## **Summary: Hands on SQL**

## **Activity overview**

****



### **What you will need**

To get started, download the automobile\_data CSV file. This is data from an external source that contains historical sales data on car prices and their features.

Click the link to the automobile\_data file to download it. Or you may download the CSV file directly from the attachments below.

Link to data: [automobile\_data](https://drive.google.com/u/0/uc?id=1cJtuw-6mxZk7BNkcsLYEvfjW0l_PdKxA&export=download)

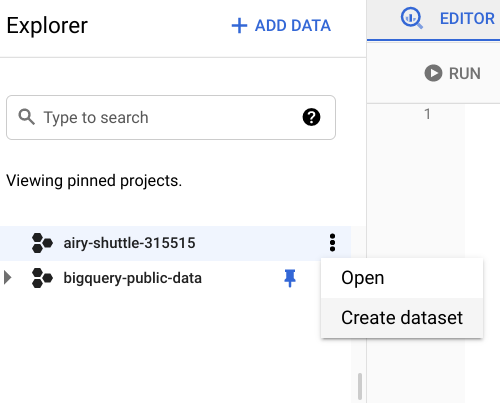
## **Upload your data**

****

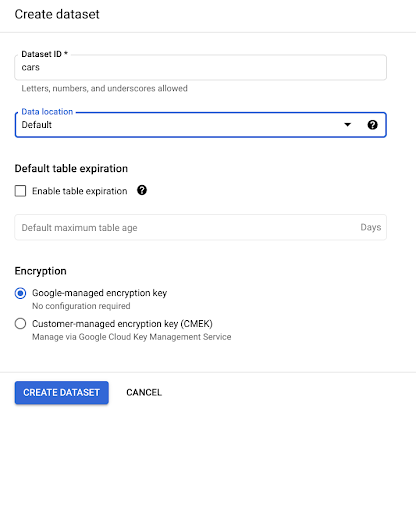
Similarly to a previous BigQuery activity, you will need to create a dataset and a custom table to house your data. Then, you’ll be able to use SQL queries to explore and clean it. Once you’ve downloaded the automobile\_data file, you can create your dataset.

### **Step 1: Create a dataset**

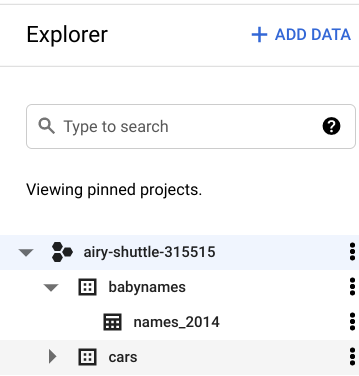
Go to the **Explorer pane** in your workspace and **click the three dots next to your pinned project** to open the menu. From here, **select** **Create dataset.**

****

From the Create dataset menu, fill out some information about the dataset. **Input the Dataset ID as *cars*;** you can leave the Data location as Default. Then **click CREATE DATASET**.



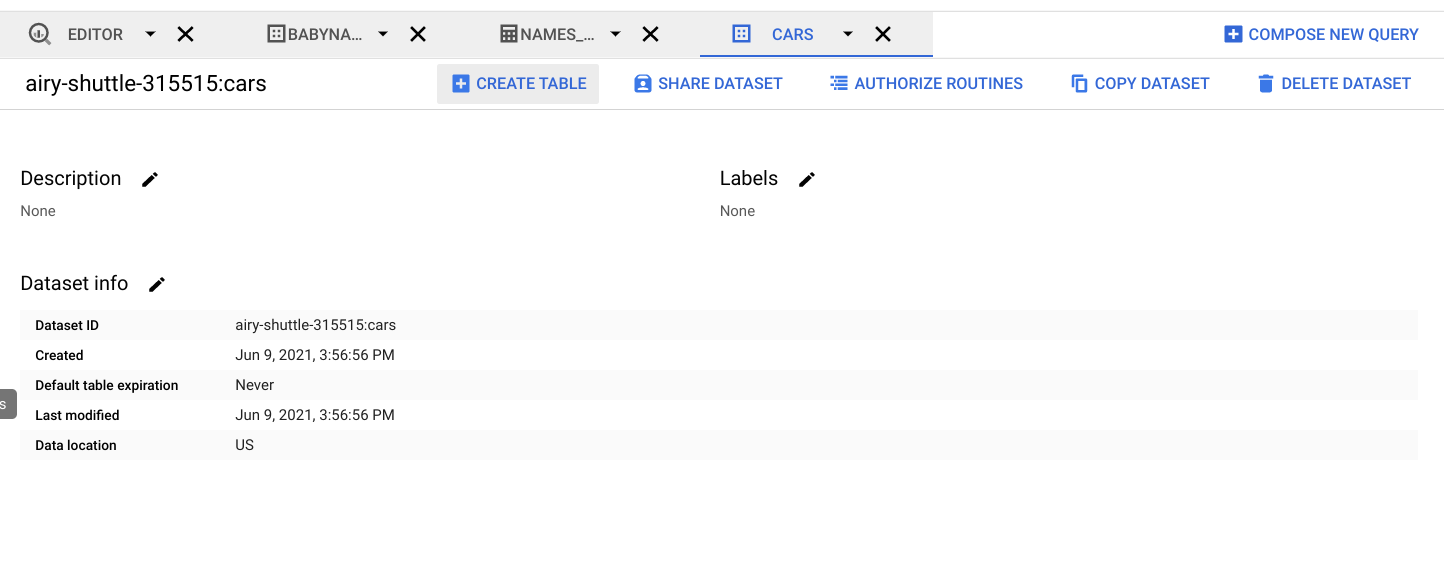
The cars dataset should appear under your project in the Explorer pane as shown below. **Click on the three dots next to the cars dataset** to open it.



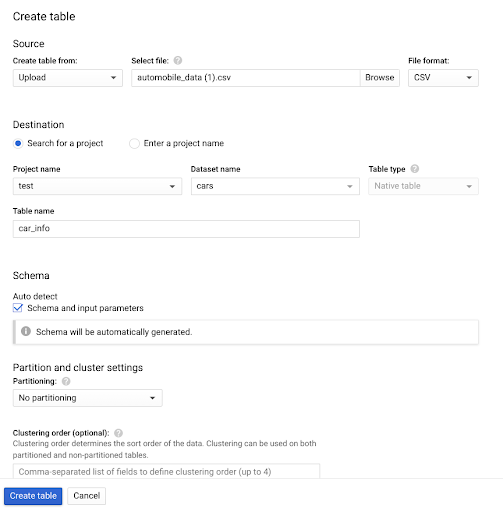
### **Step 2: Create table**

After you open your newly created dataset, you will be able to add a custom table for your data.

From the cars dataset, **click CREATE TABLE**.



**Under Source, upload the automobile\_data CSV**. Under Destination, make sure you are uploading into your cars dataset and **name your table *car\_info*.** You can **set the schema to Auto-detect**. **Then, click Create table.**

****

After creating your table, it will appear in your Explorer pane. You can **click on the table to explore the schema and preview your data.** Once you have gotten familiar with your data, you can start querying it.

## **Cleaning your data**

****

Your new dataset contains historical sales data, including details such as car features and prices. You can use this data to find the top 10 most popular cars and trims. But before you can perform your analysis, you’ll need to make sure your data is clean. If you analyze dirty data, you could end up presenting the wrong list of cars to the investors. That may cause them to lose money on their car inventory investment.

### **Step 1: Inspect the fuel\_type column**

The first thing you want to do is inspect the data in your table so you can find out if there is any specific cleaning that needs to be done. According to the [data’s description](https://archive.ics.uci.edu/ml/datasets/Automobile), the **fuel\_type column** should only have **two unique string values: diesel and gas**. To check and make sure that’s true, **run the following query:**

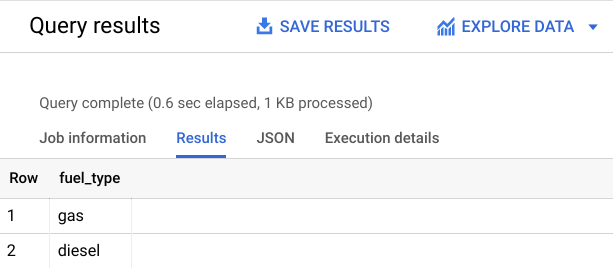
**SELECT**

**DISTINCT fuel\_type**

**FROM**

**cars.car\_info;**

This returns the following results:



This confirms that the fuel\_type column doesn’t have any unexpected values.

### **Step 2: Inspect the length column**

Next, you will inspect a column with numerical data. The length column should contain numeric measurements of the cars. So you will check that the minimum and maximum lengths in the dataset align with the [data description](https://archive.ics.uci.edu/ml/datasets/Automobile), which states that the lengths in this column should range from 141.1 to 208.1. **Run this query to confirm**

**SELECT**

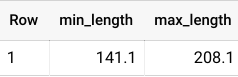
**MIN(length) AS min\_length,**

**MAX(length) AS max\_length**

**FROM**

**cars.car\_info;**

Your results should confirm that 141.1 and 208.1 are the minimum and maximum values respectively in this column.



### **Step 3: Fill in missing data**

Missing values can create errors or skew your results during analysis. You’re going to want to check your data for null or missing values. These values might appear as a blank cell or the word *null* in BigQuery.

You can **check to see if the num\_of\_doors column contains null values using this query:**

**SELECT**

**\***

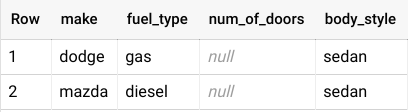
**FROM**

**cars.car\_info**

**WHERE**

**num\_of\_doors IS NULL;**

This will select any rows with missing data for the num\_of\_doors column and return them in your results table. You should get two results, one Mazda and one Dodge:



In order to fill in these missing values, you check with the sales manager, who states that all Dodge gas sedans and all Mazda diesel sedans sold had four doors. If you are using the BigQuery free trial, you can **use this query to update your table so that all Dodge gas sedans have four doors:**

**UPDATE**

**cars.car\_info**

**SET**

**num\_of\_doors = "four"**

**WHERE**

**make = "dodge"**

**AND fuel\_type = "gas"**

**AND body\_style = "sedan";**

You should get a message telling you that three rows were modified in this table. To make sure, you can **run the previous query again:**

**SELECT**

**\***

**FROM**

**cars.car\_info**

**WHERE**

**num\_of\_doors IS NULL;**

Now, you only have one row with a NULL value for num\_of\_doors. **Repeat this process to replace the null value for the Mazda.**

If you are using the BigQuery Sandbox, you can skip these UPDATE queries; they will not affect your ability to complete this activity.

### **Step 4: Identify potential errors**

Once you have finished ensuring that there aren’t any missing values in your data, you’ll want to check for other potential errors. You can use SELECT DISTINCT to check what values exist in a column. You can **run this query to check the num\_of\_cylinders column:**

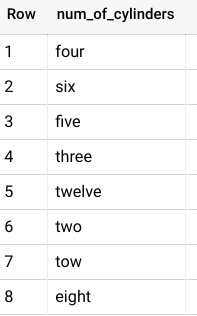
**SELECT**

**DISTINCT num\_of\_cylinders**

**FROM**

**cars.car\_info;**

After running this, you notice that there are one too many rows. **There are two entries for two cylinders: rows 6 and 7. But the *two* in row 7 is misspelled.**

****

To **correct the misspelling for all rows, you can run this query if you have the BigQuery free trial:**

**UPDATE**

**cars.car\_info**

**SET**

**num\_of\_cylinders = "two"**

**WHERE**

**num\_of\_cylinders = "tow";**

You will get a message alerting you that one row was modified after running this statement. To **check that it worked, you can run the previous query again:**

**SELECT**

**DISTINCT num\_of\_cylinders**

**FROM**

**cars.car\_info;**

Next, you can check the compression\_ratio column. According to the [data description](https://archive.ics.uci.edu/ml/datasets/Automobile), **the compression\_ratio column values should range from 7 to 23.** Just like when you checked the length values , you can **use MIN and MAX to check if that’s correct:**

**SELECT**

**MIN(compression\_ratio) AS min\_compression\_ratio,**

**MAX(compression\_ratio) AS max\_compression\_ratio**

**FROM**

**cars.car\_info;**

Notice that **this returns a maximum of 70**. But you know this is an error because the maximum value in this column should be 23, not 70. So the 70 is most likely a 7.0. Run the above query again without the row with 70 to make sure that the rest of the values fall within the expected range of 7 to 23.

**SELECT**

**MIN(compression\_ratio) AS min\_compression\_ratio,**

**MAX(compression\_ratio) AS max\_compression\_ratio**

**FROM**

**cars.car\_info**

**WHERE**

**compression\_ratio <> 70;**

Now the highest value is 23, which aligns with the data description. So you’ll want to correct the 70 value. You check with the sales manager again, who says that this row was made in error and should be removed. Before you delete anything, you should check to see how many rows contain this erroneous value as a precaution so that you don’t end up deleting 50% of your data. If there are too many (for instance, 20% of your rows have the incorrect 70 value), then you would want to check back in with the sales manager to inquire if these should be deleted or if the 70 should be updated to another value. Use the query below to count how many rows you would be deleting:

**SELECT**

**COUNT(\*) AS num\_of\_rows\_to\_delete**

**FROM**

**cars.car\_info**

**WHERE**

**compression\_ratio = 70;**

Turns out there is only one row with the erroneous 70 value. So you can **delete that row using this query:**

**DELETE cars.car\_info**

**WHERE compression\_ratio = 70;**

If you are using the BigQuery sandbox, you can replace DELETE with SELECT to see which row would be deleted.

### **Step 5: Ensure consistency**

Finally, you want to check your data for any inconsistencies that might cause errors. These inconsistencies can be tricky to spot — sometimes even something as simple as an extra space can cause a problem.

**Check the drive\_wheels column** for inconsistencies by **running a query with a SELECT DISTINCT statement:**

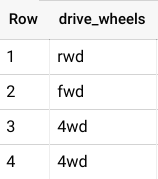
**SELECT**

**DISTINCT drive\_wheels**

**FROM**

**cars.car\_info;**

It appears that 4wd appears twice in results. However, because you used a SELECT DISTINCT statement to return unique values, this probably means there’s an extra space in one of the 4wd entries that makes it different from the other 4wd.



To check if this is the case, you can **use a LENGTH statement** to determine the length of how long each of these string variables:

**SELECT**

**DISTINCT drive\_wheels,**

**LENGTH(drive\_wheels) AS string\_length**

**FROM**

**cars.car\_info;**

According to these results, some instances of the 4wd string have four characters instead of the expected three (4wd has 3 characters). In that case, you can **use the TRIM function to remove all extra spaces in the drive\_wheels column if you are using the BigQuery free trial**:

**UPDATE**

**cars.car\_info**

**SET**

**drive\_wheels = TRIM(drive\_wheels)**

**WHERE TRUE;**

Then, you **run the SELECT DISTINCT statement again** to ensure that there are only three distinct values in the drive\_wheels column:

**SELECT**

**DISTINCT drive\_wheels**

**FROM**

**cars.car\_info;**

And now there should only be three unique values in this column! Which means your data is clean, consistent, and ready for analysis!