Data Wrangling with Python: Activity 11, page 320

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

In [78]:
```
#SQLite3 library
import sqlite3
```

In [79]:
```
#testing if connection was successful
def check_conn(conn):
    try:
        conn.cursor()
        return True
    except Exception as ex:
        return False
```

In [80]:
```
#connect to petsDB
conn = sqlite3.connect("petsdb")
```

In [81]:
```
#check connection
check_conn(conn)
```

Out[81]:
```
True
```

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

In [82]:
```
#initiate cursor for executing statements
c = conn.cursor()
```

In [83]:
```
#execute SQL query for grouping ages
rows = c.execute("SELECT COUNT(*),age FROM persons GROUP BY age")

#printing SQL results
for ppl,age in rows:
    print("We have {} people that are {} years old".format(ppl, age))
```

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

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

In [84]:
```
#executing SQL query for maximum number of people
max_rows = c.execute("SELECT MAX(mycount),age FROM (SELECT age, COUNT(*) mycount FROM persons GROUP BY age)")
```

In [85]:
```
#retrieve age group with maximum number of people
for ppl,age in max_rows:
    print("The maximum is at {} people, who are all {} years old".format(ppl, age))
```

```
The maximum is at 5 people, who are all 73 years old
```

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

In [86]:
```
#execute query for getting rows in persons table without last name (null)
null_last = c.execute("SELECT * FROM persons WHERE last_name IS NULL")
```

In [87]:
```
#printing persons results
for row in null_last:
    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, 'Treasa', 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, 'Melissa', 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 [88]:
```
#executing query for getting count of people with NULL last name
null_last_count = c.execute("SELECT COUNT(*) FROM persons WHERE last_name IS NULL")
```

In [89]:
```
#print count of people with no last name
for count in null_last_count:
    print("Number of people with no last name: {}".format(count[0]))
```

```
Number of people with no last name: 60
```

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

In [90]:
```
#execute query on pets table
more_pet = c.execute("SELECT COUNT(*) FROM (SELECT COUNT(owner_id) petcount FROM pets GROUP BY owner_id HAVING
```

In [91]:
```
#retrieve count of people with more than one pet
for petcount in more_pet:
    print("Number of people with more than one pet: {}".format(petcount[0]))
```

```
Number of people with more than one pet: 43
```

1. Find out how many pets have received treatment.

In [92]:
```
#execute SQL to get number of pets who have received treatment
treatment = c.execute("SELECT COUNT(*) FROM pets WHERE treatment_done = 1")
```

In [93]:
```
#retrieve count for pets with treatment
for treat_count in treatment:
    print("Number of pets that have received treatment: {}".format(treat_count[0]))
```

```
Number of pets that have received treatment: 36
```

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

In [94]:
```
#execute SQL with where statements for treatment and pet_type
treat_type = c.execute("SELECT COUNT(*) FROM pets WHERE treatment_done=1 AND pet_type IS NOT NULL")
```

In [95]:
```
#retrieve count for pets that have received treatment and pet_type is known
for tt_count in treat_type:
    print("Number of pets that have received treatment and the type of pet is known: {}".format(tt_count[0]))
```

```
Number of pets that have received treatment and the type of pet is known: 16
```

In [96]:
```
#close the connection for conn
conn.close()
```

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

In [97]:
```
#perform join between persons and pets tables
with sqlite3.connect('petsdb') as conn2:
    cursor = conn2.cursor()
    cursor.execute("PRAGMA foreign_keys = 1")
    #sql for join and where condition for persons.city
    sql = """
    SELECT COUNT(*) FROM persons
    JOIN pets ON persons.id = pets.owner_id
    WHERE persons.city = 'east port'
    """
    #execute query
    east_port = cursor.execute(sql)
    #print query results
    for row in east_port:
        print("Number of pets from the city called east port: {}".format(row[0]))
```

```
Number of pets from the city called east port: 49
```

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

In [98]:
```
#perform join between persons and pets tables
with sqlite3.connect('petsdb') as conn2:
    cursor = conn2.cursor()
    cursor.execute("PRAGMA foreign_keys = 1")
    #sql for join and where condition for persons.city and pets.treatment_done
    sql = """
    SELECT COUNT(*) FROM persons
    JOIN pets ON persons.id = pets.owner_id
    WHERE persons.city = 'east port'
    AND pets.treatment_done = 1
    """
    #execute query
    eastport_treat = cursor.execute(sql)
    #print query results
    for row in eastport_treat:
        print("Number of pets from the city called east port and that received a treatment: {}".format(row[0]))
```

```
Number of pets from the city called east port and that received a treatment: 11
```

In [99]:
```
#closing the connections for conn2
conn2.close()
```

In [ ]:
```

```