DATABASE SYSTEMS ASSIGNMENT

Airlines Management System

PROJECT BY :

NETHRADEVI BASKAR 2021A7PS0031U

MARYAM SHERRIF 2021A7PS0128U

ANN RAI JACOB 2021A7PS0027U

ASHI KUMBHARE 2021A7PS0075U

DEEPTHI GOUD GOUNDLA 2021A7PS0204U

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Professor Tamizharasan Periyasamy ,Professor Sujala Shetty and Professor Pramod Gaur for giving us the opportunity to learn about the various concepts featured in Database Systems through the use of MySQL.

The extensive use of the various concepts learnt in this assignment has proven to be very efficient and useful in understanding the working of the code and applying concepts to create real-time applications.

This assignment has also proved to be beneficial when it comes to learning about working as a team and incorporating each member's ideas and combining them suitably to obtain the desired outputs.

We would like to thank all the professors of the DBS department for their constant support, patience and guidance.

CONTENTS

INTRODUCTION

An airlines management system is a software application designed to manage and optimize airline operations. It covers a wide range of functions, including flight scheduling, ticketing and reservations, baggage handling, passenger management, and maintenance management. Airlines management systems enable airlines to streamline their operations, improve efficiency, reduce costs, and enhance the passenger experience.

The system usually consists of a central database that stores all the necessary information about flights, passengers, and
aircraft. The system's components work together to manage and automate various airline functions, from ticket booking and check-in to boarding and baggage handling. With the use of advanced analytics and machine learning algorithms, airlines management systems can optimize various airline operations, such as pricing, scheduling, and maintenance, to achieve maximum profitability.

The benefits of airlines management systems are numerous, including increased efficiency, improved customer service, better inventory management, and reduced costs. Additionally, airlines management systems can enhance safety and security by providing real-time information on flight status and aircraft maintenance.

Overall, airlines management systems play a crucial role in modern air travel, helping airlines to manage and optimize their operations effectively, enhance the passenger experience, and ensure safe and efficient air travel.

PROBLEM STATEMENT

The software required to implement our Airlines Management System includes:

1. MySQL Workbench 8.0.24
2. JAVA
3. APACHE NETBEANS 17 VERSION

The hardware requirements for this project can be satisfied by any device and operating system that can support the full working of the above mentioned softwares.

As per assignment requirements, the following must be implemented:

1. Conceptual Design - ER Diagram
2. Conversion to relational model
3. Normalization
4. Creation and population of tables in SQL
5. Queries
6. Functions and procedures
7. Triggers
8. Front end design

The Airline Management System is a comprehensive application designed to help airline companies manage their operations effectively. The system provides an easy-to-use interface. The system has the following capabilities:

1. The Administrator has access to make any kinds of changes to the system.

2. Passengers can make reservations for flights based on their preferred dates, times, and destinations. The system allows customers to select their preferred seats such as business or economy seats.

3. Our system also provides all the details regarding the ticket.We can easily get details on the class of ticket the passenger has booked and the corresponding ticket fare with the booking date and airline name.

4. Airlines also have all the details of the staff members working for it.We just have to search for the particular airline and we can get the exact details of the pilot working for that particular airline easily through the database.

5. Our system has details of all the passengers personal details, flight details, passport details, and contact information,which can be easily accessed.

6. Airline management system handles all the airline details and one airline can have multiple flights, and one passenger can book multiple tickets at different times.

7. This database ensures that every airline or every flight member has a crew member because total participation is required as every airline has all crew members. Any airline must consist of pilot,cabin crew and staff for the ER diagram to be satisfied.

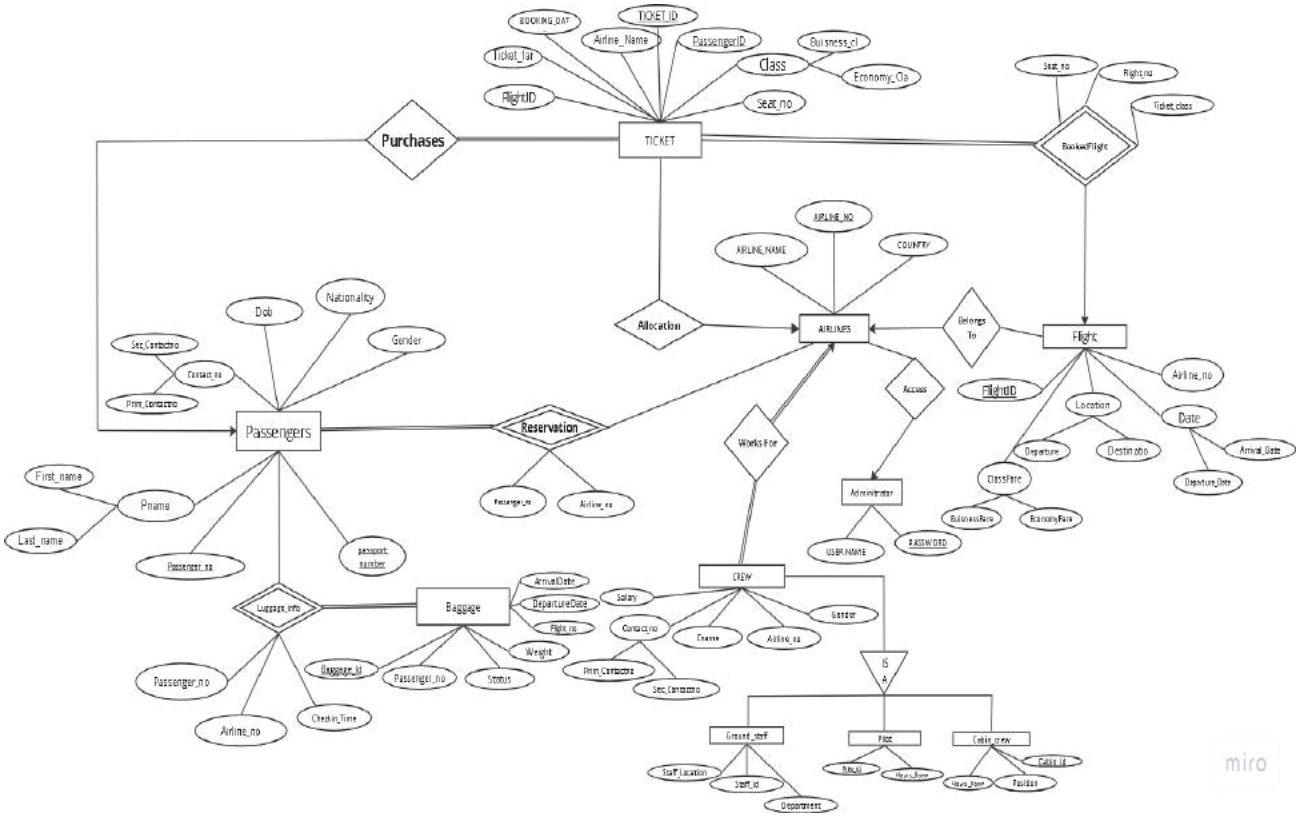
8. Our database can easily provide details about everything ranging to check whether a passenger's luggage has been checked in or not,gives all the baggage details or the details of the particular flight the passenger is boarding.

9. The front-end design of our application has simplified and streamlined the process of editing the database, making it more efficient and user-friendly.

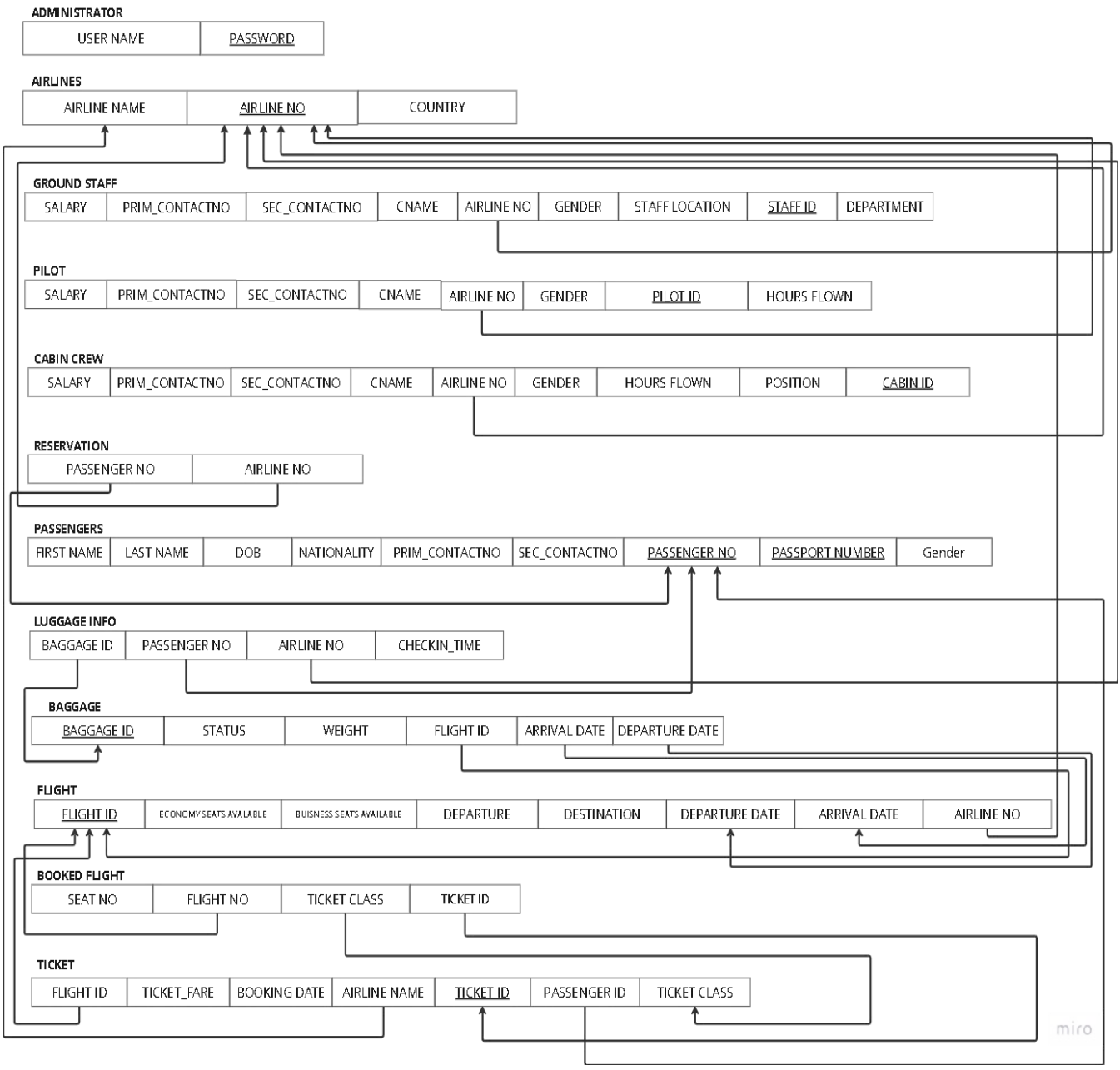
10. We have created functions, procedures and views to know about the revenue generated, most oftenly used routes, number of available seats in economy and business class, salary calculations etc.

In summary, the Airline Management System provides a complete solution for airline companies to manage their operations efficiently, from managing flights and reservations to tracking employee performance and generating reports.

ER DIAGRAM


BOOKING_DAT
TICKET_ID
Airline_Name
PassengerID
Business_cl
Class
Economy_Cla
Ticket_Tar
FlightID
Seat_no
TICKET
Purchases
Seat_no
Flight no
Ticket_class
BookedFlight
AIRLINE_NO
AIRLINE_NAME
COUNTRY
Allocation
AIRLINES
Belongs To
Flight
Nationality
Dob
Gender
Sec_Contactno
Contact_no
Prim_Contactno
Passengers
Reservation
Access
Works For
FlightID
Airline_no
Location
Date
Departure
Destination
Arrival_Date
Departure_Date
ClassFare
BusinessFare
EconomyFare
First_name
Pname
Last_name
Passenger_no
passport number
Administrator
USER_NAME
PASSWORD
Passenger_no
Airline_no
CREW
Luggage_info
Baggage
ArrivalDate
DepartureDate
Flight_no
Weight
Status
Passenger_no
Baggage_id
Passenger_no
Airline_no
Check-in_Time
Salary
Contact_no
Course
Airline_no
Gender
Prim_Contactno
Sec_Contactno
IS A
Ground_staff
Staff_Location
Staff_id
Department
Pilot
Pilot_id
Hours_flown
Cabin_crew
Cabin_id
Hours_flown
Position
miro

RELATIONAL MODEL



NORMALIZATION

Database normalization is the process of restructuring the relational database according to a series of normal forms to reduce data redundancy and improve data integrity. For a table to be in 3NF, it should satisfy two conditions:

- The table should not contain any transitive dependencies and
- The table should not contain any partial dependencies (i.e., it should be in 2NF form) ,

where

- Partial dependency is defined as a functional dependency of the form (proper subset of a candidate key) → (non prime attribute)
- Transitive dependency is defined as a functional dependency of the form (non prime attribute) → (non prime attribute)

And a functional dependency is said to be in BCNF if these properties hold:

- It should already be in 3NF.
- For a functional dependency, say P->Q, P should be a super key.

Let's consider all the tables in our relational model one by one

TABLES

1. **ADMINISTRATOR**

   Attributes:

   1. Username
   2. Password (Primary Key)

   Functional Dependency -

   Password -> Username

   For our Airline Management System, we have decided to have only one administrator with access to the system.

   The Password attribute uniquely determines the Username attribute. Therefore, this table is in 2NF as there are no partial dependencies.

However, since there are only two attributes, there cannot be any transitive dependencies. So, we can say that the given table is in 2NF, 3NF and BCNF.

In summary, the "administrator" table has been designed to be simple and efficient, ensuring that our airline management system is secure and accessible only to authorized personnel.

2. AIRLINES

Attributes:
Let's consider -

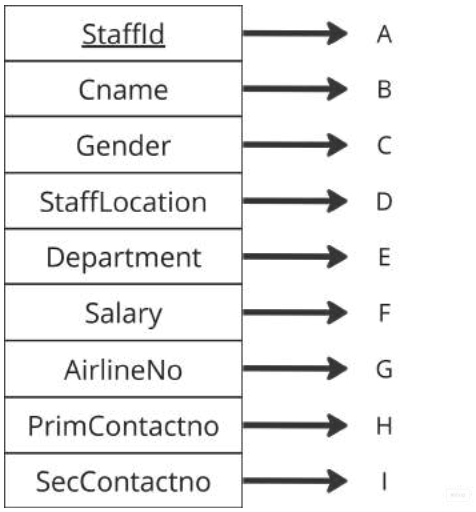
Functional Dependency:
{A → B , C}
This functional dependency satisfies the requirements of 1NF(First Normal Form) because there is no multivalued attribute hence, each row in the table represents a unique airline with a distinct primary key ("airlineNo"). and 2NF (Second Normal Form) because all non-key attributes (airlineName and country) are fully functionally dependent on the primary key (airlineNo).

Since there are no transitive dependencies or partial dependencies in the table, it also satisfies the requirements of 3NF and BCNF.

3. GROUND STAFF

Attributes**:**

Let's consider -



Functional Dependency:

{A → B , C , D , E , F , H , I}

{A → G} (Where G is a foreign key)

G : AirlineNo -> AirlineName, Country (From Airlines table)

All the attributes are atomic and there are no repeating groups .There are no partial dependencies also.

Hence the table is already in INF and 2NF.

This means that "staffId" determines all the other attributes in the table Including AirlineNo.

The "groundStaff" table is in *BCNF* and *3NF* because all **non-trivial** functional dependencies have determinants that are candidate keys or superkeys.

"StaffId" is already a candidate key, "AirlineNo" is not a candidate key because multiple ground staff members can work for the same airline.

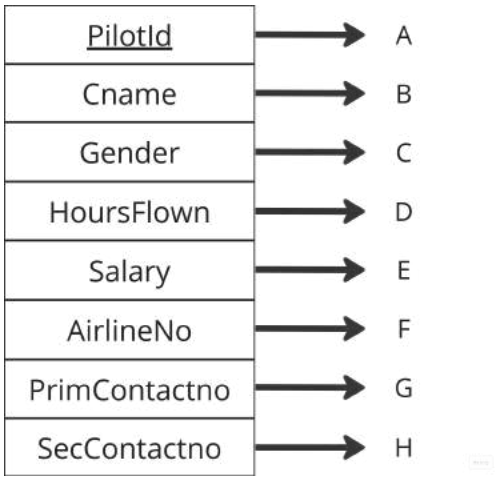
Therefore, we need to check if "airlineNo" is a **superkey or not**. But in this case, the functional dependency of "airlineNo -> airlineName, country" is trivial since the information about airlineName and country is already present in the airlines table.

Similar Explanation goes for Pilot and CabinCrew tables.

4. **PILOT**

Attributes**:**

Let's consider -



Functional Dependency:

{A → B , C , D , E , G , H }

{A → F} (Where F is a foreign key)

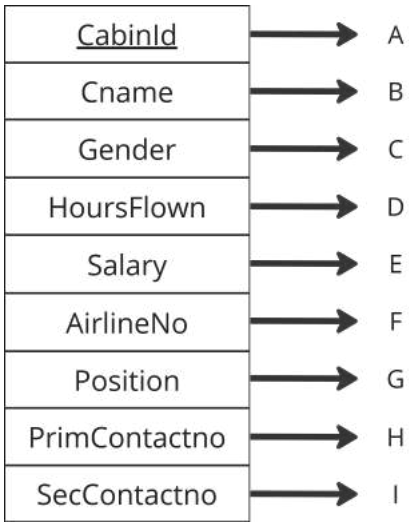
F : AirlineNo -> AirlineName, Country (From Airlines table)

This means that "Pilot Id" determines all the other attributes in the table Including AirlineNo.
The "pilot" table is already in 1NF because all the attributes are atomic and there are no repeating groups.
Hence, "pilotId" is a candidate key for the "pilot" table because it uniquely identifies each row.
There are **no partial dependencies** because each non-key attribute depends on the whole candidate key(2NF).Therefore, there are no non-trivial functional dependencies where the determinant is not a candidate key (Here, Airline no is not a candidate key).
Thus, the "pilot" table is in both BCNF and 3NF.

5. CABIN CREW

Attributes**:**

Let's consider -



Functional Dependency:

{A → B , C , D , E , G , H , I}

{A → F} (Where F is a foreign key)

F : AirlineNo -> AirlineName, Country (From Airlines table)

This means that "CabinId" determines all the other attributes in the table Including AirlineNo. The "CabinCrew" table is already in 1NF because all the attributes are atomic and there are no repeating groups.

Hence, "CabinId" is a candidate key for the "CabinCrew" table because it uniquely identifies each row. There are **no partial dependencies** because each non-key attribute depends on the whole candidate key (2NF).Therefore, there are no non-trivial functional dependencies where the determinant is not a candidate key (Here, Airline no is not a candidate key).

Thus, the "CabinCrew" table is in both BCNF and 3NF.

6. **RESERVATION**

Attributes**:**

1. PassengerNo
2. AirlineNo

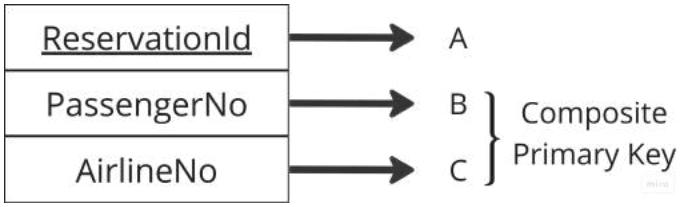
Here, There is no and Without having a primary key in the reservation table,This can create confusion and make it impossible to ensure that each reservation is unique.
This could also cause: Data duplication, Difficulty in querying data,Data integrity issues.

Hence to Solve this issue let's add a new attribute called “ReservationId” and make it a Primary Key. And also to ensure that a passenger can make multiple reservations with different airlines, but not multiple reservations with the same airline.
The reservation table should include a **composite primary key** consisting of both passengerNo and airlineNo to ensure this issue.

After Making these changes :

Attributes:

Let's consider,



Therefore - **Eg**: (P001,A001),(P002,A001),(P001,A002)-this Entry will be allowed(since P001 can make reservations with both A001 and A002 Airlines. But (P001,A001) Cannot be inserted because (P001,A001) entry already exists and duplicacy is eliminated with composite Primary key.
The given table reservation has a composite primary key (passengerNo, airlineNo) and the following

Functional dependencies:

{ A -> B } (B is Foreign Key from Passenger Table)
{ A -> C } (C is Foreign Key from Airline Table)

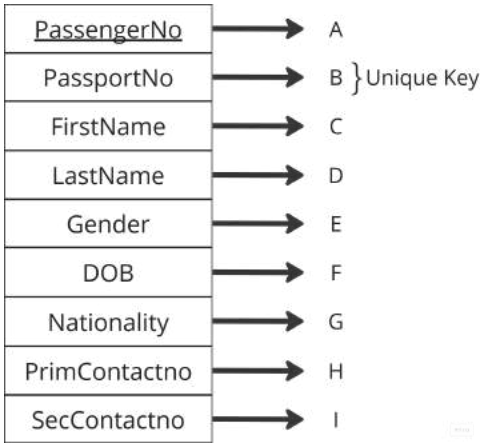
Based on these dependencies, the table is already in **3NF**, because there are **no transitive dependencies** between non-prime attributes and each non-prime attribute is functionally dependent on the primary key.
In this case, both reservationId and the composite key (passengerNo, airlineNo) are candidate keys, since they uniquely identify each row.

Therefore, the table is in **BCNF**, because every determinant is a candidate key.

7. **PASSENGERS**

Attributes**:**

Let's consider -



Functional Dependency:

{A → B , C , D , E , G , H, I }

{B → A , C , D , E , G , H, I } (B is a Unique Key)

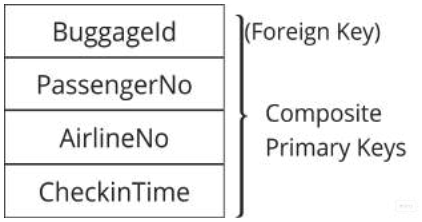
Here there are no transitive dependencies between the attributes, as each attribute is directly dependent on either passengerNo or passportNumber.

The primary key is passengerNo, which uniquely identifies each passenger, and the passportNumber is made unique to ensure data integrity. All the non-key attributes are dependent only on the passengerNo, so there are no partial dependencies.

Therefore, this table is in **3NF** and **BCNF.**

8. **LUGGAGE INFO**

Attributes:



Note : Here a Single Passenger can have multiple Luggages and linked with the same Airline no.

Therefore, entries like (B1,P001,A001,"12-2-21"),(B2,P001,A001,"12-2-21") are allowed and (B1,P001,A001,"12-2-21") is not allowed because will create duplicates.

Functional Dependencies:

PassengerNo, AirlineNo, BuggageId -> CheckinDate

PassengerNo, AirlineNo, CheckinDate -> BuggageId

PassengerNo, BuggageId, CheckinDate -> AirlineNo

AirlineNo, BuggageId, CheckinDate -> PassengerNo

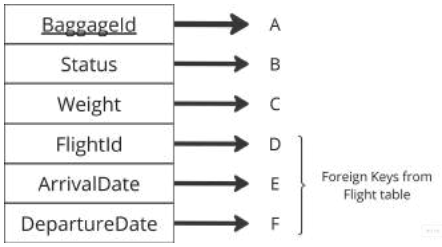
Note In this table, all non-key attributes are directly dependent on the composite primary key (PassengerNo, AirlineNo, BuggageId, CheckinDate) through the given functional dependencies.

As there are no transitive dependencies and each non-key attribute depends on the primary key.

Hence we can conclude that The table now is in **3NF and BCNF form.**

9. **BAGGAGE**

Attributes:



Functional Dependencies:

{ A -> B , C } (Partial Dependency)

{ D , E , F -> A }

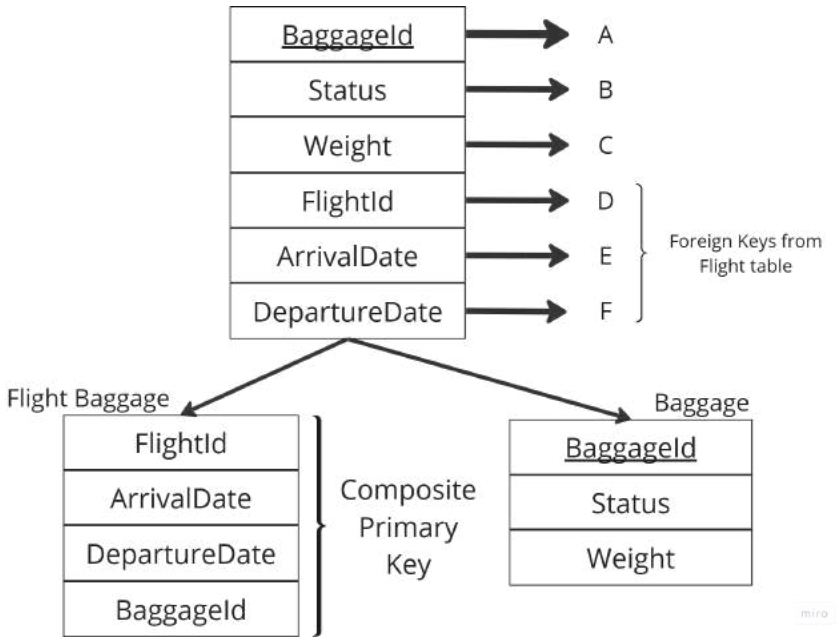
Here there is an issue of Partial Dependency since BaggageId alone cannot determine FlightId.

Also Transitive Dependency because this would also mean {D , E , F -> B , C } .

This means that status and weight are indirectly dependent on flightNo, departureDate, and

departureTime through the intermediary of baggageId, which violates the third normal form (3NF).

To solve the issue of partial dependency and transitive dependency, we can split the table into two tables:



The functional dependencies for the Baggage table are:

{ baggageId -> status, weight }

This table is already in 3NF and BCNF since there is only one candidate key (baggageId) and no transitive dependencies or partial dependencies.

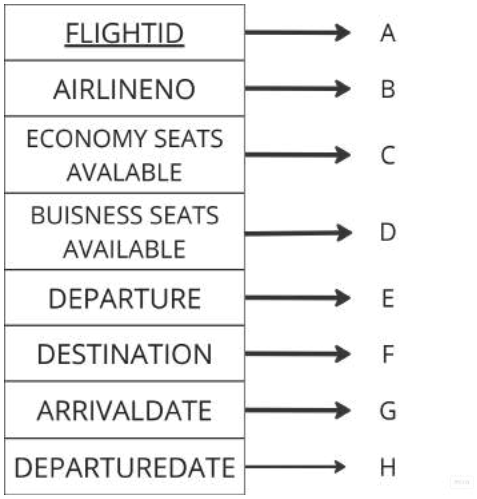
The functional dependencies for the FlightBaggage table are:

{ flightNo, departureDate, ArrivalDate -> baggageId }

This table is also in 3NF and BCNF since there is only one candidate key (flightNo, departureDate, departureTime, baggageId) and no transitive dependencies or partial dependencies.

10. FLIGHT

Attributes:



Functional Dependencies:

{ A -> B , C , D , E , F , G , H } (B is foreign key)

B : AirlineNo -> AirlineName, Country (From Airlines table)

This functional dependency satisfies the requirements of 1NF(First Normal Form) because there is no multivalued attribute hence, each row in the table represents a unique Flight details with a distinct primary key ("FlightID"). and 2NF (Second Normal Form) because all non-key attributes are fully functionally dependent on the primary key ("FlightID"). And since there is no partial dependency and transitivity , the table is also in 3NF and BCNF.

11. BOOKED FLIGHT

1. FlightNo
2. SeatNo
3. TicketClass

Here FlightNo and SeatNo are composite Primary Keys.

Functional dependencies:

seatNo, flightNo -> ticketClass

This condition ensures that a seat number (e.g. 21C) can only have one ticket class (e.g.

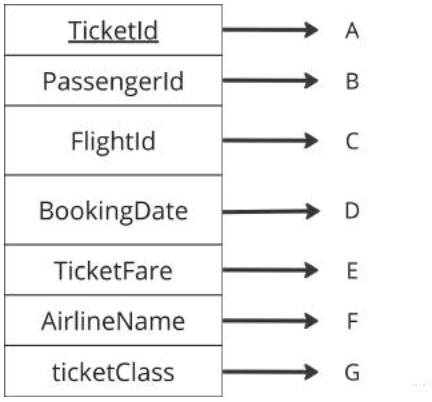
Economy or Business) booked at a time for a given flight (e.g. F001). In other words, if a seat is booked for Economy class on a particular flight, it cannot also be booked for Business class on the same flight.

This functional dependency does not violate 3NF because seatNo and flightNo together form the primary key of the table. Therefore, ticketClass is fully dependent on the primary key and there are no partial dependencies.

Here , flightNo = flightId (from the reference to the flight table)

Since there are no partial dependencies or transitive dependencies, the bookedFlight table is in **3NF**. Additionally, it is also in **BCNF** since all functional dependencies are already in the form of candidate keys.

12. TICKET



Functional Dependencies:

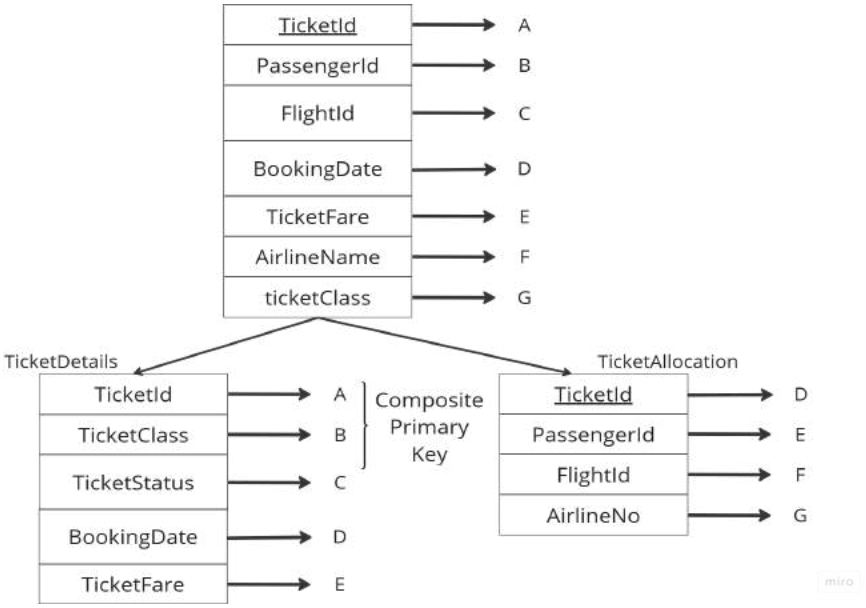
{ A -> B , C , D , E , G }

{ B , C -> F }

This will cause transitivity { A -> F }.

In the context of an airline system, it may not be possible to uniquely determine the airline name with just the ticket ID. However, the airline name can be determined based on the passenger ID and the flight ID.

To solve this issue we can **split the table into two tables**:



Here we have replaced the AirlineName attribute with AirlineNo in the TicketAllocation table.

In Ticket Details ,The ticketId column is set to be a **unique key**(ticketId is also a foreign key from TicketAllocation), ensuring that each ticket has a unique identifier. Additionally, the combination of ticketId and ticketClass is set as the primary key, **ensuring that each ticket has a unique ticket class combination.**

The TicketAllocation table is designed to store information about the allocation of tickets to passengers for specific flights and airlines, and the foreign keys enforce the relationship between the ticket and the passenger, and between the ticket and the flight.

The FOREIGN KEY constraint ensures that the ticketId column in the TicketDetails table references the ticket_id column in the TicketAllocation table.

Functional Dependencies:

TicketDetails:

{A , B -> C , D , E} (here with respect to TicketClass the ticketFare will differ)

There is **no transitive dependency** in this relation. (ticketId, ticketClass) can be a candidate key for this relation, since it uniquely identifies each tuple.

Therefore, TicketDetails is in both **3NF and BCNF.**

TicketAllocation:

{D -> E , F , G}

There is **no transitive dependency** in this relation.

ticketId can be a candidate key for this relation, since it uniquely identifies each tuple.

Therefore, TicketAllocation is in both **3NF and BCNF.**

CREATING AND POPULATING TABLES

create database AirlineSystem;
use AirlineSystem;

1) **create table Administrator (username varchar(25),password varchar(25) primary key);**

The table "Administrator" has two columns: "username" and "password", where "password" is the primary key.

2) create table Airlines (airlineNo varchar(25) primary key ,airlineName varchar(25),country varchar(25));

"Airlines" has three columns: "airlineNo" (primary key), "airlineName", and "country".

The "airlineNo" column is set as the primary key, which means that each value in that column must be unique and cannot be null.

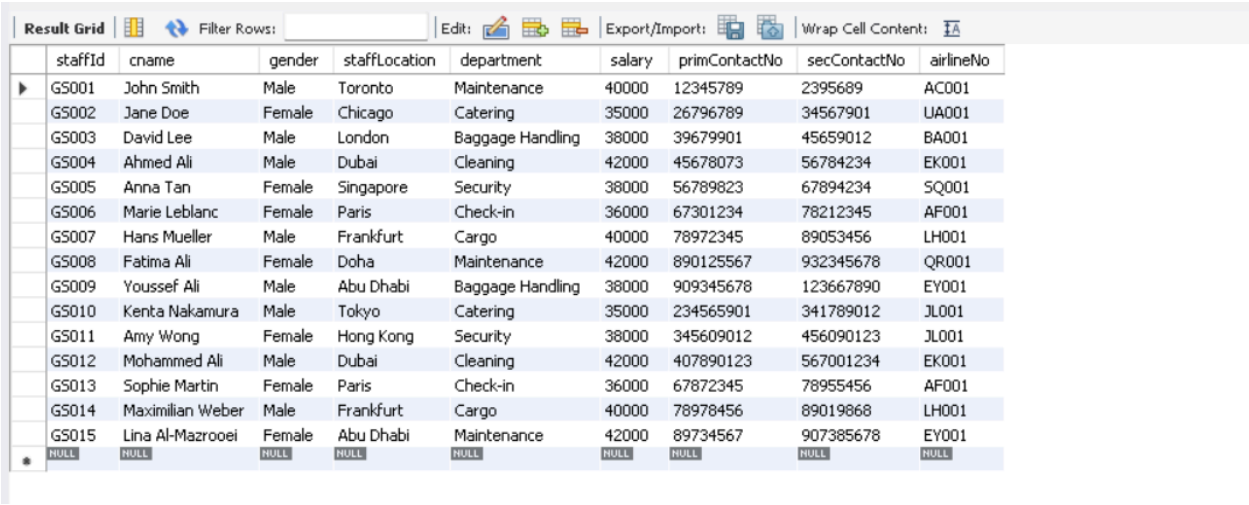
Result Grid | Filter Rows: | Edit:

airlineNo	airlineName	country
AC001	Air Canada	Canada
AF001	Air France	France
BA001	British Airways	UK
EK001	Emirates	UAE
EY001	Etihad Airways	UAE
JL001	Japan Airlines	Japan
LH001	Lufthansa	Germany
QR001	Qatar Airways	Qatar
SQ001	Singapore Airlines	Singapore
UA001	United Airlines	USA
NULL	NULL	NULL

3)create table GroundStaff (staffId varchar(25) primary key, cname varchar(50), gender char(15), staffLocation varchar(25), department varchar(25),salary int(10), primContactNo int(22) not null, secContactNo int(22), airlineNo varchar(25),FOREIGN KEY (airlineNo) REFERENCES airlines(airlineNo));

This table "GroundStaff" has a primary key "staffId" and a foreign key "airlineNo" that references the "airlines" table.

This ensures that each record in the "GroundStaff" table is associated with a valid airline record in the “Airlines" table.



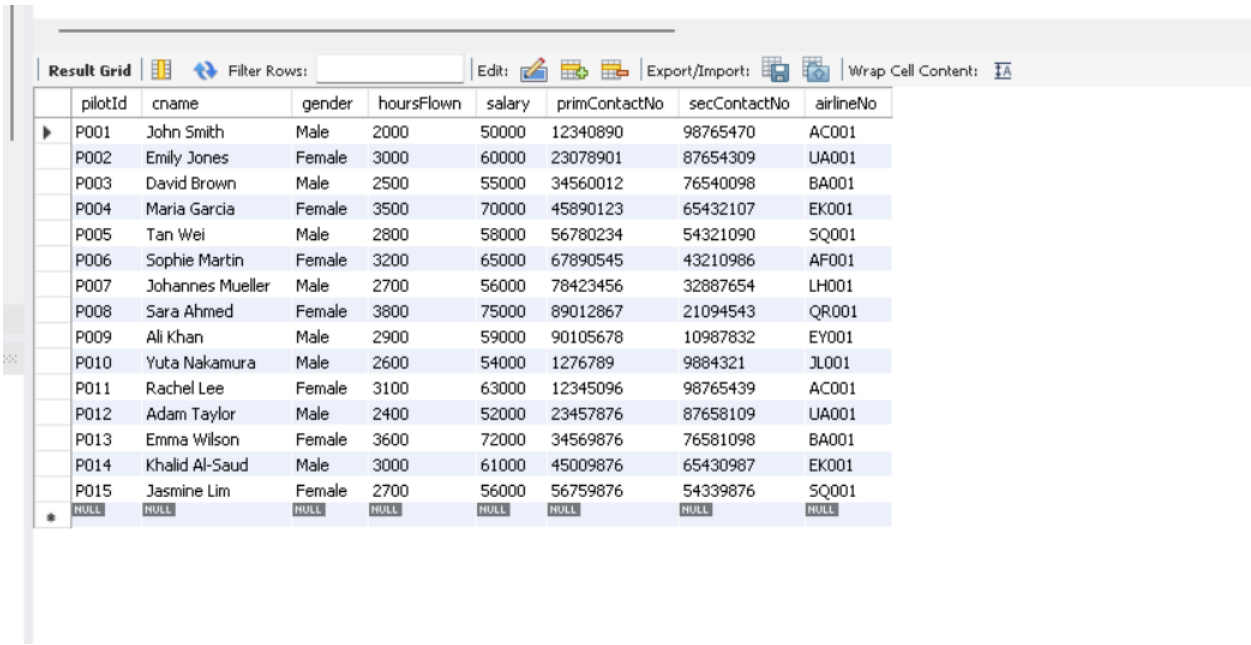
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

staffId	cname	gender	staffLocation	department	salary	primContactNo	secContactNo	airlineNo
GS001	John Smith	Male	Toronto	Maintenance	40000	12345789	2395689	AC001
GS002	Jane Doe	Female	Chicago	Catering	35000	26796789	34567901	UA001
GS003	David Lee	Male	London	Baggage Handling	38000	39679901	45659012	BA001
GS004	Ahmed Ali	Male	Dubai	Cleaning	42000	45678073	56784234	EK001
GS005	Anna Tan	Female	Singapore	Security	38000	56789823	67894234	SQ001
GS006	Marie Leblanc	Female	Paris	Check-in	36000	67301234	78212345	AF001
GS007	Hans Mueller	Male	Frankfurt	Cargo	40000	78972345	89053456	LH001
GS008	Fatima Ali	Female	Doha	Maintenance	42000	890125567	932345678	QR001
GS009	Youssef Ali	Male	Abu Dhabi	Baggage Handling	38000	909345678	123667890	EY001
GS010	Kenta Nakamura	Male	Tokyo	Catering	35000	234565901	341789012	JL001
GS011	Amy Wong	Female	Hong Kong	Security	38000	345609012	456090123	JL001
GS012	Mohammed Ali	Male	Dubai	Cleaning	42000	407890123	567001234	EK001
GS013	Sophie Martin	Female	Paris	Check-in	36000	67872345	78955456	AF001
GS014	Maximilian Weber	Male	Frankfurt	Cargo	40000	78978456	89019868	LH001
GS015	Lina Al-Mazrooei	Female	Abu Dhabi	Maintenance	42000	89734567	907385678	EY001
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4) create table Pilot (pilotId varchar(25) primary key , cname varchar(50), gender char(15), hoursFlown int(10), salary int(10), primContactNo int(11) not null , secContactNo int (11), airlineNo varchar(25),FOREIGN KEY (airlineNo) REFERENCES airlines(airlineNo));

The table "Pilot" has a primary key column "pilotId", and a foreign key column "airlineNo" that references the primary key column "airlineNo" in the "Airlines" table.

This constraint ensures that data integrity is maintained between the "Pilot" and "Airlines" tables.



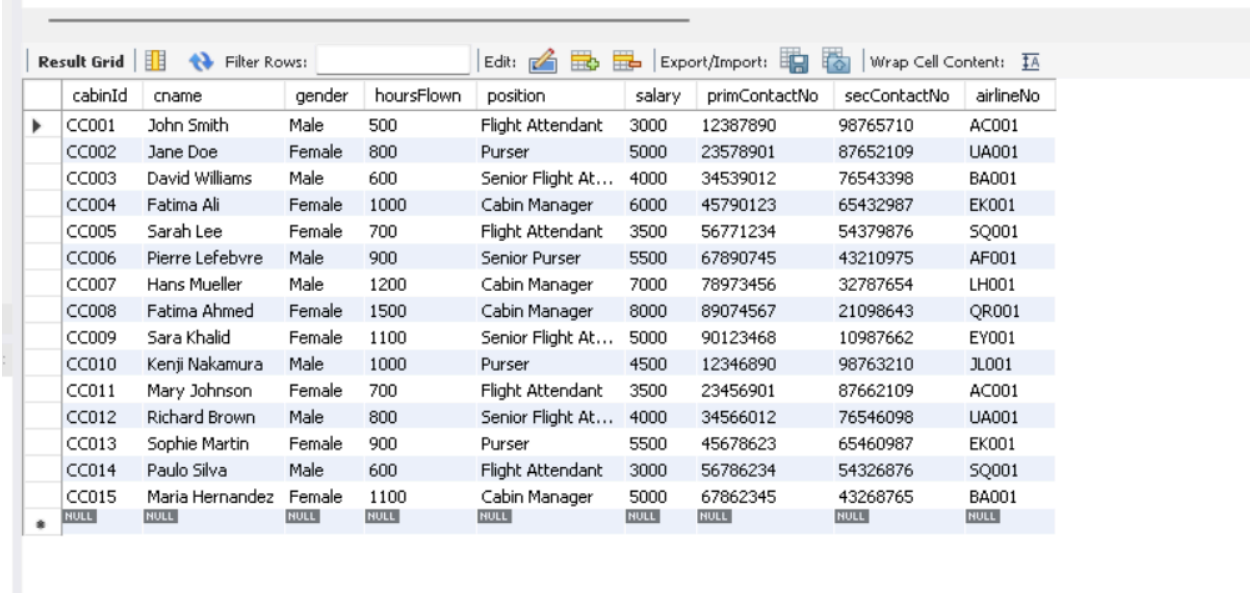
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

pilotId	cname	gender	hoursFlown	salary	primContactNo	secContactNo	airlineNo
P001	John Smith	Male	2000	50000	12340890	98765470	AC001
P002	Emily Jones	Female	3000	60000	23078901	87654309	UA001
P003	David Brown	Male	2500	55000	34560012	76540098	BA001
P004	Maria Garcia	Female	3500	70000	45890123	65432107	EK001
P005	Tan Wei	Male	2800	58000	56780234	54321090	SQ001
P006	Sophie Martin	Female	3200	65000	67890545	43210986	AF001
P007	Johannes Mueller	Male	2700	56000	78423456	32887654	LH001
P008	Sara Ahmed	Female	3800	75000	89012867	21094543	QR001
P009	Ali Khan	Male	2900	59000	90105678	10987832	EY001
P010	Yuta Nakamura	Male	2600	54000	1276789	9884321	JL001
P011	Rachel Lee	Female	3100	63000	12345096	98765439	AC001
P012	Adam Taylor	Male	2400	52000	23457876	87658109	UA001
P013	Emma Wilson	Female	3600	72000	34569876	76581098	BA001
P014	Khalid Al-Saud	Male	3000	61000	45009876	65430987	EK001
P015	Jasmine Lim	Female	2700	56000	56759876	54339876	SQ001
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5) create table CabinCrew (cabinId varchar(25) primary key ,cname varchar(50), gender char(15), hoursFlown int(10), position varchar(25), salary int(10), primContactNo int(40) not null, secContactNo int (40), airlineNo varchar(25),FOREIGN KEY (airlineNo) REFERENCES airlines(airlineNo));

The table "CabinCrew" has a foreign key "airlineNo" which references the primary key "airlineNo" of the table "airlines". The first column "cabinId" is set as the primary key, which means that each value in this column must be unique and not null.

This constraint ensures that only valid airlines can be associated with cabin crew members in the system.

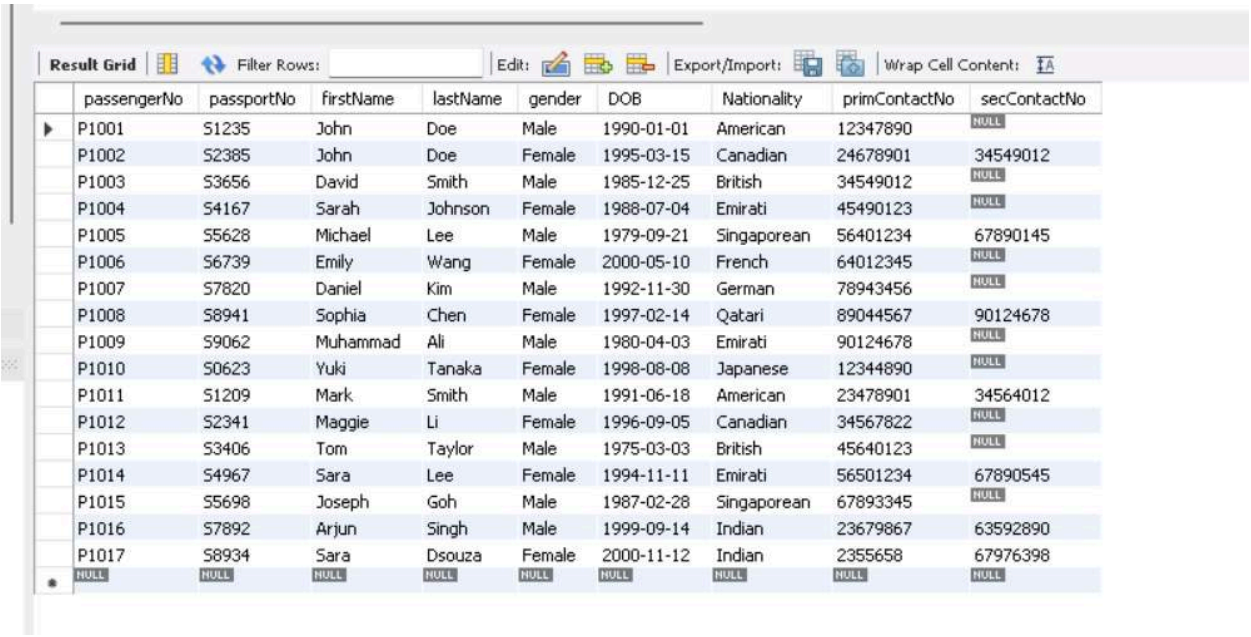

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

cabinId	cname	gender	hoursFlown	position	salary	primContactNo	secContactNo	airlineNo
CC001	John Smith	Male	500	Flight Attendant	3000	12387890	98765710	AC001
CC002	Jane Doe	Female	800	Purser	5000	23578901	87652109	UA001
CC003	David Williams	Male	600	Senior Flight At...	4000	34539012	76543398	BA001
CC004	Fatima Ali	Female	1000	Cabin Manager	6000	45790123	65432987	EK001
CC005	Sarah Lee	Female	700	Flight Attendant	3500	56771234	54379876	SQ001
CC006	Pierre Lefebvre	Male	900	Senior Purser	5500	67890745	43210975	AF001
CC007	Hans Mueller	Male	1200	Cabin Manager	7000	78973456	32787654	LH001
CC008	Fatima Ahmed	Female	1500	Cabin Manager	8000	89074567	21098643	QR001
CC009	Sara Khalid	Female	1100	Senior Flight At...	5000	90123468	10987662	EY001
CC010	Kenji Nakamura	Male	1000	Purser	4500	12346890	98763210	JL001
CC011	Mary Johnson	Female	700	Flight Attendant	3500	23456901	87662109	AC001
CC012	Richard Brown	Male	800	Senior Flight At...	4000	34566012	76546098	UA001
CC013	Sophie Martin	Female	900	Purser	5500	45678623	65460987	EK001
CC014	Paulo Silva	Male	600	Flight Attendant	3000	56786234	54326876	SQ001
CC015	Maria Hernandez	Female	1100	Cabin Manager	5000	67862345	43268765	BA001
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6) CREATE TABLE Passengers (passengerNo varchar(25) PRIMARY KEY, passportNo VARCHAR(25) UNIQUE, firstName VARCHAR(25),lastName VARCHAR(25), gender CHAR(15), DOB DATE, Nationality VARCHAR(50), primContactNo INT(10) not null,secContactNo INT(10));

The "Passengers" table has a primary key of "passengerNo" and a unique constraint on "passportNo". It contains columns for the passenger's first and last name, gender, date of birth, nationality, and primary and secondary contact numbers. There are no foreign keys in this

table.



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

passengerNo	passportNo	firstName	lastName	gender	DOB	Nationality	primContactNo	secContactNo
P1001	S1235	John	Doe	Male	1990-01-01	American	12347890	NULL
P1002	S2385	John	Doe	Female	1995-03-15	Canadian	24678901	34549012
P1003	S3656	David	Smith	Male	1985-12-25	British	34549012	NULL
P1004	S4167	Sarah	Johnson	Female	1988-07-04	Emirati	45490123	NULL
P1005	S5628	Michael	Lee	Male	1979-09-21	Singaporean	56401234	67890145
P1006	S6739	Emily	Wang	Female	2000-05-10	French	64012345	NULL
P1007	S7820	Daniel	Kim	Male	1992-11-30	German	78943456	NULL
P1008	S8941	Sophia	Chen	Female	1997-02-14	Qatari	89044567	90124678
P1009	S9062	Muhammad	Ali	Male	1980-04-03	Emirati	90124678	NULL
P1010	S0623	Yuki	Tanaka	Female	1998-08-08	Japanese	12344890	NULL
P1011	S1209	Mark	Smith	Male	1991-06-18	American	23478901	34564012
P1012	S2341	Maggie	Li	Female	1996-09-05	Canadian	34567822	NULL
P1013	S3406	Tom	Taylor	Male	1975-03-03	British	45640123	NULL
P1014	S4967	Sara	Lee	Female	1994-11-11	Emirati	56501234	67890545
P1015	S5698	Joseph	Goh	Male	1987-02-28	Singaporean	67893345	NULL
P1016	S7892	Arjun	Singh	Male	1999-09-14	Indian	23679867	63592890
P1017	S8934	Sara	Dsouza	Female	2000-11-12	Indian	2355658	67976398
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

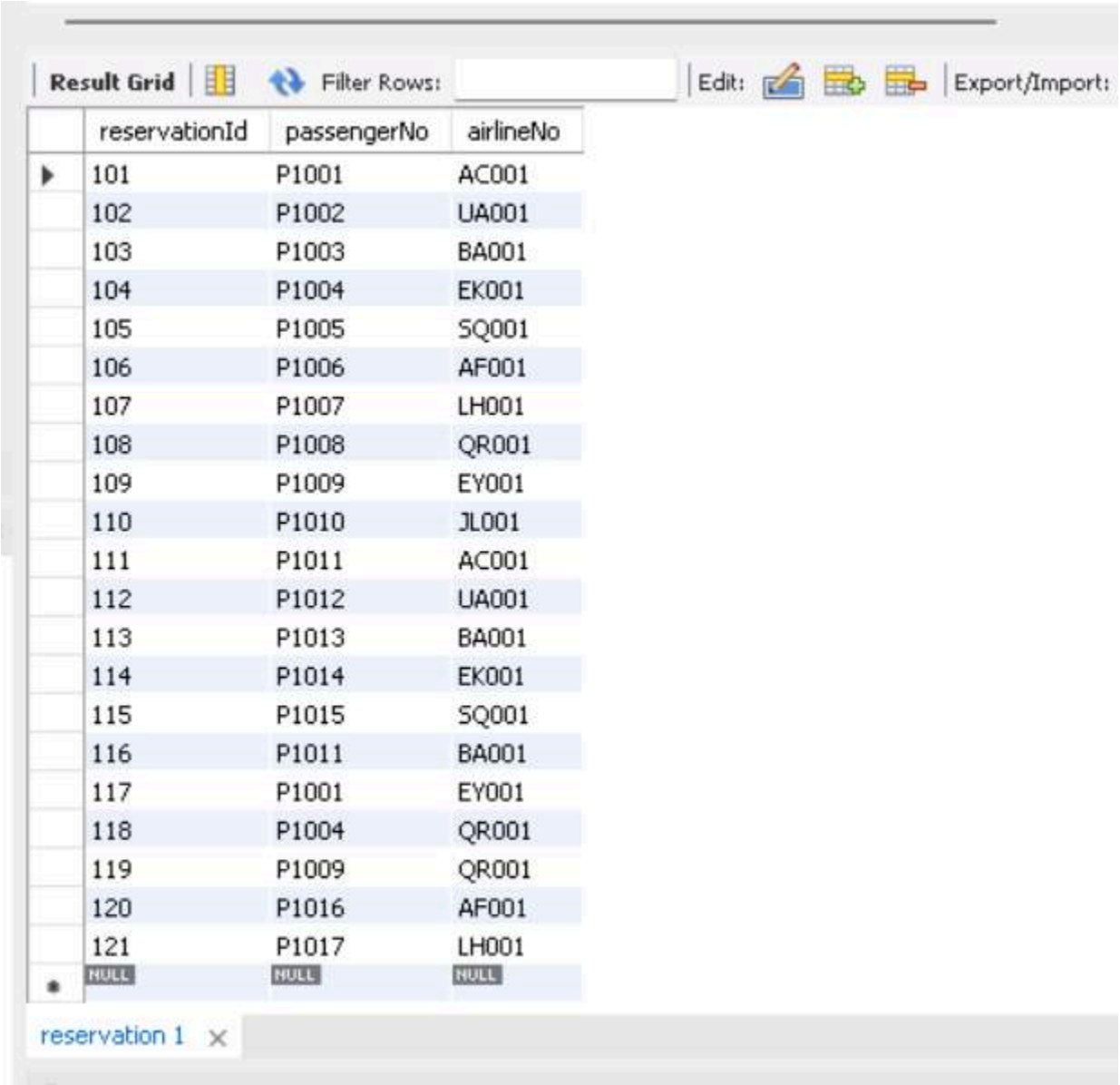
7) CREATE TABLE Reservation (reservationId int PRIMARY KEY,passengerNo varchar(25), airlineNo varchar(25), FOREIGN KEY (passengerNo) REFERENCES passengers(passengerNo),FOREIGN KEY (airlineNo) REFERENCES airlines(airlineNo),CONSTRAINT unique_reservation UNIQUE (passengerNo, airlineNo));

Table `Reservation` has a primary key `reservationId`, foreign keys `passengerNo` referencing `passengers(passengerNo)` and `airlineNo` referencing `airlines(airlineNo)`, and a unique constraint on the combination of `passengerNo` and `airlineNo`.

The "reservationId" column is set as the primary key, which means that each value in this column must be unique and not null.

The foreign key constraint ensures that each value in these columns must exist in the referenced tables or will be rejected.

The Unique constraint ensures that each reservation can only be made once by a passenger on a specific airline.



Result Grid | Filter Rows: | Edit: | Export/Import:

reservationId	passengerNo	airlineNo
101	P1001	AC001
102	P1002	UA001
103	P1003	BA001
104	P1004	EK001
105	P1005	SQ001
106	P1006	AF001
107	P1007	LH001
108	P1008	QR001
109	P1009	EY001
110	P1010	JL001
111	P1011	AC001
112	P1012	UA001
113	P1013	BA001
114	P1014	EK001
115	P1015	SQ001
116	P1011	BA001
117	P1001	EY001
118	P1004	QR001
119	P1009	QR001
120	P1016	AF001
121	P1017	LH001
NULL	NULL	NULL

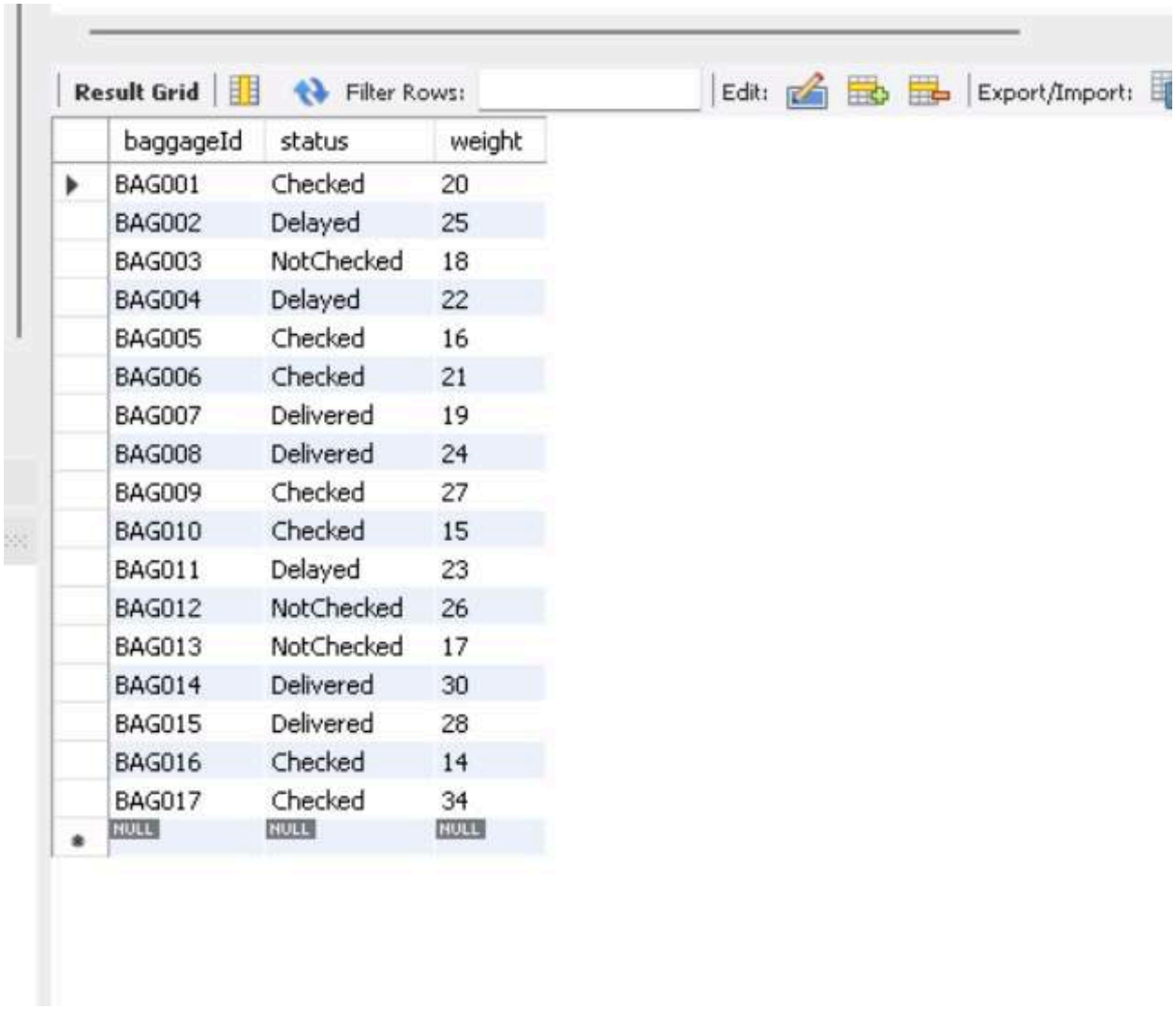
reservation 1

8) CREATE TABLE Baggage (baggageId VARCHAR(25) PRIMARY KEY, status VARCHAR(25), weight INT(10));

This table has a primary key 'baggageId' and columns 'status' and 'weight' for baggage status and weight. There are no foreign keys.Therefore, there are no relationships between the "Baggage" table and any other tables in the database.

The "baggageId" column is set as the primary key, which means that each value in this column must be unique and not null.

It is a standalone table that stores information about the baggage, such as the baggage ID, status, and weight.



baggageId	status	weight
BAG001	Checked	20
BAG002	Delayed	25
BAG003	NotChecked	18
BAG004	Delayed	22
BAG005	Checked	16
BAG006	Checked	21
BAG007	Delivered	19
BAG008	Delivered	24
BAG009	Checked	27
BAG010	Checked	15
BAG011	Delayed	23
BAG012	NotChecked	26
BAG013	NotChecked	17
BAG014	Delivered	30
BAG015	Delivered	28
BAG016	Checked	14
BAG017	Checked	34
NULL	NULL	NULL

9) CREATE TABLE LuggageInfo (passengerNo varchar(25), airlineNo varchar (25),baggageId varchar(25), CheckinTime TIME,PRIMARY KEY (PassengerNo, AirlineNo, baggageId, CheckinTime),FOREIGN KEY (PassengerNo,airlineNo)

REFERENCES reservation(PassengerNo,airlineNo),FOREIGN KEY (baggageId) REFERENCES Baggage(baggageId));

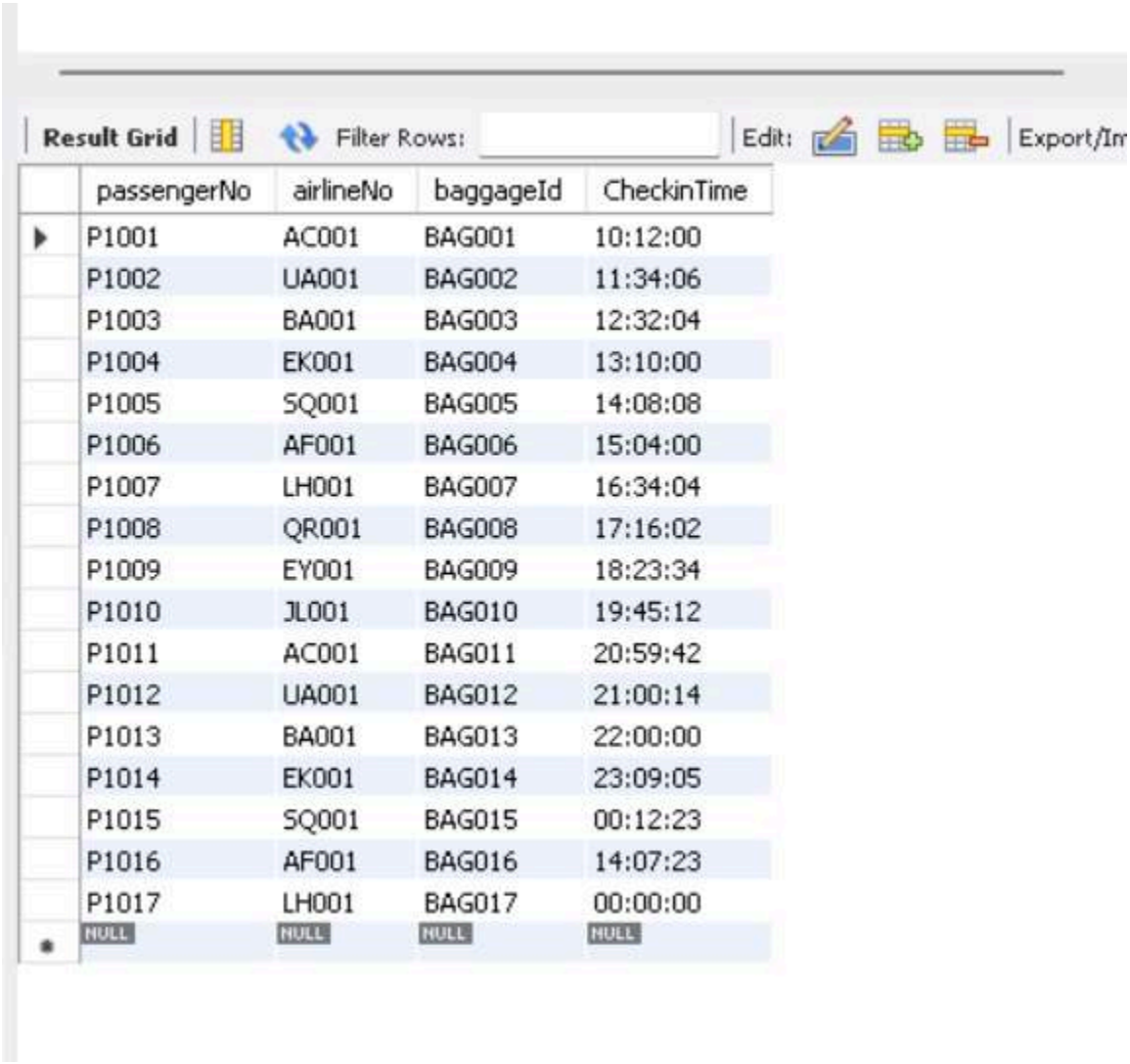
The "LuggageInfo" table has foreign key constraints that reference the `reservation` and `baggage` tables.The first three columns ("passengerNo", "airlineNo", and "baggageId") are foreign keys that reference columns in other tables.

Specifically, "passengerNo" and "airlineNo" together reference the "PassengerNo" and "AirlineNo" columns in the "Reservation" table, while "baggageId" references the "BaggageId" column in the "Baggage" table.

The FOREIGN KEY constraint ensures that each value in the "passengerNo" and "airlineNo" columns of the "LuggageInfo" table must exist in the "PassengerNo" and "AirlineNo" columns of the "Reservation" table, respectively.

Likewise, each value in the "baggageId" column of the "LuggageInfo" table must exist in the "BaggageId" column of the "Baggage" table.

The primary key constraint ensures that each combination of "passengerNo", "airlineNo", "baggageId", and "CheckinTime" is unique in the "LuggageInfo" table.

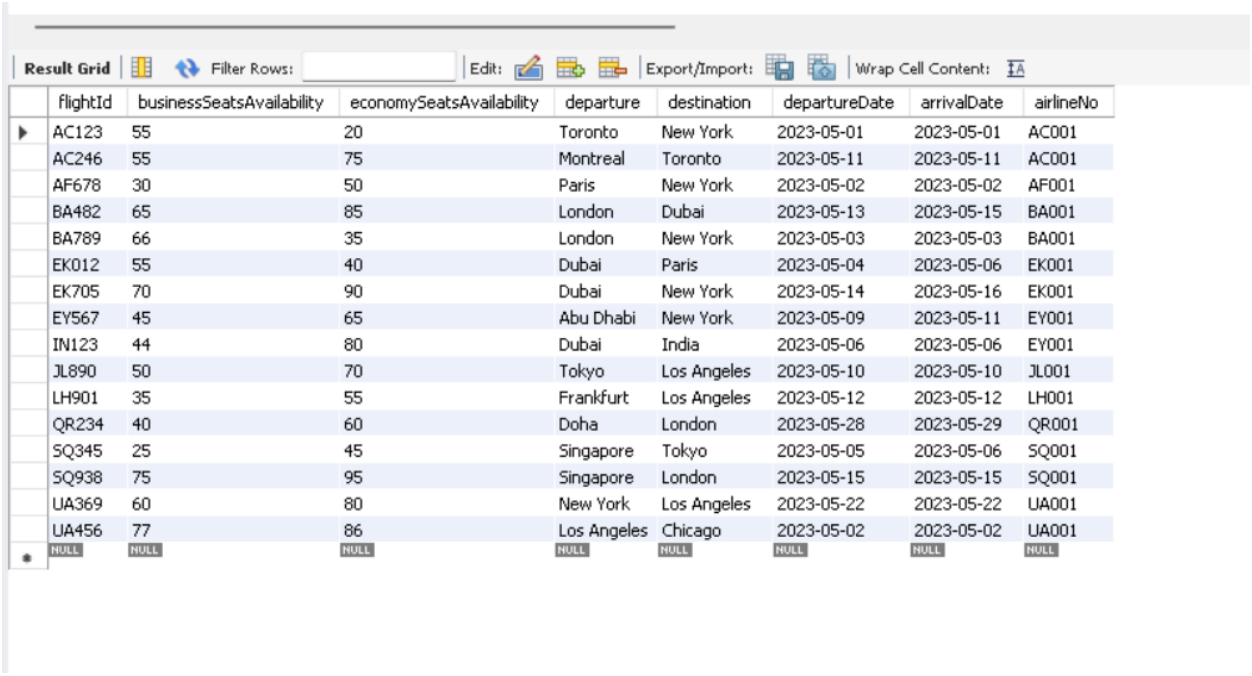

Result Grid | Filter Rows: | Edit: | Export/Im

passengerNo	airlineNo	baggageId	CheckinTime
P1001	AC001	BAG001	10:12:00
P1002	UA001	BAG002	11:34:06
P1003	BA001	BAG003	12:32:04
P1004	EK001	BAG004	13:10:00
P1005	SQ001	BAG005	14:08:08
P1006	AF001	BAG006	15:04:00
P1007	LH001	BAG007	16:34:04
P1008	QR001	BAG008	17:16:02
P1009	EY001	BAG009	18:23:34
P1010	JL001	BAG010	19:45:12
P1011	AC001	BAG011	20:59:42
P1012	UA001	BAG012	21:00:14
P1013	BA001	BAG013	22:00:00
P1014	EK001	BAG014	23:09:05
P1015	SQ001	BAG015	00:12:23
P1016	AF001	BAG016	14:07:23
P1017	LH001	BAG017	00:00:00
NULL	NULL	NULL	NULL

10) CREATE TABLE Flight (flightId varchar(25) PRIMARY KEY, businessSeatsAvailability int(10), economySeatsAvailability int(25), departure varchar(25), destination varchar(25), departureDate date, arrivalDate date, airlineNo varchar(25) REFERENCES Airlines(airlineNo),CHECK(arrivalDate >= departureDate));

This table 'Flight' has a primary key of 'flightId' and a foreign key 'airlineNo' that references the 'airlineNo' column in the 'Airlines' table. It also has a check constraint that ensures the 'arrivalDate' is later than or equal to the 'departureDate',which means that the arrival date should not be earlier than the departure date.

The columns 'businessSeatsAvailability' and 'economySeatsAvailability' indicate the availability of seats in their respective class for a particular flight.



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

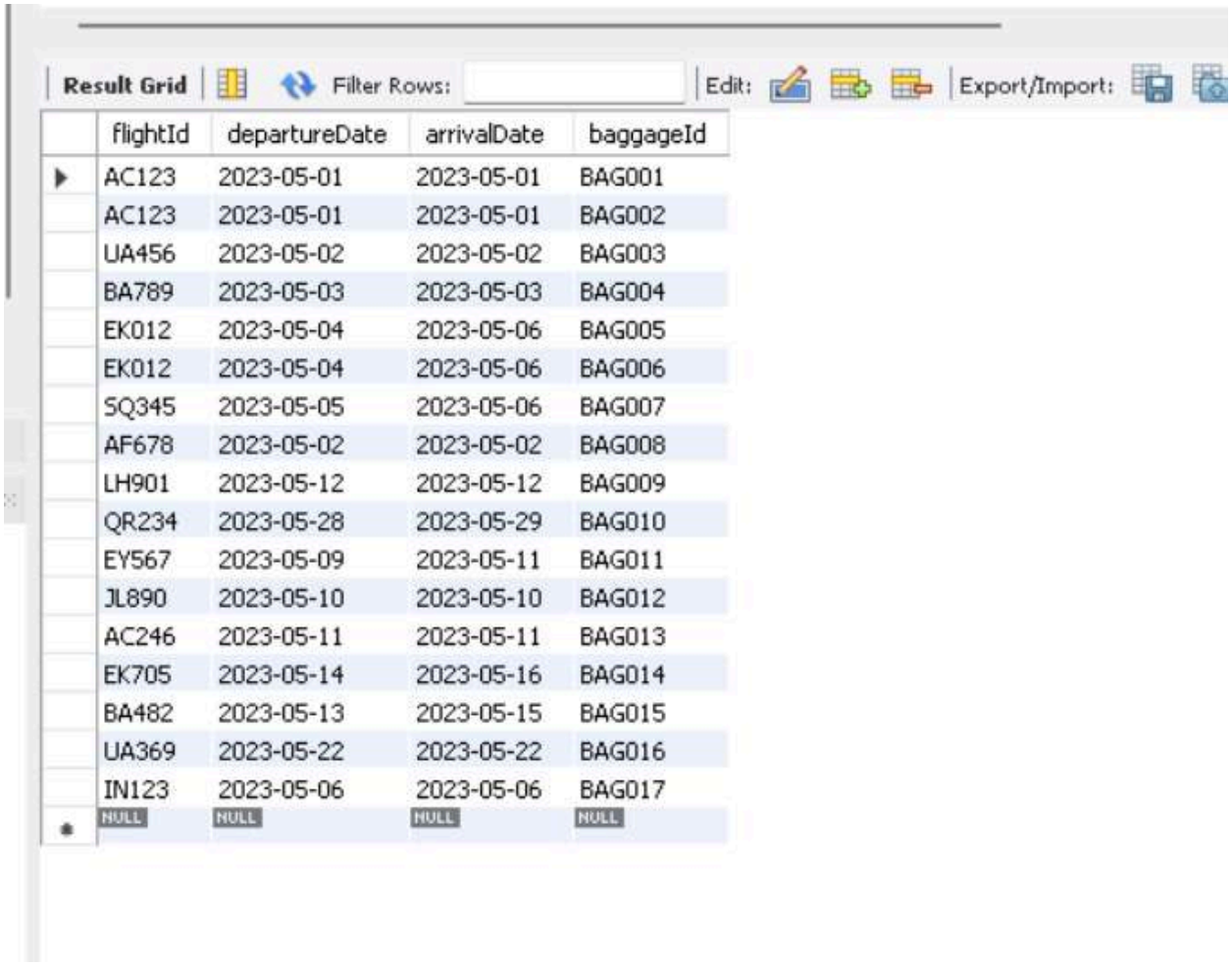
flightId	businessSeatsAvailability	economySeatsAvailability	departure	destination	departureDate	arrivalDate	airlineNo
AC123	55	20	Toronto	New York	2023-05-01	2023-05-01	AC001
AC246	55	75	Montreal	Toronto	2023-05-11	2023-05-11	AC001
AF678	30	50	Paris	New York	2023-05-02	2023-05-02	AF001
BA482	65	85	London	Dubai	2023-05-13	2023-05-15	BA001
BA789	66	35	London	New York	2023-05-03	2023-05-03	BA001
EK012	55	40	Dubai	Paris	2023-05-04	2023-05-06	EK001
EK705	70	90	Dubai	New York	2023-05-14	2023-05-16	EK001
EY567	45	65	Abu Dhabi	New York	2023-05-09	2023-05-11	EY001
IN123	44	80	Dubai	India	2023-05-06	2023-05-06	EY001
JL890	50	70	Tokyo	Los Angeles	2023-05-10	2023-05-10	JL001
LH901	35	55	Frankfurt	Los Angeles	2023-05-12	2023-05-12	LH001
QR234	40	60	Doha	London	2023-05-28	2023-05-29	QR001
SQ345	25	45	Singapore	Tokyo	2023-05-05	2023-05-06	SQ001
SQ938	75	95	Singapore	London	2023-05-15	2023-05-15	SQ001
UA369	60	80	New York	Los Angeles	2023-05-22	2023-05-22	UA001
UA456	77	86	Los Angeles	Chicago	2023-05-02	2023-05-02	UA001
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

11) CREATE TABLE FlightBaggage (flightId VARCHAR(25), departureDate DATE, arrivalDate DATE, baggageId VARCHAR(25), PRIMARY KEY (flightId, departureDate, arrivalDate, baggageId),FOREIGN KEY (flightId, departureDate, arrivalDate) REFERENCES Flight(flightId, departureDate, arrivalDate), FOREIGN KEY (baggageId) REFERENCES Baggage(baggageId));

This table has a composite primary key made up of the 'flightId', 'departureDate', 'arrivalDate', and 'baggageId'. It has two foreign key constraints, one referencing the 'Flight' table and

another referencing the 'Baggage' table.These foreign key constraints ensure that data in this table is consistent with data in the referenced tables.

The primary key uniquely identifies each row in the table based on these columns.


Result Grid | Filter Rows: | Edit: | Export/Import:

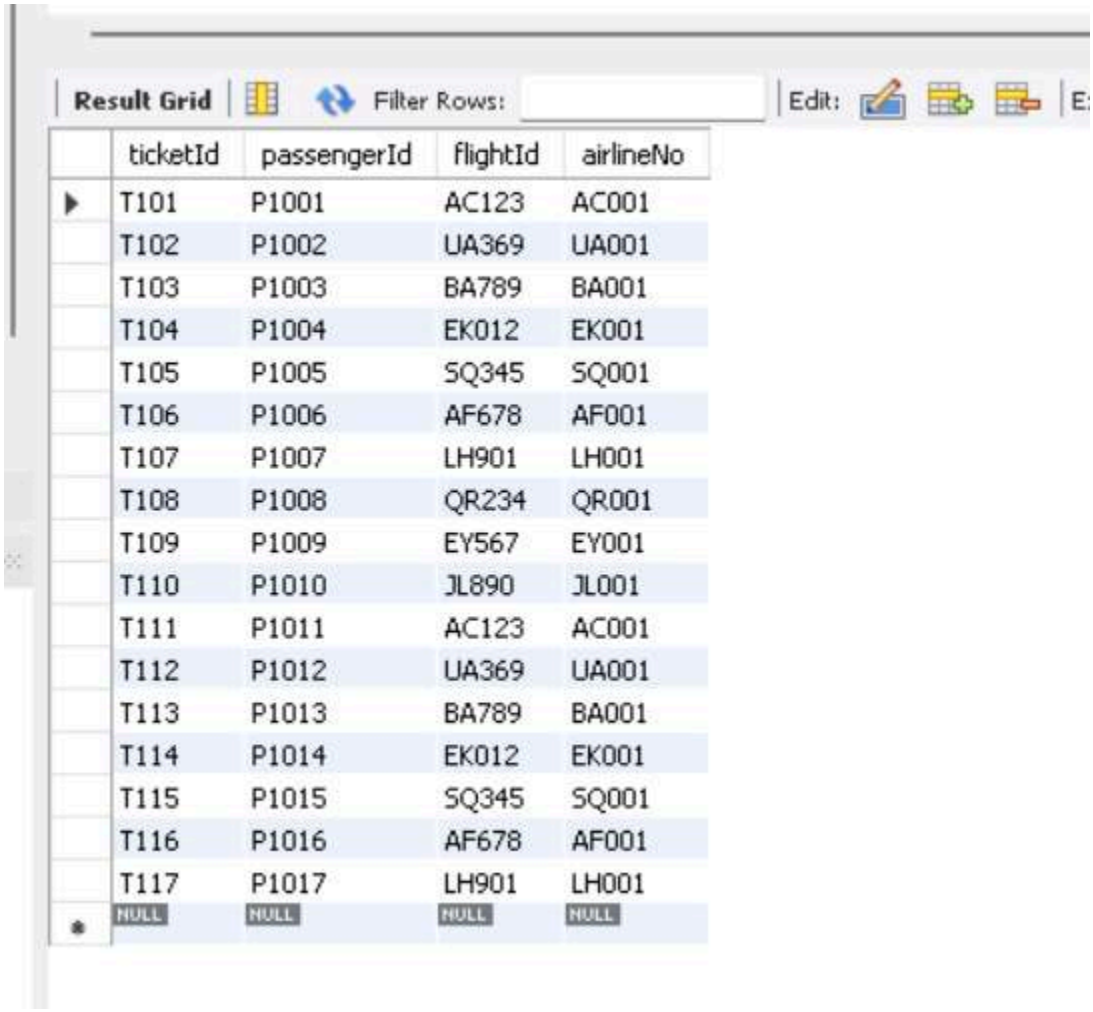
flightId	departureDate	arrivalDate	baggageId
AC123	2023-05-01	2023-05-01	BAG001
AC123	2023-05-01	2023-05-01	BAG002
UA456	2023-05-02	2023-05-02	BAG003
BA789	2023-05-03	2023-05-03	BAG004
EK012	2023-05-04	2023-05-06	BAG005
EK012	2023-05-04	2023-05-06	BAG006
SQ345	2023-05-05	2023-05-06	BAG007
AF678	2023-05-02	2023-05-02	BAG008
LH901	2023-05-12	2023-05-12	BAG009
QR234	2023-05-28	2023-05-29	BAG010
EY567	2023-05-09	2023-05-11	BAG011
JL890	2023-05-10	2023-05-10	BAG012
AC246	2023-05-11	2023-05-11	BAG013
EK705	2023-05-14	2023-05-16	BAG014
BA482	2023-05-13	2023-05-15	BAG015
UA369	2023-05-22	2023-05-22	BAG016
IN123	2023-05-06	2023-05-06	BAG017
NULL	NULL	NULL	NULL

12) CREATE TABLE TicketAllocation (ticketId varchar(25) primary key,passengerId varchar(25), flightId varchar(25), airlineNo varchar(25),FOREIGN KEY (passengerId,airlineNo) REFERENCES reservation(passengerNo,airlineNo), FOREIGN KEY (flightId,airlineNo) REFERENCES Flight(flightId,airlineNo));

The TicketAllocation table has foreign key constraints referencing the Reservation and Flight tables based on passenger, airline, and flight information.

The passengerId and airlineNo columns reference a unique record in the reservation table. The flightId and airlineNo columns reference a unique record in the Flight table.

This ensures data integrity in the TicketAllocation table.



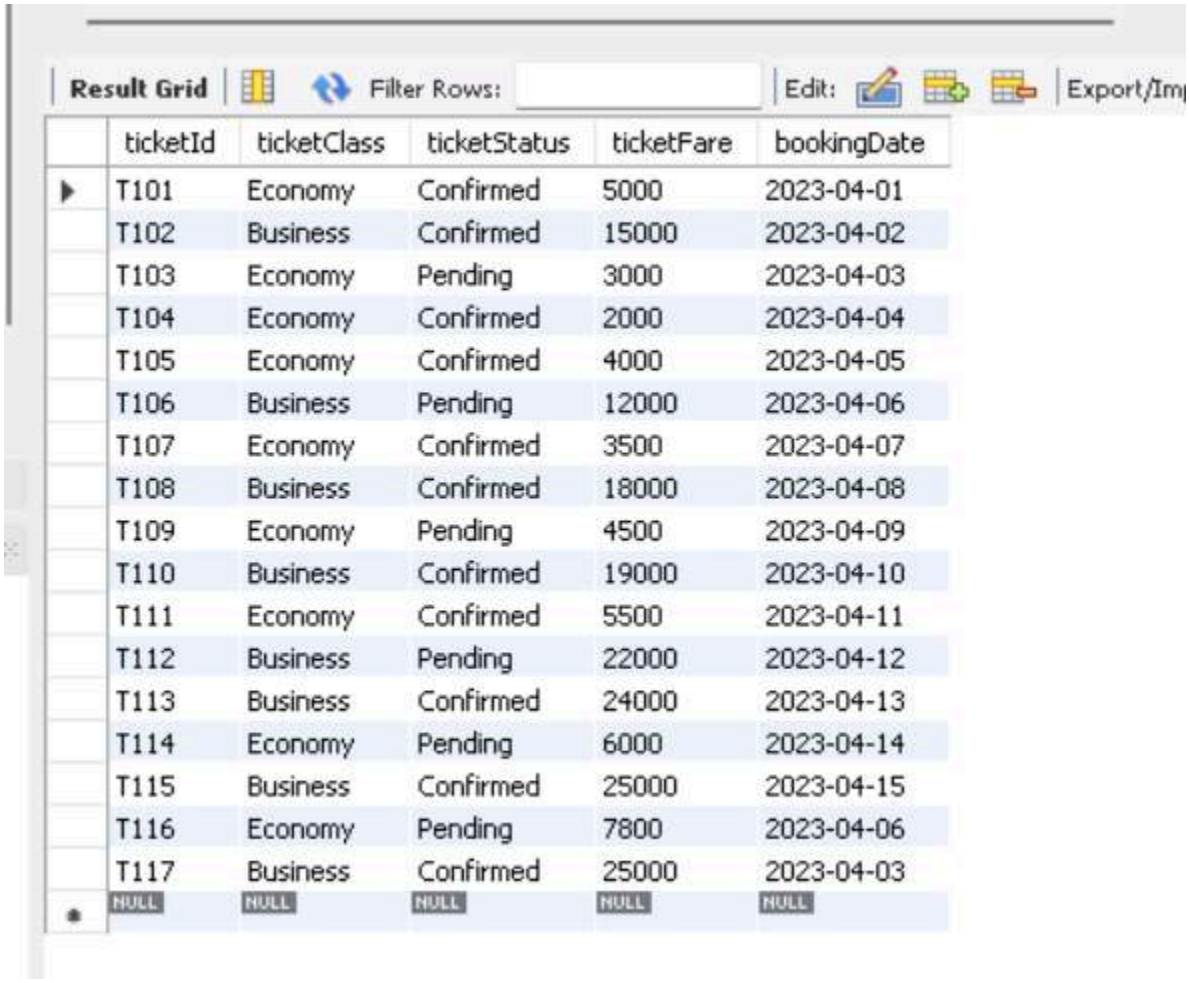
Result Grid | Filter Rows: | Edit:

ticketId	passengerId	flightId	airlineNo
T101	P1001	AC123	AC001
T102	P1002	UA369	UA001
T103	P1003	BA789	BA001
T104	P1004	EK012	EK001
T105	P1005	SQ345	SQ001
T106	P1006	AF678	AF001
T107	P1007	LH901	LH001
T108	P1008	QR234	QR001
T109	P1009	EY567	EY001
T110	P1010	JL890	JL001
T111	P1011	AC123	AC001
T112	P1012	UA369	UA001
T113	P1013	BA789	BA001
T114	P1014	EK012	EK001
T115	P1015	SQ345	SQ001
T116	P1016	AF678	AF001
T117	P1017	LH901	LH001
NULL	NULL	NULL	NULL

13) **CREATE TABLE TicketDetails (ticketId varchar(25) UNIQUE, ticketClass varchar(25), ticketStatus varchar(25), ticketFare int(10), bookingDate date, FOREIGN KEY (ticketId) REFERENCES TicketAllocation(ticketId), PRIMARY KEY (ticketId, ticketClass));**

The"TicketDetails" is created with a primary key and a unique constraint on the column "ticketId". In addition, a foreign key constraint is defined on the same column "ticketId", which references the "ticketId" column in the "TicketAllocation" table. This foreign key constraint ensures that the "ticketId" values in the "TicketDetails" table must exist in the

"TicketAllocation" table.



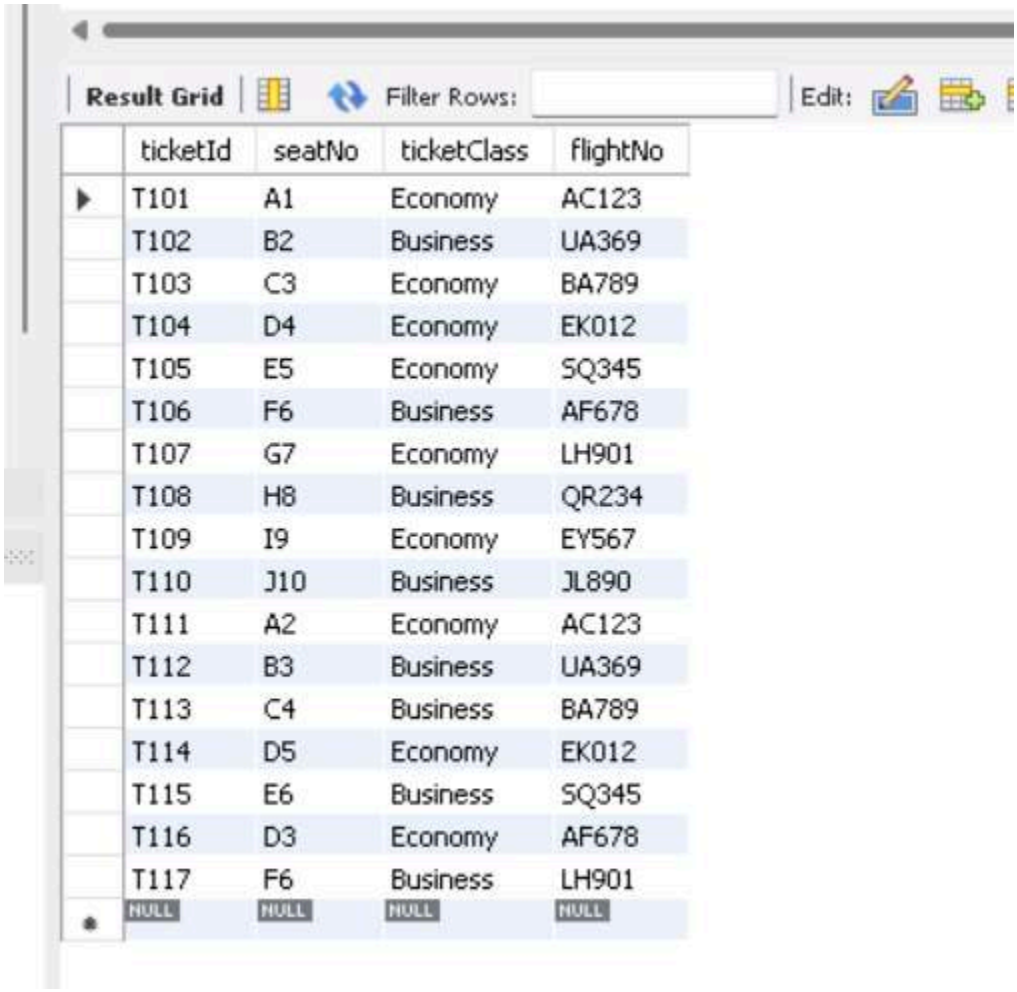
Result Grid | Filter Rows: | Edit: | Export/Im

ticketId	ticketClass	ticketStatus	ticketFare	bookingDate
T101	Economy	Confirmed	5000	2023-04-01
T102	Business	Confirmed	15000	2023-04-02
T103	Economy	Pending	3000	2023-04-03
T104	Economy	Confirmed	2000	2023-04-04
T105	Economy	Confirmed	4000	2023-04-05
T106	Business	Pending	12000	2023-04-06
T107	Economy	Confirmed	3500	2023-04-07
T108	Business	Confirmed	18000	2023-04-08
T109	Economy	Pending	4500	2023-04-09
T110	Business	Confirmed	19000	2023-04-10
T111	Economy	Confirmed	5500	2023-04-11
T112	Business	Pending	22000	2023-04-12
T113	Business	Confirmed	24000	2023-04-13
T114	Economy	Pending	6000	2023-04-14
T115	Business	Confirmed	25000	2023-04-15
T116	Economy	Pending	7800	2023-04-06
T117	Business	Confirmed	25000	2023-04-03
NULL	NULL	NULL	NULL	NULL

14) **CREATE TABLE BookedFlight (ticketId varchar(25), seatNo varchar(25), ticketClass varchar(25), flightNo varchar(25), PRIMARY KEY (seatNo, flightNo), FOREIGN KEY (ticketId,flightNo) REFERENCES TicketAllocation(ticketId,flightId),FOREIGN KEY (ticketId,ticketClass) REFERENCES TicketDetails(ticketId,ticketClass));**

The "BookedFlight" has columns ticketId, seatNo, ticketClass, and flightNo. Two foreign key constraints are defined, with one referencing the "TicketAllocation" table on columns "ticketId" and "flightNo", and the other referencing the "TicketDetails" table on columns "ticketId" and "ticketClass". The primary key is set on the combination of "seatNo" and "flightNo" columns.

This means that the combination of values in these two columns must be unique for each record in the table. No two records can have the same combination of values in the 'seatNo' and 'flightNo' columns. This constraint ensures the uniqueness of the records in the table and facilitates faster querying and indexing of the table.



ticketId	seatNo	ticketClass	flightNo
T101	A1	Economy	AC123
T102	B2	Business	UA369
T103	C3	Economy	BA789
T104	D4	Economy	EK012
T105	E5	Economy	SQ345
T106	F6	Business	AF678
T107	G7	Economy	LH901
T108	H8	Business	QR234
T109	I9	Economy	EY567
T110	J10	Business	JL890
T111	A2	Economy	AC123
T112	B3	Business	UA369
T113	C4	Business	BA789
T114	D5	Economy	EK012
T115	E6	Business	SQ345
T116	D3	Economy	AF678
T117	F6	Business	LH901
NULL	NULL	NULL	NULL

SQL QUERIES

1. To select the staff details along with the airline name for a particular airline using a Nested Query:

```
SELECT gs.staffId, gs.cname, gs.gender, gs.staffLocation, gs.department, gs.salary,
gs.primContactNo, gs.secContactNo,

   (SELECT airlineName FROM Airlines WHERE airlineNo = gs.airlineNo) AS
airlineName

FROM GroundStaff gs

WHERE gs.airlineNo = 'EK001';
```

2. Write a select statement to retrieve the average salary of pilots working in each country where airlines operate, considering their airline's location and years of experience, and the name of the pilot in each country with the highest salary. Using Join and GroupBy Clause.

```
SELECT Airlines.Country, Pilot.CName AS PilotName, Pilot.HoursFlown,
AVG(Pilot.salary) AS avg_salary

FROM Airlines

JOIN Pilot ON Airlines.airlineNo = Pilot.airlineNo

GROUP BY Airlines.country, Pilot.CName, Pilot.HoursFlown;
```

3. To select the passenger details along with the reservation details for a particular airline using an inner join:

```
SELECT ps.passengerNo, ps.passportNo, ps.firstName, ps.lastName, ps.gender, ps.DOB, ps.Nationality, ps.primContactNo, ps.secContactNo, r.reservationId

FROM Passengers ps

INNER JOIN Reservation r ON ps.passengerNo = r.passengerNo

WHERE r.airlineNo = 'SQ001';
```

4. To select the passenger details for a particular reservation:(Nested Query)

```
SELECT * FROM Passengers WHERE passengerNo IN (SELECT passengerNo FROM Reservation WHERE airlineNo = 'EY001');
```

5. To select the baggage details along with the flight and airline details for a particular passenger using a left join:

```
SELECT b.baggageId, b.status, b.weight, f.flightId, f.departure, f.destination, f.departureDate, f.arrivalDate, a.airlineName

FROM Baggage b

LEFT JOIN LuggageInfo li ON b.baggageId = li.baggageId

LEFT JOIN Reservation r ON li.passengerNo = r.passengerNo

LEFT JOIN Flight f ON r.airlineNo = f.airlineNo
```

LEFT JOIN Airlines a ON f.airlineNo = a.airlineNo

WHERE li.passengerNo = 'P1004';

6. Select all passengers who have made a reservation with a specific airline:(Nested Query)

SELECT * FROM passengers WHERE passengerNo IN (SELECT passengerNo FROM reservation WHERE airlineNo = 'QR001');

7. Select all flights that depart from a specific location and have at least one available business class seat:

SELECT * FROM flight WHERE departure = 'Dubai' AND businessSeatsAvailability > 0;

8. Select all booked flights for a specific passenger:

SELECT * FROM bookedflight WHERE ticketId IN (SELECT ticketId FROM ticketallocation WHERE passengerNo = 'P1009');

9. Select the total number of reservations made by each airline:

SELECT Airlines.airlineName, COUNT(*) AS total_reservations

FROM Airlines

JOIN Reservation ON Airlines.airlineNo = Reservation.airlineNo

GROUP BY Airlines.airlineName;

10.Write the SQL query to extract the flightIds and airlineNos for which there are two different flightIds with the same airlineNo in the Flight table:(Using Self Join)

```
SELECT f1.flightId, f1.airlineNo

FROM Flight f1

INNER JOIN Flight f2

ON f1.airlineNo = f2.airlineNo AND f1.flightId <> f2.flightId;
```

SQL FUNCTIONS

Functions are a set of instructions that take some input and perform a specific task and return some value. We have created the following functions for our database as follows:

1. A function that returns the total number of passengers for a given airline:

```
CREATE FUNCTION getPassengerCountForAirline(airlineNo VARCHAR(25))

RETURNS INT

DETERMINISTIC

BEGIN

    DECLARE passengerCount INT;
```

```
    SELECT COUNT(*) INTO passengerCount
    FROM Reservation
    WHERE Reservation.airlineNo = airlineNo;
    RETURN passengerCount;
END //
```

SELECT getPassengerCountForAirline('SQ001');

2. A function that returns the average salary of all pilots for a given airline:

```
CREATE FUNCTION getAverageSalaryForPilots(x VARCHAR(25))
RETURNS DECIMAL(10,2) DETERMINISTIC
BEGIN
    DECLARE avgSalary DECIMAL(10,2);
    SELECT AVG(salary) INTO avgSalary
    FROM Pilot WHERE Pilot.airlineNo = x;
    RETURN avgSalary;
END :
SELECT getAverageSalaryForPilots('AC001');
```

SELECT getAverageSalaryForPilots('AC001');

3. A function that returns the total number of bags checked in by a given passenger for a given airline:

delimiter :

CREATE FUNCTION getBagCountForPassenger(passengerNo VARCHAR(25), airlineNo VARCHAR(25))

RETURNS INT deterministic

BEGIN

DECLARE bagCount INT;

SELECT COUNT(*) INTO bagCount

FROM LuggageInfo

WHERE LuggageInfo.passengerNo = passengerNo AND LuggageInfo.airlineNo = airlineNo;

RETURN bagCount;

END :

SELECT getBagCountForPassenger('P1001','AC001');

4. Function to get the number of available economy seats on a given flight:

delimiter :

CREATE FUNCTION get_num_available_economy_seats(flight_id VARCHAR(25), departure_date DATE)

RETURNS INT deterministic

BEGIN

DECLARE num_available_seats INT;

SELECT economySeatsAvailability INTO num_available_seats FROM Flight WHERE Flight.flightId = flight_id AND Flight.departureDate =departure_date ;

RETURN num_available_seats;

END :

SELECT get_num_available_economy_seats ('AC123','2023-05-01');

5. Function to calculate the total salary of all ground staff members in a given department:

DELIMITER //

CREATE FUNCTION calculate_total_salary(department_name VARCHAR(25))

RETURNS INT DETERMINISTIC

BEGIN

DECLARE total_salary INT;

SELECT SUM(salary) INTO total_salary FROM GroundStaff WHERE GroundStaff.department = department_name;

RETURN total_salary;

END //

DELIMITER ;

SELECT calculate_total_salary('Catering');

SQL PROCEDURES

A SQL procedure is a group of SQL statements and logic, compiled and stored together to perform a specific task.

1.Write a procedure that takes in an 'airlineNo' and returns the total number of staff members working for the airline, broken down by their job position (Cabin Crew, Pilot, Ground Staff) and sorted by the total salary for each position in descending order.

Solution-

Delimiter :

CREATE PROCEDURE staff_summary(IN airline_code VARCHAR(25))

BEGIN SELECT 'Cabin Crew' AS Job_Position, COUNT(*) AS Total_Count, SUM(salary) AS Total_Salary FROM CabinCrew

WHERE airlineNo = airline_code UNION

SELECT 'Pilot' AS Job_Position, COUNT(*) AS Total_Count, SUM(salary) AS Total_Salary FROM Pilot

WHERE airlineNo = airline_code UNION

SELECT 'Ground Staff' AS Job_Position, COUNT(*) AS Total_Count, SUM(salary) AS Total_Salary FROM GroundStaff

WHERE airlineNo = airline_code

ORDER BY Total_Salary DESC;

END :

This procedure helps Us find out how many cabin crew, pilots, and ground staff work for a specific airline and how much they're paid. We just need to input

the airline code to get the results. The results are sorted based on the highest total salary for each job position.

2.Write a procedure to find the top 5 routes with the highest number of sold tickets, along with the total revenue generated from those tickets.

A route is defined by a combination of departure and destination airports.

Solution-

delimiter :

CREATE PROCEDURE top_routes()

BEGIN SELECT CONCAT(f1.departure, ' - ', f1.destination) AS route,COUNT(*) AS total_tickets_sold,SUM(td.ticketFare) AS total_revenue

FROM BookedFlight bf

INNER JOIN TicketDetails td ON bf.ticketId = td.ticketId AND bf.ticketClass = td.ticketClass

INNER JOIN TicketAllocation ta ON bf.ticketId = ta.ticketId

INNER JOIN Flight f1 ON ta.flightId = f1.flightId AND ta.airlineNo = f1.airlineNo

GROUP BY f1.departure, f1.destination

ORDER BY total_tickets_sold DESC;

END :

This procedure joins the BookedFlight, TicketDetails, TicketAllocation, and Flight tables to obtain the required information.

It filters the results based on the booking date and groups the results by the route.

Finally, it sorts the routes by the total number of tickets sold and returns the top 5 routes along with the total revenue generated from those tickets.

This information can be useful for airlines to identify their most popular routes and optimize their pricing and capacity planning strategies accordingly.

3.Write a procedure To find ticket details of a particular passenger, assuming that the passenger is identified by their passengerNo:

CREATE PROCEDURE GetTicketDetailsByPassengerNo(IN p_passengerNo VARCHAR(25))

BEGIN

SELECT td.ticketId, td.ticketClass, td.ticketStatus, td.ticketFare, td.bookingDate

FROM TicketDetails td

INNER JOIN TicketAllocation ta ON td.ticketId = ta.ticketId

WHERE ta.passengerId = p_passengerNo;

END

This procedure takes a passenger number as input and returns the ticket details (ticket ID, ticket class, ticket status, ticket fare, and booking date)

for that particular passenger from the TicketDetails and TicketAllocation tables using an inner join.

4.Write a procedure that returns a list of all passengers who have checked in luggage on a particular airline's flights.

The procedure should display each passenger's name and the total weight of their checked-in luggage.

Solution-

delimiter :

CREATE PROCEDURE getPassengersWithLuggage(IN p_airlineNo VARCHAR(25))

BEGIN

SELECT p.firstName, p.lastName, SUM(b.weight) AS total_luggage_weight

FROM Passengers p

INNER JOIN Reservation r ON p.passengerNo = r.passengerNo

INNER JOIN LuggageInfo l ON r.passengerNo = l.passengerNo AND r.airlineNo = l.airlineNo

INNER JOIN Baggage b ON l.baggageId = b.baggageId

WHERE r.airlineNo = p_airlineNo

GROUP BY p.passengerNo

ORDER BY total_luggage_weight DESC;

END :

The procedure takes an input parameter AirlineNo. It then selects the details of the luggage for each passenger who has checked in luggage on a flight operated by that airline,

using the INNER JOIN statement to join the Passengers, Reservation, LuggageInfo, and Baggage tables.

The SUM function is used to calculate the total weight of the luggage for each passenger, and the GROUP BY statement is used to group the results by passenger. The results are ordered by the total weight of the luggage in descending order.

5.Write a procedure that takes an airline number as input and returns the details of all flights operated by that airline,

along with the count of passengers booked on each flight.

Solution:

delimiter :

CREATE PROCEDURE GetFlightDetailsByAirlineNo(IN p_airlineNo VARCHAR(25))

BEGIN

SELECT f.flightId, f.departure, f.destination, COUNT(ta.passengerId) AS passengerCount

FROM Flight f

LEFT JOIN TicketAllocation ta ON f.flightId = ta.flightId AND f.airlineNo = ta.airlineNo

WHERE f.airlineNo = p_airlineNo

GROUP BY f.flightId, f.departure, f.destination;

END :

This procedure displays the flight number, departure and destination airports, departure and arrival times,

and the number of passengers booked on each flight.

SQL VIEWS

In SQL, a view is a virtual table that is based on the result set of a SELECT statement. It is essentially a named and saved query that can be accessed like a regular table, but does not store any data itself.

1.Create a view to display passenger name, airline name, departure and destination cities for all reservations :

```
CREATE VIEW ReservationView AS

SELECT Passengers.firstName, Passengers.lastName, Airlines.airlineName,
Flight.departure, Flight.destination

FROM Reservation

JOIN Passengers ON Reservation.passengerNo = Passengers.passengerNo

JOIN Airlines ON Reservation.airlineNo = Airlines.airlineNo

JOIN Flight ON Reservation.airlineNo = Flight.airlineNo;
```

2. Create a view to display the total weight of baggage checked in by passengers for each flight :

```
CREATE VIEW FlightBaggageView AS

SELECT Flight.flightId, SUM(Baggage.weight) AS totalWeight

FROM Flight
```

JOIN FlightBaggage ON Flight.flightId = FlightBaggage.flightId AND Flight.departureDate = FlightBaggage.departureDate AND Flight.arrivalDate = FlightBaggage.arrivalDate

JOIN Baggage ON FlightBaggage.baggageId = Baggage.baggageId

GROUP BY Flight.flightId;

3. Create a view to displaY the list of flights and their current availability of business and economy seats, along with the total number of booked seats for each class.

CREATE VIEW AvailableSeats AS

SELECT f.flightId, f.departureDate, f.arrivalDate,

f.businessSeatsAvailability - COUNT(CASE WHEN bf.ticketClass = 'Business' THEN bf.seatNo END) AS availableBusinessSeats,

f.economySeatsAvailability - COUNT(CASE WHEN bf.ticketClass = 'Economy' THEN bf.seatNo END) AS availableEconomySeats

FROM Flight f

LEFT JOIN BookedFlight bf ON f.flightId = bf.flightNo

GROUP BY f.flightId, f.departureDate, f.arrivalDate;

4. View to display the list of passengers with their reservation and ticket details :

CREATE VIEW passenger_details AS

SELECT p.passengerNo, p.firstName, p.lastName, r.airlineNo, r.reservationId, ta.ticketId, td.ticketClass, td.ticketStatus, td.ticketFare

FROM passengers p

JOIN reservation r ON p.passengerNo = r.passengerNo

JOIN ticketallocation ta ON r.passengerNo = ta.passengerId AND r.airlineNo = ta.airlineNo

JOIN ticketdetails td ON ta.ticketId = td.ticketId;

5. View to display flight information along with airline name :

CREATE VIEW flight_info AS

SELECT f.flightId, f.businessSeatsAvailability, f.economySeatsAvailability, f.departure, f.destination, f.departureDate, f.arrivalDate, a.airlineName FROM Flight f

JOIN Airlines a ON f.airlineNo = a.airlineNo;

SQL TRIGGERS

Triggers in SQL are a type of stored procedure that are automatically executed in response to certain database events such as insert, update, or delete operations on a table.

1.Write a trigger that will automatically update the status of a baggage to "Checked" in the Baggage table, when the baggage is added to a flight.

The trigger should fire after a new row is inserted into the LuggageInfo table:

DELIMITER :

CREATE TRIGGER baggage_loaded_trigger AFTER INSERT ON LuggageInfo

FOR EACH ROW

BEGIN

```
UPDATE Baggage SET status = 'Checked' WHERE baggageId = NEW.baggageId;

END :
```

2. Trigger to delete corresponding rows from Reservations table when a passenger is deleted from Passengers table:

```
CREATE TRIGGER delete_passenger_reservations

AFTER DELETE ON Passengers

FOR EACH ROW

BEGIN

  DELETE FROM Reservations WHERE passengerNo = OLD.passengerNo;

END;
```

3.Trigger that will increment the salary of all the pilots by 5% of flight when new pilots are added to the Pilot table:

```
Delimiter :

CREATE TRIGGER increment_pilot_salary

AFTER INSERT ON Pilot

FOR EACH ROW

BEGIN

   DECLARE increment_amount DECIMAL(10, 2);

   SELECT 0.05 * AVG(flightFare) INTO increment_amount FROM Flight;
```

```
   UPDATE Pilot SET salary = salary + increment_amount WHERE pilotId =
NEW.pilotId;

END :
```

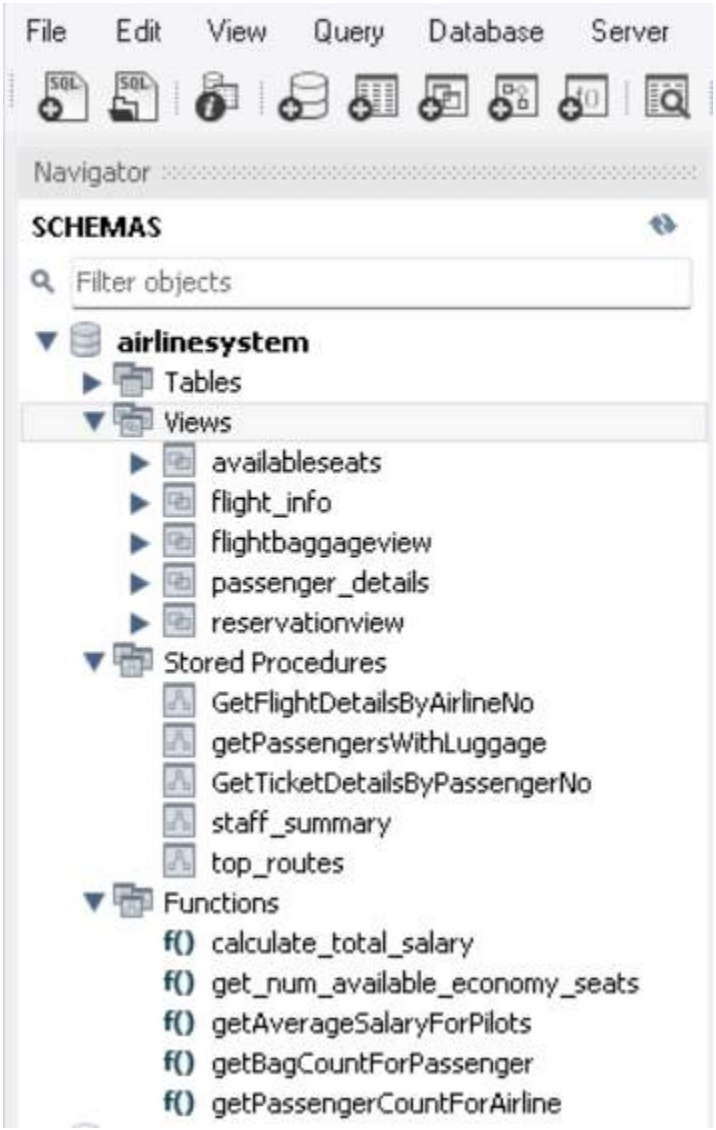
4. Trigger will fire before a new row is inserted into the Passengers table and will verify that the passport number is unique.

```
CREATE TRIGGER unique_passport_no

BEFORE INSERT ON Passengers

FOR EACH ROW

BEGIN

 IF EXISTS (SELECT 1 FROM Passengers WHERE passportNo = NEW.passportNo)
THEN

   SIGNAL SQLSTATE '45000'

    SET MESSAGE_TEXT = 'Passport number must be unique';

 END IF;

END
```

5. Write a trigger that automatically adds a new baggage record to the Baggage table when a passenger checks in a luggage.

```
DELIMITER //

CREATE TRIGGER add_baggage AFTER INSERT ON LuggageInfo FOR EACH ROW
BEGIN  INSERT INTO Baggage (baggageId, status, weight)VALUES (NEW.baggageId,
'Checked-In', 0);
END //
```

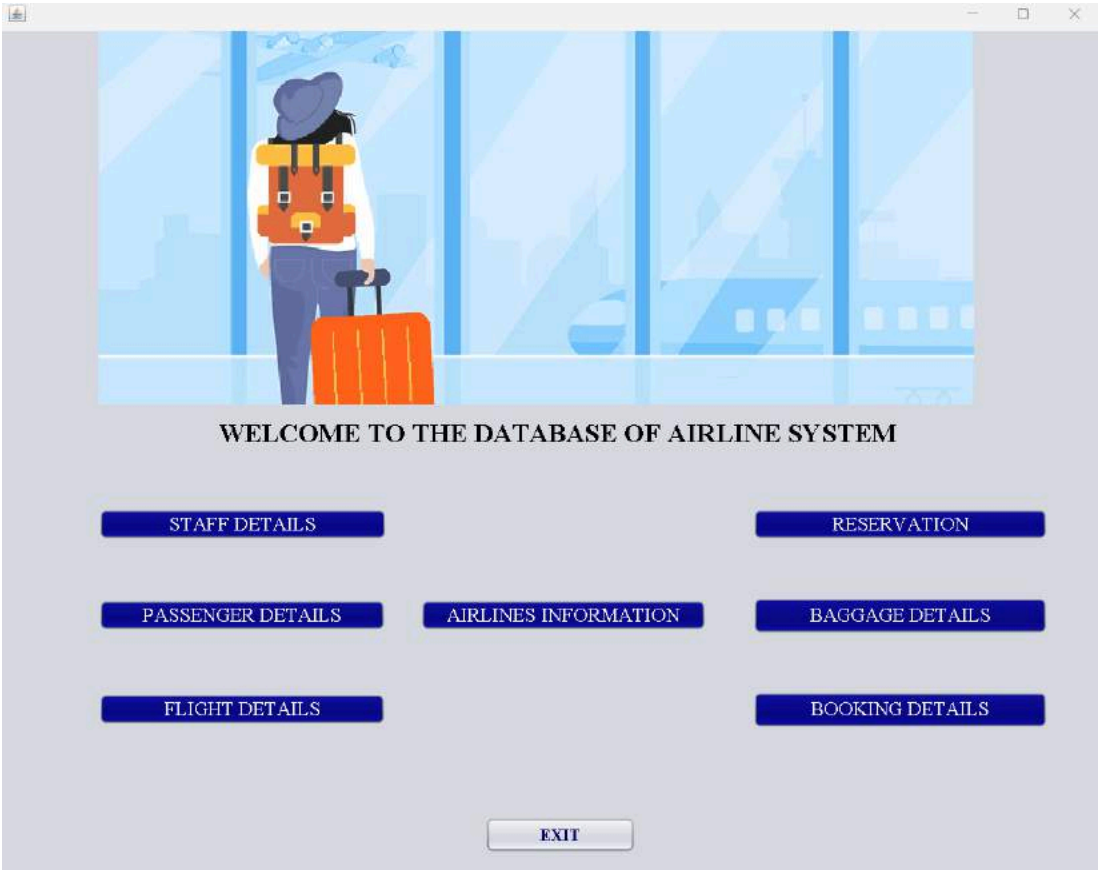
ALL THE CORRESPONDING VIEWS, PROCEDURES , FUNCTIONS AND TRIGGERS HAS BEEN IMPLEMENTED INTO THE DATABASE.



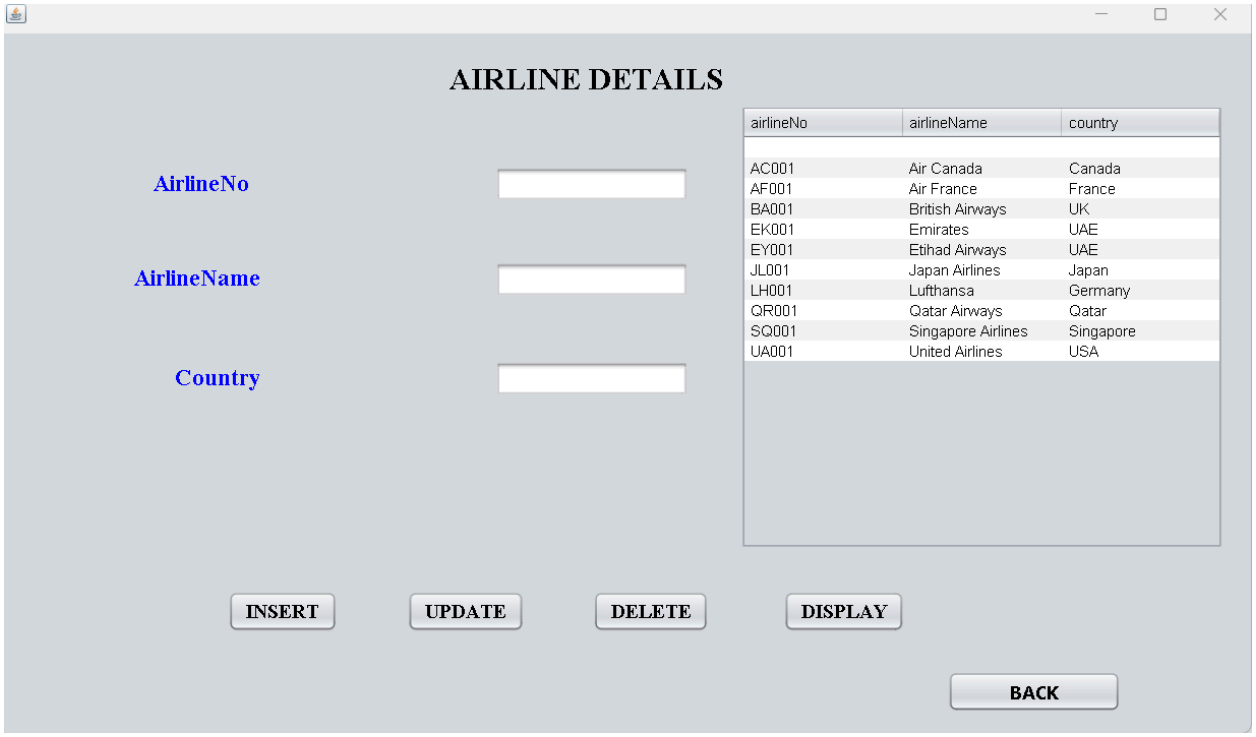
FRONT END DESIGN

Login details are from the Administrator Table.

Welcome page-Each Buttons Contain the related Tables.



Demonstrating Insert ,Update, Delete and Display Buttons for Airline Table

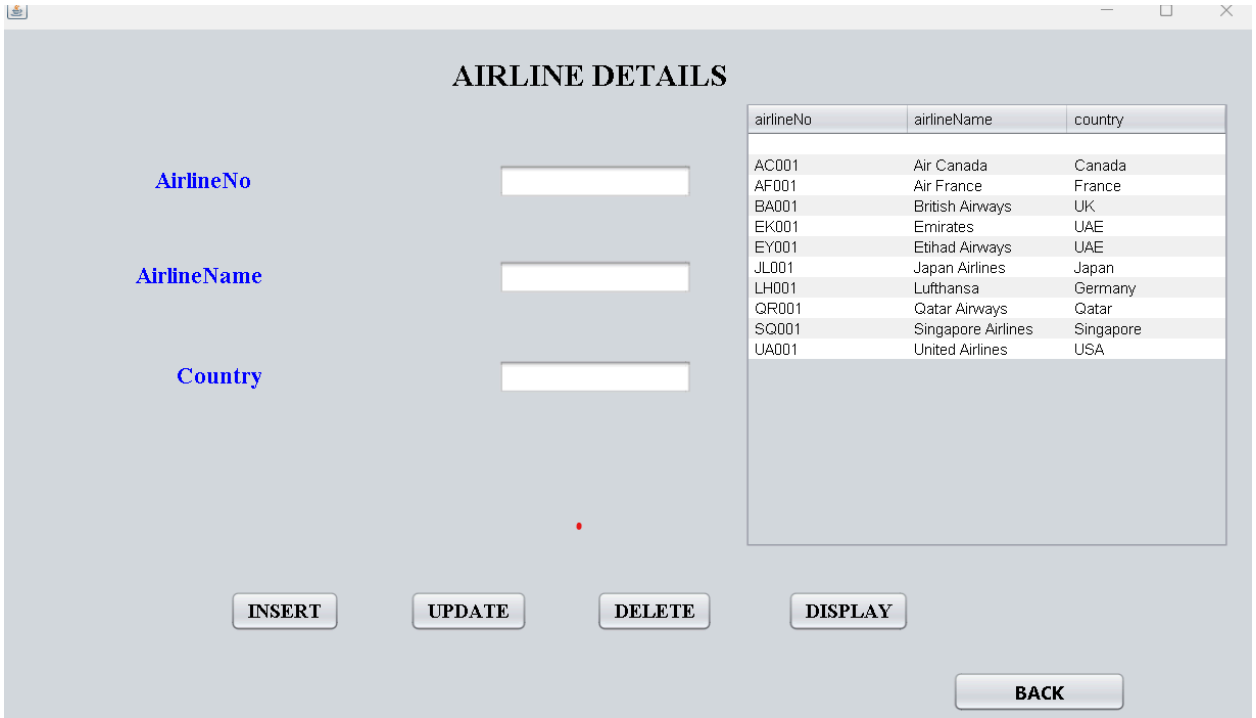



AIRLINE DETAILS
AirlineNo
JT001
AirlineName
Jet Airways
Country
India
Message
Data Inserted
OK
INSERT
DISPLAY
BACK
airlineNo
airlineName
country
AC001 Air Canada Canada
AF001 Air France France
BA001 British Airways UK
EK001 Emirates UAE
EY001 Etihad Airways UAE
JL001 Japan Airlines Japan
LH001 Lufthansa Germany
QR001 Qatar Airways Qatar
SQ001 Singapore Airlines Singapore
UA001 United Airlines USA


AIRLINE DETAILS
AirlineNo
JT001
AirlineName
Jet Airlines
Country
India
Message
Data Updated
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK
airlineNo
airlineName
country
AC001 Air Canada Canada
AF001 Air France France
BA001 British Airways UK
EK001 Emirates UAE
EY001 Etihad Airways UAE
JL001 Japan Airlines Japan
JT001 Jet Airways India
LH001 Lufthansa Germany
QR001 Qatar Airways Qatar
SQ001 Singapore Airlines Singapore
UA001 United Airlines USA

airlineNo	airlineName	country
AC001	Air Canada	Canada
AF001	Air France	France
BA001	British Airways	UK
EK001	Emirates	UAE
EY001	Etihad Airways	UAE
JL001	Japan Airlines	Japan
JT001	Jet Airlines	India
LH001	Lufthansa	Germany
QR001	Qatar Airways	Qatar
SQ001	Singapore Airlines	Singapore
UA001	United Airlines	USA


AIRLINE DETAILS
AirlineNo
JT001
AirlineName
Jet Airlines
Country
India
airlineNo
airlineName
country
AC001 Air Canada Canada
AF001 Air France France
BA001 British Airways UK
EK001 Emirates UAE
EY001 Etihad Airways UAE
JL001 Japan Airlines Japan
JT001 Jet Airlines India
LH001 Lufthansa Germany
QR001 Qatar Airways Qatar
SQ001 Singapore Airlines Singapore
UA001 United Airlines USA
Message
Data Deleted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

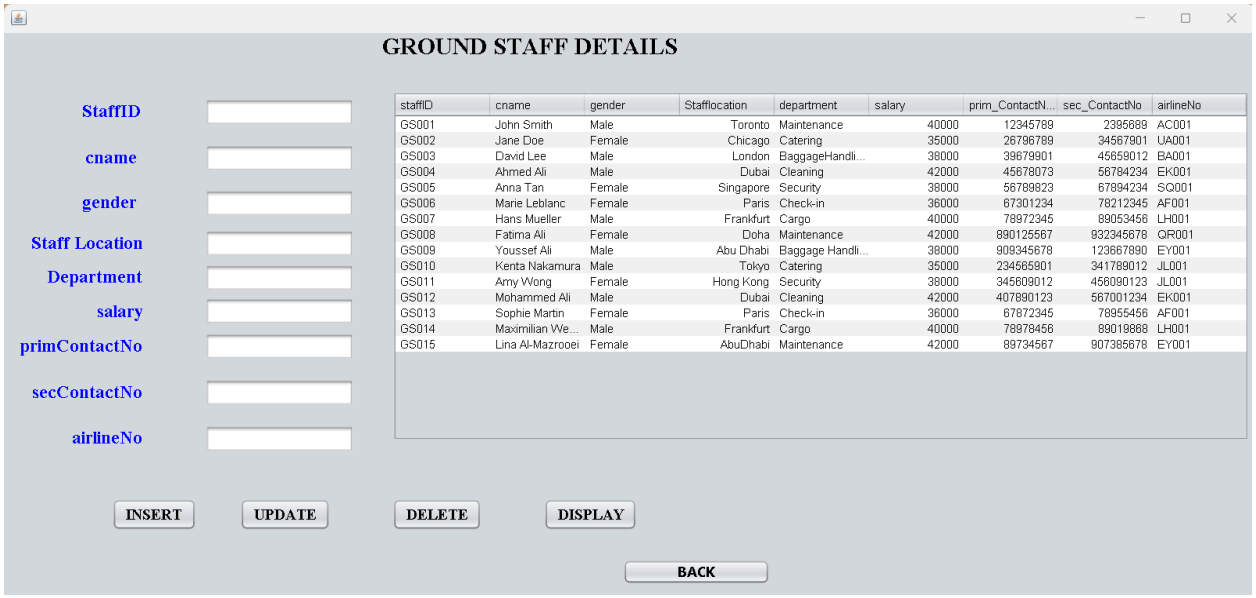

AIRLINE DETAILS
AirlineNo
AirlineName
Country
airlineNo
airlineName
country
AC001 Air Canada Canada
AF001 Air France France
BA001 British Airways UK
EK001 Emirates UAE
EY001 Etihad Airways UAE
JL001 Japan Airlines Japan
LH001 Lufthansa Germany
QR001 Qatar Airways Qatar
SQ001 Singapore Airlines Singapore
UA001 United Airlines USA
INSERT
UPDATE
DELETE
DISPLAY
BACK

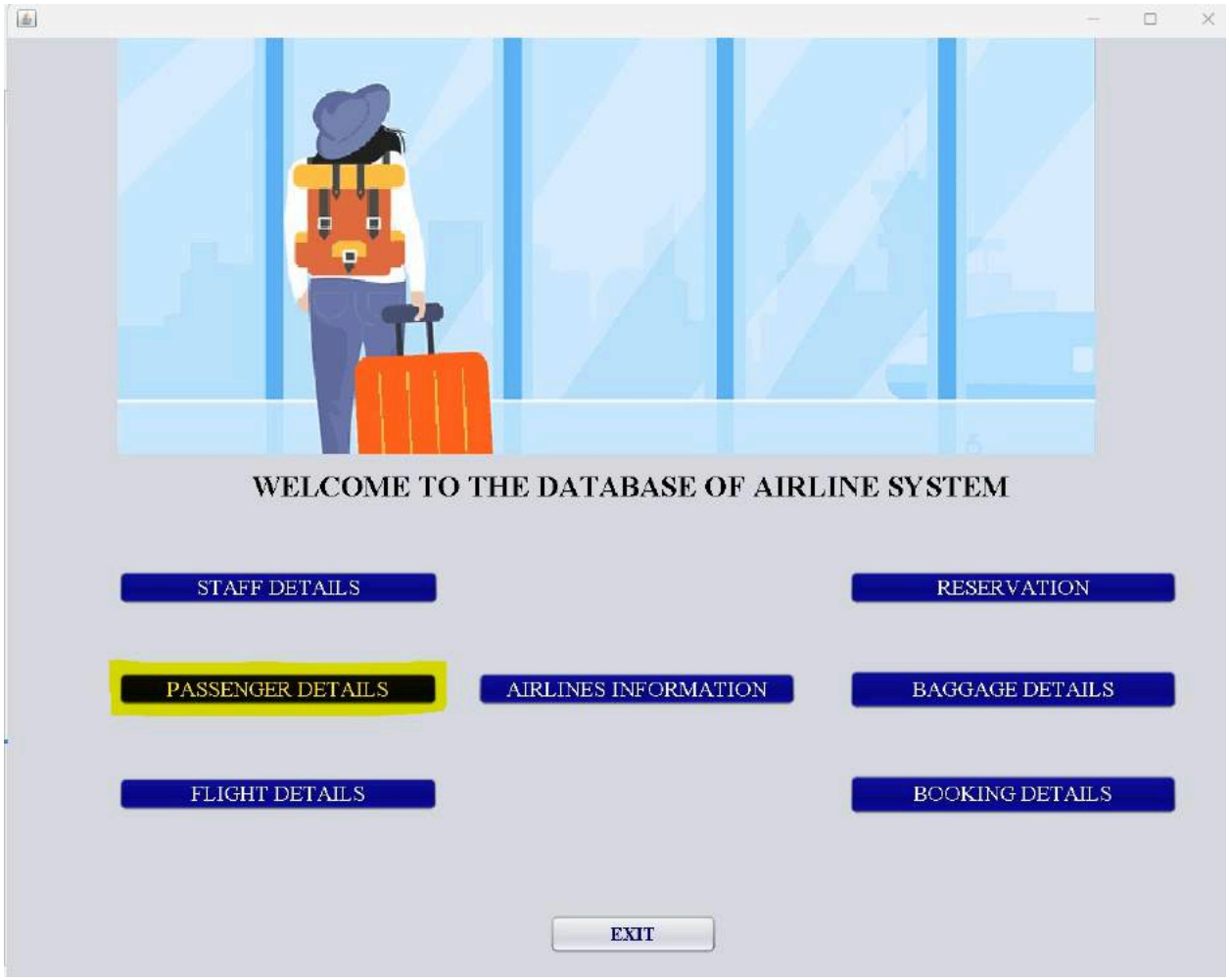
Under StaffDetails We have 3 Tables

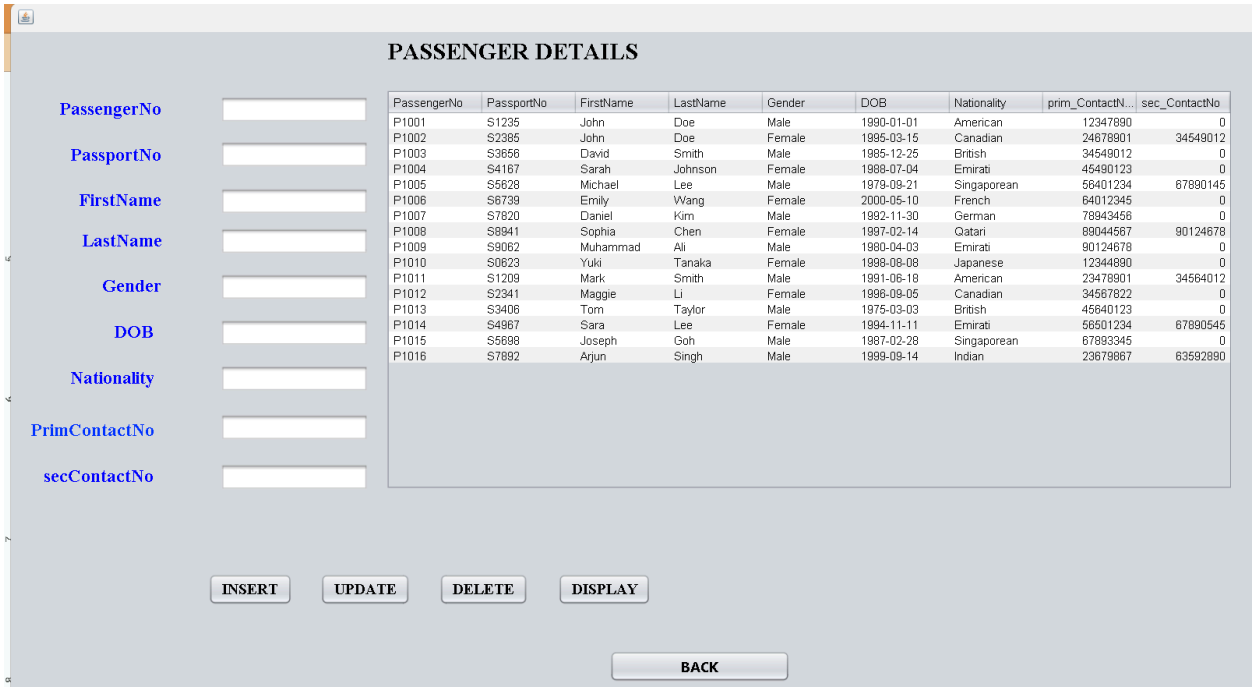
Demonstrating All Buttons for Cabin Crew Table

cabinID	cname	gender	hours flown	position	salary	prim_ContactN...	sec_ContactNo	airlineNo
CC001	John Smith	Male	500	FlightAttendant	3000	12387890	98765710	AC001
CC002	Jane Doe	Female	800	Purser	5000	23578901	87652109	UA001
CC003	David Williams	Male	600	SeniorFlight Att...	4000	34539012	76543398	BA001
CC004	Fatima Ali	Female	1000	CabinManager	6000	45790123	65432987	EK001
CC005	Sarah Lee	Female	700	FlightAttendant	3500	56771234	54379876	SQ001
CC006	Pierre Lefebvre	Male	900	SeniorPurser	5500	67890745	43210975	AF001
CC007	Hans Mueller	Male	1200	CabinManager	7000	78973456	32787654	LH001
CC008	Fatima Ahmed	Female	1500	CabinManager	8000	89074567	21098643	QR001
CC009	Sara Khalid	Female	1100	SeniorFlight Att...	5000	90123468	10987662	EY001
CC010	Kenji Nakamura	Male	1000	Purser	4500	12346890	98763210	JL001
CC011	Mary Johnson	Female	700	FlightAttendant	3500	23456901	87662109	AC001
CC012	Richard Brown	Male	800	SeniorFlight Att...	4000	34566012	76546098	UA001
CC013	Sophie Martin	Female	900	Purser	5500	45678623	65460987	EK001
CC014	Paulo Silva	Male	600	FlightAttendant	3000	56786234	54326876	SQ001
CC015	Maria Hernand...	Female	1100	Cabin Manager	5000	67862345	43268765	BA001
CC016	Janet Johnson	Female	550	Purser	9000	45743275	77432873	LH001

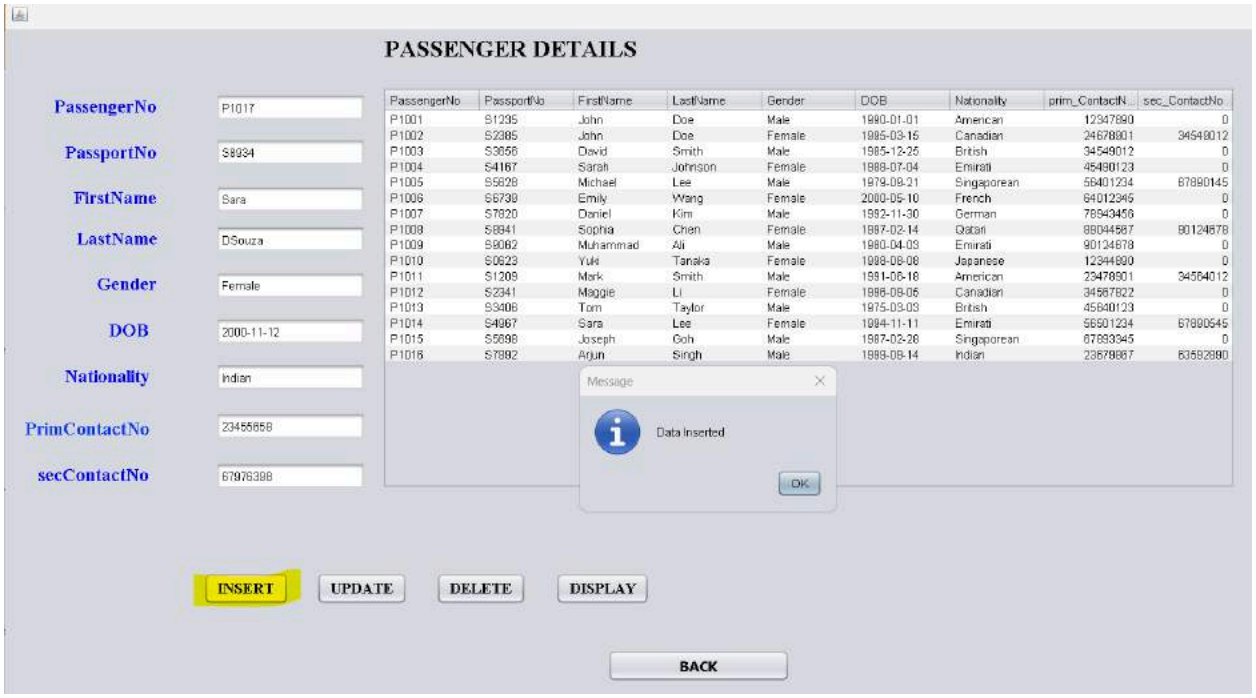
Ground Staff and Pilot table also have Insert , Update , Delete and Display with same Functionality as Cabin Crew







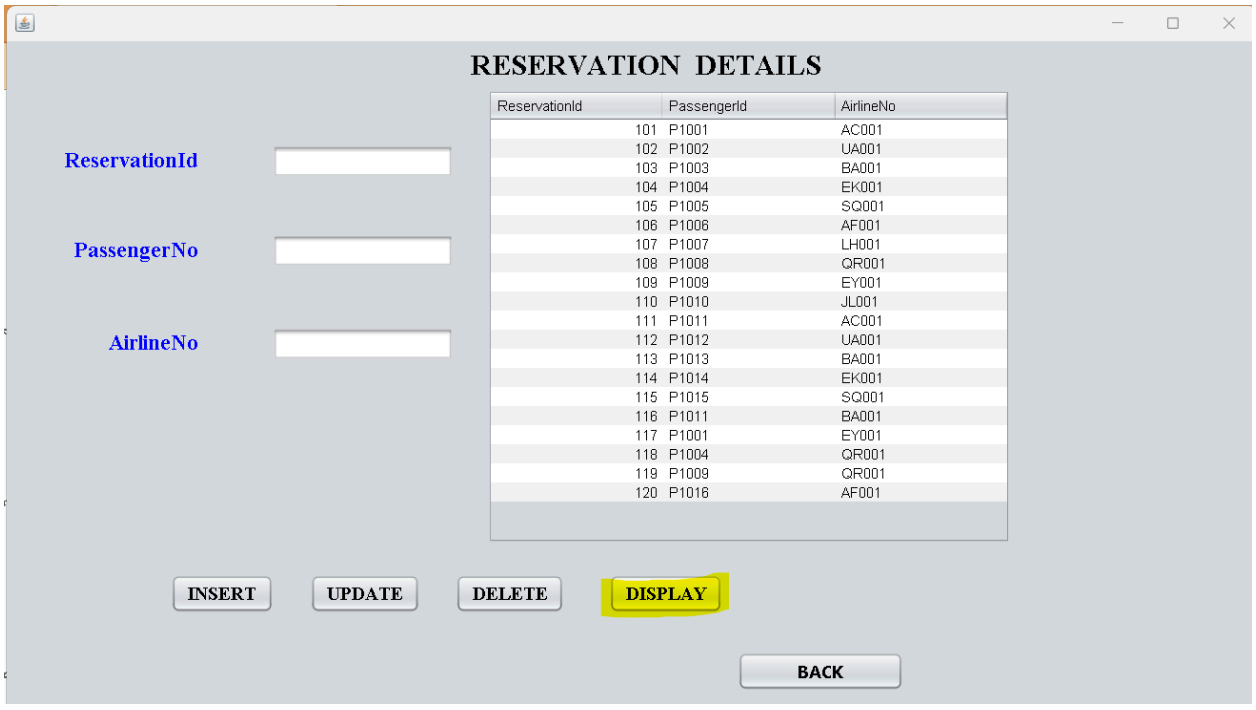
PassengerNo	PassportNo	FirstName	LastName	Gender	DOB	Nationality	prim_ContactN...	sec_ContactNo
P1001	S1235	John	Doe	Male	1990-01-01	American	12347890	0
P1002	S2385	John	Doe	Female	1995-03-15	Canadian	24678901	34549012
P1003	S3656	David	Smith	Male	1985-12-25	British	34549012	0
P1004	S4167	Sarah	Johnson	Female	1988-07-04	Emirati	45490123	0
P1005	S5628	Michael	Lee	Male	1979-09-21	Singaporean	56401234	67890145
P1006	S6739	Emily	Wang	Female	2000-05-10	French	64012345	0
P1007	S7820	Daniel	Kim	Male	1992-11-30	German	78943456	0
P1008	S8941	Sophia	Chen	Female	1997-02-14	Qatari	89044567	90124678
P1009	S9062	Muhammad	Ali	Male	1980-04-03	Emirati	90124678	0
P1010	S0623	Yuki	Tanaka	Female	1998-08-08	Japanese	12344890	0
P1011	S1209	Mark	Smith	Male	1991-06-18	American	23478901	34564012
P1012	S2341	Maggie	Li	Female	1998-09-05	Canadian	34567822	0
P1013	S3406	Tom	Taylor	Male	1975-03-03	British	45640123	0
P1014	S4967	Sara	Lee	Female	1994-11-11	Emirati	56501234	67890545
P1015	S5698	Joseph	Goh	Male	1987-02-28	Singaporean	67893345	0
P1016	S7892	Arjun	Singh	Male	1999-09-14	Indian	23679867	63592890


PASSENGER DETAILS
PassengerNo
P1017
PassportNo
S8934
FirstName
Sara
LastName
DSouza
Gender
Female
DOB
2000-11-12
Nationality
Indian
PrimContactNo
23455658
secContactNo
67976398
PassengerNo PassportNo FirstName LastName Gender DOB Nationality prim_ContactN... sec_ContactNo
P1001 S1235 John Doe Male 1990-01-01 American 12347890 0
P1002 S2385 John Doe Female 1995-03-15 Canadian 24678901 34549012
P1003 S3656 David Smith Male 1985-12-25 British 34549012 0
P1004 S4167 Sarah Johnson Female 1988-07-04 Emirati 45490123 0
P1005 S5628 Michael Lee Male 1979-09-21 Singaporean 56401234 67890145
P1006 S6739 Emily Wang Female 2000-05-10 French 64012345 0
P1007 S7820 Daniel Kim Male 1992-11-30 German 78943456 0
P1008 S8941 Sophia Chen Female 1997-02-14 Qatari 89044567 90124678
P1009 S9062 Muhammad Ali Male 1980-04-03 Emirati 90124678 0
P1010 S0623 Yuki Tanaka Female 1998-08-08 Japanese 12344890 0
P1011 S1209 Mark Smith Male 1991-06-18 American 23478901 34564012
P1012 S2341 Maggie Li Female 1996-09-05 Canadian 34567822 0
P1013 S3406 Tom Taylor Male 1975-03-03 British 45640123 0
P1014 S4967 Sara Lee Female 1994-11-11 Emirati 56501234 67890545
P1015 S5698 Joseph Goh Male 1987-02-28 Singaporean 67893345 0
P1016 S7892 Arjun Singh Male 1999-09-14 Indian 23679867 63592890
Message
Data Inserted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK


PASSENGER DETAILS
PassengerNo PassportNo FirstName LastName Gender DOB Nationality prim_ContactN... sec_ContactNo
P1001 S1235 John Doe Male 1990-01-01 American 12347890 0
P1002 S2385 John Doe Female 1995-03-15 Canadian 24678901 34549012
P1003 S3656 David Smith Male 1985-12-25 British 34549012 0
P1004 S4167 Sarah Johnson Female 1988-07-04 Emirati 45490123 0
P1005 S5628 Michael Lee Male 1979-09-21 Singaporean 56401234 67890145
P1006 S6739 Emily Wang Female 2000-05-10 French 64012345 0
P1007 S7820 Daniel Kim Male 1992-11-30 German 78943456 0
P1008 S8941 Sophia Chen Female 1997-02-14 Qatari 89044567 90124678
P1009 S9062 Muhammad Ali Male 1980-04-03 Emirati 90124678 0
P1010 S0623 Yuki Tanaka Female 1998-08-08 Japanese 12344890 0
P1011 S1209 Mark Smith Male 1991-06-18 American 23478901 34564012
P1012 S2341 Maggie Li Female 1996-09-05 Canadian 34567822 0
P1013 S3406 Tom Taylor Male 1975-03-03 British 45640123 0
P1014 S4967 Sara Lee Female 1994-11-11 Emirati 56501234 67890545
P1015 S5698 Joseph Goh Male 1987-02-28 Singaporean 67893345 0
P1016 S7892 Arjun Singh Male 1999-09-14 Indian 23679867 63592890
P1017 S8934 Sara DSouza Female 2000-11-12 Indian 23455658 67976398
DELETE
DISPLAY

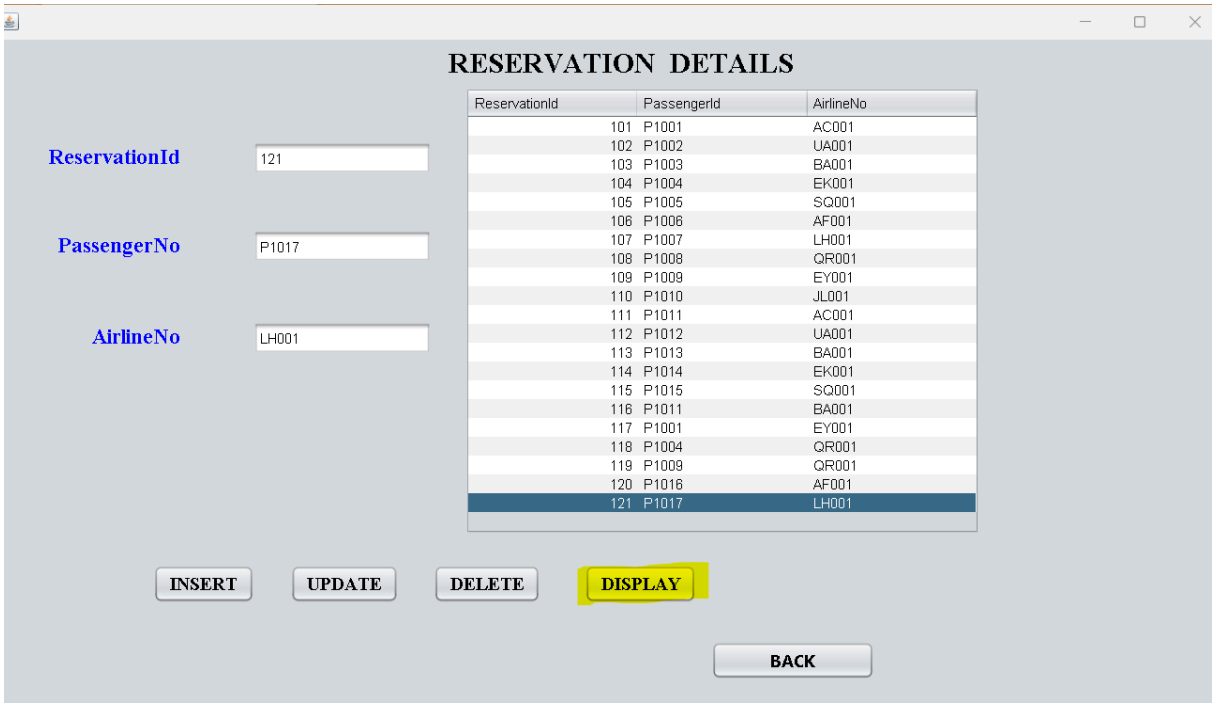

PASSENGER DETAILS
PassengerNo
P1017
PassportNo
S8934
FirstName
Sara
LastName
DSouza
Gender
Female
DOB
2001-10-12
Nationality
Indian
PrimContactNo
23455658
secContactNo
67976398
PassengerNo PassportNo FirstName LastName Gender DOB Nationality prim_ContactN.. sec_ContactNo
P1001 S1235 John Doe Male 1990-01-01 American 12347890 0
P1002 S2365 John Doe Female 1995-03-15 Canadian 24678901 34549012
P1003 S3656 David Smith Male 1985-12-25 British 34549012 0
P1004 S4187 Sarah Johnson Female 1988-07-04 Emirati 45490123 0
P1005 S5628 Michael Lee Male 1979-09-21 Singaporean 56401234 67890145
P1006 S6739 Emily Wang Female 2000-05-10 French 64012345 0
P1007 S7820 Daniel Kim Male 1992-11-30 German 78943456 0
P1008 S8941 Sophia Chen Female 1997-02-14 Qatari 89044567 90124678
P1009 S9062 Muhammad Ali Male 1980-04-03 Emirati 90124678 0
P1010 S0623 Yuki Tanaka Female 1998-08-08 Japanese 12344890 0
P1011 S1209 Mark Smith Male 1991-06-18 American 23478901 34564012
P1012 S2341 Maggie Li Female 1996-09-05 Canadian 34567822 0
P1013 S3406 Tom Taylor Male 1975-03-03 British 45640123 0
P1014 S4987 Sara Lee Female 1994-11-11 Emirati 56501234 67890545
P1015 S5698 Joseph Smith Male 1987-02-28 Singaporean 67893345 0
P1016 S7892 Arjun Patel Male 1998-09-14 Indian 23679867 63592890
P1017 S8934 Sara DSouza Female 2001-10-12 Indian 23455658 67976398
Message
Data Updated
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

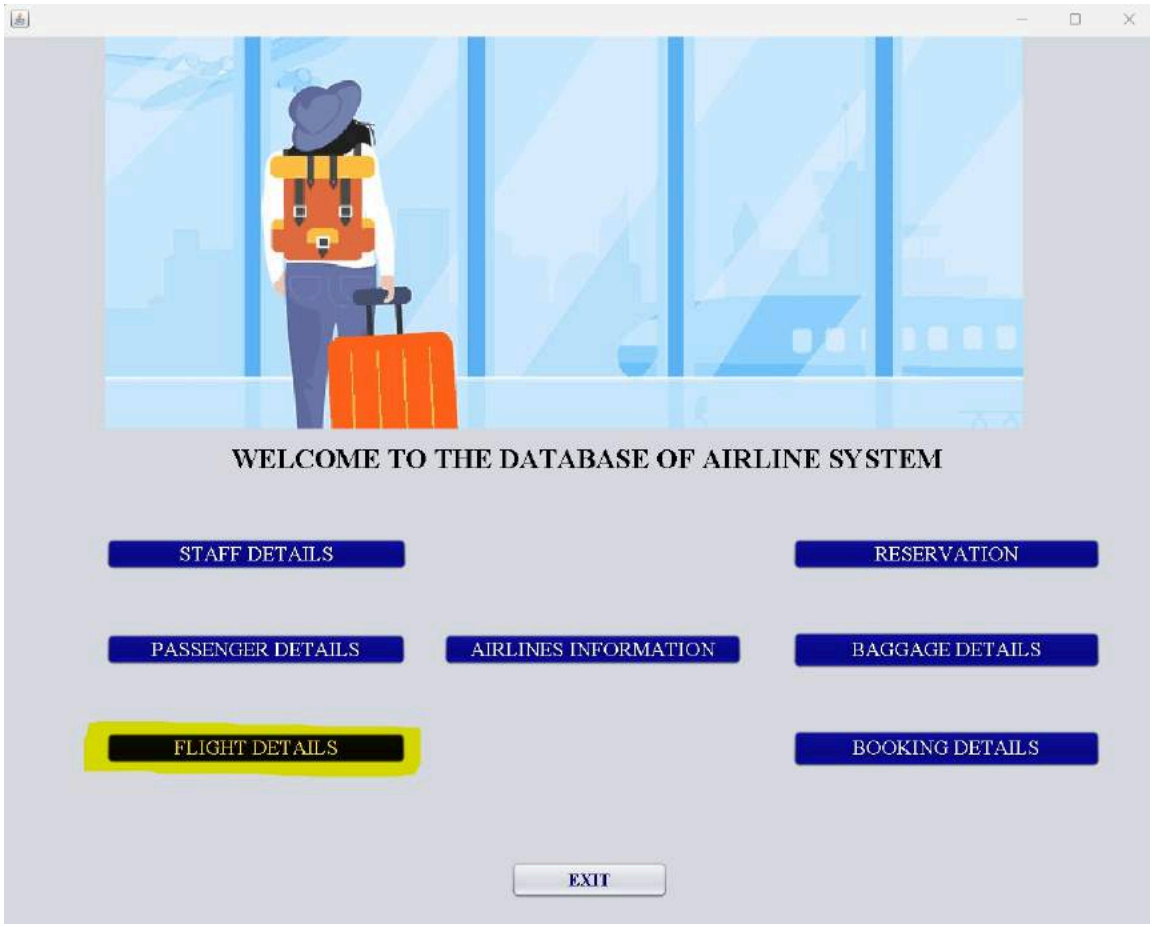

PASSENGER DETAILS
PassengerNo
P1017
PassportNo
S8934
FirstName
Sara
LastName
DSouza
Gender
Female
DOB
2001-10-12
Nationality
Indian
PrimContactNo
23455658
secContactNo
67976398
PassengerNo PassportNo FirstName LastName Gender DOB Nationality prim_ContactN.. sec_ContactNo
P1001 S1235 John Doe Male 1990-01-01 American 12347890 0
P1002 S2365 John Doe Female 1995-03-15 Canadian 24678901 34549012
P1003 S3656 David Smith Male 1985-12-25 British 34549012 0
P1004 S4187 Sarah Johnson Female 1988-07-04 Emirati 45490123 0
P1005 S5628 Michael Lee Male 1979-09-21 Singaporean 56401234 67890145
P1006 S6739 Emily Wang Female 2000-05-10 French 64012345 0
P1007 S7820 Daniel Kim Male 1992-11-30 German 78943456 0
P1008 S8941 Sophia Chen Female 1997-02-14 Qatari 89044567 90124678
P1009 S9062 Muhammad Ali Male 1980-04-03 Emirati 90124678 0
P1010 S0623 Yuki Tanaka Female 1998-08-08 Japanese 12344890 0
P1011 S1209 Mark Smith Male 1991-06-18 American 23478901 34564012
P1012 S2341 Maggie Li Female 1996-09-05 Canadian 34567822 0
P1013 S3406 Tom Taylor Male 1975-03-03 British 45640123 0
P1014 S4987 Sara Lee Female 1994-11-11 Emirati 56501234 67890545
P1015 S5698 Joseph Smith Male 1987-02-28 Singaporean 67893345 0
P1016 S7892 Arjun Patel Male 1998-09-14 Indian 23679867 63592890
P1017 S8934 Sara DSouza Female 2001-10-12 Indian 23455658 67976398
Message
Data Deleted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

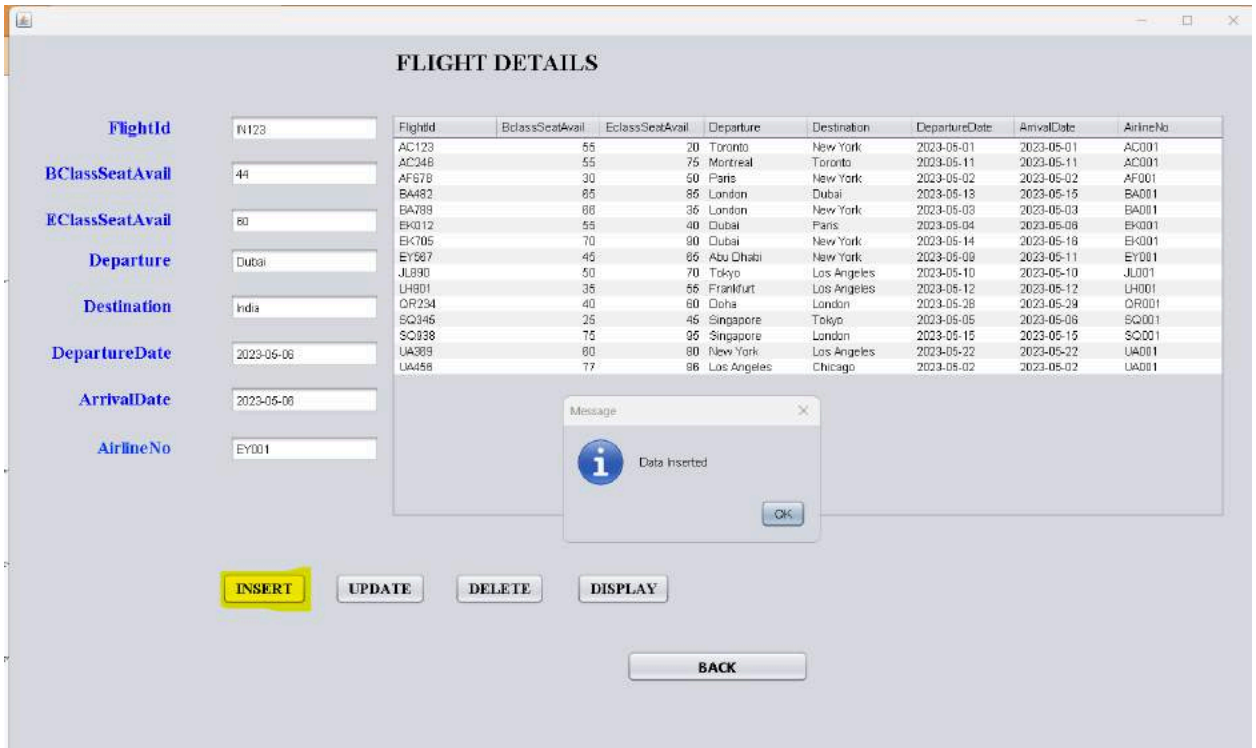


ReservationId	PassengerId	AirlineNo
101	P1001	AC001
102	P1002	UA001
103	P1003	BA001
104	P1004	EK001
105	P1005	SQ001
106	P1006	AF001
107	P1007	LH001
108	P1008	QR001
109	P1009	EY001
110	P1010	JL001
111	P1011	AC001
112	P1012	UA001
113	P1013	BA001
114	P1014	EK001
115	P1015	SQ001
116	P1011	BA001
117	P1001	EY001
118	P1004	QR001
119	P1009	QR001
120	P1016	AF001


RESERVATION DETAILS
ReservationId
121
PassengerNo
P1017
AirlineNo
BA001
ReservationId PassengerId AirlineNo
101 P1001 AC001
102 P1002 UA001
103 P1003 BA001
104 P1004 EK001
105 P1005 SQ001
106 P1006 AF001
107 P1007 LH001
108 P1008 QR001
109 P1009 EY001
110 P1010 JL001
111 P1011 AC001
112 P1012 UA001
113 P1013 BA001
114 P1014 EK001
115 P1015
116 P1011
117 P1001
118 P1004
119 P1009
120 P1018
Message
Data Inserted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK


RESERVATION DETAILS
ReservationId
121
PassengerNo
P1017
AirlineNo
LH001
ReservationId PassengerId AirlineNo
101 P1001 AC001
102 P1002 UA001
103 P1003 BA001
104 P1004 EK001
105 P1005 SQ001
106 P1006 AF001
107 P1007 LH001
108 P1008 QR001
109 P1009 EY001
110 P1010 JL001
111 P1011 AC001
112 P1012 UA001
113 P1013 BA001
114 P1014 EK001
115 P1015 SQ001
116 P1011 BA001
117 P1001 EY001
118 P1004 QR001
119 P1009 QR001
120 P1016 AF001
121 P1017 LH001
INSERT
UPDATE
DELETE
DISPLAY
BACK


WELCOME TO THE DATABASE OF AIRLINE SYSTEM
STAFF DETAILS
RESERVATION
PASSENGER DETAILS
AIRLINES INFORMATION
BAGGAGE DETAILS
FLIGHT DETAILS
BOOKING DETAILS
EXIT


FLIGHT DETAILS
FlightId
IN123
BClassSeatAvail
44
EClassSeatAvail
60
Departure
Dubai
Destination
India
DepartureDate
2023-05-06
ArrivalDate
2023-05-06
AirlineNo
EY001
FlightId BclassSeatAvail EclassSeatAvail Departure Destination DepartureDate ArrivalDate AirlineNo
AC123 55 20 Toronto New York 2023-05-01 2023-05-01 AC001
AC248 55 75 Montreal Toronto 2023-05-11 2023-05-11 AC001
AF678 30 50 Paris New York 2023-05-02 2023-05-02 AF001
BA482 65 85 London Dubai 2023-05-13 2023-05-15 BA001
BA789 66 36 London New York 2023-05-03 2023-05-03 BA011
EK012 55 40 Dubai Paris 2023-05-04 2023-05-06 EK001
EK705 70 90 Dubai New York 2023-05-14 2023-05-18 EK001
EY567 45 66 Abu Dhabi New York 2023-05-09 2023-05-11 EY001
JL890 50 70 Tokyo Los Angeles 2023-05-10 2023-05-10 JL001
LH801 35 55 Frankfurt Los Angeles 2023-05-12 2023-05-12 LH001
QR234 40 60 Doha London 2023-05-28 2023-05-29 QR001
SQ345 25 45 Singapore Tokyo 2023-05-05 2023-05-06 SQ001
SQ838 75 95 Singapore London 2023-05-15 2023-05-15 SQ001
UA389 60 80 New York Los Angeles 2023-05-22 2023-05-22 UA001
UA456 77 96 Los Angeles Chicago 2023-05-02 2023-05-02 UA001
Message
Data Inserted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

Demonstrating insert,update,delete and display under some of the tables under all luggage information.

ALL LUGGAGE INFORMATIONS

BAGGAGE DETAILS

PASSENGER-LUGGAGE

FLIGHT-LUGGAGE

BACK

BAGGAGE DETAILS

BaggageId: BAG017

Status: NotChecked

Weight: 12

BaggageId	Status	Weight
BAG001	Checked	20
BAG002	Delayed	25
BAG003	NotChecked	18
BAG004	Delayed	22
BAG005	Checked	16
BAG006	Checked	21
BAG007	Delivered	19
BAG008	Delivered	24
BAG009	Checked	27
BAG010	Checked	15
BAG011	Delayed	23
BAG012	NotChecked	26
BAG013	NotChecked	17
BAG014	Delivered	30
BAG015	Delivered	28
BAG016	Checked	14

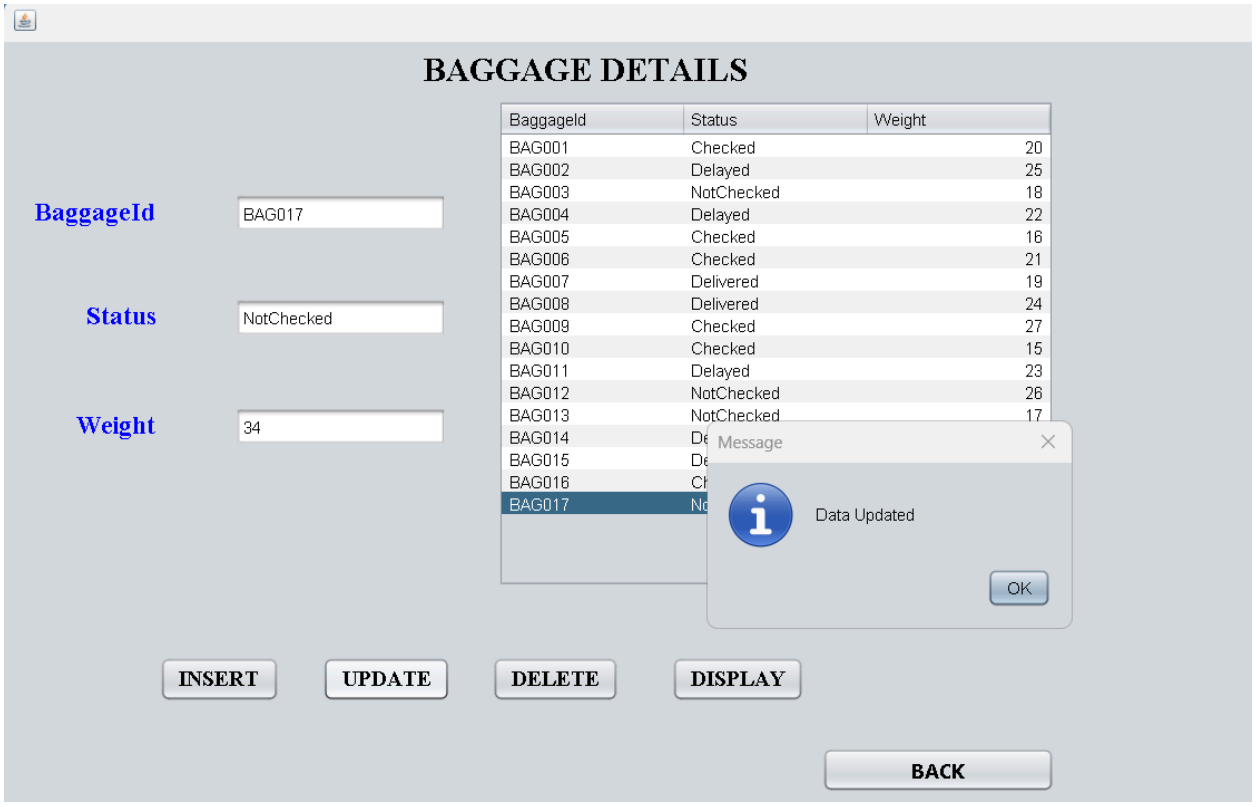
Message

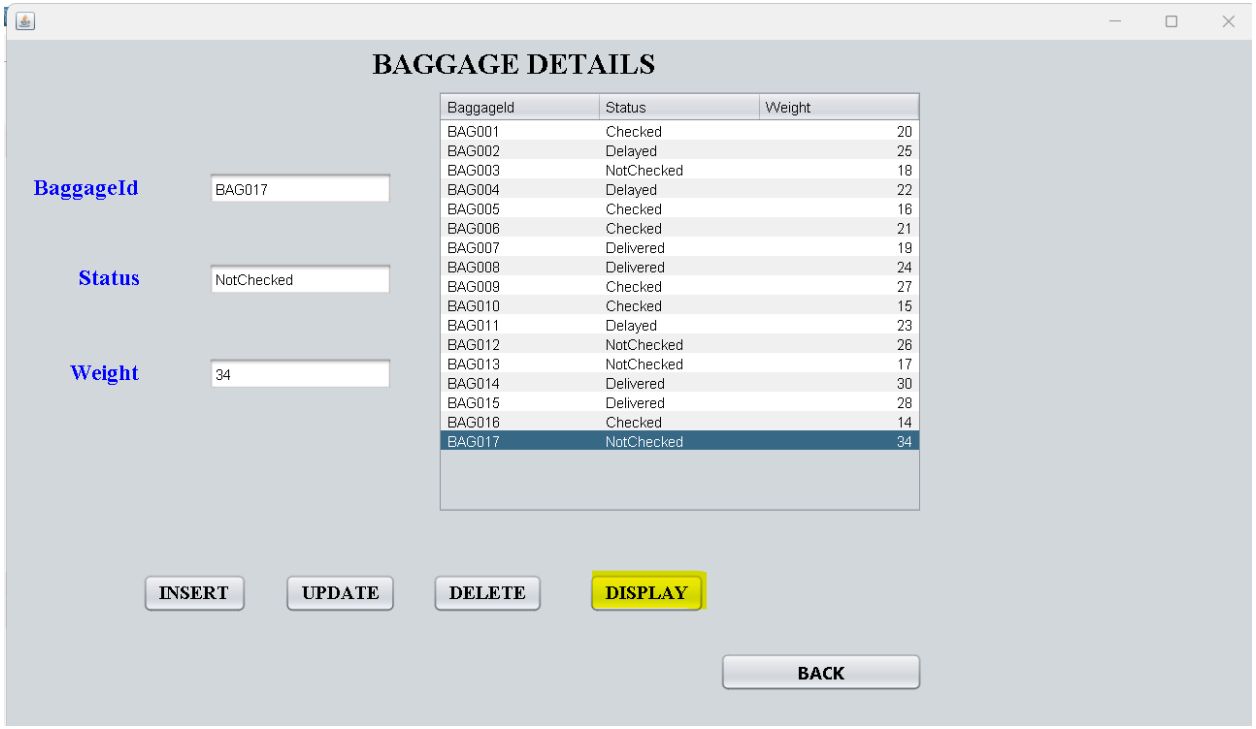
Data Inserted

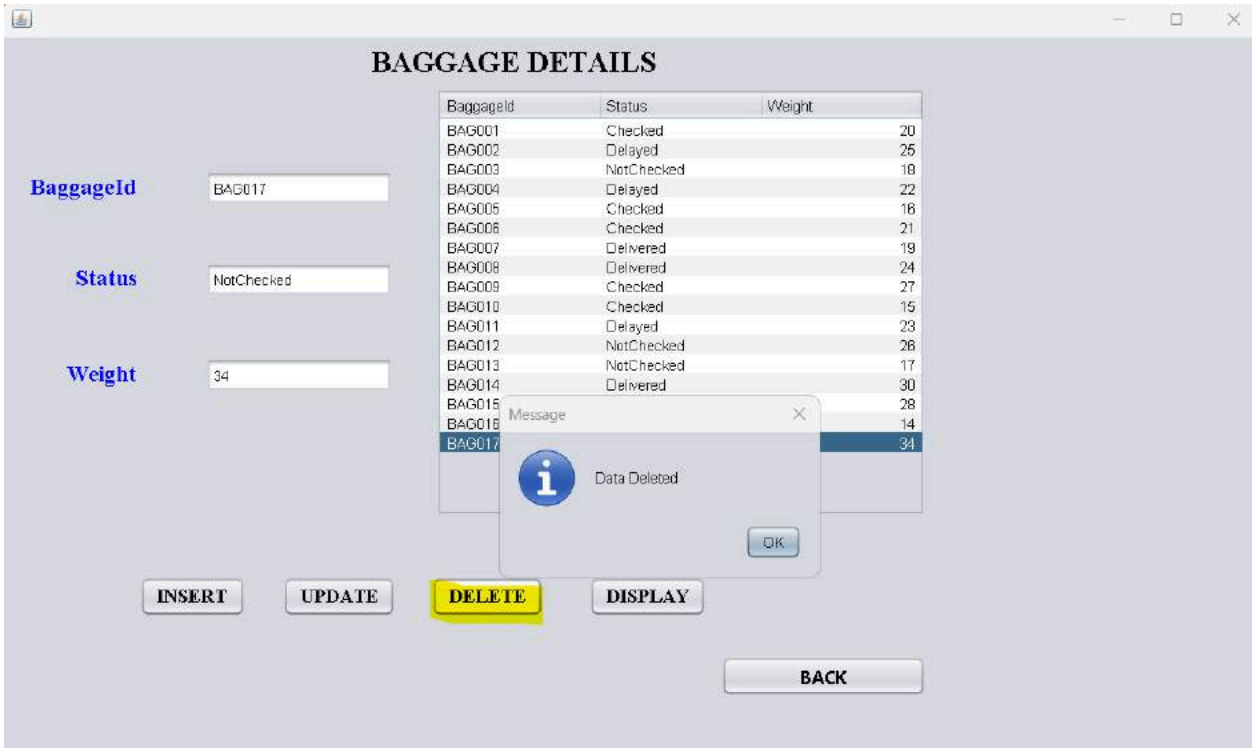
OK

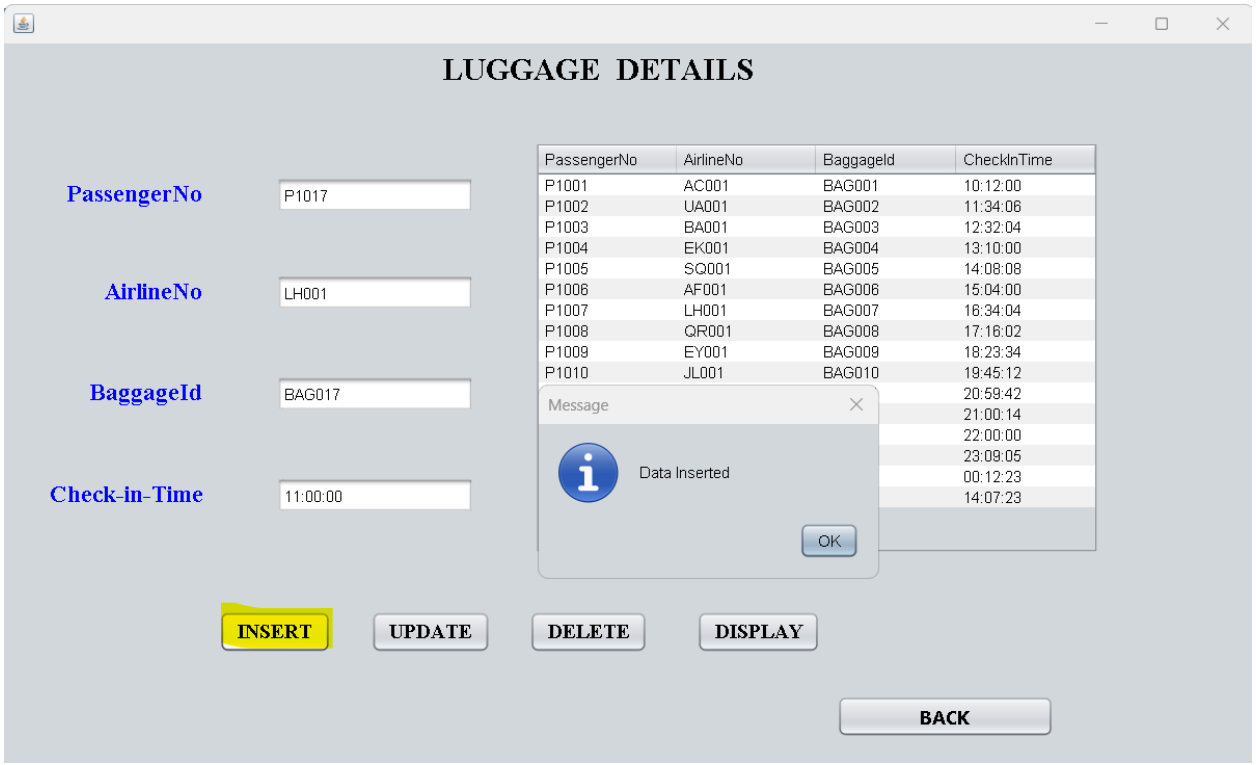
INSERT UPDATE DELETE DISPLAY

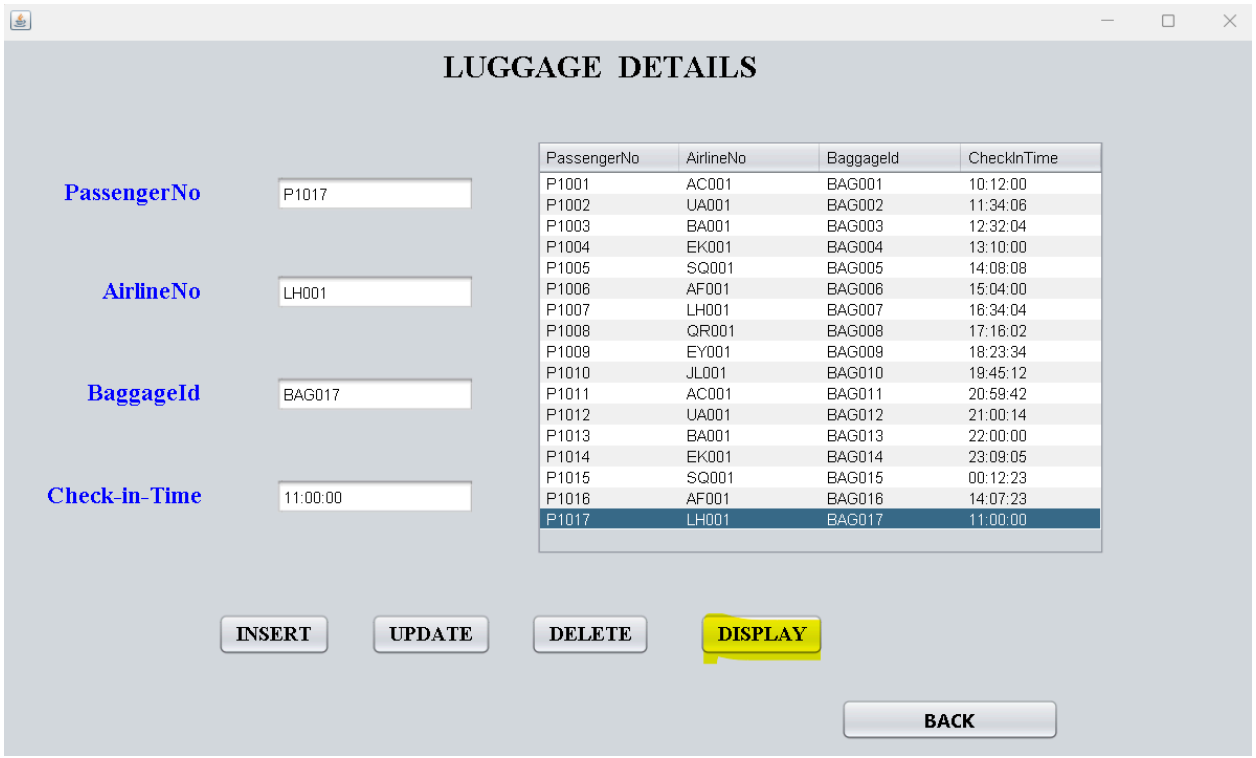
BACK

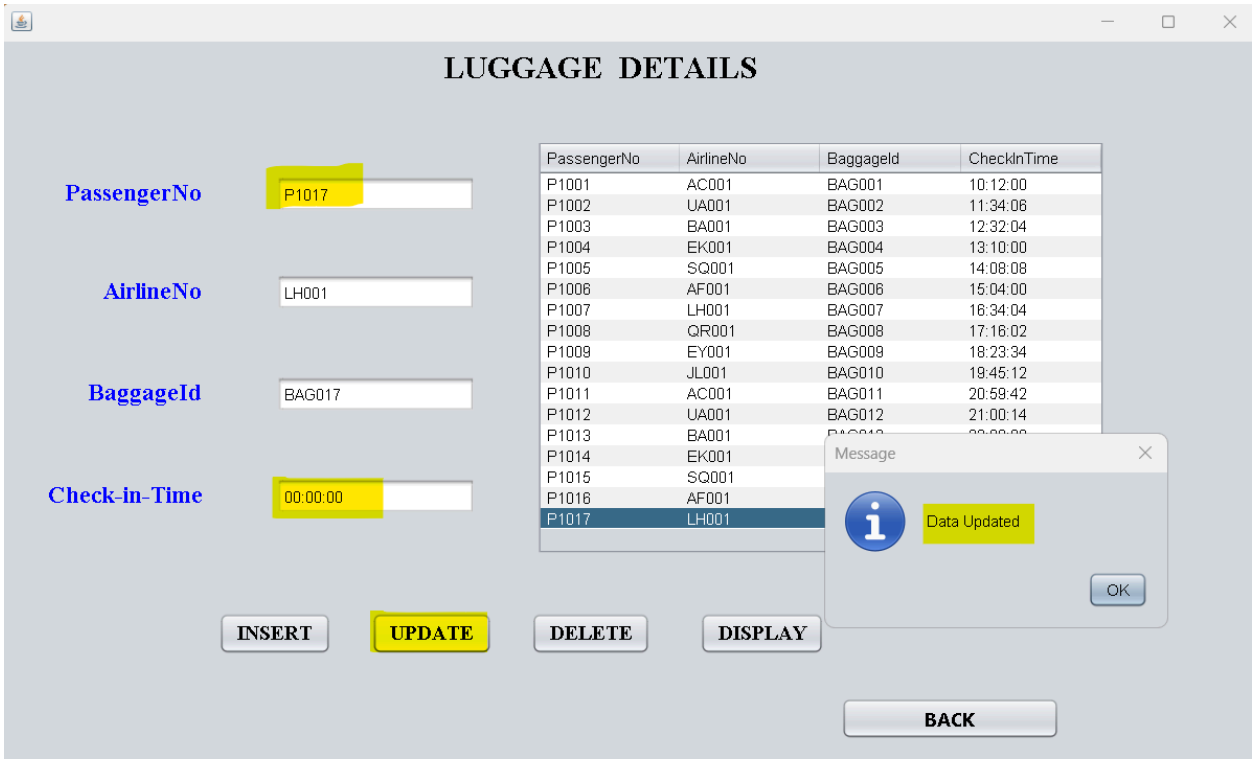

BAGGAGE DETAILS
BaggageId
BAG017
Status
NotChecked
Weight
34
BaggageId Status Weight
BAG001 Checked 20
BAG002 Delayed 25
BAG003 NotChecked 18
BAG004 Delayed 22
BAG005 Checked 16
BAG006 Checked 21
BAG007 Delivered 19
BAG008 Delivered 24
BAG009 Checked 27
BAG010 Checked 15
BAG011 Delayed 23
BAG012 NotChecked 26
BAG013 NotChecked 17
BAG017
Message
Data Updated
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

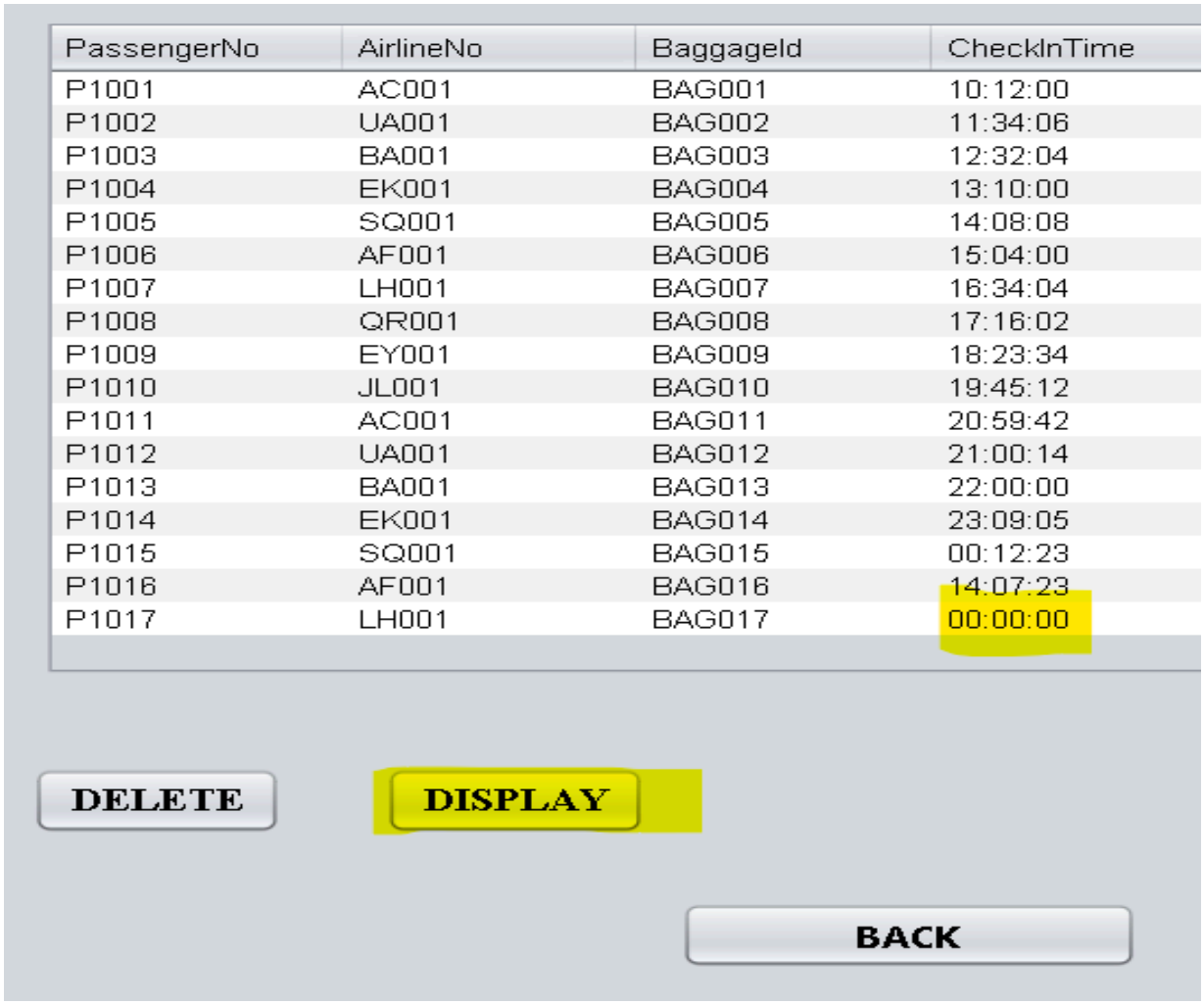

BAGGAGE DETAILS
BaggageId
BAG017
Status
NotChecked
Weight
34
BaggageId Status Weight
BAG001 Checked 20
BAG002 Delayed 25
BAG003 NotChecked 18
BAG004 Delayed 22
BAG005 Checked 16
BAG006 Checked 21
BAG007 Delivered 19
BAG008 Delivered 24
BAG009 Checked 27
BAG010 Checked 15
BAG011 Delayed 23
BAG012 NotChecked 26
BAG013 NotChecked 17
BAG014 Delivered 30
BAG015 Delivered 28
BAG016 Checked 14
BAG017 NotChecked 34
INSERT
UPDATE
DELETE
DISPLAY
BACK

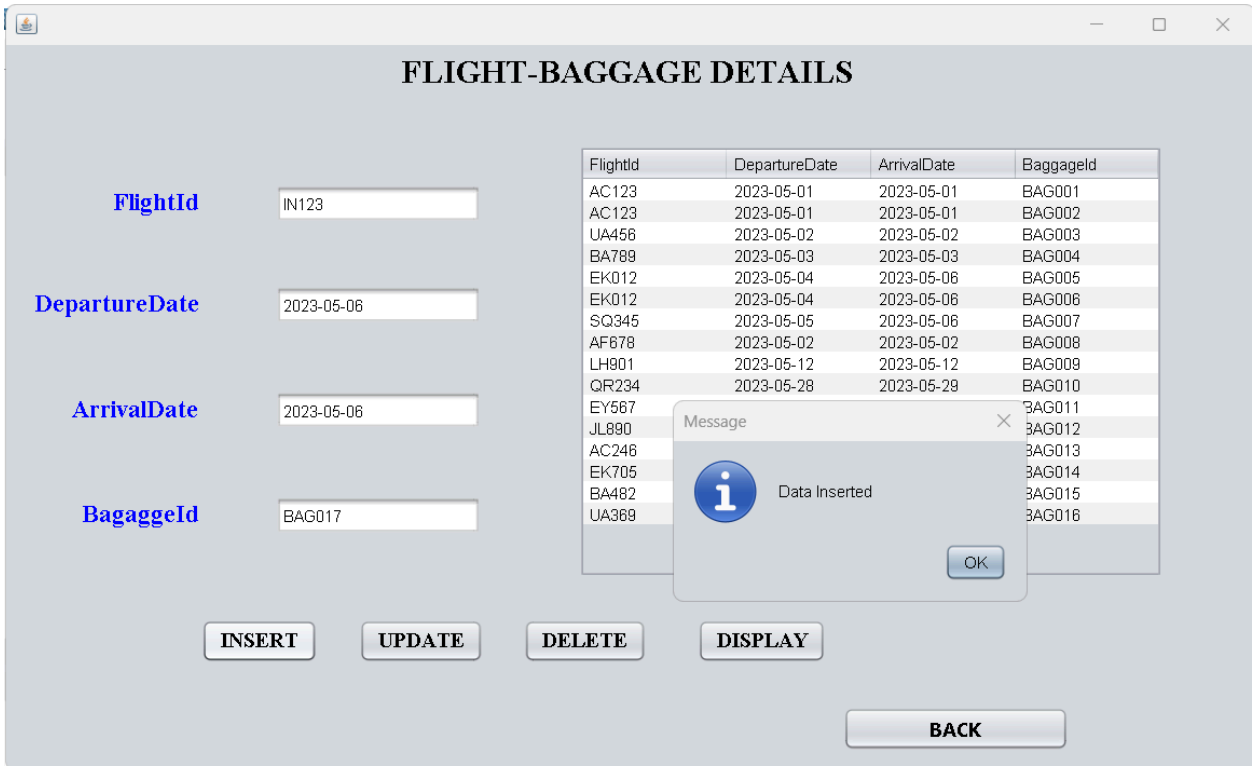

BAGGAGE DETAILS
BaggageId
BAG017
Status
NotChecked
Weight
34
BaggageId Status Weight
BAG001 Checked 20
BAG002 Delayed 25
BAG003 NotChecked 18
BAG004 Delayed 22
BAG005 Checked 16
BAG006 Checked 21
BAG007 Delivered 19
BAG008 Delivered 24
BAG009 Checked 27
BAG010 Checked 15
BAG011 Delayed 23
BAG012 NotChecked 26
BAG013 NotChecked 17
BAG014 Delivered 30
BAG015 28
BAG016 14
BAG017 34
Message
Data Deleted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

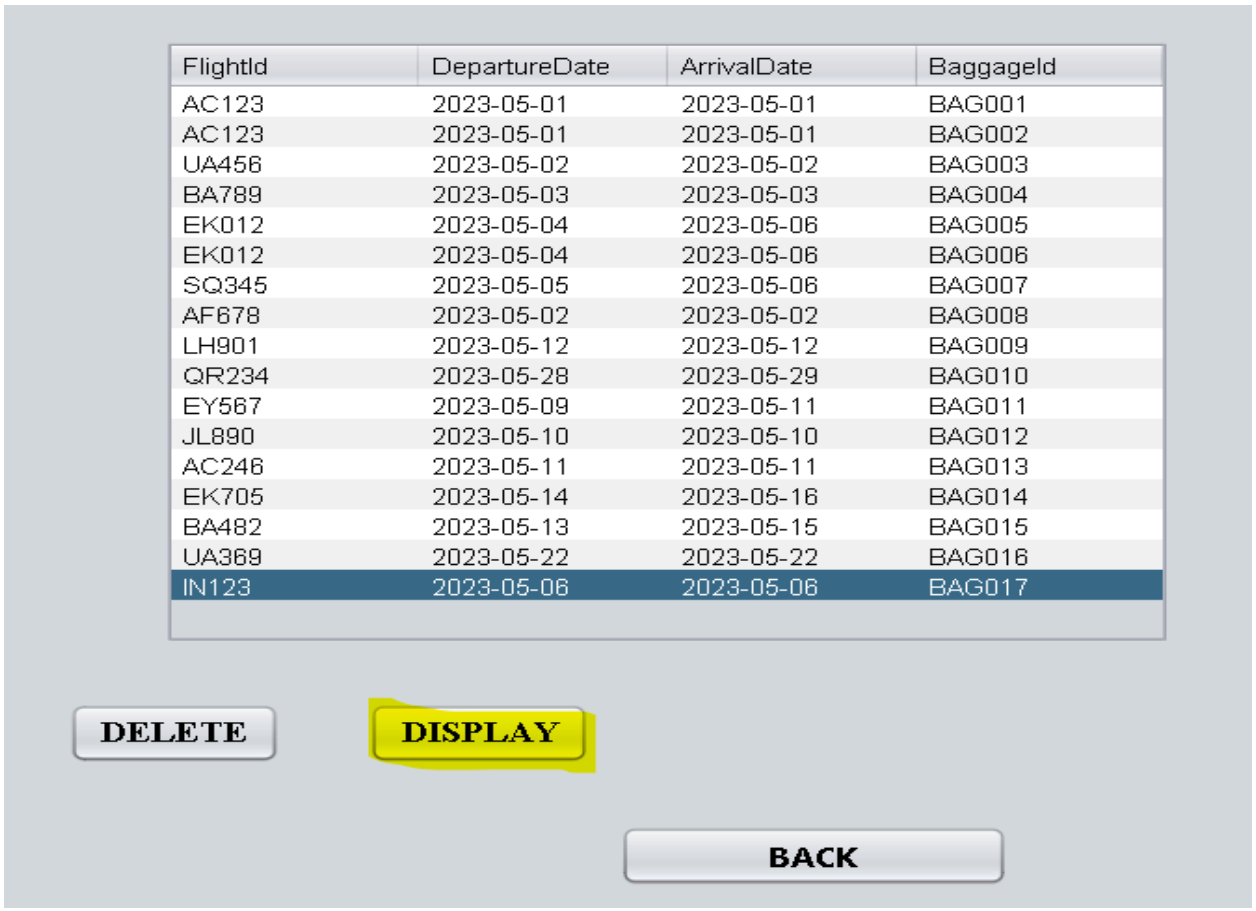

LUGGAGE DETAILS
PassengerNo
P1017
AirlineNo
LH001
BaggageId
BAG017
Check-in-Time
11:00:00
PassengerNo AirlineNo BaggageId CheckInTime
P1001 AC001 BAG001 10:12:00
P1002 UA001 BAG002 11:34:06
P1003 BA001 BAG003 12:32:04
P1004 EK001 BAG004 13:10:00
P1005 SQ001 BAG005 14:08:08
P1006 AF001 BAG006 15:04:00
P1007 LH001 BAG007 16:34:04
P1008 QR001 BAG008 17:16:02
P1009 EY001 BAG009 18:23:34
P1010 JL001 BAG010 19:45:12
20:59:42
21:00:14
22:00:00
23:09:05
00:12:23
14:07:23
Message
Data Inserted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK


LUGGAGE DETAILS
PassengerNo
P1017
AirlineNo
LH001
BaggageId
BAG017
Check-in-Time
11:00:00
PassengerNo AirlineNo BaggageId CheckInTime
P1001 AC001 BAG001 10:12:00
P1002 UA001 BAG002 11:34:06
P1003 BA001 BAG003 12:32:04
P1004 EK001 BAG004 13:10:00
P1005 SQ001 BAG005 14:08:08
P1006 AF001 BAG006 15:04:00
P1007 LH001 BAG007 16:34:04
P1008 QR001 BAG008 17:16:02
P1009 EY001 BAG009 18:23:34
P1010 JL001 BAG010 19:45:12
P1011 AC001 BAG011 20:59:42
P1012 UA001 BAG012 21:00:14
P1013 BA001 BAG013 22:00:00
P1014 EK001 BAG014 23:09:05
P1015 SQ001 BAG015 00:12:23
P1016 AF001 BAG016 14:07:23
P1017 LH001 BAG017 11:00:00
INSERT
UPDATE
DELETE
DISPLAY
BACK


LUGGAGE DETAILS
PassengerNo
P1017
AirlineNo
LH001
BaggageId
BAG017
Check-in-Time
00:00:00
PassengerNo AirlineNo BaggageId CheckInTime
P1001 AC001 BAG001 10:12:00
P1002 UA001 BAG002 11:34:06
P1003 BA001 BAG003 12:32:04
P1004 EK001 BAG004 13:10:00
P1005 SQ001 BAG005 14:08:08
P1006 AF001 BAG006 15:04:00
P1007 LH001 BAG007 16:34:04
P1008 QR001 BAG008 17:16:02
P1009 EY001 BAG009 18:23:34
P1010 JL001 BAG010 19:45:12
P1011 AC001 BAG011 20:59:42
P1012 UA001 BAG012 21:00:14
P1013 BA001
P1014 EK001
P1015 SQ001
P1016 AF001
P1017 LH001
Message
Data Updated
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

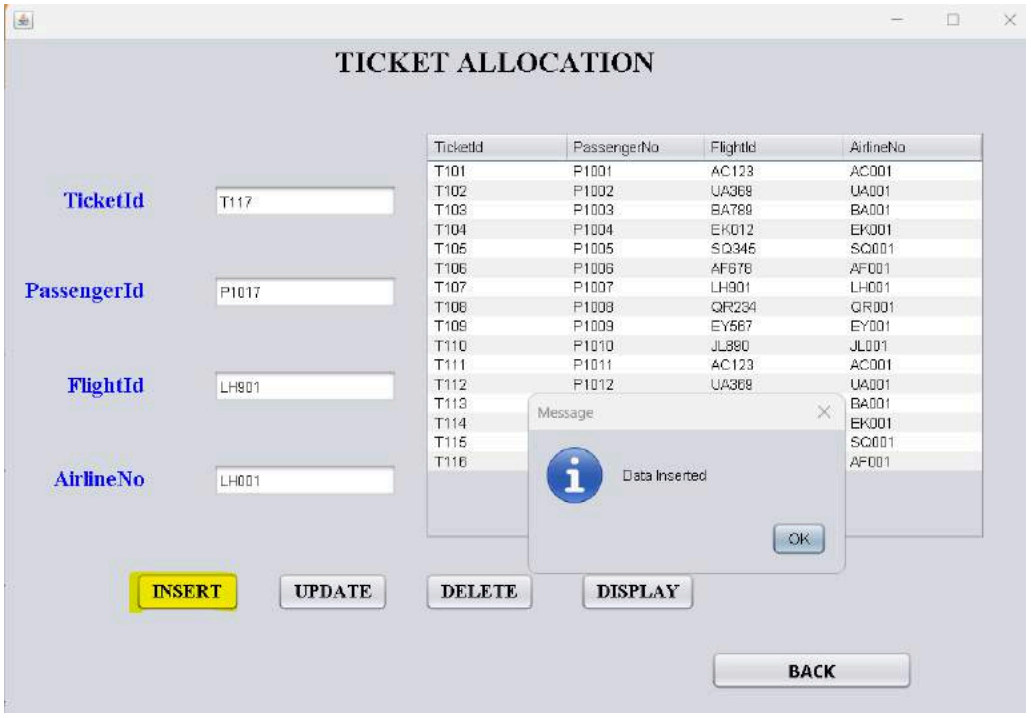

PassengerNo
AirlineNo
BaggageId
CheckInTime
P1001 AC001 BAG001 10:12:00
P1002 UA001 BAG002 11:34:06
P1003 BA001 BAG003 12:32:04
P1004 EK001 BAG004 13:10:00
P1005 SQ001 BAG005 14:08:08
P1006 AF001 BAG006 15:04:00
P1007 LH001 BAG007 16:34:04
P1008 QR001 BAG008 17:16:02
P1009 EY001 BAG009 18:23:34
P1010 JL001 BAG010 19:45:12
P1011 AC001 BAG011 20:59:42
P1012 UA001 BAG012 21:00:14
P1013 BA001 BAG013 22:00:00
P1014 EK001 BAG014 23:09:05
P1015 SQ001 BAG015 00:12:23
P1016 AF001 BAG016 14:07:23
P1017 LH001 BAG017 00:00:00
DELETE
DISPLAY
BACK


FLIGHT-BAGGAGE DETAILS
FlightId
IN123
DepartureDate
2023-05-06
ArrivalDate
2023-05-06
BagaggeId
BAG017
FlightId DepartureDate ArrivalDate BaggageId
AC123 2023-05-01 2023-05-01 BAG001
AC123 2023-05-01 2023-05-01 BAG002
UA456 2023-05-02 2023-05-02 BAG003
BA789 2023-05-03 2023-05-03 BAG004
EK012 2023-05-04 2023-05-06 BAG005
EK012 2023-05-04 2023-05-06 BAG006
SQ345 2023-05-05 2023-05-06 BAG007
AF678 2023-05-02 2023-05-02 BAG008
LH901 2023-05-12 2023-05-12 BAG009
QR234 2023-05-28 2023-05-29 BAG010
EY567 BAG011
JL890 BAG012
AC246 BAG013
EK705 BAG014
BA482 BAG015
UA369 BAG016
Message
Data Inserted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK


FlightId DepartureDate ArrivalDate BaggageId
AC123 2023-05-01 2023-05-01 BAG001
AC123 2023-05-01 2023-05-01 BAG002
UA456 2023-05-02 2023-05-02 BAG003
BA789 2023-05-03 2023-05-03 BAG004
EK012 2023-05-04 2023-05-06 BAG005
EK012 2023-05-04 2023-05-06 BAG006
SQ345 2023-05-05 2023-05-06 BAG007
AF678 2023-05-02 2023-05-02 BAG008
LH901 2023-05-12 2023-05-12 BAG009
QR234 2023-05-28 2023-05-29 BAG010
EY567 2023-05-09 2023-05-11 BAG011
JL890 2023-05-10 2023-05-10 BAG012
AC246 2023-05-11 2023-05-11 BAG013
EK705 2023-05-14 2023-05-16 BAG014
BA482 2023-05-13 2023-05-15 BAG015
UA369 2023-05-22 2023-05-22 BAG016
IN123 2023-05-06 2023-05-06 BAG017
DELETE
DISPLAY
BACK

Demonstrating insert,update,delete and display under some of the tables under all Booking Details.

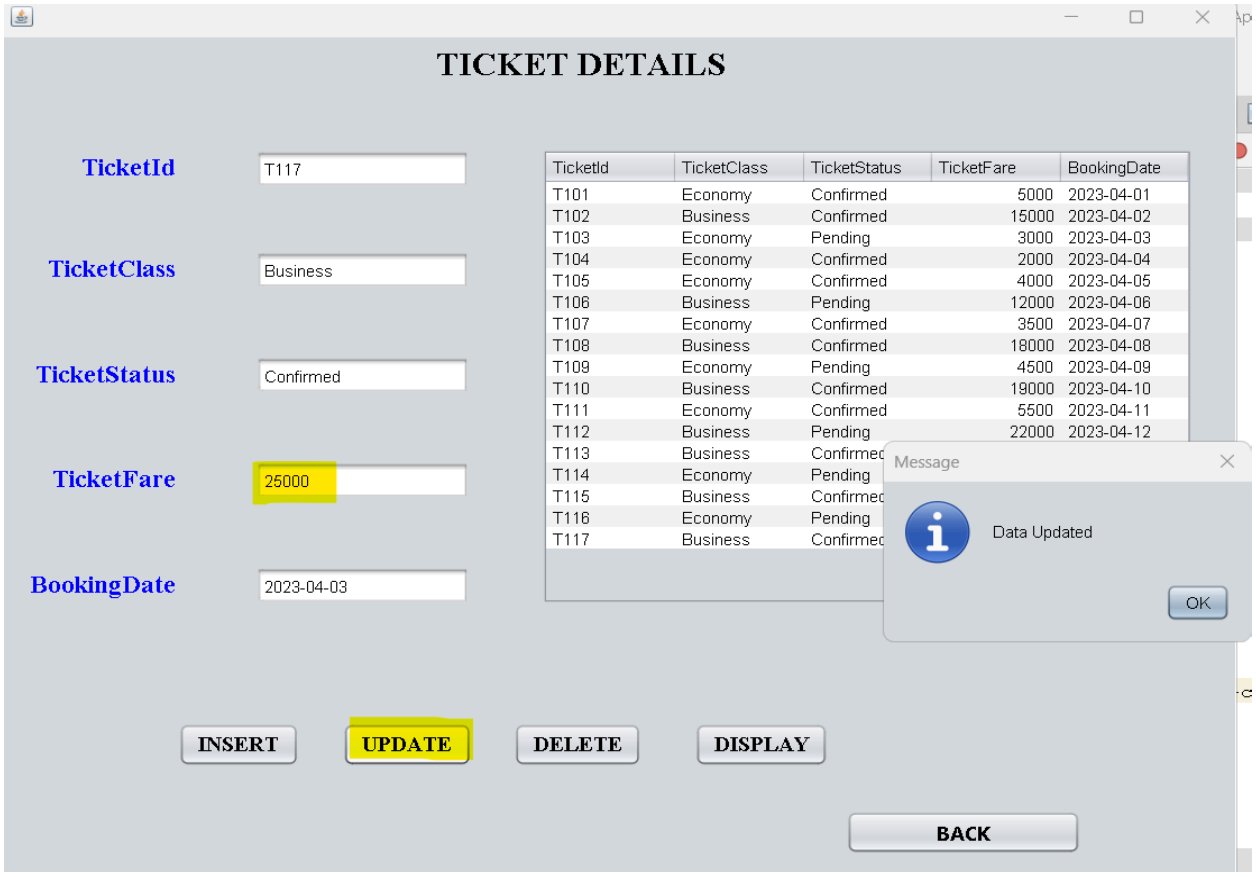

3B
3B
DC
BOOKING DETAILS
TICKET-PASSENGER-ALLOCATION
TICKET DETAILS
BOOKED-FLIGHTS
BACK

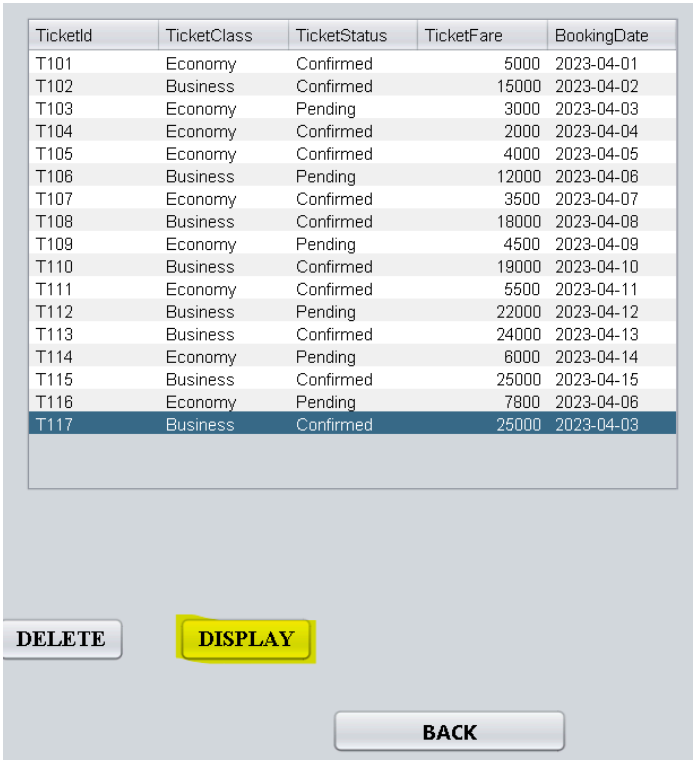

TICKET ALLOCATION
TicketId
T117
PassengerId
P1017
FlightId
LH901
AirlineNo
LH001
TicketId PassengerNo FlightId AirlineNo
T101 P1001 AC123 AC001
T102 P1002 UA369 UA001
T103 P1003 BA789 BA001
T104 P1004 EK012 EK001
T105 P1005 SQ345 SQ001
T106 P1006 AF678 AF001
T107 P1007 LH901 LH001
T108 P1008 QR234 QR001
T109 P1009 EY567 EY001
T110 P1010 JL890 JL001
T111 P1011 AC123 AC001
T112 P1012 UA369 UA001
T113 BA001
T114 EK001
T115 SQ001
T116 AF001
Message
Data Inserted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK

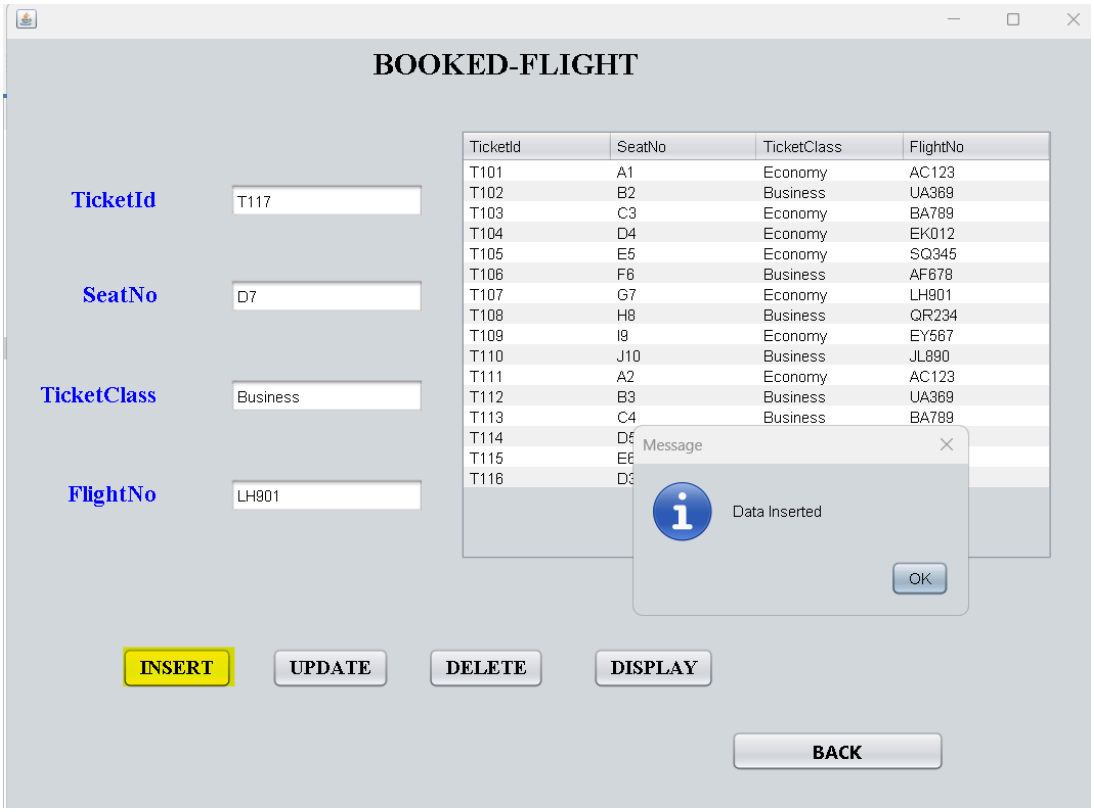
TicketId	PassengerNo	FlightId	AirlineNo
T101	P1001	AC123	AC001
T102	P1002	UA369	UA001
T103	P1003	BA789	BA001
T104	P1004	EK012	EK001
T105	P1005	SQ345	SQ001
T106	P1006	AF678	AF001
T107	P1007	LH901	LH001
T108	P1008	QR234	QR001
T109	P1009	EY567	EY001
T110	P1010	JL890	JL001
T111	P1011	AC123	AC001
T112	P1012	UA369	UA001
T113	P1013	BA789	BA001
T114	P1014	EK012	EK001
T115	P1015	SQ345	SQ001
T116	P1016	AF678	AF001
T117	P1017	LH901	LH001

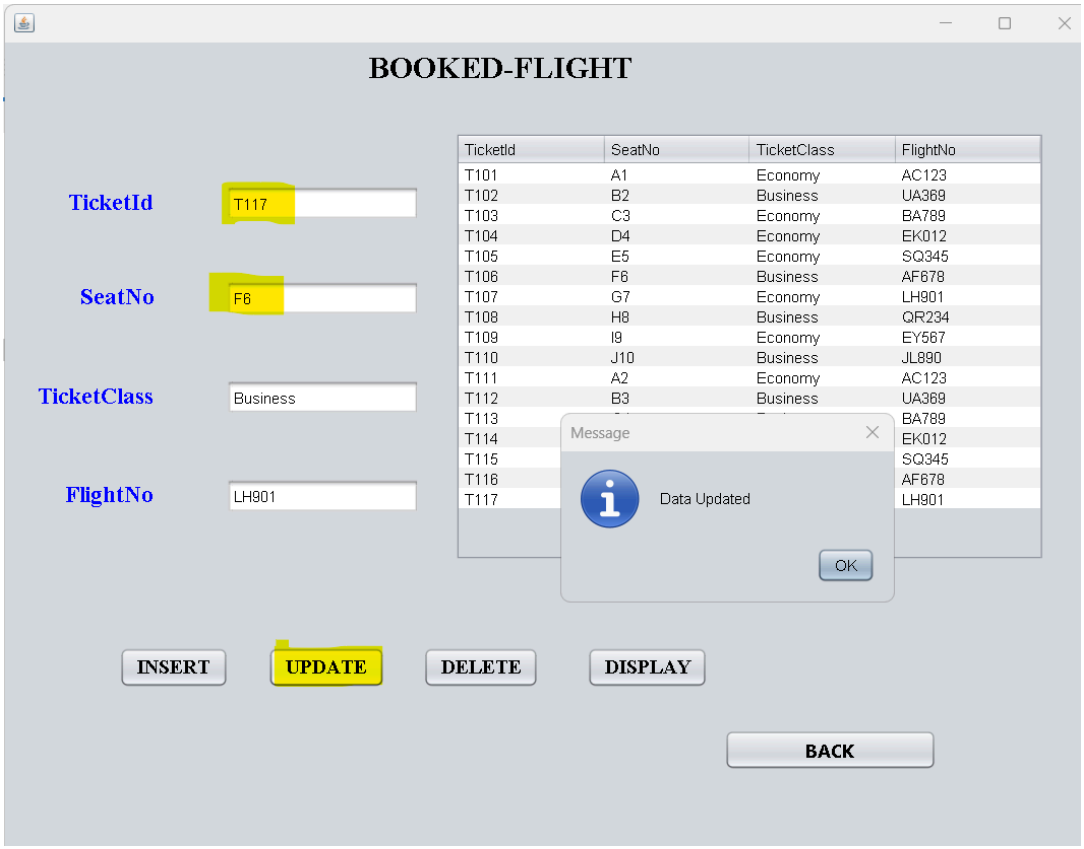
DELETE DISPLAY

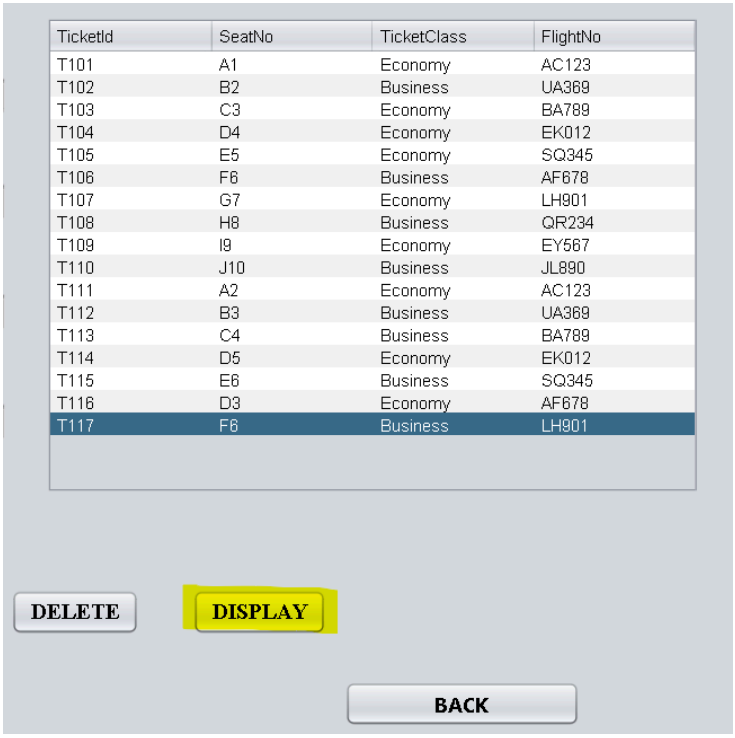
BACK


TICKET DETAILS
TicketId
T117
TicketClass
Business
TicketStatus
Confirmed
TicketFare
25000
BookingDate
2023-04-03
TicketId | TicketClass | TicketStatus | TicketFare | BookingDate
T101 Economy Confirmed 5000 2023-04-01
T102 Business Confirmed 15000 2023-04-02
T103 Economy Pending 3000 2023-04-03
T104 Economy Confirmed 2000 2023-04-04
T105 Economy Confirmed 4000 2023-04-05
T106 Business Pending 12000 2023-04-06
T107 Economy Confirmed 3500 2023-04-07
T108 Business Confirmed 18000 2023-04-08
T109 Economy Pending 4500 2023-04-09
T110 Business Confirmed 19000 2023-04-10
T111 Economy Confirmed 5500 2023-04-11
T112 Business Pending 22000 2023-04-12
T113 Business Confirmed
T114 Economy Pending
T115 Business Confirmed
T116 Economy Pending
T117 Business Confirmed
Message
Data Updated
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK


TicketId | TicketClass | TicketStatus | TicketFare | BookingDate
T101 Economy Confirmed 5000 2023-04-01
T102 Business Confirmed 15000 2023-04-02
T103 Economy Pending 3000 2023-04-03
T104 Economy Confirmed 2000 2023-04-04
T105 Economy Confirmed 4000 2023-04-05
T106 Business Pending 12000 2023-04-06
T107 Economy Confirmed 3500 2023-04-07
T108 Business Confirmed 18000 2023-04-08
T109 Economy Pending 4500 2023-04-09
T110 Business Confirmed 19000 2023-04-10
T111 Economy Confirmed 5500 2023-04-11
T112 Business Pending 22000 2023-04-12
T113 Business Confirmed 24000 2023-04-13
T114 Economy Pending 6000 2023-04-14
T115 Business Confirmed 25000 2023-04-15
T116 Economy Pending 7800 2023-04-06
T117 Business Confirmed 25000 2023-04-03
DELETE
DISPLAY
BACK


BOOKED-FLIGHT
TicketId
T117
SeatNo
D7
TicketClass
Business
FlightNo
LH901
TicketId SeatNo TicketClass FlightNo
T101 A1 Economy AC123
T102 B2 Business UA369
T103 C3 Economy BA789
T104 D4 Economy EK012
T105 E5 Economy SQ345
T106 F6 Business AF678
T107 G7 Economy LH901
T108 H8 Business QR234
T109 I9 Economy EY567
T110 J10 Business JL890
T111 A2 Economy AC123
T112 B3 Business UA369
T113 C4 Business BA789
T114
T115
T116
Message
Data Inserted
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK


BOOKED-FLIGHT
TicketId
T117
SeatNo
F6
TicketClass
Business
FlightNo
LH901
TicketId SeatNo TicketClass FlightNo
T101 A1 Economy AC123
T102 B2 Business UA369
T103 C3 Economy BA789
T104 D4 Economy EK012
T105 E5 Economy SQ345
T106 F6 Business AF678
T107 G7 Economy LH901
T108 H8 Business QR234
T109 I9 Economy EY567
T110 J10 Business JL890
T111 A2 Economy AC123
T112 B3 Business UA369
T113 BA789
T114 EK012
T115 SQ345
T116 AF678
T117 LH901
Message
Data Updated
OK
INSERT
UPDATE
DELETE
DISPLAY
BACK


TicketId SeatNo TicketClass FlightNo
T101 A1 Economy AC123
T102 B2 Business UA369
T103 C3 Economy BA789
T104 D4 Economy EK012
T105 E5 Economy SQ345
T106 F6 Business AF678
T107 G7 Economy LH901
T108 H8 Business QR234
T109 I9 Economy EY567
T110 J10 Business JL890
T111 A2 Economy AC123
T112 B3 Business UA369
T113 C4 Business BA789
T114 D5 Economy EK012
T115 E6 Business SQ345
T116 D3 Economy AF678
T117 F6 Business LH901
DELETE
DISPLAY
BACK