 76543    |
+----------+----------+
3 rows in set (0.00 sec)
```

4. Query for flights departing from Chennai.

SELECT *

FROM FLIGHT

WHERE departure = 'Chennai';



```
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
| flight_id | departure | destination | departure_time | arrival_time | class_selection | seat | price | flight_number |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
|    789123 | Chennai   | Hyderabad   | 16:15PM        | 17:25PM      | economy         | 5A   |  6500 | JET4567       |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
1 row in set (0.00 sec)
```

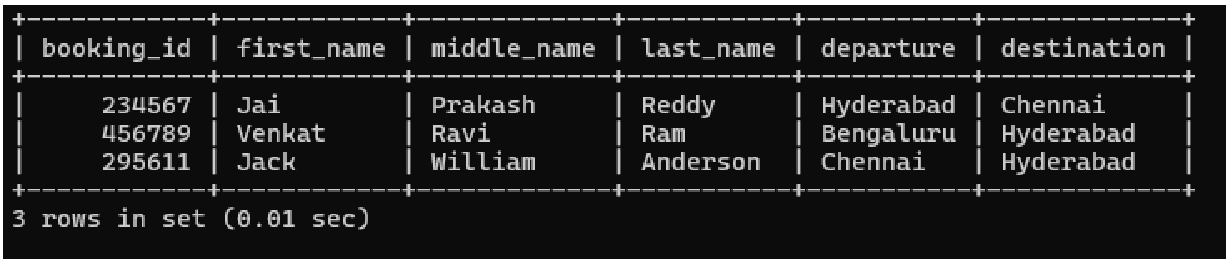
- Join:

1. Query for retrieving Booking Details with Flight Information.

SELECT B.booking_id, B.first_name, B.middle_name, B.last_name, F.departure, F.destination

FROM BOOKING B

INNER JOIN FLIGHT F ON B.flight_id = F.flight_id;



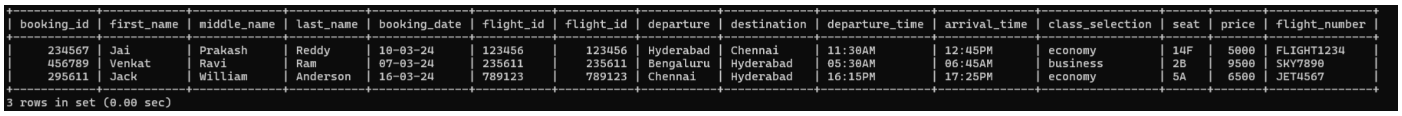
```
+------------+------------+-------------+-----------+-----------+-------------+
| booking_id | first_name | middle_name | last_name | departure | destination |
+------------+------------+-------------+-----------+-----------+-------------+
|     234567 | Jai        | Prakash     | Reddy     | Hyderabad | Chennai     |
|     456789 | Venkat     | Ravi        | Ram       | Bengaluru | Hyderabad   |
|     295611 | Jack       | William     | Anderson  | Chennai   | Hyderabad   |
+------------+------------+-------------+-----------+-----------+-------------+
3 rows in set (0.01 sec)
```

2. Query to retrieve all the bookings along with the corresponding flight information.

SELECT *

FROM BOOKING

INNER JOIN FLIGHT ON BOOKING.flight_id = FLIGHT.flight_id;



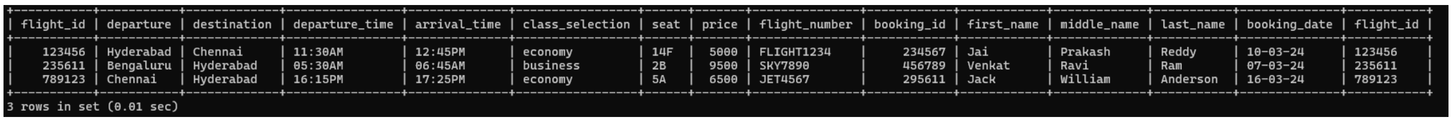
```
+------------+------------+-------------+-----------+--------------+-----------+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
| booking_id | first_name | middle_name | last_name | booking_date | flight_id | flight_id | departure | destination | departure_time | arrival_time | class_selection | seat | price | flight_number |
+------------+------------+-------------+-----------+--------------+-----------+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
|     234567 | Jai        | Prakash     | Reddy     | 10-03-24     | 123456    |    123456 | Hyderabad | Chennai     | 11:30AM        | 12:45PM      | economy         | 14F  |  5000 | FLIGHT1234    |
|     456789 | Venkat     | Ravi        | Ram       | 07-03-24     | 235611    |    235611 | Bengaluru | Hyderabad   | 05:30AM        | 06:45AM      | business        | 2B   |  9500 | SKY7890       |
|     295611 | Jack       | William     | Anderson  | 16-03-24     | 789123    |    789123 | Chennai   | Hyderabad   | 16:15PM        | 17:25PM      | economy         | 5A   |  6500 | JET4567       |
+------------+------------+-------------+-----------+--------------+-----------+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
3 rows in set (0.00 sec)
```

3. Query to retrieve all flights along with any corresponding bookings.

SELECT DISTINCT F.*, B.*

FROM FLIGHT F

LEFT JOIN BOOKING B ON F.flight_id = B.flight_id;



```
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+------------+------------+-------------+-----------+--------------+-----------+
| flight_id | departure | destination | departure_time | arrival_time | class_selection | seat | price | flight_number | booking_id | first_name | middle_name | last_name | booking_date | flight_id |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+------------+------------+-------------+-----------+--------------+-----------+
|    123456 | Hyderabad | Chennai     | 11:30AM        | 12:45PM      | economy         | 14F  |  5000 | FLIGHT1234    |     234567 | Jai        | Prakash     | Reddy     | 10-03-24     | 123456    |
|    235611 | Bengaluru | Hyderabad   | 05:30AM        | 06:45AM      | business        | 2B   |  9500 | SKY7890       |     456789 | Venkat     | Ravi        | Ram       | 07-03-24     | 235611    |
|    789123 | Chennai   | Hyderabad   | 16:15PM        | 17:25PM      | economy         | 5A   |  6500 | JET4567       |     295611 | Jack       | William     | Anderson  | 16-03-24     | 789123    |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+------------+------------+-------------+-----------+--------------+-----------+
3 rows in set (0.01 sec)
```

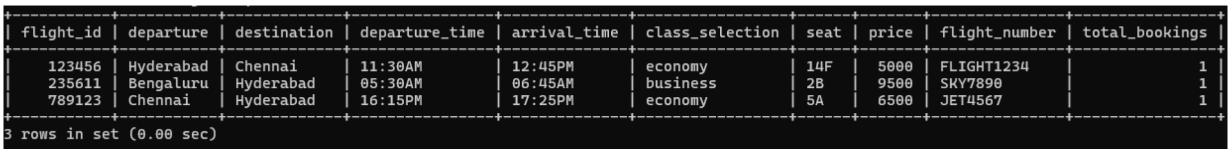
4. Query to retrieve flights along with the total number of bookings for each flight.

SELECT F.*, COUNT(B.booking_id) AS total_bookings

FROM FLIGHT F

LEFT JOIN BOOKING B ON F.flight_id = B.flight_id

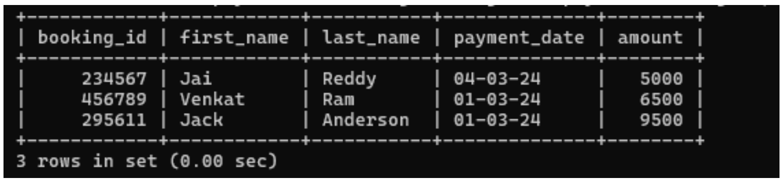
GROUP BY F.flight_id;



```
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+----------------+
| flight_id | departure | destination | departure_time | arrival_time | class_selection | seat | price | flight_number | total_bookings |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+----------------+
|    123456 | Hyderabad | Chennai     | 11:30AM        | 12:45PM      | economy         | 14F  |  5000 | FLIGHT1234    |              1 |
|    235611 | Bengaluru | Hyderabad   | 05:30AM        | 06:45AM      | business        | 2B   |  9500 | SKY7890       |              1 |
|    789123 | Chennai   | Hyderabad   | 16:15PM        | 17:25PM      | economy         | 5A   |  6500 | JET4567       |              1 |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+----------------+
3 rows in set (0.00 sec)
```

5. Query to retrieve booking details payment date and amount.

SELECT booking.booking_id, booking.first_name, booking.last_name, payment.payment_date, payment.amount
FROM booking
INNER JOIN payment ON booking.booking_id = payment.booking_id;

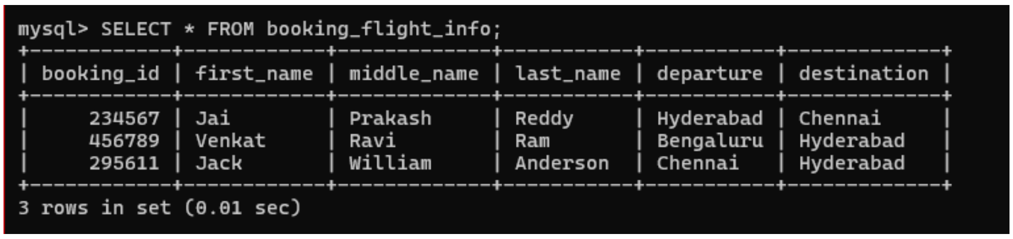


```
+------------+------------+-----------+--------------+--------+
| booking_id | first_name | last_name | payment_date | amount |
+------------+------------+-----------+--------------+--------+
|     234567 | Jai        | Reddy     | 04-03-24     |   5000 |
|     456789 | Venkat     | Ram       | 01-03-24     |   6500 |
|     295611 | Jack       | Anderson  | 01-03-24     |   9500 |
+------------+------------+-----------+--------------+--------+
3 rows in set (0.00 sec)
```

- Views:

1. View to display all bookings with corresponding flight information.

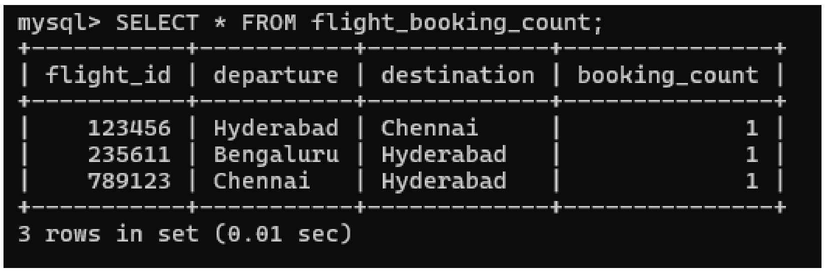
CREATE VIEW booking_flight_info AS
SELECT b.booking_id, b.first_name, b.middle_name, b.last_name, f.departure, f.destination
FROM booking b
INNER JOIN flight f ON b.flight_id = f.flight_id;



```
mysql> SELECT * FROM booking_flight_info;
+------------+------------+-------------+-----------+-----------+-------------+
| booking_id | first_name | middle_name | last_name | departure | destination |
+------------+------------+-------------+-----------+-----------+-------------+
|     234567 | Jai        | Prakash     | Reddy     | Hyderabad | Chennai     |
|     456789 | Venkat     | Ravi        | Ram       | Bengaluru | Hyderabad   |
|     295611 | Jack       | William     | Anderson  | Chennai   | Hyderabad   |
+------------+------------+-------------+-----------+-----------+-------------+
3 rows in set (0.01 sec)
```

2. View to show all flights along with their total number of bookings:

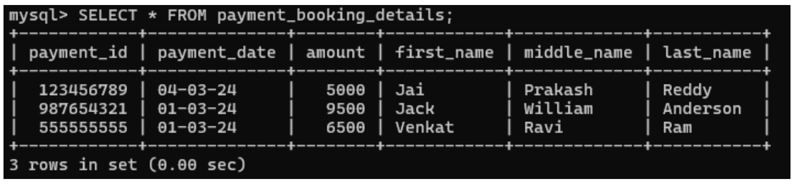
CREATE VIEW flight_booking_count AS
SELECT f.flight_id, f.departure, f.destination, COUNT(b.booking_id) AS booking_count
FROM flight f
LEFT JOIN booking b ON f.flight_id = b.flight_id
GROUP BY f.flight_id, f.departure, f.destination;



```
mysql> SELECT * FROM flight_booking_count;
+-----------+-----------+-------------+---------------+
| flight_id | departure | destination | booking_count |
+-----------+-----------+-------------+---------------+
|    123456 | Hyderabad | Chennai     |             1 |
|    235611 | Bengaluru | Hyderabad   |             1 |
|    789123 | Chennai   | Hyderabad   |             1 |
+-----------+-----------+-------------+---------------+
3 rows in set (0.01 sec)
```

3. View to list all payments made along with the corresponding booking details.

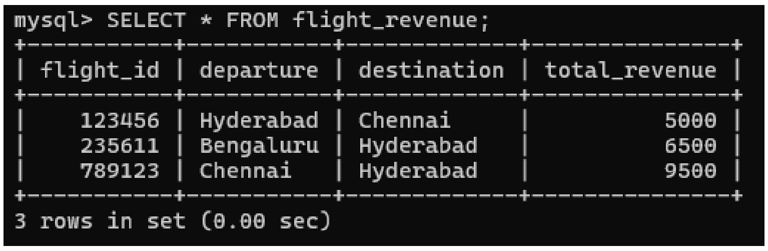
```
CREATE VIEW payment_booking_details AS
SELECT p.payment_id, p.payment_date, p.amount, b.first_name, b.middle_name, b.last_name
FROM payment p
INNER JOIN booking b ON p.booking_id = b.booking_id;
```



```
mysql> SELECT * FROM payment_booking_details;
+------------+--------------+--------+------------+-------------+-----------+
| payment_id | payment_date | amount | first_name | middle_name | last_name |
+------------+--------------+--------+------------+-------------+-----------+
|  123456789 | 04-03-24     |   5000 | Jai        | Prakash     | Reddy     |
|  987654321 | 01-03-24     |   9500 | Jack       | William     | Anderson  |
|  555555555 | 01-03-24     |   6500 | Venkat     | Ravi        | Ram       |
+------------+--------------+--------+------------+-------------+-----------+
3 rows in set (0.00 sec)
```

4. View to show the total revenue generated from each flight.

```
 CREATE VIEW flight_revenue AS
SELECT f.flight_id, f.departure, f.destination, SUM(p.amount) AS total_revenue
FROM flight f
LEFT JOIN booking b ON f.flight_id = b.flight_id
LEFT JOIN payment p ON b.booking_id = p.booking_id
GROUP BY f.flight_id, f.departure, f.destination;
```



```
mysql> SELECT * FROM flight_revenue;
+-----------+-----------+-------------+---------------+
| flight_id | departure | destination | total_revenue |
+-----------+-----------+-------------+---------------+
|    123456 | Hyderabad | Chennai     |          5000 |
|    235611 | Bengaluru | Hyderabad   |          6500 |
|    789123 | Chennai   | Hyderabad   |          9500 |
+-----------+-----------+-------------+---------------+
3 rows in set (0.00 sec)
```

- ## Triggers:

1. Trigger to update booking count on insert.

```
DELIMITER //
CREATE TRIGGER update_booking_count
AFTER INSERT ON BOOKING
FOR EACH ROW
BEGIN
    UPDATE FLIGHT
    SET booking_count = booking_count + 1
    WHERE flight_id = NEW.flight_id;
END;
//
```

```
DELIMITER ;
```

2. Trigger to update Booking count on delete.

```
DELIMITER //
CREATE TRIGGER update_booking_count_delete
AFTER DELETE ON BOOKING
FOR EACH ROW
BEGIN
   UPDATE FLIGHT
   SET booking_count = booking_count - 1
   WHERE flight_id = OLD.flight_id;
END;
//
DELIMITER ;
```

3. Trigger to prevent booking if flight capacity is exceeded.

```
DELIMITER //
CREATE TRIGGER prevent_booking_capacity
BEFORE INSERT ON BOOKING
FOR EACH ROW
BEGIN
   DECLARE total_bookings INT;
   SELECT COUNT(*) INTO total_bookings
   FROM BOOKING
   WHERE flight_id = NEW.flight_id;

   DECLARE max_capacity INT;
   SELECT capacity INTO max_capacity
   FROM FLIGHT
   WHERE flight_id = NEW.flight_id;

   IF total_bookings >= max_capacity THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Flight capacity exceeded, booking not allowed';
  END IF;
END;
```

```
//
DELIMITER ;
```

4. Trigger to calculate total revenue on booking payment.

```
DELIMITER //
CREATE TRIGGER update_total_revenue
AFTER INSERT ON PAYMENT
FOR EACH ROW
BEGIN
    UPDATE FLIGHT
    SET total_revenue = total_revenue + NEW.amount
    WHERE flight_id = (
        SELECT flight_id
        FROM BOOKING
        WHERE booking_id = NEW.booking_id
    );
END;
//
DELIMITER ;
```

5. Trigger on prevent deleting administrator account.

```
DELIMITER //
CREATE TRIGGER prevent_admin_deletion
BEFORE DELETE ON ADMINISTRATOR
FOR EACH ROW
BEGIN
    IF OLD.Username IN ('Admin1', 'Admin2', 'Admin3') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete administrator accounts';
    END IF;
END;
//
DELIMITER ;
```

- Cursors:

1. This stored procedure prints details for each booking in the “BOOKING” table.

```
DELIMITER $$

CREATE PROCEDURE print_booking_details()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE booking_id INT;
   DECLARE first_name VARCHAR(20);
   DECLARE middle_name VARCHAR(20);
   DECLARE last_name VARCHAR(20);
   DECLARE booking_date VARCHAR(40);
   DECLARE flight_id VARCHAR(20);

   DECLARE cur CURSOR FOR
      SELECT * FROM BOOKING;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;

   read_loop: LOOP
      FETCH cur INTO booking_id, first_name, middle_name, last_name, booking_date, flight_id;
      IF done THEN
         LEAVE read_loop;
      END IF;

      -- Print booking details
      SELECT CONCAT('Booking ID: ', booking_id, ', Name: ', first_name, ' ', middle_name, ' ', last_name,
        ', Booking Date: ', booking_date, ', Flight ID: ', flight_id);

   END LOOP;

   CLOSE cur;
END$$

DELIMITER ;
```

2. This stored procedure calculates the total payment amount for all bookings.

```
DELIMITER $$

CREATE PROCEDURE calculate_total_payments(OUT total_amount INT)
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE booking_amount INT;

   SET total_amount = 0;

   DECLARE cur CURSOR FOR
      SELECT Amount FROM PAYMENT;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;

   read_loop: LOOP
      FETCH cur INTO booking_amount;
      IF done THEN
         LEAVE read_loop;
      END IF;

      -- Accumulate total payment amount
      SET total_amount = total_amount + booking_amount;

   END LOOP;

   CLOSE cur;
END$$

DELIMITER ;
```

3. This stored procedure counts the number of bookings for each destination.

```
DELIMITER $$

CREATE PROCEDURE count_bookings_by_destination()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE dest VARCHAR(50);
   DECLARE booking_count INT;
   DECLARE cur CURSOR FOR
      SELECT destination FROM FLIGHT;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;

   read_loop: LOOP
   FETCH cur INTO dest;
      IF done THEN
         LEAVE read_loop;
      END IF;

      -- Count bookings for current destination
      SELECT COUNT(*) INTO booking_count FROM BOOKING WHERE flight_id IN (SELECT
        flight_id FROM FLIGHT WHERE destination = dest);
      SELECT CONCAT('Destination: ', dest, ', Booking Count: ', booking_count);

   END LOOP;

   CLOSE cur;
END$$

DELIMITER ;
```

4. This stored procedure calculates the total revenue generated from all the bookings.

```
DELIMITER $$

CREATE PROCEDURE calculate_total_revenue(OUT total_revenue DECIMAL(10,2))
BEGIN
```

```
    -- Calculate total revenue
    SELECT SUM(Amount) INTO total_revenue FROM PAYMENT;
END$$

DELIMITER ;
```

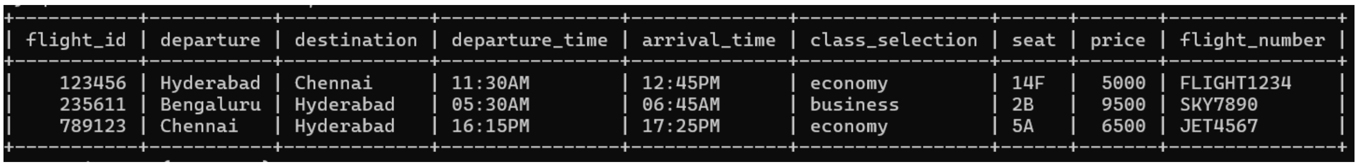
5. This stored procedure updates the price of a flight based on the flight ID.

```
DELIMITER $$

CREATE PROCEDURE update_flight_price_by_flight_id(
    IN flight_id_param INT,
    IN new_price INT
)
BEGIN
    -- Update flight price
    UPDATE FLIGHT
    SET price = new_price
    WHERE flight_id = flight_id_param;
END$$

DELIMITER ;
```

Chapter-4

Analyzing the pitfalls, identifying the dependencies, and applying normalizations

FLIGHT:



```
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
| flight_id | departure | destination | departure_time | arrival_time | class_selection | seat | price | flight_number |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
|    123456 | Hyderabad | Chennai     | 11:30AM        | 12:45PM      | economy         | 14F  |  5000 | FLIGHT1234    |
|    235611 | Bengaluru | Hyderabad   | 05:30AM        | 06:45AM      | business        | 2B   |  9500 | SKY7890       |
|    789123 | Chennai   | Hyderabad   | 16:15PM        | 17:25PM      | economy         | 5A   |  6500 | JET4567       |
+-----------+-----------+-------------+----------------+--------------+-----------------+------+-------+---------------+
```

```
CREATE TABLE FLIGHT (

    flight_id INT(10) NOT NULL PRIMARY KEY,

    departure VARCHAR(50) NOT NULL,

    destination VARCHAR(50) NOT NULL,

    departure_time VARCHAR(8) NOT NULL,

    arrival_time VARCHAR(8) NOT NULL,

    class_selection CHAR(30) NOT NULL,

    seat VARCHAR(4) NOT NULL,

    price INT(12) NOT NULL,

    flight_number VARCHAR(20) NOT NULL

);
```

Pitfalls:

Redundancy: There are no apparent issues with redundancy in the FLIGHT table. Each attribute appears to store unique information about a flight.

Inconsistency: There are no inconsistencies observed in the data structure provided.

Inefficiency: The table structure seems efficient for storing information about flights.

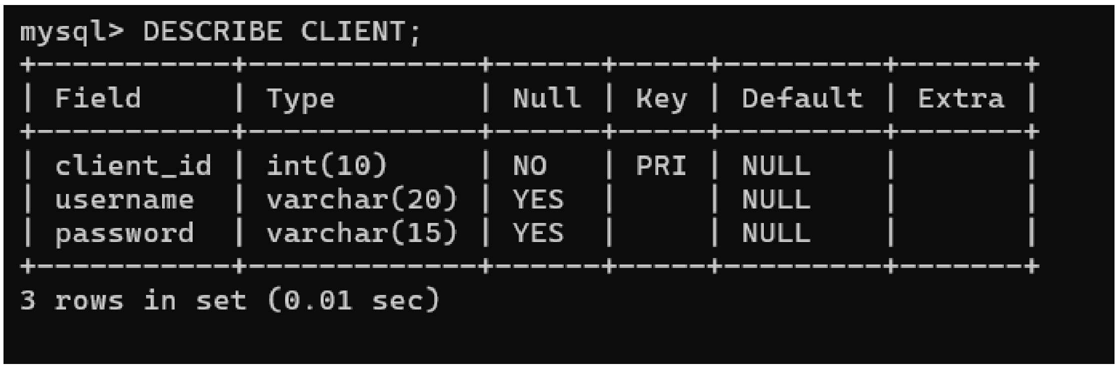
Complexity: The table structure is relatively straightforward and does not exhibit complexity issues.

Dependencies: In the FLIGHT table, there are no partial dependencies or transitive dependencies; each attribute is fully functionally dependent on the primary key flight_id.

From its pitfalls and dependencies there are no major issues so it satisfies 1NF to 5NF.

CLIENT:

```
mysql> CREATE TABLE CLIENT(
        -> client_id INT(10) PRIMARY KEY,
        -> username VARCHAR(20),
        -> password VARCHAR(15) );
```



```
mysql> DESCRIBE CLIENT;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| client_id | int(10)     | NO   | PRI | NULL    |       |
| username  | varchar(20) | YES  |     | NULL    |       |
| password  | varchar(15) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Pitfalls:

Redundancy: There are no apparent issues with redundancy in the CLIENT table. Each attribute appears to store unique information about a client.

Inconsistency: There are no inconsistencies observed in the data structure provided.

Inefficiency: The table structure seems efficient for storing information about clients.

Complexity: The table structure is relatively straightforward and does not exhibit complexity issues.

Dependencies:

In the `CLIENT` table, there are no partial dependencies or transitive dependencies both the username and password attributes are fully functionally dependent on the client_id primary key.

From its pitfalls and dependencies there are no major issues so it satisfies 1NF to 5NF.

PAYMENT:

Original table:

CREATE TABLE PAYMENT (

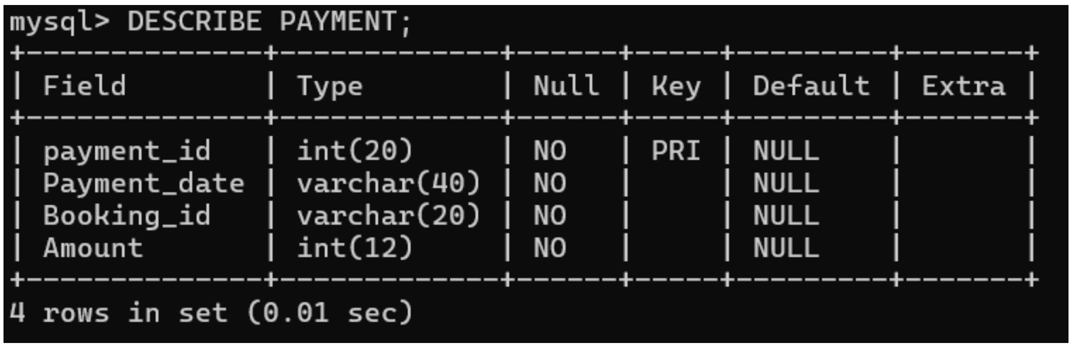
payment_id INT(20) NOT NULL PRIMARY KEY,

payment_date VARCHAR(40) NOT NULL,

booking_id VARCHAR(20) NOT NULL,

amount INT(12) NOT NULL

);



```
mysql> DESCRIBE PAYMENT;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| payment_id   | int(20)     | NO   | PRI | NULL    |       |
| Payment_date | varchar(40) | NO   |     | NULL    |       |
| Booking_id   | varchar(20) | NO   |     | NULL    |       |
| Amount       | int(12)     | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Pitfalls:

Redundancy: With the decomposition, redundancy is minimized as each table now stores information about distinct entities.

Inconsistency: The separation of payment details and booking associations into two tables helps maintain consistency within the database schema.

Inefficiency: While decomposition adds complexity, it improves efficiency by reducing redundant data storage and ensuring each table focuses on a single aspect of the payment process.

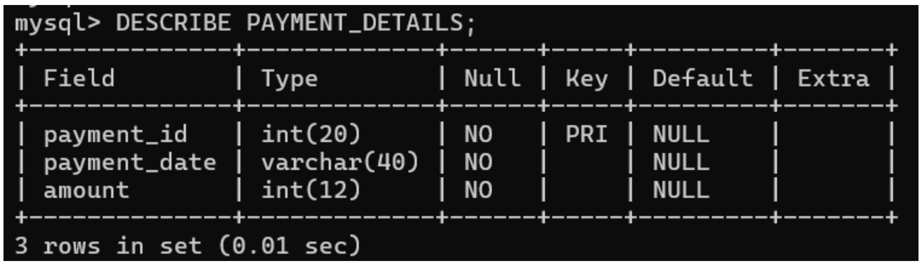
Complexity: The introduction of two tables might increase complexity, but it ensures better organization and adherence to normalization principles.

Dependencies:

In the original PAYMENT table, each payment ID uniquely determines the associated payment date, booking ID, and amount, but there is a transitive dependency with payment_id and booking_id.

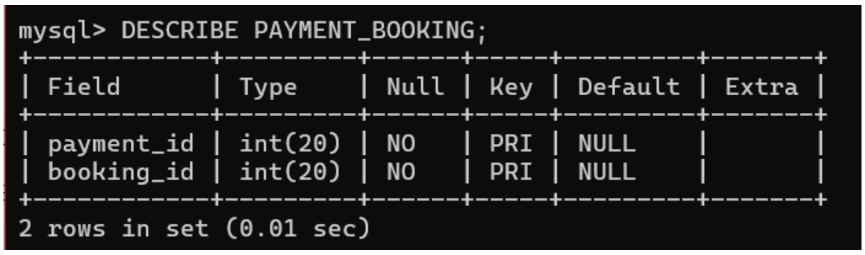
Normalized table:

```
CREATE TABLE PAYMENT_DETAILS (
    payment_id INT(20) NOT NULL PRIMARY KEY,
    payment_date VARCHAR(40) NOT NULL,
    amount INT(12) NOT NULL
);
```



```
mysql> DESCRIBE PAYMENT_DETAILS;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| payment_id   | int(20)     | NO   | PRI | NULL    |       |
| payment_date | varchar(40) | NO   |     | NULL    |       |
| amount       | int(12)     | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

```
CREATE TABLE PAYMENT_BOOKING (
    payment_id INT(20) NOT NULL,
    booking_id INT(20) NOT NULL,
    FOREIGN KEY (payment_id) REFERENCES PAYMENT_DETAILS(payment_id),
    FOREIGN KEY (booking_id) REFERENCES BOOKING(booking_id),
    PRIMARY KEY (payment_id, booking_id)
) ENGINE=InnoDB;
```



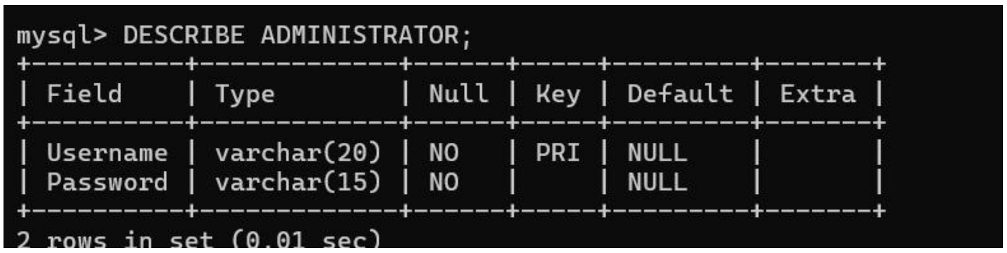
```
mysql> DESCRIBE PAYMENT_BOOKING;
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| payment_id | int(20) | NO   | PRI | NULL    |       |
| booking_id | int(20) | NO   | PRI | NULL    |       |
+------------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

This table had dependency required to be changed to Third Normal Form (3NF),

By decomposing PAYMENT table into PAYMENT_DETAILS and PAYMENT_BOOKING separate the payment details from booking associations, ensuring each table represents a distinct entity while maintaining data integrity.

ADMINISTRATOR:

```
mysql> CREATE TABLE ADMINISTRATOR(
    -> Username VARCHAR(20) NOT NULL PRIMARY KEY,
    -> Password VARCHAR(15) NOT NULL );
```



```
mysql> DESCRIBE ADMINISTRATOR;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Username | varchar(20) | NO   | PRI | NULL    |       |
| Password | varchar(15) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Pitfalls:

Redundancy: There are no apparent issues with redundancy in the ADMINISTRATOR table. Each attribute appears to store unique information about an administrator.

Inconsistency: There are no inconsistencies observed in the data structure provided.

Inefficiency: The table structure seems efficient for storing information about administrators.

Complexity: The table structure is relatively straightforward and does not exhibit complexity issues.

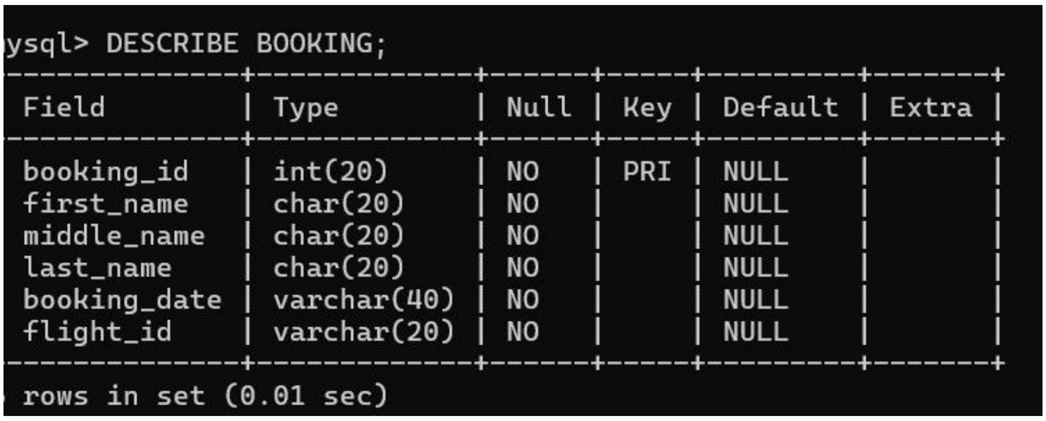
Dependencies: In the ADMINISTRATOR table, there are no partial dependencies or transitive dependencies; the attribute Password is fully functionally dependent on the candidate key Username, as each username uniquely determines its associated password.

From its pitfalls and dependencies there are no major issues so it satisfies 1NF to 5NF.

BOOKING:

Original table:

```
mysql> CREATE TABLE BOOKING(
    -> booking_id INT(20) NOT NULL PRIMARY KEY,
    -> first_name CHAR(20) NOT NULL,
    -> middle_name CHAR(20) NOT NULL,
    -> last_name CHAR(20) NOT NULL,
   -> booking_date VARCHAR(40) NOT NULL,
   -> flight_id VARCHAR(20) NOT
```



```
ysql> DESCRIBE BOOKING;
--------------+-------------+------+-----+---------+-------+
 Field        | Type        | Null | Key | Default | Extra |
--------------+-------------+------+-----+---------+-------+
 booking_id   | int(20)     | NO   | PRI | NULL    |       |
 first_name   | char(20)    | NO   |     | NULL    |       |
 middle_name  | char(20)    | NO   |     | NULL    |       |
 last_name    | char(20)    | NO   |     | NULL    |       |
 booking_date | varchar(40) | NO   |     | NULL    |       |
 flight_id    | varchar(20) | NO   |     | NULL    |       |
--------------+-------------+------+-----+---------+-------+
 rows in set (0.01 sec)
```

Pitfalls:

Redundancy: With the creation of the FLIGHT_BOOKING table to handle flight-related information, redundancy in the BOOKING table is effectively reduced. Each attribute in the BOOKING table seems to store unique information about a booking, minimizing redundancy.

Inconsistency: The separation of flight-related data into its own table maintains consistency within the database schema, ensuring that flight information is accurately associated with bookings.

Inefficiency: The table structure appears to be efficient for storing information about bookings, as flight-related data has been appropriately moved to a separate table.

Complexity: The separation of flight-related data simplifies the database structure and reduces complexity, making it easier to manage and maintain.

Dependencies:

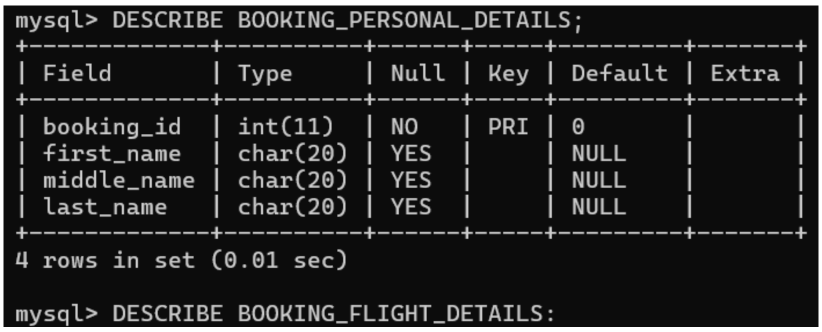
There is a functional dependency in the original BOOKING table: booking_id → flight_id. Each booking ID uniquely determines the associated flight ID.

The creation of the FLIGHT_BOOKING table effectively removes this dependency from the BOOKING table, improving normalization and reducing potential update anomalies.

Normalized table:

BOOKING_PERSONAL_DETAILS:

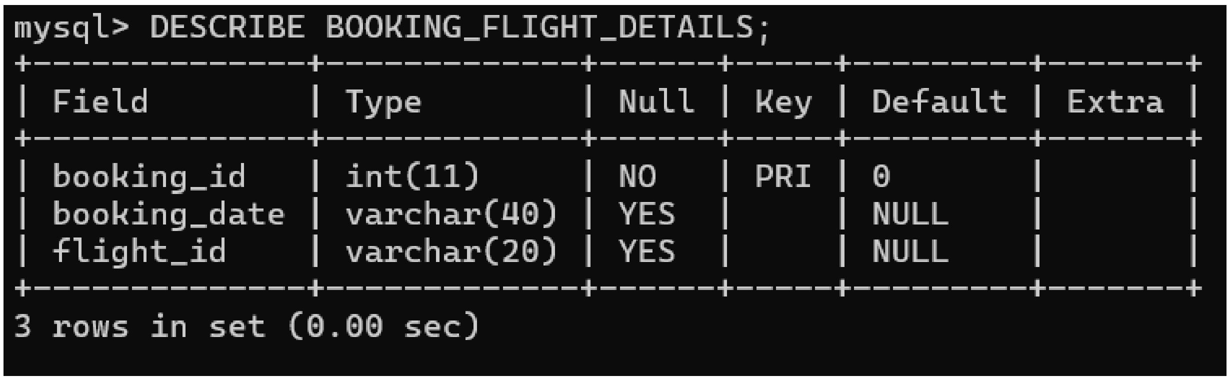
```
mysql> CREATE TABLE BOOKING_PERSONAL_DETAILS (
   ->     booking_id INT,
   ->     first_name CHAR(20),
   ->     middle_name CHAR(20),
   ->     last_name CHAR(20),
   ->     PRIMARY KEY (booking_id),
   ->     FOREIGN KEY (booking_id) REFERENCES BOOKING(booking_id)
   -> );
mysql> INSERT INTO BOOKING_PERSONAL_DETAILS (booking_id, first_name, middle_name, last_name)
   -> SELECT booking_id, first_name, middle_name, last_name
   -> FROM BOOKING;
```



```
mysql> DESCRIBE BOOKING_PERSONAL_DETAILS;
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| booking_id  | int(11)  | NO   | PRI | 0       |       |
| first_name  | char(20) | YES  |     | NULL    |       |
| middle_name | char(20) | YES  |     | NULL    |       |
| last_name   | char(20) | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> DESCRIBE BOOKING_FLIGHT_DETAILS:
```

BOOKING_FLIGHT_DETAILS:

```
mysql> -- Create the BOOKING_FLIGHT_DETAILS table
mysql> CREATE TABLE BOOKING_FLIGHT_DETAILS (
   ->     booking_id INT,
   ->     booking_date VARCHAR(40),
   ->     flight_id VARCHAR(20),
   ->     PRIMARY KEY (booking_id),
   ->     FOREIGN KEY (booking_id) REFERENCES BOOKING(booking_id)
   -> );
mysql> INSERT INTO BOOKING_FLIGHT_DETAILS (booking_id, booking_date, flight_id)
   -> SELECT booking_id, booking_date, flight_id
   -> FROM BOOKING;
```

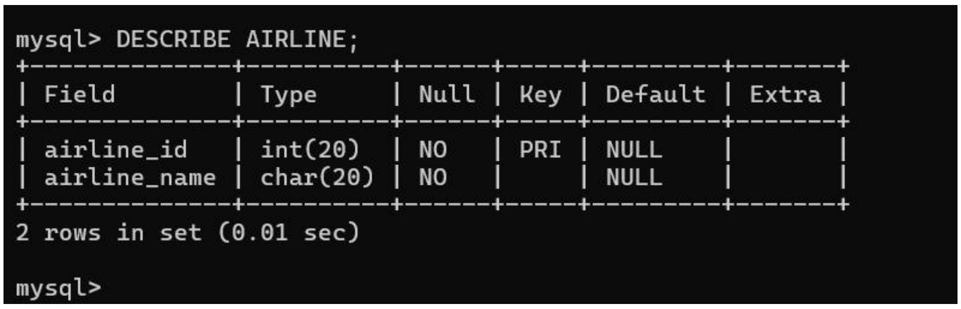


```
mysql> DESCRIBE BOOKING_FLIGHT_DETAILS;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| booking_id   | int(11)     | NO   | PRI | 0       |       |
| booking_date | varchar(40) | YES  |     | NULL    |       |
| flight_id    | varchar(20) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

The FLIGHT_BOOKING table needed to be changed to 4NF to prevent, by decomposing it into two tables BOOKING_PERSONAL_DETAILS and BOOKING_FLIGHT_DETAILS effectively addresses potential multivalued dependencies and aligns well with database normalization principles.

AIRLINE:

```
mysql> CREATE TABLE AIRLINE(
    -> airline_id INT(20) NOT NULL PRIMARY KEY,
    -> airline_name CHAR(20) NOT NULL );
```



```
mysql> DESCRIBE AIRLINE;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| airline_id   | int(20)  | NO   | PRI | NULL    |       |
| airline_name | char(20) | NO   |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql>
```

Pitfalls:

Redundancy: If multiple airlines have the same name, there could be redundancy in storing the airline names

repeatedly.
Inconsistency: Inconsistencies may arise if the same airline is referred to by different names in different records.
Complexity: While the table is simple, it may become complex if additional attributes are added in the future without proper normalization.

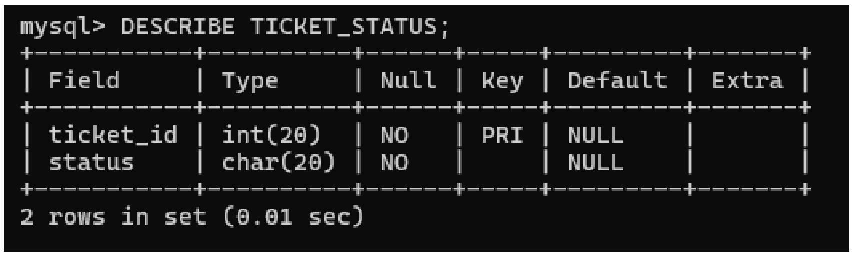
Dependencies:

Partial Dependency: There are no partial dependencies present in the airline table since each attribute is fully functionally dependent on the primary key (airline_id).
Transitive Dependency: No transitive dependencies are present as each attribute is directly dependent on the primary key.

From its pitfalls and dependencies there are no major issues so it satisfies 1NF to 5NF.

TICKET_STATUS:

```
mysql> CREATE TABLE TICKET_STATUS(
    -> ticket_id INT(20) NOT NULL PRIMARY KEY,
    -> status CHAR(20) NOT NULL );
```



```
mysql> DESCRIBE TICKET_STATUS;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| ticket_id | int(20)  | NO   | PRI | NULL    |       |
| status    | char(20) | NO   |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Pitfalls:

Redundancy: If multiple tickets share the same status, storing the status repeatedly may lead to redundancy.
Inconsistency: Inconsistencies can arise if the same ticket status is represented differently in different records.
Inefficiency: Storing the status as a string may lead to inefficiencies compared to using a numerical code for each status.

Dependencies:

Partial Dependency: There are no partial dependencies present in the ticket_status table since each attribute is fully functionally dependent on the primary key (ticket_id).

Transitive Dependency: No transitive dependencies are present as each attribute is directly dependent on the primary key.

From its pitfalls and dependencies there are no major issues so it satisfies 1NF to 5NF.

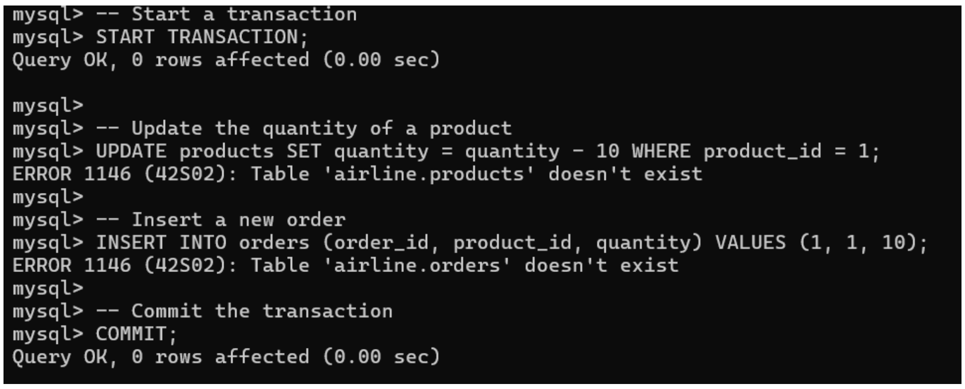
Chapter-5

Implementation of concurrency control and recovery mechanisms

5.1 Commit, Rollback and Savepoint code

```
-- Start a transaction

START TRANSACTION;

-- Update the quantity of a product

UPDATE products SET quantity = quantity - 10 WHERE product_id = 1;

-- Insert a new order

INSERT INTO orders (order_id, product_id, quantity) VALUES (1, 1, 10);

-- Commit the transaction
COMMIT;


-- Start a transaction
START TRANSACTION;

-- Update the quantity of a product
UPDATE products SET quantity = quantity - 10 WHERE product_id = 1;

-- Insert a new order
INSERT INTO orders (order_id, product_id, quantity) VALUES (1, 1, 10);

-- Rollback the transaction in case of an error
ROLLBACK;
-- Start a transaction
START TRANSACTION;

-- Create a savepoint
SAVEPOINT my_savepoint;

-- Update the quantity of a product
UPDATE products SET quantity = quantity - 10 WHERE product_id = 1;

-- Insert a new order
```

```
INSERT INTO orders (order_id, product_id, quantity) VALUES (1, 1, 10);

-- Rollback to the savepoint if necessary
ROLLBACK TO SAVEPOINT my_savepoint;

-- Continue with other statements
-- If no rollback is needed, commit the transaction
COMMIT;
```



```
mysql> -- Start a transaction
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> -- Update the quantity of a product
mysql> UPDATE products SET quantity = quantity - 10 WHERE product_id = 1;
ERROR 1146 (42S02): Table 'airline.products' doesn't exist
mysql>
mysql> -- Insert a new order
mysql> INSERT INTO orders (order_id, product_id, quantity) VALUES (1, 1, 10);
ERROR 1146 (42S02): Table 'airline.orders' doesn't exist
mysql>
mysql> -- Commit the transaction
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

5.2 ACID

Atomicity: The code utilizes transactions to ensure that a series of operations either all succeed (commit) or all fail (rollback). For example, when inserting new data into tables or updating existing records, these operations are grouped within transactions. If any part of the transaction fails, all changes made by the transaction are rolled back, preserving atomicity.

Consistency: Database constraints such as foreign key constraints and unique constraints ensure data consistency by enforcing rules about the relationships between tables and the uniqueness of data values.

These constraints help maintain the integrityof the data and prevent inconsistencies.

Isolation: Transactions are executed in isolation from each other, meaning that changes made by one transaction are not visible to other transactions until the changes are committed. This isolation prevents interference between concurrent transactions and maintains data integrity.

Durability: Once a transaction is committed, the changes made by the transaction are permanently saved in the database even in the event of a system failure. Thisdurability is ensured by the database system's logging and recovery mechanisms, which maintain a record of committed transactions to recover data in case of failure.

Overall, by using transactions and database constraints, the code adheres to the principles of ACID to ensure data integrity, consistency, and reliability.

Chapter-6

Code for the Project

```
<?php include_once 'helpers/helper.php'; ?>
<?php subview('header.php'); ?>

<link rel="stylesheet" href="assets/css/login.css">
<style>
@font-face {
  font-family: 'product sans';
  src: url('assets/css/Product Sans Bold.ttf');
}
h1{
   font-family :'product sans' !important;
            font-size:48px !important;
            margin-top:20px;
            text-align:center;
}
body {
    background: -webkit-linear-gradient(left, #3931af, #00c6ff);
}
.login-form {
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
    border-radius: 0px;
}
</style>
<?php
if(isset($_GET['err']) || isset($_GET['pwd'])) {
    if($_GET['err'] === 'pwdnotmatch') {
        echo '<script>alert("Passwords do not match");</script>';
    } else if($_GET['err'] === 'sqlerr') {
        echo '<script>alert("An error occured");</script>';
    } else if($_GET['pwd'] === 'updated') {
        echo '<script>alert("Your password has been updated");</script>';
    }
    exit();
}
?>
<div class="flex-container">
<div class="login-form mt-4" style="height: 300px;">
    <h1 class="text-primary mb-3 text-center">Reset Password</h1>
    <?php
    $selector = $_GET['selector'];
    $validator = $_GET['validator'];
    if(empty($selector) || empty($validator)){
        // echo $_GET;
        echo '<script>alert("Could not validate your request")</script>';
    } else {
        if(ctype_xdigit($selector) !== false && ctype_xdigit($validator) !== false){
            ?>
            <form method="POST" action="includes/reset-password.inc.php">
                <input type="hidden" name="selector" value="<?php echo $selector ?>">
                <input type="hidden" name="validator" value="<?php echo $validator ?>">
                <div class="flex-container">
                    <div>
                        <i class="fa fa-lock text-primary"></i>
                    </div>
                    <div>
                        <input type="password" name="password" class="form-input"
                            placeholder="Enter password"
                            required pattern="(?=.\d)(?=.[a-z])(?=.*[A-Z]).{8,}"
                            title="Must contain at least one number and one uppercase and lowercase letter,
                            and at least 8 or more characters">
                    </div>
                </div>
                <div class="flex-container">
                    <div>
                        <i class="fa fa-lock text-primary"></i>
                    </div>
                    <div>
                        <input type="password" name="password_repeat" class="form-input"
                            placeholder="Confirm password" required>
                    </div>
                </div>
                <div class="submit">
                <button name="new-pwd-submit" type="submit" class="button">
                    Submit</button>
                </div>
            </form>
            <?php
            }
    }
    ?>
</div>
</div>
<?php subview('footer.php'); ?>

<?php include_once 'helpers/helper.php'; ?>
<?php subview('header.php');    ?>
<link rel="stylesheet" href="assets/css/form.css">
<style>

.rating {
  display: inline-block;
  position: relative;
  height: 50px;
  line-height: 50px;
  font-size: 50px;
}
.rating label {
  position: absolute;
  top: 0;
  left: 0;
  height: 100%;
  cursor: pointer;
}

.rating label:last-child {
  position: static;
}
```

```
.rating label:nth-child(1) {
  z-index: 5;
}

.rating label:nth-child(2) {
  z-index: 4;
}

.rating label:nth-child(3) {
  z-index: 3;
}

.rating label:nth-child(4) {
  z-index: 2;
}

.rating label:nth-child(5) {
  z-index: 1;
}

.rating label input {
  position: absolute;
  top: 0;
  left: 0;
  opacity: 0;
}

.rating label .icon {
  float: left;
  color: transparent;
}

.rating label:last-child .icon {
  color: #000;
}

.rating:not(:hover) label input:checked ~ .icon,
.rating:hover label:hover input ~ .icon {
  color: #09f;
}

.rating label input:focus:not(:checked) ~ .icon:last-child {
  color: #000;
  text-shadow: 0 0 5px #09f;
}

@font-face {
  font-family: 'product sans';
  src: url('assets/css/Product Sans Bold.ttf');
  }
h1 {
  font-size: 50px !important;
  margin-bottom: 20px;
  color: #393939;
  font-family :'product sans' !important;
  text-align: center;
}

textarea {
  color: cornflowerblue !important;
  border :3px solid #31B0D5 !important;
  background-color: whitesmoke !important;
  font-weight: bold !important;
}
textarea:focus {
  outline-style: none !important;
  outline: none !important;
}
*:focus {
    outline: none !important;
}
input {
    border :0px !important;
    border-bottom: 2px solid #31B0D5 !important;
    color :cornflowerblue !important;
    border-radius: 0px !important;
    font-weight: bold !important;
    border: none;
    border-bottom: 2px solid #31B0D5;
  }
  label {
    color : #79BAEC !important;
    font-size: 19px;
  }
  div.form-group label {
    color: cornflowerblue !important;
    font-weight: bold;
  }
  div.rating label{
    font-size: 40px !important;
  }
.input-group {
  position: relative;
  display: inline-block;
  width: 100%;
}
.form-box {
  padding: 40px;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
}
</style>

<main>
<?php
if(isset($_GET['error'])) {
    if($_GET['error'] === 'invalidemail') {
        echo '<script>alert("Invalid email")</script>';
    } else if($_GET['error'] === 'sqlerror') {
        echo"<script>alert('Database error')</script>";
    } else if($_GET['error'] === 'success') {
      echo"<script>alert('Thank you for your Feedback')</script>";
    }
}
?>
```

```
<div class="container mb-4">
  <h1> <i class="far fa-comment-alt"></i> FEEDBACK</h1>
  <div class="row justify-content-center">
  <div class="col-md-6 bg-light form-box">
    <form action="includes/feedback.inc.php" method="POST">
      <div class="row justify-content-center">
          <div class="col-12 ">
            <div class="input-group">
                <label for="user_id">Email</label>
                <input type="text" name="email" id="user_id" required >
              </div>
          </div>
          <div class="col-12 mt-4">
            <div class="form-group">
              <label for="exampleFormControlTextarea1">What was your first impression
                  when you entered the website?</label>
              <textarea class="form-control" id="exampleFormControlTextarea1" name="1"
                rows="3" required></textarea>
            </div>
          </div>

          <div class="col-12 mt-4">
            <div class="form-group">
              <select class="mt-4" name="2" style="border: 0px; border-bottom:
              2px solid #31B0D5; background-color: whitesmoke !important;
              font-weight: bold !important;color :cornflowerblue !important;
              width:100%" required>
                <option  selected disabled>How did you first hear about us?</option>
                <option >Search Engine</option>
                <option >Social Media</option>
                <option >Friend/Relative</option>
                <option >Word of Mouth</option>
                <option >Television</option>
                <option>Other</option>
              </select>
            </div>
          </div>
      <div class="col-12 mt-4">
        <div class="form-group">
          <label for="exampleFormControlTextarea1">Is there anything missing on this page?</label>
          <textarea class="form-control" id="exampleFormControlTextarea1" name="3"
            rows="3" required></textarea>
        </div>
      </div>
    </div>

    <div class="row">
      <div class="rating ml-3">
        <label>
          <input type="radio" name="stars" value="1" required />
          <span class="icon">★</span>
        </label>
        <label>
          <input type="radio" name="stars" value="2" required />
          <span class="icon">★</span>
          <span class="icon">★</span>
        </label>
        <label>
          <input type="radio" name="stars" value="3" required />
          <span class="icon">★</span>
          <span class="icon">★</span>
          <span class="icon">★</span>
        </label>
        <label>
          <input type="radio" name="stars" value="4" required />
          <span class="icon">★</span>
          <span class="icon">★</span>
          <span class="icon">★</span>
          <span class="icon">★</span>
        </label>
        <label>
          <input type="radio" name="stars" value="5" required />
          <span class="icon">★</span>
          <span class="icon">★</span>
          <span class="icon">★</span>
          <span class="icon">★</span>
          <span class="icon">★</span>
        </label>
      </div>
    </div>
    <div class="row">
      <div class="col text-center">
        <button name="feed_but" type="submit"
          class="btn btn-primary mt-3">
          <div style="font-size: 1.5rem;">
          <i class="fa fa-lg fa-arrow-right"></i>
          </div>
        </button>
      </div>
    </div>

   </form>
  </div>
  </div>
</div>
<?php subview('footer.php'); ?>
<script>
 $(document).ready(function(){
 $('.input-group input').focus(function(){
   me = $(this) ;
   $("label[for='"+me.attr('id')+"']").addClass("animate-label");
 }) ;
 $('.input-group input').blur(function(){
   me = $(this) ;
   if ( me.val() == ""){
     $("label[for='"+me.attr('id')+"']").removeClass("animate-label");
   }
 }) ;
 // $('#test-form').submit(function(e){
 //   e.preventDefault() ;
 //   alert("Thank you") ;
```

```
  // })
});
</script>
</main>
<footer>
            <em><h5 class=" text-center p-0 brand mt-2">
                                    <img src="assets/images/plane-logo.png"
                                            height="40px" width="40px" alt="">
                        </h5></em>
            <p class=" text-center">&copy; <?php echo date('Y');?> - Online Flight Booking</p>

th {
  font-size: 22px;
  /* font-family: 'Courier New', Courier, monospace; */
}
td {
  margin-top: 10px !important;
  font-size: 16px;
  font-weight: bold;
  /* color: #3931af; */
  color: #424242;
}
</style>
    <main>
        <?php if(isset($_POST['search_but'])) {
            $dep_date = $_POST['dep_date'];
            $ret_date = isset($_POST['ret_date'])? $_POST['ret_date'] : '';
            $dep_city = $_POST['dep_city'];
            $arr_city = $_POST['arr_city'];
            $type = $_POST['type'];
            $f_class = $_POST['f_class'];
            $passengers = $_POST['passengers'];
            if($dep_city === $arr_city){
              header('Location: index.php?error=sameval');
              exit();
            }
```

```
<?php include_once 'helpers/helper.php'; ?>
<?php subview('header.php');
require 'helpers/init_conn_db.php';
?>
<link href="https://fonts.googleapis.com/css2?family=Assistant:wght@200&display=swap"
rel="stylesheet">
<style>
table {
  background-color: white;
}
@font-face {
  font-family: 'product sans';
  src: url('assets/css/Product Sans Bold.ttf');
}
h1{
    font-family :'product sans' !important;
            color:#424242 ;
            font-size:40px !important;
            margin-top:20px;
            text-align:center;
}

th {
  font-size: 22px;
  /* font-family: 'Courier New', Courier, monospace; */
}
td {
  margin-top: 10px !important;
  font-size: 16px;
  font-weight: bold;
  /* color: #3931af; */
  color: #424242;
}
</style>
```

```
<main>
    <?php if(isset($_POST['search_but'])) {
        $dep_date = $_POST['dep_date'];
        $ret_date = isset($_POST['ret_date'])? $_POST['ret_date'] : '';
        $dep_city = $_POST['dep_city'];
        $arr_city = $_POST['arr_city'];
        $type = $_POST['type'];
        $f_class = $_POST['f_class'];
        $passengers = $_POST['passengers'];
        if($dep_city === $arr_city){
          header('Location: index.php?error=sameval');
          exit();
        }
        if($dep_city === '0') {
          header('Location: index.php?error=seldep');
          exit();
        }
        if($arr_city === '0') {
          header('Location: index.php?error=selarr');
          exit();
        }
        ?>
      <div class="container-md mt-2">
        <h1 class="display-4 text-center "
          >FLIGHTS FROM: <br> <?php echo $dep_city; ?>
             to <?php echo $arr_city; ?> </h1>
        <table class="table table-striped table-bordered table-hover">
          <thead>
            <tr class="text-center">
              <th scope="col">Airline</th>
              <th scope="col">Departure</th>
              <th scope="col">Arrival</th>
              <th scope="col">Status</th>
              <th scope="col">Fare</th>
              <th scope="col">Buy</th>
```

```
  </tr>
</thead>
<tbody>
  <?php
  $sql = 'SELECT * FROM Flight WHERE source=? AND Destination =? AND
      DATE(departure)=? ORDER BY Price';
  $stmt = mysqli_stmt_init($conn);
  mysqli_stmt_prepare($stmt,$sql);
  mysqli_stmt_bind_param($stmt,'sss',$dep_city,$arr_city,$dep_date);
  mysqli_stmt_execute($stmt);
  $result = mysqli_stmt_get_result($stmt);
  while ($row = mysqli_fetch_assoc($result)) {
    $price = (int)$row['Price']*(int)$passengers;
    if($type === 'round') {
      $price = $price*2;
    }
    if($f_class == 'B') {
        $price += 0.5*$price;
    }
    if($row['status'] === '') {
        $status = "Not yet Departed";
        $alert = 'alert-primary';
    } else if($row['status'] === 'dep') {
        $status = "Departed";
        $alert = 'alert-info';
    } else if($row['status'] === 'issue') {
        $status = "Delayed";
        $alert = 'alert-danger';
    } else if($row['status'] === 'arr') {
        $status = "Arrived";
        $alert = 'alert-success';
    }
    echo "
    <tr class='text-center'>
      <td>".$row['airline']."</td>
```

```
 <td>".$row['departure']."</td>
 <td>".$row['arrivale']."</td>
 <td>
   <div>
       <div class='alert ".$alert." text-center mb-0 pt-1 pb-1'
           role='alert'>
           ".$status."
       </div>
   </div>
 </td>
 <td>$ ".$price."</td>
 ";
if(isset($_SESSION['userId']) && $row['status'] === '') {
 echo " <td>
 <form action='pass_form.php' method='post'>
 <input name='flight_id' type='hidden' value=".$row['flight_id'].">
   <input name='type' type='hidden' value=".$type.">
   <input name='passengers' type='hidden' value=".$passengers.">
   <input name='price' type='hidden' value=".$price.">
   <input name='ret_date' type='hidden' value=".$ret_date.">
   <input name='class' type='hidden' value=".$f_class.">
   <button name='book_but' type='submit'
   class='btn btn-success mt-0'>
   <div style=''>
   <i class='fa fa-lg fa-check'></i>
   </div>
 </button>
 </form>
 </td>
 ";
} elseif (isset($_SESSION['userId']) && $row['status'] === 'dep') {
                                   echo "<td>Not Available</td>";
                          } else {
 echo "<td>Login to continue</td>";
}
```

```
                echo '</tr> ';
              }
              ?>

            </tbody>
          </table>

        </div>
      <?php } ?>

  </main>
  <?php subview('footer.php'); ?>
  <footer style="
      position: absolute;
    bottom: 0;
    width: 100%;
    height: 2.5rem;
  ">
        <em><h5 class=" text-center p-0 brand mt-2">
                                    <img src="assets/images/plane-logo.png"
                                          height="40px" width="40px" alt="">
                              </h5></em>
        <p class=" text-center">&copy; <?php echo date('Y');?> - Online Flight Booking</p>
</footer>
<?php include_once 'helpers/helper.php'; ?>

<?php subview('header.php'); ?>
<?php if(isset($_SESSION['userId'])) {
  require 'helpers/init_conn_db.php';
?>
<style>
body {
 background: transparent;  /* fallback for old browsers */

}
```

```
@font-face {
  font-family: 'product sans';
  src: url('assets/css/Product Sans Bold.ttf');
}
div.out {
    padding: 30px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
    border-top-left-radius: 20px;
    border-bottom-right-radius: 20px;
}
.city {
    font-size: 24px;
}
p {
    margin-bottom: 10px;
    font-family: product sans;
}
.alert {
    /* font-family: 'Courier New', Courier, monospace; */
    font-weight: bold;
}
.date {
    font-size: 24px;
}
.time {
    font-size: 27px;
    margin-bottom: 0px;
}
.stat {
    font-size: 17px;
}
h1 {
    font-weight: lighter !important;
    font-size: 45px !important;
    margin-bottom: 20px;
    font-family :'product sans' !important;
    font-weight: bolder;
  }
.row {
    background-color: white;
}
@font-face {
  font-family: 'product sans';
  src: url('assets/css/Product Sans Bold.ttf');
  }

</style>
<main>
    <div class="container">
    <h1 class="text-center  mt-4 mb-4">FLIGHT STATUS</h1>
    <?php
    $stmt_t = mysqli_stmt_init($conn);
    $sql_t = 'SELECT * FROM Ticket WHERE user_id=?';
    $stmt_t = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt_t,$sql_t)) {
        header('Location: ticket.php?error=sqlerror');
        exit();
    } else {
        mysqli_stmt_bind_param($stmt_t,'i',$_SESSION['userId']);
        mysqli_stmt_execute($stmt_t);
        $result_t = mysqli_stmt_get_result($stmt_t);
        while($row_t = mysqli_fetch_assoc($result_t)) {
            $stmt = mysqli_stmt_init($conn);
            $sql = 'SELECT * FROM Passenger_profile WHERE passenger_id=?';
            $stmt = mysqli_stmt_init($conn);
            if(!mysqli_stmt_prepare($stmt,$sql)) {
                header('Location: my_flights.php?error=sqlerror');
                exit();
            } else {
                mysqli_stmt_bind_param($stmt,'i',$row_t['passenger_id']);
```

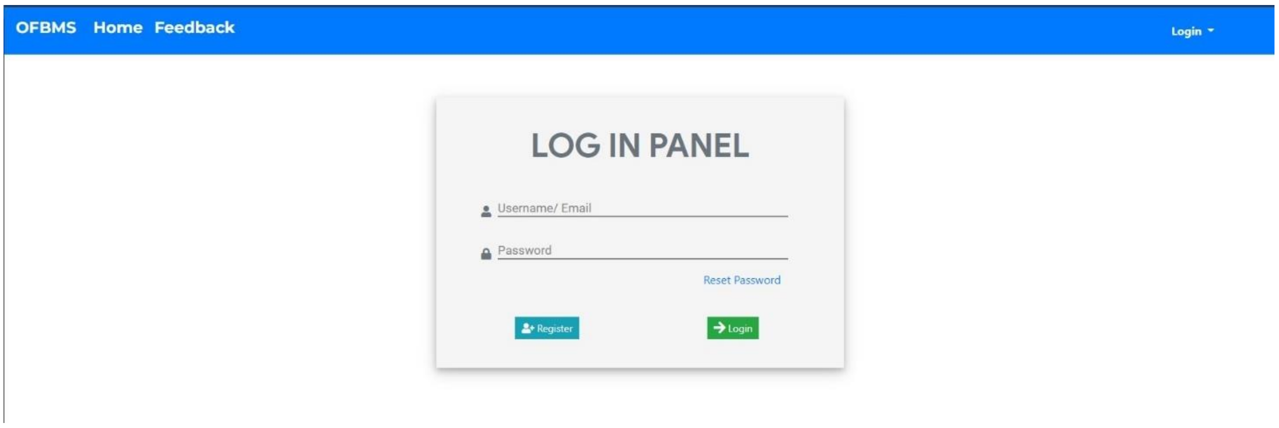
```
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
if ($row = mysqli_fetch_assoc($result)) {
   $sql_f = 'SELECT * FROM Flight WHERE flight_id=? ';
   $stmt_f = mysqli_stmt_init($conn);
   if(!mysqli_stmt_prepare($stmt_f,$sql_f)) {
      header('Location: my_flights.php?error=sqlerror');
      exit();
   } else {
      mysqli_stmt_bind_param($stmt_f,'i',$row_t['flight_id']);
      mysqli_stmt_execute($stmt_f);
      $result_f = mysqli_stmt_get_result($stmt_f);
      if($row_f = mysqli_fetch_assoc($result_f)) {
         $date_time_dep = $row_f['departure'];
         $date_dep = substr($date_time_dep,0,10);
         $time_dep = substr($date_time_dep,10,6) ;
         $date_time_arr = $row_f['arrivale'];
         $date_arr = substr($date_time_arr,0,10);
         $time_arr = substr($date_time_arr,10,6) ;
         if($row_f['status'] === '') {
            $status = "Not yet Departed";
            $alert = 'alert-primary';
         } else if($row_f['status'] === 'dep') {
            $status = "Departed";
            $alert = 'alert-info';
         } else if($row_f['status'] === 'issue') {
            $status = "Delayed";
            $alert = 'alert-danger';
         } else if($row_f['status'] === 'arr') {
            $status = "Arrived";
            $alert = 'alert-success';
         }
         echo '
         <div class="row out mb-5 ">
            <div class="col-md-4 order-lg-3 order-md-1"> ';
```

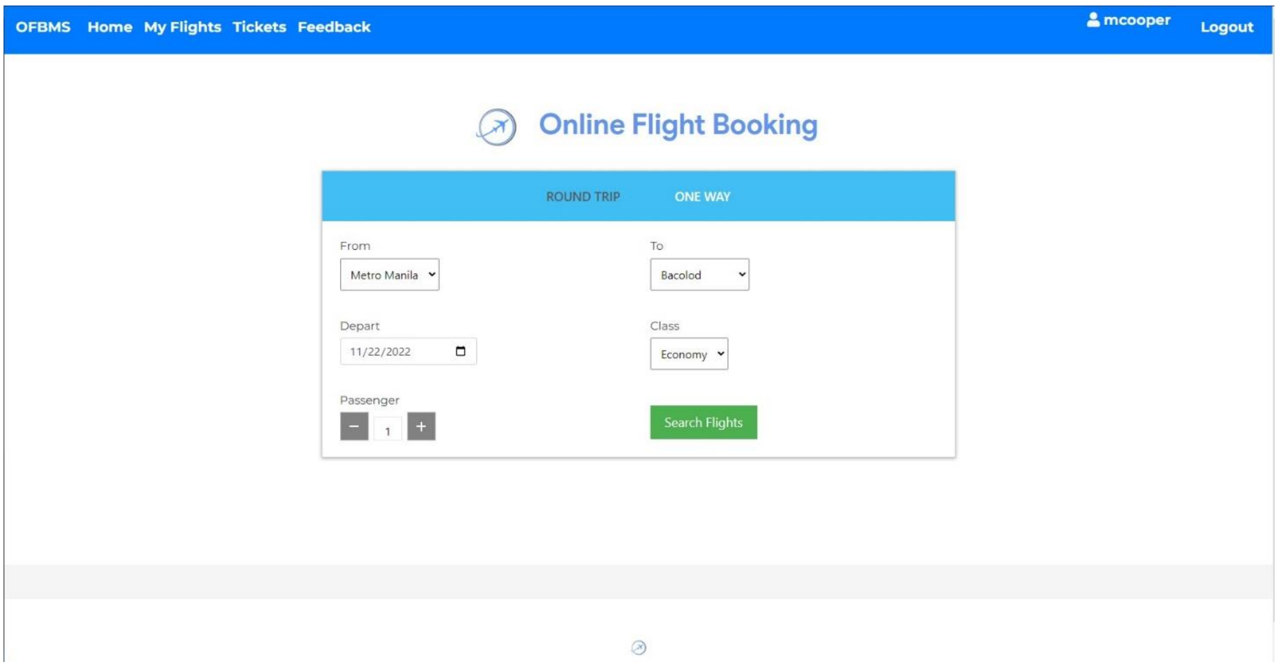
```
if($row_f['status'] === 'arr') {
    echo '
    <div class="row">
        <div class="col-1 p-0 m-0">
            <i class="fa fa-circle mt-4 text-success"
                style="float: right;"></i>
        </div>
        <div class="col-10 p-0 m-0 mt-3" style="float: right;">
            <hr class="bg-success">
        </div>
        <div class="col-1 p-0 m-0">
            <i class="fa fa-2x fa-fighter-jet mt-3 text-success"
                ></i>
        </div>
    </div>
    ';
} else {
    echo '
    <div class="row">
        <div class="col-1 p-0 m-0">
            <i class="fa fa-2x fa-fighter-jet mt-3 text-success"
                style="float: right;"></i>
        </div>
        <div class="col-10 p-0 m-0 mt-3" style="float: right;">
            <hr style="background-color: lightgrey;">
        </div>
        <div class="col-1 p-0 m-0">
            <i class="fa fa-circle mt-4"
                style="color: lightgrey;"></i>
        </div>
    </div>
    ';
}
    echo '
</div>
```

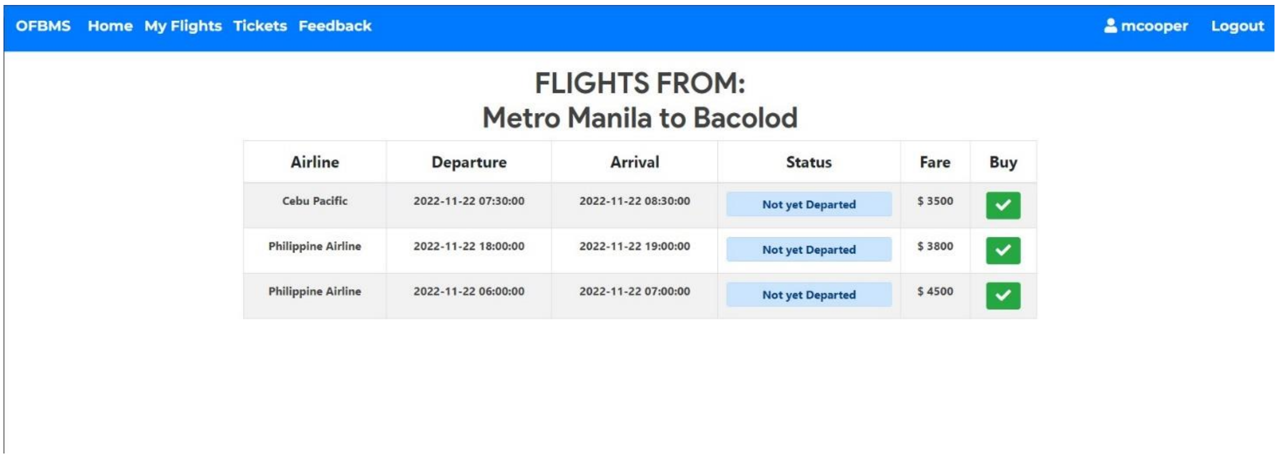
```
                    <div class="col-md-3 col-6 order-md-2 pl-0 text-center
                        order-lg-2 card-dep">
                        <p class="city">'.$row_f['source'].'</p>
                        <p class="stat">Scheduled Departure:</p>
                        <p class="date">'.$date_dep.'</p>
                        <p class="time">'.$time_dep.'</p>
                    </div>
                    <div class="col-md-3 col-6 order-md-4 pr-0 text-center
                        order-lg-4 card-arr"
                        style="float: right;">
                        <p class="city">'.$row_f['Destination'].'</p>
                        <p class="stat">Scheduled Arrival:</p>
                        <p class="date">'.$date_arr.'</p>
                        <p class="time">'.$time_arr.'</p>
                    </div>
                    <div class="col-lg-2 order-md-12">
                        <div class="alert '.$alert.' mt-5 text-center"
                            role="alert">
                            '.$status.'
                        </div>
                    </div>
                </div> ';
              }
            }
          }
        }
      }
    }
    ?>
</div>

</main>
<?php } ?>
<?php subview('footer.php'); ?>
```

Chapter-7

Results and Discussions

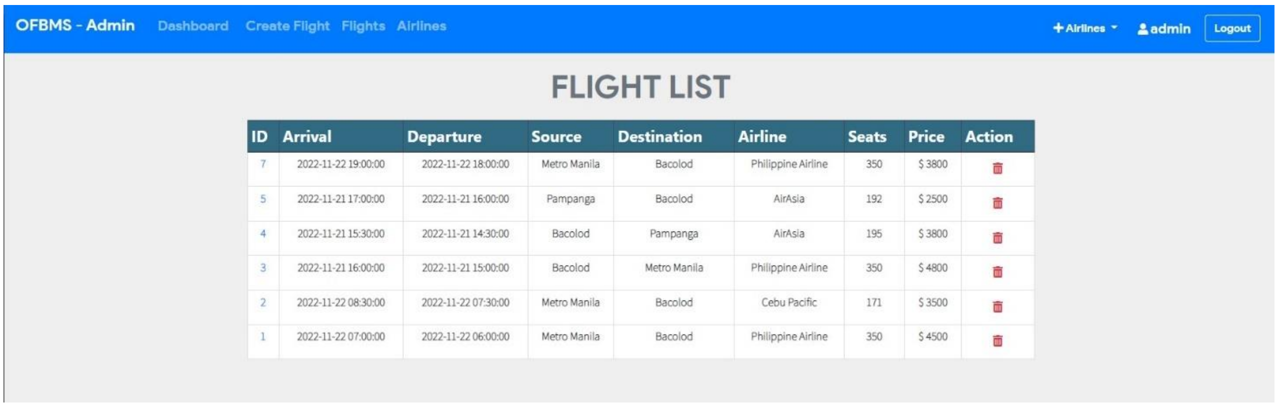






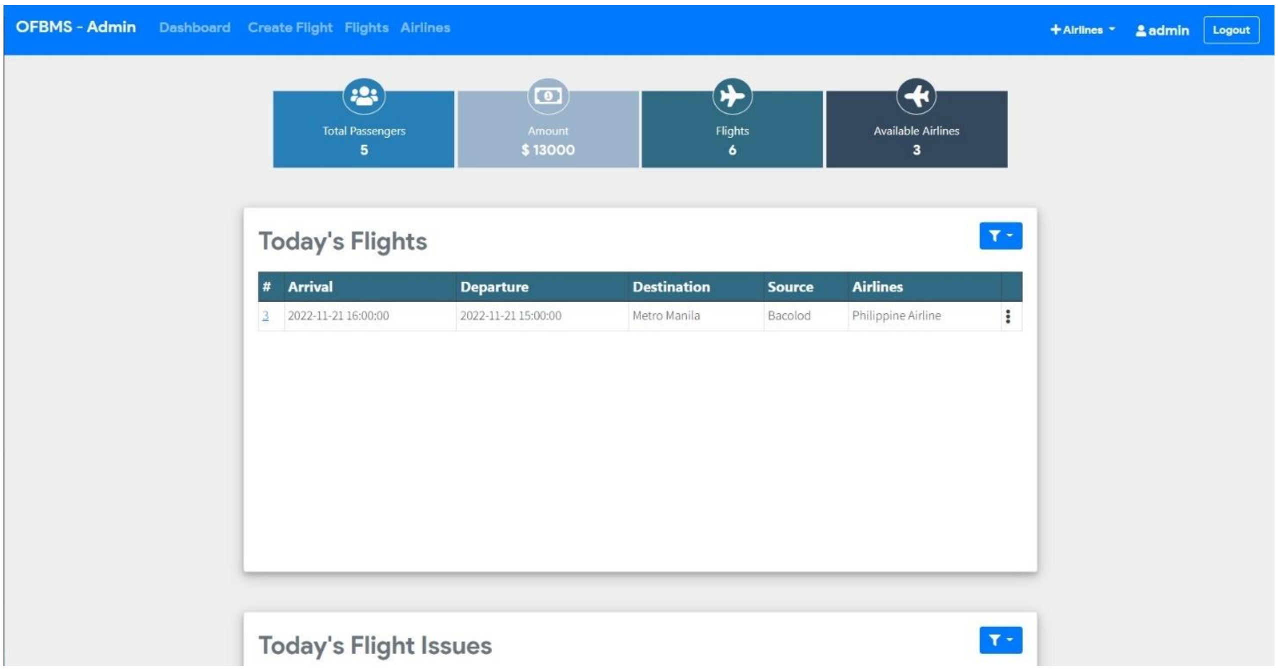
FLIGHTS FROM:
Metro Manila to Bacolod

Airline	Departure	Arrival	Status	Fare	Buy
Cebu Pacific	2022-11-22 07:30:00	2022-11-22 08:30:00	Not yet Departed	$ 3500	✓
Philippine Airline	2022-11-22 18:00:00	2022-11-22 19:00:00	Not yet Departed	$ 3800	✓
Philippine Airline	2022-11-22 06:00:00	2022-11-22 07:00:00	Not yet Departed	$ 4500	✓



FLIGHT LIST

ID	Arrival	Departure	Source	Destination	Airline	Seats	Price	Action
7	2022-11-22 19:00:00	2022-11-22 18:00:00	Metro Manila	Bacolod	Philippine Airline	350	$ 3800	🗑
5	2022-11-21 17:00:00	2022-11-21 16:00:00	Pampanga	Bacolod	AirAsia	192	$ 2500	🗑
4	2022-11-21 15:30:00	2022-11-21 14:30:00	Bacolod	Pampanga	AirAsia	195	$ 3800	🗑
3	2022-11-21 16:00:00	2022-11-21 15:00:00	Bacolod	Metro Manila	Philippine Airline	350	$ 4800	🗑
2	2022-11-22 08:30:00	2022-11-22 07:30:00	Metro Manila	Bacolod	Cebu Pacific	171	$ 3500	🗑
1	2022-11-22 07:00:00	2022-11-22 06:00:00	Metro Manila	Bacolod	Philippine Airline	350	$ 4500	🗑

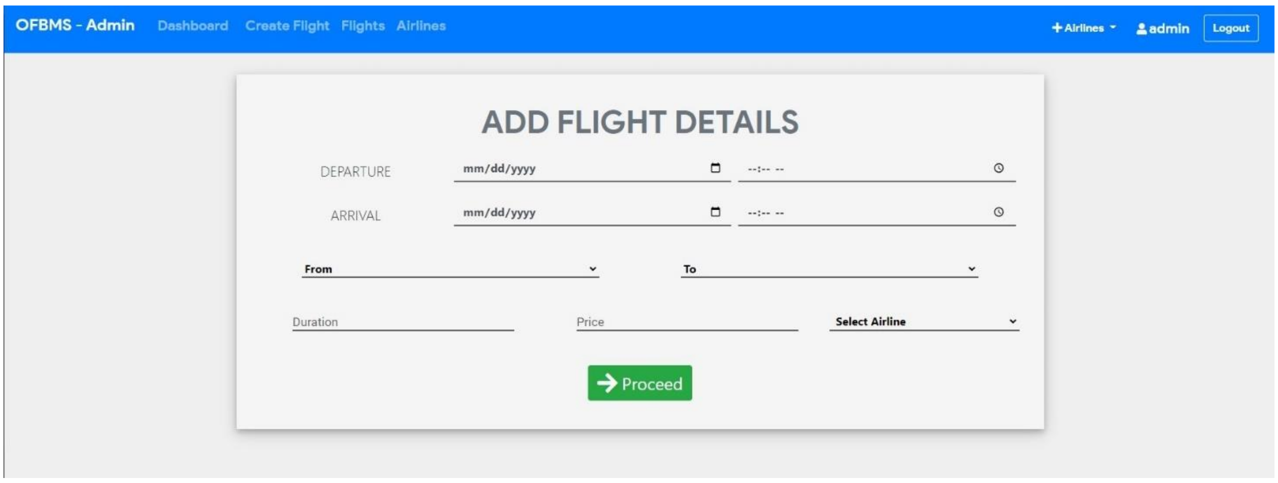


Total Passengers 5
Amount $ 13000
Flights 6
Available Airlines 3

Today's Flights

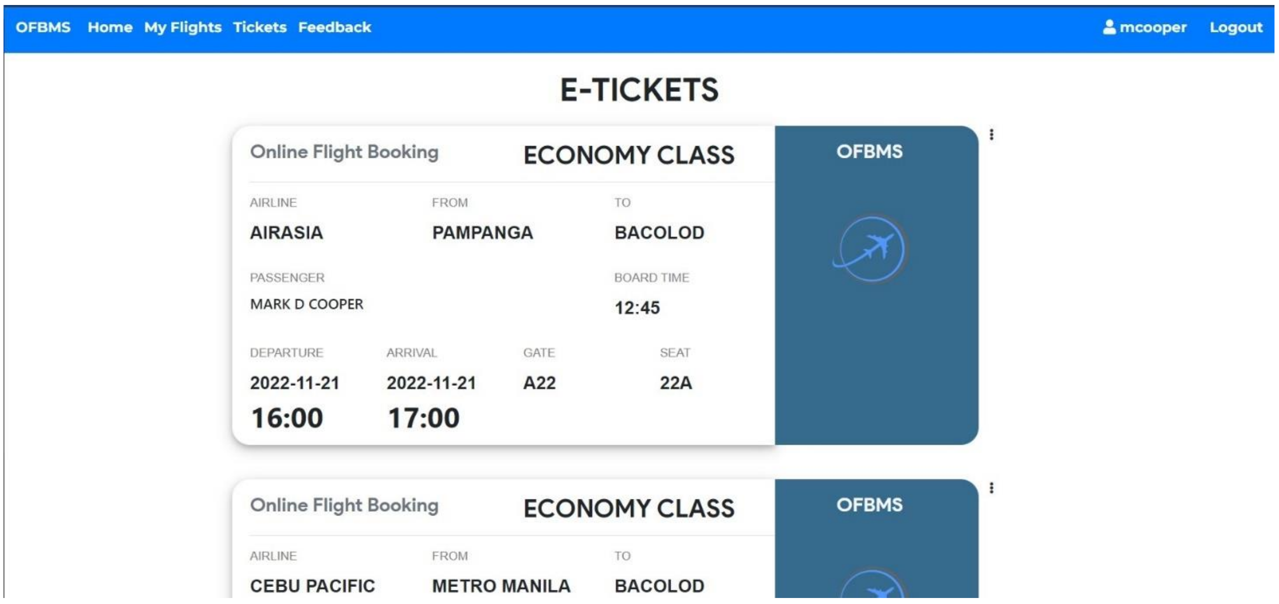
#	Arrival	Departure	Destination	Source	Airlines
3	2022-11-21 16:00:00	2022-11-21 15:00:00	Metro Manila	Bacolod	Philippine Airline

Today's Flight Issues



ADD FLIGHT DETAILS

DEPARTURE mm/dd/yyyy --:-- --
ARRIVAL mm/dd/yyyy --:-- --
From
To
Duration
Price
Select Airline



E-TICKETS

Online Flight Booking ECONOMY CLASS

OFBMS

AIRLINE	FROM	TO
AIRASIA	PAMPANGA	BACOLOD

PASSENGER
MARK D COOPER

BOARD TIME
12:45

DEPARTURE	ARRIVAL	GATE	SEAT
2022-11-21	2022-11-21	A22	22A
16:00	17:00		

Online Flight Booking ECONOMY CLASS

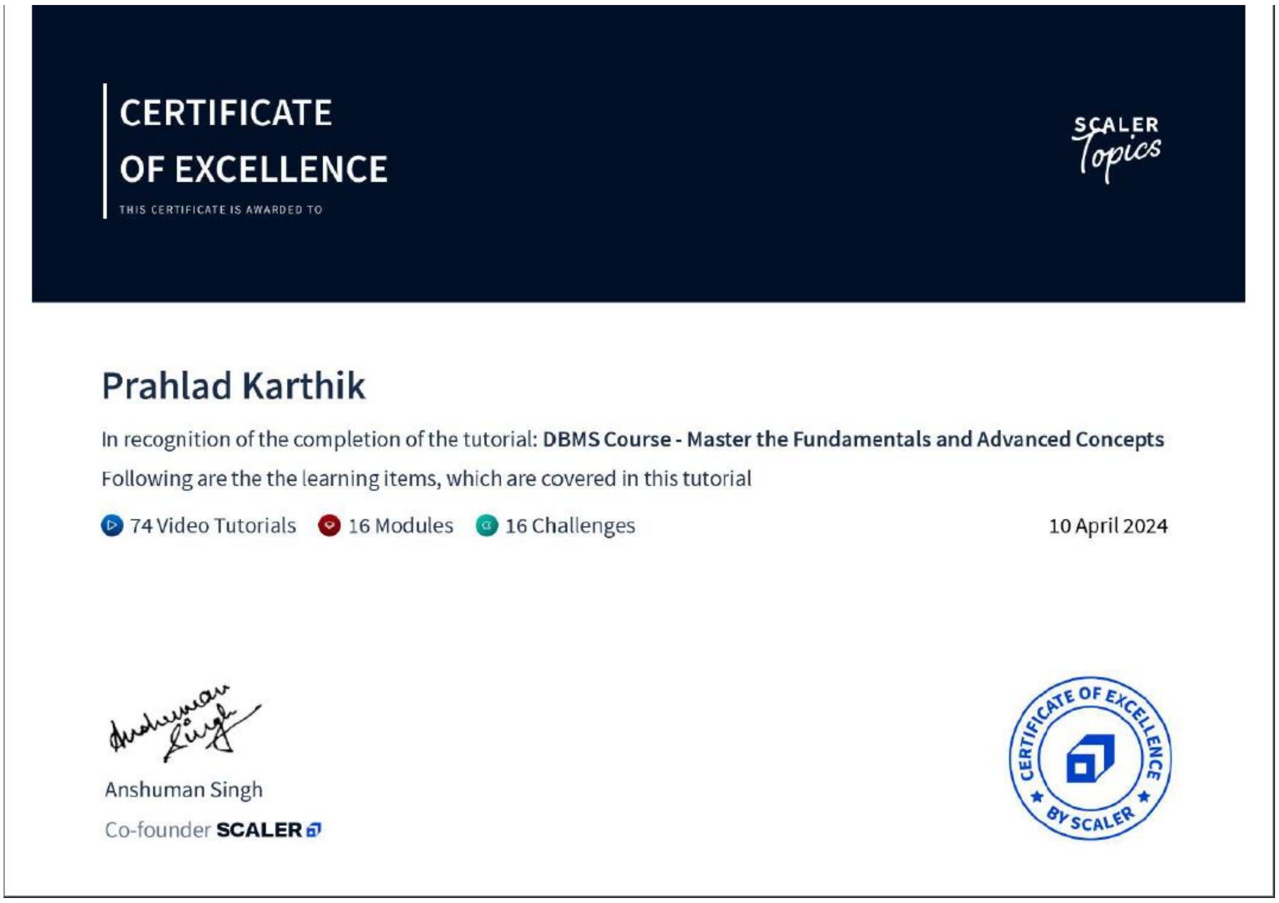
OFBMS

AIRLINE	FROM	TO
CEBU PACIFIC	METRO MANILA	BACOLOD

Chapter-8

Online Course Certificate

1. **M V Prahlad Karthik :**



CERTIFICATE
OF EXCELLENCE
THIS CERTIFICATE IS AWARDED TO

SCALER Topics

Prahlad Karthik

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

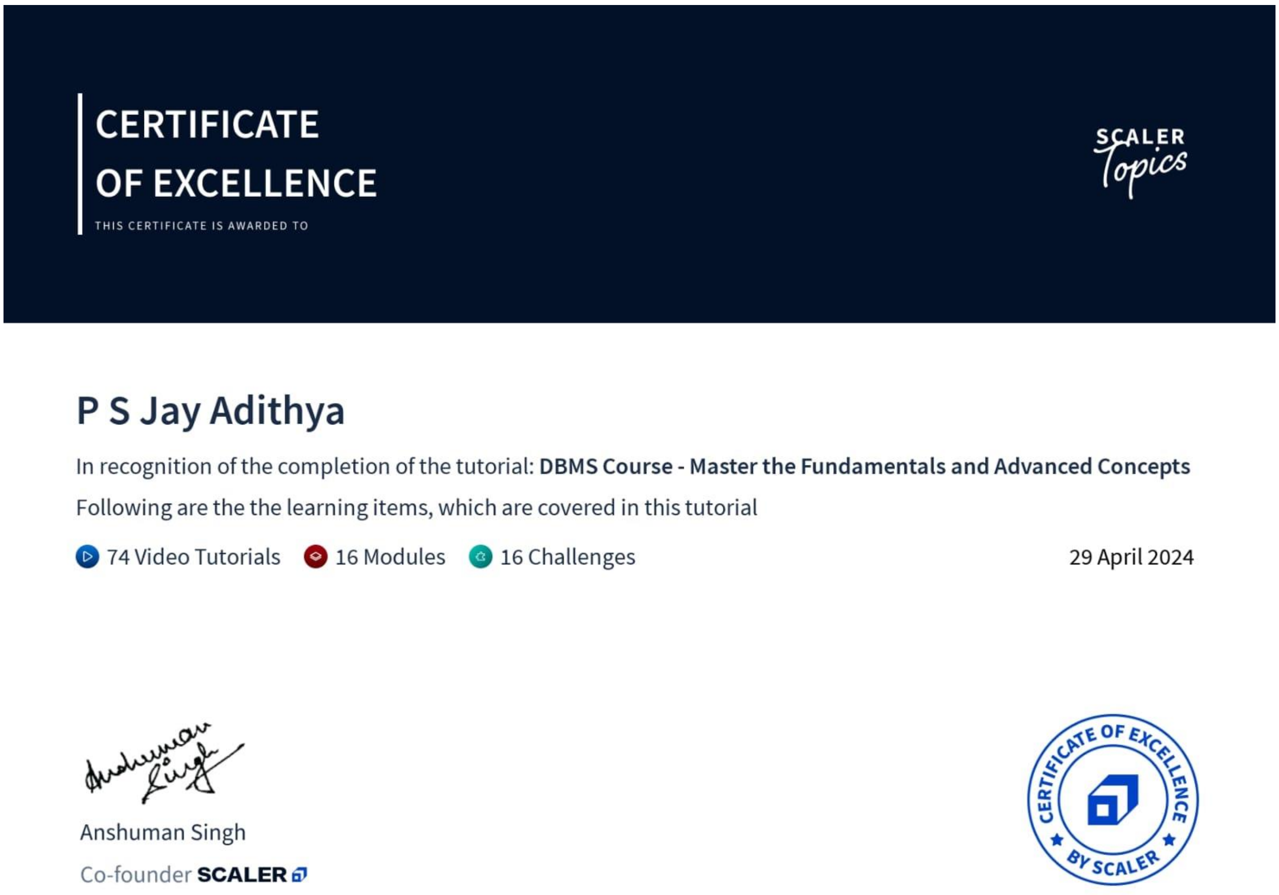
74 Video Tutorials 16 Modules 16 Challenges 10 April 2024

[illegible]

Anshuman Singh
Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE ★ BY SCALER ★

2. Jay Adithya:



CERTIFICATE
OF EXCELLENCE
THIS CERTIFICATE IS AWARDED TO

SCALER Topics

P S Jay Adithya

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 29 April 2024

[illegible]

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE ★ BY SCALER ★

65