Pauline Michel_Emma Serfaty_Thomas Taylor

Databases - Group Project

The conceptual schema

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

No. Several hotels can have the same name, as in the case of hotel chains such as Campanile, Hilton, Ibis and so on. We need to use unique data, such as an hotel_id.

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.*

No. Flight numbers are not unique as the same flight number can denote two flights operated by the same airline company on different days. However, flight_number can be paired with departure_date to form a primary key in order to identify a flight. To make it simpler, we can even create a flight_id and use it as the primary key.

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. Firstly, although it is unlikely, it is still possible for two customers to leave exactly the same review (especially when it comes to short comments such as "great!"). Secondly, even if all the textual content were different, some reviews can be very long and using them as a primary key would not be practical nor optimal. It seems preferable to use the author and the date_of_publication as the primary key, or even to create a review_id and use it as the primary key.

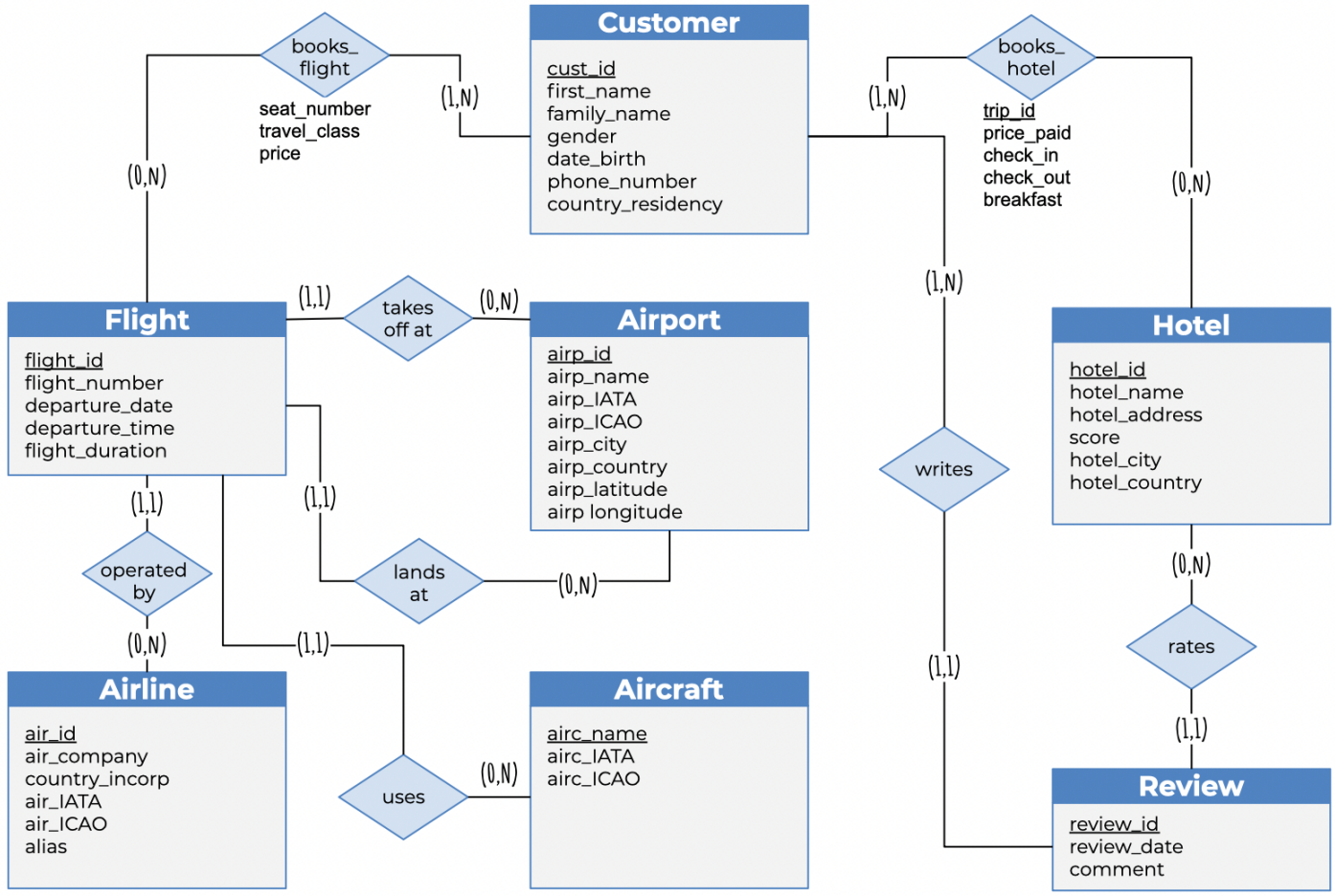
d. *Knowing that the IATA code uniquely identifies an airport, would you choose it as a primary key for the entity Airport? Justify your answer.*

In theory, unique data can be used as the primary key. However, the IATA code contains NULL data for some airports, and therefore cannot be used as a primary key. The same applies to the ICAO code. On the other hand, airport names can sometimes be quite complex (especially with the special characters of foreign airports). It is therefore better to use a primary key such as airport_id.

Exercise 1:

Propose an Entity-Relationship diagram describing the conceptual model of a relational database for the given application context:

- *Specify all the attributes for each entity and relation.*
- *For each entity, underline the attributes composing the primary key.*
- *For each relation, clearly indicate the minimum and maximum cardinality.*



Exercise 2

Translate the logical schema into a collection of tables. For each table:

- *Indicate its name and the names of the columns (but not their types).*
- *Underline the columns that are part of the primary key.*
- *Indicate the entity in the ER diagram to which the table corresponds.*

To make sure you choose the right data types for the columns, you can also check the values in the dataset.

Aircraft (airc_name, airc_IATA, airc_ICAO) → entity "aircraft" in the ER diagram.
Airport (airp_id, airp_name, airp_IATA, airp_ICAO, airp_city, airp_country, airp_latitude, airp_longitude) → entity "airport" in the ER diagram.
Customer (cust_id, first_name, family_name, gender, date_birth, phone_number, country_residency) → entity "customer" in the ER diagram.
Airline (air_id, air_company, country_incorp, air_IATA, air_ICAO, alias) → entity "airline" in the ER diagram.
Hotel (hotel_id, hotel_name, hotel_address, score, hotel_city, hotel_country) → entity "hotel" in the ER diagram.
Review (review_id, review_date, comment, *hotel_id, cust_id*) → entity "review" in the ER diagram.

Flight (flight_id, flight_number, departure_date, departure_time, flight_duration, *airp_id_src*, *airp_id_dst*, *air_id, airc_IATA*) → entity "flight" in the ER diagram.

Booking_Flight (*cust_id, flight_id*, seat_number, travel_class, price) → relationship "books_flight" in the ER diagram.
Booking_Hotel (trip_id, *cust_id, hotel_id*, price_paid, check_in, check_out, breakfast)
→ relationship "books_hotel" in the ER diagram.

Exercise 3 & 4

For each table:

- *Indicate a minimal set of functional dependencies.*
- *Indicate the normal form that the table satisfies. Justify your answer.*

- *Normalize each table up to the Boyce-Codd Normal Form (BCNF).*

Aircraft (airc_name, airc_IATA, airc_ICAO)
Functional dependencies: airc_name → airc_ICAO airc_name → airc_IATA
Normal form: 1NF ✓ *yes* because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value. 2NF ✓ *yes* because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key. 3NF ✓ *yes* because the table is in 2NF and no non-prime column depends on non-prime columns.
Normalization to the Boyce-Codd Normal Form: The table is already in BCNF. **Aircraft** (airc_name, airc_IATA, airc_ICAO)

Airport (airp_id, airp_name, airp_IATA, airp_ICAO, airp_city, airp_country, airp_latitude, airp_longitude)
Functional dependencies: airp_id → airp_name airp_id → airp_IATA airp_id → airp_ICAO airp_id → airp_latitude

airp_id → airp_longitude
{airp_latitude, airp_longitude} → airp_country
{airp_latitude, airp_longitude} → airp_city

Normal form:

1NF ✓ yes because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value.
2NF ✓ yes because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key.
3NF × no because the table is in 2NF but airp_country and airp_latitude, non-prime columns, depend on {airp_latitude, airp_longitude}, also non-prime columns.

Transition to 3NF:
Airport (airp_id, airp_name, airp_ICAO, airp_IATA, airp_latitude, airp_longitude)
Airport_Location (airp_latitude, airp_longitude, airp_ country, airp_city)

Normalization to the Boyce-Codd Normal Form:

The Third Normal Form is already in BCNF.
Airport (airp_id, airp_name, airp_ICAO, aip_IATA, airp_latitude, airp_longitude)
Airport_Location (airp_latitude, airp_longitude, airp_ country, airp_city)

Customer (cust_id, first_name, family_name, gender, date_birth, phone_number, country_residency)

Functional dependencies:

cust_id → first_name
cust_id → family_name
cust_id → gender
cust_id → date_birth
cust_id → phone_number
cust_id → country_residency

Normal form:

1NF ✓ yes because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value.
2NF ✓ yes because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key.
3NF ✓ yes because the table is in 2NF and no non-prime column depends on non-prime columns.

Normalization to the Boyce-Codd Normal Form:

The table is already in BCNF.

Customer (cust_id, first_name, family_name, gender, date_birth, phone_number, country_residency)

Airline (air_id, air_company, country_incorp, air_IATA, air_ICAO, alias)

Functional dependencies:

air_id → air_company
air_id → country_incorp
air_id → air_IATA
air_id → air_ICAO
air_id → alias

Normal form:

1NF ✓ yes because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value.
2NF ✓ yes because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key.
3NF ✓ yes because the table is in 2NF and no non-prime column depends on non-prime columns.

Normalization to the Boyce-Codd Normal Form:

The Third Normal Form is already in BCNF.
Airline (air_id, air_company, country_incorp, air_IATA, air_ICAO, alias)

Hotel (hotel_id, hotel_name, hotel_address, score, hotel_city, hotel_country)

Functional dependencies:

hotel_id → hotel_name
hotel_id → hotel_address
hotel_id → score
hotel_id → hotel_city
hotel_id → hotel country

Normal form:

1NF ✓ yes because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value.
2NF ✓ yes because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key.
3NF ✓ yes because the table is in 2NF and no non-prime column depends on non-prime columns.

Normalization to the Boyce-Codd Normal Form:

The table is already in BCNF.
Hotel (hotel_id, hotel_name, hotel_address, score, hotel_city, hotel_country)

Review (review_id, review_date, comment, *hotel_id, cust_id*)

Functional dependencies:

review_id → review_date
review_id → comment
review_id → *hotel_id*
review_id → *cust_id*

Normal form:

1NF ✓ yes because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value.
2NF ✓ yes because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key.
3NF ✓ yes because the table is in 2NF and no non-prime column depends on non-prime columns.

Normalization to the Boyce-Codd Normal Form:

The table is already in BCNF.
Review (review_id, review_date, comment, *hotel_id, cust_id*)

Flight (flight_id, flight_number, departure_date, departure_time, flight_duration, *airp_id_src*, *airp_id_dst*, *air_id, airc_IATA*)

Functional dependencies:

flight_id → flight_number
flight_id → departure_date
flight_id → departure_time
flight_id → flight_duration
flight_id → *airp_id_src*
flight_id → *airp_id_dst*
flight_id → *air_id*
flight_id → *airc_IATA*
flight_number → *air_id*

Normal form:

1NF ✓ yes because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value.

2NF ✓ yes because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key.
3NF × no because the table is in 2NF but air_id, a non-prime column, depends on flight_number, also a non-prime column.

Transition to 3NF:
Flight (flight_id, flight_number, departure date, departure_time, flight_duration, *airp_id_src*, *airp_id_dst, airc_IATA*)
Flight_Airline (flight_number, *air_id*)

Normalization to the Boyce-Codd Normal Form:

The Third Normal Form is already in BCNF.
Flight (flight_id, flight_number, departure date, departure_time, flight_duration, *airp_id_src*, *airp_id_dst, airc_IATA*)
Flight_Airline (flight_number, *air_id*)

Even though primary keys are not necessarily unique by definition, they are in MySQL. Thus, we will not be able to create the table Flight_Airline in SQL (because flight_number does have unique values). Therefore, we will actually create the following table:
Flight (flight_id, flight_number, departure_date, departure_time, flight_duration, *airp_id_src*, *airp_id_dst*, *air_id, airc_IATA*)

Booking_Flight (*cust_id, flight_id*, seat_number, travel_class, price)

Functional dependencies:

{*flight_id, cust_id*} → seat_number
{*flight_id, cust_id*} → travel_class
{*flight_id, cust_id*} → price
seat_number → travel_class

Normal form:

1NF ✓ yes because each row is unique and identified by a primary key; there are no duplicate columns; and each cell contains a single value.
2NF ✓ yes because the table is in 1NF and all non-prime columns are functionally dependent on all the columns of each candidate key.
3NF × no because the table is in 2NF but travel_class, a non-prime column, depends on seat_number, also a non-prime column.

Transition to 3NF:
Booking_Flight (*flight_id, cust_id*, seat_number, price)
Booking_Travel_Class (seat_number, travel_class)

Normalization to the Boyce-Codd Normal Form:

The Third Normal Form is already in BCNF.
Booking_Flight (*flight_id*, *cust_id*, seat_number, price)
Booking_Travel_Class (seat_number, travel_class)

For the same reasons as for the previous table, we will not be able to create the table Booking_Travel_Class in SQL (because seat_number does have unique values). Therefore, we will actually create the following table:
Booking_Flight (*cust_id*, *flight_id*, seat_number, travel_class, price)

Booking_Hotel (trip_id, *cust_id*, *hotel_id*, price_paid, check_in, check_out, breakfast)

Functional dependencies:

trip_id → *cust_id*
trip_id → *hotel_id*
trip_id → price_paid
trip_id → check_in
trip_id → check_out
trip_id → breakfast

Normal form:

1NF ✓ yes
2NF ✓ yes
3NF ✓ yes

Normalization to the Boyce-Codd Normal Form:

The table is already in BCNF.
Booking_Hotel (trip_id, *cust_id*, *hotel_id*, price_paid, check_in, check_out, breakfast)

QUESTION 5 :

Write the SQL code to create the tables. For each table:

- *Indicate the primary key.*
- *Indicate the foreign keys.*
- *Indicate NOT NULL and UNIQUE constraints, if needed.*

Aircraft:
Aircraft (airc_name, airc_IATA, airc_ICAO)

```
CREATE TABLE Aircraft (
        airc_name TEXT PRIMARY KEY,
        airc_IATA TEXT
```

```
    airc_ICAO TEXT
)
```

Airport:
Airport (airp_id, airp_name, airp_ICAO, aip_IATA, airp_latitude, airp_longitude)
Airport_Location (airp_latitude, airp_longitude, airp_ country, airp_city)

```
CREATE TABLE Airport (
    airp_id_src INTEGER PRIMARY KEY,
    airp_name TEXT,
    airp_ICAO TEXT,
    airp_IATA TEXT,
    airp_latitude REAL NOT NULL,
    airp_longitude REAL NOT NULL,
    FOREIGN KEY (airp_latitude, airp_longitude) references to Airport_Location
    (airp_latitude, airp_longitude)
)

CREATE TABLE Airport_Location (
    airp_latitude REAL,
    airp_longitude REAL,
    airp_country TEXT,
    airp_city TEXT,
    PRIMARY KEY (airp_latitude, airp_longitude)
)
```

Customer:
Customer (cust_id, first_name, family_name, gender, date_birth, phone_number, country_residency)

```
CREATE TABLE Customer (
    cust_id INTEGER PRIMARY KEY,
    first_name TEXT,
    family_name TEXT,
    gender TEXT,
    date_birth TEXT,
    phone_number INTEGER,
    country_residency TEXT
)
```

Airline:
Airline (air_id, air_company, country_incorp, air_IATA, air_ICAO, alias)

```
CREATE TABLE Airline (
        air_id INTEGER PRIMARY KEY,
        air_company TEXT,
        country_incorp TEXT,
        air_IATA TEXT,
        air_ICAO TEXT,
        alias TEXT
)
```

Hotel:
Hotel (hotel_id, hotel_name, hotel_address, score, hotel_city, hotel_country)

```
CREATE TABLE Hotel (
        hotel_id INTEGER PRIMARY KEY,
        hotel_name TEXT,
        hotel_address TEXT,
        score REAL
        hotel_city TEXT,
        hotel_country TEXT
)
```

Review:
Review (review_id, review_date, comment, *hotel_id, cust_id*)

```
CREATE TABLE Review (
        review_id INTEGER PRIMARY KEY,
        review_date TEXT,
        comment TEXT,
        hotel_id INTEGER NOT NULL,
        cust_id INTEGER NOT NULL,
        FOREIGN KEY (hotel_id) references to Hotel (hotel_id),
        FOREIGN KEY (cust_id) references to Customer (cust_id)
)
```

Flight:
Flight (flight_id, flight_number, departure_date, departure_time, flight_duration, *airp_id_src*, *airp_id_dst*, *air_id, airc_IATA*)

```
CREATE TABLE Flight(
        flight_id INTEGER PRIMARY KEY,
        flight_number TEXT,
        departure_date TEXT,
        departure_time INTEGER,
```

```
        flight_duration TEXT,
        airp_id_src  INTEGER NOT NULL,
        airp_id_dst INTEGER NOT NULL,
        air_id INTEGER NOT NULL,
        airc_IATA TEXT NOT NULL,
        FOREIGN KEY (air_id) references to Airline (air_id)
        FOREIGN KEY (airc_IATA) references Aircraft (airc_IATA)
)
```

Booking_Flight:
Booking_Flight (*cust_id*, *flight_id*, seat_number, travel_class, price)

```
CREATE TABLE Booking_Flight (
        cust_id INTEGER NOT NULL,
        flight_id INTEGER NOT NULL,
        seat_number TEXT,
        travel_class TEXT,
        price REAL,
        FOREIGN KEY (cust_id) references to Customer (cust_id),
        FOREIGN KEY (flight_id) references to Flight (flight_id)
)
```

Booking_Hotel:
Booking_Hotel (trip_id, *cust_id*, *hotel_id*, price_paid, check_in, check_out, breakfast)

```
CREATE TABLE Booking_Hotel (
        trip_id INTEGER,
        cust_id INTEGER NOT NULL,
        hotel_id INTEGER NOT NULL,
        price_paid REAL,
        check_in TEXT,
        check_out TEXT,
        breakfast BOOLEAN,
        FOREIGN KEY (cust_id) references to Customer (cust_id),
        FOREIGN KEY (hotel_id) references to Hotel (hotel_id)
)
```

Exercise 8

Question 1

SELECT *

FROM Customer c

WHERE c.family_name = 'POMMIER'

After running the query multiple times we can say that the average is around 6ms, with the query taking from 5 to 8ms.

Question 2

We create the index with the following query:

CREATE INDEX index_family_name ON Customer(family_name)

Question 3

We try the same query as Question 1, but with the index created this time.

SELECT *

FROM Customer c

WHERE c.family_name = 'POMMIER'

Question 4

We can observe that the query takes about the same time, in average 4ms, going from 4ms to 6ms. There is a small difference between the time with index and not, the one with index could be slightly faster but it is not very noticeable. Maybe if we have a larger dataset it would be quicker.

Ne pas imprimer en PDF à partir d'ici, mais ne pas supp pour garder une trace

Exercise 7

Question 1

SELECT avg(b.price)

FROM Booking_Flight b JOIN Flight f USING (flight_id) JOIN Airline a using (air_id)

WHERE a.air_company = "Air France"

```
1   SELECT avg(b.price)
2   FROM Booking_Flight b JOIN Flight f USING (flight_id) JOIN Airline a using (air_id)
3   WHERE a.air_company = "Air France"
```

	avg(b.price)
1	370.210526315789

Question 2

SELECT COUNT(*) as t, country_residency

FROM Customer

GROUP BY country_residency

order by t desc

Question 3

SELECT a.airp_name

FROM Airport_Location al JOIN Airport a USING (airp_latitude,airp_longitude)

WHERE al.airp_city = "Paris" and al.airp_country = "France"

Question 4

SELECT al.airp_city, al.airp_country, count(airp_id)

FROM Airport a JOIN Airport_Location al USING (airp_latitude, airp_longitude)

GROUP BY airp_city, airp_country

order by count(airp_id) DESC limit 1 ;

Question 5

SELECT a.air_company, ac.airc_name

FROM Aircraft ac JOIN Flight f using (airc_iata) Join Airline a using (air_id)

WHERE ac.airc_name ="Airbus A300"

Question 6

Select c.cust_id, c.first_name, c.family_name

FROM Customer c , Booking_Flight bf , Flight f , Airport a, Airport_Location al

where c.cust_id=bf.cust_id and bf.flight_id=f.flight_id and f.airp_id_dst=a.airp_id and a.airp_latitude=al.airp_latitude and a.airp_longitude=al.airp_longitude and al.airp_city = "Sydney" and al.airp_country="Australia"

Question 7

SELECT (count(f.airp_id_src) + count(f.airp_id_dst)), a.airp_name,a.airp_id,al.airp_city,al.airp_country

FROM Airport a, Flight f, Airport_Location al

WHERE (a.airp_id = f.airp_id_dst) and a.airp_latitude= al.airp_latitude and a.airp_longitude=al.airp_longitude

GROUP BY a.airp_name

```
order by  count(a.airp_name) DESC

LIMIT 1 ;
```

Question 8

```
SELECT avg(price)

FROM Booking_Flight

WHERE travel_class = "economy"
```

Question 9

```
SELECT avg(price)

FROM Booking_Flight

WHERE travel_class = "business"
```

Question 10

```
SELECT a.airp_name, al.airp_city, al.airp_country, c.country_residency

FROM Customer c , Booking_Flight bf , Flight f , Airport a , Airport_Location al

WHERE bf.flight_id=f.flight_id and f.airp_id_dst = a.airp_id and a.airp_latitude=al.airp_latitude and a.airp_longitude=al.airp_longitude and c.country_residency="France"

Group by a.airp_name
```

Question 11

```
select count(al.airp_city) as nb_airp, al.airp_city, al.airp_country

from Flight f , Booking_Flight bf , Customer c, Airport a ,Airport_Location al

WHERE bf.flight_id=f.flight_id and bf.cust_id=c.cust_id and f.airp_id_dst=a.airp_id and a.airp_latitude=al.airp_latitude and a.airp_longitude=al.airp_longitude and c.gender="F"

group by al.airp_city
```

```
order by  nb_airp DESC

LIMIT 1 ;
```

Question 12

```
select count(al.airp_city) as nb_airp, al.airp_city, al.airp_country

from Flight f , Booking_Flight bf , Customer c, Airport a ,Airport_Location al

WHERE bf.flight_id=f.flight_id and bf.cust_id=c.cust_id and f.airp_id_dst=a.airp_id and a.airp_latitude=al.airp_latitude and a.airp_longitude=al.airp_longitude and c.gender="M"

group by al.airp_city

order by  count(al.airp_city) DESC

LIMIT 1 ;
```

Question 13

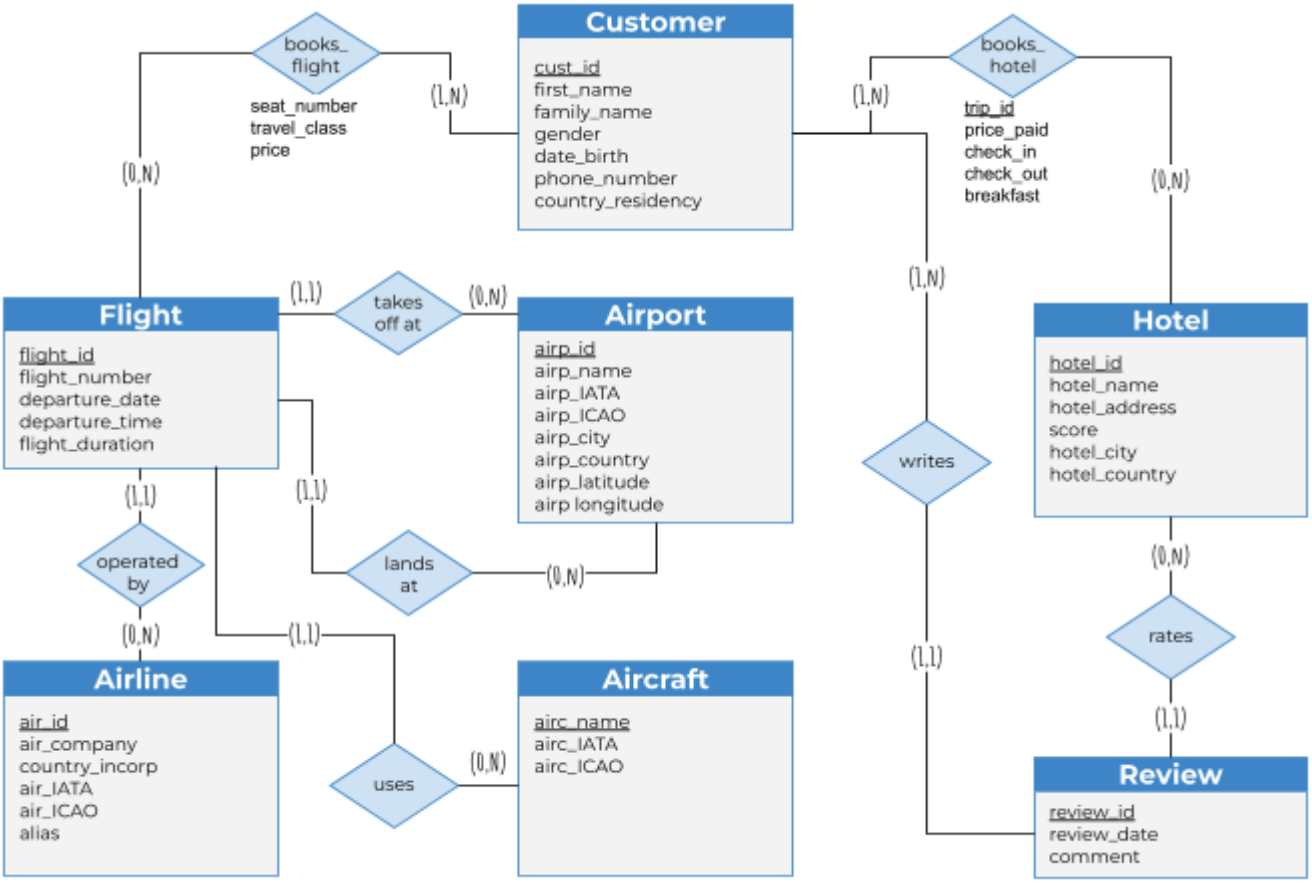
```
select Count(*),al.airp_city,al.airp_country,c.country_residency

FROM Flight f, Booking_Flight bf , Customer c, Airport a ,Airport_Location al

WHERE f.airp_id_dst=a.airp_id and f.flight_id=bf.flight_id and bf.cust_id=c.cust_id and a.airp_latitude=al.airp_latitude and a.airp_longitude=al.airp_longitude and airp_city="Paris"

GROUP BY c.country_residency
```

Question 14

```
SELECT COUNT(*),h.hotel_city,h.hotel_country

FROM Hotel h

GROUP BY h.hotel_city,h.hotel_country
```

Question 15

```
SELECT SUM(bf.price)

FROM Customer c, Booking_Flight bf

WHERE bf.cust_id=c.cust_id and c.first_name ="Tatiana" and c.family_name="REZE"
```


Customer
cust_id
first_name
family_name
gender
date_birth
phone_number
country_residency
books_flight
seat_number
travel_class
price
(1,N)
(0,N)
books_hotel
trip_id
price_paid
check_in
check_out
breakfast
(1,N)
(0,N)
Flight
flight_id
flight_number
departure_date
departure_time
flight_duration
(1,1)
takes off at
(0,N)
Airport
airp_id
airp_name
airp_IATA
airp_ICAO
airp_city
airp_country
airp_latitude
airp longitude
(1,N)
writes
Hotel
hotel_id
hotel_name
hotel_address
score
hotel_city
hotel_country
(1,1)
operated by
(0,N)
(1,1)
lands at
(0,N)
(0,N)
rates
(1,1)
Airline
air_id
air_company
country_incorp
air_IATA
air_ICAO
alias
(1,1)
uses
(0,N)
Aircraft
airc_name
airc_IATA
airc_ICAO
Review
review_id
review_date
comment