

Lab 4: Watson Studio Visualizations

In this lab we will cover several data visualization capabilities provided by **IBM Watson Studio**:

- Two **IBM Watson Studio** programatic visualization add-ons on top of the regular support provided by Jupyter Notebooks and its cohort of visualization librairies (`matplotlib` , `seaborn` , `bokeh` , `ggplot` , ...):
 - **Brunel**
 - **PixieDust**
- The **IBM Watson Studio Dashboards** service, with a UI-driven capability to build and publish dashboards largely inspired by **IBM Cognos Analytics** capabilities.

Brunel Visualizations

Brunel is an IBM-contributed open source tool designed to generate powerful visualizations with no 'plumbing' or boilerplate code.

Brunel will allow to build complex displays from `Pandas DataFrames` with only a single declarative line describing the display.

Brunel has been integrated as an add-on to Jupyter notebooks and is included in the Watson Studio environment setup.

If you are interested of understanding more about Brunel, feel free to explore the following resources:

- Brunel GitHub repo: <https://github.com/Brunel-Visualization/Brunel>
- Brunel documentation: <https://brunel.mybluemix.net/docs/>
- Brunel Visualization Tech Talks: <https://developer.ibm.com/code/videos/brunel-visualization-tech-talk/>

As we have seen during Lab1, brunel is also used within Watson Studio to quickly visualize data assets loaded within a project.

The hands-on will be driven by the `Visualize_car_data_with_Brunel_cleared.ipynb` notebook, using data from the `cars.csv` file.

Lab Setup

- From your `WatStud_Workshop` project's `Assets` tab, add the `cars.csv` file as Data Asset if not already part of your `Assets`
- Create a new Notebook from the file `Visualize_car_data_with_Brunel.ipynb`, with a `Default Python 3.5 Free` runtime environment as in the first lab:



- Follow the instructions within the notebook and come back to this material once completed.

PixieDust Visualizations

What is PixieDust

PixieDust is an IBM-initiated open source framework for interactively exploring data sets within Jupyter Notebooks.

Lab introduction

The notebook used in this hands-on lab makes use of **PixieDust** capabilities to handle a Spark dataset, such as direct downloading of dataset from source URL, and interactive visualization.

The notebook is about analyzing traffic incidents from the city of San Francisco, based on the Open Data set provided by the authorities.

- Create a new notebook from file, selecting `pixiedust-traffic-analysis_pandas.ipynb` as source, and selecting the same runtime as in the previous notebooks, e.g. `Default Python 3.5 Free`.
- Follow the instructions in the notebook. You will notice that the notebook uses the **PixieDust** display widget to guide you through interaction with the data, and therefore there is not a lot of actual code used in this notebook.



Watson Studio Dashboard

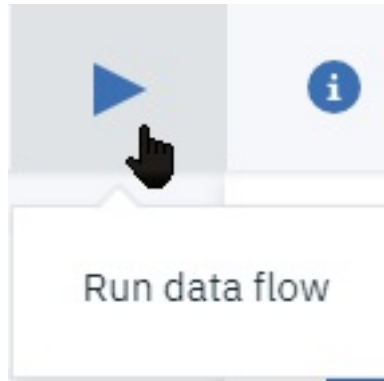
IBM Watson Studio has a built-in capability to build interactive publishable dashboard.

Converting to CSV format

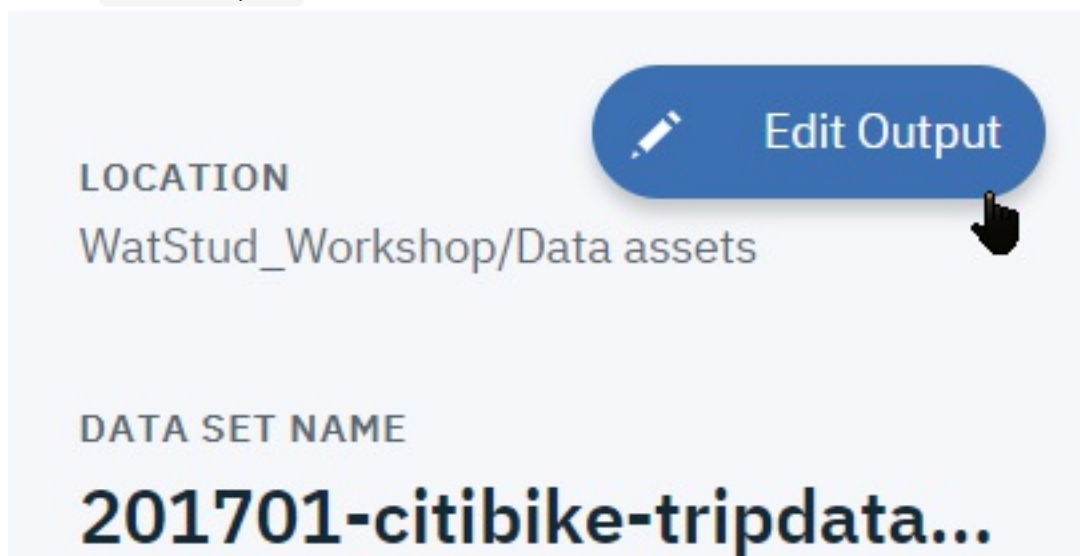
We need to convert back the parquet file to CSV for use by the Dashboarding service:

1. From the Data Assets list, select Refine for the `201701-citibike-tripdata_cleansed.parquet` file:

<input type="checkbox"/>		201701-citibike-tripdata_cleansed.parquet	Data Asset	Project	Workshop September	24 Sep 2018, 11:47:30 am	
		GoSales_Tx.csv	Data Asset	Project	Workshop September	17 Sep 2018, 6:02:07	Refine



2. Click on the Run icon
3. Select `(Edit output)` button:



4. Change the file name to `201701-citibike-tripdata_cleansed.csv` , and make sure format is CSV:

Edit output

DATA SET NAME *

01701-citibike-tripdata_cleansed.csv

63

DESCRIPTION

Enter a description of the resulting data set.

300

FILE FORMAT

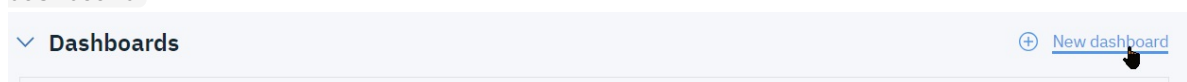
CSV

☒ The first line of the file contains column headers

5. Validate changes and click [Save and Run]
6. Click on [View Flow] and wait for conversion to complete.

Setting up a dashboard

1. Back to your project's Assets list, navigate to the Dashboards section and select (+)New dashboard :



2. Enter a name, e.g. NYC Bike Rentals
3. We will need to create a dashboard service instance, select the Associate a Cognos Dashboard Embedded service instance link

Associate a Dynamic Dashboard Embedded service instance

No Dynamic Dashboard Embedded service instances associated with your project.

[Associate a Dynamic Dashboard Embedded service instance](#) with your project on the project settings page, then

4. Choose the 'Lite' configuration, and confirm default org and space
5. Click the **Reload** link and select the instance, then the **Save** button:

New Dashboard

Name*
NYC Bike Rentals

84

Description

Type your description here

300

Dynamic Dashboard Embedded Service
dynamic-dashboard-embedded-watstud

Cancel

Save

6. In the **Select a template**, use **Tabbed** and **Freeform**:

Select a template

Dashboard

Single page

Tabbed

Tabbed layout

Freeform

, then **[OK]** button.

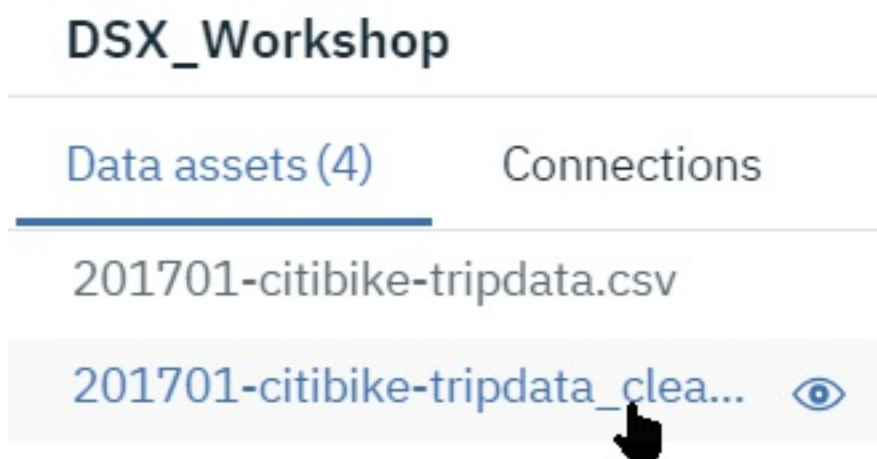
Adding data to a dashboard

We will now use the data produced by Data Refinery for the NYC bike share dataset.

1. Switch to the `Select` tab and expand `Selected sources`

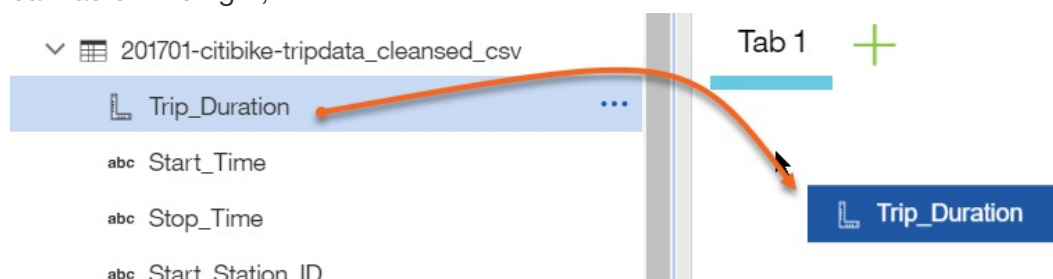


2. Select the `201701-citibike-tripdata_cleansed.csv` file:

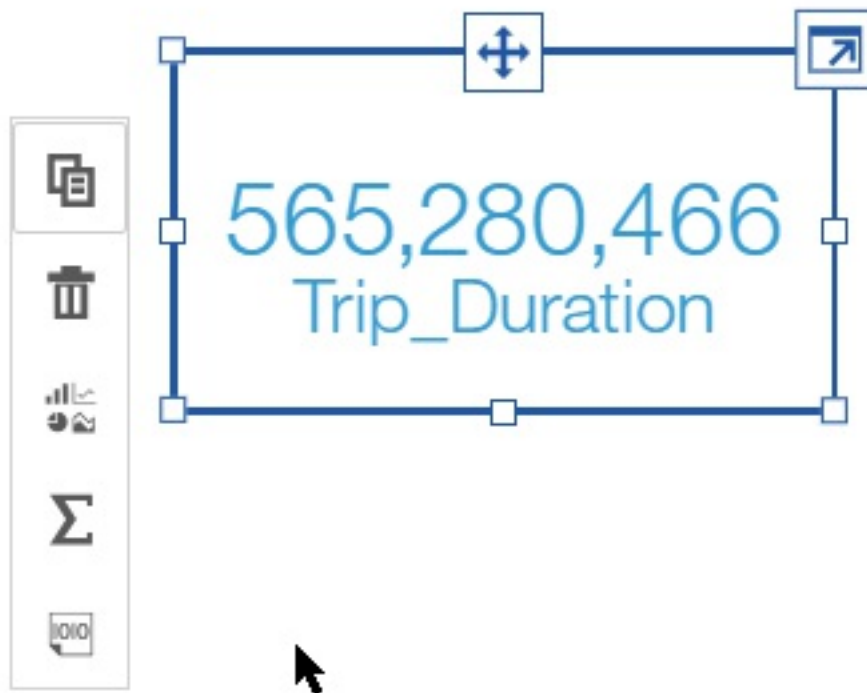


3. The dashboarding has the ability to propose a graph type based on the data. We will start by displaying the `Trip Duration` by `Age` :

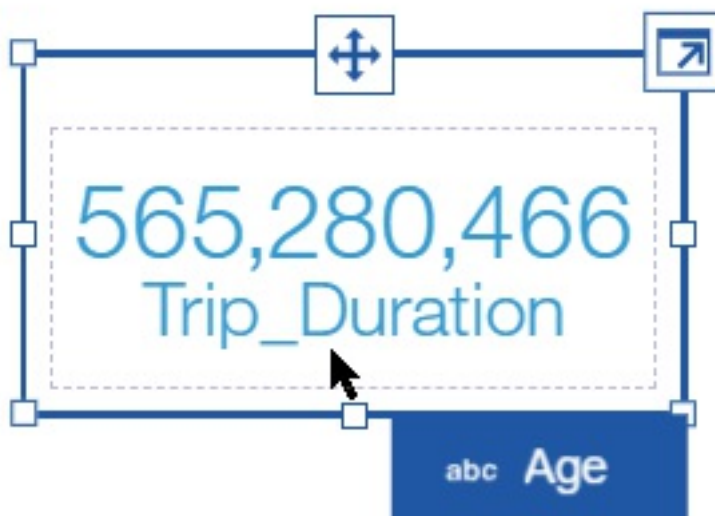
- i. Drag&Drop the `Trip_Duration` from the data panel on the left to the dashboard canvas on the right;



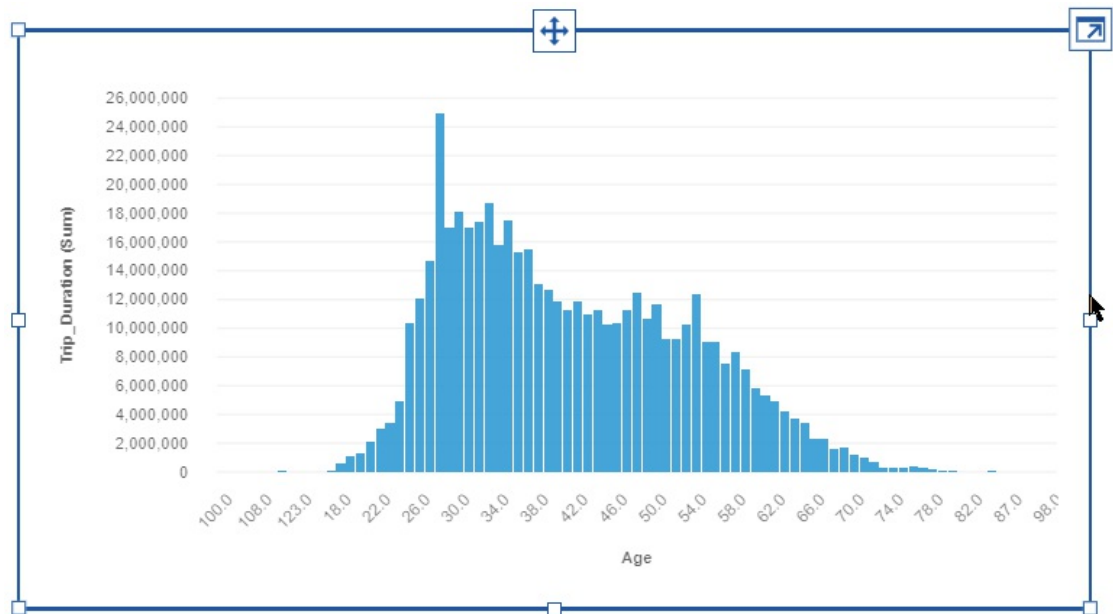
The `Trip_Duration` total aggregated sum is displayed as a big number:



ii. Drop the `Age` field onto the `Trip_Duration` widget:



iii. **IBM Watson Studio** changes the graph to a more suitable representation, in this case a bar graph:



- iv. Unfortunately, our data has not been cleansed enough and we have erroneous values for `Age`. Right-click on the `Age` label to display the menu, and select the filter icon



- v. In the filter definition box, select all values which do not make sense (no value, values above 100):

Age

×

Add a filter condition

>

🔍 Search

☒ (no value)

☒ 100.0

☒ 102.0

☒ 105.0

☒ 107.0

☒ 116.0

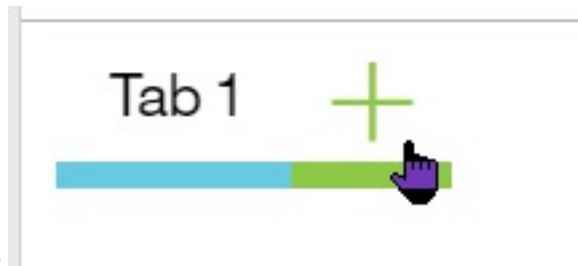
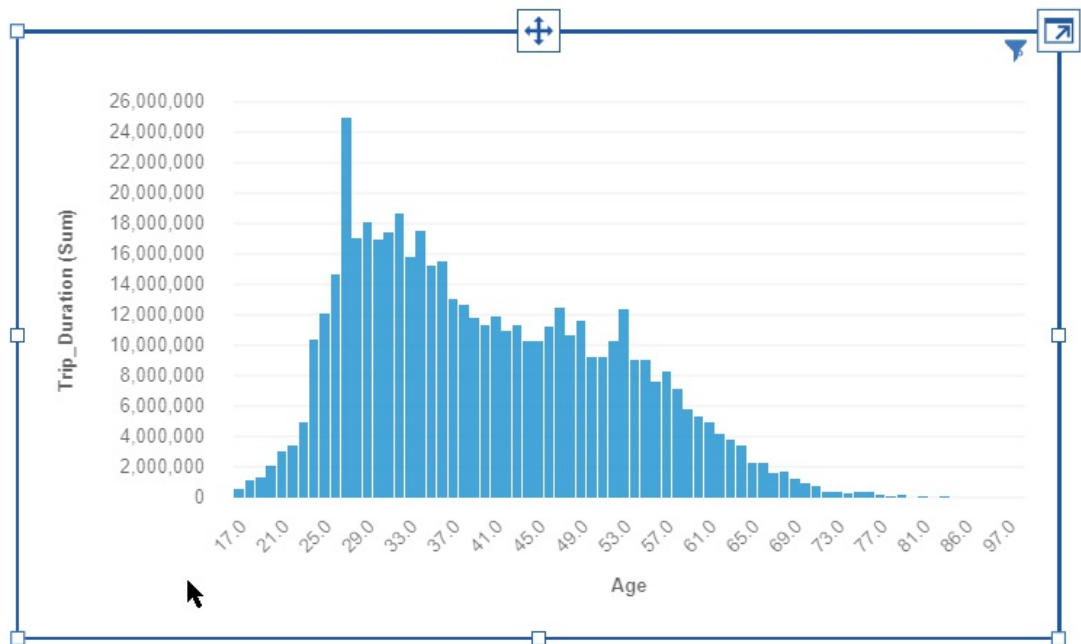
☒ 117.0

☒ 118.0

Clear all

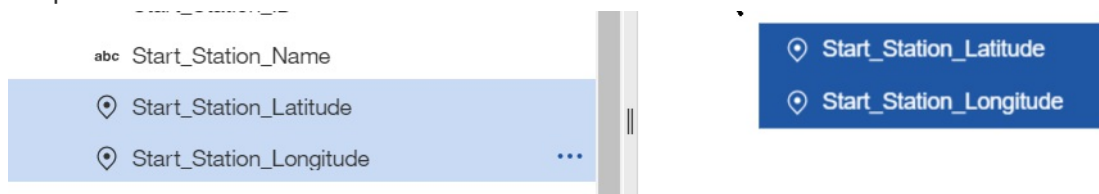
Invert

, then click the `Invert` button and OK. We get a better-looking graph where we can see the trip duration distribution by age



4. Now add a new Freeform tab and we will create a map display of the stations by count of rentals:

- i. Select the two `Start_Station_Latitude` and `Start_Station_Longitude` fields and drop them on the canvas:



- ii. The system automatically creates a map display:



- iii. Unfortunately, there is some parasitic data with erroneous coordinates that show up in the middle of the ocean at coordinate (0,0) below the African continent. Select the **Expand** button at the top right of the widget:



- iv. Expand the **Start_Station_Latitude** :

▼ **Latitude/longitude**



Start_Station_Latitude

...



Start_Station_Longitude

...

abc

Label



Point size



Point color

then select filter

▼ **Latitude/longitude**

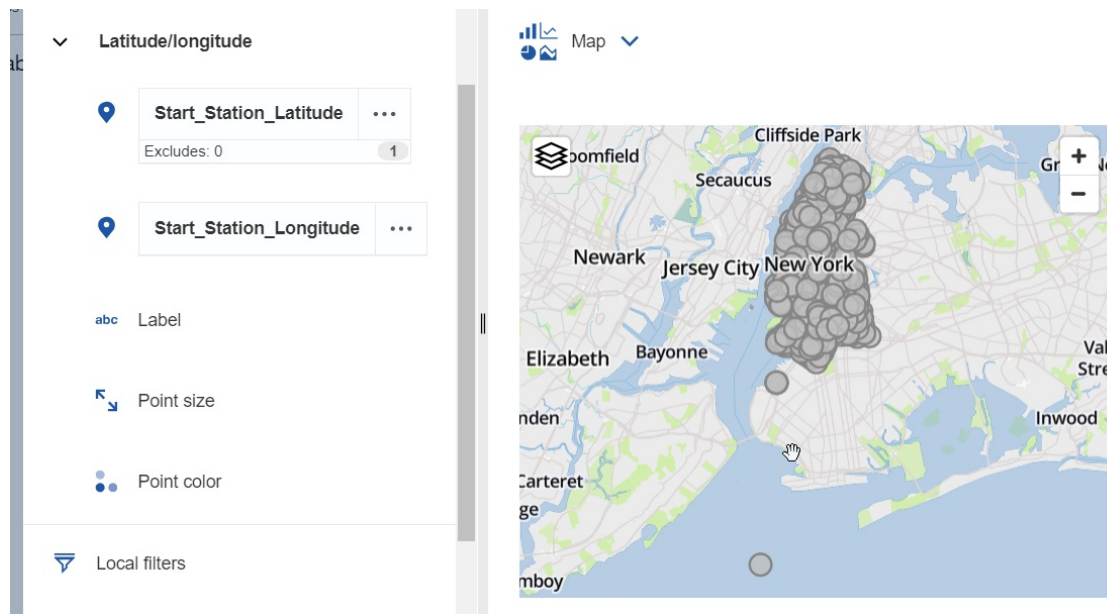


Start_Station_Latitude

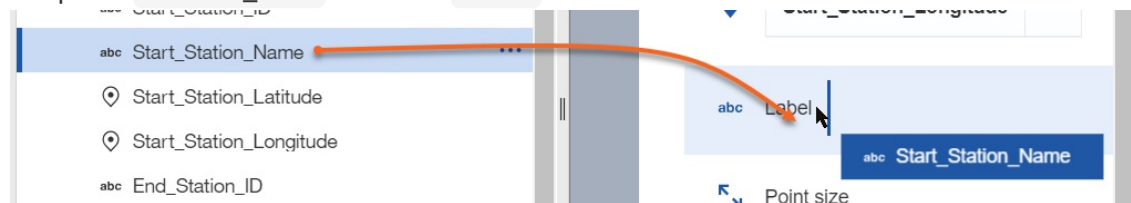
...

Filter

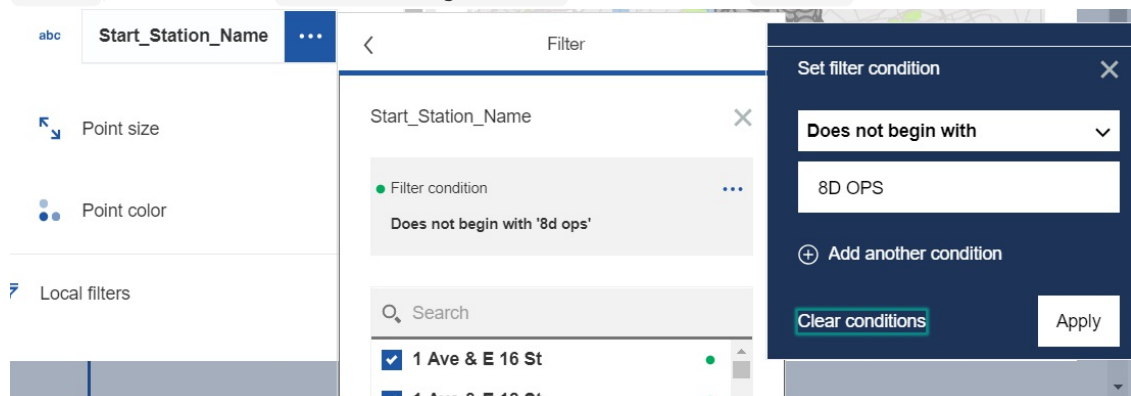
- v. In the filter definition, select the first 0 value, then **Invert** and OK button. The map will center itself on NYC:



vi. Drop the `Station_Name` onto the `Label` :

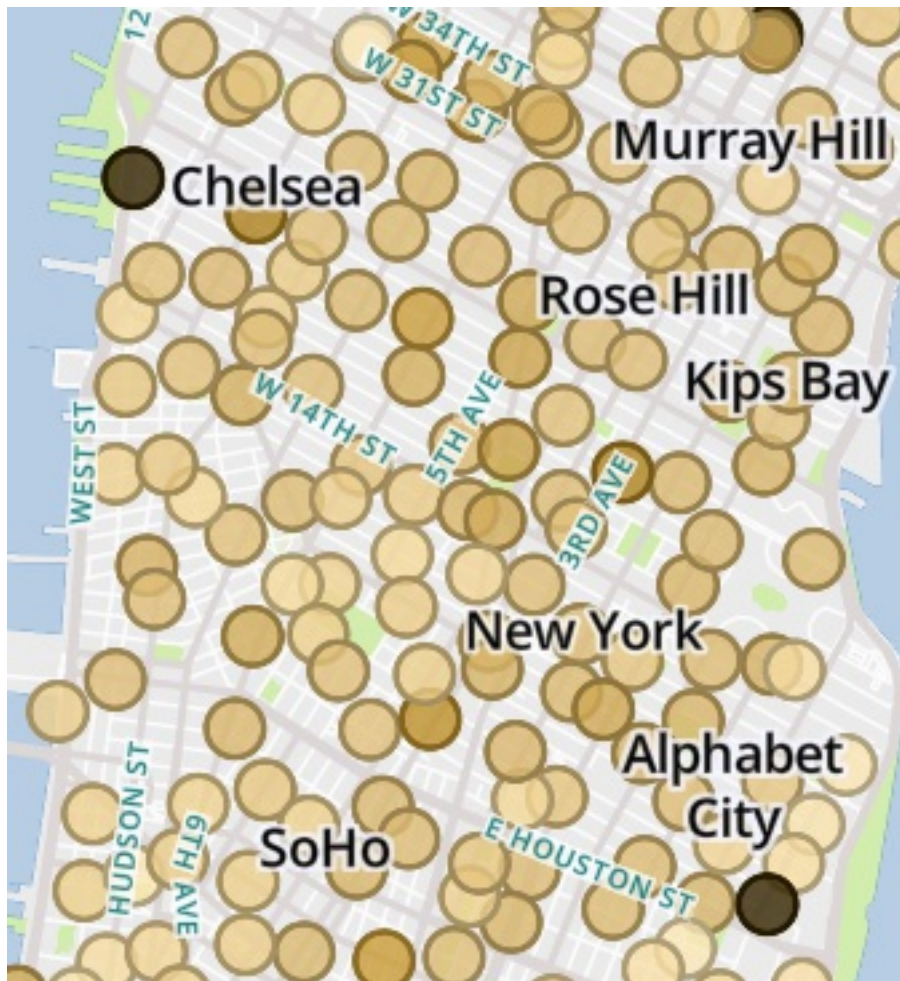


vii. We see on the map an outlier, south on the ocean, we can filter it out by name, as we can get the `8D 0PS 01` label now by hovering over it. Click the Filter button for `Label` , and enter a `Does not begin with` condition for `8D OPS` :

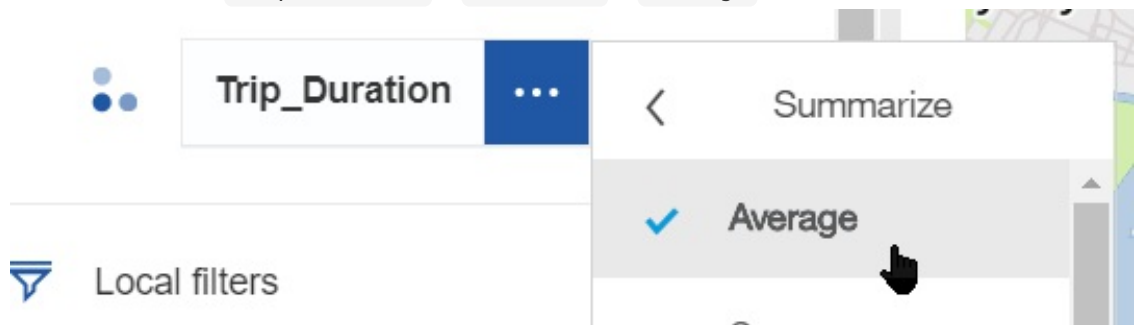


. The outlier point will disappear from the display.

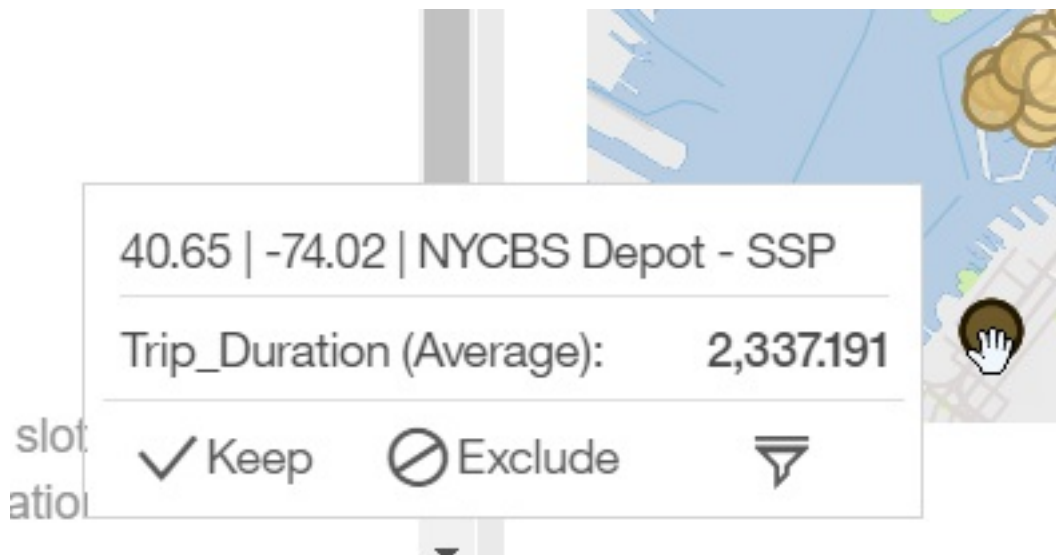
viii. Drop the `Trip_Duration` field onto the `Point color` . The default aggregation is `SUM` which will show stations from where the cumulative trip are longer. This shows that a few stations are issuing longer rides than others, as they show in darker colors:



- ix. Change the aggregation used for the coloring, now based on the average trip duration. Select `Trip_Duration` -> Summarize -> Average :



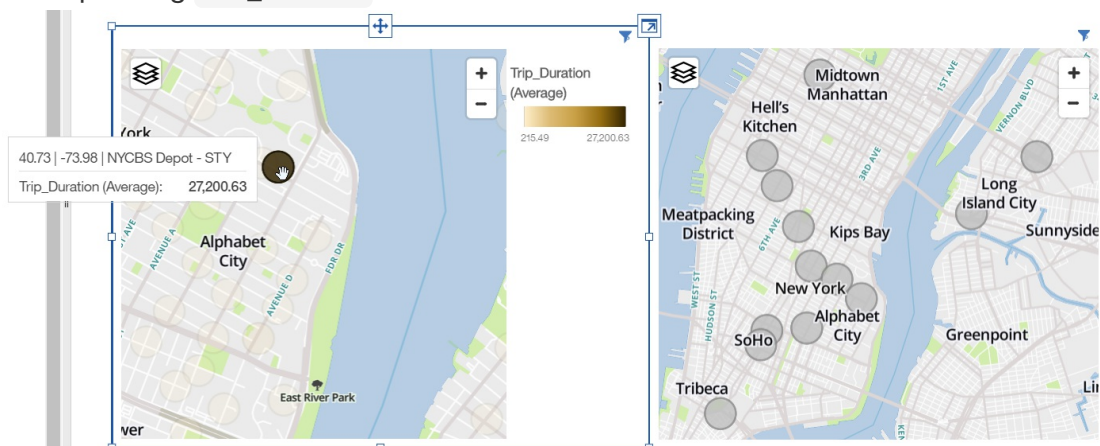
- x. All points now look similar. You can remove outlier manually by right-click selecting them on the map and selecting `exclude` :



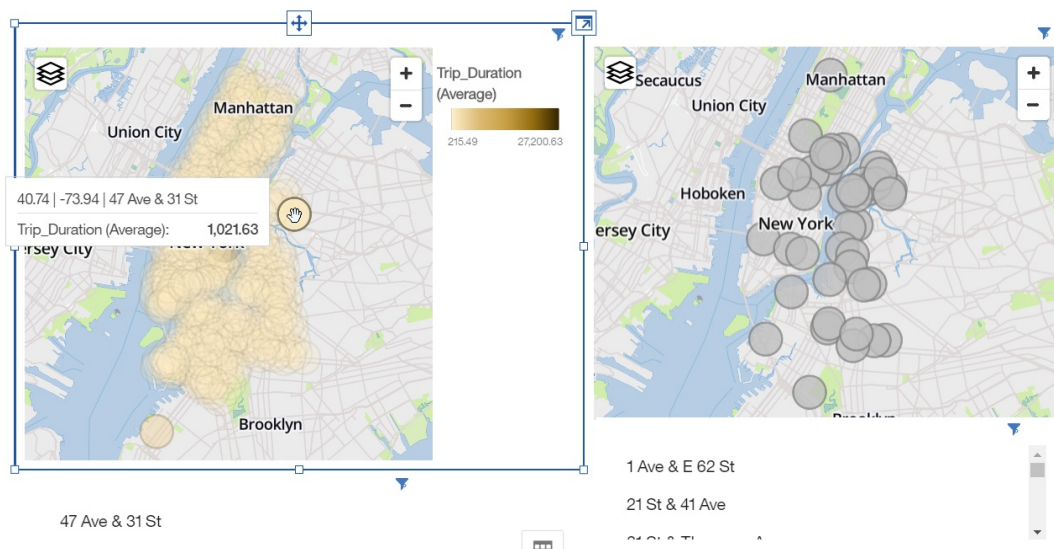
5. Correlated graphs selections (Widget connections)

- Drop the `End_Station_Latitude/Longitude` on the freespace besides the `Start_Station` map to create a new map.

Now, when clicking on a Station in the first map, the second map adjusts to show the corresponding `End_Station` :



- The same applies if you drop a label field, for example `Start_Station_Name` and `End_Station_Name`, the fields will reflect the currently selected data points subsets:



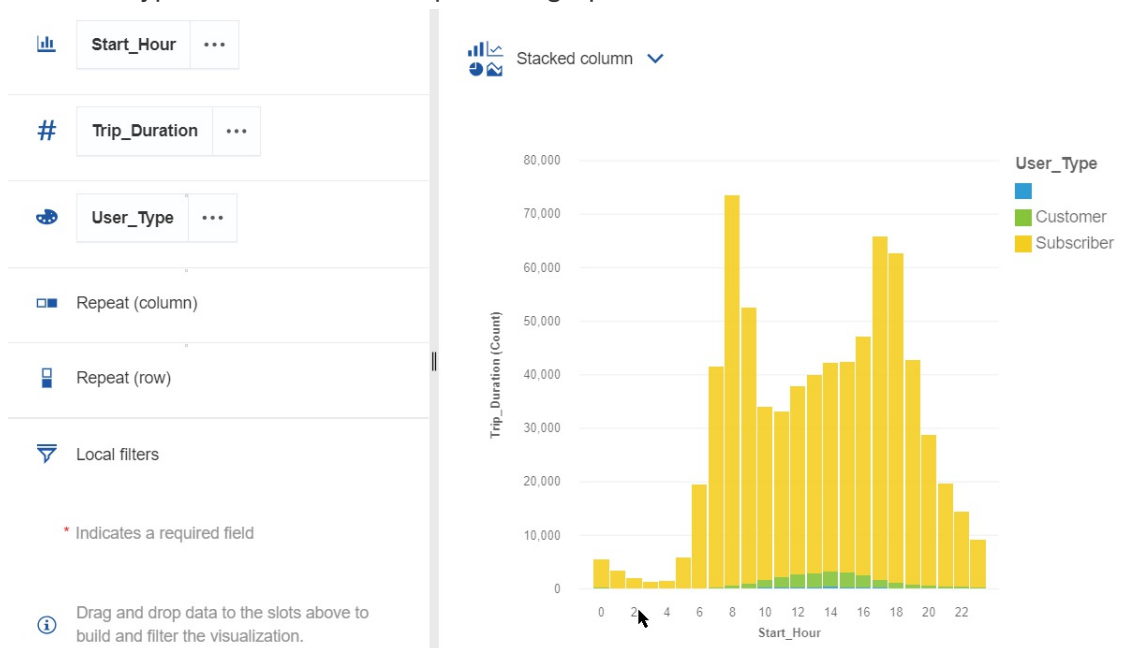
iii. Note that the selection groups can be adjusted using the **Widget connections** icon



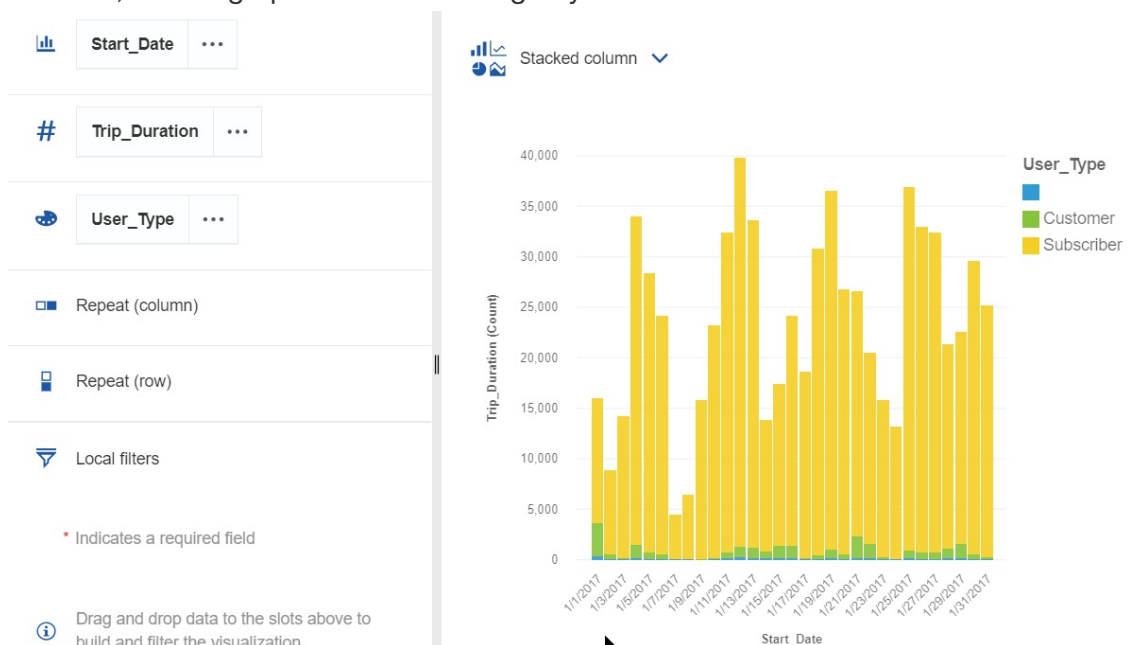
at the top

6. Many other types of graphs can be built, as an exercise, build:

- i. a graph on another tab that will show the distribution of rentals by the hour of the day and user type. You should end up with a graph such as:

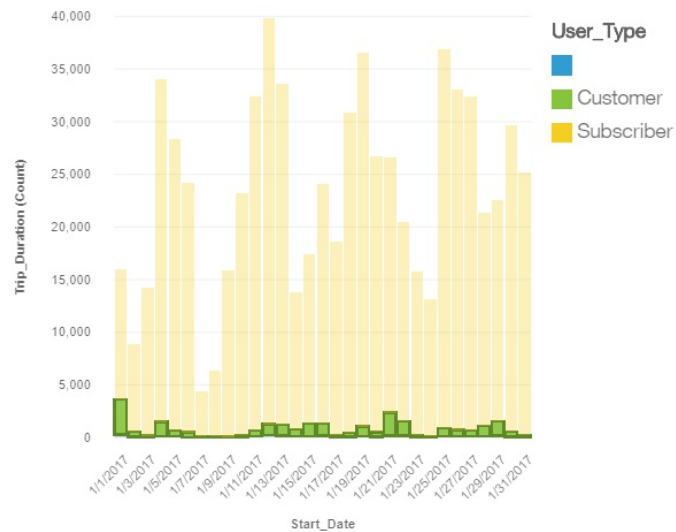
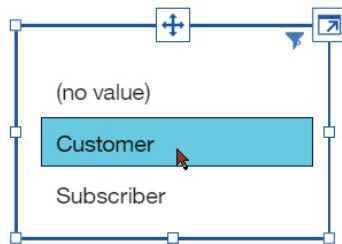


- ii. And last, build a graph that shows usage by date:

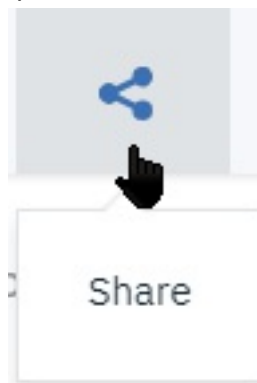


. We see the weekly cycle, and probably the impact of weather conditions.

7. Graph elements can also act on others on the same tab. On the last tab you created, drop a `User_Type` field. This will drive the behavior of the usage by date graph:



8. Finally, dashboards can be published:



- Click the `Share` icon:
- Enable sharing:

☒ Share with anyone who has the link.

i The link always points to the most recent version of the dashboard.

Permalink to view dashboard

<https://eu-gb.dataplatform.ibm.com/dashboards/1701e5a9-0770-4c>



- Open the link from another tab or browser to get a web view on the dashboard