

# Flight Booking System

---

By Nicholas Imperius, Kristopher Poulin, Jimmy Tsang

## INTRODUCTION

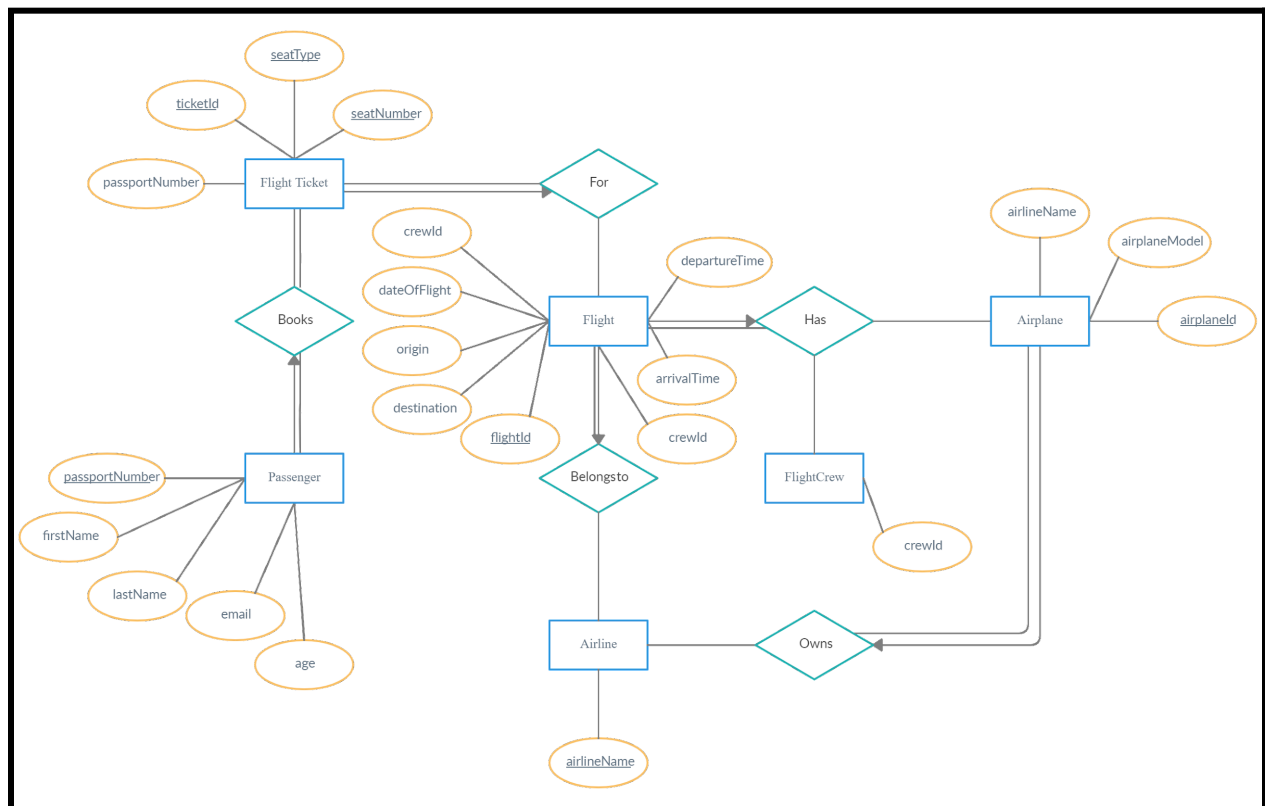
The idea for our project was to design a flight booking system. In order to accomplish this task, we had to design a database that would allow us to focus on certain functionalities and perfect them. In our interface, we give the user the ability to book flights for passengers, create passengers as well as see what passengers are on which flights; flights could be created as well as the flight tickets they are composed of and more. Our goal, again, was not to create something huge and super complex but to create something that would allow users to efficiently do certain tasks to maximize time spent doing something. In the end, all three of us felt fairly comfortable with the subject matter which is a large part of the reason that we decided to do this project on the topic of flight booking. This allowed us to get to the design phase quicker since a lot of the background knowledge was there and not a lot of research was needed.

Throughout the entire project, we were always learning new things from the PHP language to HTML to ER Diagrams. We also feel that working in a team allowed us to gain valuable experience that could help us later in our careers.

# DESIGN

## ER Diagram

Appendix A has an ERD that is the first attempt of our database. After implementing the tables and relationships into MySQL we realized that it was not going to work the way we had intended. We took a step back and thought about what actually needs to connect with each other. By doing this, we moved a few relationships around and added some participation to certain relationships that we feel gave us exactly what we were trying to achieve in the first place, but this time it worked.



The ER Diagram has many relationships that connect all the entities to one another as described below

### **Books Relationship**

Each flight ticket can only have one passenger (you can't have a flight ticket without a passenger) and a passenger must have at least one flight ticket. Hence, there is total participation.

### **For Relationship**

Flight tickets must belong to a flight, but a flight does not have to have flight tickets. This logically makes sense because without a flight there would not be a flight ID for that flight ticket.

### **Belong To Relationship**

Each flight must belong to only one airline and an airline does not need to have a flight. In normal circumstances, the same flight is not for 2 different airlines hence why this is why it is.

### **Has Relationship**

Every flight has to have one airplane and one flight crew, but an airplane and flight crew do not have to belong to a flight. Without an airplane and/or flight crew, firstly there would not be a way to get to the destination and secondly, there would be no one to fly the plane thus our reasoning for why it is like this.

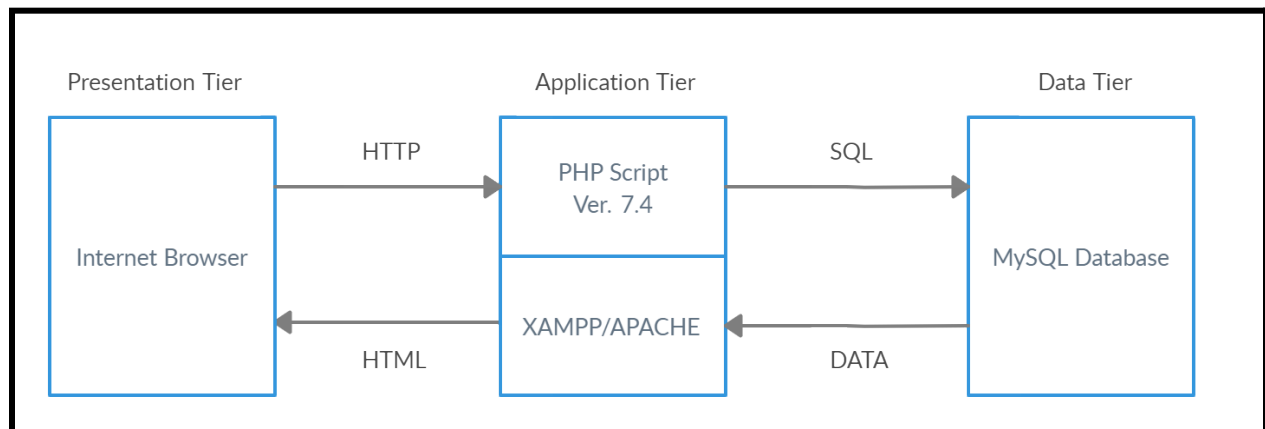
## Owns Relationship

Each airplane must be owned by an airline, but an airline does not need to own an airplane. Since every airline owns their respective airplanes, there is not any sharing amongst airlines since usually, they have decals and colour palettes that support the company's colours.

## Creation of Tables

After some research on flight booking systems and their function, we came up with a general idea of the entities/participants as well as the relationships between them. From there we came up with all possible entities that would fit into our use case and then narrowed it down to a selection that we would be able to implement within the time allotted for the project. As we integrated the diagram into our SQL tables, there were some constraints that needed to be transferred over. For instance, in our Flight table (view Appendix B) we have a set of 3 primary keys that each represent a unique flight and it is a child table to Airplane, Airline, and FlightCrew since we have combined the *Has* relationship into this table. Another example would be our FlightTicket table (view Appendix C) where, again, we have another set of 3 primary keys this time identifying a passenger's seat on a flight; this table is also a child table since it takes on information from Passengers and Flight.

## Technology Used



To first start developing our project, we needed to design our Entity-Relationship diagram, which was built using online software called Creately. Three-tier architecture is a software development structure consisting of three tiers, the data, application, and presentation tiers. The data tier consists of the database where information is stored, retrieved, and altered. For the development of our database, there were a few software products used. The data tier software used in our project was structured query language (SQL) through a software known as MySQL. The application tier contains the functional information which is used to drive an application's core functionality by performing detailed processing. The application tier software used in our project was hypertext processor (PHP) in conjunction with XAMPP, which is a web server software used to connect the Apache Web Server and MySQL database with PHP which allows us to run our interface locally on our machines. The presentation tier consists of the user interface; essentially, how the user interacts with our database. The presentation tier

software utilized in our presentation is a hypertext markup language (HTML) in an internet browser.

## **Querying**

Our interface has many different forms that allow the user to add or remove tuples as well as search and update functionality too. Appendix D to Appendix G are the forms for adding information to the tables while Appendix H to Appendix K is for removing information. As you can see in Appendix L we ask the user to enter the flight ID for the flight they are wanting to update along with the new information that needs to be updated; logically, this is mainly to be used when a flight is late or early leaving or arriving from the origin or destination. In Appendix M we are allowing the user to update a passenger's seat, we just need their passport number since that is the main form of identification and then we just require the new seat information as well as for which flight it is for that passenger. We also give the user the ability to search and display what passengers are on which flights as can be seen in Appendix N. Lastly, the ability to search for flights based on airlines or origin cities has been given to the user. In both instances, we require either the airline name or origin city and the date for which to search as seen in Appendix O.

## **Testing and Implementation**

In order to test our database and confirm correct functionality we had to create and insert fake data into the system. We created 25 passengers, 4 airlines, 11 airplanes, 10

flight crews, 13 flights, and 25 flight tickets for a total of 88 entries as seen in Appendix P to Appendix U. Before writing any PHP code we wanted to make sure the logic of our database was working properly. We designed test SQL queries that would test all the boundaries of our system to ensure correct functionality. For instance, we created different types of UPDATE, INSERT and REMOVE queries to test the relationships between tables. To test our interface, we also had to verify that the queries were running correctly in addition to the functionality of the form. To test this we input values into the text boxes located on each form to verify that it could handle correct and incorrect input and respond accordingly for the user to see. The hardest part of the testing was discovering an error in the structure of our tables and having to rethink how they interact with each other, such as child and parent tables and how the user interacts with them. One of the easier aspects was designing the HTML interface, since we had an idea of what we wanted it to look like it was much easier to translate our sketches to code. A large amount of time was spent in the testing and implementation phase and we even had to go back to the design phase to rethink how a certain piece of functionality was supposed to operate on the back end. Although we felt the final project is satisfactory, that is not to say there were no errors along the way. Many errors occurred, causing great frustration; however, we persevered and were able to present a project we are satisfied with. Some of these errors include running a php file and connecting it to a server, passing more than one value through a php file, table constraints, and conflicts between how we originally designed our tables versus how we were to query the tuples. Although, the biggest issue encountered was getting input

from the user in a textbox while using a submit button to confirm the information. Since PHP is server-side, it means that it runs once unlike client-side code which can continuously loop and wait for user input. However, we were able to resolve all of our issues. With research and help from the TA, we were able to get our server up and running, Nicholas actually had to use a different computer in order for the server to work. To resolve the issue of passing more than one value through a php file, we had to simplify the amount of information we take in at once by adding a separate file for each individual dropdown item for the add and remove buttons. To resolve the issue of the constraints on our tables based on our ER diagram, we simply needed to redesign our ER diagram. Lastly, HTML has an `isset()` function that would allow us to run code when a button is pressed. Thus, all major issues have been settled and we end up with our final product.

## Conclusion

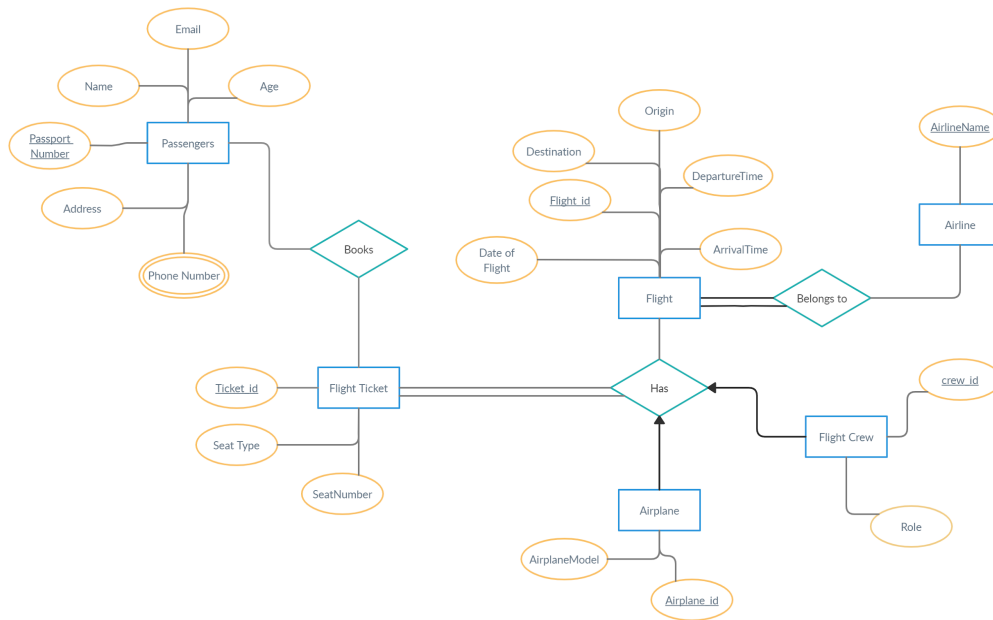
In conclusion, the project allowed us to deepen our understanding of MySQL and PHP programming, as well as develop a deeper understanding of how each component of a flight system is incorporated as well as its purpose in the database. We sought to help with the operational side of flight booking by creating a system that would allow an operator to quickly and effectively manage their booking software. Along the way, we dealt with numerous challenges from learning PHP to translating ER Diagrams into MySQL tables. We were; however, able to overcome them and gain newfound knowledge in these areas which will help us in our future endeavours as software engineers. If we



given more time, we feel as if we could have expanded our project to include more features, such as allowing the flight crew table to include names of the pilot and co-pilot as well as the flight attendant names which would allow for more control of who consists of that particular flight crew. Additionally, correcting the flight times to account for time zones would have like to have been implemented if given more time. Lastly, the individual efforts of each team member came together to form this flight management system. Nicholas was the lead on query implementation and query creation, in addition to database design. Jimmy was the lead on system specification, data collection, and modelling, in addition to the ER diagram design. Kristopher was the lead on the user interface, as well as systems testing. All other aspects of the projects were collectively completed with equal contribution. Although there were leads, that is not to say that the others did not contribute to that section. Being lead was more about giving directions so we could efficiently complete the task under one person's vision.

# Appendix

A:



B:

```

CREATE TABLE Flight(
    flightId CHAR(6),
    dateOfFlight DATE,
    origin CHAR(15),
    destination CHAR(15),
    departureTime DATETIME,
    arrivalTime DATETIME,
    airplaneId CHAR(6),
    crewId INTEGER,
    airlineName CHAR(20),
    PRIMARY KEY (flightId, dateOfFlight, departureTime),
    FOREIGN KEY (airplaneId) REFERENCES Airplane(airplaneId) ON DELETE SET NULL ON UPDATE CASCADE,
    FOREIGN KEY (airlineName) REFERENCES Airline(airlineName) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (crewId) REFERENCES FlightCrew(crewId) ON DELETE SET NULL ON UPDATE CASCADE
); #assume departure and arrival time are in the same time zone as origin
  
```

**C:**

```
CREATE TABLE FlightTicket(
    passportNumber INTEGER,
    ticketId CHAR(6),
    seatType CHAR(30),
    seatNumber CHAR(3),
    PRIMARY KEY (ticketId, seatType, seatNumber),
    FOREIGN KEY (passportNumber) REFERENCES Passengers(passportNumber) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (ticketId) REFERENCES Flight(flightId) ON DELETE CASCADE ON UPDATE CASCADE
);
```

**D:**



**Add Airline**

Enter the name of the Airline:

```
INSERT INTO Airline VALUES (airlineName);
```

**E:**



**Create Passenger**

Enter the Passport Number:

Enter the First Name:

Enter the Last Name:

Enter the Email:

Enter the Age:

```
INSERT INTO Passenger VALUES (passportNumber, firstName, lastName,  
email, age);
```

**F:**

## Create Flight

Enter the Flight ID:

Enter the Date of Departure:

Enter the Origin:

Enter the Destination:

Enter the Departure Time:

Enter the Arrival Time:

Enter the Airplane ID:

Enter the Crew ID:

Enter the Airline Name:

```
INSERT INTO Flight VALUES (flightID, departureDate, origin,
destination, departureTime, arrivalTime, airplaneID, crewID,
airlineName);
```

**G:**

## Create Flight Ticket

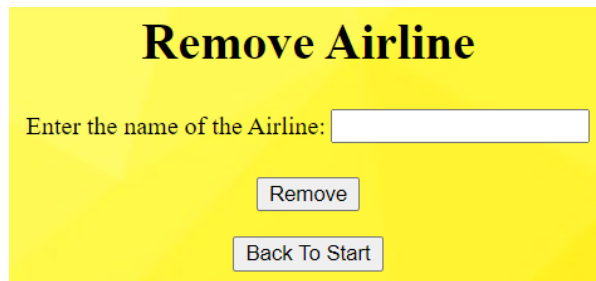
Enter the Passport Number:

Enter the Ticket ID:

Enter the Seat Type:

Enter the Seat Number:

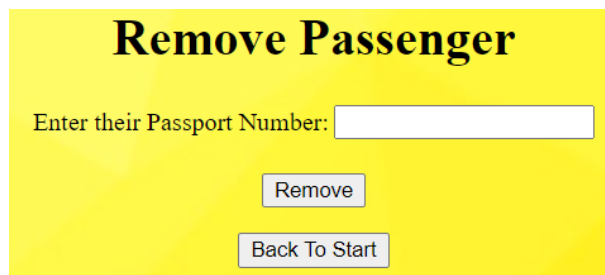
```
INSERT INTO FlightTicket VALUES (passportNumber, tioketID, seatType,
seatNumber);
```

**H:**

**Remove Airline**

Enter the name of the Airline:

```
DELETE FROM Airline WHERE airlineName = airlineName;
```

**I:**

**Remove Passenger**

Enter their Passport Number:

```
DELETE FROM Passenger WHERE passportNumber = passportNumber;
```

**J:**

**Remove Flight**

Enter the flight ID:

```
DELETE FROM Flight WHERE flightId = flightID;
```

**K:**

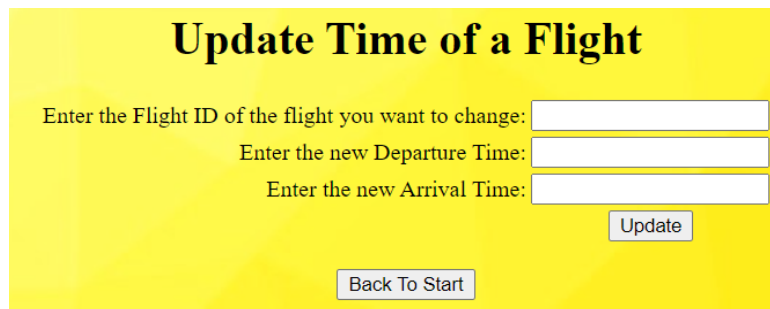


**Remove Flight Ticket**

Enter the Flight Ticket ID:

```
DELETE FROM FlightTicket WHERE ticketId = ticketID;
```

**L:**



**Update Time of a Flight**

Enter the Flight ID of the flight you want to change:

Enter the new Departure Time:

Enter the new Arrival Time:

```
UPDATE Flight SET departureTime = departureTime, arrivalTime = arrivalTime WHERE flightId = flightID;
```

**M:**



**Update the Seat of a Passenger**

Enter the passenger's Passport Number:

Enter their ticket ID:

Enter their new Seat Number:

Enter their new Seat Type:

```
UPDATE FlightTicket SET seatType = seatType, seatNumber = seatNumber WHERE ticketId = ticketID AND passportNumber = passportNumber;
```

**N:**



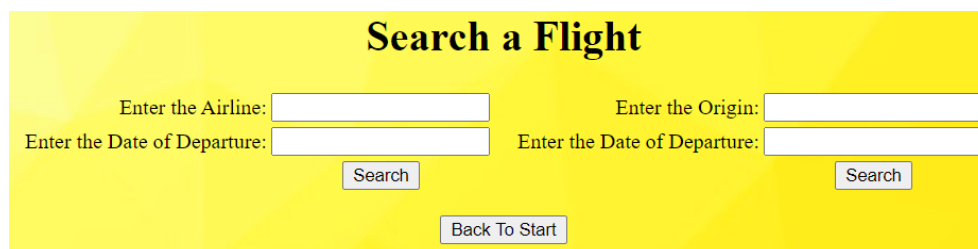
**Search Passengers**

Enter the Flight ID:

Enter the Date of Departure:

```
SEARCH Passengers.passportNumber, Passengers.firstName,
Passengers.lastName FROM Flight INNER JOIN FlightTicket ON
Flight.flightID = FlightTicket.ticketId INNER JOIN Passengers ON
FlightTicket.passportNumber = Passengers.passportNumber WHERE
Flight.flightId LIKE flightID AND Flight.dateOfFlight LIKE date;
```

**O:**



**Search a Flight**

Enter the Airline:  Enter the Origin:

Enter the Date of Departure:  Enter the Date of Departure:

```
SELECT * FROM Flight WHERE flightId LIKE airlinePrefix AND
dateOfFlight = date;
```

```
SELECT * FROM Flight WHERE origin LIKE origin AND dateOfFlight =
date;
```

**P:**

## Airline

airlineName
AirlessCanada
Eastjet
Sigma Airlines
Wolfskin Airlines

```
SELECT * FROM Airline;
```

**Q:**

## Airplane

airplaneId	airplaneModel	airlineName
A198UU	Boeing 777	Sigma Airlines
BD121A	Boeing 777	AirlessCanada
F354JY	Boeing 777	AirlessCanada
J23HG2	Bombardier Q400	EastJet
J372LK	Boeing 787	EastJet
K372JP	Boeing 787	EastJet
N973JM	Boeing 777	EastJet
P0238Y	Boeing 777	Sigma Airlines
Q32HT8	Boeing 777	AirlessCanada
UHY654	Bombardier Q400	AirlessCanada
X32H1L	Bombardier Q400	Wolfskin Airlines

```
SELECT * FROM Airplane;
```



R:

## Passengers

passportNumber	firstName	lastName	email	age
165912112	Kevin Z.	Hernandez	KHernandez@gmail.com	69
183465870	Carrie L.	Jefferies	CJeffries83@outlook.com	54
226799985	Ralph K.	Edmond	REdmond52@hotmail.com	51
229477478	Irene S.	Grosebeck	IGrossbeck61@hotmail.com	63
323033958	Colleen D.	Muniz	CMuniz55@gmail.com	61
336524494	James B.	Rodriguez	JRodriguez97@outlook.com	47
347338810	Hunter L.	Richards	HRichards71@outlook.com	20
348502853	Sean S.	Lefler	SLefler14@gmail.com	23
361054356	Katie E.	Holt	KHolt14@gmail.com	25
384175811	Sarah R.	Horton	SHorton51@icloud.com	78
411858833	Maria J.	Wheeler	MWheeler63@icloud.com	59
413816856	Rodolfo M.	Neblett	RNeblett19@gmail.com	35
431415524	Jennifer A.	Cotter	JAcotter@gmail.com	23
455423760	Jacqueline W.	Johnson	JJohnson24@outlook.com	42
501872285	Robert J.	Taylor	RTaylor82@yahoo.ca	27
523864301	John N.	Gerke	JGerke58@icloud.com	55
570383268	Callie J.	Reeder	CReeder@gmail.com	39
585558727	Daniel D.	Hardison	DHardison21@gmail.com	19
611058959	Johnnie R.	Blanco	JBlanco98@icloud.com	31
657075248	Judy B.	Otto	JOtto69@yahoo.ca	16
705872764	Linda T.	Martinez	LMartinez09@hotmail.com	52
768437119	Gilbert A.	Overstreet	GOverstreet11@yahoo.ca	46
917962210	Mary A.	Vera	MVera67@hotmail.com	87
929417213	Adria E.	Chesson	AChesson18@yahoo.ca	34
975176880	Josefina J.	Buchanan	JBuchanan98@hotmail.com	48

[Back To Start](#)

```
SELECT * FROM Passengers;
```

S:

## Flight

flightId	dateOfFlight	origin	destination	departureTime	arrivalTime	airplaneId	crewId
AC1201	2020-10-23	Calgary	Vancouver	2020-10-23 06:50:00	2020-10-23 08:20:00	UHY654	10
AC1313	2020-10-22	Winnipeg	Calgary	2020-10-22 22:45:00	2020-10-23 01:00:00	Q32HT8	4
AC1314	2020-10-23	Calgary	Winnipeg	2020-10-23 09:05:00	2020-10-23 11:00:00	Q32HT8	4
AC1567	2020-10-22	Toronto	Ottawa	2020-10-22 12:00:00	2020-10-22 13:00:00	F354JY	5
AC1874	2020-10-22	Winnipeg	Toronto	2020-10-22 09:42:00	2020-10-22 12:02:00	BD121A	9
EJ2034	2020-10-23	Calgary	Toronto	2020-10-23 08:15:00	2020-10-23 11:55:00	N973JM	3
EJ2122	2020-10-22	Vancouver	Toronto	2020-10-22 14:30:00	2020-10-22 18:40:00	J372LK	1
EJ2134	2020-10-22	Vancouver	Toronto	2020-10-22 09:00:00	2020-10-22 13:10:00	K372JP	2
EJ2425	2020-10-22	Ottawa	Toronto	2020-10-22 15:15:00	2020-10-22 16:30:00	J23HG2	8
EJ2518	2020-10-23	Toronto	Thunder Bay	2020-10-23 09:20:00	2020-10-23 11:15:00	J23HG2	8
SA3487	2020-10-23	Winnipeg	Thunder Bay	2020-10-23 14:30:00	2020-10-23 15:55:00	A198UU	6
SA3488	2020-10-23	Thunder Bay	Winnipeg	2020-10-23 17:00:00	2020-10-23 18:30:00	A198UU	6
WA4582	2020-10-23	Thunder Bay	Sioux Lookout	2020-10-23 11:30:00	2020-10-23 12:25:00	X32H1L	7

[Back To Start](#)

```
SELECT * FROM Flight;
```

T:

## FlightTicket

passportNumber	ticketId	seatType	seatNumber
929417213	AC1201	Economy	B03
975176880	AC1313	Business	J01
705872764	AC1313	Economy	D04
336524494	AC1313	Economy	J01
361054356	AC1314	Economy	B06
384175811	AC1314	First	G06
768437119	AC1567	First	J04
657075248	AC1567	First	J06
523864301	AC1874	Business	A05
348502853	EJ2034	Business	B02
585558727	EJ2034	First	B02
501872285	EJ2122	Economy	D02
917962210	EJ2122	Economy	D03
413816856	EJ2122	First	A01
570383268	EJ2122	First	B01
347338810	EJ2134	Business	C04
165912112	EJ2134	Economy	C04
183465870	EJ2134	First	E06
455423760	EJ2425	Economy	I06
611058959	EJ2518	Economy	C03
411858833	EJ2518	Economy	G04
323033958	SA3487	Economy	F03
226799985	SA3488	Economy	E05
229477478	SA3488	Economy	F02
431415524	WA4582	Economy	E01

[Back To Start](#)

```
SELECT * FROM FlightTicket;
```

U:

# FlightCrew

crewId
1
2
3
4
5
6
7
8
9
10

Back To Start

```
SELECT * FROM FlightCrew;
```