12/16/2023

Airline Management System

DAMG6210 Data Mgt and Database Design
Vasudha Ambre(002676424)

INSTRUCTOR:
PROF. YUSUF OZBEK

Introduction

Background:

The Airline Management System (AMS) project aims to optimize airport operations, enhance passenger experience, and facilitate data-driven decision-making for airport authorities. This report provides a comprehensive overview of the system's design, implementation, and key functionalities.

Objectives of the AMS:

1) Efficient Flight Management:
   Enable the tracking of flights, including scheduling, departure, and arrival details.

2) Streamlined Passenger Experience:
   Streamline passenger interactions with features like online ticket booking and comprehensive booking management.

3) Employee Assignments:
   Manage crew schedules and assignments for optimal workforce utilization.

4) Data-Driven Decision Making:
   Provide airport authorities with comprehensive data for informed decision-making.

System Overview

Entities and Relationships:

The AMS encompasses key entities such as Airlines, Flights, Airports, Routes, Employees, Passengers, Bookings, and more. Relationships between these entities are established to maintain data integrity and facilitate seamless information flow.

1. **Airlines:**

   Attributes: airline_id (Primary Key), name.

   Relationships:

   - Connected to Flights via airline_id.
   - Linked to Routes through airline_id.

2. **Flights:**

   Attributes: flight_id (Primary Key), airline_id (Foreign Key), departure_time, arrival_time, status, price, class.

   Relationships:

   - Associated with Airlines using airline_id.
   - Connected to BookingDetails through flight_id.
   - Links to Flight_Scheduling using flight_id.

3. **Airports:**

   Attributes: airport_code (Primary Key), name, city.

   Relationships:

   - Connected to Routes via departure_airport and arrival_airport.
   - Used in TicketPrices through source and destination

4. **Routes:**

   Attributes: airline_id (Foreign Key), departure_airport (Foreign Key), arrival_airport (Foreign Key).

   Relationships:

   - Associated with Airlines using airline_id.
   - Linked to Airports through departure_airport and arrival_airport.

5. **Cities:**

Attributes: city_id (Primary Key), name, country.

Relationships:

- Used in Airports through city.

6. **Aircraft:**

Attributes: aircraft_code (Primary Key), model, manufacturer**.**

Relationships:

- Connected to Flight_Scheduling via aircraft_code.

7. **Flight_Scheduling:**

Attributes: flight_id (Foreign Key), aircraft_code (Foreign Key), departure_time, arrival_time.

Relationships:

- Linked to Flights through flight_id.
- Connected to Aircraft via aircraft_code.

8. **Passengers:**

Attributes: passenger_id (Primary Key), full_name, email, phone, gender, age, passport_number, country_of_issue, issue_date, expiration_date.

Relationships:

- Used in BookingDetails through passenger_id.

9. **BookingDetails:**

Attributes: booking_id (Primary Key), passenger_id (Foreign Key), flight_id (Foreign Key), booked_at, payment_status, payment_method, booking_status.

Relationships:

- Connected to Passengers using passenger_id.
- Associated with Flights through flight_id.

10. **Employees:**

Attributes: emp_id (Primary Key), user_id (Foreign Key), first_name, last_name, job_title, phone, hire_date, airline_id (Foreign Key), manager_id (Foreign Key).

Relationships:

- Linked to Users using user_id.
- Connected to Airlines via airline_id.
- Associated with Employees through manager_id.

11. **CrewMembers:**

Attributes: crew_id (Primary Key), emp_id (Foreign Key), flight_id (Foreign Key), role.

Relationships:

- Connected to Employees via emp_id.
- Associated with Flights using flight_id.

12. **Users:**

Attributes: user_id (Primary Key), username, password, role.

Relationships:

- Used in Employees through user_id.

13. **TicketPrices:**

Attributes: date_of_booking, source (Foreign Key), destination (Foreign Key), source_city, destination_city, class, price.

Relationships:

- Linked to Airports through source and destination.

2.Database Schema:

The database schema includes tables for Airlines, Flights, Airports, Routes, Employees, Passengers, Bookings, and others. Foreign key constraints ensure data consistency, and indexes enhance query performance.

1. Airlines Table:

i. airline_id VARCHAR(3) (Primary Key)
ii. name VARCHAR(50) NOT NULL

2. Flights Table:

i. flight_id INT (Primary Key)
ii. airline_id VARCHAR(3) NOT NULL (Foreign Key)

iii. departure_time DATETIME NOT NULL
iv. arrival_time DATETIME NOT NULL
v. status VARCHAR(20)
vi. price DECIMAL(10, 2)
vii. class VARCHAR(20) NOT NULL

3. Airports Table:

i. airport_code VARCHAR(3) (Primary Key)
ii. name VARCHAR(50) NOT NULL
iii. city VARCHAR(50) NOT NULL

4. Routes Table:

i. airline_id INT VARCHAR(3) NOT NULL (Foreign Key)
ii. departure_airport VARCHAR(3) NOT NULL (Foreign Key)
iii. arrival_airport VARCHAR(3) NOT NULL (Foreign Key)

5. Cities Table:

i. city_id INT (Primary Key)
ii. name VARCHAR(50) NOT NULL
iii. country VARCHAR(50) NOT NULL

6. Aircraft Table:

i. aircraft_code VARCHAR(10) (Primary Key)
ii. model VARCHAR(50) NOT NULL
iii. manufacturer VARCHAR(50) NOT NULL

7. Flight_Scheduling Table:

i. flight_id INT NOT NULL (Foreign Key)
ii. aircraft_code VARCHAR(10) NOT NULL (Foreign Key)
iii. departure_time DATETIME NOT NULL
iv. arrival_time DATETIME NOT NULL

8. Passengers Table:

i. passenger_id INT (Primary Key)
ii. full_name VARCHAR(50) NOT NULL
iii. email VARCHAR(50) NOT NULL

iv. phone VARCHAR(15) UNIQUE CHECK
v. gender VARCHAR(1)
vi. age INT
vii. passport_number VARCHAR(10) UNIQUE
viii. country_of_issue VARCHAR(50)
ix. issue_date DATE
x. expiration_date DATE

9. BookingDetails Table:

i. booking_id INT NOT NULL (Primary Key)
ii. passenger_id INT NOT NULL (Foreign Key)
iii. flight_id INT NOT NULL (Foreign Key)
iv. booked_at DATETIME NOT NULL
v. payment_status VARCHAR(20)
vi. payment_method VARCHAR(50)
vii. booking_status VARCHAR(20)

10. Employees Table:

i. emp_id INT (Primary Key)
ii. user_id INT (Foreign Key)
iii. first_name VARCHAR(50) NOT NULL
iv. last_name VARCHAR(50) NOT NULL
v. job_title VARCHAR(50) NOT NULL
vi. phone VARCHAR(15) UNIQUE CHECK
vii. hire_date DATE NOT NULL
viii. airline_id VARCHAR(3) (Foreign Key)
ix. manager_id INT (Foreign Key)

11. CrewMembers Table:

i. crew_id INT (Primary Key)
ii. emp_id INT NOT NULL (Foreign Key)
iii. flight_id INT NOT NULL (Foreign Key)
iv. role VARCHAR(50) NOT NULL

12. Users Table:

i. user_id INT (Primary Key)
ii. username VARCHAR(50)

iii. password VARCHAR(255)
iv. role VARCHAR(20) NOT NULL

13. TicketPrices Table:

i. date_of_booking DATE NOT NULL
ii. source VARCHAR(3) NOT NULL (Foreign Key)
iii. destination VARCHAR(3) NOT NULL (Foreign Key)
iv. source_city VARCHAR(255) NOT NULL
v. destination_city VARCHAR(255) NOT NULL
vi. class VARCHAR(20) NOT NULL
vii. price INT

3. Key Functionalities:

1. Booking Management:

The system allows users to book flights through the BookingDetails table.

Information such as passenger details, flight details, booking time, payment status, and method are recorded.

2.Employee Management:

The Employees table manages information about airline staff, including pilots, flight attendants, engineers, crew managers, and ground staff.

3. Flight Scheduling:

The Flight_Scheduling table maintains the schedule of flights, associating each flight with a specific aircraft and departure/arrival times.

4. User Authentication:

The Users table handles user authentication and authorization, controlling access to the system based on roles (e.g., admin, staff, customer).

5. Ticket Pricing:

The TicketPrices table stores information on ticket prices for different routes and classes, allowing the airline to manage pricing strategies.

6. Relationship Management:

The relationships between entities, such as the association between flights and airlines, airports and routes, and employees and roles, ensure data consistency and accuracy.

7. Data Integrity:

Foreign key constraints are in place to maintain data integrity, preventing inconsistencies in the relationships between tables.

8. Reporting and Analysis:

The database structure supports reporting and analysis, enabling the airline to gather insights into booking patterns, revenue, and employee performance.

9 .Dynamic Updates:

The system allows for dynamic updates to flight schedules, ticket prices, and employee information, ensuring adaptability to changing operational requirements.

Normalization

Normalization is a process in database design that organizes tables and columns of a relational database to reduce data redundancy and improve data integrity. The most common normalization forms are First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF). I will evaluate the provided database tables against these normalization forms.

1. First Normal Form (1NF):

A table is in 1NF if it contains only atomic values, and there are no repeating groups or arrays.

All tables are in 1NF. They have atomic values in each column, and there are no repeating groups.

2. Second Normal Form (2NF):

A table is in 2NF if it is in 1NF and all non-prime attributes are fully functionally dependent on the primary key.

Flights: The status column is dependent on the flight_id, which is part of the primary key. No partial dependencies.

Routes: No partial dependencies.

3. Third Normal Form (3NF):

A table is in 3NF if it is in 2NF, and no transitive dependencies exist.

Flights: The airline_id determines the status, and price is dependent on class. Move status to an Airlines table, and class to a separate table.

Airports: The city column is dependent on the airport_code. Move city to a Cities table.

Changes Overview Using Normalization

1. **Creation of New Tables:**

   AirlinesInfo:

   - Introduces a new table to store additional information about airlines.
   - Maintains a one-to-one relationship with the existing Airlines table.
   - Contains details such as the founding year, headquarters, and contact information.

   FlightDetails:

- Introduces a new table to store detailed flight information.
- Maintains a one-to-one relationship with the existing Flights table.
- Includes information about aircraft type, departure gate, and additional flight-specific details.

CityInfo:

- Introduces a new table to store comprehensive city information.
- Maintains a one-to-one relationship with the existing Cities table.
- Encompasses details like population, timezone, and local attractions.

2. **Routes Table Enhancement:**

   Routes Table Modification:

   - The Routes table has been updated to utilize surrogate keys for referencing airports and airlines.
   - Introduces new columns for departure and arrival airport codes, eliminating redundancy and improving data consistency.

3. Data Redundancy Reduction:

   Removal of Redundant Columns:

   - Columns such as status in Airlines, price and class in Flights, and city in Airports have been identified as redundant and can be safely removed.

Implementation Steps

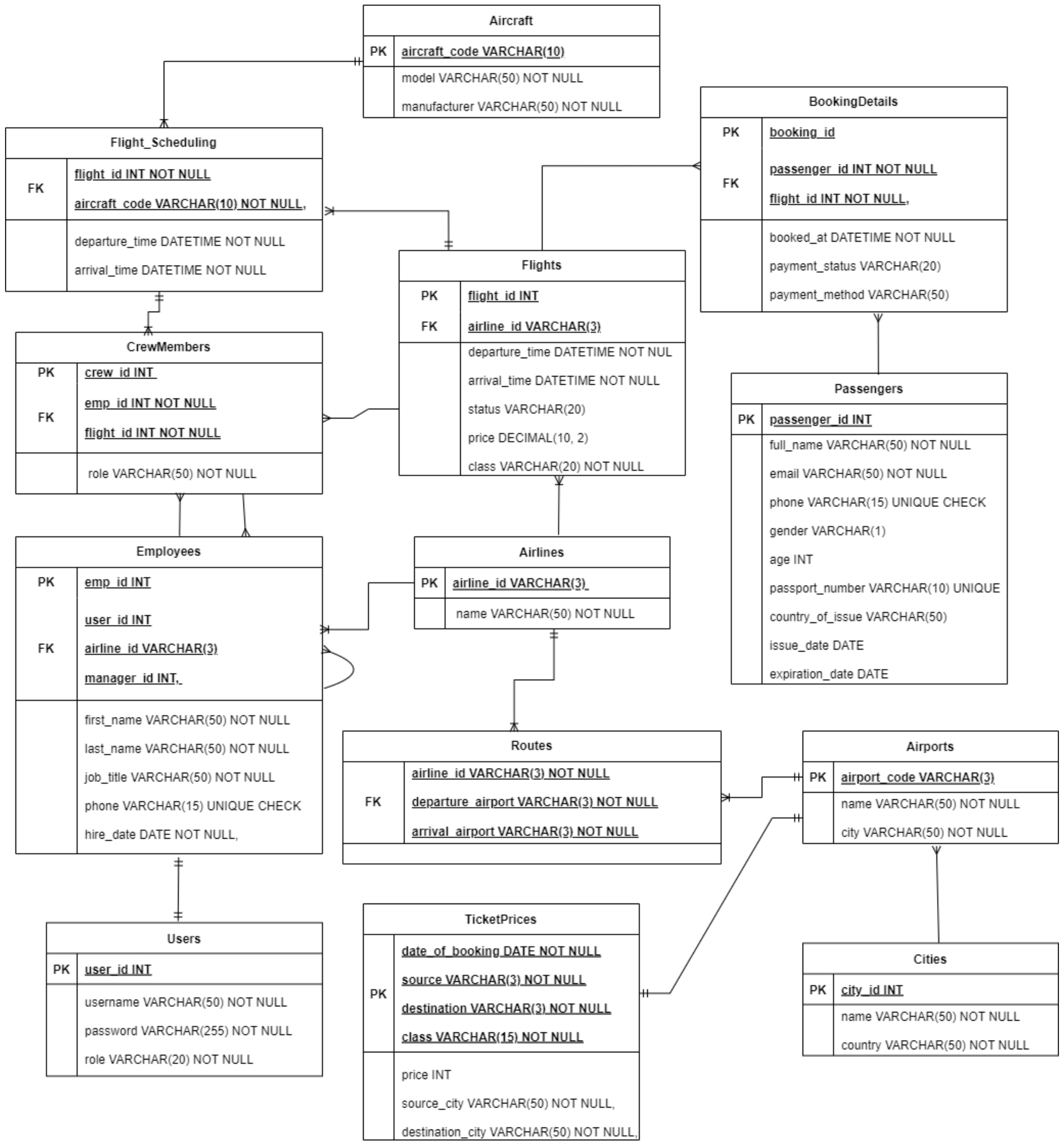
1. Create New Tables:
   - Execute SQL statements to create new tables (AirlinesInfo, FlightDetails, and CityInfo).
   -
2. Populate New Tables:
   - Copy data from existing tables (Airlines, Flights, Airports) to the newly created tables.

3. Routes Table Modification:
   - Update the Routes table by replacing existing foreign key references with surrogate keys.

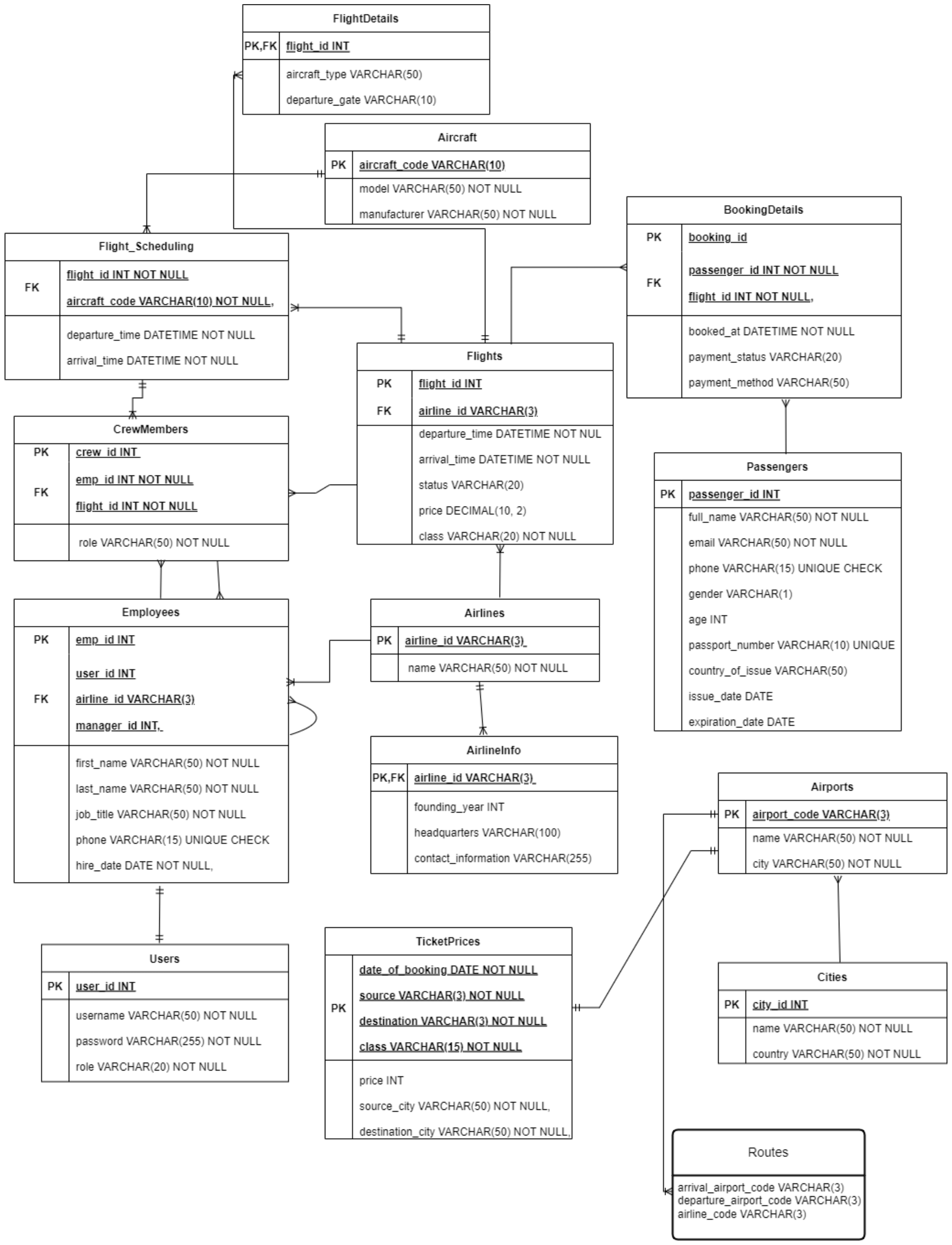
4. Remove Redundant Columns:
   - Safely remove identified redundant columns from relevant tables.

ER Diagram

Before Normalization : ER Diagram


Aircraft
PK aircraft_code VARCHAR(10)
model VARCHAR(50) NOT NULL
manufacturer VARCHAR(50) NOT NULL
BookingDetails
PK booking_id
FK passenger_id INT NOT NULL
flight_id INT NOT NULL,
booked_at DATETIME NOT NULL
payment_status VARCHAR(20)
payment_method VARCHAR(50)
Flight_Scheduling
FK flight_id INT NOT NULL
aircraft_code VARCHAR(10) NOT NULL,
departure_time DATETIME NOT NULL
arrival_time DATETIME NOT NULL
Flights
PK flight_id INT
FK airline_id VARCHAR(3)
departure_time DATETIME NOT NUL
arrival_time DATETIME NOT NULL
status VARCHAR(20)
price DECIMAL(10, 2)
class VARCHAR(20) NOT NULL
CrewMembers
PK crew_id INT
FK emp_id INT NOT NULL
flight_id INT NOT NULL
role VARCHAR(50) NOT NULL
Passengers
PK passenger_id INT
full_name VARCHAR(50) NOT NULL
email VARCHAR(50) NOT NULL
phone VARCHAR(15) UNIQUE CHECK
gender VARCHAR(1)
age INT
passport_number VARCHAR(10) UNIQUE
country_of_issue VARCHAR(50)
issue_date DATE
expiration_date DATE
Employees
PK emp_id INT
FK user_id INT
airline_id VARCHAR(3)
manager_id INT,
first_name VARCHAR(50) NOT NULL
last_name VARCHAR(50) NOT NULL
job_title VARCHAR(50) NOT NULL
phone VARCHAR(15) UNIQUE CHECK
hire_date DATE NOT NULL,
Airlines
PK airline_id VARCHAR(3)
name VARCHAR(50) NOT NULL
Routes
FK airline_id VARCHAR(3) NOT NULL
departure_airport VARCHAR(3) NOT NULL
arrival_airport VARCHAR(3) NOT NULL
Airports
PK airport_code VARCHAR(3)
name VARCHAR(50) NOT NULL
city VARCHAR(50) NOT NULL
Users
PK user_id INT
username VARCHAR(50) NOT NULL
password VARCHAR(255) NOT NULL
role VARCHAR(20) NOT NULL
TicketPrices
PK date_of_booking DATE NOT NULL
source VARCHAR(3) NOT NULL
destination VARCHAR(3) NOT NULL
class VARCHAR(15) NOT NULL
price INT
source_city VARCHAR(50) NOT NULL,
destination_city VARCHAR(50) NOT NULL,
Cities
PK city_id INT
name VARCHAR(50) NOT NULL
country VARCHAR(50) NOT NULL

After Normalization : ER Diagram


FlightDetails
PK,FK flight_id INT
aircraft_type VARCHAR(50)
departure_gate VARCHAR(10)
Aircraft
PK aircraft_code VARCHAR(10)
model VARCHAR(50) NOT NULL
manufacturer VARCHAR(50) NOT NULL
BookingDetails
PK booking_id
FK passenger_id INT NOT NULL
flight_id INT NOT NULL,
booked_at DATETIME NOT NULL
payment_status VARCHAR(20)
payment_method VARCHAR(50)
Flight_Scheduling
FK flight_id INT NOT NULL
aircraft_code VARCHAR(10) NOT NULL,
departure_time DATETIME NOT NULL
arrival_time DATETIME NOT NULL
Flights
PK flight_id INT
FK airline_id VARCHAR(3)
departure_time DATETIME NOT NUL
arrival_time DATETIME NOT NULL
status VARCHAR(20)
price DECIMAL(10, 2)
class VARCHAR(20) NOT NULL
CrewMembers
PK crew_id INT
FK emp_id INT NOT NULL
flight_id INT NOT NULL
role VARCHAR(50) NOT NULL
Passengers
PK passenger_id INT
full_name VARCHAR(50) NOT NULL
email VARCHAR(50) NOT NULL
phone VARCHAR(15) UNIQUE CHECK
gender VARCHAR(1)
age INT
passport_number VARCHAR(10) UNIQUE
country_of_issue VARCHAR(50)
issue_date DATE
expiration_date DATE
Employees
PK emp_id INT
FK user_id INT
airline_id VARCHAR(3)
manager_id INT,
first_name VARCHAR(50) NOT NULL
last_name VARCHAR(50) NOT NULL
job_title VARCHAR(50) NOT NULL
phone VARCHAR(15) UNIQUE CHECK
hire_date DATE NOT NULL,
Airlines
PK airline_id VARCHAR(3)
name VARCHAR(50) NOT NULL
AirlineInfo
PK,FK airline_id VARCHAR(3)
founding_year INT
headquarters VARCHAR(100)
contact_information VARCHAR(255)
Airports
PK airport_code VARCHAR(3)
name VARCHAR(50) NOT NULL
city VARCHAR(50) NOT NULL
Users
PK user_id INT
username VARCHAR(50) NOT NULL
password VARCHAR(255) NOT NULL
role VARCHAR(20) NOT NULL
TicketPrices
PK date_of_booking DATE NOT NULL
source VARCHAR(3) NOT NULL
destination VARCHAR(3) NOT NULL
class VARCHAR(15) NOT NULL
price INT
source_city VARCHAR(50) NOT NULL,
destination_city VARCHAR(50) NOT NULL,
Cities
PK city_id INT
name VARCHAR(50) NOT NULL
country VARCHAR(50) NOT NULL
Routes
arrival_airport_code VARCHAR(3)
departure_airport_code VARCHAR(3)
airline_code VARCHAR(3)

ER Diagram Description

1. **Airlines Entity:**

   Attributes: airline_id (Primary Key), name

   Relationship: Connected to Flights, Routes, Employees, and AirlinesInfo through foreign key relationships.

2. **Flights Entity:**

   Attributes: flight_id (Primary Key), airline_id (Foreign Key), departure_time, arrival_time, status, price, class

   Relationship: Connected to Airlines, Flight_Scheduling, Routes, and BookingDetails through foreign key relationships.

3. **Airports Entity:**

   Attributes: airport_code (Primary Key), name, city

   Relationship: Connected to Routes, TicketPrices, and AirlinesInfo through foreign key relationships.

4. **Routes Entity:**

   Attributes: airline_code (Foreign Key), departure_airport_code (Foreign Key), arrival_airport_code (Foreign Key)

   Relationship: Connected to Airlines, Airports, Flights, and TicketPrices through foreign key relationships.

5. **Cities Entity:**

   Attributes: city_id (Primary Key), name, country

   Relationship: Connected to Airports.

6. **Aircraft Entity:**

   Attributes: aircraft_code (Primary Key), model, manufacturer

   Relationship: Connected to Flight_Scheduling.

7. **Flight_Scheduling Entity:**

   Attributes: flight_id (Foreign Key), aircraft_code (Foreign Key), departure_time, arrival_time

   Relationship: Connected to Flights and Aircraft.

8. **Passengers Entity:**

   Attributes: passenger_id (Primary Key), full_name, email, phone, gender, age, passport_number, country_of_issue, issue_date, expiration_date

   Relationship: Connected to BookingDetails.

9. **BookingDetails Entity:**

   Attributes: booking_id (Primary Key), passenger_id (Foreign Key), flight_id (Foreign Key), booked_at, payment_status, payment_method, booking_status

   Relationship: Connected to Flights and Passengers.

10. **Employees Entity:**

    Attributes: emp_id (Primary Key), user_id (Foreign Key), first_name, last_name, job_title, phone, hire_date, airline_id (Foreign Key), manager_id (Foreign Key)

    Relationship: Connected to Airlines, Users, and CrewMembers.

11. **CrewMembers Entity:**

    Attributes: crew_id (Primary Key), emp_id (Foreign Key), flight_id (Foreign Key), role

    Relationship: Connected to Employees and Flights.

12. **Users Entity:**

    Attributes: user_id (Primary Key), username, password, role

    Relationship: Connected to Employees.

13. **TicketPrices Entity:**

    Attributes: date_of_booking, source, destination, source_city, destination_city, class, price

    Relationship: Connected to Airports and Routes.

14. **AirlinesInfo Entity:**

    Attributes: airline_id (Primary Key), founding_year, headquarters, contact_information

    Relationship: Connected to Airlines.

15. **FlightDetails Entity:**

    Attributes: flight_id (Primary Key), aircraft_type, departure_gate

    Relationship: Connected to Flights.

Database Constraints and Special Features

Database Constraints

1. Primary Key Constraint:
   - Examples: Airlines.airline_id, Flights.flight_id, Airports.airport_code, Routes.airline_code, etc.
   - Ensures the uniqueness and non-null values of the specified columns.

2. Foreign Key Constraint:
   - Examples: Flights.airline_id, Routes.airline_code, Flight_Scheduling.flight_id, CrewMembers.emp_id, etc.
   - Establishes a link between two tables, enforcing referential integrity.

3. Unique Constraint:
   - Examples: Airlines.airline_id, Airports.airport_code, Passengers.phone, Users.username, etc.
   - Ensures that values in specified columns are unique.

4. Check Constraint:
   - Example: Passengers.phone CHECK (phone REGEXP '^\\+\\d{2}-\\d{10}$')
   - Enforces a condition on the values that can be inserted into a column.

5. Index:
   - Examples: idx_city_name on Cities(name), idx_passenger_phone on Passengers(phone), etc.
   - Improves the speed of data retrieval operations by creating an index on specified columns.

6. Unique Index:
   - Examples: idx_passenger_phone on Passengers(phone), idx_airline_id_flights on Flights(airline_id), etc.
   - Similar to a regular index but enforces uniqueness on indexed columns.

Special Features:

1. CASCADE and SET NULL Actions:
    - Example: ON DELETE CASCADE ON UPDATE CASCADE in Flights and Flight_Scheduling foreign key constraints.
    - Specifies that when a referenced row is deleted or updated, the action should be cascaded to the referencing table or set to NULL.

2. Datetime Data Types:
    - Examples: departure_time and arrival_time columns in Flights and Flight_Scheduling tables.
    - Stores date and time information.

3. Regular Expression Check Constraint:
    - Example: Passengers.phone CHECK (phone REGEXP '^\\+\\d{2}-\\d{10}$')
    - Validates that the phone column in Passengers table follows a specific regular expression pattern.

4. Indexing for Better Performance:
    - Examples: idx_airline_id_flights, idx_passenger_phone, idx_flight_id_fs, etc.
    - Improves query performance by creating indexes on columns used frequently in search conditions.

5. Sample Data Insertion:
    - Example: Insertion of sample data into tables like Airlines, Flights, Airports, etc.
    - Provides a foundation for testing and development.

6. Normalization:
    - Example: Splitting Airlines into Airlines and AirlinesInfo tables.
    - A design technique to organize data and reduce data redundancy.

7. Dynamic Data:
    - Example: TicketPrices table with dynamic pricing based on the date of booking, source, destination, and class.
    - Allows for variable pricing based on different factors.

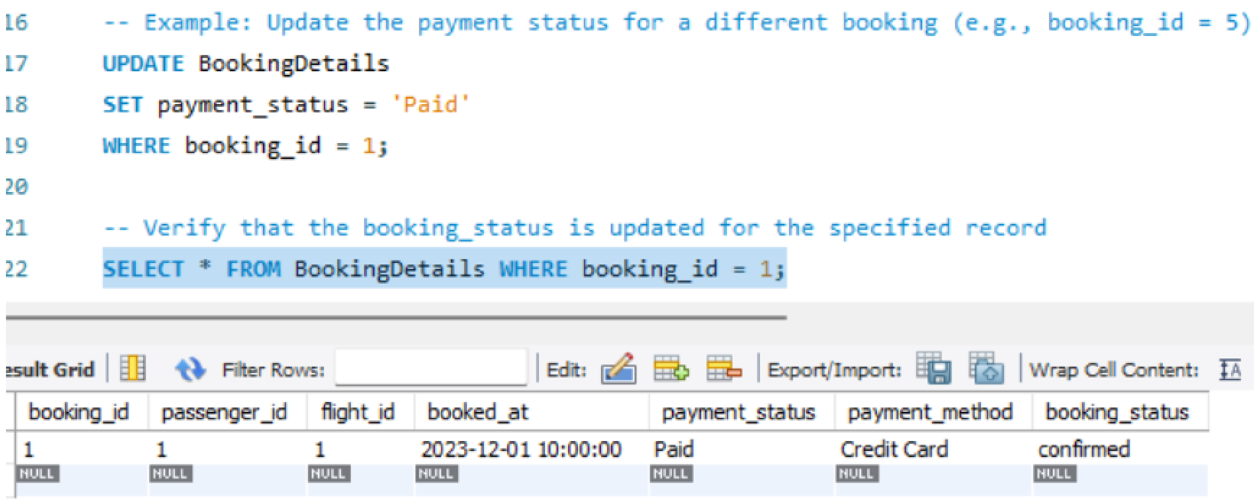
Trigger

Update Booking Status based on Payment Status:

```
DELIMITER //
CREATE TRIGGER update_booking_status_update
BEFORE INSERT ON BookingDetails
FOR EACH ROW
BEGIN

   IF NEW.payment_status = 'Paid' THEN

      SET NEW.booking_status = 'confirmed';

   ELSE

      SET NEW.booking_status = 'cancelled';

   END IF;

END;
//
DELIMITER
```

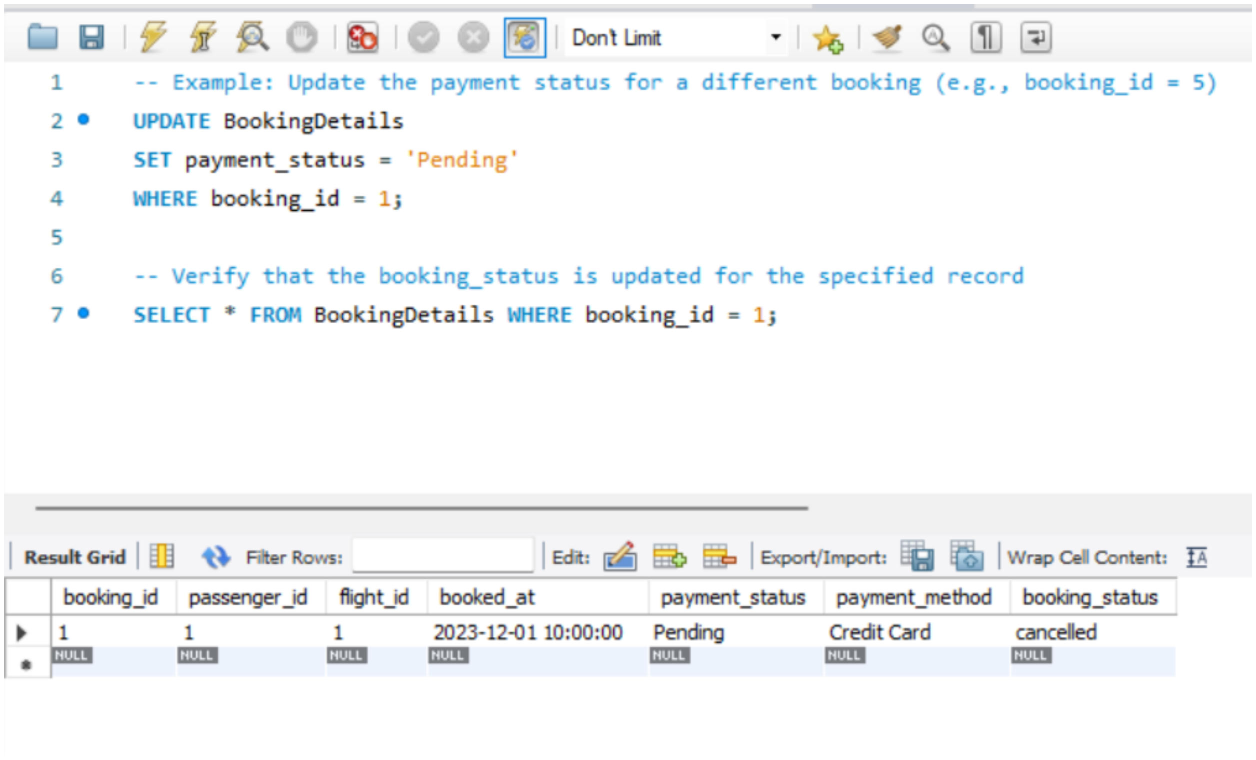
Explanation:

- This trigger is designed to automatically update the booking_status in the BookingDetails table based on the payment_status when a new record is inserted.
- If the payment_status is 'Paid,' the booking_status will be set to 'confirmed.' Otherwise, it will be set to 'cancelled.'



```
-- Example: Update the payment status for a different booking (e.g., booking_id = 5)
UPDATE BookingDetails
SET payment_status = 'Paid'
WHERE booking_id = 1;

-- Verify that the booking_status is updated for the specified record
SELECT * FROM BookingDetails WHERE booking_id = 1;
```

booking_id	passenger_id	flight_id	booked_at	payment_status	payment_method	booking_status
1	1	1	2023-12-01 10:00:00	Paid	Credit Card	confirmed
NULL	NULL	NULL	NULL	NULL	NULL	NULL



```
-- Example: Update the payment status for a different booking (e.g., booking_id = 5)
UPDATE BookingDetails
SET payment_status = 'Pending'
WHERE booking_id = 1;

-- Verify that the booking_status is updated for the specified record
SELECT * FROM BookingDetails WHERE booking_id = 1;
```

booking_id	passenger_id	flight_id	booked_at	payment_status	payment_method	booking_status
1	1	1	2023-12-01 10:00:00	Pending	Credit Card	cancelled
NULL	NULL	NULL	NULL	NULL	NULL	NULL

<u>Stored Procedure</u>

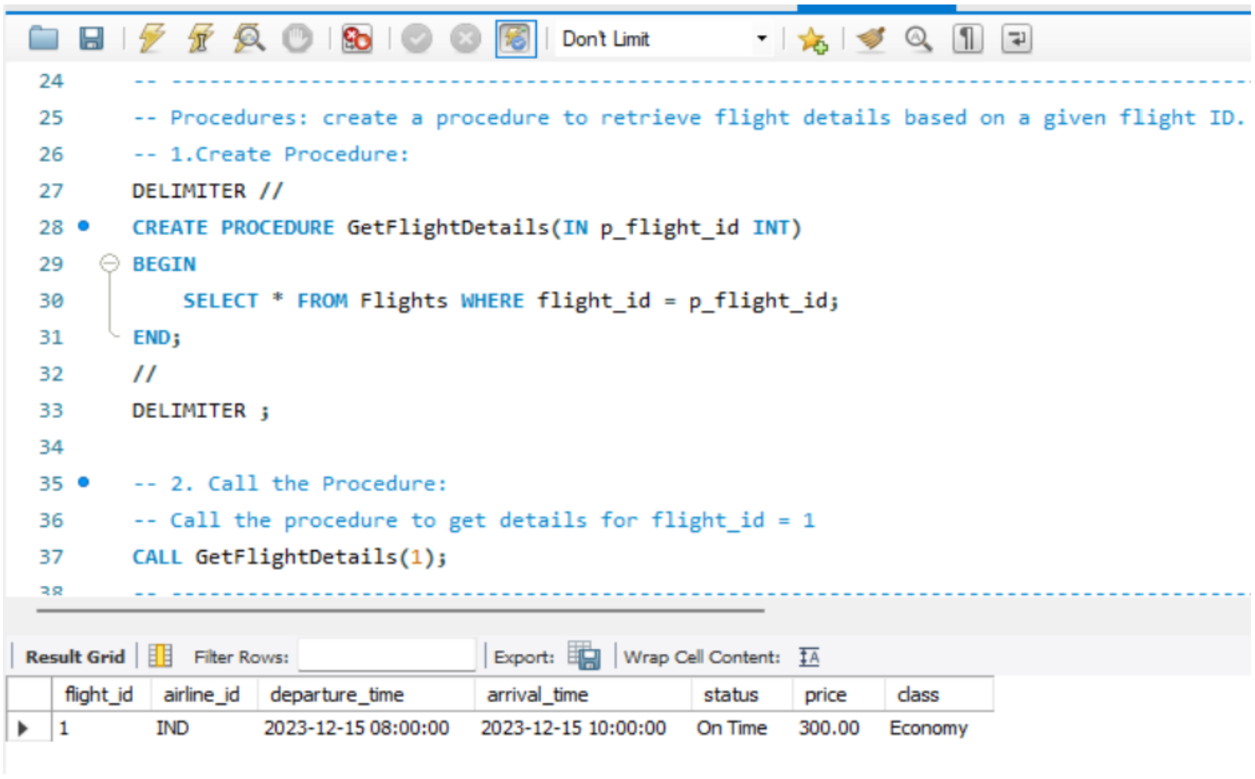
To Retrieve Flight Details:

```
DELIMITER //
CREATE PROCEDURE GetFlightDetails(IN p_flight_id INT)
BEGIN
   SELECT * FROM Flights WHERE flight_id = p_flight_id;
END;
//
DELIMITER ;
```

Explanation:

- This is a stored procedure named GetFlightDetails that takes a parameter p_flight_id and retrieves details for the specified flight ID from the Flights table.

Call the Procedure:



Don't Limit

```
-- ----------------------------------------------------------------------------
-- Procedures: create a procedure to retrieve flight details based on a given flight ID.
-- 1.Create Procedure:
DELIMITER //
CREATE PROCEDURE GetFlightDetails(IN p_flight_id INT)
BEGIN
    SELECT * FROM Flights WHERE flight_id = p_flight_id;
END;
//
DELIMITER ;

-- 2. Call the Procedure:
-- Call the procedure to get details for flight_id = 1
CALL GetFlightDetails(1);
-- ----------------------------------------------------------------------------
```

flight_id	airline_id	departure_time	arrival_time	status	price	class
1	IND	2023-12-15 08:00:00	2023-12-15 10:00:00	On Time	300.00	Economy

Explanation:

- This is an example of how to call the stored procedure GetFlightDetails to retrieve details for a specific flight ID (in this case, flight_id = 3).

<u>View</u>

Create View for Confirmed Bookings:

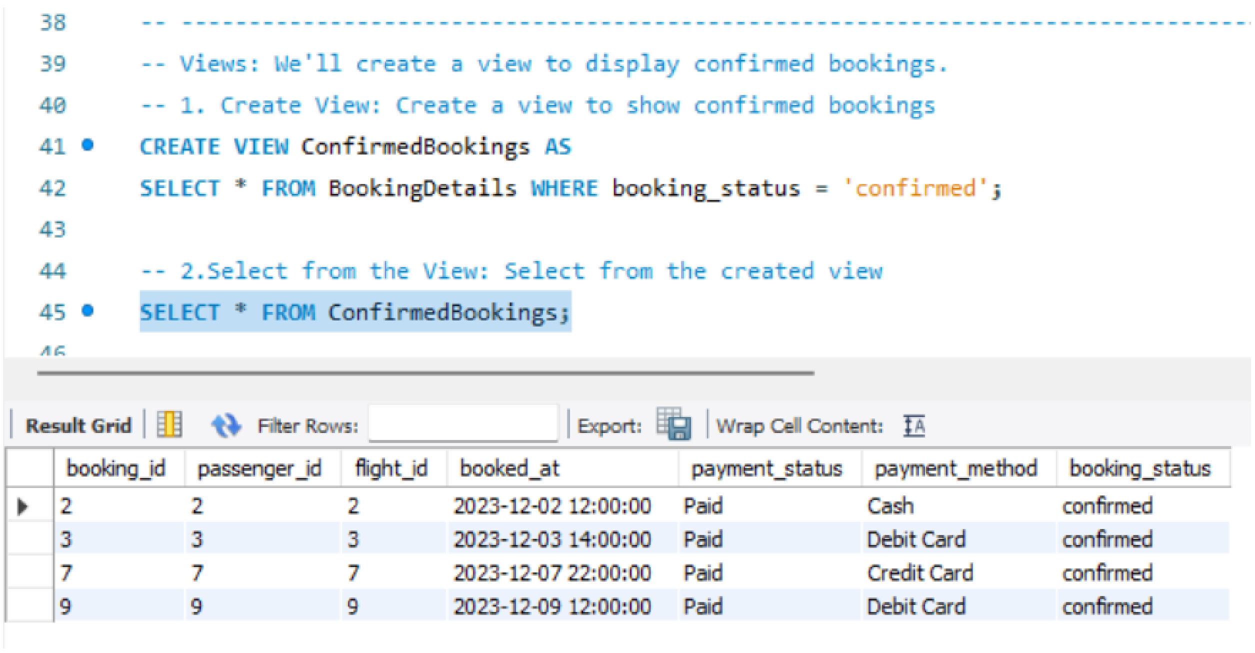
CREATE VIEW ConfirmedBookings AS

SELECT * FROM BookingDetails WHERE booking_status = 'confirmed';

Explanation:

- This creates a view named ConfirmedBookings that displays all records from the BookingDetails table where the booking_status is 'confirmed.'

Select from the View:

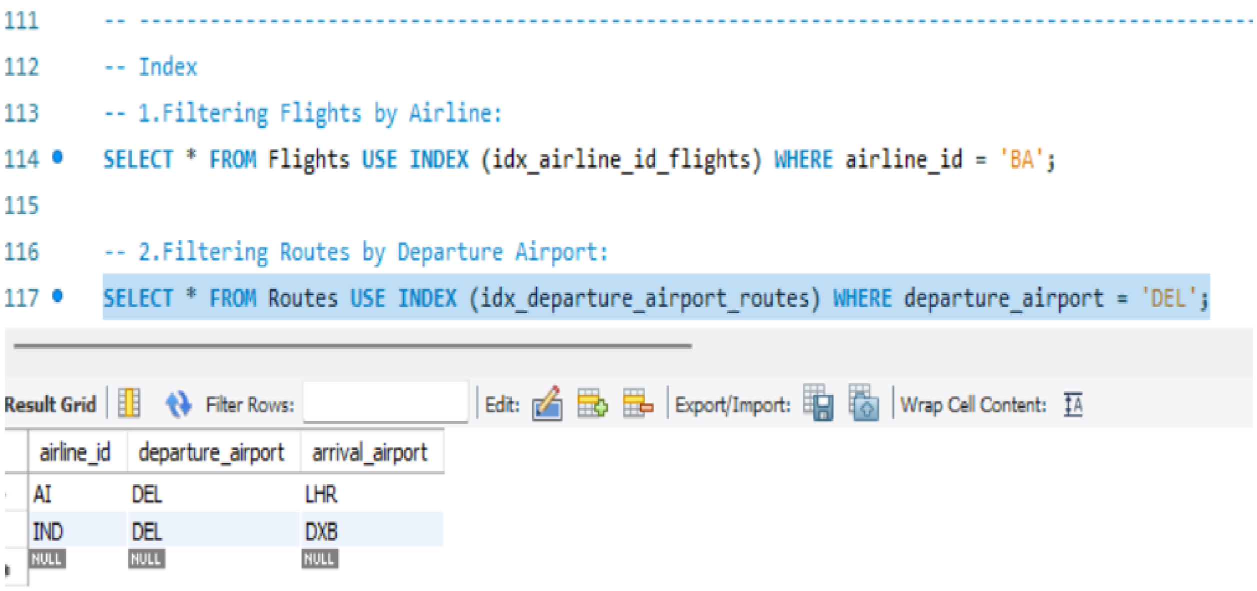


```
-- ---------------------------------------------------------------------
-- Views: We'll create a view to display confirmed bookings.
-- 1. Create View: Create a view to show confirmed bookings
CREATE VIEW ConfirmedBookings AS
SELECT * FROM BookingDetails WHERE booking_status = 'confirmed';

-- 2.Select from the View: Select from the created view
SELECT * FROM ConfirmedBookings;
```

booking_id	passenger_id	flight_id	booked_at	payment_status	payment_method	booking_status
2	2	2	2023-12-02 12:00:00	Paid	Cash	confirmed
3	3	3	2023-12-03 14:00:00	Paid	Debit Card	confirmed
7	7	7	2023-12-07 22:00:00	Paid	Credit Card	confirmed
9	9	9	2023-12-09 12:00:00	Paid	Debit Card	confirmed

Index:

- Enhance query performance by providing quick access to rows in a table.
- Example: Indexes on frequently queried columns like airline_id or flight_id.
- These constraints and features collectively contribute to the integrity, security, and efficiency of our Airline Management System Database.



```
-- ---------------------------------------------------------------------
-- Index
-- 1.Filtering Flights by Airline:
SELECT * FROM Flights USE INDEX (idx_airline_id_flights) WHERE airline_id = 'BA';

-- 2.Filtering Routes by Departure Airport:
SELECT * FROM Routes USE INDEX (idx_departure_airport_routes) WHERE departure_airport = 'DEL';
```

airline_id	departure_airport	arrival_airport
AI	DEL	LHR
IND	DEL	DXB
NULL	NULL	NULL

Analytical Queries

1. What is the total revenue generated by each travel class?

```
SELECT class, SUM(price) AS total_revenue
FROM Flights
WHERE status = 'On Time'
GROUP BY class;
```

2. Which are the top 5 routes with the highest total revenue?

```
SELECT r.departure_airport, r.arrival_airport, SUM(f.price) AS total_revenue
FROM Routes r
JOIN Flights f ON r.airline_id = f.airline_id
WHERE f.status = 'On Time'
GROUP BY r.departure_airport, r.arrival_airport
ORDER BY total_revenue DESC
LIMIT 5;
```

3. What is the average booking amount for each payment method?

```
SELECT payment_method, AVG(price) AS avg_booking_amount
FROM BookingDetails b
JOIN Flights f ON b.flight_id = f.flight_id
WHERE f.status = 'On Time'
GROUP BY payment_method;
```

4. How many passengers fall into each age group?

```
SELECT
  CASE
    WHEN age BETWEEN 1 AND 18 THEN '1-18'
    WHEN age BETWEEN 19 AND 35 THEN '19-35'
    WHEN age BETWEEN 36 AND 50 THEN '36-50'
    WHEN age BETWEEN 51 AND 65 THEN '51-65'
    ELSE '66+'
  END AS age_group,
  COUNT(*) AS total_passengers
FROM Passengers
GROUP BY age_group;
```

CONCLUSION

In the course of this database design and management project, we have successfully established a comprehensive relational database for an airline management system. The system encompasses various entities such as Airlines, Flights, Airports, Routes, Cities, Aircraft, Employees, Crew Members, Users, Ticket Prices, Booking Details, and more. The design adheres to relational database principles and normalization techniques to ensure data integrity, minimize redundancy, and enhance overall system efficiency.

Key Achievements:

Normalization: We applied normalization techniques to organize data into logical structures, minimizing data redundancy and ensuring consistency across the database. This contributes to the overall robustness and maintainability of the system.

Triggers and Procedures: Utilizing triggers, we implemented automatic updates for booking statuses based on payment statuses. Additionally, we created a stored procedure for retrieving flight details, showcasing the versatility and extensibility of the database.

Views and Indexing: We designed views to display confirmed bookings, providing a convenient way to access relevant data. Moreover, index usage examples were incorporated to demonstrate the optimization of query performance, particularly when filtering data.

Foreign Key Constraints: Foreign key constraints were leveraged to establish relationships between tables, ensuring referential integrity and supporting effective data management.

User Roles: The system incorporates user roles to manage access and permissions efficiently. Users, including employees, crew members, and passengers, have distinct roles and corresponding privileges within the system.

Overall, this airline management system database serves as a robust foundation for the effective organization and retrieval of information related to flights, bookings, passengers, and various operational aspects of an airline. The implemented features and design choices contribute to a scalable and maintainable solution that can adapt to the evolving needs of the airline industry.

The success of this project is attributed to the careful consideration of database design principles, thoughtful implementation of database objects, and adherence to best practices in relational database management. Moving forward, the system can be further enhanced with additional features, optimization strategies, and integration with broader airline management systems.