jupyter index (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 3

Run Markdown

Review: Using Sqlite - Lab

Introduction

In this lab, we will1 write more `SELECT` statements to review everything we've learned and solidify our ability to query a SQL database. We will also write more specific queries using the tools we learned in the previous lesson.

Objectives

You will be able to:

- Solidify our ability to interact with SQL databases by writing more `SELECT` statements
- Use `SELECT` with `ORDER BY` and `DESC` / `ASC` to order our results by the values of a specific column
- Use `LIMIT` to select only a certain number of rows
- Use `BETWEEN` to obtain results that fit between specified values

Famous Dogs

We have a database full of famous dogs! The `dogs` table is populated with the following data:

name	age	gender	breed	temperament	hungry
Snoopy	3	M	beagle	friendly	1
McGruff	10	M	bloodhound	aware	0
Scooby	6	M	great dane	hungry	1
Little Ann	5	F	coonhound	loyal	0
Pickles	13	F	black lab	mischievous	1
Clifford	4	M	big red	smiley	1
Lassie	7	F	collie	loving	1
Snowy	8	F	fox terrier	adventurous	0
NULL	4	M	golden retriever	playful	1

Connect to SQL Database with Python

Before we can query the `dogs.db` database, we need to connect to it. In the cell below:

- Import the `sqlite3` library
- Create a connection object that has connected to `dogs.db`
- Create a cursor object using the connection object

In [1]:
```
import sqlite3

conn = sqlite3.connect('dogs.db')

c = conn.cursor()
```

Queries

In the cells below:

- Write the corresponding sql queries in the appropriate variables
- Use the cursor object to `execute()` each query
- Call `c.fetchall()` to see the results for each query

- `select_all_female_dogs_name_and_breed` returns the name and breed for all female dogs

In [2]:
```
select_all_female_dogs_name_and_breed = "SELECT name, breed FROM dogs WHERE gender = 'F';"
c.execute(select_all_female_dogs_name_and_breed)
c.fetchall()

# Expected Output:
```

```
# [('Little Ann', 'coonhound'),
#  ('Pickles', 'black lab'),
#  ('Lassie', 'collie'),
#  ('Snowy', 'fox terrier')]
```

Out[2]:
```
[('Little Ann', 'coonhound'),
 ('Pickles', 'black lab'),
 ('Lassie', 'collie'),
 ('Snowy', 'fox terrier')]
```

- `select_all_dogs_names_in_alphabetical_order` returns the names of all dogs listed in alphabetical order. Notice that SQL lists the nameless dog first.

In [3]:
```
select_all_dogs_names_in_alphabetical_order = "SELECT name FROM dogs ORDER BY name;"
c.execute(select_all_dogs_names_in_alphabetical_order)
c.fetchall()

# Expected Output:
# [(None,),
#  ('Clifford',),
#  ('Lassie',),
#  ('Little Ann',),
#  ('McGruff',),
#  ('Pickles',),
#  ('Scooby',),
#  ('Snoopy',),
#  ('Snowy',)]
```

Out[3]:
```
[(None,),
 ('Clifford',),
 ('Lassie',),
 ('Little Ann',),
 ('McGruff',),
 ('Pickles',),
 ('Scooby',),
 ('Snoopy',),
 ('Snowy',)]
```

- `select_nameless_dog` returns all information for any dog that doesn't have a name

In [4]:
```
select_nameless_dog = "SELECT * FROM dogs WHERE name IS NULL;"
c.execute(select_nameless_dog)
c.fetchall()

# Expected Output:
# [(9, None, 4, 'M', 'golden retriever', 'playful', 1)]
```

Out[4]:
```
[(9, None, 4, 'M', 'golden retriever', 'playful', 1)]
```

- `select_hungry_dogs_name_and_breed_ordered_by_youngest_to_oldest` returns the name and breed of only the hungry dogs and lists them from youngest to oldest

In [5]:
```
select_hungry_dogs_name_and_breed_ordered_by_oldest_to_youngest = "SELECT name, breed FROM dogs WHE
c.execute(select_hungry_dogs_name_and_breed_ordered_by_oldest_to_youngest)
c.fetchall()

# Expected Output:
# [('Snoopy', 'beagle'),
#  ('Clifford', 'big red'),
#  (None, 'golden retriever'),
#  ('Scooby', 'great dane'),
#  ('Lassie', 'collie'),
#  ('Pickles', 'black lab')]
```

Out[5]:
```
[('Snoopy', 'beagle'),
 ('Clifford', 'big red'),
 (None, 'golden retriever'),
 ('Scooby', 'great dane'),
 ('Lassie', 'collie'),
 ('Pickles', 'black lab')]
```

- `select_name_age_and_temperament_of_oldest_dog` returns the oldest dog's name, age, and temperament

In [6]:
```
select_name_and_age_of_oldest_dog = "SELECT name, age FROM dogs ORDER BY age DESC LIMIT 1;"
c.execute(select_name_and_age_of_oldest_dog)
c.fetchall()

# Expected Output:
# [('Pickles', 13)]
```

Out[6]:
```
[('Pickles', 13)]
```

- `select_name_and_age_of_three_youngest_dogs` returns the three youngest dogs

In [7]:
```
select_name_and_age_of_three_youngest_dogs = "SELECT name, age FROM dogs ORDER BY age LIMIT 3;"
c.execute(select_name_and_age_of_three_youngest_dogs)
c.fetchall()

# Expected Output:
# [('Snoopy', 3), ('Clifford', 4), (None, 4)]
```

Out[7]:
```
[('Snoopy', 3), ('Clifford', 4), (None, 4)]
```

- `select_name_and_breed_of_dogs_between_age_five_and_ten_ordered_by_oldest_to_youngest` returns the name and breed of only the dogs who are between five and ten years old

In [8]:
```
select_name_and_temperament_of_dogs_between_age_five_and_ten_ordered_by_oldest_to_youngest = "SELEC
c.execute(select_name_and_temperament_of_dogs_between_age_five_and_ten_ordered_by_oldest_to_younges
c.fetchall()

# Expected Output:
# [('McGruff', 'bloodhound'),
#  ('Snowy', 'fox terrier'),
#  ('Lassie', 'collie'),
#  ('Scooby', 'great dane'),
#  ('Little Ann', 'coonhound')]
```

Out[8]:
```
[('McGruff', 'bloodhound'),
 ('Snowy', 'fox terrier'),
 ('Lassie', 'collie'),
 ('Scooby', 'great dane'),
 ('Little Ann', 'coonhound')]
```

- `select_name_age_and_hungry_of_hungry_dogs_between_age_two_and_seven_in_alphabetical_order` returns the name, age, and hungry columns for hungry dogs between the ages of two and seven. This query should also list these dogs in alphabetical order.

In [9]:
```
select_name_and_age_of_hungry_dogs_between_age_two_and_seven_in_alphabetical_order = "SELECT name,
c.execute(select_name_and_age_of_hungry_dogs_between_age_two_and_seven_in_alphabetical_order)
c.fetchall()

# Expected Output:
# [(None, 4, 1),
#  ('Clifford', 4, 1),
#  ('Lassie', 7, 1),
#  ('Scooby', 6, 1),
#  ('Snoopy', 3, 1)]
```

Out[9]:
```
[(None, 4, 1),
 ('Clifford', 4, 1),
 ('Lassie', 7, 1),
 ('Scooby', 6, 1),
 ('Snoopy', 3, 1)]
```

Summary

Great work! In this lab we practiced writing more complex SQL statements to not only query specific information but also define the quantity of results and the order of our results.