Dognition data exploration

Dognition is a way for dog owners and dog lovers all over the world to learn more about their dogs. Laboratory games have been done on dogs all over the country (USA) and all over the world for dog owners to learn more about their dogs. The dataset contains 6 relational tables. In this notebook, I will be performing some explorations with SQL extension to learn more about the dataset.

In this notebook, I will be applying some querying skills and attempt to answer some questions to test my SQL skill. The main objective of this notebook is to test my knowledge and skills. The analysis section will follow in another notebook.

To begin, load the sql library, connect to the Dognition database, and set the Dognition database as the default.

In [1]:
```
%load_ext sql
%sql mysql://studentuser:studentpw@localhost/dognitiondb
%sql USE dognitiondb
```

```
 * mysql://studentuser:***@localhost/dognitiondb
0 rows affected.
```

Out[1]:
```
[]
```

To explore the tables, I will first count the number of distinct dog and user id in different tables.

Questions 1: How many unique dog_guids and user_guids are there in the reviews and dogs table independently?

In [2]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid)
FROM reviews;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
1 rows affected.
```

Out[2]:

COUNT(DISTINCT dog_guid)
5991

In [3]:
```
%%sql
SELECT COUNT(DISTINCT user_guid)
FROM reviews;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
1 rows affected.
```

Out[3]:

COUNT(DISTINCT user_guid)
5586

In [4]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid)
FROM dogs;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
1 rows affected.
```

Out[4]:

COUNT(DISTINCT dog_guid)
35050

In [5]:
```
%%sql
SELECT COUNT(DISTINCT user_guid)
FROM dogs;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
1 rows affected.
```

Out[5]:

COUNT(DISTINCT user_guid)
30967

These counts indicate that:

- Many customers in both the reviews and the dogs table have multiple dogs
- There are many more unique dog_guids and user_guids in the dogs table than the reviews table

Practicing using join in SQL

Question 2: How many unique Golden Retrievers who live in North Carolina are there in the Dognition database (you should get 30)?

In [6]:
```
%%sql
SELECT u.state, d.breed, COUNT(DISTINCT d.dog_guid) as Count
FROM dogs d, users u
WHERE d.user_guid = u.user_guid
AND breed = 'Golden retriever'
GROUP BY state
having state = 'NC'
;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
1 rows affected.
```

Out[6]:

state	breed	Count
NC	Golden Retriever	30

Question 3: For which 3 dog breeds do we have the greatest amount of site_activity data, (as defined by non-NULL values in script_detail_id)(your answers should be "Mixed", "Labrador Retriever", and "Labrador Retriever-Golden Retriever Mix"?

In [7]:
```
%%sql
SELECT breed, COUNT(script_detail_id) AS activity
FROM dogs d, site_activities s
WHERE d.dog_guid = s.dog_guid
AND script_detail_id IS NOT NULL
GROUP BY breed
ORDER BY COUNT(script_detail_id) DESC
LIMIT 3;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
3 rows affected.
```

Out[7]:

breed	activity
Mixed	93415
Labrador Retriever	38804
Labrador Retriever-Golden Retriever Mix	27498

Question 4: Extract all the data from exam_answers that had test durations that were greater than the average duration for the "Yawn Warm-Up" game (you will get 11059 rows).

In [8]:
```
%%sql
SELECT *
FROM exam_answers
WHERE TIMESTAMPDIFF(minute, start_time, end_time) >
    (SELECT avg(TIMESTAMPDIFF(minute, start_time, end_time)) as Avgtime
    FROM exam_answers
    WHERE test_name = 'Yawn Warm-Up'
    AND TIMESTAMPDIFF(minute, start_time, end_time) > 0);
```

```
 * mysql://studentuser:***@localhost/dognitiondb
11059 rows affected.
```

Question 5: Use a NOT IN operator to determine how many unique dogs in the dog table are NOT in the "Working", "Sporting", or "Herding" breeding groups. You should get an answer of 7961.

In [9]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid)
FROM dogs
WHERE breed_group NOT IN ("Working", "Sporting", "Herding");
```

```
 * mysql://studentuser:***@localhost/dognitiondb
1 rows affected.
```

Out[9]:

COUNT(DISTINCT dog_guid)
7961

Question 6: Use a NOT EXISTS clause to examine all the users in the dogs table that are not in the users table (you should get 2 rows in your output).

In [10]:
```
%%sql
SELECT d.user_guid
FROM dogs d
WHERE NOT EXISTS
    (SELECT u.user_guid
    FROM users u
    WHERE u.user_guid = d.user_guid);
```

```
 * mysql://studentuser:***@localhost/dognitiondb
2 rows affected.
```

Out[10]:

user_guid
None
None

Similarly, we can do it the other way.

In [11]:
```
%%sql
SELECT u.user_guid
FROM users u
WHERE NOT EXISTS
    (SELECT d.user_guid
    FROM dogs d
    WHERE u.user_guid = d.user_guid);
```

```
 * mysql://studentuser:***@localhost/dognitiondb
2226 rows affected.
```

Question 7: Only join unique UserIDs from the users table with UserIDs from the dog table.

In [12]:
```
%%sql
SELECT DistinctUserID.user_guid, count(*) as nrows
FROM (SELECT DISTINCT user_guid
    FROM users) AS DistinctUserID
LEFT JOIN dogs d
ON DistinctUserID.user_guid = d.user_guid
GROUP BY DistinctUserID.user_guid
ORDER BY nrows desc
;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
33193 rows affected.
```

Question 8: Only join unique UserIDs from the users table with unique UserIDs from the dog table.

In [13]:
```
%%sql
SELECT DistinctUuserID.user_guid, DistinctDuserID.user_guid, count(*) as nrows
FROM (SELECT DISTINCT user_guid
    FROM users) AS DistinctUuserID
LEFT JOIN
    (SELECT DISTINCT user_guid
    FROM dogs) AS DistinctDuserID
ON DistinctUuserID.user_guid = DistinctDuserID.user_guid
GROUP BY DistinctUuserID.user_guid
ORDER BY nrows desc
;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
33193 rows affected.
```

Question 9: Adapt the query from Question 8 so that, in theory, you would retrieve a full list of all the DogIDs a user in the users table owns, with its accompagnying breed information whenever possible.

In [14]:
```
%%sql
SELECT DistinctUuserID.user_guid, DistinctDuserID.user_guid, DistinctDuserID.dog_
FROM (SELECT DISTINCT user_guid
    FROM users) AS DistinctUuserID
LEFT JOIN
    (SELECT DISTINCT user_guid, dog_guid, breed
    FROM dogs) AS DistinctDuserID
ON DistinctUuserID.user_guid = DistinctDuserID.user_guid
ORDER BY DistinctUuserID.user_guid
;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
37274 rows affected.
```

Question 10: Determine the number of unique user_guids who reside in the United States (abbreviated "US") and outside of the US.

In [15]:
```
%%sql
SELECT IF(cleaned_users.country='US','In US','Outside US') AS user_location,
count(cleaned_users.user_guid) AS num_guids
FROM (SELECT DISTINCT user_guid, country
FROM users
WHERE user_guid IS NOT NULL AND country IS NOT NULL) AS cleaned_users
GROUP BY user_location;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
2 rows affected.
```

Out[15]:

user_location	num_guids
In US	9356
Outside US	6905

Question 11: Write a query that uses a CASE statement to report the number of unique user_guids associated with customers who live in the United States and who are in the following groups of states:

Group 1: New York (abbreviated "NY") or New Jersey (abbreviated "NJ")
Group 2: North Carolina (abbreviated "NC") or South Carolina (abbreviated "SC")
Group 3: California (abbreviated "CA")
Group 4: All other states with non-null values

You should find 898 unique user_guids in Group1.

In [16]:
```
%%sql
SELECT COUNT(DISTINCT user_guid),
    CASE
        WHEN (state = 'NY' OR state = 'NJ') THEN 'Group 1'
        WHEN (state = 'NC' OR state = 'SC') THEN 'Group 2'
        WHEN (state = 'CA') THEN 'Group 3'
        ELSE 'Group 4'
    END AS Grouping
FROM users
WHERE country = 'US'
GROUP BY Grouping;
```

```
 * mysql://studentuser:***@localhost/dognitiondb
4 rows affected.
```

Out[16]:

COUNT(DISTINCT user_guid)	Grouping
898	Group 1
653	Group 2
1417	Group 3
6388	Group 4