Database Systems Report

Asrorbek Orzikulov, Ying Ding, Meduri Venkata Shivaditya

About the Database

The objective of this project is to create a database for a travel reservation system.

The various use cases of the travel reservation system are:

- Customer Registration – Worldwide
- Customers can book a trip consisting of at least 1 flight booking and any number of hotel bookings
- Flights connecting 2 airports are operated by airlines and take place on aircrafts should be stored in the system
- A flight connecting two airports at specific departure and arrival times is identified by a flight number. Two flights operated by two different airline companies cannot have the same flight number, but the same flight number can denote two flights operated by the same airline company on different days.
- For each flight booked by a customer, the system keeps the seat number, the travel class (e.g., economy or business), the price and the date of the flight
- The system maintains a list of hotels, with their names, addresses and an average review score
- When a customer books an hotel, the system keeps the price paid, the check-in and check-out dates and whether the breakfast is included.
- Customers can write a review for an hotel; in which case the system stores the text of the review, the date and its author

About the data

The dataset consists of 7 CSV files:

- aircrafts.csv
- airlines.csv
- airports.csv
- hotels.csv
- customers.csv
- hotel_bookings.csv
- flight_bookings.csv

Software Used

- SQLite – Embedded DBMS to create the database
- Python – Pre-process the data

Questions regarding the conceptual schema

a. Can you use the name of the hotel as a primary key? Justify your answer.

No, we cannot use the hotel name as a primary key because there is a possibility that 2 (or more) hotels have the same name. Even if we assume that 2 hotels cannot have the same name, using the hotel name as a primary key is very inefficient because of the large overhead of string comparisons. Using hotel ID as a primary key is a better option**.**

b. Can you use the flight number as a primary key to identify a flight? Justify your answer and, in case of a negative answer, propose a solution.

The flight number can uniquely identify neither one individual air travel nor a set of air travels that take off and land at the same time. The reason is the same flight number can be used on several dates and between different pairs of airports. Therefore, the flight number should be accompanied by departure date to identify one individual air travel, or it can be combined with source and destination airports and time information to identify a set of flights (air travels). One more solution could be creating a flight id that uniquely identifies a connection between two specific airports at specific departure and arrival times.

c. Knowing that it is unlikely that two reviews have the same textual content, would you use it as a primary key? Justify your answer.

No, even if it is unlikely that 2 reviews have the same contextual content, it is not ideal to use it as a primary key. The first reason is short reviews have a very high chance of having the same textual content. For example, “Good”, ”Bad”, etc. The second reason is string comparisons to identify a particular review among other reviews can be costly because string comparisons are done character-wise.

d. Knowing that the IATA code uniquely identifies an airport, would you choose it as a primary key for the entity Airport? Justify your answer.

No, airport IATA code cannot be used to identify an airport because a few airports can have null IATA values. (Their IATA codes might be unavailable.) It is necessary to have all the values of a column to use it as a primary key. A safe alternative would be to use the airport id.

Based on the project description and datasets, we have made several assumptions to validate our ER diagram and arrived at the following diagram.

Notes:

Arrival time – we created airport time zones from their longitude and latitudes. Then, we joined the departure_date with departure_time to create a datetime departure_date object. After adding flight_duration to it, we created arrival_time in the source airport's time zone. Finally, this column was converted it to arrival_time in the destination airport's time zone, accounting for time zone differences and daylight saving schemes.

Flight schedule – it denotes one instance of air travel (on a specific date) that a specific airline company offers under a specific flight number. This entity is thus identified by flight_number and departure_date.

Flight_id – an identifier for the unique combinations of airport_src, airport_dst, departure_time, and arrival time. For the flight_id, different flight_ids might have same departure_time, arrival_time, flight_duration, while in nature they are different because those attributes are connected with different airports. Therefore, we assigned different flight_id to them. A similar case in real life will be different user_ids to the same names because they are referring to different people.

Aircraft_iata – In the process of data processing, it is noticed that the different aircraft_iata have identical information. After consultation with professor, we assume it as abnormal data. Therefore, we deleted the aircarft ID and removed the rows with same information. In the end, we use aircraft_iata as the primary key, which can identify the rest of columns.

Aircraft_icao, airline_iata, airline_icao – these columns should uniquely identify the rows they belong to. However, it was noticed that the same values of these attributes correspond to different observations. Therefore, they do not appear in functional dependencies as candidate keys, and they do not have UNIQUE constraints in the physical schema.

Flight_booking csv file – From the flight booking csv file, we have identified inconsistency observations. Some rows include same flight number, same departure date and time but different arrival time and destinations. In real life, it is impossible to see a flight flying to different places at the same time. Therefore, it is assumed to be abnormal data and 11 observations have been deleted.

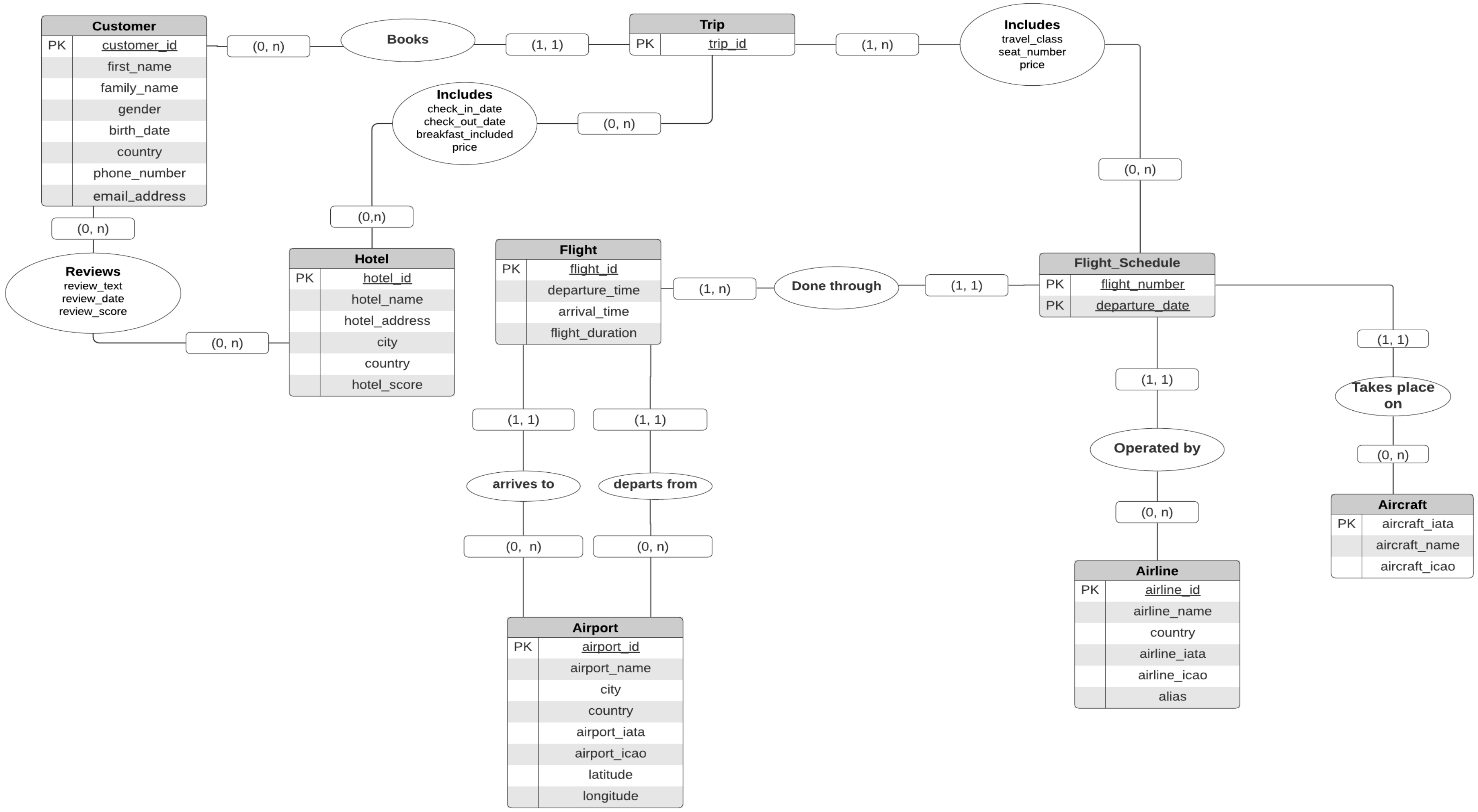
Flight – the project description says that flight_number identifies a flight that connects two airports at specific departure and arrival times. Thus, we tried to identify airport_src and airport_dst using flight_number alone and then using flight_number with different combinations of departure_time, arrival_time, and flight_duration. However, even flight_number, departure_time, arrival_time, flight_duration used together failed to identify source and destination airports. Therefore, we considered this functional dependency invalid and created flight_id that uniquely identifies a flight between two specific airports at specific departure and arrival times.

Failure to impose a foreign key constraint – we were not able to impose a foreign key constraint to the flight_number column of the flight_airline table since the primary key of the Flight_Schedule table consists of two columns (flight_number and departure_date). We do understand that this problem can bring inconsistency in our database since the flight_airline table is now disconnected from Flight_Schedule; however, we were not able to impose an FK constraint in SQLite.

Question 1 ER diagram

ER diagram for the Database

Ying Ding, Asrorbek Orzikulov, Meduri Venkata Shivaditya



Selected Cardinalities

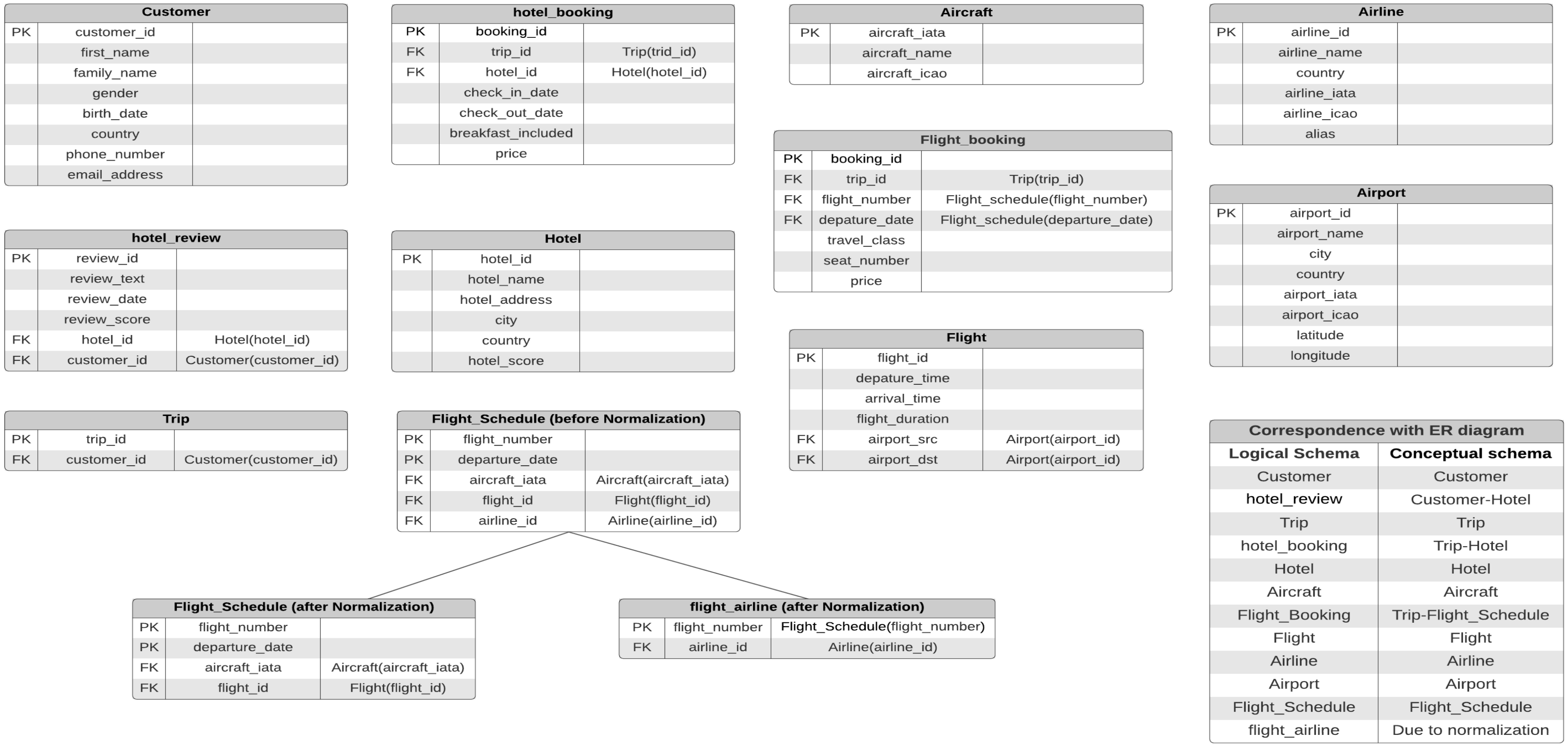
Related Entities	Side	Explanation
Customer-Trip	Trip	A trip_id uniquely identifies 1 particular trip made by 1 particular customer. We don’t create a trip_id if it is not attached to any customer, and a particular trip_id is given to 1 customer only.
Customer-Hotel	Customer	A customer can review none or several hotels.
Customer-Hotel	Hotel	A hotel may have none or several reviews.
Trip-Hotel	Hotel	A hotel can be a part of several trips, or it might not be chosen by any customer.
Trip-Flight Schedule	Flight Schedule	There might be some scheduled flights that are not booked by any customer. Conversely, several customers may book the same flight departing on the same date.
Flight-Flight Schedule	Flight Schedule	One specific instance of air travel can have one and only one departure and arrival times, duration, source and destination airports.
Flight-Flight Schedule	Flight	Lower cardinality: There cannot be a flight that has never been flown (with zero scheduled air travels). Upper cardinality: Since flight_number fails to uniquely identify airport_src and airport_dst, we explicitly allowed for the following: Two airports can be connected by two different flight_numbers at the same departure and arrival times.
Flight-Airport	Flight	By construction, flight_id denotes a flight from one airport to another at specific times. So, the cardinality is (1, 1) for both source and destination.
Flight Schedule -Aircraft	Flight Schedule	One specific instance of air travel (on a specific date) must take place on one and only one aircraft. Sometimes, an airline company may not provide aircraft information, but this practical problem is not relevant on a conceptual level.
Airport-Flight Airline-Flight Schedule Aircraft-Flight Schedule	Airport Airline Aircraft	Lower cardinality: A travel agency might store information on some unused-in-trips airports, airline companies, and aircrafts for future use.

Question 2 Logical Schema

From the conceptual schema to the logical schema, the entities become tables with the same name. For the (1,n) or (0,n) relationship, we inserted foreign keys to the tables with lower cardinalities (e.g. foreign key customer to the Trip table). For many to many relationships, we have created new tables for the relationships (e.g Flight_booking table).

Logical Schema for the Database

Ying Ding, Asrorbek Orzikulov, Meduri Venkata Shivaditya



Customer

PK	customer_id	
	first_name	
	family_name	
	gender	
	birth_date	
	country	
	phone_number	
	email_address	

hotel_booking

PK	booking_id	
FK	trip_id	Trip(trid_id)
FK	hotel_id	Hotel(hotel_id)
	check_in_date	
	check_out_date	
	breakfast_included	
	price	

Aircraft

PK	aircraft_iata	
	aircraft_name	
	aircraft_icao	

Airline

PK	airline_id	
	airline_name	
	country	
	airline_iata	
	airline_icao	
	alias	

Flight_booking

PK	booking_id	
FK	trip_id	Trip(trip_id)
FK	flight_number	Flight_schedule(flight_number)
FK	depature_date	Flight_schedule(departure_date)
	travel_class	
	seat_number	
	price	

Airport

PK	airport_id	
	airport_name	
	city	
	country	
	airport_iata	
	airport_icao	
	latitude	
	longitude	

hotel_review

PK	review_id	
	review_text	
	review_date	
	review_score	
FK	hotel_id	Hotel(hotel_id)
FK	customer_id	Customer(customer_id)

Hotel

PK	hotel_id	
	hotel_name	
	hotel_address	
	city	
	country	
	hotel_score	

Flight

PK	flight_id	
	depature_time	
	arrival_time	
	flight_duration	
FK	airport_src	Airport(airport_id)
FK	airport_dst	Airport(airport_id)

Trip

PK	trip_id	
FK	customer_id	Customer(customer_id)

Flight_Schedule (before Normalization)

PK	flight_number	
PK	departure_date	
FK	aircraft_iata	Aircraft(aircraft_iata)
FK	flight_id	Flight(flight_id)
FK	airline_id	Airline(airline_id)

Flight_Schedule (after Normalization)

PK	flight_number	
PK	departure_date	
FK	aircraft_iata	Aircraft(aircraft_iata)
FK	flight_id	Flight(flight_id)

flight_airline (after Normalization)

PK	flight_number	Flight_Schedule(flight_number)
FK	airline_id	Airline(airline_id)

Correspondence with ER diagram

Logical Schema	Conceptual schema
Customer	Customer
hotel_review	Customer-Hotel
Trip	Trip
hotel_booking	Trip-Hotel
Hotel	Hotel
Aircraft	Aircraft
Flight_Booking	Trip-Flight_Schedule
Flight	Flight
Airline	Airline
Airport	Airport
Flight_Schedule	Flight_Schedule
flight_airline	Due to normalization

Q3 and Q4 Tables, Functional dependencies and Normal form

Trivial functional dependencies (primary keys implying columns that comprise those primary keys) are omitted.

Customer (BCNF)

customer_id → first_name
customer_id → family_name
customer_id → gender
customer_id → birth_date
customer_id → country
customer_id → phone_number
customer_id → email_address

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys – Boyce-Codd NF ✔

Trip (BCNF)

trip_id → customer_id

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Hotel booking (BCNF)

booking_id → trip_id
booking_id → hotel_id
booking_id → check_in_date
booking_id → check_out_date
booking_id → breakfast_included
booking_id → price

We assumed even the unlikeliest situations that can occur in order to avoid future problems. For example, trip_id, hotel_id, check_in_date, check_out_date will not imply all the columns because a customer can check in and check out and then again check in and check out in the same hotel on the same day.

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Hotel (BCNF)

hotel_id → hotel_name
hotel_id → hotel_address
hotel_id → city
hotel_id → country
hotel_id → hotel_score

We admit that the hotel address is a composite field, but it is assumed to be an atomic value in this database because all addresses are not in the same format.

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Hotel_review (BCNF)

review_id → review_text
review_id → review_date
review_id → review_score
review_id → hotel_id
review_id → customer_id

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Airline (BCNF)

airline_id → airline_name
airline_id → country
airline_id → airline_iata
airline_id → airline_icao
airline_id → alias

ASSUMPTION: More than 1 airline company can have the same alias.

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Airport (BCNF)

airport_id → airport_name
airport_id → city
airport_id → country
airport_id → airport_iata
airport_id → airport_icao
airport_id → latitude
airport_id → longitude
latitude, longitude → airport_id

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Aircraft (BCNF)

aircraft_iata → aircraft_name
aircraft_iata → aircraft_icao

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Flight booking (BCNF)

booking_id → trip_id
booking_id → flight_number
booking_id → departure_date
booking_id → travel_class
booking_id → seat_number
booking_id → price
(flight_number, departure_date, seat_number) → booking_id
(flight_number, departure_date, trip_id) → booking_id

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Flight Schedule (1NF)

Flight_number, departure_date → aircraft_iata
Flight_number, departure_date → flight_id
Flight_number → airline_id (Partial dependency)

A primary key is present; no lists and no repeated rows. – 1NF ✔

Partial dependency exists – 2NF X

To convert it into the BCNF, the table is split into Flight_Schedule and flight_airline

Flight Schedule (BCNF)

Flight_number, departure_date → aircraft_iata
Flight_number, departure_date → flight_id

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Flight Airline (BCNF)

Flight_number → airline_id

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Flight (BCNF)

flight_id → departure_time
flight_id → arrival_time
flight_id → flight_duration
flight_id → airport_src
flight_id → airport_dst
airport_src, airport_dst, departure_time, arrival_time → flight_id
The last functional dependency holds by construction.

A primary key is present; no lists and no repeated rows. – 1NF ✔

No partial dependency as PK consists of only 1 col – 2NF ✔

No non-prime columns imply other non-prime columns– 3NF ✔

All determinants are super keys– Boyce-Codd NF ✔

Question 5 & 6:

Please refer to the db file.

Question 7:

Please refer to the sql file for the queries.

For query 13, we assumed that the destination ‘Paris’ refers to Paris, France since we noticed **another** country (the USA) has a city with the same name. Since there is no trip made to another Paris, it is assumed that it refers to Paris, France.

Question 8:

1. Write a query to get all the information of a customer with a given family name. Run the query multiple times and note the average running time of the query.

We have selected the family name 'BELLARD' to create the query. For the codes, they have been included in the sql file:

Number of running	Time
1	15ms
2	18ms
3	12ms
4	12ms
5	14ms
6	13ms
Average runtime	**13ms**

2. Create an index on the column containing the family name of a customer.

CREATE INDEX my_index ON Customer(family_name)

3. Rerun the same query multiple times and note the average running times

After creating the index,

Number of running	Time
1	10ms
2	12ms
3	10ms
4	10ms
5	8ms
6	11ms
Average runtime	**10.2ms**

4. Do you observe any difference? Can you explain what is going on here?

After creating an index, the average running time decreased as shown above. The reason is that an index makes it efficient to find the rows that have some specific values for those columns. The runtime decreased by around 3ms, a 22% increase in speed. Indexes are key to speed up queries by creating pointers to where data is stored within a database. They allow us to create sorted lists without having to create new sorted tables. Therefore, the search engine will speed up without having to check every row of the table.