Report Summary

We tried to make the database and the website as realistic as possible. Though, some of the attributes were to hard to fully implement it so we simplify it and those will be explained in this document.

1. ER-diagram:
   - Below is the ER-diagram with 12 tables that satisfy most of the requirements. The database based on the bookings schema from homework 1 and the database from homework 3.

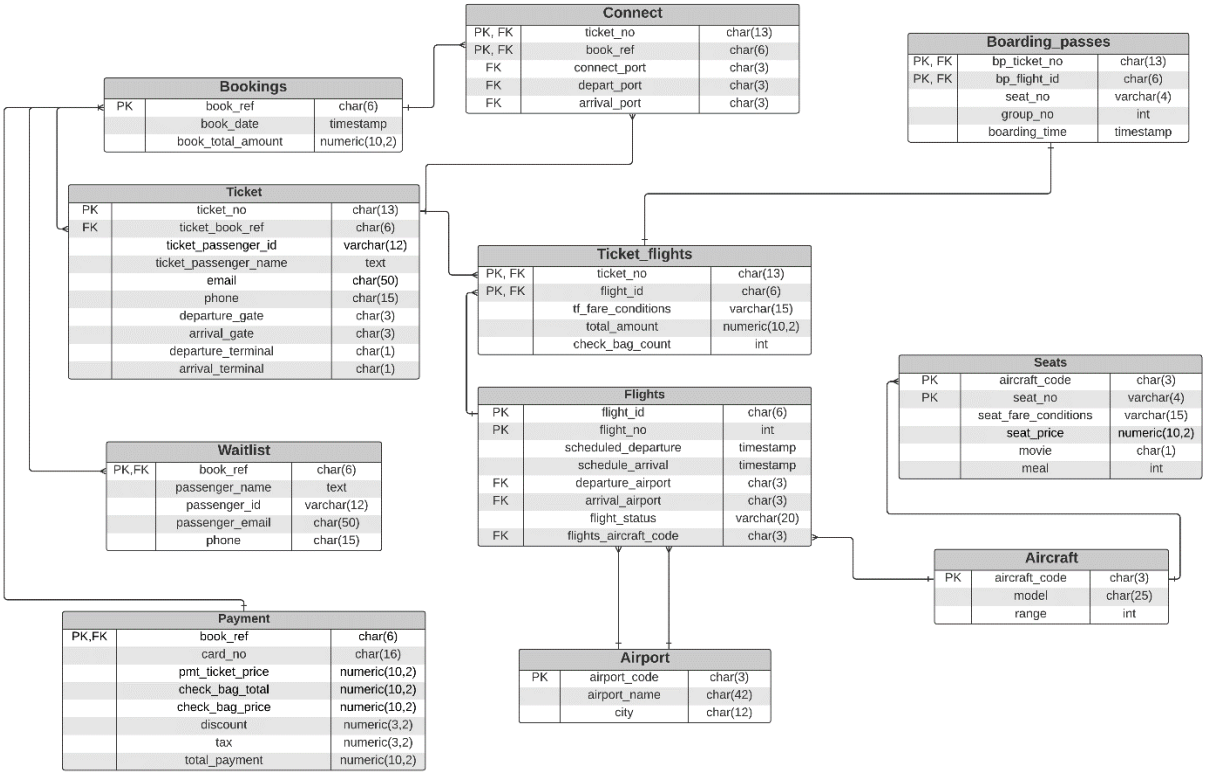

Figure 1. Original ER-Diagram

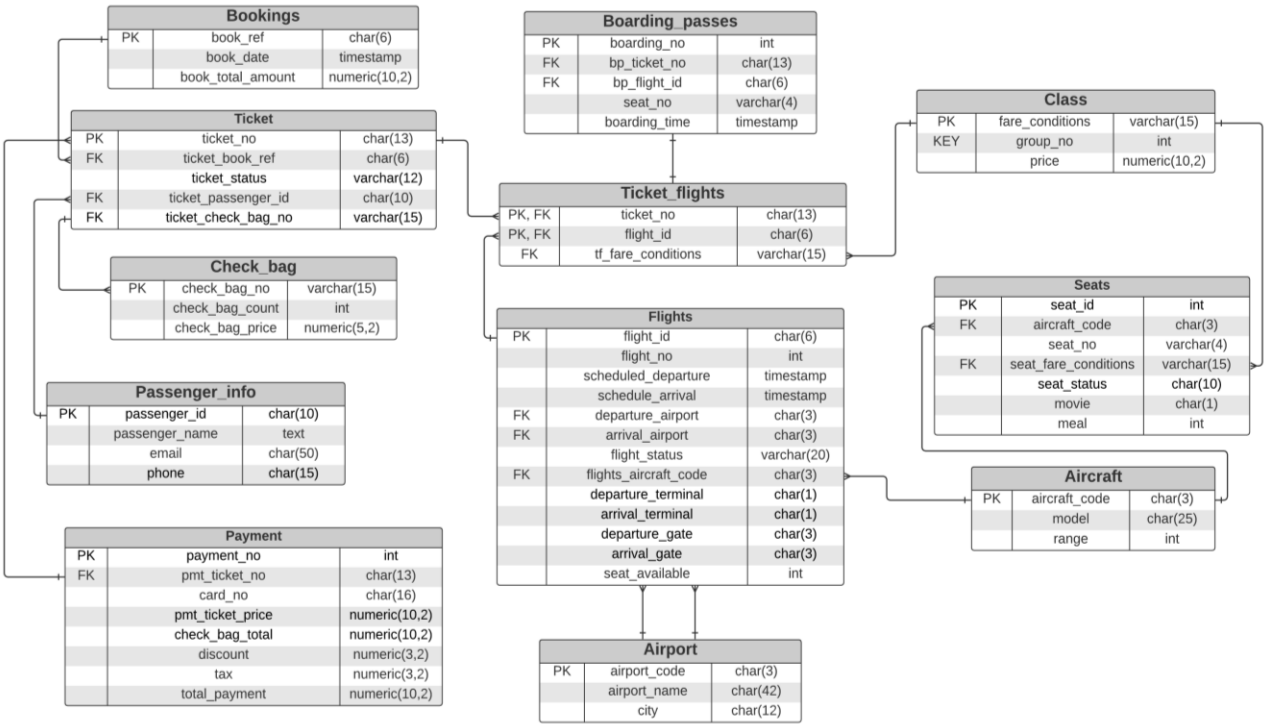

Figure 2. Final ER-Diagram

- Comparing to the original diagram, we have more table to satisfy the normalization requirement. Though, we could not get to 20 tables, but we for sure meet most of the extended requirements. We also move some of the attributes around to make a better reference.
- Almost all of the relationship cardinalities is 1:N, except for Boarding_passes and Ticket_flights is 1:1. Notice that most of the table only have 1 primary key except for the Ticket_flights table which has 2 primary keys (composite PK). It is design like that so help simplify the normalization requirement which it would achieve 3NF/BCNF. A NOT NULL constraint are applied to the database when creating the tables, so there will be no nulls in anycase. The database is designed to satisfy no nulls requirement. Some of the information such as book_ref, ticket_no, flight_id, flight_no, passenger_id, boarding_no, seat_no, and seat_id we choose the increment system to generate the information for those, so it is easier for us to keep up with the database.
- A short description of the database and the realationship would be: A customer booking ticket(s). Ticket(s) will have passerger information, check bag information, and payment information. An airport have many flight leaving from and coming to. A type of aircraft could have many flights. Ticket and flight made up many ticket flights and boarding passes is the result of each ticket flights. Each ticket flights have fare condition that a customer get to pick (in our database we provide Economy, Business, and First Class type). Each aircraft has many seats which define by class.

2. Initial database:
   - We initialize our database with 2 aircraft codes in out Aircraft table which is 773 and 763 based on the old hw.
   - To make it small we give 20 seats per aircraft. 4 seats for First Class type (2 rows, 2 seat each row), 6 seats for Business Class type (2 rows, 3 seats each), and 10 seats for Economy Class type (2 rows, 5 seats each). To simplize our program, we automatic assign seat number to the customer, so we can keep track of what seat already ‘Booked’ or still ‘Available’. We made the seats limited to hope that we can show the waitlist system is working. There is no limit amount of people on the waitlist for in our case.
   - For Seats table, we set all seat_status to ‘Available’, yes for movie, and 1 meal for all seats. We assign $1200 for First Class, $500 for Business Class (whenever a customer choose this type seat, they will get 10% discount so that we can show out discount system work), and $150 for Economy Class. We also charge $50 for every check bag that the customer want to added to the ticket. Tax is the same for all tickets which is 8% per ticket price. So the total amount the customer had to pay for each ticket is calculated by *ticket price* +

*(check bag count * 50.00) + (ticket price * 0.08) – discount.* This is for 1 ticket only, in the case that a customer book many ticket, then we can account all the ticket in the same book reference, calculate total price for each ticket and add them up for final total price.

- For airports and cities we chose IAH-Houston, DFW-Dallas, LAX-Los Angeles, JFK-New York, MCO-Orlando, and MDW-Chicago to work with.
- Right now in the database we initial it with only 20 flights but the time span is for 2 months (from Dec 10 – Feb 10) so the customer (which is the team that test us had to book the right time, the right departure and arrival city). So mostly there is only one flight for a customer to choose for a specific from/to location and date, and satisfy customer inquires. We share the excel sheet link in README to show all the options for booking a flight ticket.
- In the flight table it is initialize with all the 20 flights that customer can book. On the flight status since it is for the future the status always be 'Schedule'. For connecting flight, we will do a join and recursive query to show flight that connecting. In our database, only 0 connection (direct) flight or 1 connection (1 stop) flight are the options. In the initial database we have the forward and reverse flights for customer who wants to book round trip, but the user would have to book 2 separate flights (new booking transaction) on the different dates.
- Those are the tables that we initialized. The other tables, we will update and insert it during the booking process. We basically get all of the extending requirements. Unfortunately, we did get the cancellation working, so people on the waitlist can't really have a chance to book the ticket they want. Since the canellation system is not working, so the refund is not applicable here either.

3. GUI:
   - We show the message to the customer whenever they are successfully book a ticker or if they have to be put in waitlist. It will show on the ticket page. We don't have message if customer enter wrong information or missing information. Though, if the customer miss entering any information, they still be direct to the ticket page but on the incorrect information will be show on the boarding pass.
   - Every time someone need to use or test our webpage, they need to run **./i make_airline_sql.sql** in **postgres**. That will automatically reset and initialize all the tables.
   - We tried to make the booking ticket flight website to be as realistic as possible so there is for sure that customer does not have to deal with and SQL part. Also, we tested and it work on Chrome and Microsoft Edge.
   - There are 4 different pages for our booking flight ticket website. First the customer start on the ***airlineweb*** page where the customer gets to choose how many passenger they want to book ticket for, seat class, start and final destination, and departure date. They done have to type anything, we have the drop down menu with the options which to make sure they don't do anything invalid. When they satisfy, they will click *'Select Flights'* and it will direct them to the ***flights*** page. If there is any flight available for their requirement that they put, it will appear on this page showing all the options with 0/1 connections, the departure date, time and location for start and to destination. **Notice: if there is a blank page then there is no flight available to satisfy all of their requirement, they need to choose again, for that reason we are providing the specific flights customer should book on our README to test our program.* If the customer satisfies with this, then they will click *'Book'* which direct them to ***booking*** page. On that page the customer will need to enter name, phone, email for all the people that book the tickets. Enter check bag, discount code (which did not work if they enter a code, but as mentioned above, the discount will automatically apply if the customer book Business Class type seat), and the card number for payment. The calculation of total price a customer has to pay will appear at the end of the page. When the customer done with that, click *'Book'* then they will be direct to ***ticket*** page which show them all the ticket information such as flight id, flight number, destination, departure date, name of the passenger, etc. Under the ticket, we also have the option if customer want to book another flight then they will click *'Book Another Flight'* it will redirect them to the ***airlineweb*** page. If there are more than 1 person traveling together each boarding pass for each customer will show in the ticket page. Same for connection flights, there will be each boarding pass show for each flight.

4. Queries on the clerk side:
   - Listing passenger on each seat of a flight (in this case seat number is 05A, and flight id is TG0010):
     SELECT passenger_name

```
FROM passenger_info
WHERE passenger_id = (
  SELECT ticket_passenger_id
  FROM ticket
  WHERE ticket_no = (
    SELECT bp_ticket_no
    FROM boarding_passes
    WHERE seat_no = '05A' AND bp_flight_id = 'TG0010')
  );
```

- Listing people on waitlist:

```
SELECT passenger_name
FROM passenger_info
INNER JOIN ticket
ON passenger_id = ticket_passenger_id AND ticket_status = 'Waitlist';
```

- Listing the people that are traveling together:

```
SELECT ticket_passenger_id INTO TABLE temp
FROM ticket
WHERE ticket_book_ref IN
(SELECT ticket_book_ref
FROM ticket
GROUP BY ticket_book_ref
HAVING COUNT(*) >= 2);

SELECT passenger_name
FROM passenger_info
INNER JOIN temp
ON passenger_id = ticket_passenger_id;

DROP TABLE temp;
```

- Listing available seats for each aircraft (in this case aircraft number is 773):
-

```
SELECT ticket_passenger_id INTO TABLE temp
SELECT seat_no
FROM seats
WHERE seat_status = 'Available' AND aircraft_code = '773';
```

- Show the price of each class and number of seats for each class:

```
SELECT DISTINCT fare_conditions, price INTO TABLE temp
FROM class
INNER JOIN
seats ON fare_conditions = seat_fare_conditions;

SELECT DISTINCT fare_conditions, price, COUNT(seat_no)
FROM seats
INNER JOIN temp
ON seat_fare_conditions = fare_conditions
GROUP BY fare_conditions, price;

DROP TABLE temp;
```

- Show how many check bag per passenger has:

```
SELECT ticket_passenger_id, check_bag_count INTO TABLE temp
FROM ticket
INNER JOIN check_bag
ON ticket_check_bag_no = check_bag_no;
```

```
SELECT passenger_name, check_bag_count
FROM passenger_info
INNER JOIN temp
ON passenger_id = ticket_passenger_id;

DROP TABLE temp;
```

- The price per ticket(without tax) a person has to pay:

```
SELECT ticket_passenger_id, pmt_ticket_price INTO TABLE temp
FROM payment
INNER JOIN ticket
ON pmt_ticket_no = ticket_no;

SELECT passenger_name, pmt_ticket_price
FROM passenger_info
INNER JOIN temp
ON passenger_id = ticket_passenger_id;

DROP TABLE temp;
```

- Show if the customer get the discount:

```
SELECT ticket_passenger_id, discount INTO TABLE temp
FROM payment
INNER JOIN ticket
ON pmt_ticket_no = ticket_no;

SELECT passenger_name, discount
FROM passenger_info
INNER JOIN temp
ON passenger_id = ticket_passenger_id;

DROP TABLE temp;
```