4

**Visualizing and Exporting Your Queries**

SQL queries generally return rows and columns which is often enough, especially with small datasets. However, if you have a larger result set, and/or you have the need to share the information, this chapter will illustrate how Azure Data Studio can help. As data professionals it is often helpful to visualize a set of data, as well as to share results and analysis with our co-workers, internal and external clients, or even feed into other workstreams. Instead of relying on external applications, ADS is happily a one-stop-shop to create, run, visualize, and export your database queries.

## Sample Data

Beginning with this chapter, we are going to use a database named ‘car\_crash’ which is a fairly small DataMart created from data provided by the “[NYC OpenData](https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95/data)”[[1]](#footnote-1) website. This dataset provides information on car accidents that have occurred in the 5 major NYC boroughs, over an approximate 8-year period (July 2012 through Feb 2020). The database is provided as a SQL Server 2017 backup (.bak) file which can be downloaded with the following link: <https://bit.ly/3f5bbmK>

After downloading, you can use Azure Data Studio to restore this database using the following steps:

Note: Restoring databases should always be performed with caution as you have the potential to overwrite an existing database.

1. Unzip the downloaded file called: car\_crash.zip (which is most likely located in your ‘download’ folder), and place the .bak file a convenient location to be referenced in step 5 below. For this example, I have placed the unzipped file (called: car\_crash.bak) into folder C:\temp.
2. Using a ‘Server Level’ connection from the SERVERS sidebar, right-click the SQL Server where you would like to ‘restore’ this database, and select ‘Manage’ as shown in figure 4-1:

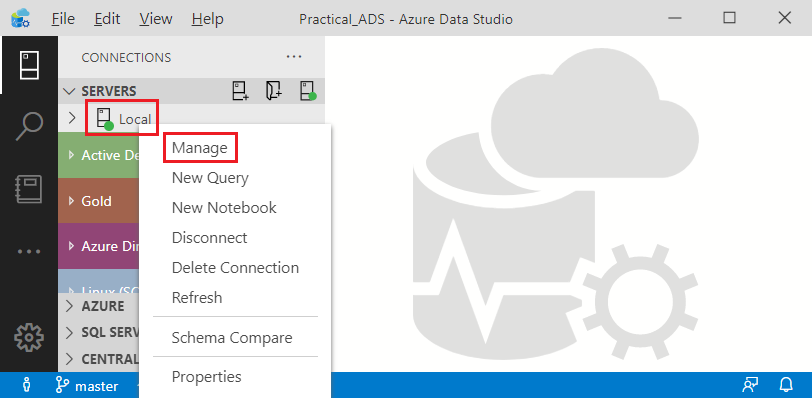


Figure 4-1. Manage a Server Level Connection

1. Click on ‘Restore’ from the Server ‘Dashboard’ as displayed in figure 4-2:

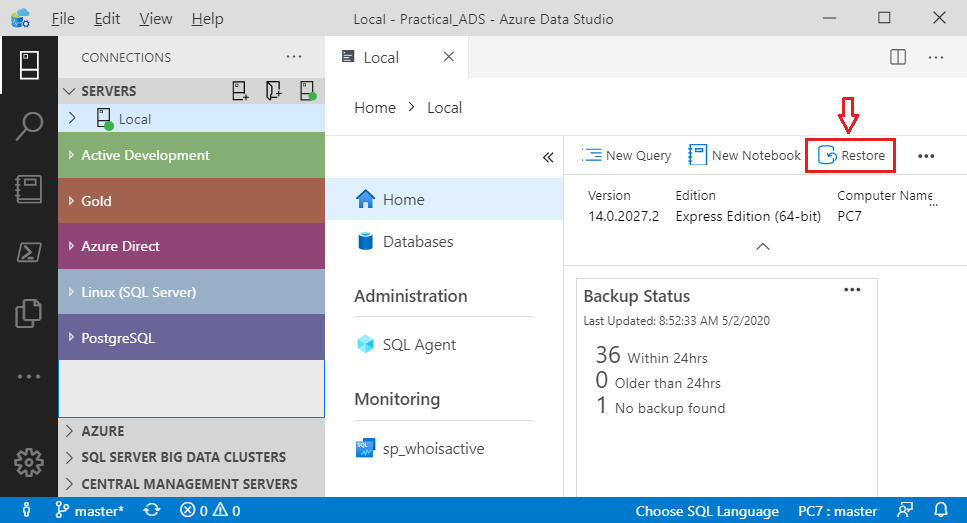


Figure 4-2. Restore Task from Server Dashboard

1. Select ‘Backup file’ in the ‘Restore from’ field
2. Click the ellipses (...) in the Backup file path field and navigate to the location where you stored the downloaded backup file (in this example we are using folder: C:\temp) and click on the file called: car\_crash.bak. Your screen should look like figure 4-3:

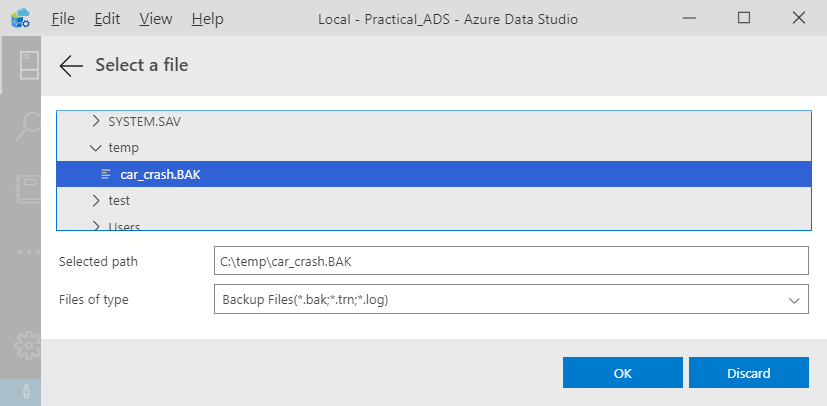


Figure 4-3. Restore Task from Server Dashboard

1. Click on ‘OK’ and you should see the ‘Restore plan” as shown in Figure 4-4. Also be sure you have enough disk storage for this restore action. The database ‘size’ for the car\_crash database is nearly 1 GB:

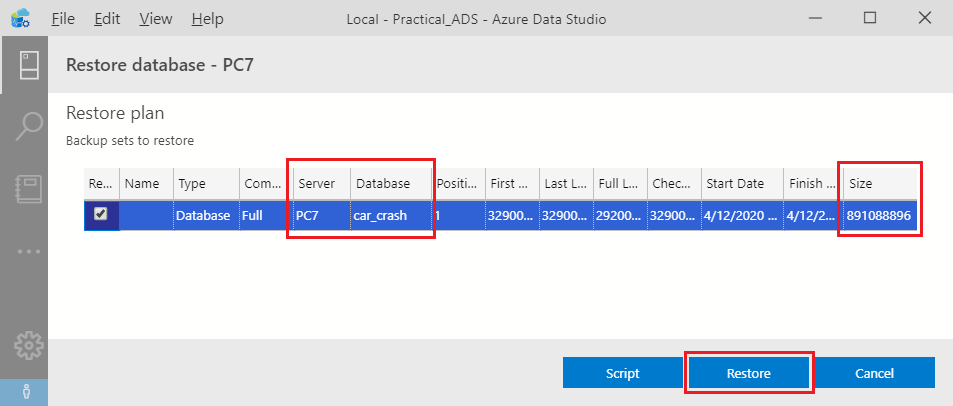


Figure 4-4. Restore Plan from the Restore Form

Finally, verify the ‘Server’ and ‘Database’ names which are boxed in red in figure 4-4, and click ‘Restore’. Once complete, you should see the car\_crash database listed in the Server Dashboard by clicking on the ‘Databases’ tab as displayed in Figure 4-5:

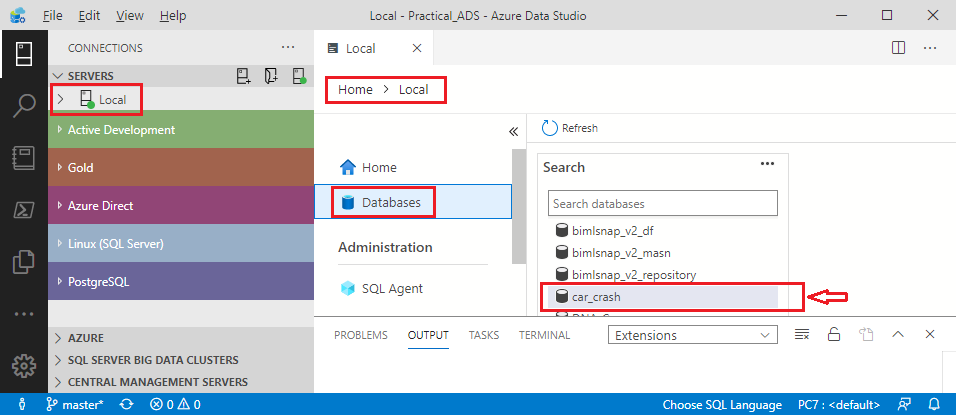


Figure 4-5. Viewing Databases with the Server Dashboard

For more details on the schemas, tables, and relationships of this ‘sample’ database, please refer to Appendix ‘A’ – Understanding the car\_crash DataMart.

# Anatomy of the Results Pane

With our newly installed car\_crash database, we are now set to write SQL queries, and create interesting tabular, as well as visual results. First let’s open a new query from the server dashboard using the car\_crash database as shown in Figure 4-6:

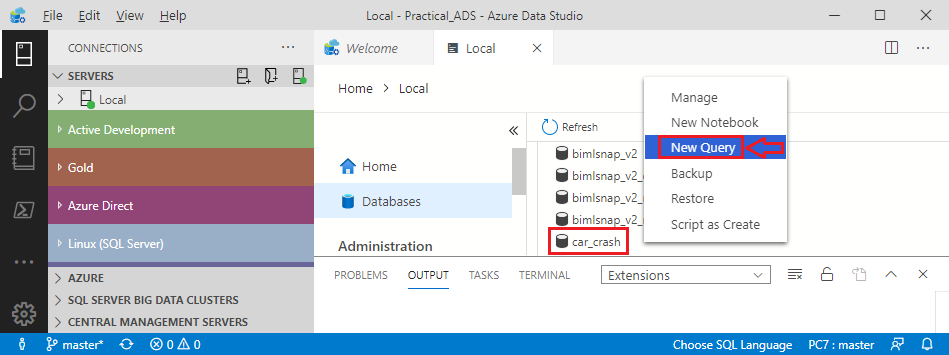


Figure 4-6. Create a New SQL Query for the car\_crash database

Next enter the query:

select distinct BOROUGH

  from edw.D\_LOCATION

And then click on ‘run’. You should now see the query results as shown in Figure 4-7:

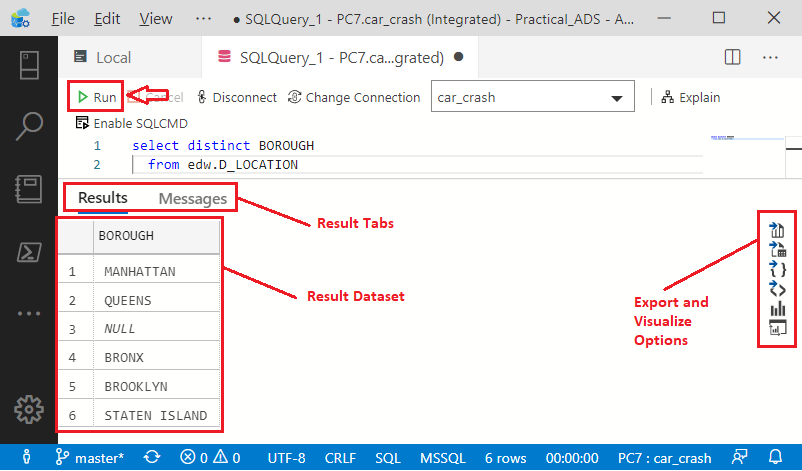


Figure 4-7. Results Pane from running a SQL Query

The above results pane provides the query results, which in this case lists all the boroughs contained in the D\_LOCATION table (which in Kimball terminology would be referred to as the ‘Location’ Dimension[[2]](#footnote-2)). Of the 6 rows returned, only 5 have real ‘boroughs’ (i.e., true values). The NULL represents the absence of any value, which is also considered an unknown location.

The right side of the results pane provides the following options (from top to bottom)

* Save as CSV
* Save as Excel
* Save as JSON
* Save as XML
* Chart
* Visualizer

The four ‘Save as’ (i.e., Export) options listed above can be used to move your dataset into other programs or workstreams. The last two options are used for charting or visualizing your results. If you do not see the ‘Visualizer’ option, you will want to install the ‘SandDance’ extension for ADS. This is a very nice ‘visualization’ addition and is available by clicking the ‘Extensions’ icon in the Activity Bar, and then the entering ‘SandDance’ in the ‘Search…’ text box at the top of the Side Bar.

There are separate contents in results pane which are accessible by clicking on Messages tab as displayed in Figure 4-8:

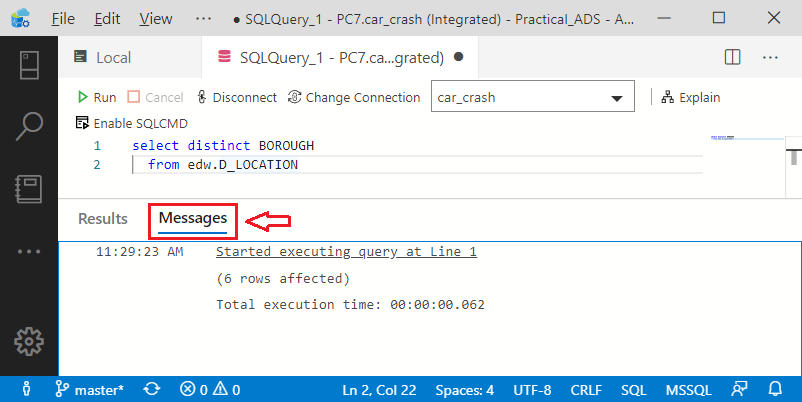


Figure 4-8. Messages Tab on the Results Pane

This pane provides runtime information (e.g., row counts, execution time) along with any other messages that the SQL script happens to generate, such as “runtime errors” or from embedded PRINT statements.

# Querying for Easy Visuals

We will now extend our query to count the total number of accidents recorded for each borough by using the following query:

select l.BOROUGH

     , count(\*) as ACCIDENTS

  from edw.D\_LOCATION l

  join edw.F\_COLLISIONS c

    on c.LOCATION\_ID = l.LOCATION\_ID

 where l.BOROUGH is not null

 group by l.BOROUGH

Unlike our first query, we are introducing a measure with the name of ‘ACCIDENTS’. Without a ‘measure’ in your result-set, it is difficult to directly create a meaningful visual since you may only have a list of ‘text’ values. For our purposes, we will define a measure as “a numerical value calculated with an aggregate function”. In this case we are aggregating using ‘count(\*)’ which will add ‘1’ for each row found. The columnar results of this query are shown in Figure 4-9. Note the ‘Chart’ icon (boxed in red) will be our next step:

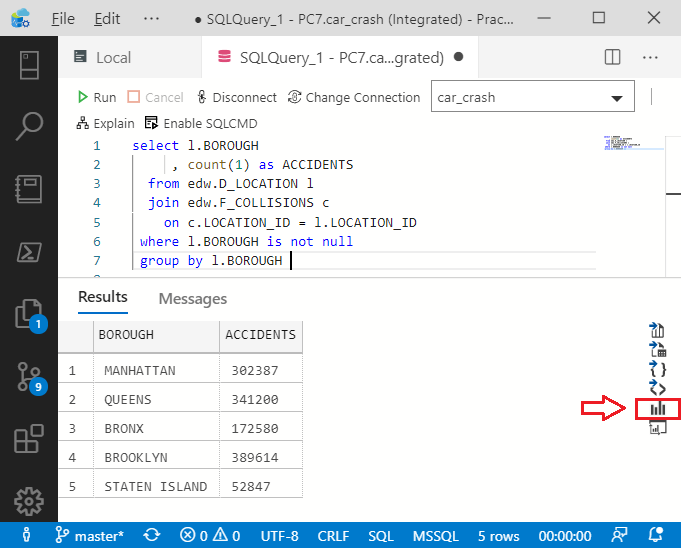


Figure 4-9. Results Pane primed for Creating a Chart

Go ahead and click on the ‘Chart’ icon and you will get an almost presentable chart as displayed in Figure 4-10:

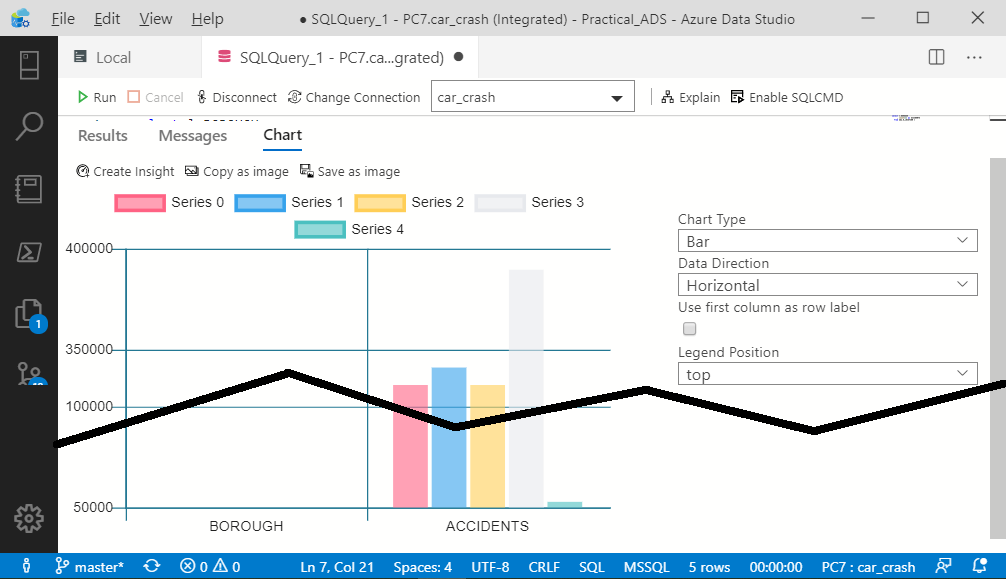


Figure 4-10. Bar Chart in Results Pane

To quickly clean-up this visual, click on “Use first column as row label” as shown in figure 4-11:

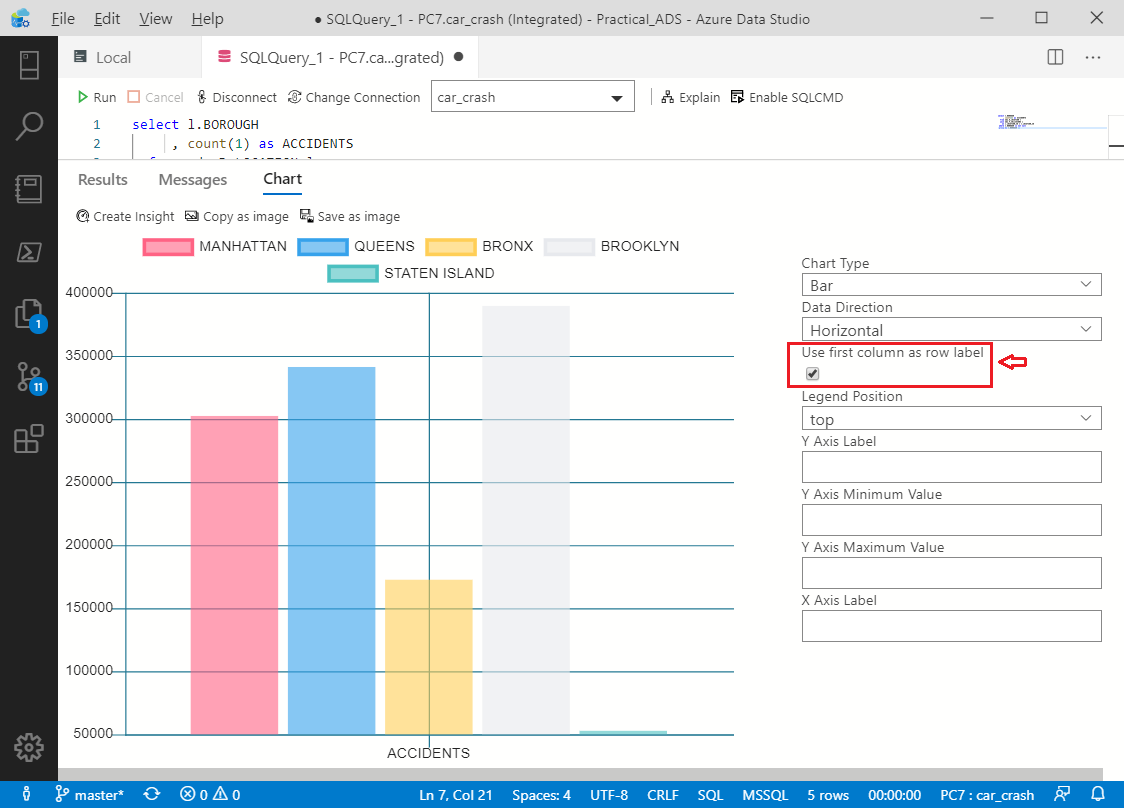


Figure 4-11. Improved Bar Chart in Results Pane

You now have a bar chart, produced in seconds from your current result set, and without ever leaving Azure Data Studio. With this visual, you can now copy/paste into an email, save it as an image, or even create an ADS ‘insight’ (Dashboard insights are covered in Chapter 10).

There are of course other chart options that you can play with, which are shown on the right side of Figure 4-11. ADS charts are also interactive, allowing you to dynamically filter by clicking on a chart series. Next, we will create visualizations using the ‘SandDance’ extension.

# Getting Pixel Perfect

For more control of your visual creation, you can try using the SandDance visualizer. Since we will gain an additional axis using SandDance, we will add the MONTH\_NAME to our query by joining to the DATE dimension. The updated query is as follows:

select l.BOROUGH

     , d.MONTH\_NAME

     , count(\*) as ACCIDENTS

  from edw.D\_LOCATION l

  join edw.F\_COLLISIONS c

    on c.LOCATION\_ID = l.LOCATION\_ID

  join edw.D\_DATE d

    on d.DATE\_KEY = c.DATE\_KEY

 where l.BOROUGH is not null

 group by l.BOROUGH

        , d.MONTH\_NAME

Update your query in ADS using the above SQL script, and then click ‘Run’. Your results pane will now have an additional column as shown in Figure 4-12. Notice the ‘Visualizer’ icon (boxed in red) which will be our next step:

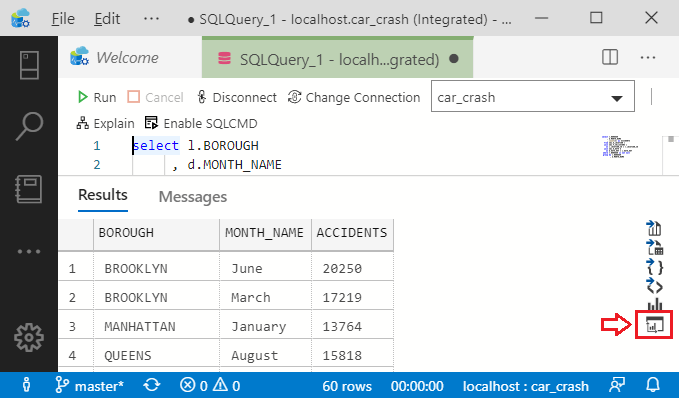


Figure 4-12. Adding ‘MONTH\_NAME’ to Query Results

Next click on the ‘Visualizer’ icon outlined in red above. A new main tab will open in ADS showing the SandDance designer as displayed in Figure 4-13. Note the cube icon in the top right of designer:

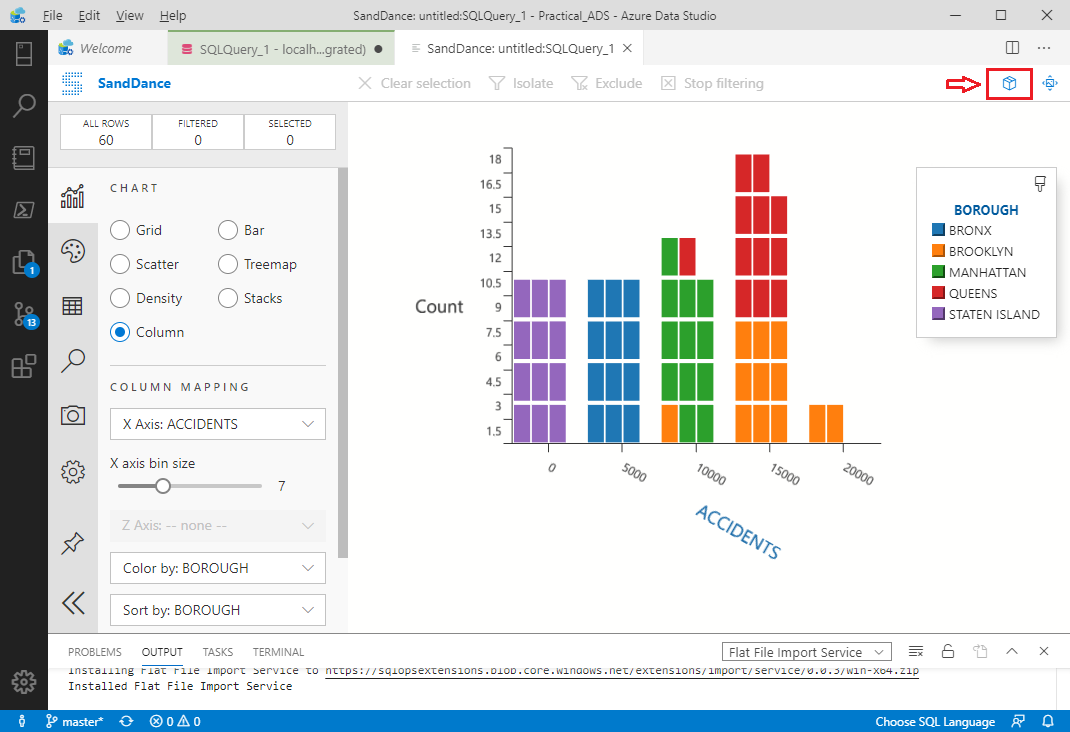


Figure 4-13. SandDance (Extension) Chart

Next, change the chart rendering to ‘3D’ by clicking on the cube icon shown above. This provides a ‘Z Axis’ which we will set to ACCIDENTS. To reproduce the SandDance visual shown below in Figure 4-14, set the ‘X Axis’ to BOROUGH, ‘Color by’ MONTH\_NAME, and ‘Sort by: ACCIDENTS.

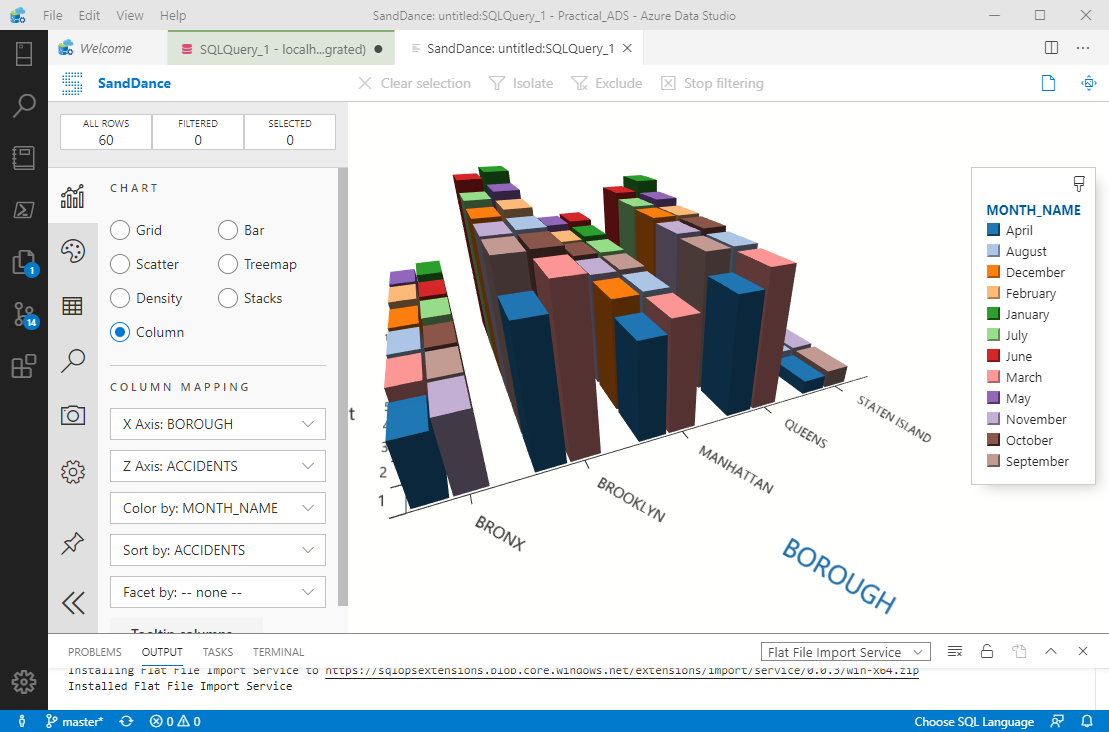


Figure 4-13. SandDance with Three Dimensional Visual

This visualization reveals that Brooklyn edges out Queens for having the greatest number of car accidents (from July 2012 through Feb 2020). Another thing disclosed above is the worst month for car accidents in NYC is January, probably due to snow and ice. On the other hand, the safest driving month for all boroughs is April. Besides being a slightly shorter month (30 days), there must be something else about the month of ‘April’ to account for a significant dip in car accidents.

Perhaps we will find out more? as we delve deeper into our car-crash DataMart in the next chapter, which is on my favorite ADS feature: Juypter ‘Notebooks’.

1. <https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95/data> [↑](#footnote-ref-1)
2. <https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/> [↑](#footnote-ref-2)