Assignment Weeks 11 & 12: Exercises

'''

Name : Karthikeyan Chellamuthu

Date : 06-05-2022

'''

ACTIVITY 11: RETRIEVING DATA CORRECTLY FROM DATABASES

As we can see, the id column in the persons table (which is an integer) serves as the primary key for that table and as a foreign key for the pet table, which is linked via the owner_id column.

The persons table has the following columns:

first_name: The first name of the person last_name: The last name of the person (can be 'null') age: The age of the person city: The city from where he/she is from zip_code: The zip code of the city The pets table has the following columns:

pet_name: The name of the pet. pet_type: What type of pet it is, for example, cat, dog, and so on. Due to a lack of further information, we do not know which number represents what, but it is an integer and can be null. treatment_done: It is also an integer column, and 0 here represents 'No', whereas 1 represents 'Yes'. The name of the SQLite DB is petsdb and it is supplied along with the Activity notebook.

In [1]:
```
# 1. Importing the libraries required for these exercises.

import sqlite3
```

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

In [11]:
```
# 1. Data Wrangling with Python: Activity 11, page 320

# 2. Connect to DB and check if the connection is successful Below program validates

with sqlite3.connect('petsdb') as conn:
    cursor = conn.cursor()
#     rows = cursor.execute('PRAGMA foreign_keys = 1')
    try:
        conn.execute("SELECT * FROM persons LIMIT 1")
        print("Connection is sucessfull")
    except sqlite3.ProgrammingError as e:
        print(f"Error: {e}")
        print("Connection Failure")
conn.close()
```

```
Connection is sucessfull
```

In [13]:
```
# Checking the connection for failure scenario

try:
    conn.execute("SELECT * FROM persons")
    print("Connection is sucessfull")
except sqlite3.ProgrammingError as e:
    print(f"Error: {e}")
    print(False)
```

```
Error: Cannot operate on a closed database.
False
```

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

In [17]:
```
# EStablishing the connecion

conn = sqlite3.connect('petsdb')
cursor = conn.cursor()
```

In [18]:
```
# Establishing the connection to petsdb and printing the output showing different ag

rows = cursor.execute("SELECT COUNT(*), age FROM persons GROUP BY age")
for c, a in rows:
    print(f"Age {a}:\t{c} person(s)")
```

```
Age 5:  2 person(s)
Age 6:  1 person(s)
Age 7:  1 person(s)
Age 8:  3 person(s)
Age 9:  1 person(s)
Age 11: 2 person(s)
Age 12: 3 person(s)
Age 13: 1 person(s)
Age 14: 4 person(s)
Age 16: 2 person(s)
Age 17: 2 person(s)
Age 18: 3 person(s)
Age 19: 1 person(s)
Age 22: 3 person(s)
Age 23: 2 person(s)
Age 24: 3 person(s)
Age 25: 2 person(s)
Age 27: 1 person(s)
Age 30: 1 person(s)
Age 31: 3 person(s)
Age 32: 1 person(s)
Age 33: 1 person(s)
Age 34: 2 person(s)
Age 35: 3 person(s)
Age 36: 3 person(s)
Age 37: 1 person(s)
Age 39: 2 person(s)
Age 40: 1 person(s)
Age 42: 1 person(s)
Age 44: 2 person(s)
Age 48: 2 person(s)
```

```
Age 49: 1 person(s)
Age 50: 1 person(s)
Age 51: 2 person(s)
Age 52: 2 person(s)
Age 53: 2 person(s)
Age 54: 2 person(s)
Age 58: 1 person(s)
Age 59: 1 person(s)
Age 60: 1 person(s)
Age 61: 1 person(s)
Age 62: 2 person(s)
Age 63: 1 person(s)
Age 65: 2 person(s)
Age 66: 2 person(s)
Age 67: 1 person(s)
Age 68: 3 person(s)
Age 69: 1 person(s)
Age 70: 1 person(s)
Age 71: 4 person(s)
Age 72: 1 person(s)
Age 73: 5 person(s)
Age 74: 3 person(s)
```

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

In [19]:

```
# Calculating the age group having maximum number of people

rows = cursor.execute("SELECT COUNT(*), age FROM persons GROUP BY age ORDER BY count
for c, a in rows:
    print(f"Most people ({c} total) were {a} years old.")
    break
```

```
Most people (5 total) were 73 years old.
```

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

In [20]:

```
# Query to find the persons who don't have last name

rows = cursor.execute("SELECT * FROM persons WHERE last_name is Null")
for row in rows:
    print(row)
```

```
(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)
```

In [24]:

```
# Calculating the count of persons having null last name

for row in cursor.execute("SELECT count(*) FROM persons WHERE last_name is Null"):
    print("The number of persons not having last name: {}".format(row[0]))
```

```
The number of persons not having last name: 60
```

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

In [26]:

```
# Number of people having more than one pet

sql = """
```

```
        SELECT count(*) FROM (SELECT count(owner_id) FROM pets
        GROUP BY owner_id HAVING count(owner_id) >1)
    """
rows = cursor.execute(sql)
for row in rows:
    print(f"{row[0]} people have more than 1 pet.")
```

```
43 people have more than 1 pet.
```

6. Find out how many pets have received treatment.

In [27]:

```
# Get number of treated pets
sql = "SELECT count(*) FROM pets WHERE treatment_done=1"
for row in cursor.execute(sql):
    treated = row[0]

# Get total number of pets
rows = cursor.execute("SELECT count(*) FROM pets")
for row in rows:
    total = row[0]

print(f"{treated} out of {total} pets have been treated.")
```

```
36 out of 150 pets have been treated.
```

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

In [28]:

```
# Get number of treated pets of known type

sql = "SELECT count(*) FROM pets WHERE treatment_done=1 AND pet_type IS NOT Null"
for row in cursor.execute(sql):
    treatAndType = row[0]

# Get total number of pets with known type

sql = "SELECT count(*) FROM pets WHERE pet_type IS NOT Null"
for row in cursor.execute(sql):
    totalKnownType = row[0]

print(f"{treatAndType} pets out of {totalKnownType} with known type have been treate
```

```
16 pets out of 68 with known type have been treated.
```

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

In [29]:

```
# For total number of people from 'east port':
# SELECT count(*) FROM persons WHERE city='east port'

sql = """
        SELECT count(*) FROM pets
        JOIN persons ON pets.owner_id = id
        WHERE persons.city='east port'
```

```
"""
for row in cursor.execute(sql):
    print(f"There are {row[0]} pets from the city called 'east port'.")
```

```
There are 49 pets from the city called 'east port'.
```

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

In [30]:

```
# Calculate the count of pet from east port and received a treatment

sql = """
        SELECT count(*) FROM pets
        JOIN persons ON pets.owner_id = id
        WHERE persons.city='east port' AND pets.treatment_done=1
"""
for row in cursor.execute(sql):
    print(f"There are {row[0]} pets from 'east port' who received treatment.")
```

```
There are 11 pets from 'east port' who received treatment.
```

In [ ]: