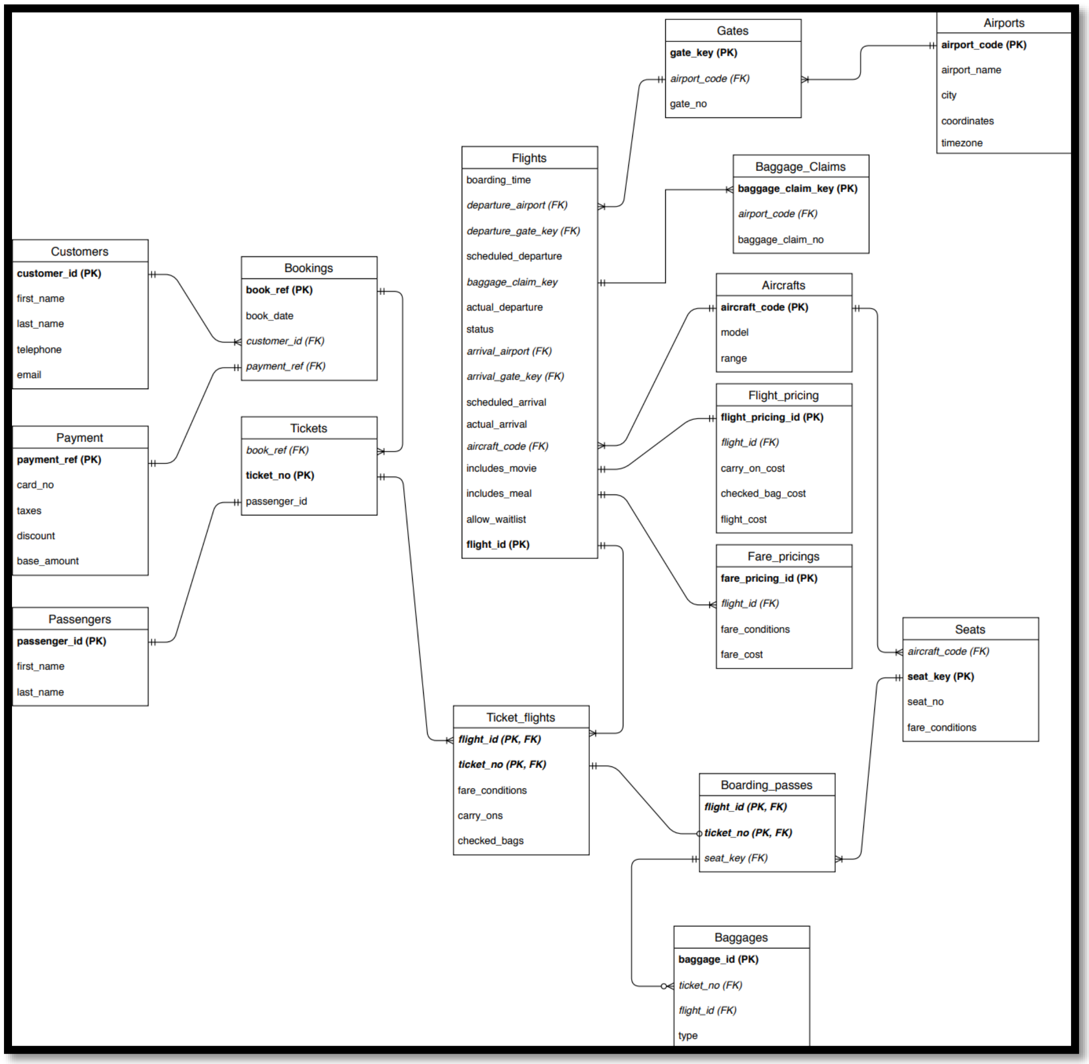
ER Model Description



Our ER model consists of 16 tables all linked together via various foreign keys to achieve normalization. Our demo first starts with the user first adding an airport to the database with its name, code, city, coordinates, and time zone in the **airports** table. The user is also asked to add gate(s) of the airport which are added into the **gates** table linked to the airports table via the airport code foreign key. The **baggage_claims** table, containing a unique key and claim number, is also linked via the airport code foreign key. This is done to comply with 1NF because all airports don't have the same number/classification-type of gates. The user is then prompted to input aircraft information including the code, model, and range into the **aircrafts** table along with the number of seats of each class of passengers and their respective prices into the **seats** table. The seats table consists of the seat key, number, fare conditions, price, and a foreign key to its respective aircraft code (in accordance to 1NF). Next, the user is prompted to add in a flight to the database which takes down a plethora of information (origin airport and gate, destination airport and gate, scheduled departure, arrivals, and boarding times, aircraft model (code found in aircrafts table), cost of flight, checked bags, and carry-on bags, and options for meal, movies, and waitlists) into the **flights** table. A flight ID generated in the flights table serves as a foreign key linked to the **ticket flights** table which also has the distinct ticket number of each passenger added by the user when creating a booking through the customer view. That same flight ID is also linked via foreign key to two other tables being **flight_pricing** which stores a unique pricing ID and costs of the flight and checked/carry on bags for the flight, as well as the **fare_pricings** table, which stores a unique pricing ID and the fare conditions and fare cost. This is done to comply with 3NF and eliminate partial/transitive dependencies. The ticket number is used to create an entry in the **boarding passes** table, along with the flight ID that has information of waitlisted fair conditions, if the booking is waitlisted and the seat key foreign key to the seats table. The ticket number from boarding passes serves as a foreign key to the **baggages** table that contains a baggage ID, flight ID foreign key and the baggage type. That same ticket number is also used to create an entry in the **tickets** tables that includes the passenger ID and booking reference (both foreign keys, book ref linked to the **bookings** table and passenger ID linked to the **passengers** table), and the amount of the carry-ons and checked bags they wish to select. This split is done for normalization purposes as multiple tickets can be created under one booking reference number. The bookings table has entries for the reference number as well as the date the booking was created and foreign keys to the **customers** table and **payment** table. The customers table has a unique ID for each customer that booked the tickets with their first and last names, phone # and email. The payment table consists of entries for the payment reference number as well as info including the card number, taxes, discount and base amount charged to the customer.