**Entities and Attributes**

Airline Company: The Airline Company collection will store the Company details like the name, address etc.

* + The Attributes present in this collection are:
    - Airline Name (airlineName)
    - Airline Registration Number (airlineRegistrationNo)
    - Company Address (companyAddress)
    - Company Mobile Number (companyMobileNo)
    - Email Id (emailId)
    - Total Number Of Employees (totalNumberOfEmployees)
    - Total Number Of Planes (totalNumberOfPlanes)

Airports: The collection, Airports, will have all the information regarding the Airport name, Airport Location etc.

* + The Attributes present in this Collection are:
    - Airport name (airportName)
    - Airport Location (airportLocation)
    - Cost per hour of the Airport (costp/h)
    - Other Charges like refueling and parking(otherCharges)

Pilots: The collection, Pilots, will store the data of all the pilots who would be flying the flights and will also give their flying test records.

* + The Attributes present in the collection are:
    - Pilot Id (pilotId)
    - Pilot Name (pName)
    - Pilot Address (pAddress)
    - Date of last fit for flying test (dateOfLastFitForFlyingTest).

Plane Flights: The collection, Plane Flights, will store the information of upcoming flights and daily schedule.

* + The Attributes present in the collection are:
    - Plane id (planeId)
    - Flight no (flightNo)
    - Starting point (startingPoint)
    - Final Destination (finalDest)
    - Departure date and time (departureDateAndTime)
    - Arrival Date and Time (arrivalDateAndTime)
    - Pilot id (pilotId)
    - Pilot Name (pilotName)

Employees: The collection, Employ Record, will consist of all the details of Employ working in the Airline Company.

* + The Attribtes Present in the collection are:
    - Employ id(empId)
    - Employ Name(EmpName)
    - Date of Joining (dateOfJoining)
    - Department (department)
    - Department ID (departmentId)
    - Reporting Manager (reportingManager)
    - Salary (salary)

Contact Details: The collection, Contact Details, is specially created to store the Employ Contact details and addresses.

* + The Attributes present in this collection are:
    - Employ id(empId)
    - Mobile Number (mobileNo)
    - Email id (emailId)
    - Employ Name (employName)
    - Employ Address (employAddress)

Flight Bookings: The collection, Flight Booking, will store all the information regarding a customer’s booking like booking id, booking date and time etc.

* + The Attributes present in this collection are:
    - Bookingid (bookingId)
    - Booking Date and Time (bookingDateAndTime)
    - Booking Done BY (bookingDoneBy)
    - Passenger on Booking (passengerOnBooking)
    - Flight Details (flightDetails)
    - Total Cost of Booking (totalCostOfBooking)

Planes: The collection, Planes, will store all the information about the planes owned by the Airline Company and their current operation status like whether they are running or not.

* + The Attributes present in this collection are:
    - Plane id (planeId)
    - Make (make)
    - Model (model)
    - Flying range (flyingRange)
    - Length of service (lengthOfService)
    - Status (status)
    - Seating Capacity (seatingCapacity)

Journeys: The collection, Journeys, will consist of data related to all the journeys carried out by the Airline Company.

* + The Attributes Present in this collection are:
    - Journey Id (journeyId)
    - Flight No (flightNo)
    - Starting Airport (startingAirport)
    - Destination Airport (destinationAirport)
    - Type of Flight (typeOfFlight)
    - Layover (layover)
    - Journey Length(KM) (journeyLength(KM))

Customer: The collection, Customer, is created to store all the Customer details.

* + The Attributes Present in this Collection are:
    - Customer Name(cName)
    - Phone Number (phoneNo)
    - Email ID (emailId)
    - Passport Number (passportNo)
    - Booking ID (bookingId)

**Relationships**

* The system has **One-to-Many** relation between Airline Company and Planes which implies that the airline company can own multiple planes but a plane can belong to only one company.
* The system has **One-to-Many** relation between Airline Company and pilots which implies that the airline company can own multiple pilots but a pilot can be employed in only one company.
* The system has **One-to-Many** relation between Airline Company and Employ Record which implies that the airline company can hire multiple employees but an employee will belong to only one company.
* The system has **One-to-Many** relation between Airline Company and Flight Bookings which implies that the airline company can have multiple bookings but a customer can have only one flight booking details.
* The system has **One-to-Many** relation between Airports and journeys which implies that the same journey path can be used for multiple flights by an airport but one flight will have a single journey plan.
* The system has **One-to-Many** relation between Airports and Plane Flights.
* The system has **One-to-Many** relation between Customer and Flight Bookings which implies that the customer can have multiple flight bookings but a particular flight booking can belong to only one customer.
* The system has **One-to-one** relation between Employ record and Pilots which implies that every pilot will be in employ record but all employ record will necessarily not have Pilots.
* The system has **One-to-one** relation between Pilots and Plane Flights which implies that all plane flights will have pilots but it is not necessary that all pilots have flight plans.
* The system has **One-to-one** relation between Customer and Flight Bookings which implies that all flight bookings will have customers but it is not necessary that all customers have flight booking.

1. A set of queries to extract information from the system. You should aim to use a good range of language constructs to demonstrate your understanding of the language.

* Once the Data is Populated in the database, you can view them all with the find () function. The usage of it given below:
  + db.collection\_name.find();
* This will give out all the documents present in the current collection. If you want to fetch the data with some constraints, you can specify them in the find() function like this:
  + db.collection\_name.find({<field\_name>: <parameter>});
* Note another way of showing the data is by giving the output in JSON format. We can do that by using the pretty() function
  + db.collection\_name.find().pretty();