**Group 15 - Assignment - Travel Agency Database**

**Exercise 1**

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

No, you cannot use the name of the hotel as a primary as two hotels might have the same name and this would break the key constraint.

1. **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.**

No, you cannot use the flight number as a primary key because an airline might have different flights with the same number on different days, which would fail to respect the key constraint.

Therefore, we can think of two possible solutions. On the one hand, we could use the flight number with the date as a composite primary key to identify a specific flight. That would be possible as a flight number may identify two flights with the same flight number but they won’t have similar dates. On the other hand, we could simply add a flight\_id to the Flight entity that would be a unique identifier for each flight.

To keep things simple later in the operation of the database and retrieval of data, we chose the latter solution.

1. **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, we cannot use the review content as a primary key for the reviews. That the content of two different reviews is unlikely to be identical doesn’t mean that it is impossible to find identical reviews, which would break the integrity rule. Also, one could even argue that short reviews (such as “Great staff”, “Great stay”, etc.) could occur regularly (this would have to be proven empirically).

Additionally, as a review has a high likelihood of being longer than 8 characters and therefore is heavier than 8 bytes (which is the size of BIGINT, the biggest possible INTEGER type)([Source - Size of BIGINT](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html)), it would likely cause great inconvenience in the operation of the database.

For all those reasons, we wouldn’t use the textual content of the review as a primary key.

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

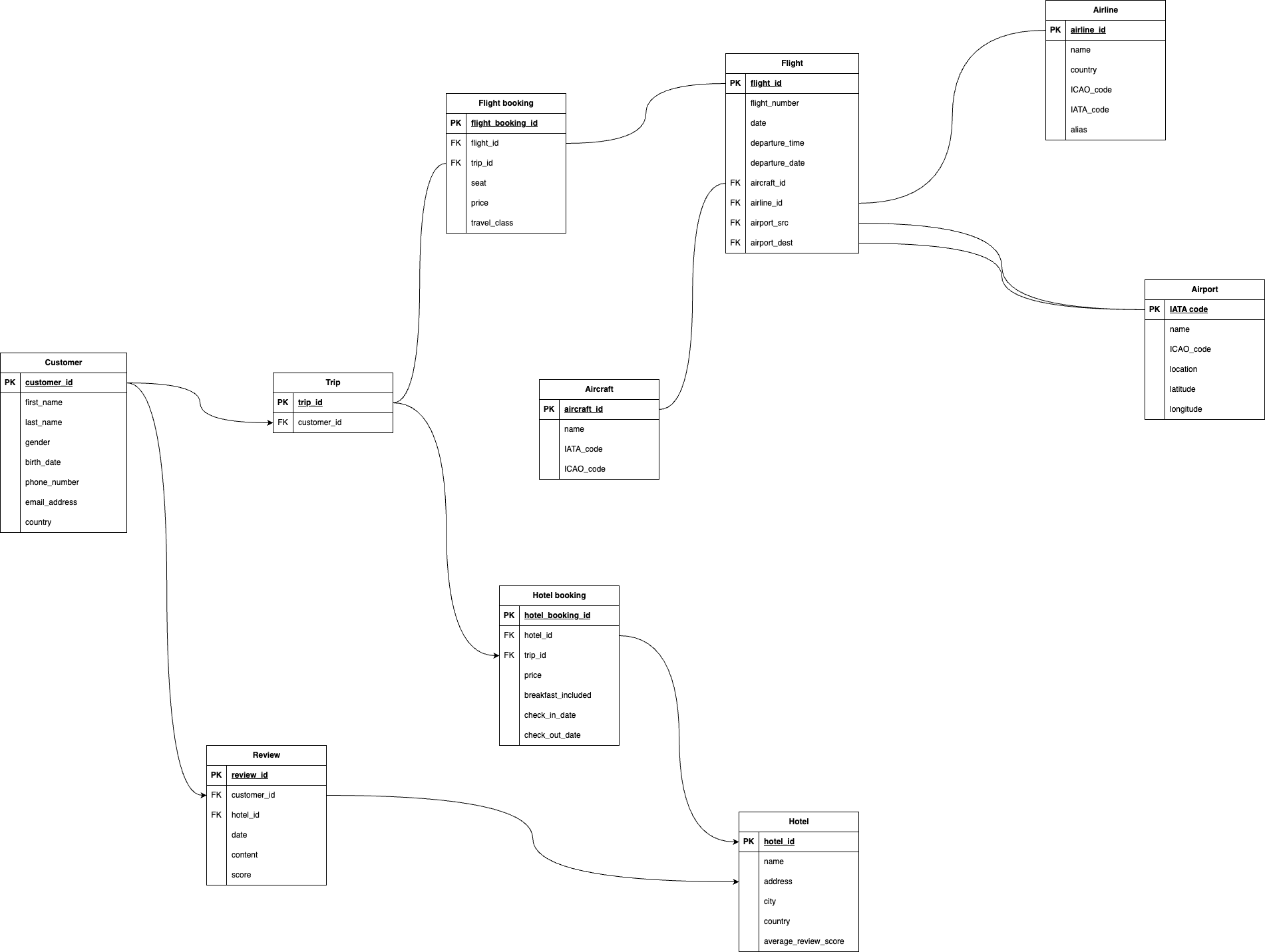
Given that the IATA code is a unique identifier, yes it is possible to use it as a primary key.

Additionally, there are a couple of reasons why we would prefer the IATA code to a unique numeric identifier (such as “airport\_id”).

Firstly, the code may carry a meaning to the end user which will make it easier for any person who uses the database as opposed to a numeric ID which carries very little to no meaning.

Secondly, adding an airport ID may introduce a redundancy in the data. Indeed, this would add very little value to our data as we already have a unique identifier at our disposal.

Finally, one could argue that a numeric value would be more memory-efficient than a string value. However, as the IATA codes have a fixed length of 3 characters ([Source - IATA code length](https://unitingaviation.com/news/general-interest/you-see-airport-codes-every-time-you-travel-but-do-you-know-the-story-behind-them/)), and each CHAR takes one byte of memory ([Source - Size of CHAR](https://dev.mysql.com/doc/refman/8.0/en/char.html)) (thus 3 bytes) versus 4 bytes for a regular INT ([Source - Size of INT](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html)), the advantages from a memory perspective would be negligible.



**Customer (I)**

Customer (customer\_id, first\_name, last\_name, gender, birth\_date, phone\_number, email\_address, country)

Functional dependencies

* {customer\_id} -> {first\_name, last\_name, gender, birth\_date, phone\_number, email\_address, country}
* {phone\_number} -> {customer\_id, first\_name, last\_name, gender, birth\_date, email\_address, country}
* {email\_address} -> {customer\_id, first\_name, last\_name, gender, birth\_date, phone\_number, country}

Even though one could argue that a home phone number could be used by several customers living at one address, most present-day websites only allow phone numbers to be registered once as they use them for account recovery through SMS which is not suitable for shared phone numbers (such as a home phone). Therefore, {phone\_number} is a valid candidate key.

Candidate keys

* {customer\_id}
* {phone\_number}
* {email\_address}

Because {phone\_number} and {email\_address} are long, and they might be changed by the users, it’s better to choose {customer\_id} for the primary key. Furthermore, {customer\_id} can be designed to contain useful information for it to be readable by database administrators. For example, it can reveal the gender of the customer by using an F or M in the identifier.

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (first\_name, last\_name, gender, birth\_date, country) are dependent functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ | Is in 2NF and there are no dependencies between non-prime columns. | |

**Trip**

Trip (trip\_id, customer\_id)

Functional dependencies

* {trip\_id} → {customer\_id}

Candidate keys

* trip\_id

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (customer\_id) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ | Is in 2NF and there are no dependencies between non-prime columns (there is only one non-prime column). | |

**Flight booking(I)**

Flight\_booking (flight\_booking\_id, flight\_id, trip\_id, seat, price, travel\_class)

Functional dependencies

* {flight\_booking\_id} → {flight\_id, trip\_id, seat, price, travel\_class}
* {flight\_id, trip\_id} → {flight\_booking\_id, seat, price, travel\_class}
* {flight\_id, seat} → {trip\_id, flight\_booking\_id, price, travel\_class}

As opposed to hotel bookings, there can’t be several flight bookings on the same flight for a given trip. Therefore {flight\_id, trip\_id} is a candidate key. As the foreign keys of this entity, the composition of them is surely a candidate key. Additionally, a seat can only be attributed to a single flight booking, therefore {flight\_id, seat} is also a candidate key. Lastly, {flight\_booking\_id} was purposely made as the primary key, it’s obvious that it implies all the keys and is one of the candidate keys.

Candidate keys

* {flight\_booking\_id}
* {flight\_id, trip\_id}
* {flight\_id, seat}

For the choice of primary key, to avoid composite primary key, we just simply choose {flight\_booking\_id} as the primary key.

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (price, travel\_class) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ | Is in 2NF and there are no dependencies between non-prime columns. | |

**Flight(S)**

Content

Flight (flight\_id, flight\_number, flight\_duration, departure\_time, departure\_date, aircraft\_id, airline\_id, airport\_src, airport\_dest)

Functional dependencies

* {flight\_id} → {flight\_number, flight\_duration, departure\_time, departure\_date, aircraft\_id, airline\_id, airport\_src, airport\_dest}
* {flight\_number, departure\_date} → {flight\_id, flight\_duration, departure\_time, aircraft\_id, airline\_id, airport\_src, airport\_dest}

Candidate keys

* {flight\_id}
* {flight\_number, departure\_date}

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (flight\_duration, departure\_time, departure\_date, airline\_id, airport\_src, airport\_dest) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ |  | |

Here we have the combination of two keys {flight\_id, {flight\_number, departure\_date} as the candidate key. All the other non prime columns are functionally dependent on all columns of the candidate key. Also the flight number and departure date more specifically denotes and it is not possible to have same flight number which departs at the same departure time from the source airport from the same airline.

**Aircraft(I)**

Aircraft (aircraft\_id, name, IATA\_code, ICAO code)

Functional dependencies

* {aircraft\_id} -> {name, IATA\_code, ICAO\_code}
* {IATA\_code} -> {aircraft\_id, name, ICAO\_code}
* {ICAO\_code} -> {aircraft\_id, name, IATA\_code}

According to the explanation of the {IATA\_code} and {ICAO\_code}, they are unique. Therefore, they are all candidate keys. {aircraft\_id} was purposely made as the primary key, it’s obvious that it implies all the keys and is one of the candidate keys.

Candidate keys

* {aircraft\_id}
* {IATA\_code}
* {ICAO\_code}

According to the document, {IATA\_code} and {ICAO\_code} are not mandatory to exist. It’s only possibly given. To avoid NULL in the primary key, it’s better to choose {aircraft\_id} as primary key.

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (name) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ | Is in 2NF and there are no dependencies between non-prime columns. | |

**Airline(S)**

Content

Airline( airline\_id, name, country, ICAO\_code,IATA\_code, alias)

Functional dependencies

* {airline\_id} → { name, country, ICAO\_code, IATA\_code, alias}
* {IATA\_code} → {ICAO\_code}

Candidate keys

* {airline\_id}
* {IATA\_code}

One could argue that the considering the ICAO code also as the candidate key instead of IATA code, but ICAO code are subjected to certain regions whereas IATA code is derived from the name of the airport.

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (name, country, ICAO\_code, IATA\_code, alias) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ |  | |

**Airport(S)**

Content

Airport (IATA\_code, name, ICAO\_code, location, latitude, longitude)

Functional dependencies

* {IATA\_code} → {name, ICAO\_code, location, latitude, longitude}
* {latitude, longitude} → {location}

Candidate keys

* {IATA\_code}
* {latitude, longitude}

One could argue that ICAO code and location as the candidate key, but whereas in the case of ICAO code it is not dependent on the other columns. In the case of location, latitude and longitude represents the location more accurately. There is a possibility that the location might have two or more airports, and doesn’t have the same co-ordinates. So choosing the latitude and longitude as the candidate key is a better choice than the location.

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (name, ) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ |  | |

**Hotel booking(H)**

Hotel booking (hotel\_booking\_id, hotel\_id, trip\_id, price, breakfast\_included, check\_in\_date, check\_out\_date)

Functional dependencies

* {hotel\_booking\_id} → {hotel\_id, trip\_id, price, breakfast\_included, check\_in\_date, check\_out\_date}
* {hotel\_id, trip\_id, check\_in\_date, check\_out\_date} → {hotel\_booking\_id, price, breakfast\_included, check\_in\_date}

In this case, one could argue that either date (check-in/check-out) could be sufficient to identify a booking. However, one customer could have several bookings with different durations over a given period (e.g., family members arriving in a sequential manner at a hotel) on the same trip. Therefore, it seems more prudent to use both dates to avoid issues.

Candidate keys

* {hotel\_booking\_id}
* {hotel\_id, trip\_id, check\_in\_date, check\_out\_date}

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (price, breakfast\_included) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ | Is in 2NF and there are no dependencies between non-prime columns (between price and breakfast\_included). | |

**Hotel(H)**

Hotel (hotel\_id, name, city, address, country, average\_review\_score)

Functional dependencies

* {hotel\_id} → {name, city, address, country, average\_review\_score}

One could argue that there can be no hotel at the same address in the same city with the same name and the same average review score. However, knowing that we are tending towards bigger and bigger structures and that hotels have a tendency to use geographical elements in their names, it isn’t impossible that such a thing might occur which is why hotel\_id is the only viable candidate key.

Candidate keys

* {hotel\_id}

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (name, city, address, country, average\_review\_score) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ | Is in 2NF and there are no dependencies between any of the non-prime columns. | |

**Review (H)**

Review (review\_id, customer\_id, hotel\_booking\_id, date, content, score)

Functional dependencies

* {review\_id} → {customer\_id, hotel\_booking\_id, date, content, score}
* {hotel\_booking\_id} → {customer\_id, review\_id, date, content, score}

Candidate keys

* {review\_id}
* {hotel\_booking\_id}

It could be argued that {customer\_id, date, content} could also be added as a candidate key. However, it is technically possible that a customer could give the exact same review two different stays on the same day and therefore, it is more prudent to rule it out as a candidate key.

Normal form

| **NF** | **Satisfied?** | **Why** | |
| --- | --- | --- | --- |
| 1NF | ✅ | Contains a primary key and all columns are atomic. | |
| 2NF | ✅ | Is in 1NF and all non-prime columns (customer\_id, date, content, score) are functionally dependent on all the columns of each candidate key. | |
| 3NF | ✅ | Is in 2NF and there are no dependencies between any of the non-prime columns. | |