SQL Module Project:

Iowa Liquor Store

Spencer Fairbairn, Allison Bruhn, Portia Gifford,

Chidi Nzerem, Jacob Pulitzer, William Tardif

MSBC 5070, Section 203

Micah McGee

Project Goal:

Can alcohol distributors predict upcoming demand in order to implement appropriate sales strategies?

Background:

Since 2012, Iowa has kept track of all alcohol sales from the state to private retailers. The reason for this is that Iowa is an alcohol beverage control state, which means that for private retailers to be able to sell alcohol, they first have to obtain a Class "E" liquor licence. Once retailers have obtained a Class "E" liquor license they can begin to purchase alcoholic beverages from the state. The Iowa Liquor Sales dataset includes, but is not limited to, variables such as date sold, volume sold, sale dollars, county, alcohol type, and the vendor name.

Problem:

The State of Iowa is considering implementing new strategies to raise alcohol sales in quarters in which they are seeing lower sales numbers. We first would like to see how the alcohol sales change year over year in our top three selling counties to be able to understand why they are not purchasing in certain quarters of the year. We then propose building a model to predict the volume demanded of each category of liquor in the top counties.

Target Feature:

By looking at the Volume of liters sold and the category of alcohol we are able to predict demand of different brands and types of alcohol within Polk County, Iowa.

Descriptive Statistics:
The dependent variable used between all four questions is the total volume sold in liters. After performing descriptive analytics on each of the four questions below, our findings include:

1. ***What are the top three best selling counties in terms of average volume sold in liters?***
    a. Polk (3.574184182E7 liters sold)
    b. Linn (1.525935946E7 liters sold)
    c. Scott (1.170415437E7 liters sold)
2. ***What is the best selling category of alcohol in terms of average volume sold in liters?***
    a. Imported Distilled Spirit Specialty (61.17 average Liters sold)
3. ***What is the best selling Zip Code within our top 3 counties in terms of average volume?***
    a. Polk = 50314 average liters = 5,278,259.1
    b. Linn = 52402 average liters = 5,260,300.93
    c. Scott = 52807 average liters = 3,664,656.01
4. ***What impact do seasons have on the volume of liquor sold?***
    a. Quarter 1: 4.041530743E7 liters sold
    b. Quarter 2: 4.83468158E7 liters sold
    c. Quarter 3: 3.975683501E7 liters sold
    d. Quarter 4: 4.466371525E7 liters sold

Based on our descriptive analytics, we found the county with the highest demand of liquor in liters is Polk county, with the highest demanding Zip Code being 50314. Overall, Iowa sees the highest demand for liquor during quarter 2 and quarter 4. We also see the best selling category of alcohol is *Imported Distilled Spirit Specialty*, with an average of 61.7 liters sold per quarter.

Predictive Analysis:
Can we predict the volume demand of each category of liquor in Polk County?

Based on our question, we want to be able to predict the volume sold in liters of each category of liquor that we are going to distribute to Polk county. We specifically wanted to be able to predict volume for each quarter. As some quarters may require more volume of a certain category than others. Polk county consumes the highest volume of liquor than any other county. By being able to predict volume for Polk based on quarter we can assure that year after year demand can be met.
Our value we want to predict is going to be Volume in Liters. Our most important features from our data set when running this model are going to be Date, Volume, country and Category. From this we also want to extract quarter from the Date to be able to see the results by quarter.

Results from our model
R-squared = .8928

For our model, the R-squared metric is a .8928 meaning approximately 90% of the variance, or change, in volume of liquor sold in Polk county can be predicted by time of year (quarter) and on the type of liquor being sold.
Our coefficients are represented by our category weights. Based on our model the categories of liquor are represented by both positive and negative values. The higher the value the more significant the category. Using an equation we are able to predict the demand for the most popular categories of liquor being sold. Based on these coefficients, we predict that in Polk county the demand for Iowa Distillery Whiskies will be approximately 49.48 liters per quarter, the demand for Imported Distilled Spirit Specialty will be 40.91 liters per quarter, and the demand for Special Order Items will be 30.78 liters per quarter.

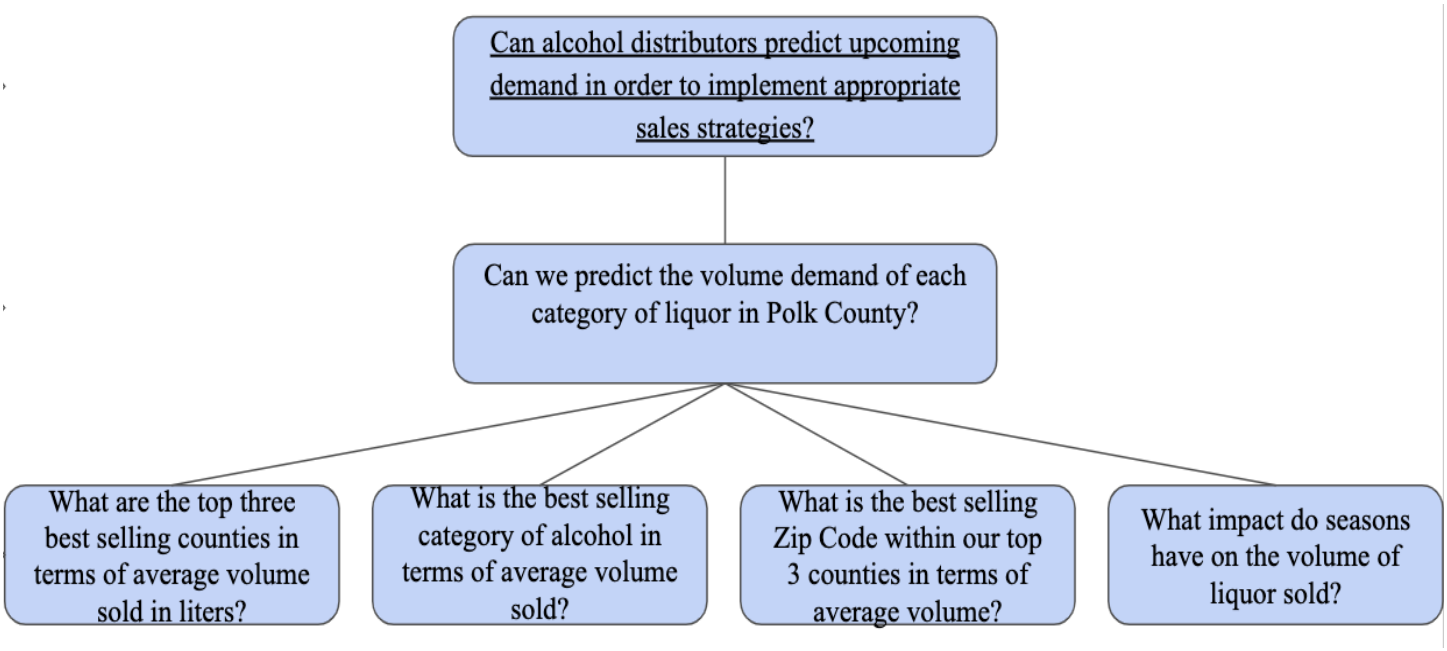
Business Actions:
Based on our analysis we can improve our business operation because we were able to pinpoint the highest type of alcohol sold as well as predict the quarters where alcohol was lowest in order to increase distributor push during that time.

Some specific business action that can be taken to increase profitability would be:

- To increase inventory in the top selling liquor which is an imported distilled spirit specialty in quarters when alcohol consumption is lowest (quarter 1) in order to raise profit merging.
- Another profitable business action would be to increase inventory of the lowest selling alcohol which is cocktails in quarters when alcohol consumption is highest which is the 4thquarter

Next Steps:
Based on the results we have gathered from our predictive and descriptive analysis, we believe that we would greatly benefit from the Iowa census data on population statistics. By using this data we would be able to drive a more meaningful analysis of what the average volume of alcohol is per person based on factors such as Gender, Age, and Annual Income. The reason for this is that from our research we can see that 14% of the population of Iowa lives in Polk county meaning that alcohol sales are going to be higher.

Question Tree



Descriptive Analytics Appendix:

Descriptive Question Answers:

1. The top three selling counties in terms of average volume sold:
   a. Polk (3.574184182E7 liters sold)

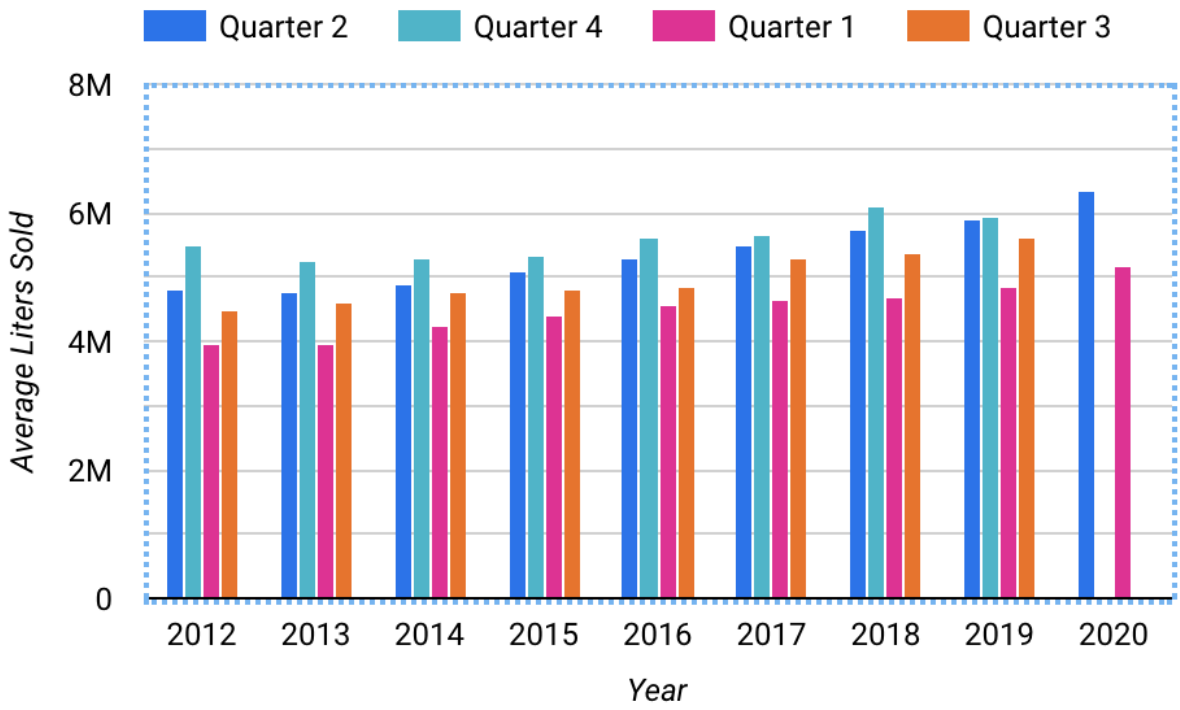
b. Linn (1.525935946E7 liters sold)
c. Scott (1.170415437E7 liters sold)

2. The best-selling category (type of Liquor) of alcohol in terms of average volume sold:
   a. Liquor Type: Imported Distilled Spirit Specialty (61.17 average Liters sold)

3. The best-selling Zip Code within our top 3 counties in terms of average volume:
   a. Polk = 50314 average liters = 5278259.1
   b. Linn = 52402 average liters = 5260300.93
   c. Scott = 52807 average liters = 3664656.0

4. The impact different quarters have on the volume of liquor sold:
   a. Quarter 1: 4.041530743E7 liters sold
   b. Quarter 2: 4.83468158E7 liters sold
   c. Quarter 3: 3.975683501E7 liters sold
   d. Quarter 4: 4.466371525E7 liters sold

Row	county_zip	average_liters_sold
1	polk (50314)	5278259.1
2	polk (50320)	5266901.73
3	linn (52402)	5260300.93
4	johnson (52240)	4722023.17
5	story (50010)	4412553.84
6	scott (52807)	3664656.01
7	johnson (52241)	3087486.69
8	polk (50311)	3087443.15
9	black hawk (50613)	3066178.27
10	dubuque (52001)	2777246.25



Descriptive Tutorials

How to write SQL code to retrieve desired data:

We use the 'SELECT' and 'FROM' statements to retrieve values of a row or column from a table. The 'FROM' statement calls the dataset we want to work with and brings it into the query. The 'SELECT' statement allows us to pull out specific elements within that dataset.

1. For example, say we have a database on Iowa Liquor sales and we want to see information relating to 'bottles_sold' and 'vendor_name' .

```
38 SELECT
39   alc.vendor_name,
40   alc.bottles_sold
41 FROM
42   `bigquery-public-data.iowa_liquor_sales.sales` as alc
43
44 Limit 100
```

   a. Using the 'as' statement allows us to call the dataset something else instead of always using its original, long name.
      i. The word "LIMIT" means that we are limiting our output to show us 100 rows.
      ii. If we were to input an asterisk(*) after 'SELECT', that would mean we are calling all the columns from the table.
      iii. Example
         1. SELECT *
            a. This would give us the whole dataset
         2. Otherwise,
            a. SELECT "bottles_sold", "vendor_name"
               i. This would only give us the "bottles_sold" and "vendor_name" columns within the dataset.

Job information | Results | JSON | Execution details

Row	vendor_name	bottles_sold
1	PERNOD RICARD USA	5
2	OLE SMOKY DISTILLERY LLC	3
3	BACARDI USA INC	12
4	LUXCO INC	48
5	DIAGEO AMERICAS	6

   b. From running this code, the result you'll get is the following:

How to write SQL code to aggregate data using various functions (SUM(), MIN(), MAX(), average(), etc.):

SUM()

Using the SUM() function allows us to get the total sum of a numeric column within the dataset. This function only works with numeric values and doesn't work with strings. This function helps us quickly see the overall total of every value within the given row. An example of how to do this is as follows:

1. First you would load up a dataset using the 'SELECT' and 'FROM' function (see how to do that above under "How to Write SQL code to retrieve data")
    a. You would also need to make sure that the column you want to analyze is a numeric column and not a string. The easiest way to do this would be to load up and run the data, then analyze all the columns to see which ones you can use.
    b. Below is the code you would use to do this. I am using the public dataset available in BigQuery called "iowa_liquor_sales".

```
SELECT
  *
FROM
  `bigquery-public-data.iowa_liquor_sales.sales`

```

2. After running this, the whole dataset should be available to view under the "results" tab, allowing you to see which things you can / want to analyze.

    i. As you can see, there's a lot of information in this dataset, so making sure you use the 'SELECT' function to choose certain columns and the 'LIMIT' function to only show a certain amount of outputs will really help the analysis process.

2. Now that you have your dataset loaded, choose which functions you want to analize. I am going to go with the "store_name" and the "bottles_sold".

Job information | Results | JSON | Execution details

Row	invoice_and_item_number	date	store_number	store_name	address	city	zip_code	store_location
1	S14533000031	2013-09-12	4829	Central City 2	1501 MICHIGAN AVE	DES MOINES	50314	POINT (-93.613739 4
2	S03516500066	2012-01-10	2603	Hy-Vee Wine and Spirits / Bettendorf	2890 DEVILS GLEN ROAD	BETTENDORF	52722	POINT (-90.483701 4
3	S24945300007	2015-04-07	2619	Hy-Vee Wine and Spirits / WDM	1725 74TH ST	WEST DES MOINES	50266	POINT (-93.808855 4
4	S04000000032	2012-02-09	2510	Hy-Vee Drugstore #3 / Cedar Rapids	2405 MT VERNON RD SE	CEDAR RAPIDS	52403	POINT (-91.632647 4
5	S32672500009	2016-06-06	3814	Costco Wholesale #788	7205 Mills Civic Pkwy	West Des Moines	50266	POINT (-93.806489 4

Rows per page: 100 | 1 - 100 of 19008336 | First page | Last page

3. After I do this, I want to get the sum of all the bottles_sold (which I renamed to Total_Sold). This is where I would add the SUM() function in.

```
SELECT
  alc.store_name,
  SUM(alc.bottles_sold) as Total_Sold,
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` as alc

```

    a. However, doing this will produce an error as the function right now is trying to match up the rows of "alc.store_name", which is a lot, to the rows of "Sum(alc.bottles_sold)", which is only one. To fix this, however, you just have to add a GROUP BY statement below it which tells the function to only take the

sum of each specific store, rather than the whole dataset. This allows us to see which store sold the most in relation to the other stores in the dataset.

```
SELECT
  alc.store_name,
  SUM(alc.bottles_sold) as Total_Sold,
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` as alc
GROUP BY
  alc.store_name
```

4. Now the function will run and there are no errors. The final thing you can add, which isn't necessary, is an 'ORDER BY' function which allows you to see which store sold the most at the top, along with the rest in descending order.

```
SELECT
  alc.store_name,
  SUM(alc.bottles_sold) as Total_Sold,
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` as alc
GROUP BY
  alc.store_name
ORDER BY
  Total_Sold DESC
```

   a. This is the result you should get if you run this code as well.

Job information | Results | JSON | Execution details

Row	store_name	Total_Sold
1	Hy-Vee #3 / BDI / Des Moines	5040137
2	Central City 2	4109263
3	Hy-Vee Wine and Spirits / Iowa City	2450294
4	Sam's Club 8162 / Cedar Rapids	1896452
5	Sam's Club 6344 / Windsor Heights	1681995

Using the SUM() function is a great way to quickly do descriptive analytics and see results in an instant. Mastering this function can make your life much simpler when trying to analyze a big dataset.

AVG()

The AVG() (Average) function in SQL allows us to get the average of all the numeric values within a column. This function is similar to the SUM() function in the sense that it won't work with string values, but instead of giving the

overall sum, it gives you the overall mean of the column. If you wanted to relate it to another column in the dataset, you must add some more lines of code to make it work. An example of how to do this is as follows, I'll be using the same dataset and columns as the examples above.

1. To see how to load the data into SQL and use the 'SELECT' function to filter out the data you want, scroll up to parts a) and b) of the SUM() function.
    b. When running this function, it can quickly give you the mean of a specific column. An example of this is as follows.

```
SELECT
  AVG(alc.bottles_sold) as Total_Sold,
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` as alc

```

Result:

Job information | Results | JSON | Execution details

Row	Total_Sold
1	10.265031405168699

    c. Just like the SUM() function, when trying to use the AVG() function alongside another column of the dataset, however, you must add more code in order for the Query to run. Below is an example of the error that'll occur.

```
SELECT
  alc.store_name,
  AVG(alc.bottles_sold) as Total_Sold,
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` as alc

```

    d. In order for the function to run, you must add the 'GROUP BY' statement. This tells the function to only take the mean of bottles sold for each specific store, rather than the whole dataset. I also added in an 'ORDER BY' statement which allows me to quickly see which store sold the most bottles on average compared to the rest.

```
SELECT
  alc.store_name,
  AVG(alc.bottles_sold) as Total_Sold,
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` as alc
GROUP BY
  alc.store_name
ORDER BY
  Total_Sold DESC

```

    e. After running this code, you should get the desired outcome, which is seeing all the stores and their average bottles sold in descending order.

Job information | Results | JSON | Execution details

Row	store_name	Total_Sold
1	Templeton Distilling LLC	920.1111111111111
2	Paradise Distilling Company	232.07142857142858
3	Costco Wholesale #788 / WDM	174.72369668246446
4	Costco Wholesale #1325 / Davenport	134.7768924302789
5	Foundry Distilling Company	132.5

The Count() function tells us how many times a certain criteria shows up in the data. Using this in the most basic form (just seeing how many items are in a column) may be unnecessary, as you'll be able to see how many rows there are and most of the time, that'll also be how many item instances there are in the column. But if you use this column to see how many items are in the range of a certain criteria, then we see just how useful this function can be. An example of this is as follows, I'll once again be using the Iowa Liquor Sales dataset and picking out the "store_name" and the "bottles_sold" columns.

1. First, just like the other examples, you must load in the dataset using the 'FROM' statement and then selecting which columns you want to analyze using the 'SELECT' statement. From there, you can easily use the COUNT() statement to see how many instances there are of a certain column. In this example, I'll be seeing how many stores there are.

```
1 SELECT
2   COUNT(alc.store_name)
3 FROM
4   `bigquery-public-data.iowa_liquor_sales.sales` as alc
5
6
```

Result:

Job information | Results | JSON | Execution details

Row	f0_
1	19008336

2. This information, although possibly useful, may not be exactly what we want. Instead, we can give the count function parameters to see how many stores sold X amount of bottles. This would require us to add the "alc.bottles_sold" under the 'SELECT' statement and change the COUNT() function to analyze "alc.bottles_sold" instead of "alc.store_name". Once we do this, however, just like the other functions, we must now add a 'GROUP BY' statement to have the count of bottles sold be associated to its store. You can also add a 'ORDER BY' statement to see what store has sold the most bottles by count.

```
SELECT
  alc.store_name,
  COUNT(alc.bottles_sold) as Number_of_bottles_sold
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` as alc
GROUP BY
  alc.store_name
ORDER BY
   Number_of_bottles_sold DESC
```

Results:

Job information | Results | JSON | Execution details

Row	store_name	Number_of_bottles_sold
1	Hy-Vee #3 / BDI / Des Moines	164731
2	Central City 2	133987
3	Central City Liquor, Inc.	128765
4	Hy-Vee Wine and Spirits / Iowa City	117667
5	Hy-Vee Food Store #1 / Mason City	100331

How to group data using SQL:

The GROUP BY statement is great to use when coding because you are telling the DBMS (Database Management Systems) what fields you want to aggregate your data by grouping one or more columns. These aggregate functions include count, max, min, sum, and average.

1. To use this statement, first start with the SELECT and FROM statement,
   a. SELECT is telling the DBMS what fields/columns to look at. FROM is telling the DBMS what rows to filter out. (Note: In order to use the GROUP BY statement, we must include what we want in SELECT with using aggregates)

```
SELECT
  sales.category_name as category,
  avg(sales.volume_sold_liters) as liters
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` AS sales
```

   b. In this SELECT statement, we are telling BigQuery that we want to look at the category name (i.e. the type of alcohol) as well as take the average sales of liters sold from the dataset. Then we

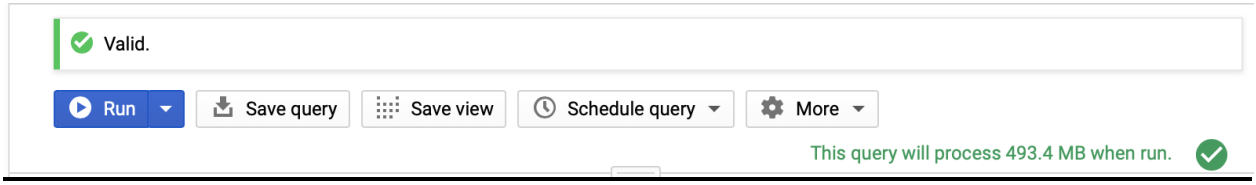
are naming the category_name column as category and the average volume_sold_liters column as liters.

c. In the FROM statement, we are telling BigQuery to extract the Iowa Liquor Sales from the public database and calling it "sales" as short.

2. Next, we add our GROUP BY statement
   a. The GROUP BY statement will tell BigQuery which of the columns in our SELECT statement we want the query to tabulate together. In this case, we want to combine the category names together with the average liters sold per category name.
   b. Although we have 2 columns in our SELECT statement, the volume sold in liters is being averaged, based on the question we are answering.

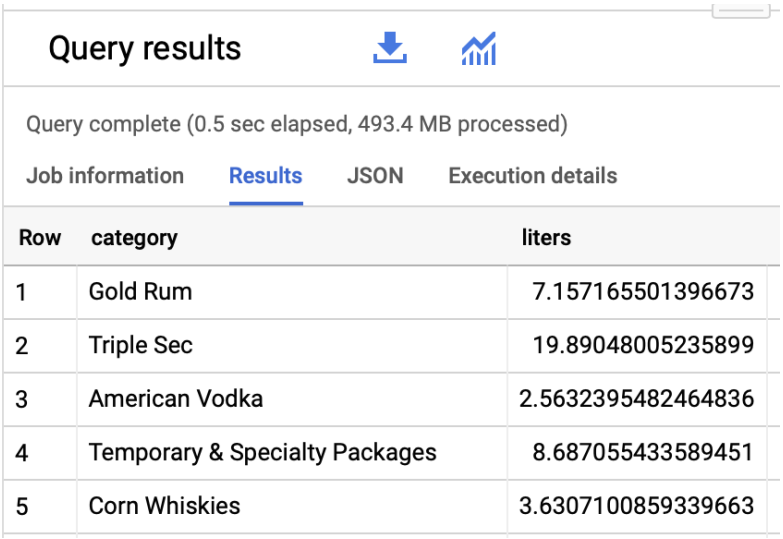
```
SELECT
  sales.category_name as category,
  avg(sales.volume_sold_liters) as liters
FROM
  `bigquery-public-data.iowa_liquor_sales.sales` AS sales
GROUP BY
  category
```

3. Then click on **"Run"** and the query will process the code given and then give the query results. To know when the code is good and is able to run, you will see a green checkmark on the right side of the screen or it will say "Valid" at the bottom of the query editor. If there isn't a checkmark, that means there is an issue with the code and what's great about BigQuery is it will tell you what and where the error is within the code.



4. After clicking **"Run"**, the query results will be displayed and you can check to make sure your code did what you wanted it to do. In this case, it did exactly what we wanted. By using the **"GROUP BY"**

statement, it took every item in the database with the same category name and grouped that data together. Then, BigQuery took the volume of liters sold and averaged that data per category name.



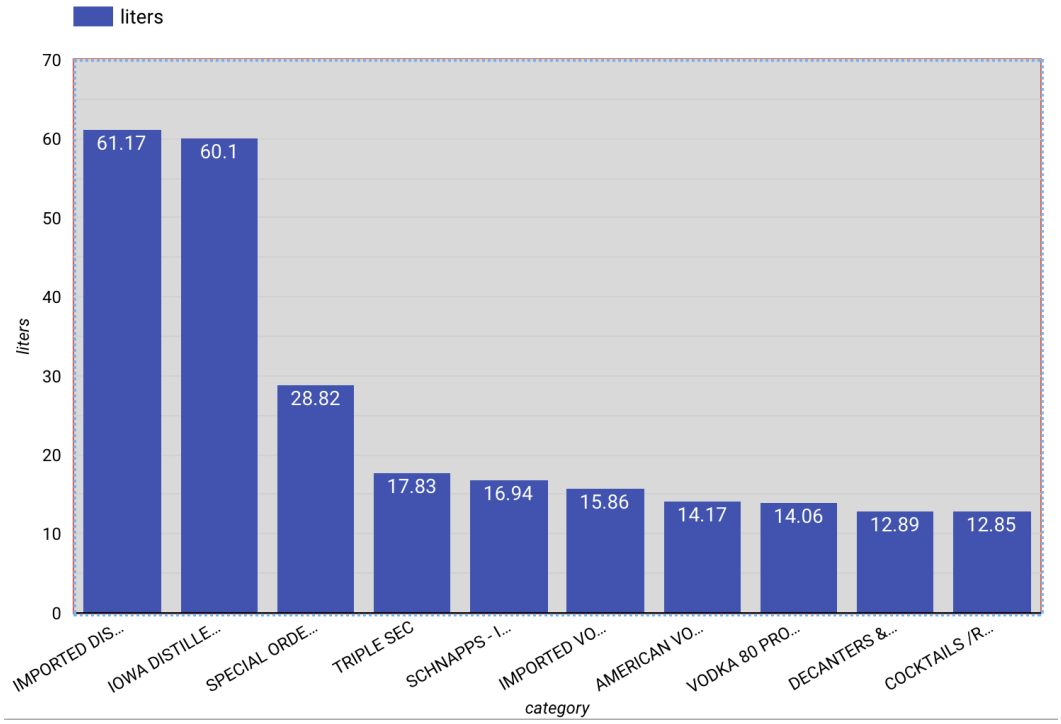
Query results

Query complete (0.5 sec elapsed, 493.4 MB processed)

Job information | Results | JSON | Execution details

Row	category	liters
1	Gold Rum	7.157165501396673
2	Triple Sec	19.89048005235899
3	American Vodka	2.5632395482464836
4	Temporary & Specialty Packages	8.687055433589451
5	Corn Whiskies	3.6307100859339663

5. If we want to take these results a step further, we can explore these results through DataStudio and see the results visually through a chart.

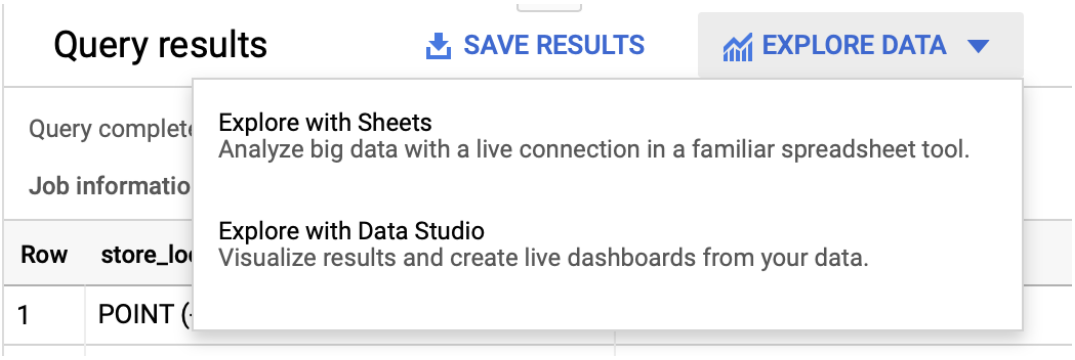


How to use Google DataStudio to visualize data:

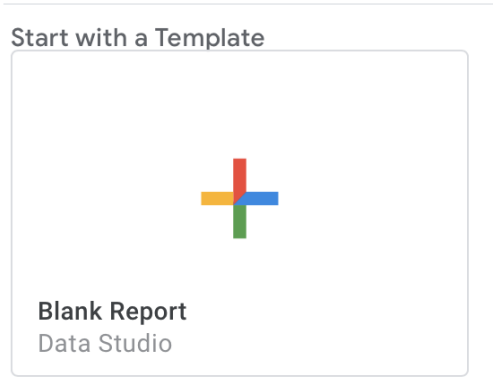
_____Google DataStudio is a feature that allows data analyzers to better interpret their data by allowing them to use visuals such as charts, graphs, scatter plots, etc.

There are two different ways to go about uploading data into DataStudio:

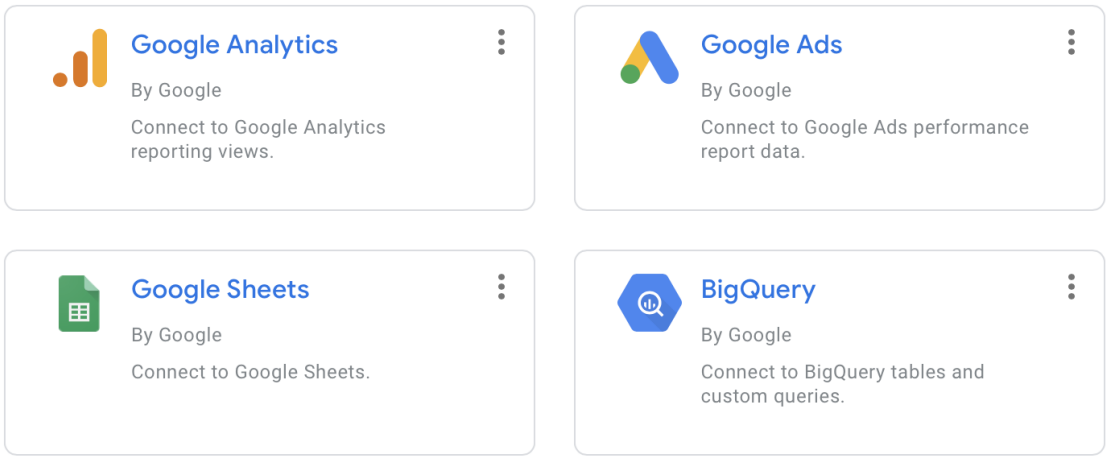
1. After writing your code through SQL, click on the **EXPLORE DATA** drop down menu and select **"Explore with Data Studio"**



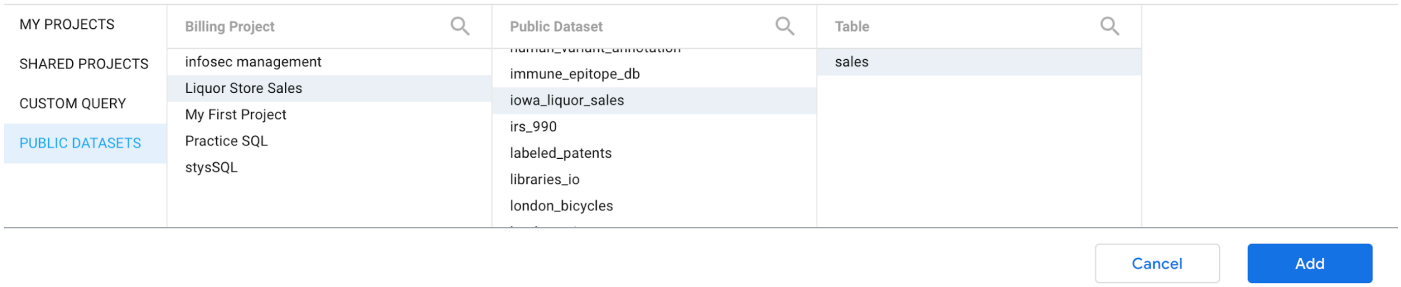
2. Go directly to **Data Studio** through Google and create a project from scratch.
    a. Step 1: Click on **"Blank Report."** There are many different layouts to choose from, but for this project we will be starting from scratch.



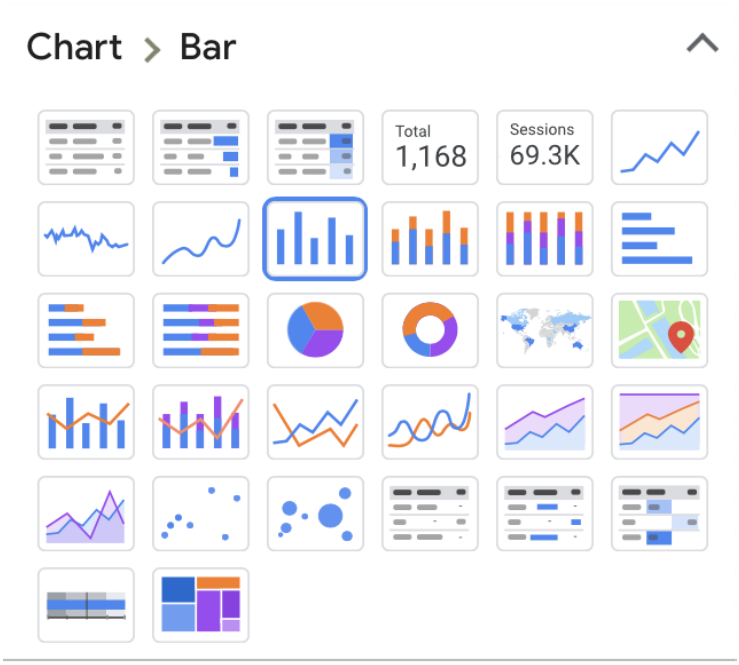
    b. Step 2: In "Add data to report," select **BigQuery**



c. Step 3: Select either “My Projects,” “Shared Project,” “Custom Query,” or “Public Datasets.” For this project, we will be using a Public Dataset given to us through BigQuery. Then select which project you want to work in, the dataset you want to use, and the table within the dataset you want to explore with. Finish by clicking **“Add.”**



d. Step 4: An “Untitled Report” will open with the default setting of a table with set Dimensions and Metrics. To change the name of the report, click on where it says, “Untitled Report” and change the name to be what you want it to be. On the right side of the screen you will see an array of different Charts to choose from. Select the Chart that best visually suits the data you are examining.

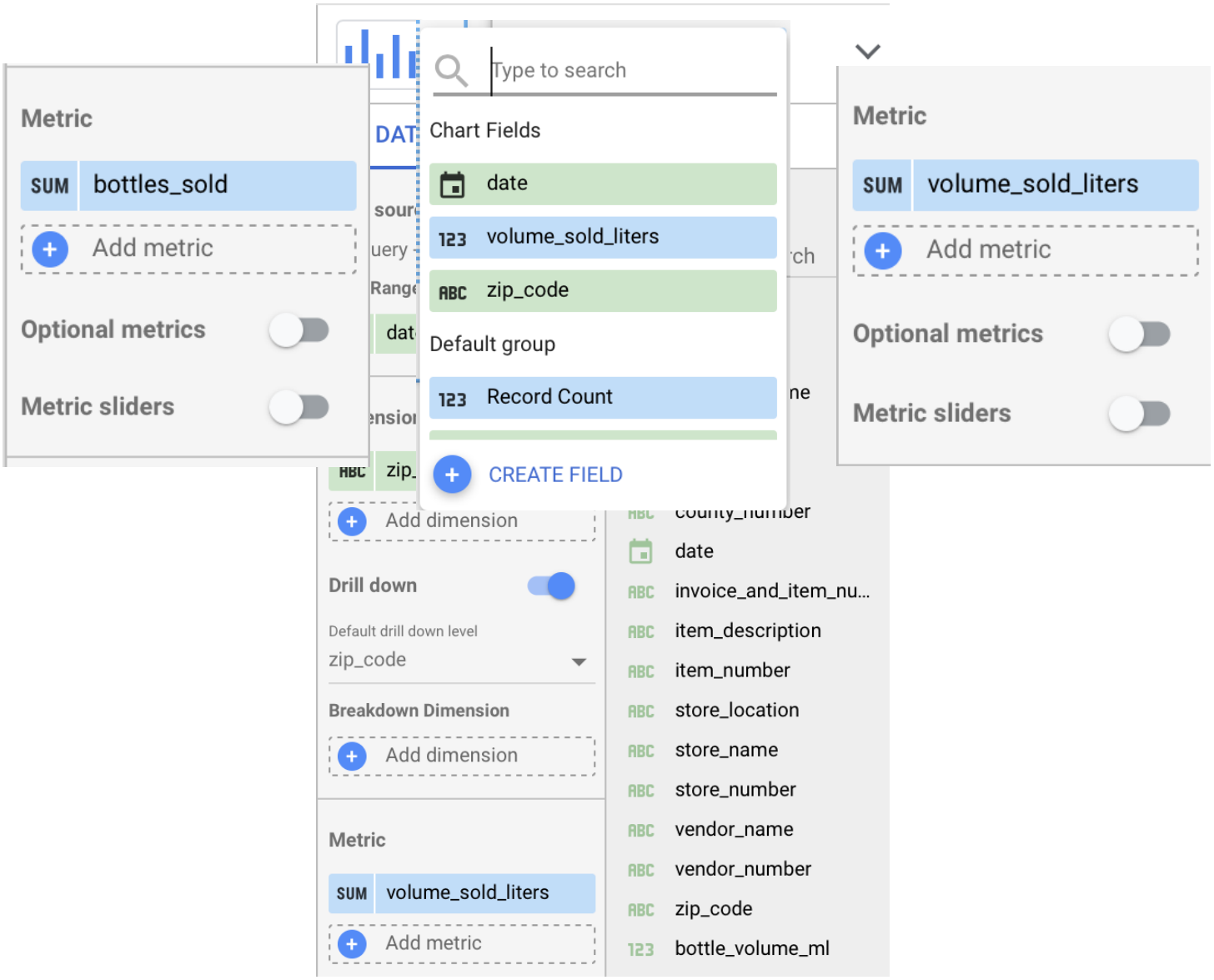


No matter which approach you take to get to DataStudio, the following steps will be the same. (Note: when you begin a project in DataStudio from scratch, the available fields to use when creating your charts is everything within the dataset. If you write code through SQL then explore with DataStudio, you will only be able to use the fields that you used in your code unless you blend the datasets together)

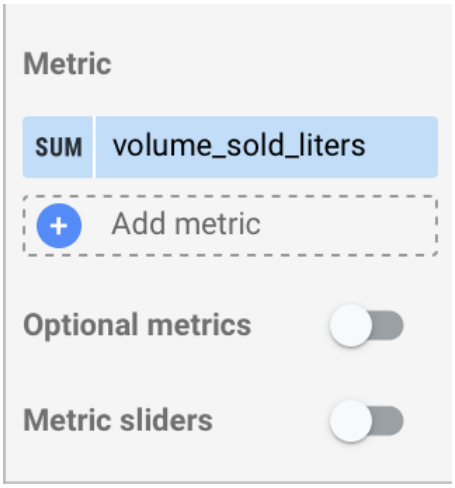
1. After selecting the Chart that best suits the data you are interpreting, it’s time to change the Dimensions, Metrics, and how you want the data sorted. (Note: if you begin with a bar graph and realize it isn’t the

best visual for the data you are looking at, you can always change the graph while keeping the same dimensions and metrics)

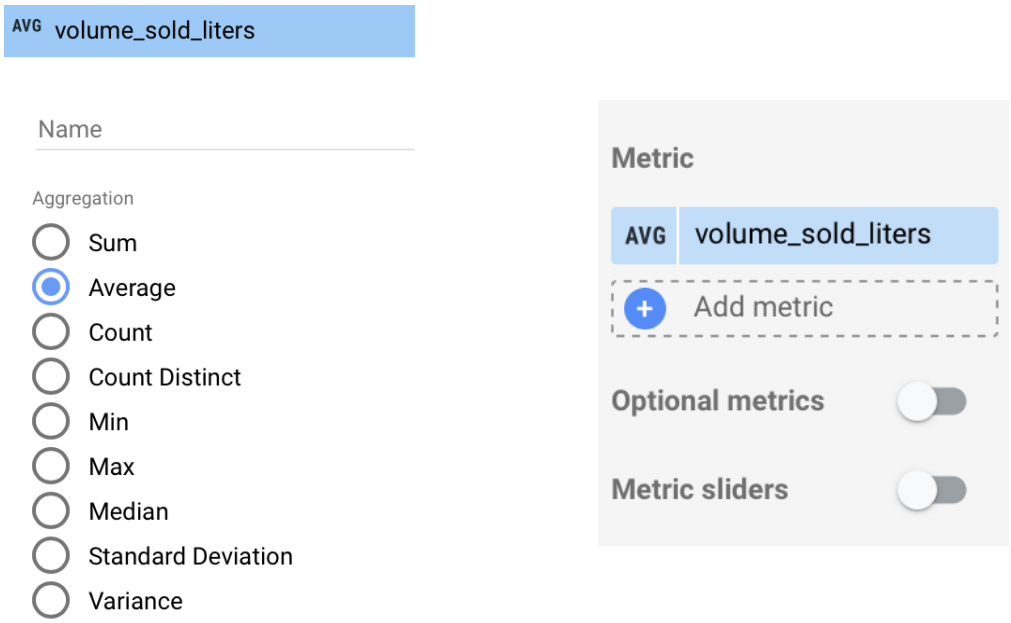
2. To change the dimensions or metrics of the chart, simply click on the field you wish to change and select the field you want to replace it with from the drop down menu that pops up or drag the field from the right column to either the dimensions or metrics column you wish to change.



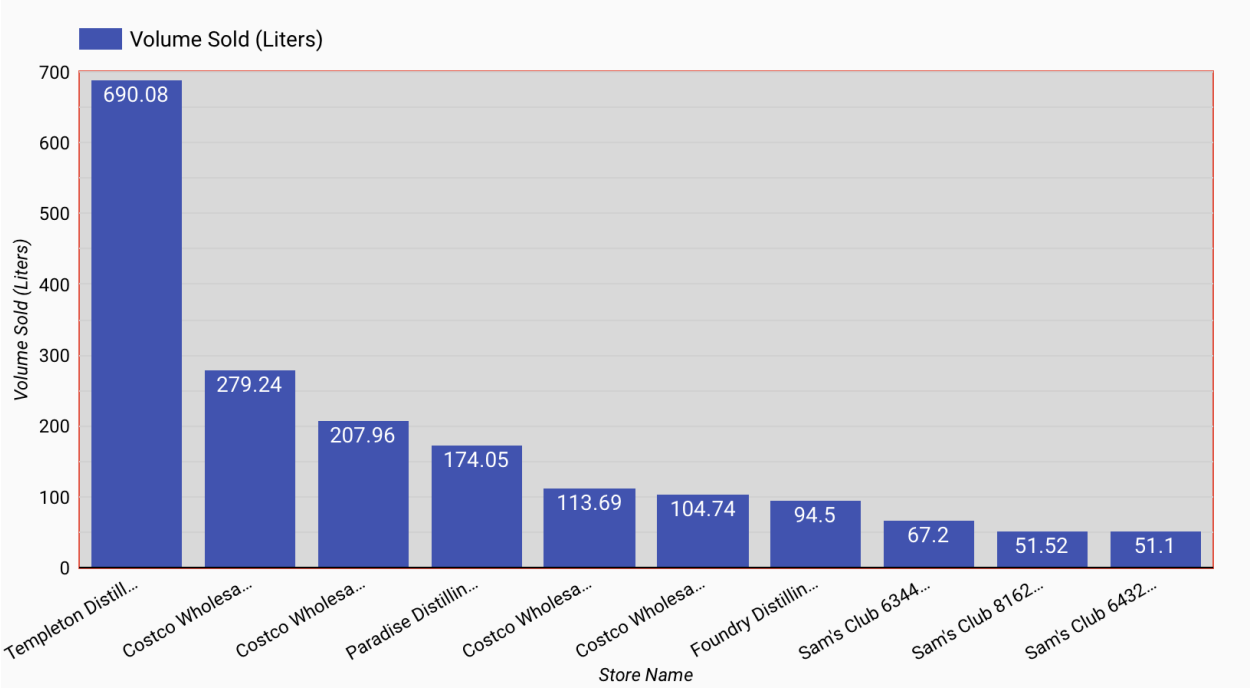
3. For the question we are examining, we are wanting the **average** of the volume of liters sold. To fix this in DataStudio, it takes two simple steps.
    a. Step 1: Click on the box next to the name of the field (in this case, where it says **SUM**)



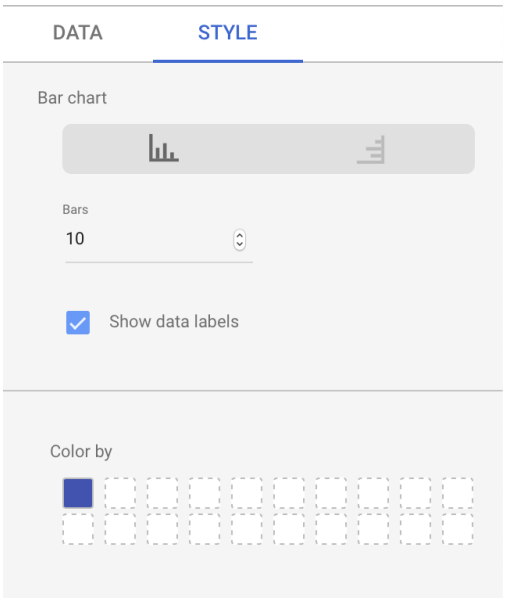
b. Step 2: Choose the aggregate you want to use instead, then click out of the window. Once you do this, the box next to the field name will change to the new aggregate you are using.



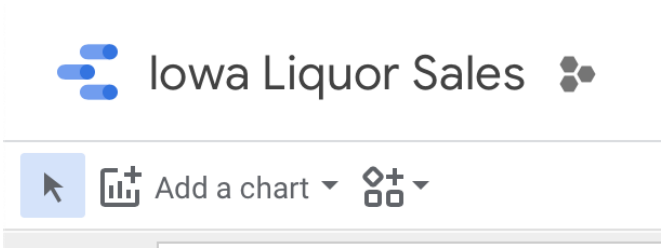
4. Now you have your graph created. As stated before, if this graph doesn't demonstrate the data you are examining the way you want, feel free to mess around with other graph options to demonstrate the data.



5. Now that you have your graph, click on **Style** to change how your graph looks. You can add the x and y axis titles to the graph, change the colors of the graph, font, size of font, etc.



Lastly, if you want to add more graphs to your DataStudio, click on the **Add a Chart** in the top left corner of your page. This allows you to have multiple charts on your DataStudio page.



ROUND():

The ROUND() function is a great tool to use to make your calculations much cleaner. The way the ROUND() function works is that it takes two parameters; the calculation and the number of decimal places you want to round the calculation too. Using this function alongside the AVG() function is very useful and makes your results look much better. Below is an example of using the ROUND() function, I'll be using is the "iowa_liquor_sales" dataset, the table "sales", and the columns "category_name" (which returns the name of the liquor) and "volume_sold_liters" (which returns the volume of liquor sold in liters).

1. Just like the other tutorials above, you must call the dataset and table using the 'FROM' statement. You must also use the 'SELECT' statement to specify which columns you want to use. Below is what your code should look like to get started.

```
60 SELECT
61   alc.category_name as Catagory,
62   alc.volume_sold_liters as Average_Volume_Sold
63 FROM
64   `bigquery-public-data.iowa_liquor_sales.sales` as alc
```

2. Next, you want to start doing your calculations. For this example, I want to get the average of the volume of liters sold, so I'll use the AVG() function (tutorial on how to use this function above). By doing this, you must also add a 'GROUP BY' statement on order for the Query to know what to aggregate the data too (a tutorial for 'GROUP BY' is further down in the document). The code should know look like this.

```
60 SELECT
61   alc.category_name as Catagory,
62   AVG(alc.volume_sold_liters) as Average_Volume_Sold
63 FROM
64   `bigquery-public-data.iowa_liquor_sales.sales` as alc
65 GROUP BY
66   Catagory
67
```

3. After doing this step, you can now start to clean up your results using the ROUND() function. Again, this function isn't necessary for the code to run, but it makes your results much more legible. You would insert the ROUND() function right before the calculation (in this case, the AVG() function) so that the function knows to round that column in the results. You must also include how many decimal places you want the function to round to, for this example, I chose 2 decimal places. After doing this your code should look like this.

```
60 SELECT
61   alc.category_name as Catagory,
62   ROUND(AVG(alc.volume_sold_liters), 2) as Average_Volume_Sold
63 FROM
64   `bigquery-public-data.iowa_liquor_sales.sales` as alc
65 GROUP BY
66   Catagory
```

4. Below is a comparison of the codes using the ROUND() function and not using the ROUND() function.

With the ROUND() function:

Job information | Results | JSON | Execution details

Row	Catagory	Average_Volume_Sold
1	Imported Distilled Spirit Specialty	61.17
2	Iowa Distillery Whiskies	60.1
3	Imported Vodka	57.64
4	Special Order Items	28.82

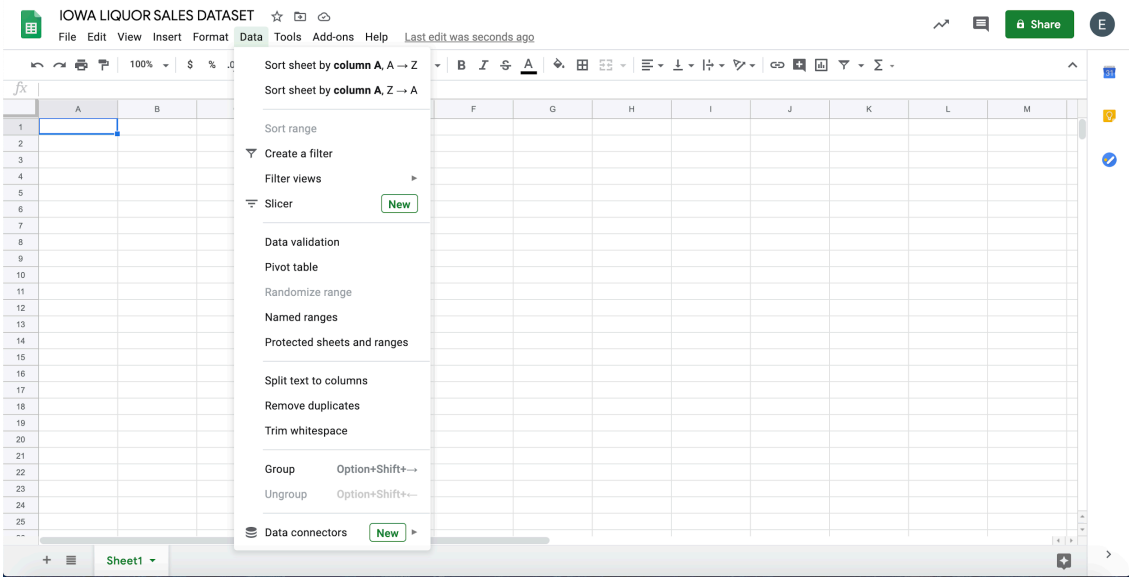
Without the ROUND() function:

Job information | Results | JSON | Execution details

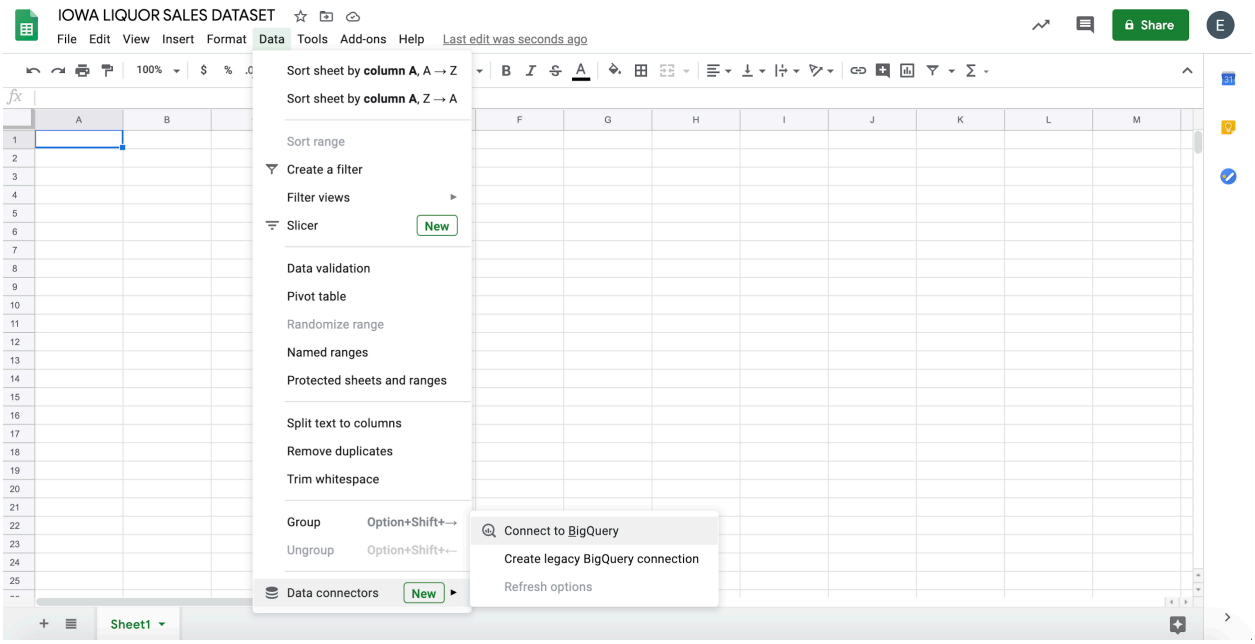
Row	Catagory	Average_Volume_Sold
1	Imported Distilled Spirit Specialty	61.16634595982755
2	Iowa Distillery Whiskies	60.099758064516124
3	Imported Vodka	57.63829252150893
4	Special Order Items	28.81688558980151

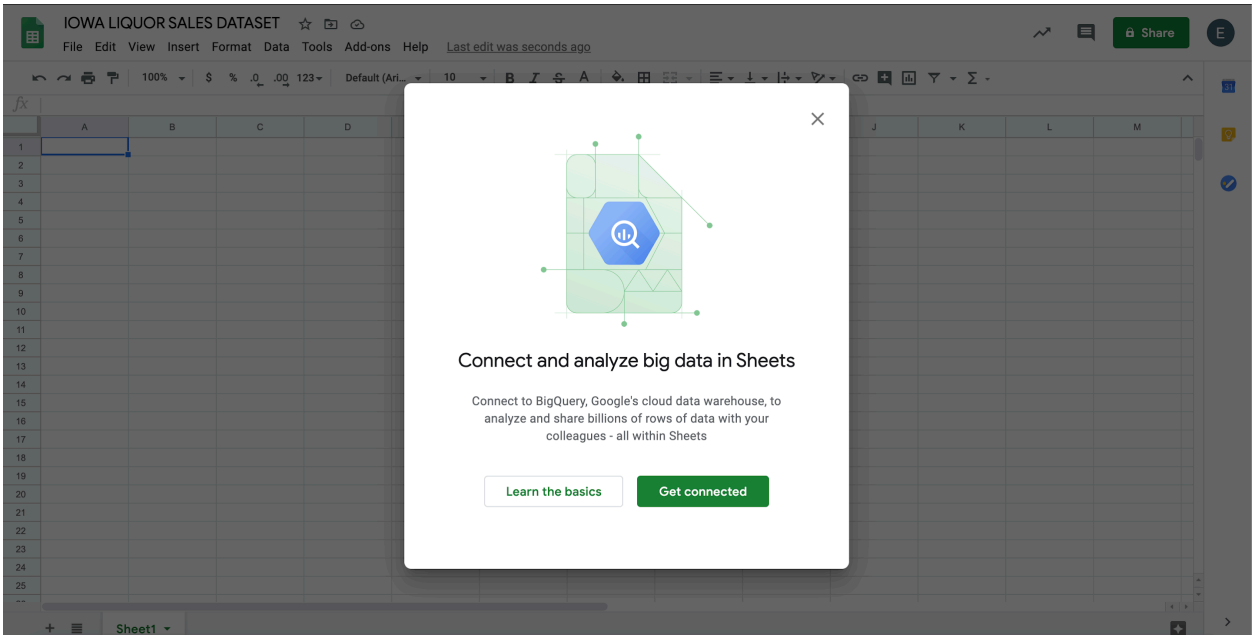
How to link BigQuery data to Google Sheets and perform descriptive analytics:

1. **Linking BigQuery data to Google Sheets and performing descriptive analytics tutorial:**
    a. Open up a new Sheets file and go to the data menu, at the bottom you will see 'Data' Connectors with the option to connect to BigQuery

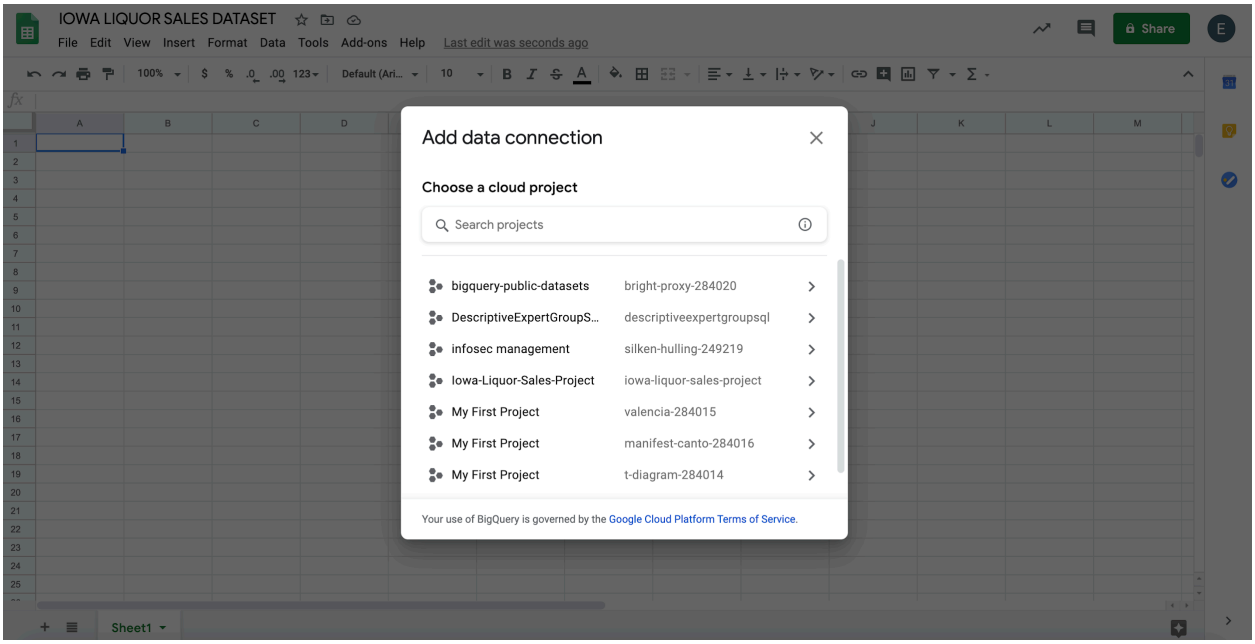


i. Hitting the data connections option will allow you to choose a cloud project. First hit connect to BigQuery and then you can search by project name and select your project.

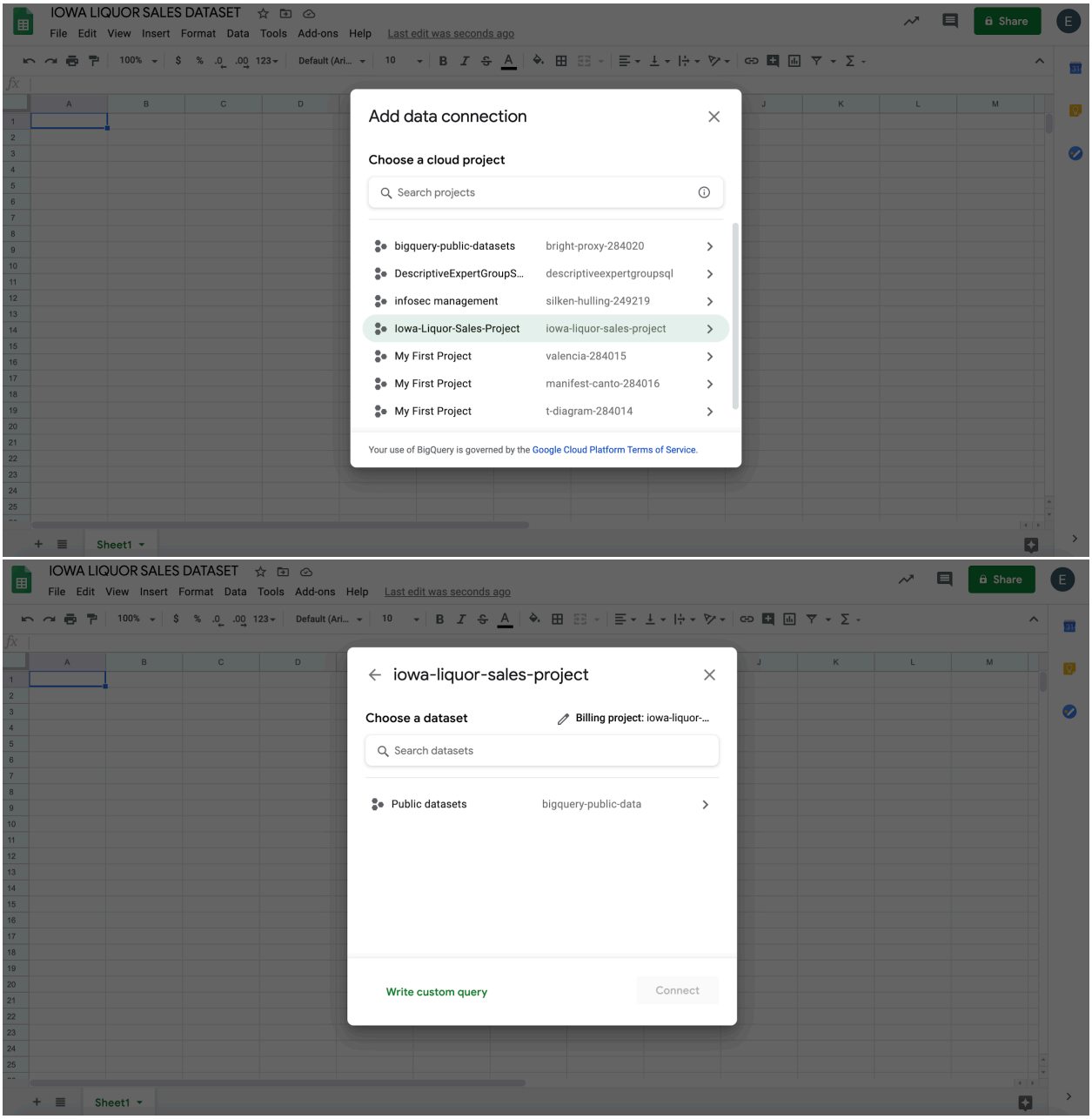




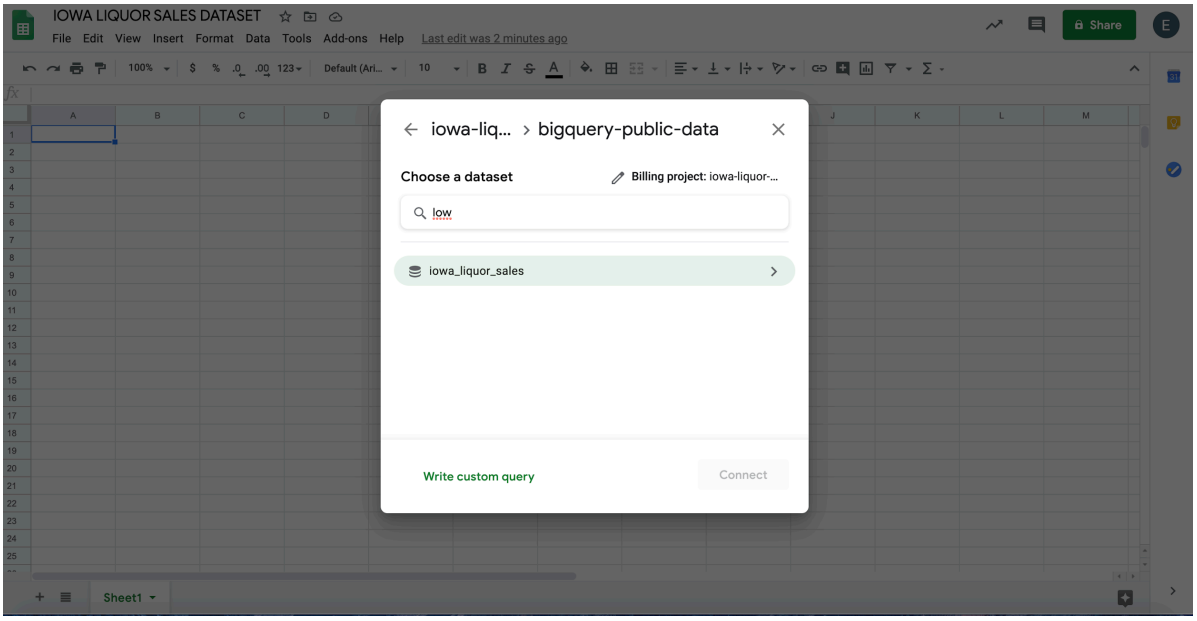
ii. After selecting get connected, a choose a cloud project option will show up where you can select a project you have created in BigQuery or select a public dataset. You can search for whatever you may need.

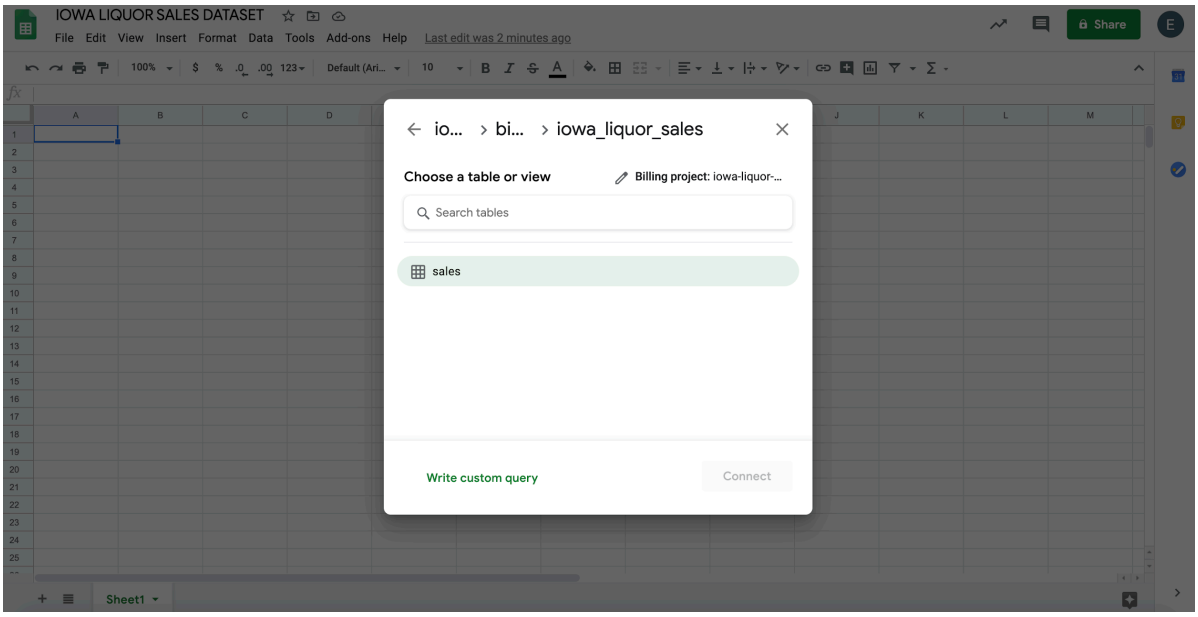


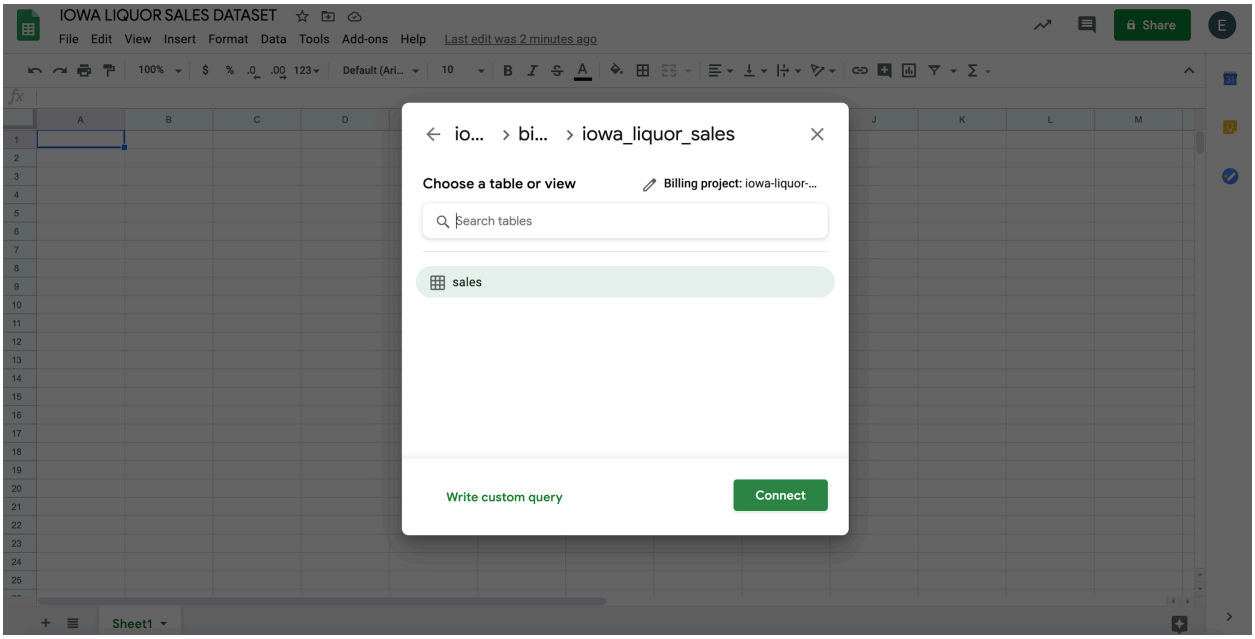
iii. For our project we will be utilizing the “Iowa Liquor Sales” dataset, which is a public dataset in BigQuery. As you can see in the image below I am selecting the Iowa-Liquor-Sales-Project that I have created in BigQuery - which holds the dataset I am trying to transfer to sheets.



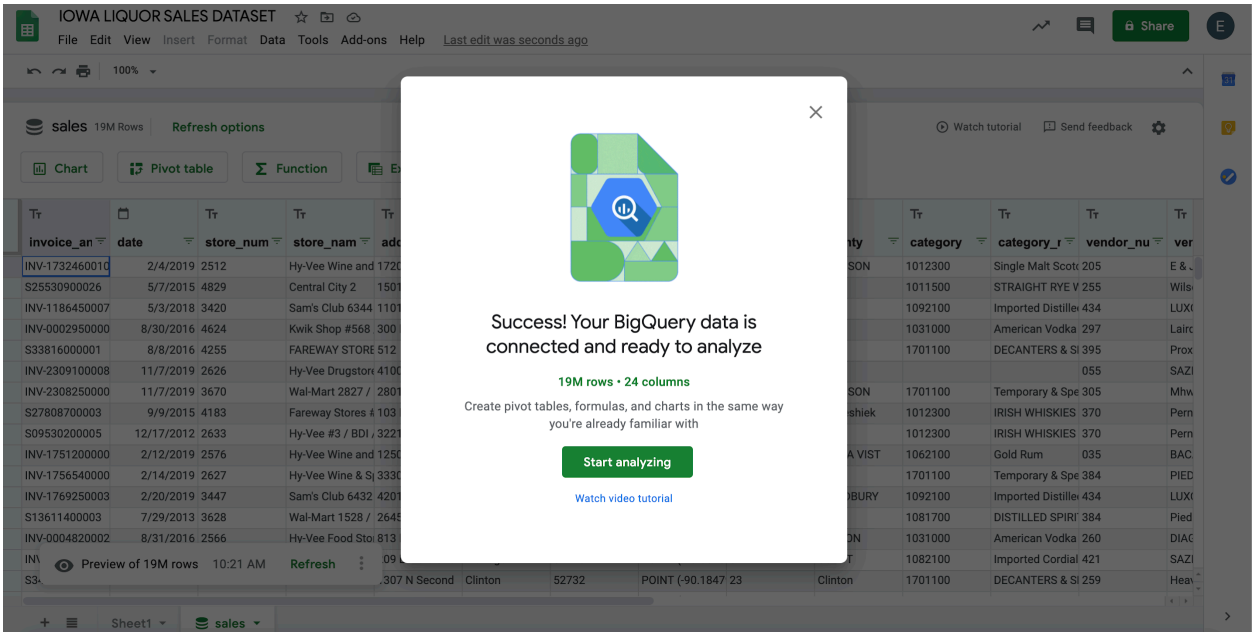
iv. After selecting the “Iowa Liquor Sales” dataset, that’s all it takes to make a connection to the full dataset.


IOWA LIQUOR SALES DATASET
File Edit View Insert Format Data Tools Add-ons Help
Last edit was 2 minutes ago
Share
iowa-liq... › bigquery-public-data
Choose a dataset
Billing project: iowa-liquor-...
low
iowa_liquor_sales
Write custom query
Connect
Sheet1

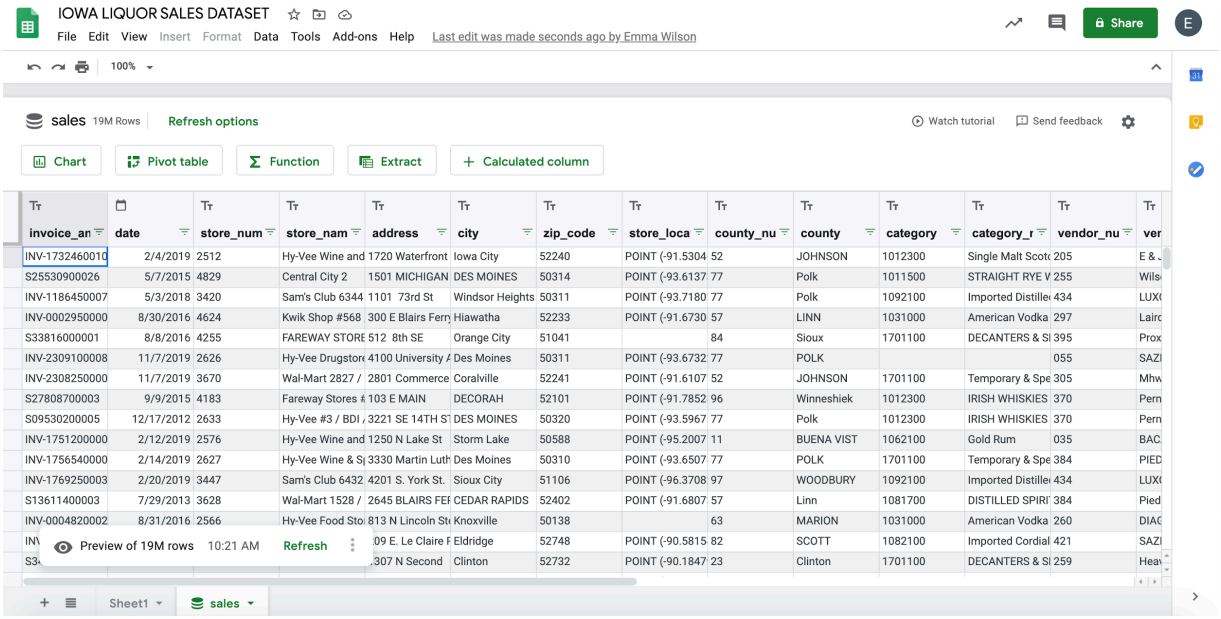

IOWA LIQUOR SALES DATASET
File Edit View Insert Format Data Tools Add-ons Help
Last edit was 2 minutes ago
Share
io... › bi... › iowa_liquor_sales
Choose a table or view
Billing project: iowa-liquor-...
Search tables
sales
Write custom query
Connect
Sheet1



v. Now we can see it successfully connected, Google Sheets only shows the first 500 rows of the dataset BUT charts, pivot tables, and functions will use the entire dataset.



vi. You can filter and sort the preview as well in order to explore the dataset and understand it more.



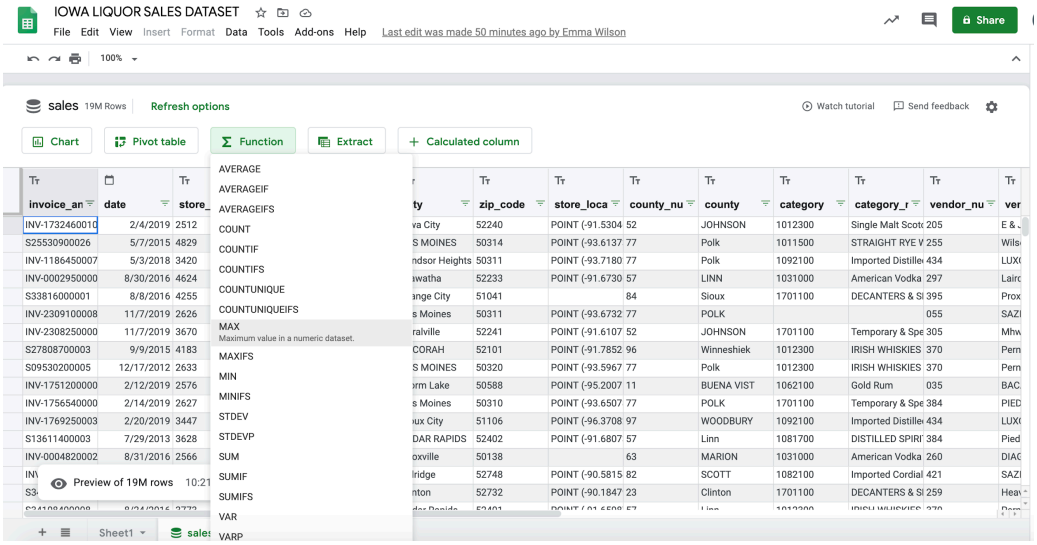
2. Formulas in Google Sheets using BigQuery Data

a. Now lets do some analysis using sheets functions, and the "Iowa Liquor Sales" dataset.

i. At the top of the preview tab, there is a function button - click the button to see options like AVG, COUNTIF, SUMIF, MIN, MAX.

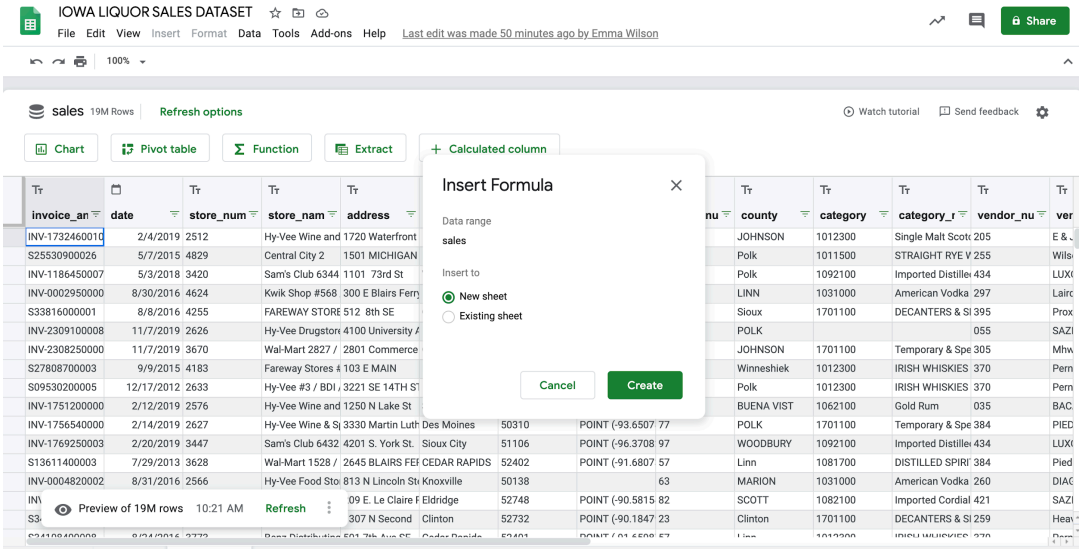
ii. Let's say I wanted to find the MAX of the Volume Sold in Liters column from the dataset.

iii. I would hit the function button then select MAX from the drop down list.

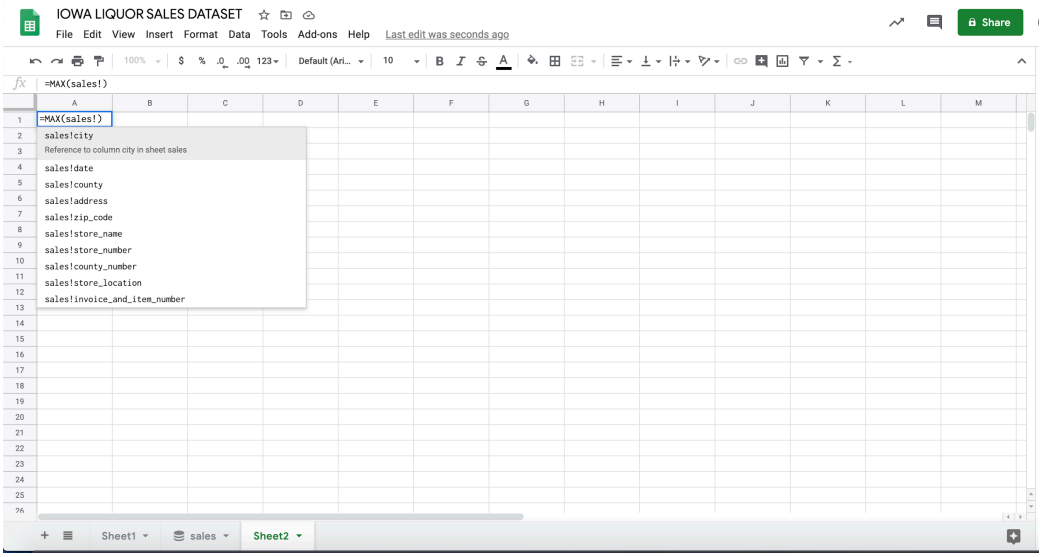


iv. After I have selected the MAX function a textbox will pop up asking if I want to insert the function into a new sheet or an existing sheet.

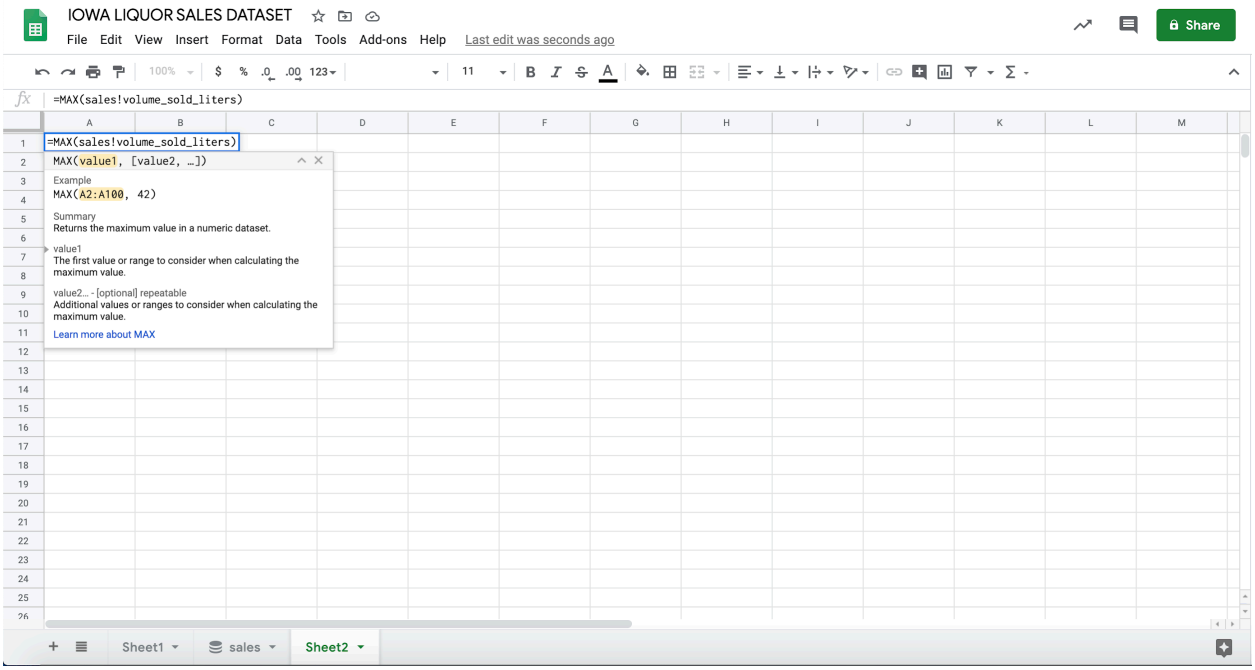
v. I will select the insert function into a new sheet option.



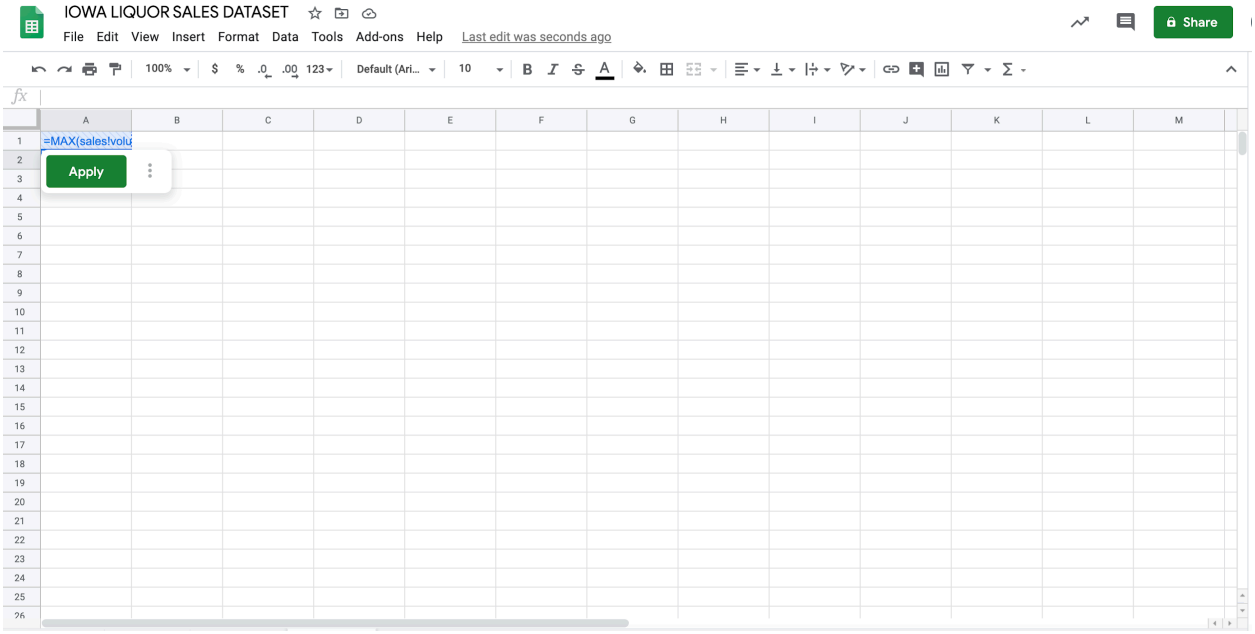
vi. Then, Google Sheets will start the formula for me =MAX(sales!_____)



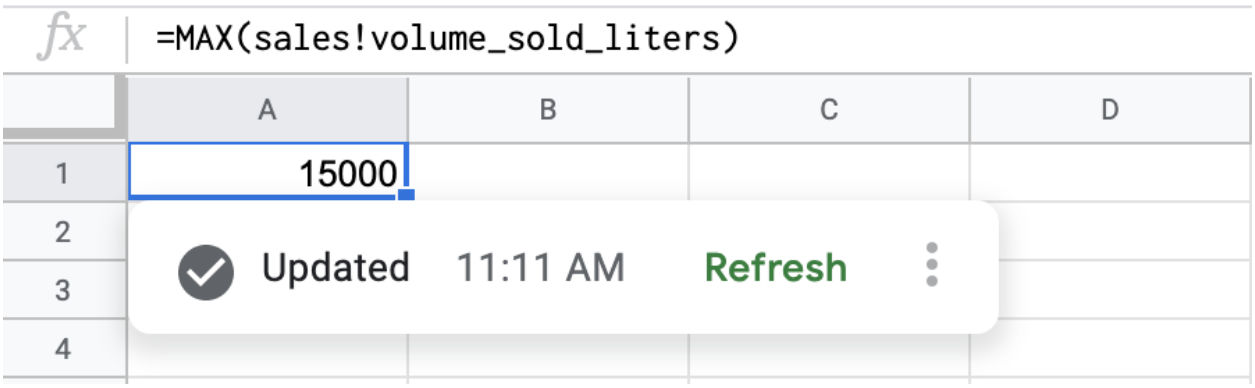
vii. I have to then start type volume_sold_liters in after the sales! Entry in the parentheses.



viii. After that I will have to hit apply in order to run the function I have entered in the new Google Sheet.



ix. Then, I will have to enter in the =MAX(sales!volume_sold_liters) and Google Sheets will run the formula for me and pop out = 15,000 which is the MAX amount of volume sold in liters. Hit the apply button to run the function in the new sheet and the output will be your answer for MAX volume of liters sold.

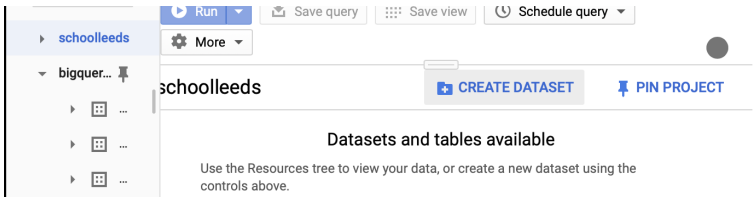


Predictive Tutorial

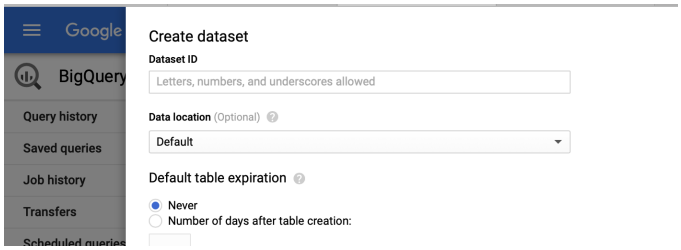
The purpose of this tutorial is to outline the steps to creating linear regression models within BigQuery using SQL queries as well as become familiar with the 'ML.EVALUATE' and 'ML.PREDICT' functions.

Step One: Creating a Data Set

1) First you want to make sure you're working within the correct project in the top left corner of the console you can select a project or you can select 'new project'
    a) Be careful to always check and make sure you're doing all the work for your project within the correct project.
2) Once you have selected the project you want to create a data set within the project to hold all your future queries and data.
    a) You can do this by click in the project name on the left hand side and from there selecting "Create Dataset"



b) You can then name the data set whatever you want, but because we are using the Iowa data from the public data set, I labeled my data set 'Iowa_data'



c) Then click **Create Dataset**

Step Two: Creating the Model

We can now create a linear regression model using the data from the Iowa sales, public data set to make a prediction on the volume in liters that will be sold to Polk county based on quarterly data.

Part 1) Use the ***CREATE MODEL*** function to simulate the creation of the model

- The correct sequence to a is: CREATE MODEL (`projectname.data set.new name`)
- You'll want to make sure that all the names are spelled correctly as well as pay attention to whether your letters are uppercase or lowercase this will result in an error message if not done correctly.

Part 2) on the next line type in the ***OPTIONS*** clause :

OPTIONS (model_type = 'linear_reg', input_label_ cols= ['avg']) AS

- This is indicating that you are running a linear regression model.
- A linear regression generates a continuous value from a linear combination of input features
- What you put on the inside of the [] should reference the feature that you are trying to predict, in this case we assigned the alias "avg" to the average volume sold in liters in the *SELECT* section

Part 3) on the next line your list your most important input features under ***SELECT*** clause

- For our example we want to select the features that we are using to predict volume in liters. The features we are including are
    - **Date: Date of Orde**r -- (extracting quarter)
    - **County:** City where the store who ordered the liquor is located (so we can group by county)
    - **Volume_sold_liters:** Total volume of liquor ordered in liters---(finding the avg to evaluate the total amount of volume sold of each category)
    - **Category Name** : category of the liquor ordered.

Part 4) insert ***FROM*** clause to indicate where the data you are using is located

- In this project:
    - FROM (`biguery-public-data.iowa_liquor.sales` AS liquor)
- We are using the "sales" data table within the Iowa_liquor sales public data set
- Alias this as sales

Part 5) insert a ***GROUP BY*** clause

- For this project we want to group our data quarter, County and category

Part 6) We inserted a ***HAVING*** clause next because we want to filter our results to only include the top county (POLK) within Iowa. We do this to try and condense our data to better answer our specific question.

Part 7) lastly an ***ORDER BY*** clause will order the results in this case in order to better see our results we want to order by quarter.

Extra notes:

- We used the EXTRACT() function to retrieve only the Quarter from our Date feature
- We used the AVG() function to only extract the sum of the volume in liters
- We use the UPPER () function to assign all upper case letters to category names and make our data easier to analyse.

SQL final project

```
CREATE OR REPLACE MODEL `schoolleeds.Iowa_Data.sales_newreg3`
OPTIONS(model_type='linear_reg', input_label_cols=['Avg']) AS

SELECT
EXTRACT(quarter from liquor.date) as Quarter,
UPPER(county) as county,
UPPER(category_name) as category,
(avg(volume_sold_liters)) as avg

FROM
`bigquery-public-data.iowa_liquor_sales.sales` as liquor

GROUP BY
Quarter,
county,
Category


HAVING
county is not null and county = 'POLK'
ORDER BY
Quarter
;
```

RESULTS: click on the evaluation tab to analyze the significance of your model

sales_newreg3

Details | Training | Evaluation | Schema

Mean absolute error	0.9209
Mean squared error	5.9548
Mean squared log error	0.0235
Median absolute error	0.3361
R squared	0.8928

- Mean absolute error:The average distance from the predicted value to the actual value. Lower is better.
- Mean squared error: Used for evaluating statistical significance. Lower is better
- Mean squared log error: Lower is better.
- Median absolute error: A measure more robust to outliers. Lower is better.
- R-squared: higher is better

Step Three: Evaluate your Model

After you have created your model, you need to evaluate the performance of the classifier using the ***ML. EVALUATE*** function. This function evaluates the predicted values (in our case volume of liquor sold in liters) against the actual data.

Part 1) Query details

- In the bigquery Query editor, comment out the ***CREATE MODEL*** model built in Step two.
- Use the ***SELECT*** statement to retrieve the columns from your model
    - Using the **(*)** after the select statement selects all columns from the model
- The FROM clause uses the ***ML.EVALUATE*** function against the model we want to evaluate. In this case, the model we are evaluating is labeled **'sales_newreg3'**.
    - Note: the ***SELECT*** statement and ***FROM*** clause are the same ones you use in the ***CREATE MODEL***. If you want to specify what exactly you are evaluating, you can use a ***WHERE*** clause to filter your data. Otherwise, you can provide no additional information and the evaluation will use the same metrics calculated during training.

```
25
26 #EVAL
27
28 SELECT
29 *
30 FROM
31 ML.EVALUATE (MODEL`schoolleeds.Iowa_Data.sales_newreg3`)
32 ;
33
```

Part 2) Run the ***ML.EVALUATE*** query

- Enter your ***SELECT*** and ***FROM*** statement in the Query editor
- Click run
- When the Query is complete, the Results tab will pop up below the query editor window.

Query complete (0.1 sec elapsed, 0 B processed)

Job information | **Results** | JSON | Execution details

Row	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
1	0.9208828762042934	5.954847584255329	0.023475558137849604	0.3360788510015924	0.8928310263861011	0.8928870744403457

- Because we performed a linear regression, the results include all the columns shown above.
- The most important metric in the evaluation results is the R-squared metric. R-squared is a statistical measure that determines if the predictions from the linear regression are appropriate for the actual data. When the R-squared metric is 0, it indicates that the model explains none of the variability of the response data around the mean, whereas a 1 indicates the model explains all of the variability.
    - For our model, the R-squared metric is a .8928 meaning approximately 90% of the variability, or change, in volume of liquor sold in Polk county can be predicted by the category of liquor being sold based on quarter.

Step Four: Use your Model to Predict Outcomes

After we have evaluated our model, the next step is to use the model we created to predict an outcome. For us, we will use our model to predict the demand of liquor (in liters) for each liquor category, of all liquor sales in POLK county. We chose POLK county because it is the county with the highest demand for liquor by volume.

Part 1) Query Details

- First, the top-most ***SELECT*** statement pulls the column you want to predict. In this instance, we use the **(*)** because this column is generated automatically depending on what regression you are running. In this instance, we used a linear regression model, so the *predicted_Avg* column will be the estimated demand of each liquor category sold in liters.
- The top-most ***FROM*** statement should include the Model you created in step two. For this case, we used the model *sales_newreg3*
- The query's nested ***SELECT*** statement and ***FROM*** clause are the same as those used in the ***CREATE MODEL*** query from step two. Just copy and paste them below!
- The ***HAVING*** clause, in this instance, is filtering out the orders where the county name is not NULL and is equal to "POLK" (polk has the highest demand for liquor in iowa, therefore we are trying to predict the quantity they are demanding).

```
-- RUN Predictions

SELECT
*
FROM
ML.PREDICT( MODEL Iowa_Data.sales_newreg3,
(
 SELECT
EXTRACT(quarter from liquor.date) as Quarter,
UPPER(county) as county,
UPPER(category_name) as category,
(avg(volume_sold_liters)) as avg

FROM
`bigquery-public-data.iowa_liquor_sales.sales` as liquor

GROUP BY
Quarter,
county,
Category

HAVING
county is not null and county = 'POLK'

ORDER BY
Quarter ))
```

-

Part 2) Run the ***ML.PREDICT*** query

- Once you have all of your ***SELECT*** statements and ***FROM*** clauses, make sure you have commented out all of your other Queries.
- Click "run"

- When the query is complete, click the results tab below the query text editor. The results should look like this:

Query complete (0.6 sec elapsed, 789.2 MB processed)

Job information | Results | JSON | Execution details

Row	predicted_Avg	Quarter	county	category	avg
1	10.053834194923677	3	POLK	TEMPORARY & SPECIALTY PACKAGES	9.902881152460983
2	7.284362990686727	3	POLK	STRAIGHT RYE WHISKIES	6.0244516016713066
3	7.388645268574124	4	POLK	STRAIGHT RYE WHISKIES	9.353760460968582
4	16.062824410226412	3	POLK	DECANTERS & SPECIALTY PACKAGES	12.954163306451612
5	9.845269639148883	1	POLK	TEMPORARY & SPECIALTY PACKAGES	8.658508853681269
6	5.767389162132833	3	POLK	AGED DARK RUM	5.742542372881355
7	7.440717157558938	3	POLK	COFFEE LIQUEURS	6.68163225649745

Step Five: Find the Model Weights

Part 1) Using ***ML.WEIGHTS***

- First, use the ***SELECT*** * statement to select all columns from your model. Next, use the ***FROM*** clause. Instead of just typing in your model name, first type ***"ML.WEIGHTS"*** and then ***(MODEL 'project-name.data-set.model-name')***. This will give you the weights for each variable you are weighing.
- Below is an example of the output. In this case, weighing is relational by liquor type, meaning that negative weights lead to less alcohol sales for each brand per quarter and positive weights correlate with an increase in sales.

```
# Weights of coefficients --


SELECT
*
FROM
ML.WEIGHTS( MODEl `Iowa_Data.sales_newreg3`)
;

#weight standardize
SELECT
*
FROM
ML.WEIGHTS(MODEL `Iowa_Data.sales_newreg3`,
  STRUCT (true as standardize))
```

Liquor Category	Category Weights
IOWA DISTILLERY WHISKIES	43.61394123
IMPORTED DISTILLED SPIRIT SPECIALTY	40.90814931
SPECIAL ORDER ITEMS	24.92149765