MySQL Exercise 10: Useful Logical Operators

There are a few more logical operators we haven't covered yet that you might find useful when designing your queries. Expressions that use logical operators return a result of "true" or "false", depending on whether the conditions you specify are met. The "true" or "false" results are usually used to determine which, if any, subsequent parts of your query will be run. We will discuss the IF operator, the CASE operator, and the order of operations within logical expressions in this lesson.

Begin by loading the sql library and database, and making the Dognition database your default database:

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

```
0 rows affected.
```

Out[1]: []

1. IF expressions

IF expressions are used to return one of two results based on whether inputs to the expressions meet the conditions you specify. They are frequently used in SELECT statements as a compact way to rename values in a column. The basic syntax is as follows:

```
IF([your conditions],[value outputted if conditions are met],[value outputted if co
nditions are NOT met])
```

So we could write:

```
SELECT created_at, IF(created_at<'2014-06-01','early_user','late_user') AS user_typ
e
FROM users
```

to output one column that provided the time stamp of when a user account was created, and a second column called user_type that used that time stamp to determine whether the user was an early or late user. User_type could then be used in a GROUP BY statement to segment summary calculations (in database systems that support the use of aliases in GROUP BY statements).

For example, since we know there are duplicate user_guids in the user table, we could combine a subquery with an IF statement to retrieve a list of unique user_guids with their classification as either an early or late user (based on when their first user entry was created):

```
SELECT cleaned_users.user_guid as UserID,
       IF(cleaned_users.first_account<'2014-06-01','early_user','late_user') AS use
r_type
FROM (SELECT user_guid, MIN(created_at) AS first_account
      FROM users
      GROUP BY user_guid) AS cleaned_users
```

We could then use a GROUP BY statement to count the number of unique early or late users:

```
SELECT IF(cleaned_users.first_account<'2014-06-01','early_user','late_user') AS use
r_type,
       COUNT(cleaned_users.first_account)
FROM (SELECT user_guid, MIN(created_at) AS first_account
      FROM users
      GROUP BY user_guid) AS cleaned_users
GROUP BY user_type
```

Try it yourself:

In [6]:
```
%%sql
SELECT cleaned_users.user_guid AS UserID,
IF(cleaned_users.first_account < '2014-06-10', 'early_user', 'late_user') AS u
ser_type,
COUNT(cleaned_users.first_account)
FROM (SELECT user_guid, MIN(created_at) AS first_account
      FROM users
      GROUP BY user_guid ) cleaned_users
GROUP BY user_type
```

2 rows affected.

Out[6]:

UserID	user_type	COUNT(cleaned_users.first_account)
ce134492-7144-11e5-ba71-058fbc01cf0b	early_user	14508
ce472c1c-7144-11e5-ba71-058fbc01cf0b	late_user	18685

Question 1: Write a query that will output distinct user_guids and their associated country of residence from the users table, excluding any user_guids or countries that have NULL values. You should get 16,261 rows in your result.

In [9]:
```
%%sql
SELECT DISTINCT user_guid, country
FROM users
WHERE country IS NOT NULL
LIMIT 20
```

20 rows affected.

Out[9]:

user_guid	country
ce134e42-7144-11e5-ba71-058fbc01cf0b	US
ce1353d8-7144-11e5-ba71-058fbc01cf0b	US
ce135ab8-7144-11e5-ba71-058fbc01cf0b	US
ce13507c-7144-11e5-ba71-058fbc01cf0b	US
ce135e14-7144-11e5-ba71-058fbc01cf0b	US
ce13615c-7144-11e5-ba71-058fbc01cf0b	US
ce135f2c-7144-11e5-ba71-058fbc01cf0b	US
ce136a1c-7144-11e5-ba71-058fbc01cf0b	US
ce136ac6-7144-11e5-ba71-058fbc01cf0b	US
ce136c24-7144-11e5-ba71-058fbc01cf0b	US
ce136e36-7144-11e5-ba71-058fbc01cf0b	US
ce136ee0-7144-11e5-ba71-058fbc01cf0b	US
ce136f94-7144-11e5-ba71-058fbc01cf0b	US
ce134be0-7144-11e5-ba71-058fbc01cf0b	US
ce1371a6-7144-11e5-ba71-058fbc01cf0b	US
ce1373ae-7144-11e5-ba71-058fbc01cf0b	US
ce13750c-7144-11e5-ba71-058fbc01cf0b	US
ce1375b6-7144-11e5-ba71-058fbc01cf0b	US
ce1377b4-7144-11e5-ba71-058fbc01cf0b	US
ce137700-7144-11e5-ba71-058fbc01cf0b	US

Question 2: Use an IF expression and the query you wrote in Question 1 as a subquery to determine the number of unique user_guids who reside in the United States (abbreviated "US") and outside of the US.

In [12]:
```
%%sql
SELECT IF(cleaned_users.country = 'US', 'In US', 'Outside US')AS user_loacatio
n, COUNT(cleaned_users.user_guid) AS num_guids
FROM
(SELECT DISTINCT user_guid, country
FROM users
WHERE country IS NOT NULL AND user_guid IS NOT NULL) AS cleaned_users
GROUP BY user_loacation
```

```
2 rows affected.
```

Out[12]:

user_loacation	num_guids
In US	9356
Outside US	6905

Single IF expressions can only result in one of two specified outputs, but multiple IF expressions can be nested to result in more than two possible outputs. When you nest IF expressions, it is important to encase each IF expression--as well as the entire IF expression put together--in parentheses.

For example, if you examine the entries contained in the non-US countries category, you will see that many users are associated with a country called "N/A." "N/A" is an abbreviation for "Not Applicable"; it is not a real country name. We should separate these entries from the "Outside of the US" category we made earlier. We could use a nested query to say whenever "country" does not equal "US", use the results of a second IF expression to determine whether the outputed value should be "Not Applicable" or "Outside US." The IF expression would look like this:

```
IF(cleaned_users.country='US','In US', IF(cleaned_users.country='N/A','Not Applicab
le','Outside US'))
```

Since the second IF expression is in the position within the IF expression where you specify "value outputted if conditions are not met," its two possible outputs will only be considered if cleaned_users.country='US' is evaluated as false.

The full query to output the number of unique users in each of the three groups would be:

```
SELECT IF(cleaned_users.country='US','In US',
          IF(cleaned_users.country='N/A','Not Applicable','Outside US')) AS
US_user,
      count(cleaned_users.user_guid)
FROM (SELECT DISTINCT user_guid, country
      FROM users
      WHERE country IS NOT NULL) AS cleaned_users
GROUP BY US_user
```

Try it yourself. You should get 5,642 unique user_guids in the "Not Applicable" category, and 1,263 users in the "Outside US" category.

In [13]:
```
%%sql
SELECT IF(cleaned_users.country = 'US', 'In US',
          IF(cleaned_users.country = 'N/A', 'Not Applicable', 'Outside US'))AS
 user_loacation,
COUNT(cleaned_users.user_guid) AS num_guids
FROM
(SELECT DISTINCT user_guid, country
FROM users
WHERE country IS NOT NULL AND user_guid IS NOT NULL) AS cleaned_users
GROUP BY user_loacation
```

3 rows affected.

Out[13]:

user_loacation	num_guids
In US	9356
Not Applicable	5642
Outside US	1263

The IF function is not supported by all database platforms, and some spell the function as IIF rather than IF, so be sure to double-check how the function works in the platform you are using.

If nested IF expressions seem confusing or hard to read, don't worry, there is a better function available for situations when you want to use conditional logic to output more than two groups. That function is called CASE.

2. CASE expressions

The main purpose of CASE expressions is to return a singular value based on one or more conditional tests. You can think of CASE expressions as an efficient way to write a set of IF and ELSEIF statements. There are two viable syntaxes for CASE expressions. If you need to manipulate values in a current column of your data, you would use this syntax:

```
CASE
 WHEN [condition set 1] THEN [result you want when the conditions in set 1 are met]
 WHEN [condition set 2] THEN [result you want when the conditions in set 2 are met]
 WHEN [condition set 3] THEN [result you want when the conditions in set 3 are met]
 ...(can include as many condition sets as you want)
 ELSE [result you want when none of the condition sets are met]
END
```

Using this syntax, our nested IF statement from above could be written as:

```
SELECT CASE WHEN cleaned_users.country="US" THEN "In US"
            WHEN cleaned_users.country="N/A" THEN "Not Applicable"
            ELSE "Outside US"
            END AS US_user,
      count(cleaned_users.user_guid)
FROM (SELECT DISTINCT user_guid, country
      FROM users
      WHERE country IS NOT NULL) AS cleaned_users
GROUP BY US_user
```

Go ahead and try it:

In [2]:
```
%%sql
SELECT CASE WHEN cleaned_users.country = 'US' THEN 'In US'
            WHEN cleaned_users.country = 'N/A' THEN 'Not Applicable'
            ELSE 'Outside US'
            END AS US_user,
        COUNT(cleaned_users.user_guid)
FROM (SELECT DISTINCT user_guid, country
     FROM users
     WHERE country IS NOT NULL) AS cleaned_users
GROUP BY US_user
```

3 rows affected.

Out[2]:

US_user	COUNT(cleaned_users.user_guid)
In US	9356
Not Applicable	5642
Outside US	1263

Since our query does not require manipulation of any of the values in the country column, though, we could also take advantage of this syntax, which is slightly more compact:

```
CASE column_name or expression
    WHEN [value 1] THEN [result you want when row=value 1]
    WHEN [value 2] THEN [result you want when row=value 2]
    WHEN [value 3] THEN [result you want when row=value 3]
    ... (can include as many values as you want)
    ELSE [result you want when row does not equal any of the specified values]
END
```

Our query written in this syntax would look like this:

```
SELECT CASE cleaned_users.country
            WHEN "US" THEN "In US"
            WHEN "N/A" THEN "Not Applicable"
            ELSE "Outside US"
            END AS US_user,
      count(cleaned_users.user_guid)
FROM (SELECT DISTINCT user_guid, country
      FROM users
      WHERE country IS NOT NULL) AS cleaned_users
GROUP BY US_user
```

Try this query as well:

In [16]:
```
%%sql
SELECT CASE cleaned_users.country
        WHEN 'US' THEN 'In US'
        WHEN 'N/A' THEN 'Not Applicable'
        ELSE 'Outside US'
        END AS US_user,
        COUNT(cleaned_users.user_guid)
FROM (SELECT DISTINCT user_guid, country
      FROM users
      WHERE country IS NOT NULL) AS cleaned_users
GROUP BY US_user
```

```
3 rows affected.
```

Out[16]:

US_user	COUNT(cleaned_users.user_guid)
In US	9356
Not Applicable	5642
Outside US	1263

There are a couple of things to know about CASE expressions:

- Make sure to include the word END at the end of the expression
- CASE expressions do not require parentheses
- ELSE expressions are optional
- If an ELSE expression is omitted, NULL values will be outputted for all rows that do not meet any of the conditions stated explicitly in the expression
- CASE expressions can be used anywhere in a SQL statement, including in GROUP BY, HAVING, and ORDER BY clauses or the SELECT column list.

You will find that CASE statements are useful in many contexts. For example, they can be used to rename or revise values in a column.

Question 3: Write a query using a CASE statement that outputs 3 columns: dog_guid, dog_fixed, and a third column that reads "neutered" every time there is a 1 in the "dog_fixed" column of dogs, "not neutered" every time there is a value of 0 in the "dog_fixed" column of dogs, and "NULL" every time there is a value of anything else in the "dog_fixed" column. Limit your results for troubleshooting purposes.

In [6]:
```
%%sql
SELECT cleaned_dogs.dog_guid, cleaned_dogs.dog_fixed ,
        CASE cleaned_dogs.dog_fixed
        WHEN 1 THEN 'neutered'
        WHEN 0 THEN 'not neutered'
        ELSE 'NULL'
        END AS 'if fixed'
FROM (SELECT DISTINCT dogs.dog_guid, dog_fixed
        FROM dogs
        WHERE dog_guid IS NOT NULL) AS cleaned_dogs
LIMIT 20
```

20 rows affected.

Out[6]:

dog_guid	dog_fixed	if fixed
fd27b272-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27b5ba-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27b6b4-7144-11e5-ba71-058fbc01cf0b	0	not neutered
fd27b79a-7144-11e5-ba71-058fbc01cf0b	0	not neutered
fd27b86c-7144-11e5-ba71-058fbc01cf0b	0	not neutered
fd27b948-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27ba1a-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27bbbe-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27c1c2-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27c5be-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27c74e-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27c7d0-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27c852-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27c8d4-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27c956-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27cb72-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27cd98-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27ce1a-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27cea6-7144-11e5-ba71-058fbc01cf0b	1	neutered
fd27cf28-7144-11e5-ba71-058fbc01cf0b	1	neutered

You can also use CASE statements to standardize or combine several values into one.

Question 4: We learned that NULL values should be treated the same as "0" values in the exclude columns of the dogs and users tables. Write a query using a CASE statement that outputs 3 columns: dog_guid, exclude, and a third column that reads "exclude" every time there is a 1 in the "exclude" column of dogs and "keep" every time there is any other value in the exclude column. Limit your results for troubleshooting purposes.

In [7]:
```
%%sql
DESCRIBE dogs
```

```
21 rows affected.
```

Out[7]:

Field	Type	Null	Key	Default	Extra
gender	varchar(255)	YES		None	
birthday	varchar(255)	YES		None	
breed	varchar(255)	YES		None	
weight	int(11)	YES		None	
dog_fixed	tinyint(1)	YES		None	
dna_tested	tinyint(1)	YES		None	
created_at	datetime	NO		None	
updated_at	datetime	NO		None	
dimension	varchar(255)	YES		None	
exclude	tinyint(1)	YES		None	
breed_type	varchar(255)	YES		None	
breed_group	varchar(255)	YES		None	
dog_guid	varchar(60)	YES	MUL	None	
user_guid	varchar(60)	YES	MUL	None	
total_tests_completed	varchar(255)	YES		None	
mean_iti_days	varchar(255)	YES		None	
mean_iti_minutes	varchar(255)	YES		None	
median_iti_days	varchar(255)	YES		None	
median_iti_minutes	varchar(255)	YES		None	
time_diff_between_first_and_last_game_days	varchar(255)	YES		None	
time_diff_between_first_and_last_game_minutes	varchar(255)	YES		None	

Question 5: Re-write your query from Question 4 using an IF statement instead of a CASE statement.

In [10]:
```
%%sql
SELECT cleaned_dogs.dog_guid, cleaned_dogs.exclude,
        IF (cleaned_dogs.exclude = 1, 'exclude', 'keep') AS 'if excluded'
FROM (SELECT DISTINCT dog_guid, exclude
        FROM dogs
        WHERE dog_guid IS NOT NULL) AS cleaned_dogs
LIMIT 20
```

20 rows affected.

Out[10]:

dog_guid	exclude	if excluded
fd27b272-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27b5ba-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27b6b4-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27b79a-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27b86c-7144-11e5-ba71-058fbc01cf0b	1	exclude
fd27b948-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27ba1a-7144-11e5-ba71-058fbc01cf0b	1	exclude
fd27bbbe-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27c1c2-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27c5be-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27c74e-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27c7d0-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27c852-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27c8d4-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27c956-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27cb72-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27cd98-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27ce1a-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27cea6-7144-11e5-ba71-058fbc01cf0b	None	keep
fd27cf28-7144-11e5-ba71-058fbc01cf0b	None	keep

Case expressions are also useful for breaking values in a column up into multiple groups that meet specific criteria or that have specific ranges of values.

Question 6: Write a query that uses a CASE expression to output 3 columns: dog_guid, weight, and a third column that reads...
"very small" when a dog's weight is 1-10 pounds
"small" when a dog's weight is greater than 10 pounds to 30 pounds
"medium" when a dog's weight is greater than 30 pounds to 50 pounds
"large" when a dog's weight is greater than 50 pounds to 85 pounds
"very large" when a dog's weight is greater than 85 pounds
Limit your results for troubleshooting purposes.

Remember that when you use AND to define values between two boundaries, you need to include the variable name in all clauses that define the conditions of the values you want to extract. In other words, you could use this combined clause in your query: "WHEN weight>10 AND weight<=30 THEN "small" …but this combined clause would cause an error: "WHEN weight>10 AND <=30 THEN "small"

In [13]:
```
%%sql
SELECT dog_guid, weight,
        CASE WHEN (weight >= 1 AND weight <= 10) THEN 'very small'
             WHEN (weight > 10 AND weight <= 30) THEN 'small'
             WHEN (weight > 30 AND weight <= 50) THEN 'medium'
             WHEN (weight > 50 AND weight <= 85) THEN 'large'
             WHEN (weight > 85 ) THEN 'very large'
        ELSE 'NULL'
        END AS weight_grouped
FROM dogs
LIMIT 20
```

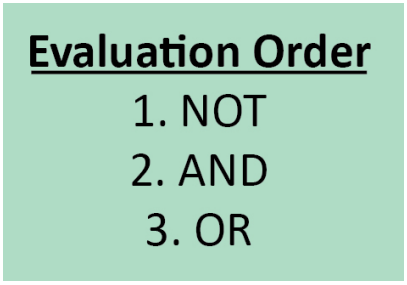
20 rows affected.

Out[13]:

dog_guid	weight	weight_grouped
fd27b272-7144-11e5-ba71-058fbc01cf0b	50	medium
fd27b5ba-7144-11e5-ba71-058fbc01cf0b	20	small
fd27b6b4-7144-11e5-ba71-058fbc01cf0b	70	large
fd27b79a-7144-11e5-ba71-058fbc01cf0b	70	large
fd27b86c-7144-11e5-ba71-058fbc01cf0b	190	very large
fd27b948-7144-11e5-ba71-058fbc01cf0b	60	large
fd27ba1a-7144-11e5-ba71-058fbc01cf0b	190	very large
fd27bbbe-7144-11e5-ba71-058fbc01cf0b	50	medium
fd27c1c2-7144-11e5-ba71-058fbc01cf0b	70	large
fd27c5be-7144-11e5-ba71-058fbc01cf0b	0	NULL
fd27c74e-7144-11e5-ba71-058fbc01cf0b	40	medium
fd27c7d0-7144-11e5-ba71-058fbc01cf0b	60	large
fd27c852-7144-11e5-ba71-058fbc01cf0b	20	small
fd27c8d4-7144-11e5-ba71-058fbc01cf0b	50	medium
fd27c956-7144-11e5-ba71-058fbc01cf0b	30	small
fd27cb72-7144-11e5-ba71-058fbc01cf0b	40	medium
fd27cd98-7144-11e5-ba71-058fbc01cf0b	30	small
fd27ce1a-7144-11e5-ba71-058fbc01cf0b	20	small
fd27cea6-7144-11e5-ba71-058fbc01cf0b	10	very small
fd27cf28-7144-11e5-ba71-058fbc01cf0b	60	large

3. Pay attention to the order of operations within logical expressions

As you started to see with the query you wrote in Question 6, CASE expressions often end up needing multiple AND and OR operators to accurately describe the logical conditions you want to impose on the groups in your queries. You must pay attention to the order in which these operators are included in your logical expressions, because unless parentheses are included, the NOT operator is always evaluated before an AND operator, and an AND operator is always evaluated before the OR operator.



When parentheses are included, the expressions within the parenthese are evaluated first. That means this expression:

```
CASE WHEN "condition 1" OR "condition 2" AND "condition 3"...
```

will lead to different results than this expression:

```
CASE WHEN "condition 3" AND "condition 1" OR "condition 2"...
```

or this expression:

```
CASE WHEN ("condition 1" OR "condition 2") AND "condition 3"...
```

In the first case you will get rows that meet condition 2 and 3, or condition 1. In the second case you will get rows that meet condition 1 and 3, or condition 2. In the third case, you will get rows that meet condition 1 or 2, and condition 3.

Let's see a concrete example of how the order in which logical operators are evaluated affects query results.

Question 7: How many distinct dog_guids are found in group 1 using this query?

```
SELECT COUNT(DISTINCT dog_guid),
CASE WHEN breed_group='Sporting' OR breed_group='Herding' AND exclude!='1' THEN "gr
oup 1"
     ELSE "everything else"
     END AS groups
FROM dogs
GROUP BY groups
```

In [14]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid),
CASE WHEN breed_group = 'Sporting' OR breed_group = 'Herding' AND exclude != 1
 THEN "group 1"
ELSE "everything else"
END AS groups
FROM dogs
GROUP BY groups
```

2 rows affected.

Out[14]:

COUNT(DISTINCT dog_guid)	groups
30179	everything else
4871	group 1

Question 8: How many distinct dog_guids are found in group 1 using this query?

```
SELECT COUNT(DISTINCT dog_guid),
CASE WHEN exclude!='1' AND breed_group='Sporting' OR breed_group='Herding' THEN "gr
oup 1"
     ELSE "everything else"
     END AS group_name
FROM dogs
GROUP BY group_name
```

In [17]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid),
CASE WHEN exclude != 1 AND breed_group = 'Sporting' OR breed_group = 'Herding'
 THEN "group 1"
ELSE "everything else"
END AS group_name
FROM dogs
GROUP BY group_name
```

2 rows affected.

Out[17]:

COUNT(DISTINCT dog_guid)	group_name
31589	everything else
3461	group 1

Question 9: How many distinct dog_guids are found in group 1 using this query?

```
SELECT COUNT(DISTINCT dog_guid),
CASE WHEN exclude!='1' AND (breed_group='Sporting' OR breed_group='Herding') THEN
"group 1"
     ELSE "everything else"
     END AS group_name
FROM dogs
GROUP BY group_name
```

In [18]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid),
CASE WHEN exclude!= 1 AND (breed_group='Sporting' OR breed_group='Herding') TH
EN "group 1"
     ELSE "everything else"
     END AS group_name
FROM dogs
GROUP BY group_name
```

```
2 rows affected.
```

Out[18]:

COUNT(DISTINCT dog_guid)	group_name
35004	everything else
46	group 1

So make sure you always pay attention to the order in which your logical operators are listed in your expressions, and whenever possible, include parentheses to ensure that the expressions are evaluated in the way you intend!

Let's practice some more IF and CASE statements

In order to make it easier to practice SQL queries with meaningful examples before we learned how to join tables, I added extra columns to the "dogs" table that were not in the original Dognition database. These extra columns included the "total_tests_completed" field and multiple inter-test-interval ("iti") summary fields. **Please do NOT try to use these extra fields in the query exercises below. Since you now know how to join tables, we will practice writing queries as if you only had the data provided in the original Dognition database.**

Question 10: For each dog_guid, output its dog_guid, breed_type, number of completed tests, and use an IF statement to include an extra column that reads "Pure_Breed" whenever breed_type equals 'Pure Breed" and "Not_Pure_Breed" whenever breed_type equals anything else. LIMIT your output to 50 rows for troubleshooting. HINT: you will need to use a join to complete this query.

In [22]:
```
%%sql
SELECT d.dog_guid, d.breed_type, COUNT(c.created_at) AS numtests,
IF (d.breed_type = 'Pure Breed', 'pure_breed', 'not_pure_breed') AS pure_breed
FROM dogs d, complete_tests c
WHERE d.dog_guid = c.dog_guid
GROUP BY d.dog_guid, d.breed_type, pure_breed
LIMIT 20
```

```
20 rows affected.
```

Out[22]:

dog_guid	breed_type	numtests	pure_breed
fd27b272-7144-11e5-ba71-058fbc01cf0b	Pure Breed	21	pure_breed
fd27b5ba-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27b6b4-7144-11e5-ba71-058fbc01cf0b	Pure Breed	2	pure_breed
fd27b79a-7144-11e5-ba71-058fbc01cf0b	Pure Breed	11	pure_breed
fd27b86c-7144-11e5-ba71-058fbc01cf0b	Pure Breed	31	pure_breed
fd27b948-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27ba1a-7144-11e5-ba71-058fbc01cf0b	Pure Breed	27	pure_breed
fd27bbbe-7144-11e5-ba71-058fbc01cf0b	Mixed Breed/ Other/ I Don't Know	20	not_pure_breed
fd27c1c2-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27c5be-7144-11e5-ba71-058fbc01cf0b	Cross Breed	20	not_pure_breed
fd27c74e-7144-11e5-ba71-058fbc01cf0b	Cross Breed	14	not_pure_breed
fd27c7d0-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27c852-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27c8d4-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27c956-7144-11e5-ba71-058fbc01cf0b	Cross Breed	11	not_pure_breed
fd27cb72-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27cd98-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed
fd27ce1a-7144-11e5-ba71-058fbc01cf0b	Pure Breed	7	pure_breed
fd27cea6-7144-11e5-ba71-058fbc01cf0b	Mixed Breed/ Other/ I Don't Know	2	not_pure_breed

fd27cf28-7144-11e5-ba71-058fbc01cf0b	Pure Breed	20	pure_breed

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 [32]:
```
 %%sql
SELECT COUNT(DISTINCT u.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 group_name
FROM users u
WHERE country = 'US' AND state IS NOT NULL
GROUP BY group_name
```

4 rows affected.

Out[32]:

COUNT(DISTINCT u.user_guid)	group_name
898	group 1
653	group 2
1417	group 3
6388	group 4

Question 12: Write a query that allows you to determine how many unique dog_guids are associated with dogs who are DNA tested and have either stargazer or socialite personality dimensions. Your answer should be 70.

In [34]:
```
%%sql
SELECT COUNT(DISTINCT dog_guid)
FROM dogs
WHERE dna_tested = 1 AND ( dimension = 'stargazer' OR dimension = 'socialite')
```

1 rows affected.

Out[34]:

COUNT(DISTINCT dog_guid)
70