You can also take advantage of aliases so that you don't have to write out the name of the tables each time. Here I will introduce another syntax for aliases that omits the AS completely. In this syntax, the alias is whatever word (or phrase, if you use quotation marks) follows immediately after the field or table name, separated by a space. So we could write:

```
SELECT d.dog_guid AS DogID, d.user_guid AS UserID, AVG(r.ratin
g) AS AvgRating,
       COUNT(r.rating) AS NumRatings, d.breed, d.breed_group,
d.breed_type
FROM dogs d, reviews r
GROUP BY d.user_guid
HAVING NumRatings >= 10
ORDER BY AvgRating DESC
LIMIT 200
```

I am tempted to tell you to run this query so that you will see what happens, but instead, I will explain what will happen and let you decide if you want to see what the output looks...and feels...like.

There is nothing built into the database table definitions that can instruct the server how to combine the tables on its own (remember, this is how relational databases save space and remain flexible). Further, the query as written does not tell the database how the two tables are related. As a consequence, rather than match up the two tables according to the values in the user_id and/or dog_id column, the database will do the only thing it knows how to do which is output every single combination of the records in the dogs table with the records in the reviews table. In other words, every single row of the dogs table will get paired with every single row of the reviews table. This is known as a Cartesian product. Not only will it be a heavy burden on the database to output a table that has the full length of one table multiplied times the full length of another (and frustrating to you, because the query would take a very long time to run), the output would be close to useless.

To prevent this from happening, tell the database how to relate the tables in the WHERE clause:

```
SELECT d.dog_guid AS DogID, d.user_guid AS UserID, AVG(r.ratin
g) AS AvgRating,
       COUNT(r.rating) AS NumRatings, d.breed, d.breed_group,
d.breed_type
FROM dogs d, reviews r
WHERE d.dog_guid=r.dog_guid
GROUP BY d.user_guid
HAVING NumRatings >= 10
ORDER BY AvgRating DESC
LIMIT 200
```

To be very careful and exclude any incorrect dog_guid or user_guid entries, you can include both shared columns in the WHERE clause:

```
SELECT d.dog_guid AS DogID, d.user_guid AS UserID, AVG(r.ratin
g) AS AvgRating,
       COUNT(r.rating) AS NumRatings, d.breed, d.breed_group,
d.breed_type
FROM dogs d, reviews r
WHERE d.dog_guid=r.dog_guid AND d.user_guid=r.user_guid
GROUP BY d.user_guid
HAVING NumRatings >= 10
ORDER BY AvgRating DESC
LIMIT 200
```

Try running this query now:

In [6]:
```
%%sql
SELECT d.dog_guid AS DogID, d.user_guid AS UserID, AVG(r.rating) AS Av
       COUNT(r.rating) AS NumRatings, d.breed, d.breed_group, d.breed_
FROM dogs d, reviews r
WHERE d.dog_guid=r.dog_guid AND d.user_guid=r.user_guid
GROUP BY d.user_guid
HAVING NumRatings >= 10
ORDER BY AvgRating DESC
LIMIT 200
```

ba71-058fbc01cf0b	ba71-058fbc01cf0b	2.5714	14	American Staffordshire Terrier Mix	None	Bree
fdbedef4-7144-11e5-ba71-058fbc01cf0b	ce982e46-7144-11e5-ba71-058fbc01cf0b	2.4615	13	Mixed	None	Mixe Bree Othe Don'
fdbec8ce-7144-11e5-ba71-058fbc01cf0b	ce980e3e-7144-11e5-ba71-058fbc01cf0b	2.4545	11	Basset Hound	Hound	Pure
fdbeff06-7144-11e5-ba71-058fbc01cf0b	ce9856f0-7144-11e5-ba71-058fbc01cf0b	2.4000	10	Bearded Collie	Herding	Pure
[illegible]	[illegible]					

The query should execute quickly. This would NOT have been the case if you did not include the WHERE clause to combine the two tables. If you accidentally request a Cartesian product from datasets with billions of rows, you could be waiting for your query output for days (and will probably get in trouble with your database administrator). So always remember to tell the database how to join your tables!

Let's examine our joined table a bit further. The joined table outputted by the query above should have 38 rows, despite the fact that we set our LIMIT at 200. The reason for this is that it turns out that a relatively small number of customers provided 10 or more reviews. If you

remove the HAVING and LIMIT BY clause from the query, you should end up with 389 rows. **Go ahead and try it:**

In [7]:
```
%%sql
SELECT d.dog_guid AS DogID, d.user_guid AS UserID, AVG(r.rating) AS Av
       COUNT(r.rating) AS NumRatings, d.breed, d.breed_group, d.breed_
FROM dogs d, reviews r
WHERE d.dog_guid=r.dog_guid AND d.user_guid=r.user_guid
GROUP BY d.user_guid
ORDER BY AvgRating DESC
```

```
389 rows affected.
```

Out[7]:

DogID	UserID	AvgRating	NumRatings	breed	breed_group	bre
fdbf178e-7144-11e5-ba71-058fbc01cf0b	ce986546-7144-11e5-ba71-058fbc01cf0b	9.0000	1	Airedale Terrier	Terrier	Pur
fdbf94de-7144-11e5-ba71-058fbc01cf0b	ce98cc34-7144-11e5-ba71-058fbc01cf0b	9.0000	2	Pembroke Welsh Corgi	Herding	Pur
fdc019c2-7144-11e5-ba71-058fbc01cf0b	ce99489e-7144-11e5-ba71-058fbc01cf0b	9.0000	1	Mixed	None	Mix Bre Oth Do

It's clear from looking at this output that (A) not many customers provided ratings, and (B) when they did, they usually were not very surprised by their dog's performance. Therefore, these ratings are probably not going to provide a lot of instructive insight into how to improve Dognition's completion rate. However, the ratings table still provides a great opportunity to illustrate the results of different types of joins.

To help prepare us for this:

Questions 1-4: How many unique dog_guids and user_guids are there in the reviews and dogs table independently?

In [9]:
```
%sql SELECT COUNT(DISTINCT dog_guid)
```

```
1 rows affected.
```

Out[9]:

COUNT(DISTINCT dog_guid)
5991

In [10]:
```
%%sql
SELECT COUNT(DISTINCT user_guid) FROM reviews;
```

```
1 rows affected.
```

Out[10]:

COUNT(DISTINCT user_guid)
5586

In [11]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid) FROM dogs;
```

```
1 rows affected.
```

Out[11]:

COUNT(DISTINCT dog_guid)
35050

In [12]:
```
%%sql
SELECT COUNT(DISTINCT user_guid) FROM dogs;
```

```
1 rows affected.
```

Out[12]:

COUNT(DISTINCT user_guid)
30967

These counts indicate some important things:

- 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
- There are many more unique dog_guids and user_guids in the reviews table than in the output of our inner join

Let's test one more thing.

Try the inner join query once with just the dog_guid or once with just the user_guid clause in the WHERE statement:

In [15]:
```
%%sql
SELECT d.user_guid AS UserID, d.dog_guid AS DogID, d.breed,
d.breed_type, d.breed_group FROM dogs d, complete_tests c
WHERE d.dog_guid=c.dog_guid;
```

193079 rows affected.

Out[15]:

UserID	DogID	breed	breed_type	breed_group
ce134e42-7144-11e5-ba71-058fbc01cf0b	fd27b272-7144-11e5-ba71-058fbc01cf0b	Labrador Retriever	Pure Breed	Sporting
ce134e42-7144-11e5-ba71-058fbc01cf0b	fd27b272-7144-11e5-ba71-058fbc01cf0b	Labrador Retriever	Pure Breed	Sporting
ce134e42-7144-11e5-ba71-058fbc01cf0b	fd27b272-7144-11e5-ba71-058fbc01cf0b	Labrador Retriever	Pure Breed	Sporting
ce134e42-7144-11e5-ba71-058fbc01cf0b	fd27b272-7144-11e5-ba71-058fbc01cf0b	Labrador Retriever	Pure Breed	Sporting

In [ ]:

When you run the query by joining on the dog_guid only, you still get 389 rows in your output. When you run the query by joining on the user_guid only, you get 5586 rows in your output. This means that:

- All of the user_guids in the reviews table are in the dogs table
- Only 389 of the over 5000 dog_guids in the reviews table are in the dogs table

Perhaps most importantly for our current purposes, these COUNT queries show you that *inner joins only output the data from rows that have equivalent values in both tables being joined*. If you wanted to include all the dog_guids or user_guids in one or both of the tables, you would have to use an outer join, which we will practice in the next lesson.

Try an inner join on your own.

Question 5: How would you extract the user_guid, dog_guid, breed, breed_type, and breed_group for all animals who completed the "Yawn Warm-up" game (you should get 20,845 rows if you join on dog_guid only)?

In [14]:
```
%%sql
SELECT d.user_guid AS UserID, d.dog_guid AS DogID, d.breed,
d.breed_type, d.breed_group FROM dogs d, complete_tests c
WHERE d.dog_guid=c.dog_guid AND test_name='Yawn Warm-up';
```

```
20845 rows affected.
```

Out[14]:

UserID	DogID	breed	breed_type	breed_group
ce134e42-7144-11e5-ba71-058fbc01cf0b	fd27b272-7144-11e5-ba71-058fbc01cf0b	Labrador Retriever	Pure Breed	Sporting
ce1353d8-7144-11e5-ba71-058fbc01cf0b	fd27b5ba-7144-11e5-ba71-058fbc01cf0b	Shetland Sheepdog	Pure Breed	Herding
ce135ab8-7144-11e5-ba71-058fbc01cf0b	fd27b6b4-7144-11e5-ba71-058fbc01cf0b	Golden Retriever	Pure Breed	Sporting
ce13507c-7144-11e5-ba71-058fbc01cf0b	fd27b79a-7144-11e5-ba71-058fbc01cf0b	Golden Retriever	Pure Breed	Sporting

2. Joining More than 2 Tables

In theory, you can join as many tables together as you want or need. To join multiple tables you take the same approach as we took when we were joining two tables together: list all the fields you want to extract in the SELECT statement, specify which table they came from in the SELECT statement, list all the tables from which you will need to extract the fields in the FROM statement, and then tell the database how to connect the tables in the WHERE statement.

To extract the user_guid, user's state of residence, user's zip code, dog_guid, breed, breed_type, and breed_group for all animals who completed the "Yawn Warm-up" game, you might be tempted to query:

```
SELECT c.user_guid AS UserID, u.state, u.zip, d.dog_guid AS
DogID, d.breed, d.breed_type, d.breed_group
FROM dogs d, complete_tests c, users u
WHERE d.dog_guid=c.dog_guid
   AND c.user_guid=u.user_guid
   AND c.test_name="Yawn Warm-up";
```

This query focuses the relationships primarily on the complete_tests table. However, it turns out that our Dognition dataset has only NULL values in the user_guid column of the complete_tests table. If you were to execute the query above, you would not get an error message, but your output would have 0 rows. However, the power of relational databases will

come in handy here. You can use the dogs table to link the complete_tests and users table (pay attention to the difference between the WHERE statement in this query vs. the WHERE statement in the query above):

```
SELECT d.user_guid AS UserID, u.state, u.zip, d.dog_guid AS
DogID, d.breed, d.breed_type, d.breed_group
FROM dogs d, complete_tests c, users u
WHERE d.dog_guid=c.dog_guid
   AND d.user_guid=u.user_guid
   AND c.test_name="Yawn Warm-up";
```

Of note, joins are very resource intensive, so try not to join unnecessarily. In general, the more joins you have to execute, the slower your query performance will be.

Question 6: How would you extract the user_guid, membership_type, and dog_guid of all the golden retrievers who completed at least 1 Dognition test (you should get 711 rows)?

In [18]:
```
%%sql
SELECT DISTINCT d.user_guid AS UserID, u.membership_type, d.dog_guid AS
DogID, d.breed
FROM dogs d, complete_tests c, users u WHERE d.dog_guid=c.dog_guid
AND d.user_guid=u.user_guid AND d.breed="golden retriever";
```

```
711 rows affected.
```

Out[18]:

UserID	membership_type	DogID	breed
ce135ab8-7144-11e5-ba71-058fbc01cf0b	1	fd27b6b4-7144-11e5-ba71-058fbc01cf0b	Golden Retriever
ce13507c-7144-11e5-ba71-058fbc01cf0b	1	fd27b79a-7144-11e5-ba71-058fbc01cf0b	Golden Retriever
ce1389d4-7144-11e5-ba71-058fbc01cf0b	1	fd27efb2-7144-11e5-ba71-058fbc01cf0b	Golden Retriever
ce21f122-7144-11e5-ba71-058fbc01cf0b	2	fd3d03fc-7144-11e5-ba71-058fbc01cf0b	Golden Retriever
ce220bb2-7144-11e5-ba71-058fbc01cf0b	2	fd3d10cc-7144-11e5-ba71-058fbc01cf0b	Golden Retriever
ce2237f4-7144-11e5-ba71-	2	fd3d3b24-7144-11e5-	Golden

Practice inner joining your own tables!

Question 7: How many unique Golden Retrievers who live in North Carolina are there in the Dognition database (you should get 30)?

In [5]:
```
%%sql
SELECT u.state AS state, d.breed AS breed, COUNT(DISTINCT d.dog_guid) 
WHERE d.user_guid=u.user_guid AND breed="Golden Retriever" and state="
GROUP BY state
```

1 rows affected.

Out[5]:

state	breed	COUNT(DISTINCT d.dog_guid)
NC	Golden Retriever	30

Question 8: How many unique customers within each membership type provided reviews (there should be 3208 in the membership type with the greatest number of customers, and 18 in the membership type with the fewest number of customers)?

In [16]:
```
%%sql
SELECT u.membership_type AS Membership, COUNT(DISTINCT r.user_guid) FR
WHERE u.user_guid=r.user_guid
GROUP BY membership_type
```

5 rows affected.

Out[16]:

Membership	COUNT(DISTINCT r.user_guid)
1	3208
2	1226
3	259
4	875
5	18

Question 9: 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 [17]:
```
%%sql
SELECT d.breed, COUNT(s.script_detail_id) AS activity
FROM dogs d, site_activities s
WHERE d.dog_guid=s.dog_guid AND s.script_detail_id IS NOT NULL GROUP B
ORDER BY activity DESC
LIMIT 3;
```

3 rows affected.

Out[17]:

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

Practice any other inner joins you would like to try here!

Practice any other inner joins you would like to try here!

In [ ]: