1. Connect to petsDB and check whether the connection has been successful.

In [106]:
```
import sqlite3
```

In [107]:
```
def connect():
    try:
        conn = sqlite3.connect("Data/petsdb")
        return conn
    except Error:
        return None
```

In [108]:
```
# Check connection by making a query.
def checkConnection(conn):
    try:
        cursor = conn.cursor()
        rows = conn.execute("SELECT * FROM persons")
        return True
    except:
        print("Error Selecting records. Check connection.")
        return False
```

In [109]:
```
conn = connect()
checkConnection(conn)
```

Out[109]:
```
True
```

In [110]:
```
conn.close()
```

In [111]:
```
checkConnection(conn)
```

```
Error Selecting records. Check connection.
```

Out[111]:
```
False
```

In [112]:
```
conn.close()
```

2. Find the different age groups in the persons database.

In [113]:
```
conn = connect()
```

In [114]:
```
cursor = conn.cursor()
```

In [115]:
```
for ppl, age in cursor.execute("SELECT count(*), age FROM persons GROUP B
Y age"):
    print("Number of people aged {} is {}".format(age, ppl))
```

```
Number of people aged 5 is 2
Number of people aged 6 is 1
Number of people aged 7 is 1
Number of people aged 8 is 3
Number of people aged 9 is 1
Number of people aged 11 is 2
Number of people aged 12 is 3
Number of people aged 13 is 1
Number of people aged 14 is 4
Number of people aged 16 is 2
Number of people aged 17 is 2
Number of people aged 18 is 3
Number of people aged 19 is 1
Number of people aged 22 is 3
Number of people aged 23 is 2
Number of people aged 24 is 3
Number of people aged 25 is 2
Number of people aged 27 is 1
Number of people aged 30 is 1
Number of people aged 31 is 3
Number of people aged 32 is 1
Number of people aged 33 is 1
Number of people aged 34 is 2
Number of people aged 35 is 3
Number of people aged 36 is 3
Number of people aged 37 is 1
Number of people aged 39 is 2
Number of people aged 40 is 1
Number of people aged 42 is 1
Number of people aged 44 is 2
Number of people aged 48 is 2
Number of people aged 49 is 1
Number of people aged 50 is 1
Number of people aged 51 is 2
Number of people aged 52 is 2
Number of people aged 53 is 2
Number of people aged 54 is 2
Number of people aged 58 is 1
Number of people aged 59 is 1
Number of people aged 60 is 1
Number of people aged 61 is 1
Number of people aged 62 is 2
Number of people aged 63 is 1
Number of people aged 65 is 2
Number of people aged 66 is 2
Number of people aged 67 is 1
Number of people aged 68 is 3
Number of people aged 69 is 1
Number of people aged 70 is 1
Number of people aged 71 is 4
Number of people aged 72 is 1
Number of people aged 73 is 5
Number of people aged 74 is 3
```

3. Find the age group that has the maximum number of people.

```
In [116]: for ppl, age in cursor.execute("SELECT count(*), age FROM persons GROUP B
Y age ORDER BY count(*) DESC"):
              print("Age {} has the highest number at {}".format(age,ppl))
              break
```

```
Age 73 has the highest number at 5
```

4. Find the people who do not have a last name.

In [117]:

```python
rows = cursor.execute("SELECT * FROM persons WHERE last_name IS null")
print("list of people with missing last name.\n")
for row in rows:
    print(row)
```

```
list of people with missing last name.

(1, 'Erica', None, 22, 'south port', 2345678)
(2, 'Jordi', None, 73, 'east port', 123456)
(3, 'Chasity', None, 70, 'new port', 76856785)
(4, 'Gregg', None, 31, 'new port', 76856785)
(6, 'Cary', None, 73, 'new port', 76856785)
(8, 'Francisca', None, 14, 'west port', 123456)
(10, 'Raleigh', None, 68, 'new port', 2345678)
(11, 'Maria', None, 42, 'west port', 123456)
(12, 'Mariane', None, 62, 'south port', 9756543)
(13, 'Mona', None, 44, 'south port', 76856785)
(14, 'Kayla', None, 36, 'south port', 2345678)
(15, 'Karlie', None, 35, 'west port', 123456)
(16, 'Morris', None, 71, 'west port', 76856785)
(17, 'Sandy', None, 23, 'east port', 2345678)
(18, 'Hector', None, 63, 'east port', 9756543)
(19, 'Hiram', None, 52, 'west port', 2345678)
(20, 'Tressa', None, 59, 'new port', 123456)
(21, 'Berry', None, 22, 'south port', 2345678)
(22, 'Pearline', None, 73, 'new port', 9756543)
(23, 'Maynard', None, 25, 'east port', 123456)
(24, 'Dorian', None, 40, 'east port', 123456)
(25, 'Mylene', None, 5, 'east port', 76856785)
(26, 'Lafayette', None, 34, 'new port', 2345678)
(29, 'Tara', None, 39, 'west port', 123456)
(30, 'Destiny', None, 18, 'south port', 2345678)
(31, 'Lesly', None, 31, 'west port', 123456)
(32, 'Perry', None, 19, 'south port', 76856785)
(35, 'Maritza', None, 73, 'east port', 9756543)
(37, 'Grant', None, 61, 'east port', 76856785)
(39, 'Laury', None, 17, 'east port', 9756543)
(40, 'Name', None, 52, 'east port', 9756543)
(41, 'Estefania', None, 32, 'new port', 76856785)
(42, 'Destiney', None, 65, 'west port', 2345678)
(43, 'Jaquelin', None, 73, 'west port', 9756543)
(45, 'Alfonzo', None, 16, 'east port', 2345678)
(46, 'Lisandro', None, 11, 'new port', 76856785)
(49, 'Priscilla', None, 65, 'east port', 76856785)
(50, 'Elenora', None, 11, 'new port', 76856785)
(52, 'Rudolph', None, 14, 'east port', 76856785)
(56, 'Ona', None, 35, 'east port', 9756543)
(57, 'Rebeca', None, 50, 'new port', 76856785)
(59, 'Sigurd', None, 12, 'west port', 76856785)
(63, 'Alice', None, 8, 'west port', 76856785)
(64, 'Dane', None, 24, 'west port', 9756543)
(65, 'Judge', None, 17, 'south port', 76856785)
(66, 'Allene', None, 9, 'new port', 9756543)
(67, 'Jalen', None, 33, 'new port', 2345678)
(70, 'Myron', None, 36, 'new port', 9756543)
(73, 'Travon', None, 16, 'south port', 2345678)
(74, 'Shayna', None, 60, 'new port', 2345678)
(75, 'Myah', None, 14, 'east port', 2345678)
(82, 'Letha', None, 44, 'new port', 9756543)
(84, 'Felton', None, 74, 'east port', 2345678)
(85, 'London', None, 66, 'east port', 9756543)
(86, 'Koby', None, 31, 'west port', 9756543)
```

```
(87, 'Golden', None, 35, 'east port', 76856785)
(89, 'Anissa', None, 8, 'south port', 76856785)
(91, 'Sid', None, 22, 'west port', 123456)
(96, 'Ernesto', None, 69, 'east port', 9756543)
(97, 'Josianne', None, 14, 'west port', 76856785)
```

5. Find out how many people have more than one pet.

In [118]:

```
rows = cursor.execute("SELECT count(*) FROM (SELECT count(owner_id) FROM
 pets GROUP BY owner_id HAVING count(owner_id) >1)")
for row in rows:
    print("{} People has more than one pets".format(row[0]))
```

```
43 People has more than one pets
```

6. Find out how many pets have received treatment.

In [119]:

```
rows = cursor.execute("SELECT count(*) FROM pets WHERE treatment_done=1")
print("number of pets who have recieved treatment")
for row in rows:
    print(row)
```

```
number of pets who have recieved treatment
(36,)
```

7. Find out how many pets have received treatment and the type of pet is known.

In [120]:

```
rows = cursor.execute("SELECT count(*) FROM pets WHERE treatment_done=1 A
ND pet_type IS NOT null")
print("number of pets with known type")
for row in rows:
    print(row)
```

```
number of pets with known type
(16,)
```

8. Find out how many pets are from the city called east port.

In [121]:

```
rows = cursor.execute("SELECT count(*) FROM pets JOIN persons ON pets.own
er_id = persons.id WHERE persons.city='east port'")
print("number of pets from east port")
for row in rows:
    print(row)
```

```
number of pets from east port
(49,)
```

9. Find out how many pets are from the city called east port and who received a treatment.

In [122]:

```
rows = cursor.execute("SELECT count(*) FROM pets JOIN persons ON pets.owner_id = persons.id WHERE persons.city='east port' AND pets.treatment_done=1")
print("number of pets from east port who received treatment")
for row in rows:
    print(row)
```

```
number of pets from east port who received treatment
(11,)
```