Airline Management System

Team Members (Name - Net ID):
- Prateek Kumar - pk2460
- Simran Shandilya - ss13890

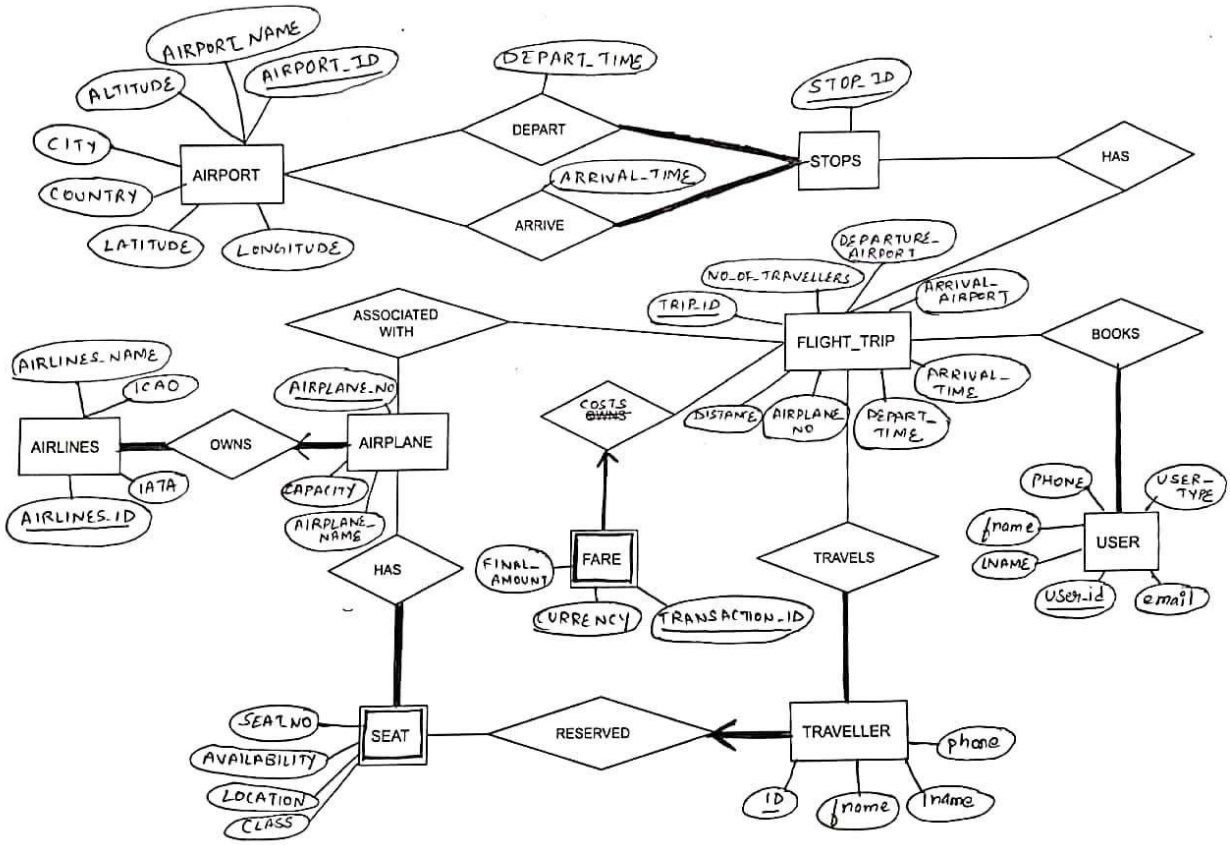
Project Deployment:

The project is deployed at Port:8632
Link: http://jedi.poly.edu:8632

Project Description:

The project is a small implementation of a widely used transaction-based processing system - the **Airline Reservation System**. It incorporates various functionalities like airline schedules, fare tariffs, passenger reservations, and ticket records, etc. We process different types of requests/transactions by implementing various database operations like insertions, selections, group by, join, order by, aggregate functions (eg: count), etc.

ER Diagram:



Entity Sets:

- Airport
- Airplane
- Airlines
- Seat
- Fare
- Flight_Trip
- Traveller
- User
- Stop

Relationship Sets:

- Access
- Owns
- Has
- Costs
- Travels
- Reserved
- Depart
- Arrive
- Has
- Books
- Associated With

Business Rules:

- Airlines own airplanes.
- Airplanes have seats.
- Travelers have their seats reserved.
- Travelers travel on flight trips.
- Flight trips have associated fares.
- Users book flight trips.
- Flight trips may have stops.
- Every airline company owns at least one airplane.
- Every airplane is owned by exactly one airline.
- Every airplane has seats.
- Every user has booked at least one trip.
- Every traveler has exactly one seat reserved.
- A traveler cannot travel without a flight trip.
- Every flight trip has a fare associated with it.
- Fare exists only if there is a flight trip.
- Each stop in the flight trip will have a departure and an arrival time.

- Every flight trip can have hops.

Data Source:
We are fetching data corresponding to flights, airports, and schedules from https://www.makemytrip.com/ The other aspects of information/data are mocked up by us.

Data Model and Limitations:

- The data shows flights operating from Delhi to Mumbai for a specific date.
- Every flight trip is associated with an airplane (uniquely identified by airplane number), and every airplane takes just one flight on a given day
- Every traveller pays the same fare for a given flight trip.
- Our dataset shows all the information for a specific date.
- A flight trip that has no hops is indicated by stop_id = 0, and it's corresponding depart and arrival time would be 0:00 (place holder)
- Our dataset has 2 kinds of users - Traveller and Agency:
  Traveller: The ones who are traveling on the flight which they have booked.
  Agency: The ones who are not traveling, but have booked tickets.

Application Features:

The application provides the user to get the following information:

- Traveler information
- Traveler Itinerary (including stops in their trip)
- The total number of flights that would operate corresponding to every airline
- The total number of bookings done by each user type - agency/traveller in each airline
- The total number of bookings done by each agency in each airline
- The total number of seats available for booking in each airplane.
- The total number of seats available for booking in each airplane, for each of the seat categories - aisle, window, and middle.
- Fare associated with every flight.

Schema:

```
CREATE TABLE AIRPORT (
   AIRPORT_ID integer PRIMARY KEY,
   AIRPORT_NAME varchar(128) NOT NULL,
   ALTITUDE integer NOT NULL,
   CITY varchar(64) NOT NULL,
   COUNTRY varchar(64) NOT NULL,
   LATITUDE integer NOT NULL,
   LONGITUDE integer NOT NULL
);

CREATE TABLE AIRPLANE_AIRLINE (
   AIRPLANE_NO integer PRIMARY KEY,
      CAPACITY integer NOT NULL,
   AIRPLANE_NAME varchar(128) NOT NULL,
   AIRLINE_ID integer UNIQUE NOT NULL,
   AIRNINES_NAME varchar(128) NOT NULL,
   AIRLINES_ICAO varchar(128) NOT NULL,
   AIRLINES_IATA varchar(128) NOT NULL
);

CREATE TABLE SEAT (
   SEAT_NO integer NOT NULL,
   AVAILABILITY boolean,
   LOCATION varchar(64) NOT NULL,
   CLASS varchar(64) NOT NULL,
   AIRPLANE_NO integer,
   PRIMARY KEY (AIRPLANE_NO, SEAT_NO),
   FOREIGN KEY (AIRPLANE_NO) REFERENCES AIRPLANE_AIRLINE (AIRPLANE_NO) ON
DELETE CASCADE
);

CREATE TABLE STOP (
   STOP_ID integer PRIMARY KEY,
   ARRIVAL_TIME time NOT NULL,
   DEPART_TIME time NOT NULL,
   AIRPORT_ID integer NOT NULL,
   FOREIGN KEY (AIRPORT_ID) REFERENCES AIRPORT (AIRPORT_ID)
);
```

```
CREATE TABLE FLIGHT_TRIP (
   TRIP_ID integer PRIMARY KEY,
   NO_OF_TRAVELLERS integer NOT NULL,
   DEPARTURE_AIRPORT varchar(128) NOT NULL,
   DEPART_TIME time NOT NULL,
   ARRIVAL_AIRPORT varchar(128) NOT NULL,
   ARRIVAL_TIME time NOT NULL,
   STOP_ID integer NOT NULL,
   DISTANCE integer NOT NULL,
   AIRPLANE_NO integer NOT NULL,
   FOREIGN KEY (STOP_ID) REFERENCES STOP (STOP_ID),
   FOREIGN KEY (AIRPLANE_NO) REFERENCES AIRPLANE_AIRLINE (AIRPLANE_NO)
);

CREATE TABLE FARE (
   TRIP_ID integer NOT NULL,
   TRANSACTION_ID integer NOT NULL,
   FINAL_AMOUNT integer NOT NULL,
   CURRENCY varchar(64) NOT NULL,
   PRIMARY KEY (TRIP_ID, TRANSACTION_ID),
   FOREIGN KEY (TRIP_ID) REFERENCES FLIGHT_TRIP (TRIP_ID) ON DELETE
CASCADE
);

CREATE TABLE TRAVELLER (
   ID integer,
   FNAME varchar(128) NOT NULL,
   LNAME varchar(128) NOT NULL,
   PHONE varchar(128) NOT NULL,
   TRIP_ID integer NOT NULL,
   PRIMARY KEY (ID, TRIP_ID),
   FOREIGN KEY (TRIP_ID) REFERENCES FLIGHT_TRIP (TRIP_ID) ON DELETE
CASCADE
);

CREATE TABLE USERS(
   EMAIL varchar(128) not null,
      USER_ID integer primary key,
   FNAME varchar(128),
```

```
   LNAME varchar(128),
   PHONE varchar(128),
   USER_TYPE varchar(128),
   TRIP_ID integer NOT NULL,
   FOREIGN KEY (TRIP_ID) REFERENCES FLIGHT_TRIP (TRIP_ID)
);
```

Data Loading Procedure:

In order to load data, we need to run the script **load_sql.sh** which has the following contents:

```
#!/bin/sh

cat ./AIRPORTS.csv | psql -U pk2460 -d pk2460_db -c "COPY airport from
STDIN CSV HEADER"
cat ./AIRPLANE_AIRLINE.csv | psql -U pk2460 -d pk2460_db -c "COPY
airplane_airline from STDIN CSV HEADER"
cat ./STOP.csv | psql -U pk2460 -d pk2460_db -c "COPY stop from STDIN CSV
HEADER"
cat ./FLIGHT_TRIP.csv | psql -U pk2460 -d pk2460_db -c "COPY flight_trip
from STDIN CSV HEADER"
cat ./SEAT.csv | psql -U pk2460 -d pk2460_db -c "COPY seat from STDIN CSV
HEADER"
cat ./FARE.csv | psql -U pk2460 -d pk2460_db -c "COPY fare from STDIN CSV
HEADER"
cat ./TRAVELLER.csv | psql -U pk2460 -d pk2460_db -c "COPY traveller from
STDIN CSV HEADER"
cat ./USERS.csv | psql -U pk2460 -d pk2460_db -c "COPY users from STDIN
CSV HEADER"
```