The hotels example — Elements of correction

Reminder: you need the three following files (make sure to put them in the same folder):

http://www.barsamian.am/2021-2022/S7ICTB/TP11_Hotels_database.py
http://www.barsamian.am/2021-2022/S7ICTB/TP11_Hotels_creations.sql
http://www.barsamian.am/2021-2022/S7ICTB/TP11_Hotels_insertions.sql

1. Give the list of the names of hotels, with their city.

   This question asks for a slight modification of the request given in the python file:

   ```
   SELECT name, city FROM hotels;
   ```

2. Give the names of hotels having 3 stars or more.

   ```
   SELECT name, city FROM hotels WHERE stars>=3;
   ```

3. Give the list of hotels for which there is at least one booking.

   This question asks for a join. We can directly output the fields name and city (no need to type hotels.name and hotels.city) because these fields are only in the hotels table. To avoid multiple outputs (when a hotel has multiple bookings), we use the keyword `DISTINCT`:

   ```
   SELECT DISTINCT name, city FROM hotels, bookings WHERE hotels.id=bookings.
       hotel_id;
   ```

4. Give the list of clients that did not occupy any room.

   This time, we cannot do it directly with a join. A possible solution is to select the table of all the clients, and to remove from this table the clients that occupied at least a room. This remove operation can be done with the keyword `EXCEPT`:

   ```
   SELECT DISTINCT surname, first_name FROM clients, occupancies
   EXCEPT
   SELECT DISTINCT surname, first_name FROM clients, occupancies
          WHERE clients.id=occupancies.client_id;
   ```

5. For each hotel, give the total number of rooms.

   ```
   SELECT name, city, COUNT(rooms.id) FROM hotels, rooms
   WHERE rooms.hotel_id=hotels.id GROUP BY hotels.id;
   ```

   Without the knowledge of the `GROUP BY` operator, we could have written multiple requests, inside a for loop. I then provide the associated python code. This is a really good example because it shows how to use the "?" with a variable, when executing a request with `sqlite3`, see Listing 1.

```
1 def requestQuestionFive(cursor):
2     for i in range(1,13): #Because there are 12 hotels with ids 1..12.
3         cursor.execute("SELECT name, city, COUNT(rooms.id) FROM hotels, rooms \
4             WHERE rooms.hotel_id=hotels.id AND hotels.id=?;", [i])
5         for row in cursor:
6             print(row)
7         print() # Add a blank line between two hotels
8     cursor.close()
```

Listing 1: Python code for question 5.

   PS: I insist that you write `rooms.hotel_id=hotels.id AND hotels.id=?`. You could be tempted to write instead `rooms.hotel_id=hotels.id=?` but this second formulation does not work.

6. Who were the clients of the "Hôtel des voyageurs", in Nice?

   This question asks for a double join: we have to join the tables hotels and occupancies (to get the occupancies of the "Hôtel des voyageurs", in Nice)

```
SELECT DISTINCT surname, first_name
FROM hotels JOIN occupancies ON hotels.id=occupancies.hotel_id
            JOIN clients ON clients.id=occupancies.client_id
WHERE hotels.name='Hôtel des voyageurs' AND hotels.city='Nice';
```

Another equivalent formulation is :

```
SELECT DISTINCT surname, first_name
FROM hotels, occupancies, clients
WHERE hotels.id=occupancies.hotel_id AND clients.id=occupancies.client_id
AND hotels.name='Hôtel des voyageurs' AND hotels.city='Nice';
```

PS: In case there is a trouble with "ô", you can write `hotels.name LIKE '%tel des voyageurs'` instead of `hotels.name='Hôtel des voyageurs'`.

7. How many triple rooms are in the "Hôtel des ambassadeurs", in Grenoble?

```
SELECT COUNT(rooms.id)
FROM hotels JOIN rooms ON hotels.id=rooms.hotel_id
WHERE hotels.name='Hôtel des ambassadeurs' AND hotels.city='Grenoble'
AND rooms.type='triple';
```

8. For each client, list the name and the city of the hotels in which he occupied a room.

   We will do as in question 5), a full implementation is shown in Listing 2. Last time, we cheated for the number of hotels: we opened `TP15_Hotels_insertions.sql` and looked at the number of hotels (we could also do the same and count 179 clients). However this time, we will ask SQL how many clients are in the database : `SELECT COUNT(*) FROM clients;`. sqlite3 will give us the results in `cursor.fetchall()`. This result contain only one line, and this line contains only one number: the number of clients, thus in `cursor.fetchall()[0][0]` (see line 4). For each client id `i`, we begin by identifying their name (lines 6–7), and we print a message with it (lines 8–10). Then we make a double join for the tables clients, occupancies and hotels, and we keep only the rows that interests us, those of the client with id `i`, taking care to remove duplicate hotels (lines 11–14). Finally, we print the rows of the answer (we could also pretty print them, as we have done for the client's name, lines 8–10).

```
1  def requestQuestionEight(cursor):
2      cursor.execute("SELECT COUNT(*) FROM clients;")
3      nb_clients = cursor.fetchall()[0][0]
4      for i in range(1,nb_clients+1):
5          cursor.execute("SELECT surname, first_name FROM clients \
6              WHERE clients.id=?;", [i])
7          result = cursor.fetchall()[0]
8          print("The client " + result[1] + " " + result[0]
9               + " has occupied the following hotels :")
10         cursor.execute("SELECT DISTINCT name, city \
11             FROM clients JOIN occupancies ON clients.id=occupancies.client_id \
12             JOIN hotels ON hotels.id=occupancies.hotel_id \
13             WHERE clients.id=?;", [i])
14         for row in cursor:
15             print(row)
16         print() # Add a blank line between two clients
17     cursor.close()
```

Listing 2: Python code for question 8.

9. List the rooms from the "Hôtel de la gare", in Bordeaux, that are available for the nights from 2014, April the 12th to 2014, April the 17th.

   This question is very long and hard, I did not expect you to come to the end of it, but at least, I expected some partial results.

   The available rooms are those for which there is no occupancy nor booking during that period. We will take the full room list, and remove those where an end date d_e verifies April, 12th $\leq d_e \leq$

April, 17th, those where a start date d_s verifies April, 12th $\leq d_s \leq$ April, 17th, those where the start date is before April, 12th and the end date is either after April, 17th either NULL (not yet ended). A possible solution is written in Listing 3.

```
def requestQuestionNine(cursor):
    # First, select the id of the hotel (it simplifies the following request)
    cursor.execute("SELECT id FROM hotels \
        WHERE name='Hôtel de la gare' AND city='Bordeaux';")
    id_hotel = cursor.fetchall()[0][0]
    # Then, does the request as explained before.
    cursor.execute("SELECT DISTINCT rooms.id FROM rooms WHERE rooms.hotel_id=? \
EXCEPT SELECT DISTINCT rooms.id FROM rooms JOIN occupancies \
  ON (rooms.id=occupancies.room_id AND rooms.hotel_id=occupancies.hotel_id) \
  WHERE \
    (((date_from<='2014-04-17' AND date_from>='2014-04-12') \
    OR (date_to<='2014-04-17' AND date_to>='2014-04-12') \
    OR (date_to>'2014-04-17' AND date_from<'2014-04-12') \
    OR (date_to IS NULL AND date_from<'2014-04-12')) \
    AND rooms.hotel_id=?) \
EXCEPT SELECT DISTINCT rooms.id FROM rooms JOIN bookings \
  ON (rooms.id=bookings.room_id AND rooms.hotel_id=bookings.hotel_id) \
  WHERE \
    (((date_from<='2014-04-17' AND date_from>='2014-04-12') \
    OR (date_to<='2014-04-17' AND date_to>='2014-04-12') \
    OR (date_to>'2014-04-17' AND date_from<'2014-04-12') \
    OR (date_to IS NULL AND date_from<'2014-04-12')) \
    AND rooms.hotel_id=?);", [id_hotel, id_hotel, id_hotel])
    for row in cursor:
        print(row)
```

Listing 3: Python code for question 9.

10. If the taxes represent 19.6% of the price excluding taxes, what will be the amount (taxes included) of each stay of Jean Némarre, that he had to pay before 2015, January the 21st? (A client only pays a stay at the end of it).

    This question was pretty straightforward, except on the computation of the number of days in a stay. In SQL, we can use DATEDIFF(start date, end date), but unfortunately this function is not available on sqlite3. In sqlite3, we can use JulianDay(end date) - JulianDay(start date) instead. In case we did not find out this solution, we could do the computation by hand, see Listing 4. In this listing, the computation of the number of days between two dates is done with the datetime module. It is a good exercise to rewrite this function without using this module:

    - input: two dates
    - output: the number of days between those dates
    - hint: you can store the number of days of each month in an array, and take special care of the leap years.

```
SELECT((JulianDay(date_to) - JulianDay(date_from)) * price_per_night_excl_tax *
    1.196)
FROM occupancies, rooms, clients
WHERE clients.first_name = 'Jean' AND clients.surname = 'Némarre'
AND occupancies.client_id = clients.id AND rooms.id = occupancies.room_id
AND rooms.hotel_id = occupancies.hotel_id
AND occupancies.date_to <= '2015-01-25';
```

```
from datetime import date

# This function takes two dates written as strings (as in SQL)
# and outputs the number of days between them.
def nbDaysBetween(date1, date2):
    array1 = date1.split("-")
    array2 = date2.split("-")
```

```
    datetime1 = date(int(array1[0]), int(array1[1]), int(array1[2]))
    datetime2 = date(int(array2[0]), int(array2[1]), int(array2[2]))
    return (datetime2 - datetime1).days

def requestQuestionTen(cursor):
    price = 0.
    request = "SELECT date_from, date_to, price_per_night_excl_tax \
FROM occupancies, rooms, clients \
WHERE clients.first_name = 'Jean' AND clients.surname = 'Némarre' \
AND occupancies.client_id = clients.id AND rooms.id = occupancies.room_id \
AND rooms.hotel_id = occupancies.hotel_id \
AND occupancies.date_to <= '2015-01-25';"
    cursor.execute(request)
    for row in cursor:
        date_from = row[0]
        date_to = row[1]
        price_per_night_excl_tax = row[2]
        price = price + 1.196 * price_per_night_excl_tax * nbDaysBetween(
            date_from, date_to)
    print("Jean Némarre paid " + price + " euros in total.")
```

Listing 4: Python code for question 10.

11. What are the rooms that have been occupied for more than 40 days?

    This question is very similar to the previous one. We will use the same `nbDaysBetween` function, see Listing 5.

```
def requestQuestionEleven(cursor):
    # First, list the hotel rooms (it simplifies the following request)
    cursor.execute("SELECT id, hotel_id FROM rooms;")
    all_rooms = cursor.fetchall()
    for (id, hotel_id) in all_rooms:
        cursor.execute("SELECT date_from, date_to \
FROM rooms JOIN occupancies ON \
(rooms.id=occupancies.room_id AND rooms.hotel_id=occupancies.hotel_id) \
WHERE rooms.id=? AND rooms.hotel_id=? AND date_to NOT NULL;", [id, hotel_id])
        nb_occupancy_days = 0
        for row in cursor:
            date_from = row[0]
            date_to = row[1]
            nb_occupancy_days = nb_occupancy_days + nbDaysBetween(date_from,
                date_to)
        if (nb_occupancy_days >= 40):
            print("Room " + str(id) + " of hotel with id " + str(hotel_id) + "
                has " + str(nb_occupancy_days) + " occupancy days in total.")
    cursor.close()
```

Listing 5: Python code for question 11.

PS: Notice the test that `date_to` is not null in this request. This is because in this request, we do not compare dates, so all occupancies will be output, even the ongoing ones. If a date is NULL, then row[1] will not contain a string, thus we won't be able to count the number of days. We have to take special care of those occupancies (here, we just remove them).