Databases with SQL — B Test from 2021, correction

1 SQL, without computer

We recall that the big picture of the database is the following:

- figurines: id, name, color, nb_legs
- collection: #figurine_id, drawer_color, drawer_room

1. If two figurines with the same id can be put in the same drawer, this means that two rows of the table collection could have the exact same values for all the fields. This is not allowed in a SQL table. To solve this problem, it is possible to add another field which could be, for example, something identifying when Bertrand went to the store (Bertrand only gets one figurine each time he goes to the store, so adding an identifier that is unique whenever he goes to the store would work). It could be for example an integer (1 the first time he goes to the store, 2 the second time, etc.), or the date when he went to the store.

2. Which of the 10 figurines are red?

```
SELECT * FROM figurines WHERE color='red';
```

3. Which of the 10 figurines have (strictly) more than 3 legs?

```
SELECT * FROM figurines WHERE nb_legs>3;
```

4. Which of Bertrand's figurines are in the bedroom?

```
SELECT * FROM collection WHERE drawer_room='bedroom';
```

5. What are Bertrand's figurines that are of the same color than the drawer they are in?

```
SELECT name, color, nb_legs FROM figurines, collection WHERE figurines.color=
    collection.drawer_color;
```

BONUS How many figurines does Bertrand own?

```
SELECT COUNT(*) FROM collection;
```

How many different ones?

```
SELECT COUNT(DISTINCT figurine_id) FROM collection;
```

2 SQL, with computer

An example of implementation is given in Listing 1. You can also download:

http://www.barsamian.am/EE_examens/S6ICT_Annales_TestB/Figurines/BTest_Figurines_database_correction.py

```
1  def doQuestionOne(cursor):
2      request = "SELECT DISTINCT drawer_room FROM collection WHERE figurine_id=0;"
3      doRequest(cursor, request)
4
5  def doQuestionTwo(cursor):
6      request = "SELECT DISTINCT name FROM figurines, collection WHERE figurines.id=
           collection.figurine_id and collection.drawer_room='living_room';"
7      doRequest(cursor, request)
8
9  def doQuestionThree(cursor):
```

```
    request = "SELECT COUNT(*) - COUNT(DISTINCT figurine_id) FROM collection WHERE figurine_id=0;"
    doRequest(cursor, request)
    request = "SELECT COUNT(*) - COUNT(DISTINCT figurine_id) FROM collection WHERE figurine_id=1;"
    doRequest(cursor, request)
    request = "SELECT COUNT(*) - COUNT(DISTINCT figurine_id) FROM collection WHERE figurine_id=2;"
    doRequest(cursor, request)
    request = "SELECT COUNT(*) - COUNT(DISTINCT figurine_id) FROM collection;"
    doRequest(cursor, request)
    # Another method (same idea than question 1 from the algorithms part)
    nb_extra = 0
    for i in range(10):
        request = "SELECT COUNT(*) FROM collection WHERE figurine_id=?;"
        cursor.execute(request, [i])
        nb_figurines = cursor.fetchall()[0][0]
        if (nb_figurines > 0):
            nb_extra = nb_extra + nb_figurines - 1
    print("Bertrand has " + str(nb_extra) + " extra figurines.")

def doQuestionBONUS(cursor):
    # We use a dictionary to store the realistic number of legs
    nb_legs = {}
    nb_legs["bird"] = 2
    nb_legs["cat"] = 4
    nb_legs["snake"] = 0
    nb_legs["fox"] = 4
    nb_legs["centipede"] = 100
    # Then for each figurine, we compare the dictionary to the database
    request = "SELECT id, name, nb_legs FROM figurines;"
    cursor.execute(request)
    for row in cursor:
        figurine_name = row[1]
        figurine_nb_legs = row[2]
        if (nb_legs[figurine_name] == figurine_nb_legs):
            print("figurine with id " + str(row[0]) + " has a realistic number of legs: " + figurine_name + " with " + str(figurine_nb_legs) + " legs")
        if (nb_legs[figurine_name] > figurine_nb_legs):
            print("figurine with id " + str(row[0]) + " has a realistic number of legs (with some broken ones): " + figurine_name + " with " + str(figurine_nb_legs) + " legs")
```

Listing 1: Python code for the computer part.