Clean Data using SQL

231030063 Tenzing Pema Thungon

Introduction

In this project, we focus on cleaning and preparing a dataset related to automobile sales using SQL. The objective is to identify the most popular car models by ensuring the data is accurate, consistent, and ready for analysis. Data cleaning is a crucial step because real-world datasets often contain errors, missing values, or inconsistencies, which can compromise the quality of analysis and decision-making. By meticulously cleaning the data, we create a reliable foundation for accurate insights, such as determining which cars a dealership should prioritize in their inventory.

Objectives

- **Ensure Data Accuracy:** Identify and correct errors, outliers, and inconsistencies in the dataset to maintain high data quality.
- **Handle Missing Data**: Detect and appropriately address missing values to ensure completeness of the dataset.
- **Optimize Data for Analysis:** Prepare the dataset for efficient querying and analysis by standardizing and cleaning data fields.
- **Enhance Query Performance:** Implement best practices in SQL to optimize the performance of data queries, particularly in large datasets.
- **Support Data-Driven Decision Making:** Provide a clean and reliable dataset that can be used to generate actionable insights, such as identifying the most popular car models for a dealership's inventory.

Methodology

Using Google Cloud BigQuery API
Link of the automible_data spreassheets:
https://docs.google.com/spreadsheets/d/13lypx5qtQEfpHpA7SjcI8rrGkHFfk389XBc7DNoluSA/edit?usp=sharing

Step 1: Understand how data is measured

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.

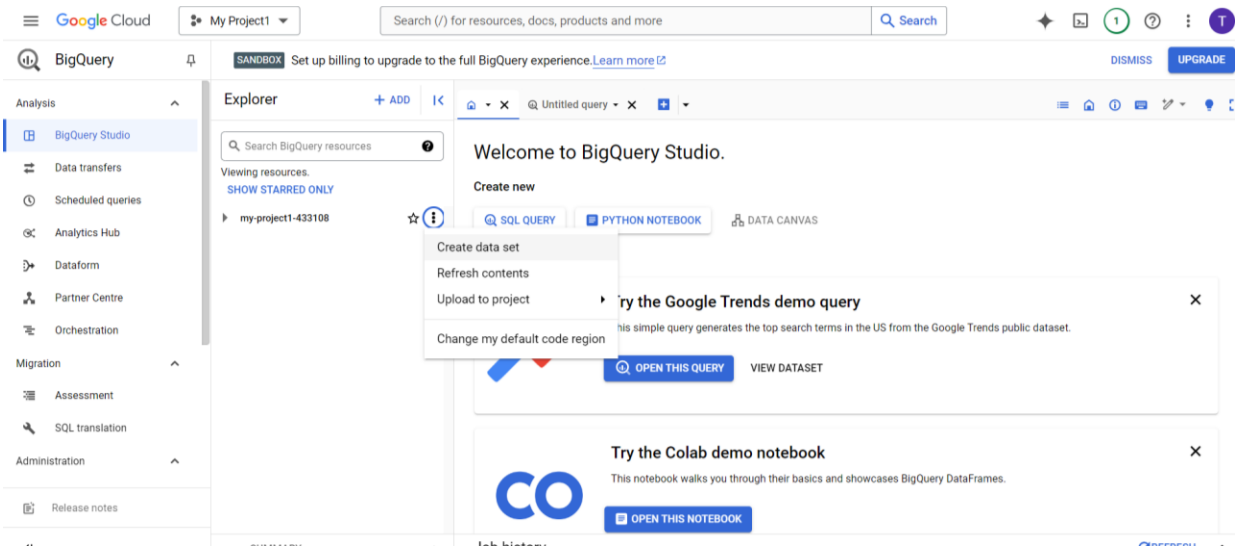
Click the link to the *automobile_data.csv* file to download it. Or you may download the .csv file directly from the attachments below.

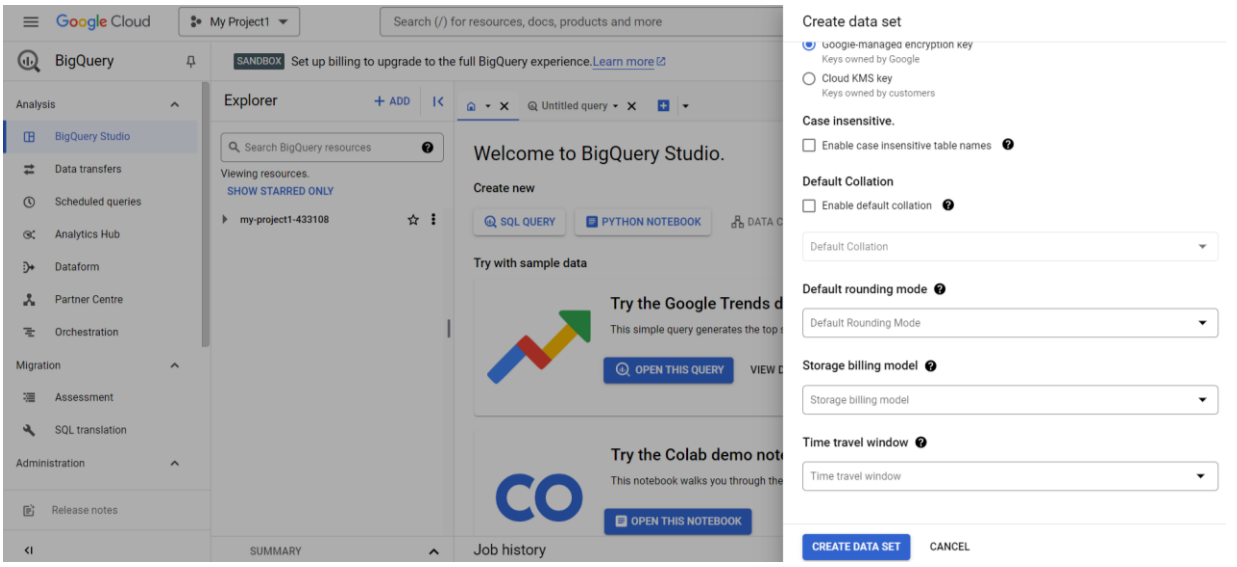
Step 2: Create a dataset

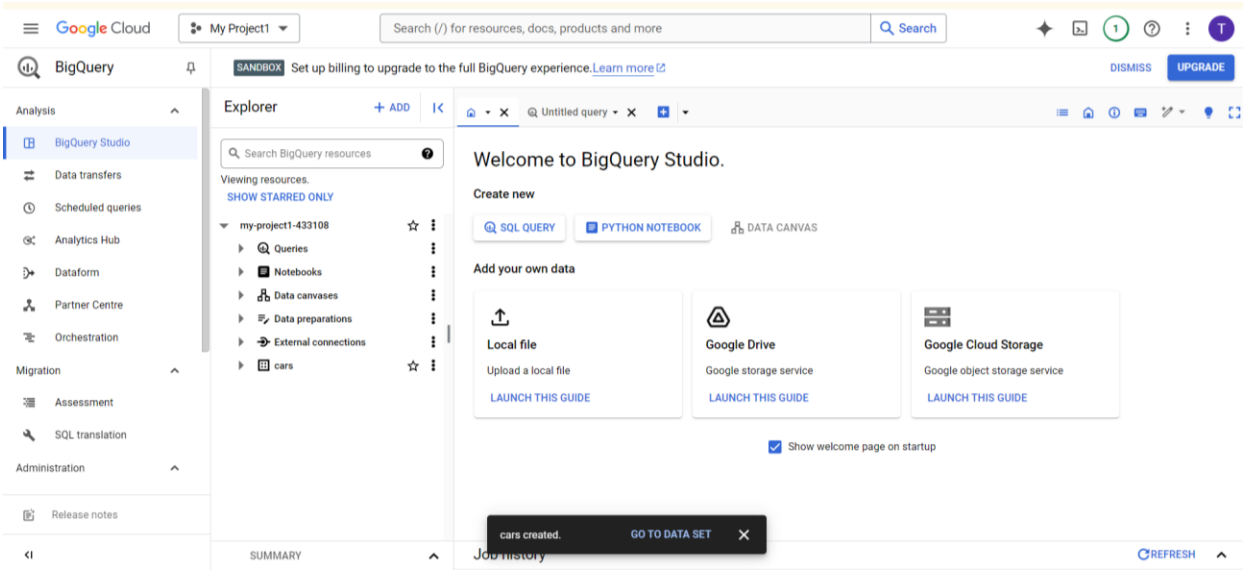
Once you've downloaded the automobile_data.csv file, create your dataset.

Go to the Explorer pane in your workspace and click the three dots next to your personal project name to open the drop-down 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 keep the Location type as Multi-region, US (multiple regions in United States), and the Encryption as Google-managed encryption key default settings. Then, click the CREATE DATASET button.







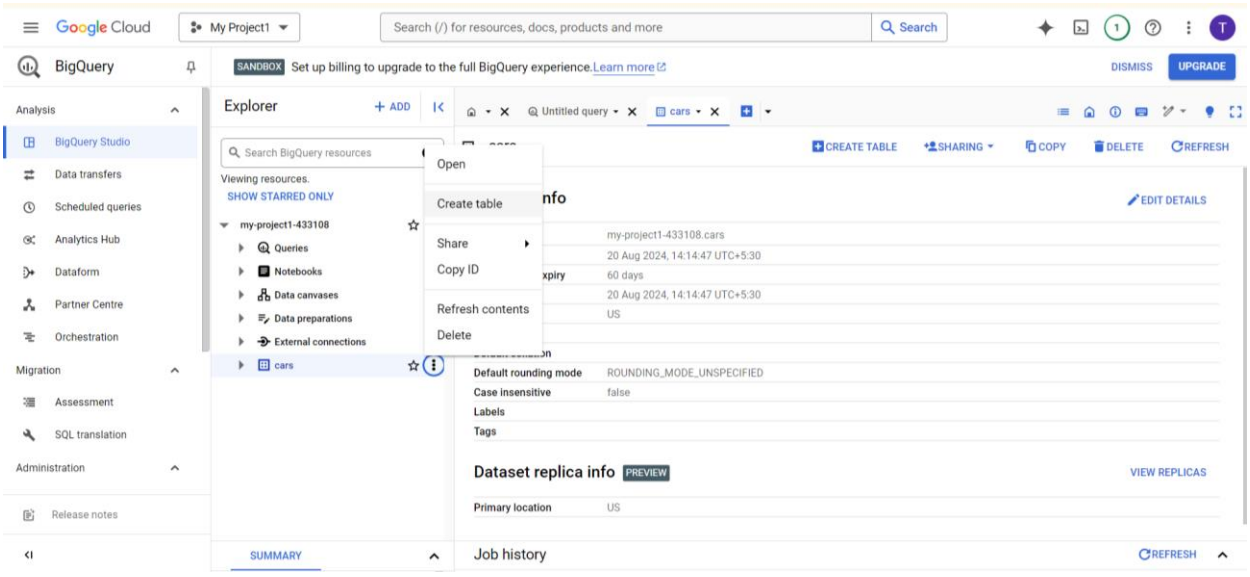
The **cars** dataset appears under our project in the Explorer pane as shown above.

Step 3: Create a table

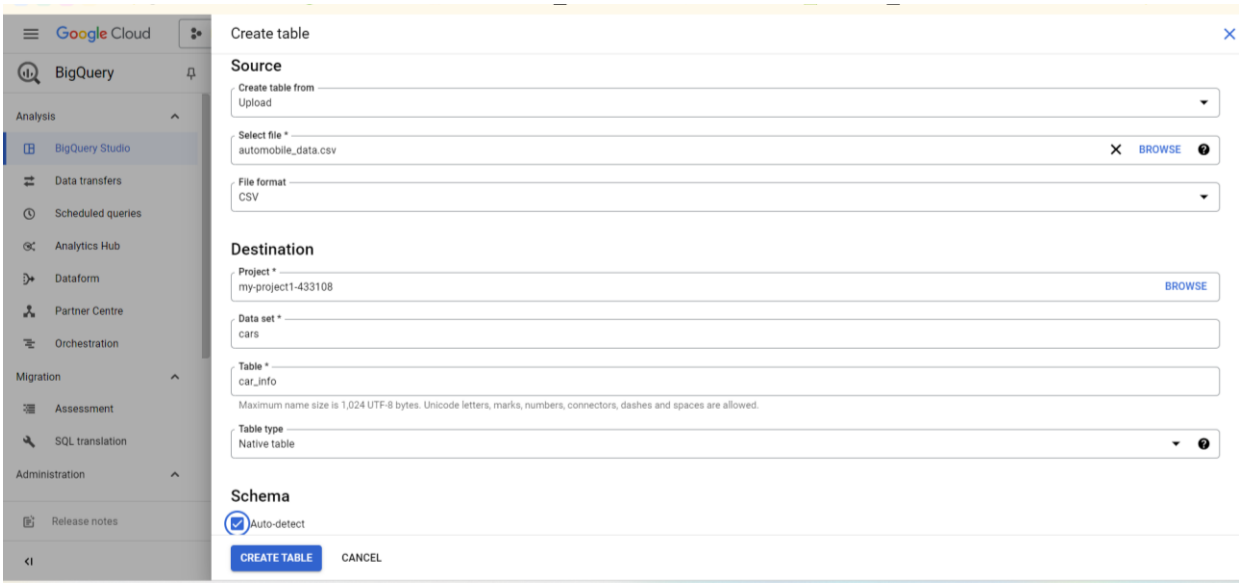
Now that we've created a dataset. We'll create a custom table to house our data. This will enable us to use SQL queries to explore and clean data.

After clicking on cars to open our newly created dataset, you will be able to add a custom table for the insertion of your downloaded data.

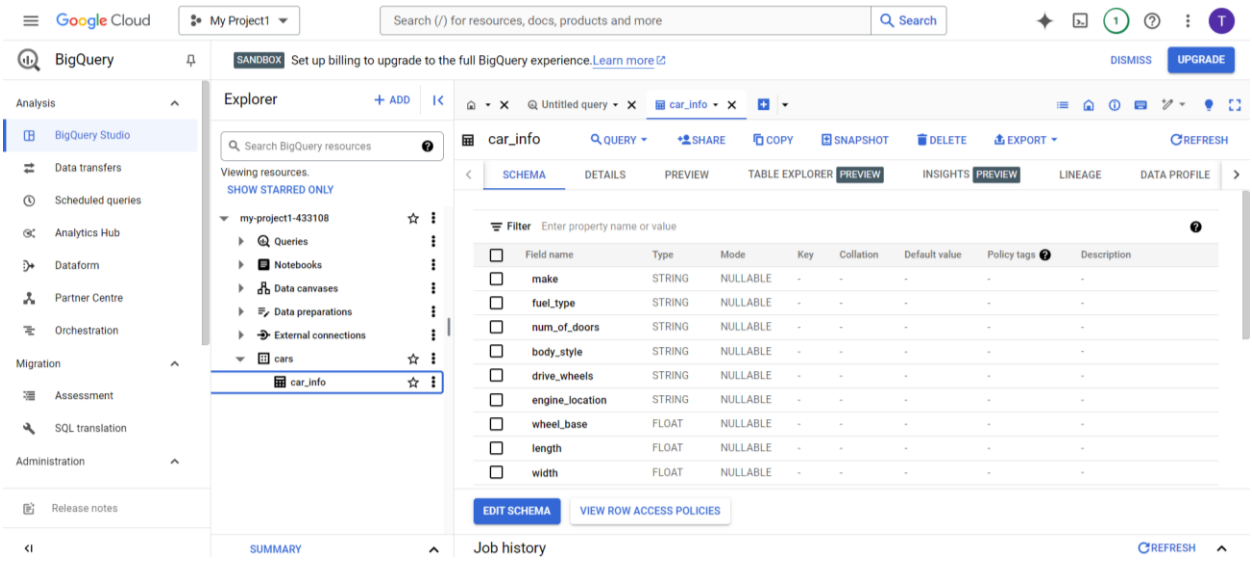
From the cars dataset info window, click CREATE TABLE.



Within the Create table window, upload the automobile_data.csv by clicking the drop-down arrow under Source and choosing the Upload option. Click the BROWSE button and navigate to the folder where your .csv document is located and notice the File format will automatically change to CSV. Ensure the dataset name is cars and name your table car_info. Set the schema to Auto-detect, and finally click the Create table button.



After creating our table, it will appear in our Explorer pane as we can see in the above image. We can click on the newly created table, car_info, to explore the SCHEMA and DETAILS buttons within your data page. Once you have gotten familiar with your data, you can start querying it.



Here we could see SCHEMA, DETAILS,PREVIEW etc.

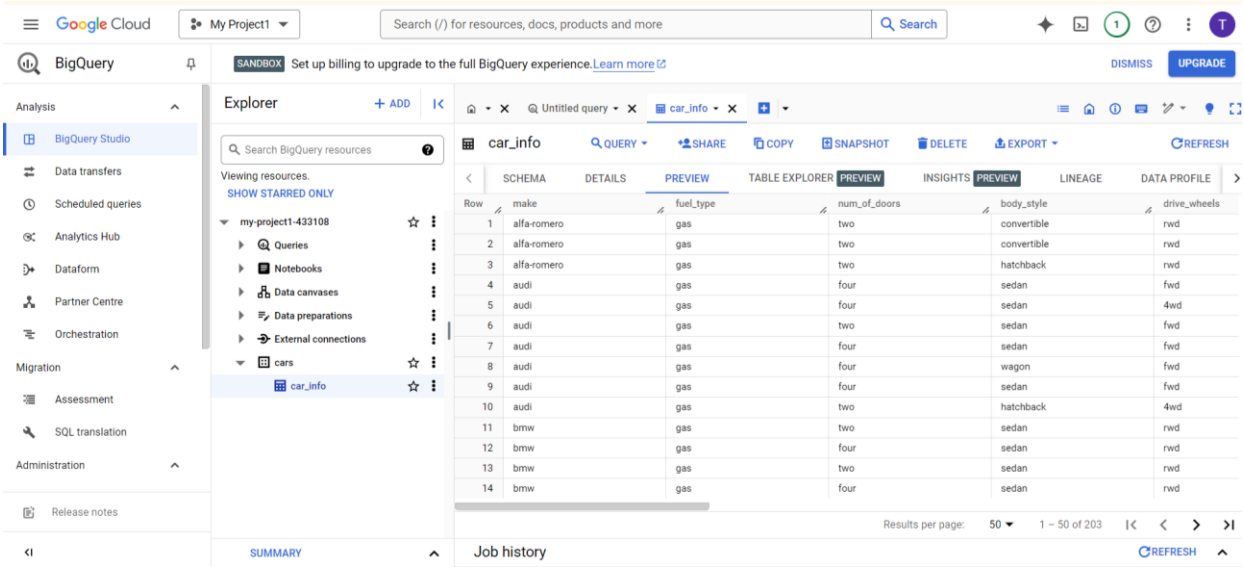
Step 4: Understanding why we need to clean our data

Our 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 our analysis, we'll need to make sure your data is clean. If we analyze dirty data, we could end up presenting the wrong list of cars to the investors. That may cause them to lose money on their car inventory investment.

Continue to next step to clean our data.

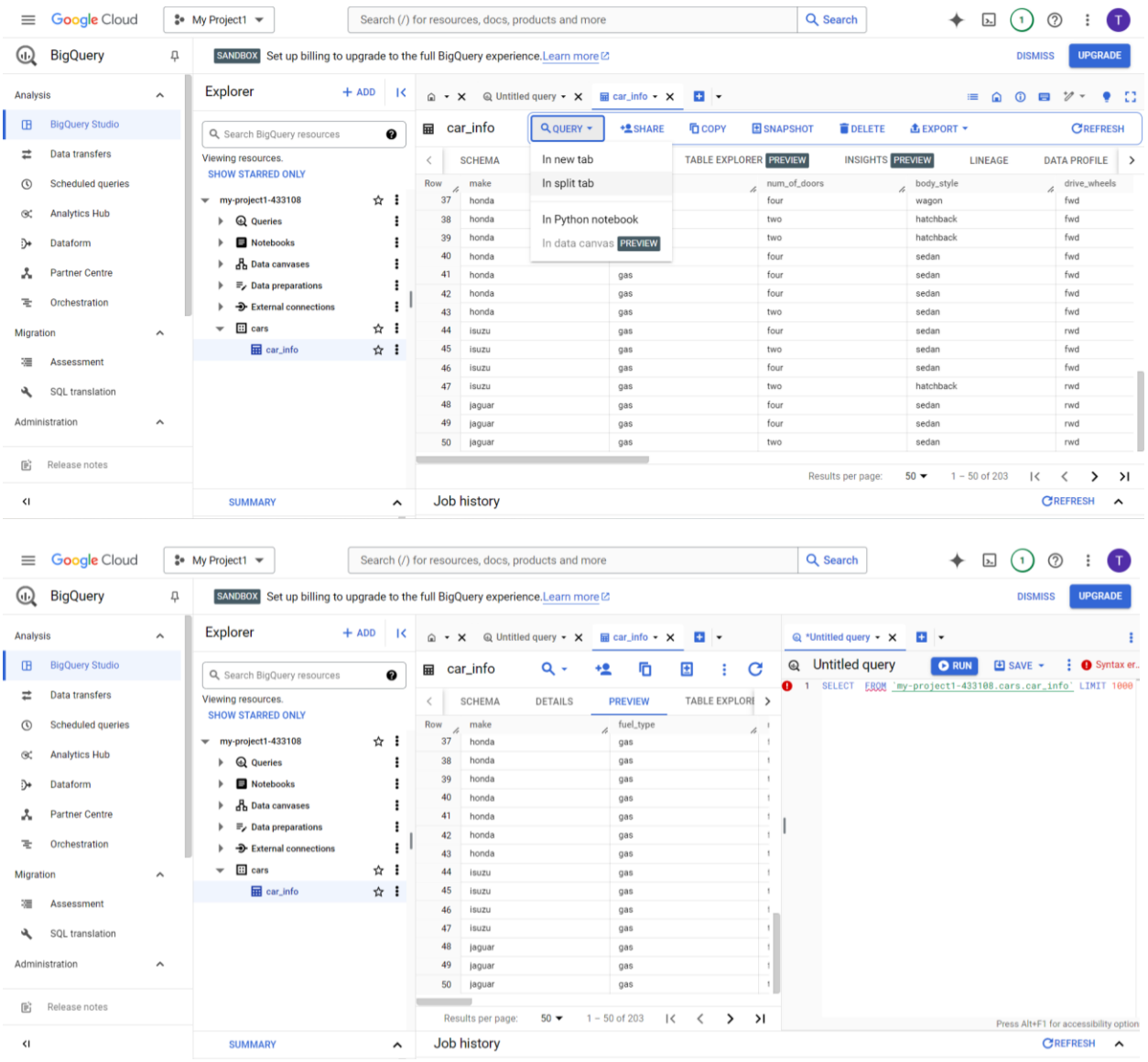
Step 5: Inspect the fuel_type column

The first thing we want to do is inspect the data in our table so we can find out if there is any specific cleaning that needs to be done. Get an initial understanding of the data table by clicking on the PREVIEW tab that sits below the car_info toolbar.



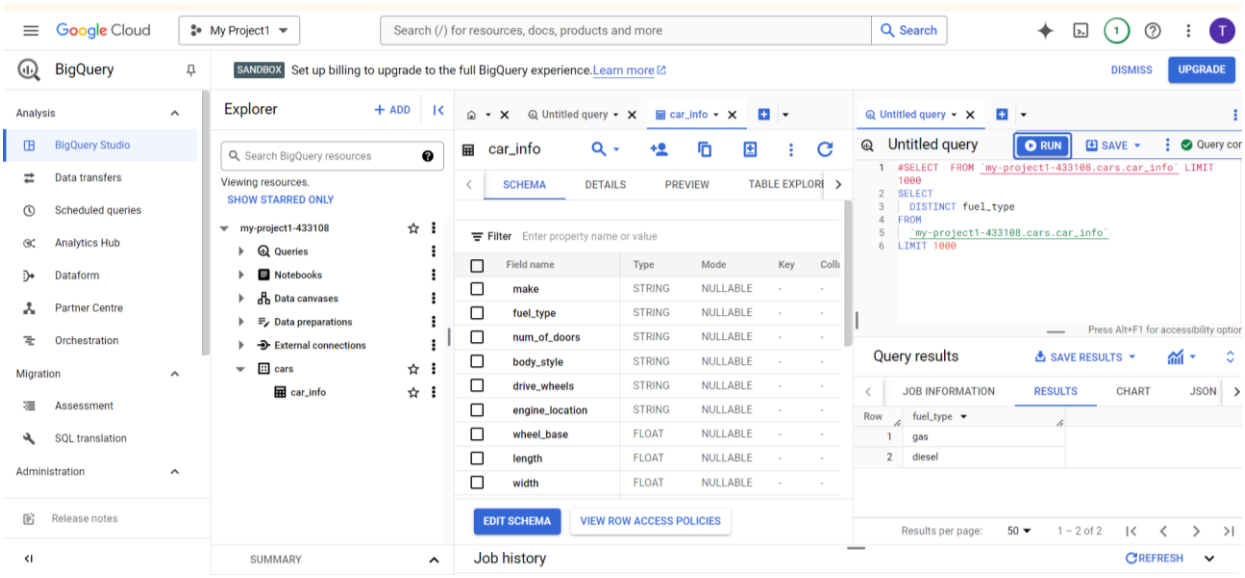
This is the PREVIEW table.

We know that 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. You can generate the default query setup by clicking on the **QUERY** button and selecting the **In split tab**.



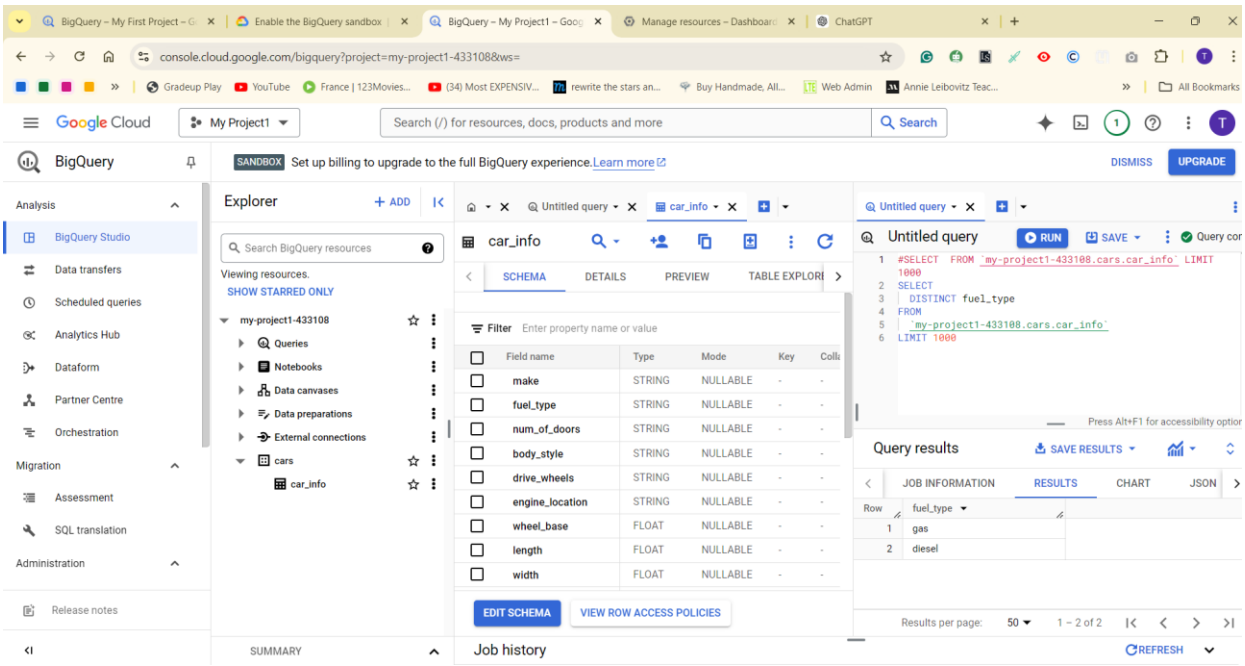
This shows us a dual view of the info window and the query.

Next, we can generate the first query in the workspace:



NOTE: Within the **FROM** clause of the syntax above, you will need to begin the **Table ID** line with your personalized project name, period, the dataset name, period, and end with the table name. It's important to understand that the personal project name will be unique to each learner. You can also locate and copy the full **Table ID** filename by clicking on the **DETAIL** option tab in your **car_info Table info** window. Once copied, paste it after the **FROM** clause and run the above query.

This returns the following results:

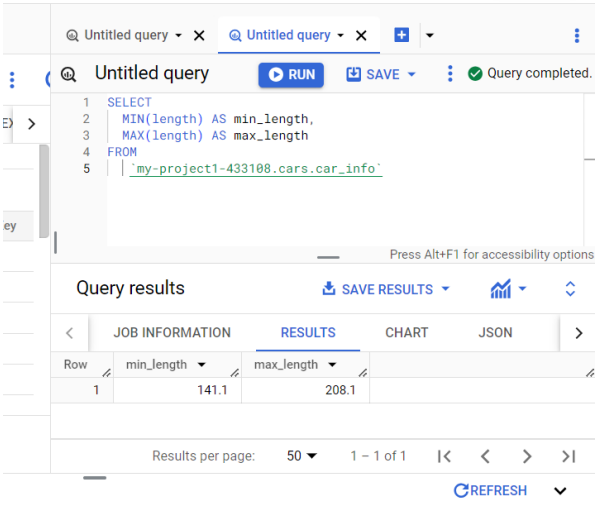


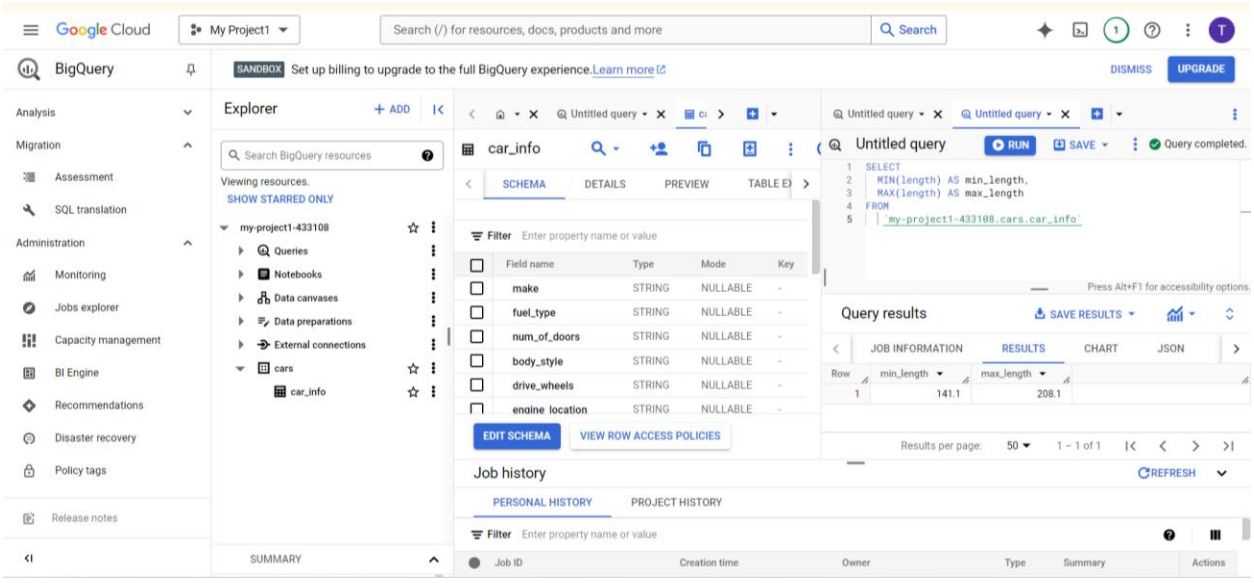
We could see gas and diesel results at the bottom right.

This confirms that the **fuel_type** column doesn't have any unexpected values. Also note that the default **LIMIT 1000** is added to your query, but in this case, BigQuery is only returning two distinct fuel types.

Step 6: Inspect the length column

We inspect a column with numerical data. The **length** column should contain numeric measurements of the cars. So we will check that the minimum and maximum lengths in the dataset align with the data description.



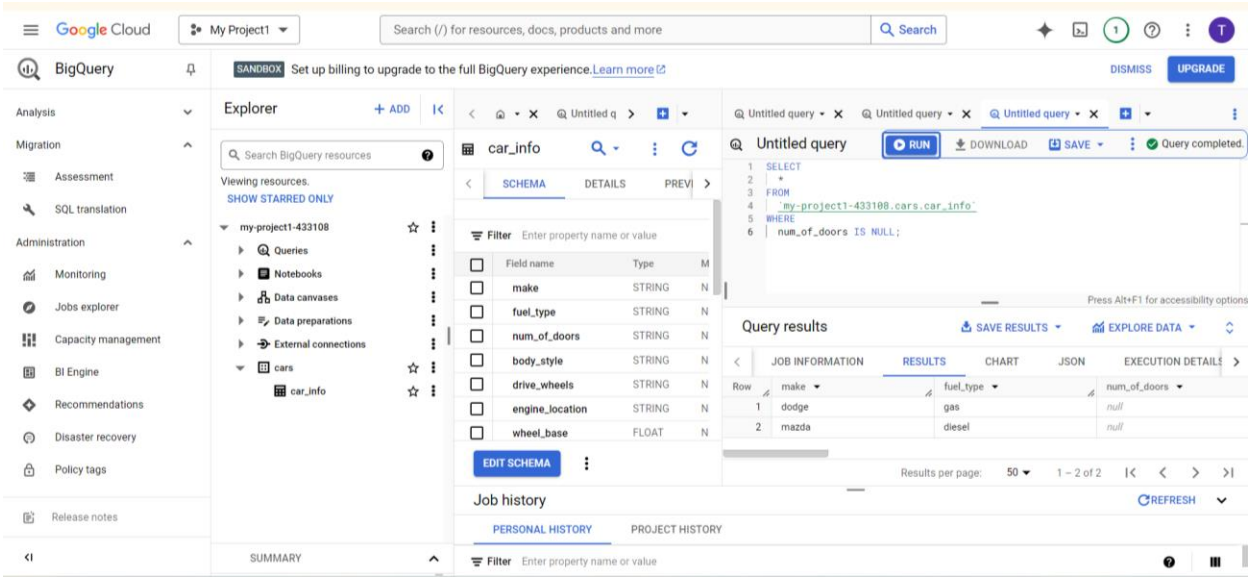


Our results confirm that 141.1 and 208.1 are the minimum and maximum values respectively in this column.

Step: Fill in missing data

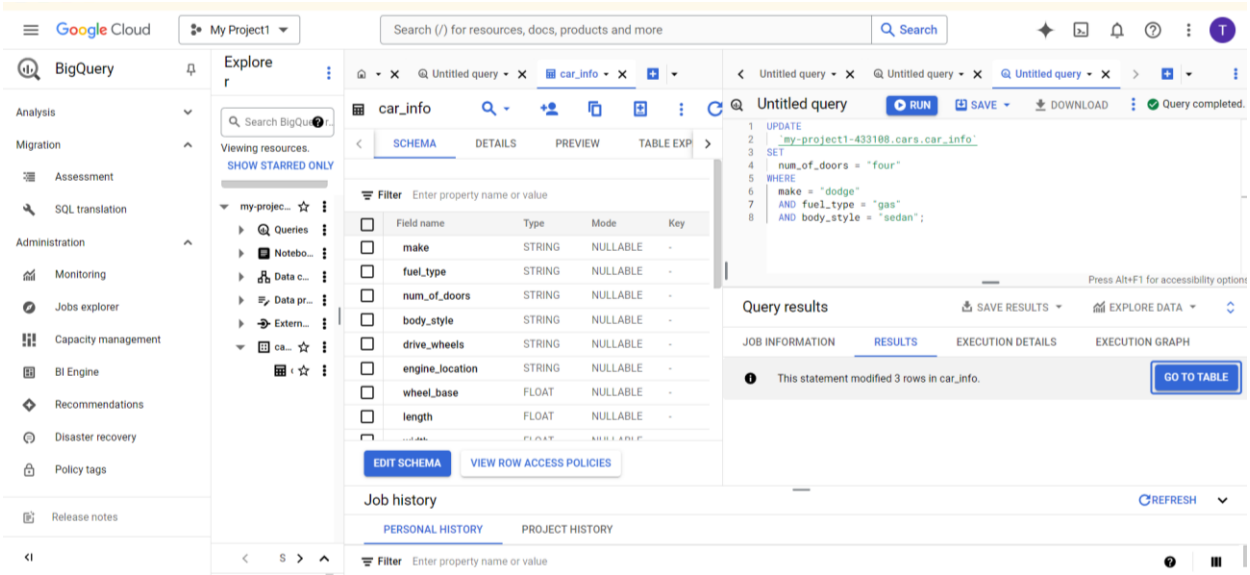
Missing values can create errors or skew our results during analysis. We're going to want to check our data for null or missing values. These values might appear as a blank cell or the word null in BigQuery.

We can check to see if the **num_of_doors** column contains null values using this query:

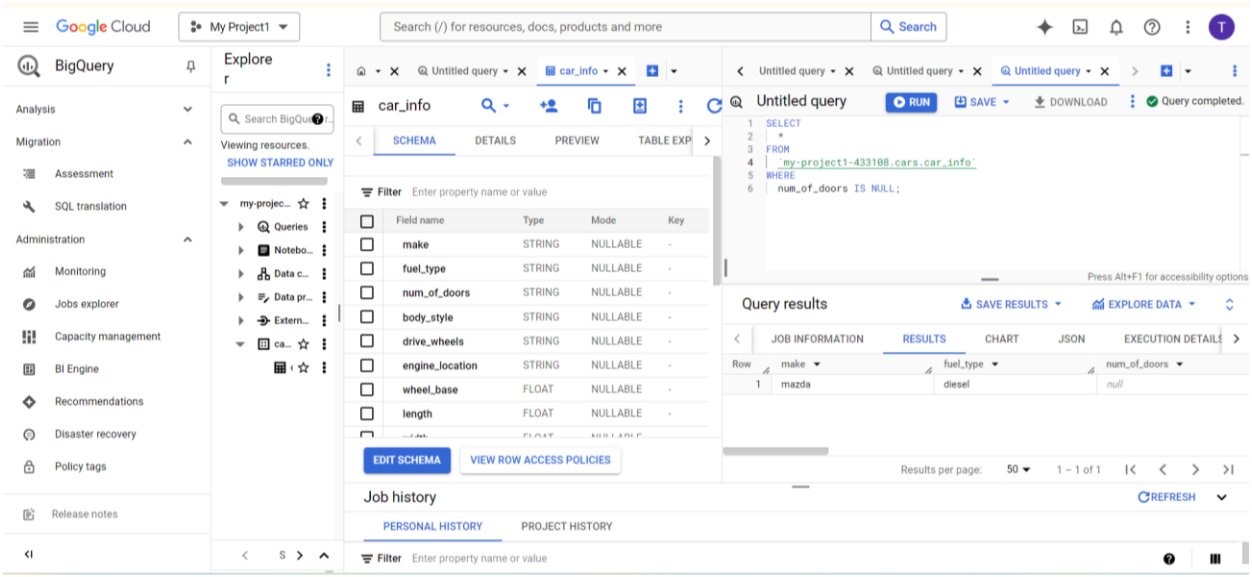


This will select rows with missing data for the **num_of_doors** column and return them in our results table. We get two results, one Mazda and one Dodge as shown in the above image.

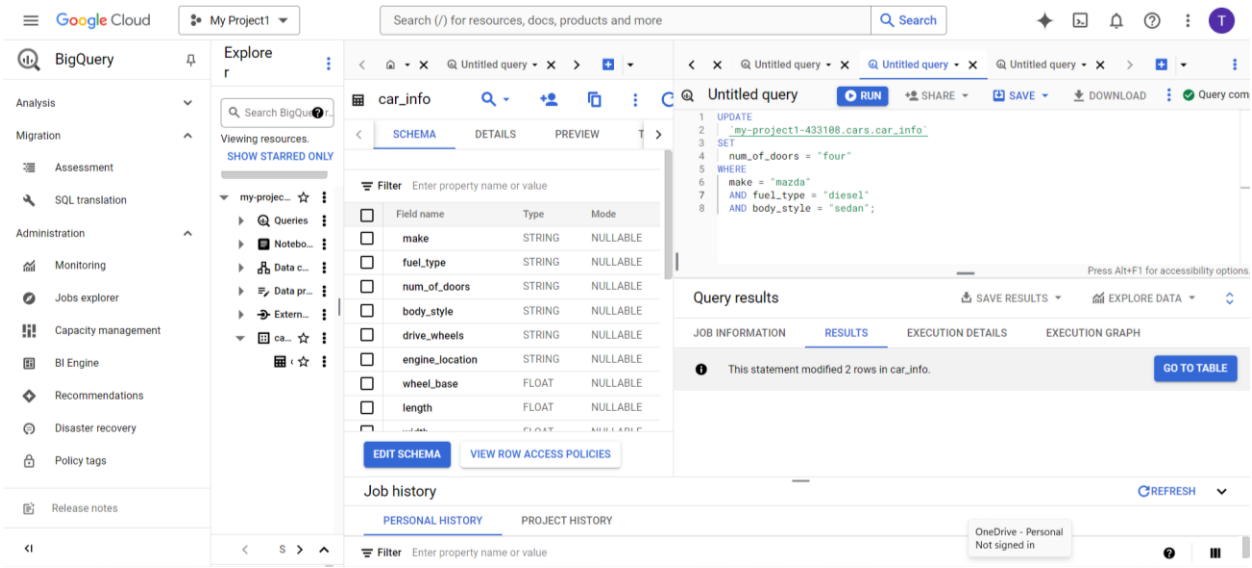
In order to fill in these missing values, we check with the sales manager, who states that all Dodge gas sedans and all Mazda diesel sedans sold had four doors. If we are using the BigQuery free trial, we can use this query to update our table so that all Dodge gas sedans have four doors: we use UPDATE,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



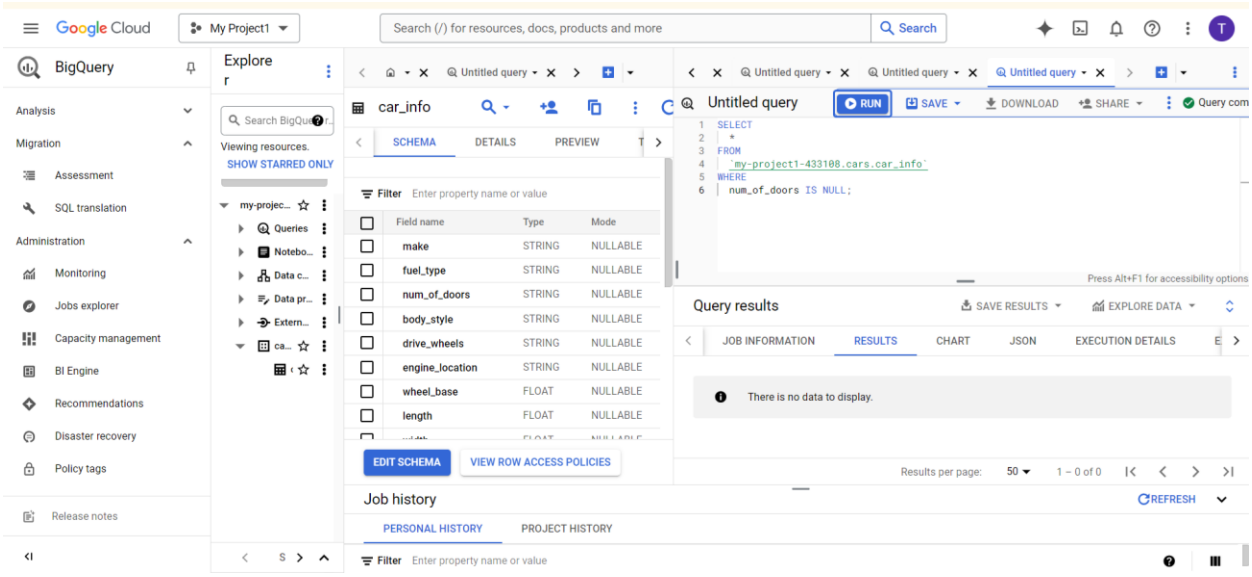
We get a message telling that three rows were modified in this table. To make sure, we can run the previous query again: using SELECT, FROM, WHERE



Results show that we only have one row with a null value for **num_of_doors**. Repeat this process to replace the null value for the Mazda.



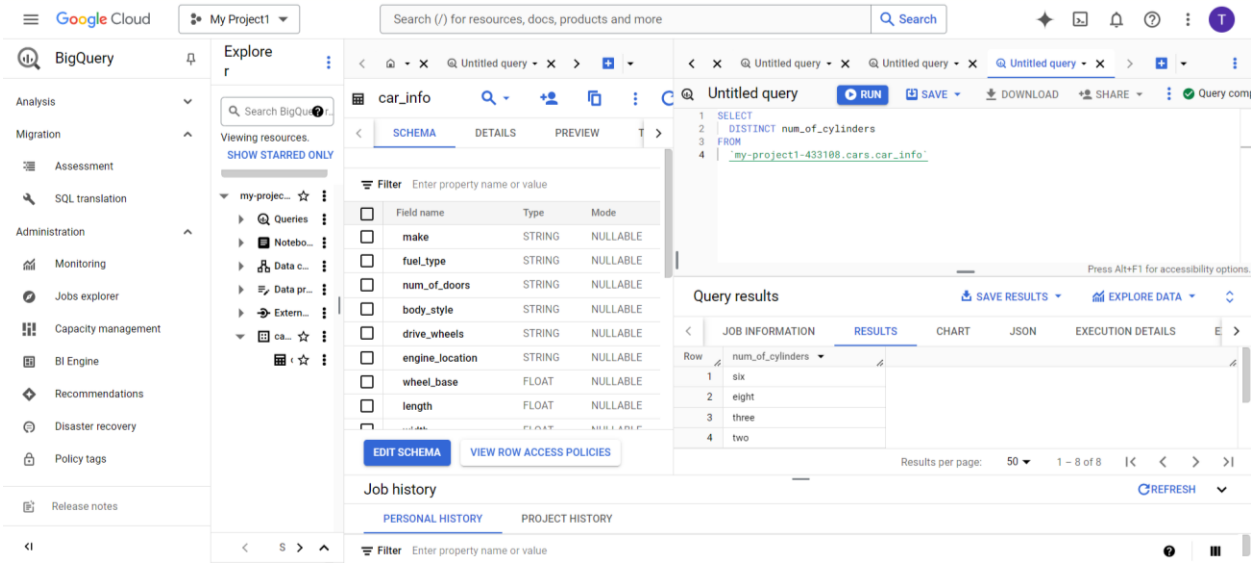
We get a message telling that two rows were modified in this table. To make sure, we can run the previous query again: using SELECT, FROM, WHERE



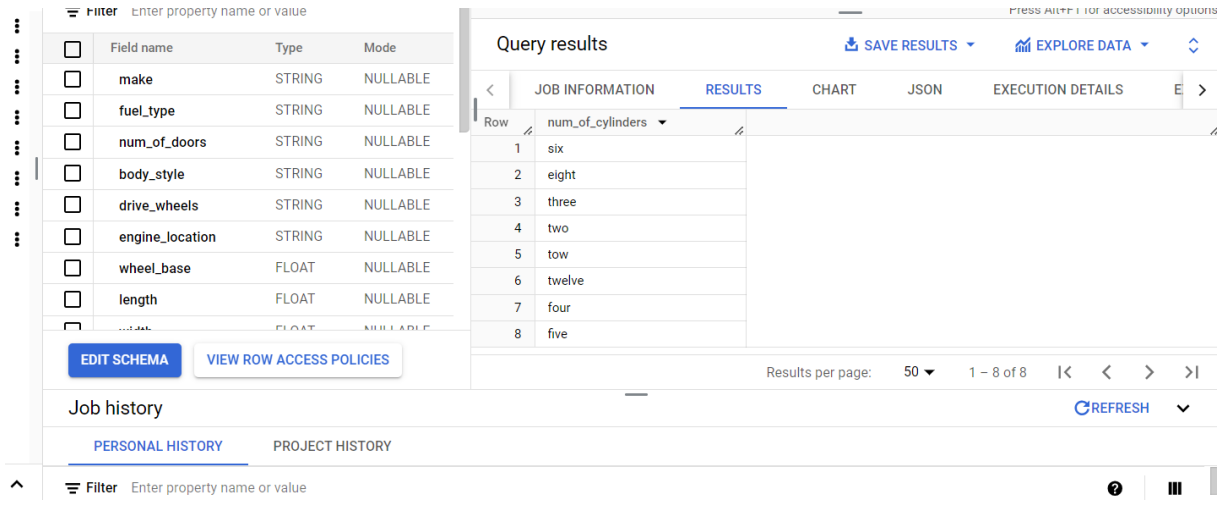
We now see that there is no data too display.

Step 8: Identify potential errors

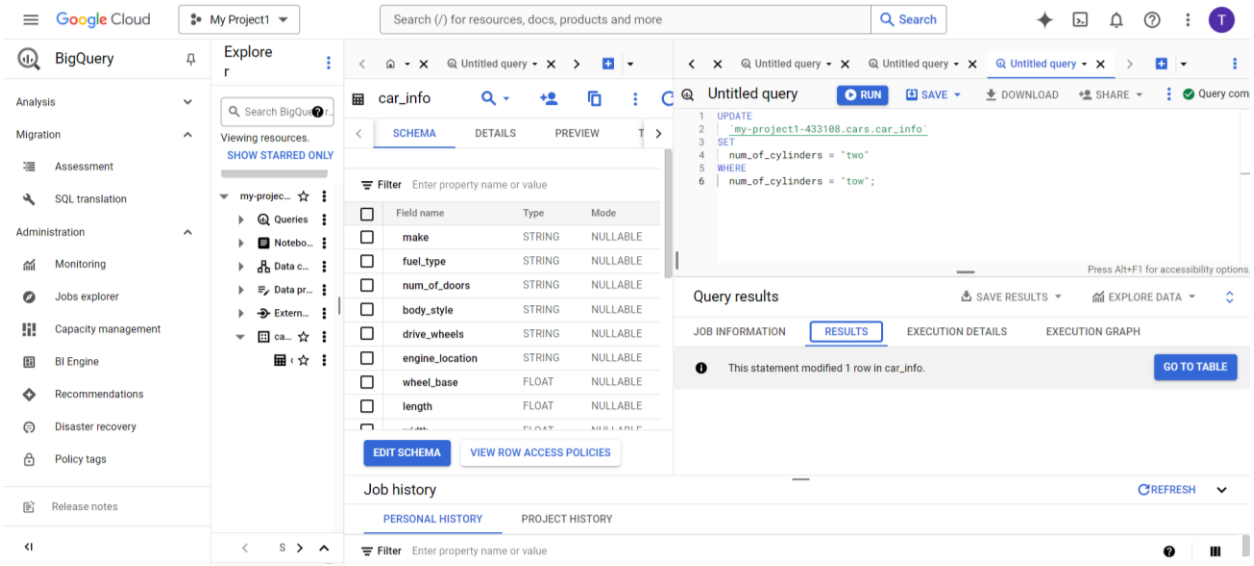
Once we have finished ensuring that there aren't any missing values in our data, we'll want to check for other potential errors. We can use **SELECT DISTINCT** to check what values exist in a column. We can run this query to check the **num_of_cylinders** column:



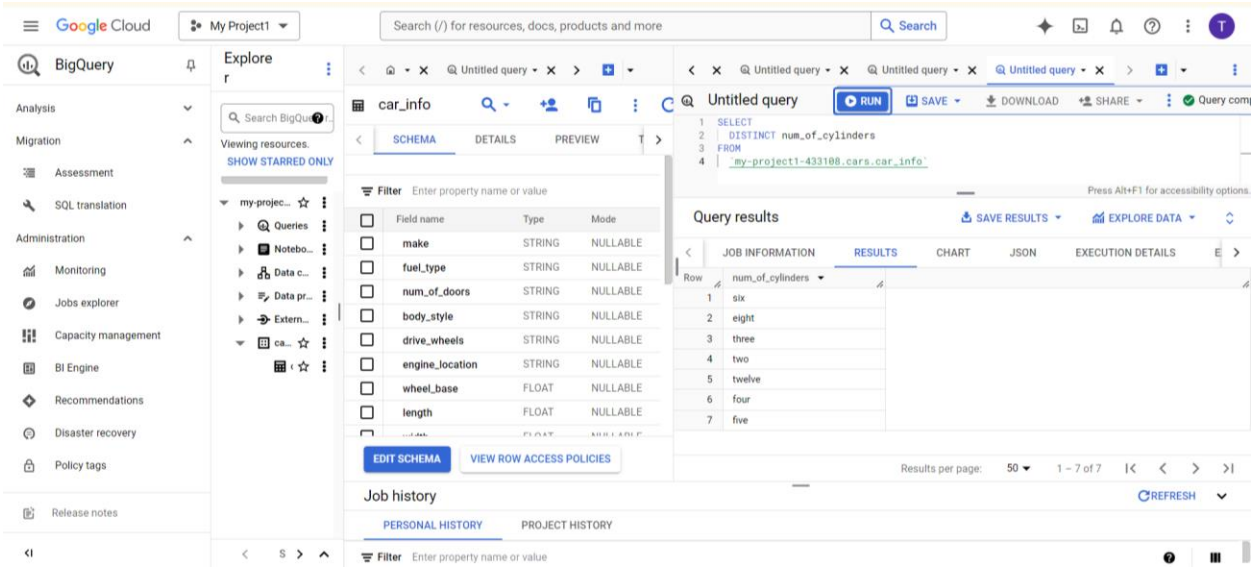
After running this, we notice that there are one too many rows. There are two entries for two cylinders: rows 4 and 5. But the *two* in row 5 is misspelled.



To correct the misspelling for all rows, we can run this query if you have the BigQuery free trial: Query contains UPDATE, SET, WHERE

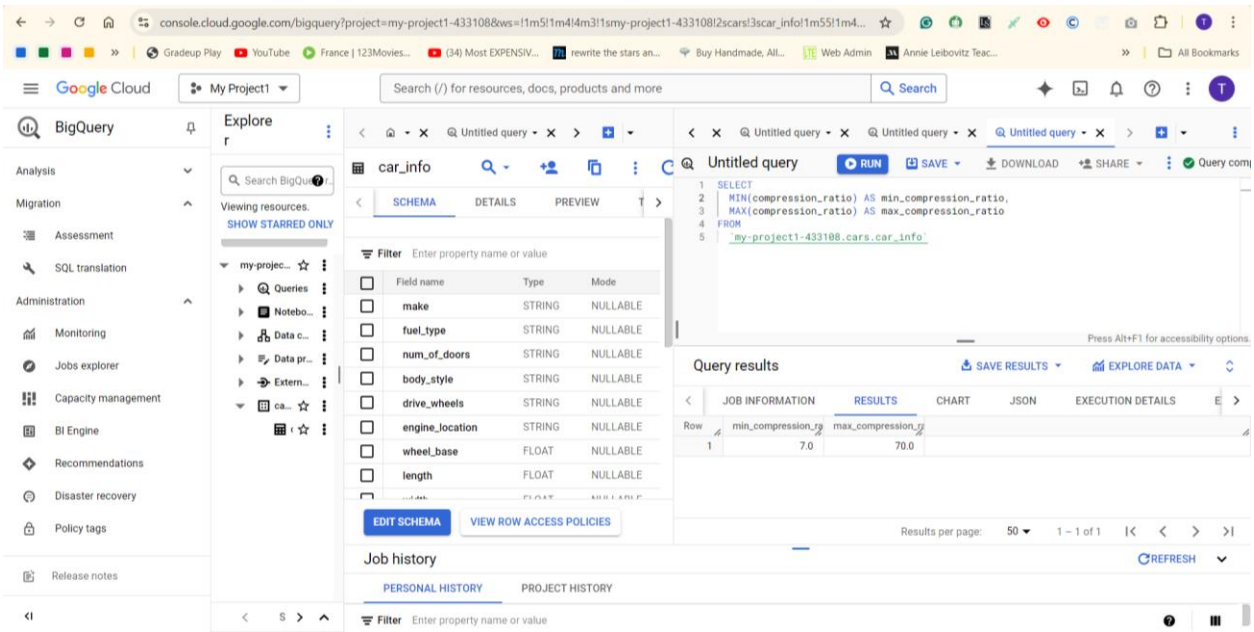


The results show this statement modified one row in car_info. To check that it worked, we can run the previous query having SELECT, DISTINCT again:

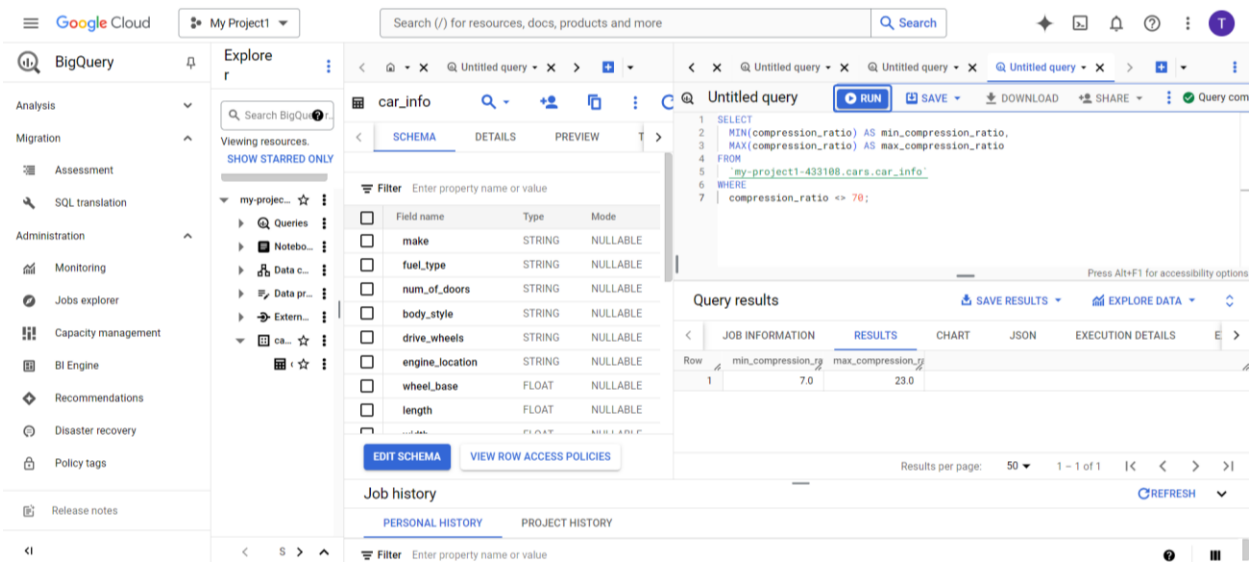


The results show that correction were carried out successfully.

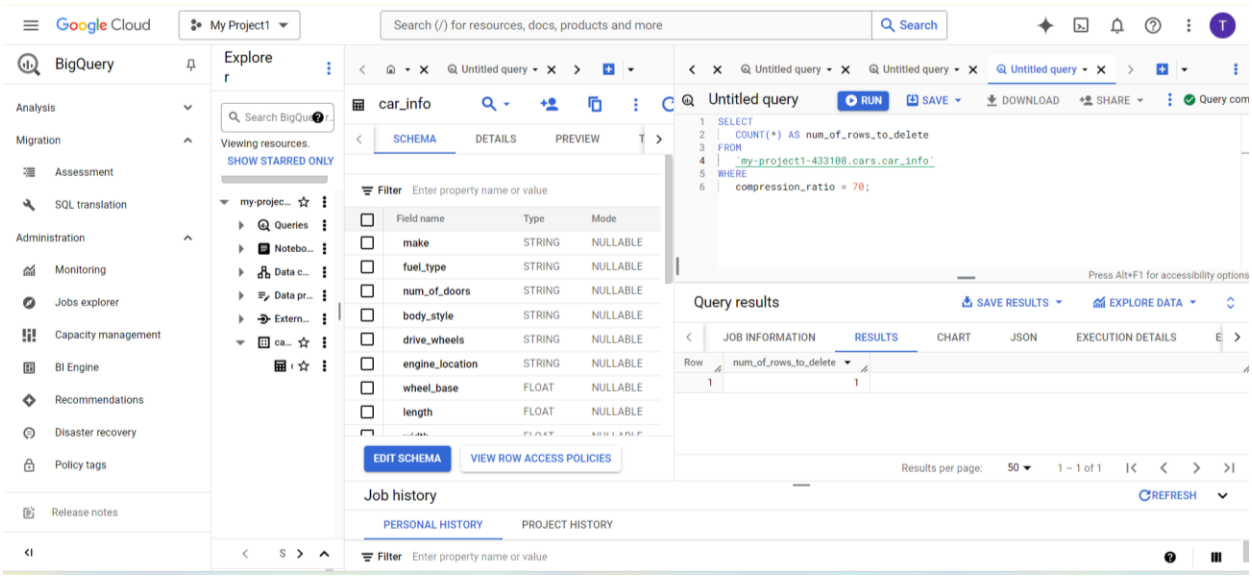
Next, we can check the **compression_ratio** column. According to the data description, the **compression_ratio** column values should range from 7 to 23. Just like when you checked the length values, we can use **MIN** and **MAX** to check if that's correct:



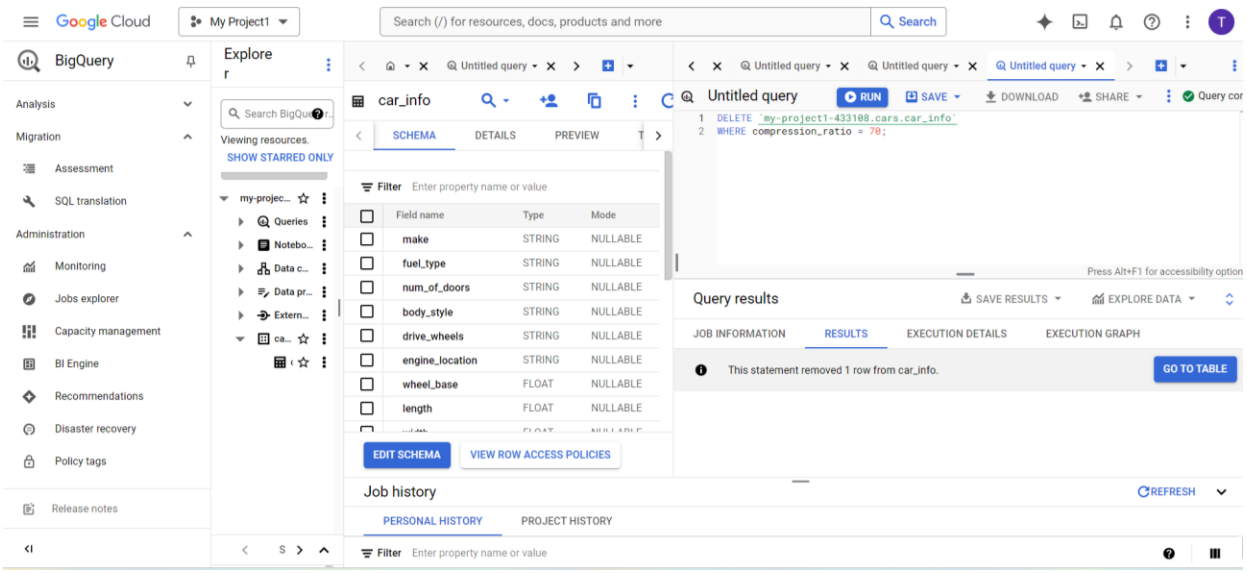
We find that **this returns a maximum of 70**. But we know this is an error because the maximum value in this column should be 23, not 70. So the 70 is most likely a 7.0. Run the above query again without the row with 70 to make sure that the rest of the values fall within the expected range of 7 to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



Now the highest value is 23, which aligns with the data description. So we'll want to correct the 70 value. We check with the sales manager again, who says that this row was made in error and should be removed. Before we delete anything, we should check to see how many rows contain this erroneous value as a precaution so that you don't end up deleting 50% of our data. If there are too many (for instance, 20% of our rows have the incorrect 70 value), then we 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 SELECT COUNT below to count how many rows we would be deleting:



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

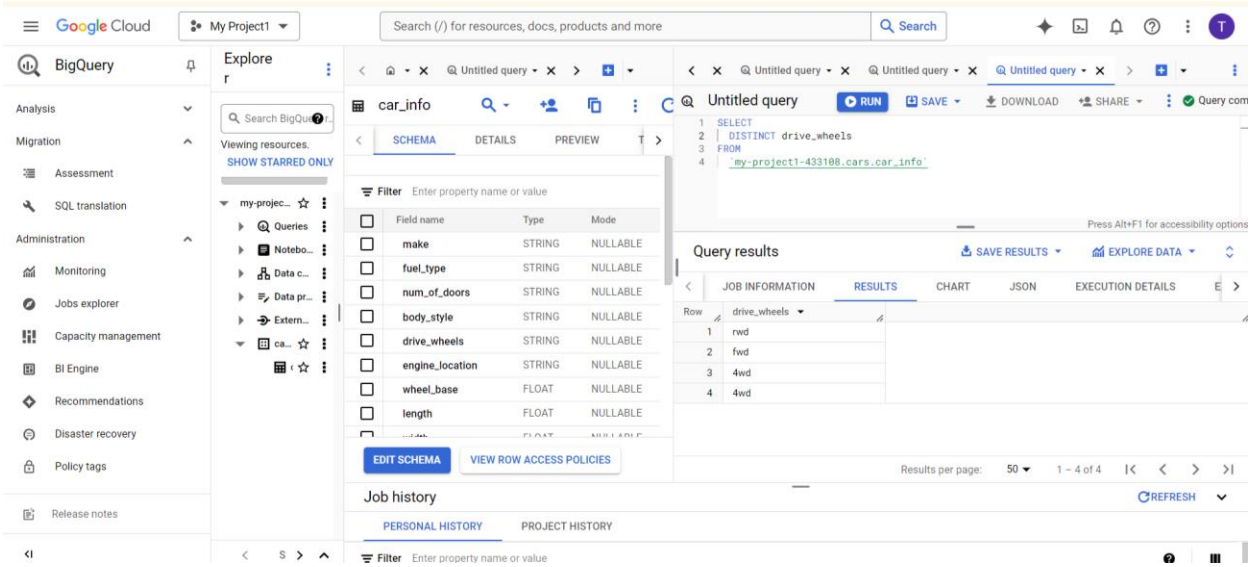


We find that this statement removed 1 row, we can also check using the previously used query SELECT DISTINCT

Step 9: Ensure Consistency

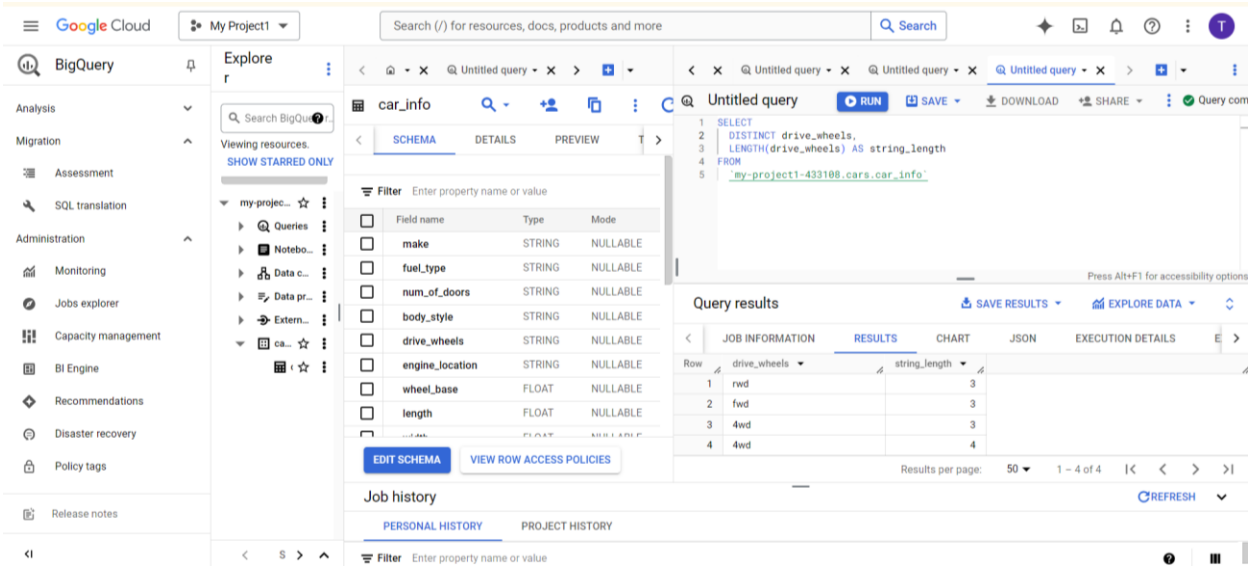
Finally, we want to check our data for any inconsistencies that might cause errors. These inconsistencies can be tricky to spot—sometimes even something as simple as an extra space can cause a problem.

Check the **drive_wheels** column for inconsistencies by running a query with a **SELECT DISTINCT** statement:

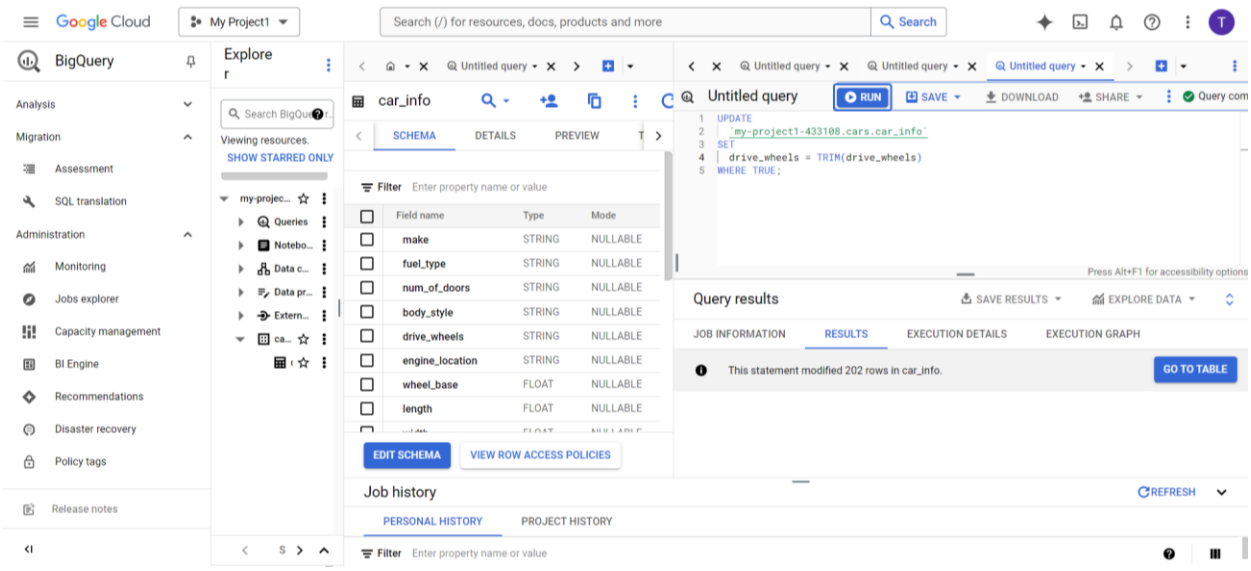


It appears that 4wd appears twice in results. However, because we used a **SELECT DISTINCT** statement to return unique values, this probably means there's an extra space in one of the 4wd entries that makes it different from the other 4wd.

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

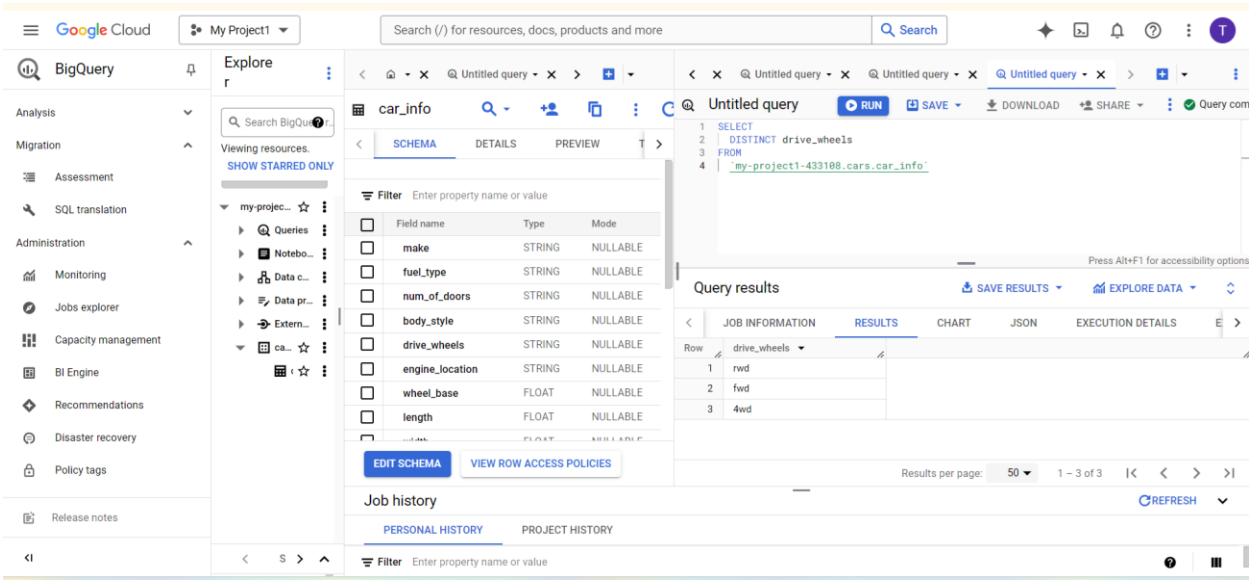


According to the results, some instances of the 4wd string have four characters instead of the expected three (4wd has 3 characters). In that case, we can use the **TRIM** function to remove all extra spaces in the **drive_wheels** column if you are using the BigQuery free trial:



The results show that this statement modified 202 rows.

Then, we run the **SELECT DISTINCT** statement again to ensure that there are only three distinct values in the drive_wheels column:



And now only three unique values are in this column! Which means our data is clean, consistent, and ready for analysis!

Results:

The data cleaning process successfully identified and corrected several issues within the dataset:

- Fuel Type Validation: Confirmed that the fuel_type column contained only expected values (gas and diesel), ensuring reliability in fuel-based analysis.
- Numerical Integrity: Verified the length and compression_ratio columns, correcting outliers and ensuring that all values fell within the expected ranges, thus preserving the accuracy of car feature metrics.
- Missing Values: Addressed missing data in the num_of_doors column by consulting with the sales manager, resulting in accurate and complete records.
- Data Consistency: Rectified inconsistencies in the drive_wheels column by removing extraneous spaces, which standardized the entries and improved query performance.

These corrections led to a clean, consistent dataset, ready for in-depth analysis to identify the top 10 most popular car models based on historical sales data.

Through a thorough data cleaning process, we successfully prepared the dataset for analysis, ensuring it was free of errors, missing values, and inconsistencies. This clean dataset enabled accurate identification of the top 10 most popular car models based on historical sales data. These insights are vital for investors, as they provide a clear understanding of customer preferences,

allowing the dealership to optimize inventory and maximize profitability. By achieving this, the project meets the investors' goal of identifying popular car models to guide their stocking decisions.

Conclusion:

By implementing best practices in SQL data cleaning, we transformed a raw, error-prone dataset into a reliable resource for business analysis. This project not only improved data quality but also enhanced query performance and laid a strong foundation for accurate insights. The cleaned data empowers the dealership to make data-driven decisions on inventory stocking, thereby maximizing profitability and meeting customer demands.