MySQL_Exercise_11_Queries_that_Test_Relationships_Between_Test_Co

December 2, 2020

1 MySQL Exercise 11: Queries that Test Relationships Between Test Completion and Dog Characteristics

This lesson we are going to integrate all the SQL syntax we've learned so far to start addressing questions in our Dognition Analysis Plan. I summarized the reasons having an analysis plan is so important in the "Start with an Analysis Plan" video accompanying this week's materials. Analysis plans ensure that you will address questions that are relevant to your business objectives as quickly and efficiently as possible. The quickest way to narrow in the factors in your analysis plan that are likely to create new insights is to combine simple SQL calculations with visualization programs, like Tableau, to identify which factors under consideration have the strongest effects on the business metric you are tasked with improving. You can then design more nuanced statistical models in other software, such as R, based on the factors you have confirmed are likely to be important for understanding and changing your business metric.

I describe a method for designing analysis plans in the Data Visualization and Communication with Tableau course earlier in this Specialization. I call that method Structured Pyramid Analysis Plans, or "sPAPs". I have provided a skeleton of an sPAP for the Dognition data set with the materials for this course that I will use as a road map for the queries we will design and practice in the next two lessons. To orient you, the SMART goal of the analysis project is at the top of the pyramid. This is a specific, measurable, attainable, relevant, and time-bound version of the general project objective, which is to make a recommendation to Dognition about what they could do to increase the number of tests customers complete. The variables you will use to assess the goal should be filled out right under where the SMART goal is written. Then under those variables, you will see ever-widening layers of categories and sub-categories of issues that will be important to analyze in order to achieve your SMART goal.

In this lesson, we will write queries to address the issues in the left-most branch of the sPAP. These issues all relate to "Features of Dogs" that could potentially influence the number of tests the dogs will ultimately complete. We will spend a lot of time discussing and practicing how to translate analysis questions described in words into queries written in SQL syntax.

To begin, load the sql library and database, and make the Dognition database your default database:

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

1.1 1. Assess whether Dognition personality dimensions are related to the number of tests completed

The first variable in the Dognition sPAP we want to investigate is Dognition personality dimensions. Recall from the "Meet Your Dognition Data" video and the written description of the Dognition Data Set included with the Week 2 materials that Dognition personality dimensions represent distinct combinations of characteristics assessed by the Dognition tests. It is certainly plausible that certain personalities of dogs might be more or less likely to complete tests. For example, "einstein" dogs might be particularly likely to complete a lot of tests.

To test the relationship between Dognition personality dimensions and test completion totals, we need a query that will output a summary of the number of tests completed by dogs that have each of the Dognition personality dimensions. The features you will need to include in your query are foreshadowed by key words in this sentence. First, the fact that you need a summary of the number of tests completed suggests you will need an aggregation function. Next, the fact that you want a different summary for each personality dimension suggests that you will need a GROUP BY clause. Third, the fact that you need a "summary of the number of tests completed" rather than just a "summary of the tests completed" suggests that you might have to have multiple stages of aggegrations, which in turn might mean that you will need to use a subquery.

Let's build the query step by step.

Question 1: To get a feeling for what kind of values exist in the Dognition personality dimension column, write a query that will output all of the distinct values in the dimension column. Use your relational schema or the course materials to determine what table the dimension column is in. Your output should have 11 rows.

[ ]:
```
%%sql
SELECT DISTINCT dimension
FROM dogs;
```

The results of the query above illustrate there are NULL values (indicated by the output value "none") in the dimension column. Keep that in mind in case it is relevant to future queries.

We want a summary of the total number of tests completed by dogs with each personality dimension. In order to calculate those summaries, we first need to calculate the total number of tests completed by each dog. We can achieve this using a subquery. The subquery will require data from both the dogs and the complete_tests table, so the subquery will need to include a join. We are only interested in dogs who have completed tests, so an inner join is appropriate in this case.

Question 2: Use the equijoin syntax (described in MySQL Exercise 8) to write a query that will output the Dognition personality dimension and total number of tests completed by each unique DogID. This query will be used as an inner subquery in the next question. LIMIT your output to 100 rows for troubleshooting purposes.

[ ]:
```
%%sql
SELECT d.dog_guid, d.dimension, COUNT(c.created_at)
FROM dogs d, complete_tests c
WHERE d.dog_guid = c.dog_guid
GROUP BY d.dog_guid
LIMIT 100;
```

Question 3: Re-write the query in Question 2 using traditional join syntax (described in MySQL Exercise 8).

[ ]:
```
%%sql
SELECT d.dog_guid, d.dimension, COUNT(c.created_at)
FROM dogs d JOIN complete_tests c
ON d.dog_guid = c.dog_guid
GROUP BY d.dog_guid
LIMIT 100;
```

Now we need to summarize the total number of tests completed by each unique DogID within each Dognition personality dimension. To do this we will need to choose an appropriate aggregation function for the count column of the query we just wrote.

Question 4: To start, write a query that will output the average number of tests completed by unique dogs in each Dognition personality dimension. Choose either the query in Question 2 or 3 to serve as an inner query in your main query. If you have trouble, make sure you use the appropriate aliases in your GROUP BY and SELECT statements.

[ ]:
```
%%sql
SELECT dimension, AVG(tests.cant) as avg_tests FROM
(SELECT d.dog_guid, d.dimension, COUNT(c.created_at) as cant
FROM dogs d JOIN complete_tests c
ON d.dog_guid = c.dog_guid
GROUP BY d.dog_guid) as tests
GROUP BY dimension;
```

You should retrieve an output of 11 rows with one of the dimensions labeled "None" and another labeled "" (nothing is between the quotation marks).

Question 5: How many unique DogIDs are summarized in the Dognition dimensions labeled "None" or ""? (You should retrieve values of 13,705 and 71)

[ ]:
```
%%sql
SELECT COUNT(DISTINCT dog_id), dimension
FROM (SELECT d.dog_guid as dog_id, d.dimension as dimension
     FROM dogs d JOIN complete_tests c
     ON d.dog_guid = c.dog_guid
    WHERE dimension = "" or dimension IS NULL
      GROUP BY dog_id) as dogs_tests
GROUP BY dimension;
```

It makes sense there would be many dogs with NULL values in the dimension column, because we learned from Dognition that personality dimensions can only be assigned after the initial "Dognition Assessment" is completed, which is comprised of the first 20 Dognition tests. If dogs did not complete the first 20 tests, they would retain a NULL value in the dimension column.

The non-NULL empty string values are more curious. It is not clear where those values would come from.

Question 6: To determine whether there are any features that are common to all dogs that have non-NULL empty strings in the dimension column, write a query that outputs the breed, weight, value in the "exclude" column, first or minimum time stamp in the complete_tests table, last or maximum time stamp in the complete_tests table, and total number of tests completed by each unique DogID that has a non-NULL empty string in the dimension column.

[ ]:
```
%%sql
SELECT d.breed, d.weight, d.exclude, MIN(c.created_at),
        MAX(c.created_at), COUNT(c.created_at) as completeTests
FROM dogs d JOIN complete_tests c
ON d.dog_guid = c.dog_guid
WHERE d.dimension = ""
GROUP BY d.dog_guid;
```

A quick inspection of the output from the last query illustrates that almost all of the entries that have non-NULL empty strings in the dimension column also have "exclude" flags of 1, meaning that the entries are meant to be excluded due to factors monitored by the Dognition team. This provides a good argument for excluding the entire category of entries that have non-NULL empty strings in the dimension column from our analyses.

Question 7: Rewrite the query in Question 4 to exclude DogIDs with (1) non-NULL empty strings in the dimension column, (2) NULL values in the dimension column, and (3) values of "1" in the exclude column. NOTES AND HINTS: You cannot use a clause that says d.exclude does not equal 1 to remove rows that have exclude flags, because Dognition clarified that both NULL values and 0 values in the "exclude" column are valid data. A clause that says you should only include values that are not equal to 1 would remove the rows that have NULL values in the exclude column, because NULL values are never included in equals statements (as we learned in the join lessons). In addition, although it should not matter for this query, practice including parentheses with your OR and AND statements that accurately reflect the logic you intend. Your results should return 402 DogIDs in the ace dimension and 626 dogs in the charmer dimension.

[ ]:
```
%%sql
SELECT dimension, AVG(tests.cant) as avg_tests, COUNT(DISTINCT dog_id)
FROM
    (SELECT d.dog_guid as dog_id, d.dimension as dimension,
            COUNT(c.created_at) as cant
    FROM dogs d JOIN complete_tests c
    ON d.dog_guid = c.dog_guid
```

```
    WHERE (dimension IS NOT NULL AND dimension != "")
         AND (exclude = 0 OR exclude IS NULL)
    GROUP BY d.dog_guid) as tests
GROUP BY dimension;
```

The results of Question 7 suggest there are not appreciable differences in the number of tests completed by dogs with different Dognition personality dimensions. Although these analyses are not definitive on their own, these results suggest focusing on Dognition personality dimensions will not likely lead to significant insights about how to improve Dognition completion rates.

1.2 2. Assess whether dog breeds are related to the number of tests completed

The next variable in the Dognition sPAP we want to investigate is Dog Breed. We will run one analysis with Breed Group and one analysis with Breed Type.

First, determine how many distinct breed groups there are.

Questions 8: Write a query that will output all of the distinct values in the breed_group field.

[ ]:
```
%%sql
SELECT DISTINCT breed_group
FROM dogs;
```

You can see that there are NULL values in the breed_group field. Let's examine the properties of these entries with NULL values to determine whether they should be excluded from our analysis.

Question 9: Write a query that outputs the breed, weight, value in the "exclude" column, first or minimum time stamp in the complete_tests table, last or maximum time stamp in the complete_tests table, and total number of tests completed by each unique DogID that has a NULL value in the breed_group column.

[ ]:
```
%%sql
SELECT d.breed, d.weight, d.exclude, MIN(c.created_at),
        MAX(c.created_at), COUNT(c.created_at)
FROM dogs d JOIN complete_tests c
ON d.dog_guid = c.dog_guid
WHERE breed_group IS NULL
GROUP BY d.dog_guid;
```

There are a lot of these entries and there is no obvious feature that is common to all of them, so at present, we do not have a good reason to exclude them from our analysis. Therefore, let's move on to question 10 now….

Question 10: Adapt the query in Question 7 to examine the relationship between breed_group and number of tests completed. Exclude DogIDs with values of "1" in the exclude column. Your results should return 1774 DogIDs in the Herding breed group.

[ ]:
```
%%sql
SELECT breed, AVG(tests.cant) as avg_tests, COUNT(DISTINCT dog_id)
FROM
    (SELECT d.dog_guid as dog_id, COUNT(c.created_at) as cant,
             d.breed_group as breed
    FROM dogs d JOIN complete_tests c
    ON d.dog_guid = c.dog_guid
    WHERE (exclude = 0 OR exclude IS NULL)
    GROUP BY d.dog_guid) as tests
GROUP BY breed;
```

The results show there are non-NULL entries of empty strings in breed_group column again. Ignoring them for now, Herding and Sporting breed_groups complete the most tests, while Toy breed groups complete the least tests. This suggests that one avenue an analyst might want to explore further is whether it is worth it to target marketing or certain types of Dognition tests to dog owners with dogs in the Herding and Sporting breed_groups. Later in this lesson we will discuss whether using a median instead of an average to summarize the number of completed tests might affect this potential course of action.

Question 11: Adapt the query in Question 10 to only report results for Sporting, Hound, Herding, and Working breed_groups using an IN clause.

[ ]:
```
%%sql
SELECT breed, AVG(tests.cant) as avg_tests, COUNT(DISTINCT dog_id)
FROM
    (SELECT d.dog_guid as dog_id, COUNT(c.created_at) as cant,
             d.breed_group as breed
    FROM dogs d JOIN complete_tests c
    ON d.dog_guid = c.dog_guid
    WHERE (exclude = 0 OR exclude IS NULL)
    GROUP BY d.dog_guid) as tests
WHERE breed IN ("Sporting", "Hound", "Herding", "Working")
GROUP BY breed;
```

Next, let's examine the relationship between breed_type and number of completed tests.

Questions 12: Begin by writing a query that will output all of the distinct values in the breed_type field.

[ ]:
```
%%sql
SELECT DISTINCT breed_type
FROM dogs;
```

Question 13: Adapt the query in Question 7 to examine the relationship between breed_type and number of tests completed. Exclude DogIDs with values of "1" in the exclude column. Your results should return 8865 DogIDs in the Pure Breed group.

[ ]:
```
%%sql
SELECT breed_type, AVG(tests.cant) as avg_tests, COUNT(DISTINCT dog_id)
```

```
FROM
    (SELECT d.dog_guid as dog_id, d.breed_type as breed_type,
            COUNT(c.created_at) as cant
    FROM dogs d JOIN complete_tests c
    ON d.dog_guid = c.dog_guid
    WHERE (exclude = 0 OR exclude IS NULL)
    GROUP BY d.dog_guid) as tests
GROUP BY breed_type;
```

There does not appear to be an appreciable difference between number of tests completed by dogs of different breed types.

1.3 3. Assess whether dog breeds and neutering are related to the number of tests completed

To explore the results we found above a little further, let's run some queries that relabel the breed_types according to "Pure_Breed" and "Not_Pure_Breed".

Question 14: For each unique DogID, output its dog_guid, breed_type, number of completed tests, and use a CASE statement to include an extra column with a string 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.

[ ]:
```
%%sql
SELECT d.dog_guid, d.breed_type, COUNT(c.created_at) as completeTests,
    CASE
        WHEN d.breed_type = "Pure Breed" THEN "Pure_Breed"
        ELSE "Not Pure_Breed"
    END as pure_breed
FROM dogs d JOIN complete_tests c
ON d.dog_guid = c.dog_guid
GROUP BY d.dog_guid
LIMIT 50;
```

Question 15: Adapt your queries from Questions 7 and 14 to examine the relationship between breed_type and number of tests completed by Pure_Breed dogs and non_Pure_Breed dogs. Your results should return 8336 DogIDs in the Not_Pure_Breed group.

[ ]:
```
%%sql
SELECT tests.pure_breed as pureBreed, AVG(tests.cant) as avg_tests,
        COUNT(DISTINCT tests.dog_id)
FROM
    (SELECT d.dog_guid as dog_id, d.breed_type, COUNT(c.created_at) as cant,
        CASE
            WHEN d.breed_type = "Pure Breed" THEN "Pure_Breed"
```

```
            ELSE "Not Pure_Breed"
        END as pure_breed
    FROM dogs d JOIN complete_tests c
    ON d.dog_guid = c.dog_guid
     WHERE d.exclude IS NULL or d.exclude = 0
    GROUP BY d.dog_guid) as tests
GROUP BY pure_breed;
```

Question 16: Adapt your query from Question 15 to examine the relationship between breed_type, whether or not a dog was neutered (indicated in the dog_fixed field), and number of tests completed by Pure_Breed dogs and non_Pure_Breed dogs. There are DogIDs with null values in the dog_fixed column, so your results should have 6 rows, and the average number of tests completed by non-pure-breeds who are neutered is 10.5681.

[ ]:
```
%%sql
SELECT tests.pure_breed as pureBreed, neutered, AVG(tests.cant) as avg_tests,
        COUNT(DISTINCT tests.dog_id)
FROM
    (SELECT d.dog_guid as dog_id, d.breed_type,
            COUNT(c.created_at) as cant, d.dog_fixed as neutered,
        CASE
            WHEN d.breed_type = "Pure Breed" THEN "Pure_Breed"
            ELSE "Not Pure_Breed"
        END as pure_breed
    FROM dogs d JOIN complete_tests c
    ON d.dog_guid = c.dog_guid
     WHERE d.exclude IS NULL or d.exclude = 0
    GROUP BY d.dog_guid) as tests
GROUP BY pure_breed, neutered;
```

These results suggest that although a dog's breed_type doesn't seem to have a strong relationship with how many tests a dog completed, neutered dogs, on average, seem to finish 1-2 more tests than non-neutered dogs. It may be fruitful to explore further whether this effect is consistent across different segments of dogs broken up according to other variables. If the effects are consistent, the next step would be to seek evidence that could clarify whether neutered dogs are finishing more tests due to traits that arise when a dog is neutered, or instead, whether owners who are more likely to neuter their dogs have traits that make it more likely they will want to complete more tests.

1.4 4. Other dog features that might be related to the number of tests completed, and a note about using averages as summary metrics

Two other dog features included in our sPAP were speed of game completion and previous behavioral training. Examing the relationship between the speed of game completion and number of games completed is best achieved through creating a scatter plot with a best fit line and/or running a statistical regression analysis. It is possible to achieve the statistical regression analysis through

very advanced SQL queries, but the strategy that would be required is outside the scope of this course. Therefore, I would recommend exporting relevant data to a program like Tableau, R, or Matlab in order to assess the relationship between the speed of game completion and number of games completed.

Unfortunately, there is no field available in the Dognition data that is relevant to a dog's previous behavioral training, so more data would need to be collected to examine whether previous behavioral training is related to the number of Dognition tests completed.

One last issue I would like to address in this lesson is the issue of whether an average is a good summary to use to represent the values of a certain group. Average calculations are very sensitive to extreme values, or outliers, in the data. This video provides a nice demonstration of how sensitive averages can be:

http://www.statisticslectures.com/topics/outliereffects/

Ideally, you would summarize the data in a group using a median calculation when you either don't know the distribution of values in your data or you already know that outliers are present (the definition of median is covered in the video above). Unfortunately, medians are more computationally intensive than averages, and there is no pre-made function that allows you to calculate medians using SQL. If you wanted to calculate the median, you would need to use an advanced strategy such as the ones described here:

https://www.periscopedata.com/blog/medians-in-sql.html

Despite the fact there is no simple way to calculate medians using SQL, there is a way to get a hint about whether average values are likely to be wildly misleading. As described in the first video (http://www.statisticslectures.com/topics/outliereffects/), strong outliers lead to large standard deviation values. Fortunately, we *CAN* calculate standard deviations in SQL easily using the STDDEV function. Therefore, it is good practice to include standard deviation columns with your outputs so that you have an idea whether the average values outputted by your queries are trustworthy. Whenever standard deviations are a significant portion of the average values of a field, and certainly when standard deviations are larger than the average values of a field, it's a good idea to export your data to a program that can handle more sophisticated statistical analyses before you interpret any results too strongly.

Let's practice including standard deviations in our queries and interpretting their values.

Question 17: Adapt your query from Question 7 to include a column with the standard deviation for the number of tests completed by each Dognition personality dimension.

[ ]:
```
%%sql
SELECT dimension, AVG(tests.cant) as avg_tests,
        COUNT(DISTINCT dog_id),STDDEV(cant)
FROM
    (SELECT d.dog_guid as dog_id, d.dimension as dimension,
            COUNT(c.created_at) as cant
    FROM dogs d JOIN complete_tests c
    ON d.dog_guid = c.dog_guid
    WHERE (dimension IS NOT NULL AND dimension != "")
         AND (exclude = 0 OR exclude IS NULL)
```

```
    GROUP BY d.dog_guid) as tests
GROUP BY dimension;
```

The standard deviations are all around 20-25% of the average values of each personality dimension, and they are not appreciably different across the personality dimensions, so the average values are likely fairly trustworthy. Let's try calculating the standard deviation of a different measurement.

Question 18: Write a query that calculates the average amount of time it took each dog breed_type to complete all of the tests in the exam_answers table. Exclude negative durations from the calculation, and include a column that calculates the standard deviation of durations for each breed_type group:

[ ]:
```
%%sql
SELECT breed_type,
       AVG(TIMESTAMPDIFF(MINUTE, e.start_time,e.end_time)) as AVG_Time,
       STDDEV(TIMESTAMPDIFF(MINUTE, e.start_time,e.end_time)) as STD_Dev
FROM dogs d JOIN exam_answers e
ON d.dog_guid = e.dog_guid
WHERE TIMESTAMPDIFF(MINUTE, e.start_time,e.end_time) > 0
GROUP BY breed_type;
```

This time many of the standard deviations have larger magnitudes than the average duration values. This suggests there are outliers in the data that are significantly impacting the reported average values, so the average values are not likely trustworthy. These data should be exported to another program for more sophisticated statistical analysis.

In the next lesson, we will write queries that assess the relationship between testing circumstances and the number of tests completed. Until then, feel free to practice any additional queries you would like to below!

[ ]:
```

```