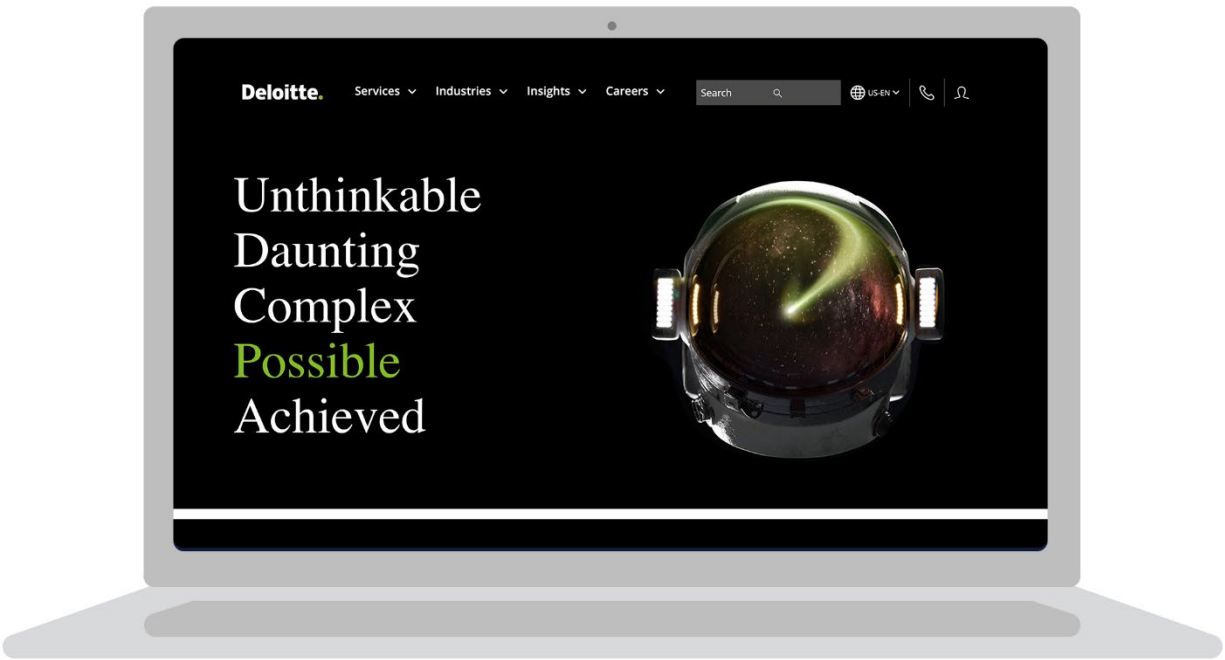
Case study: Deloitte

You've learned the definitions for each of the six practices of exploratory data analysis (EDA) — discovering, structuring, cleaning, joining, validating, and presenting. Next, you'll discover how these practices apply to the data career space. In the following case study, members of Deloitte's data team utilized the six practices of EDA to meet their client's metrics and dashboarding needs. After you read more about the experiences of Deloitte's team, you will have a tangible example of just how useful and empowering the application of these practices can be.

Get to know Deloitte

Deloitte is an audit, consulting, tax, and advisory service. The company has over 100,000 employees and partners around the globe, and they work with many of the world's biggest companies. Their services range from tax and accounting solutions to artificial intelligence and cyber security risk programs.

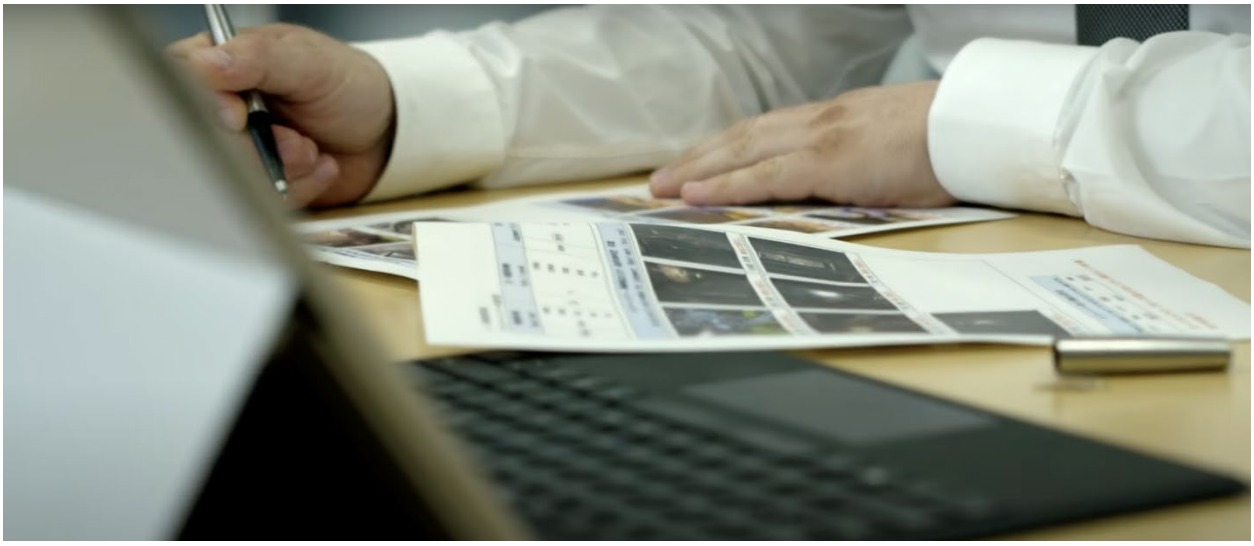


The multinational corporation's business to business (B2B) model keeps their focus on working with their customers on a number of strategic and financial fronts, according to their individual needs. Deloitte provides a range of services for client projects, such as auditing, consulting, financial advisory, risk advisory, taxes, data analysis, and regulation. Since 2008, Deloitte has made it their vision to be the standard of excellence in

professional business services. Because of this focus, Deloitte has remained at the forefront of technology advancements, including the deployment of real-time metrics for their clients.

In just one of their many success stories, Deloitte recently helped a billion-dollar cloud-based analytics and software company streamline the tracking and organization of their marketing leads and strategic analysis of their performance data. In this reading, you will see how Deloitte used EDA to analyze the client's data in order to propose and apply a comprehensive solution they still use today.

The challenges



One of Deloitte's most important cloud-based software solution clients was facing a variety of challenges with their marketing and performance data. They asked Deloitte for help on all of them. Here are the specific challenges the client presented to Deloitte:

- Difficulty in following up on marketing leads
- Struggling to promptly track their marketing campaign performance
- A lack of personalized data dashboards displaying strategic company metrics

Following up on marketing leads

Deloitte's client offers a range of cloud-based analysis services and software products to businesses around the globe, making it a monumental task to follow up on each of the thousands of inquiries and potential sales leads they receive on a monthly basis. As a result, Deloitte's client was seeking a solution to help them more adequately track and act on these leads.

Tracking marketing campaign performance

Relying on just four data analysts for analysis of marketing campaign performance for a $1B multinational corporation was quickly becoming unsustainable. Before Deloitte stepped in to help, the client's performance tracking system was a series of data tables updated manually and kept in a spreadsheet tab. Each region kept their own performance data, creating issues of consistency and hours of wasted energy for the analysts who had to compile and merge the data to get a corporation-wide performance picture.

Personalizing data dashboards

Because of differing needs for each level of management and varying marketing regions across the globe, the client needed to be able to easily parse and group data specific to their employees' needs across the corporation. Because they were working off of multiple spreadsheets across several different departments and locales, the client found they could not easily filter the data according to their needs, and therefore were not able to adequately shift strategy to improve performance.

The approach

As Deloitte began working with their new client, they designed an approach that would meet each of their client's proposed needs. Refer to the following PACE entries to review this approach in detail.

Plan

To begin, Deloitte needed to meet with client stakeholders and help develop their vision for the future according to the three areas mentioned above: **Marketing leads, measuring marketing campaign performance,** and **personalizing data dashboards.**

They needed to understand the client's business by learning its processes, objectives and key results (OKRs), and how it approached and used its customer and potential customer data.

Key project milestones were established early on, and adjusted according to any new knowledge acquired. Those milestones were:

- Alignment with client on the future vision of data architecture and dashboards
- Establishment of the new data architecture
- Launch of descriptive and diagnostic tooling

- Launch of predictive models
- Training the global sales team on new tooling

Deloitte determined that their project plan would begin with the discovering, structuring, and cleaning practices of EDA on the client's company-wide performance data.

Deloitte quickly learned what OKRs the client wanted easy access to once the personalized dashboards were complete and set plans to achieve that outcome. Those OKRs came in the form of the following questions:

- How much revenue was earned from a particular marketing campaign?
- What industry, region, sector, and company size did the new customer fall into?
- How long did it take to finalize the sale after the initial point of contact?
- What was the success rate of different types of campaigns (i.e., in person events vs. online webinars)?
- What was the average cost of marketing per customer gained?

Analyze

As part of the framing of their initial analysis of the client's data, it became clear the client was unfamiliar with Deloitte's product services offerings, including assets like easy-to-use Tableau dashboard tooling. Because of this, the client's assumptions and requests for what was possible fell far below Deloitte's actual capability. This represented an opportunity to not only meet, but actually exceed the client's expectations of the project.

The first challenges of analysis were in understanding where the data was coming from and what each data variable meant. The client used non-standard naming conventions and definitions in their data, so Deloitte had to use discovering practices to learn their language to fully understand the client's data and plan for how to help them use it.

Deloitte found that the client's approach to tracking performance at the time was clunky and overbroad at best. The performance data they gathered consisted of the use of overall averages to determine value of strategic company-wide actions. Without specific performance data per region, the company had no simple way of tracking regional performance.

- As an example of the client's overbroad data tracking at the time, this was the formula they used to derive the average revenue they gained per marketing campaign:

$$\frac{average\ customer\ contract\ value}{average\ cost\ of\ marketing\ campaigns} = average\ revenue\ gained\ per\ campaign$$

Deloitte learned that the client's internal marketing and sales teams were organized according to geographic regions, industries, and customer account size. Regional leads had the authority and autonomy to conduct marketing research and campaigns on their own, but did take guidance and occasional instruction from the global team. Deloitte's analysts began structuring the data in a consistent way company wide as part of their EDA of the client's data.

In terms of customer life cycle, Deloitte learned that the client's marketing leads would generate marketing campaigns and then inform sales partners in their regions. Field sales representatives would follow up on all leads generated from the campaigns, and they would also network outside of existing campaigns to find additional customers. This discovering part of EDA informed Deloitte's team on more specific and plausible solutions that could meet the client's needs.

The Deloitte team used discovering and structuring practices of EDA in spreadsheets to review all of the existing data from the client's Salesforce and marketing campaigns. They began to identify and define every data variable that existed in their client's sales system. Once an inventory of data variables was structured, cleaned, and validated, the Deloitte team brainstormed the type of insights they could glean from the information.

After the review, Deloitte utilized the presenting practice of EDA to share the information they had gathered with the client. They listened and asked questions to get a better understanding of how the client viewed their own business and what factors were important to them according to the data.

Construct

From their analysis of the client's systems and the discovering, structuring, and cleaning practices of EDA they did on the client's performance data, Deloitte learned that the client needed a total overhaul of their analytical process; one that provided much greater detail and allowed them to identify and address bottlenecks, unprofitable campaigns, and unproductive customer segments.

With the client's approval, Deloitte completely rebuilt the client's data internal infrastructure from the ground up to make analysis more streamlined.

On top of the infrastructure overhaul, the most hands-on solution for the client was to build customized and dynamic dashboards for specific stakeholders.

- The Deloitte team worked with each client stakeholder group to understand their needs and their OKRs.
- They then helped develop tailored solutions for the client by creating supporting collateral resources, such as a data dictionary to define what each variable represents and how it should and should not be used.

Deloitte used the information they gathered from their client's data to build automations and tools that performed predictive analysis on their OKRs. This allowed the client to establish feasible, but aggressive targets for the future.

Execute

Deloitte's overhaul of their client's data infrastructure led to a streamlined, tiered approach to global data gathering. Where before the client was forced to rely on regionally gathered and inconsistently structured spreadsheets to formulate their performance metrics, they could now easily quantify everything from global, company-wide performance down to the performance of a regional team of sales team members on specific products and subset of clients.

After implementation of personalized dashboards, Senior VPs could filter the status and performance of the OKRs to the whole departments and regions for which they were responsible. Additionally, regional leads, marketing leads, and even local sales leads were able to view data filtered down to their applicable area.

As part of implementation, Deloitte team members created training sessions according to their client's needs across the organization—everything from how to use the personalized dashboards to how to maintain the new data infrastructure.

The Deloitte-designed dashboards and data infrastructure instantly became valuable assets to multiple stakeholders across the client's global team.

Note: In their work with the client,Deloitte was not following the PACE workflow as it is listed above; rather, while documenting the details of their work into this case study, Deloitte's work for the client has been organized according to the PACE workflow to demonstrate the versatility of its application.

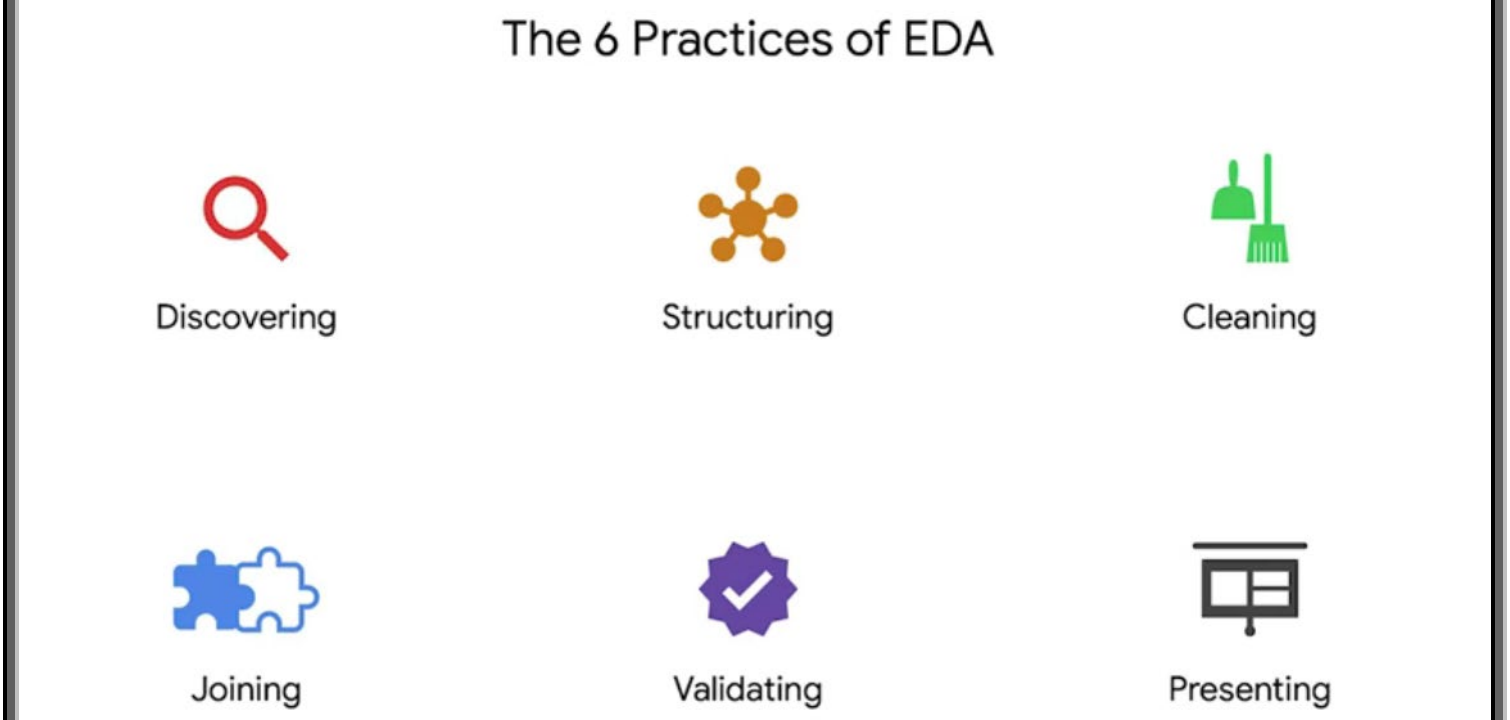
Summarizing the results

When Deloitte started working with their new client, they were hesitant to trust the accuracy of the company's existing data because of a lack of standardization between the marketing and sales teams. That lack of consistency made it difficult to associate any particular sale to a specific campaign. After a few weeks of studying the client's sales and marketing processes and EDA of their data, Deloitte rebuilt the data infrastructure in a way that directly tied all sales to their applicable marketing campaigns. They tied these sales to marketing campaigns using primary keys aligned across a number of data tables that could be easily sorted. This gave the Deloitte analysts an idea of what was possible with the data and led them to building final products that exceeded what the client thought was initially possible.

Once the infrastructure was overhauled, Deloitte's performance of EDA uncovered that the client was focusing too much effort on earning new clients, to the detriment of existing client relationships. While sales to new clients did increase because of the focus of OKRs, sales to their existing customers plummeted. This result was previously unknown to the client. As a result, the client was able to pivot their strategy by both sales region and product. The dynamic dashboards Deloitte created became the client's most effective tool for measuring and tracking performance as well as setting strategy.

Conclusion

Because of Deloitte's focus on the client's business problem in both their assessment of the client's business practices and EDA of their data, they were able to uncover truths previously hidden in unstructured data. Deloitte built interactive, dynamic data dashboards designed to filter down to any sales region or product in the company so that the client could discover these truths on their own in the future. The greatest benefit of these solutions was the client's leadership and marketing leads became a much more agile team. They were better able to make effective business decisions that directly improved revenue. The data infrastructure overhaul and data dashboard solutions also helped improve the work-life balance of their analysts. All of these outcomes were notable for not only the client but also Deloitte because the client was and is a key partner and account for the firm. Because of their focus on the data, Deloitte was able to offer a solution that exceeded the client's expectations and, in turn, helped them better serve their customers around the world.



The six practices of EDA are iterative and non-sequential

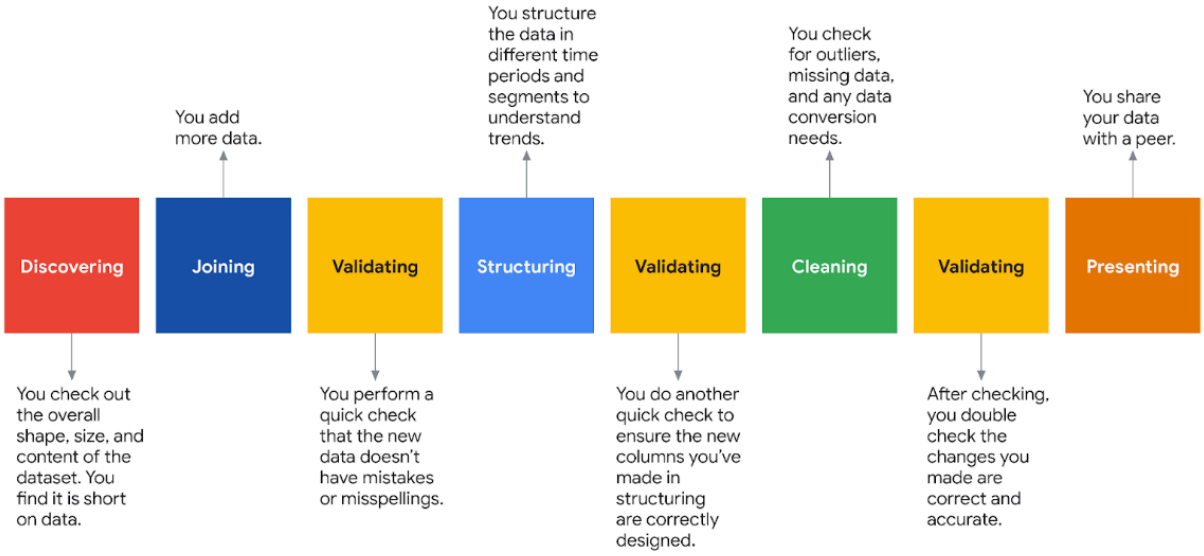
Exploratory data analysis (EDA) is not like a cake recipe. It is ***not*** a step-by-step process you follow. Instead, the six practices of EDA are iterative and non-sequential.

- **Iterative**: Relating to or involving repetition of a process
- **Non-sequential**: Not arranged in or following an order or sequence.

Because of the varying nature of datasets, the approach to exploring that data will be different each time. That means that you will need to use your logic and experience throughout the EDA process to determine which of the six practices to utilize, how many times to apply them, and when in the process you should apply them.

Visual example

Imagine you are assigned a dataset that has only 200 rows and five columns of data about trees in a coniferous forest in Norway. You know that to complete your full analysis you'll need more than 1,000 rows and at least two more columns. Even without much more detail than that, your entire EDA process might look something like this:



1. **Discovering**: You check out the overall shape, size, and content of the dataset. You find it is short on data.
2. **Joining**: You add more data.
3. **Validating**: You perform a quick check that the new data doesn't have mistakes or misspellings.
4. **Structuring**: You structure the data in different time periods and segments to understand trends.
5. **Validating:** You do another quick check to ensure the new columns you've made in structuring are correctly designed.
6. **Cleaning**: You check for outliers, missing data, and needs for conversions or transformations,
7. **Validating**: After cleaning, you double check the changes you made are correct and accurate,
8. **Presenting**: You share your dataset with a peer.

Notice you performed the "validating" practice iteratively, or multiple times, to make sure your changes to the data did not unwittingly introduce errors. Also, because you recognized the need for more data up front, the practice of "joining" was performed immediately following the practice of "discovering."

After you present your cleaned dataset to a peer, there is a good chance you will receive notes or ideas for more exploration and/or cleaning. Because of that, you will see even more iterations.

Pro tip: Data scientists expect to perform the practices of EDA multiple times on a dataset before they feel comfortable declaring it "clean" and ready for modeling or machine learning algorithms.

The importance of EDA in ethical machine learning

As algorithms and machine learning networks begin to make more and more decisions on behalf of individuals, companies, and even governments, the discussion of ethics and regulation becomes more and more important. According to the Institute for Ethical AI & Machine Learning, there are eight principles for developing machine learning systems in a responsible way.

Key principles of the EDA process

The following two principles are inherently part of the EDA process:

- **Human augmentation**: This principle ensures humans are inserted throughout the AI or machine learning algorithm systems for oversight. Thorough EDA, performed by data scientists, is perhaps one of the best ways to limit bias, imbalance, and inaccuracies being fed into an algorithm.
- **Bias evaluation**: Without human interference, bias is too easily injected and reproduced in machine learning models. Performing methodical EDA processes will lead data scientists to be aware of and act on biases and imbalances in the data.

Pro tip: The importance of assuring adherence to ethical standards cannot be overstated in the data career space. Data professionals need to continuously grow their capacities to recognize bias and discrimination by consistently applying an ethical mindset to their EDA work.

Beyond machine learning, EDA is applicable to nearly any important data-based decision. Moving forward, you will learn about many applications of EDA and the necessity of an iterative and non-sequential approach**Bias**: In data structuring, refers to organizing data results in groupings, categories, or variables that are misrepresentative of the whole dataset.

Cleaning: The process of removing errors that might distort your data or make it less useful; one of the six practices of EDA

Data visualization: A graph, chart, diagram, or dashboard that is created as a representation of information

Discovering: The process data professionals use to familiarize themselves with the data so they can start conceptualizing how to use it; one of the six practices of EDA

Exploratory data analysis (EDA): The process of investigating, organizing, and analyzing datasets and summarizing their main characteristics, often by employing data wrangling and visualization methods; the six main practices of EDA are: discovering, structuring, cleaning, joining, validating, and presenting.

Joining: The process of augmenting data by adding values from other datasets; one of the six practices of EDA

PACE: A workflow data professionals can use to remain focused on the end goal of any given dataset; stands for plan, analyze, construct, and execute

Presenting: The process of making a cleaned dataset available to others for analysis or further modeling; one of the six practices of EDA

Structuring: The process of taking raw data and organizing or transforming it to be more easily visualized, explained, or modeled; one of the six practices of EDA

Validating: The process of verifying that the data is consistent and high quality; one of the six practices of EDA

Import datasets with Python

In your career as a data professional, you will come across various datasets that have different file types or are stored in various databases. As you've learned previously, it is critical for you to know what these data types are and how to import data using Python. Below you will find examples of importing both databases through connections and data files into Python.

Although you will use the Coursera platform for Python coding, you will need to know how to work with and import CSV files if you'd like to download and open them outside of Coursera.

How to import a dataset from a CSV file

For this example, find a CSV file on your computer. If you don't have one, you can use a dataset of unicorn companies (companies that reached a valuation of $1 billion USD) from this course's Resources Opens in a new tab .

There are several different ways to import a CSV file into Python, but we will only review some of the more common ways. Start by using a with statement and open() function. Pass the **file name (or file path)** of the CSV file to the open() function along with an argument for the mode parameter of the function.

```
with open('file_path/file_name, mode=)
```

The mode is telling the Python library what to do with the file. When defining the **mode**, you use one of the following options:

- 'r' – read
- 'w' – write
- 'a' – append
- '+' – create new file

Typically, you'll be defining the mode inside the with open() argument field as 'r' because you want Python to open and read the CSV file.

Next, we'll add as file to the end, which is assigning the result to a variable name. In this case, we'll name it data.

```
with open('example_filepath/file', mode='r') as file:

        data = file.read()
```

Importing a CSV file using pandas

Instead of using Python's standard library to read a file, you can use pandas to import the CSV file into a dataframe. First, of course, you'll want to import the pandas library into your Python notebook.

```
import pandas as pd
```

Next, you'll use the read_csv() function to load the data into a dataframe. The file path then goes in the argument field.

```
df = pd.read_csv('file_path/file_name')
```

Note: *You can also use this same syntax for importing a CSV file that is stored on the internet. In the place of the filename, you would simply copy and paste the url.*

How to import data by connecting to a database

There are a number of database solutions that you can connect to with Python, such as BigQuery, MySQL, SQLite, and Oracle. Databases are a convenient way for companies and organizations to store huge amounts of data.

If the dataset is small enough, it can be downloaded and manipulated locally on your computer. However, often the datasets kept in databases are too large to access in their entirety on a personal computer. In this case you have a number of different options, most of which involve querying the database with SQL to obtain specific tables of interest. In other words, you extract select parts—usually specified rows and/or columns—from the whole dataset. The manner in which querying is done can vary with respect to systems, platforms, and interfaces. Because of this variability, this reference guide will only present a couple of different ways to query databases. Specifically, it will explore BigQuery, Google's data warehouse that provides a wide range of tools and services to facilitate analysis.

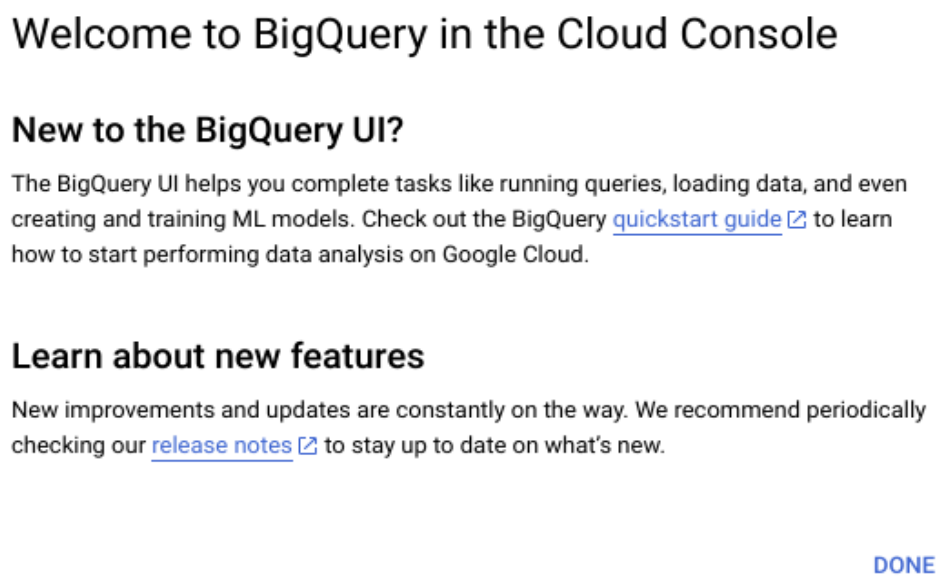
Downloading data from BigQuery

Step 1: Access BigQuery

BigQuery allows you to upload data for storage, and it also has a number of publicly available datasets to explore. You can access these public datasets for free using BigQuery Sandbox, which requires a free Google account. Sandbox gives you 10 GB of active storage and 1 TB of processed query data each month for free.

Step 2: Perform a query

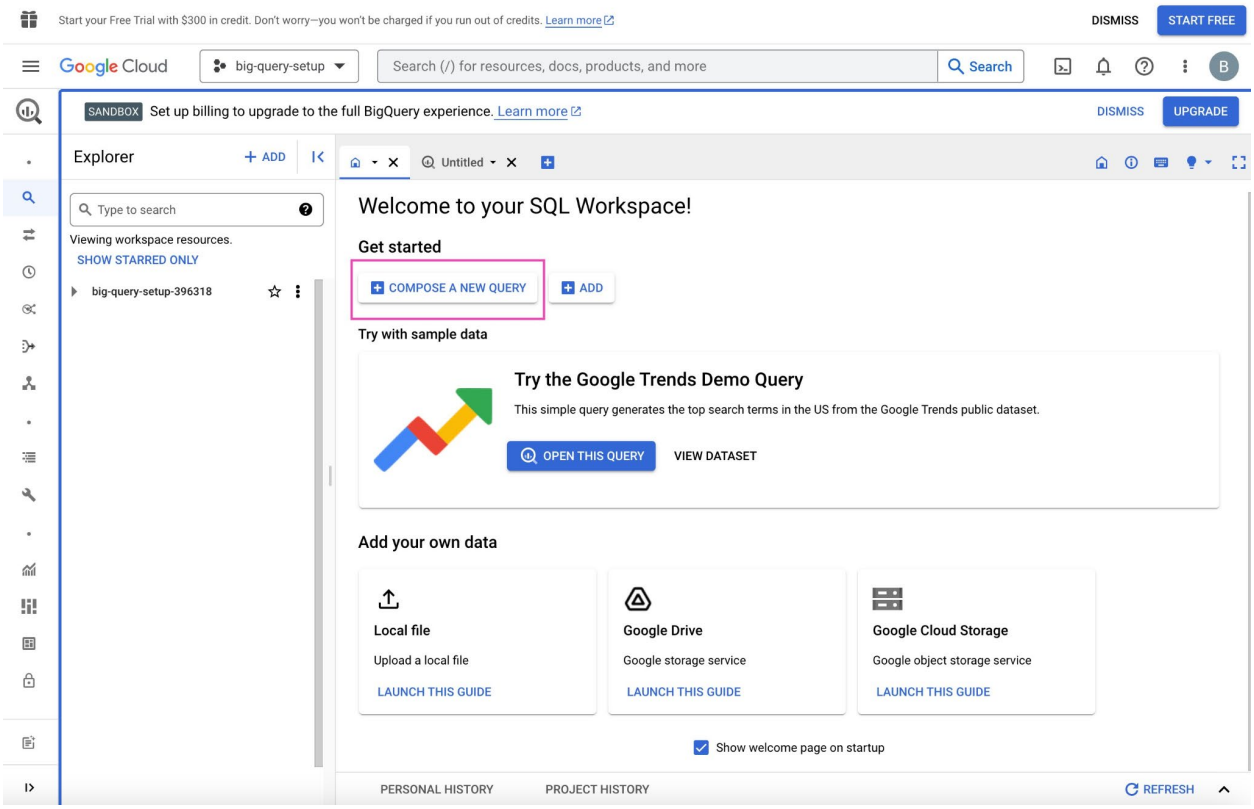
Once you have authenticated your account and created a new project as indicated in the instructions linked in step one, you're ready to query a database. Note that if it is your first time logging in, you may encounter a window asking "New to the BigQueryUI?" with a link to a quickstart guide.



[**Alt-text:** Screenshot of the "New to BigQuery UI?" window]

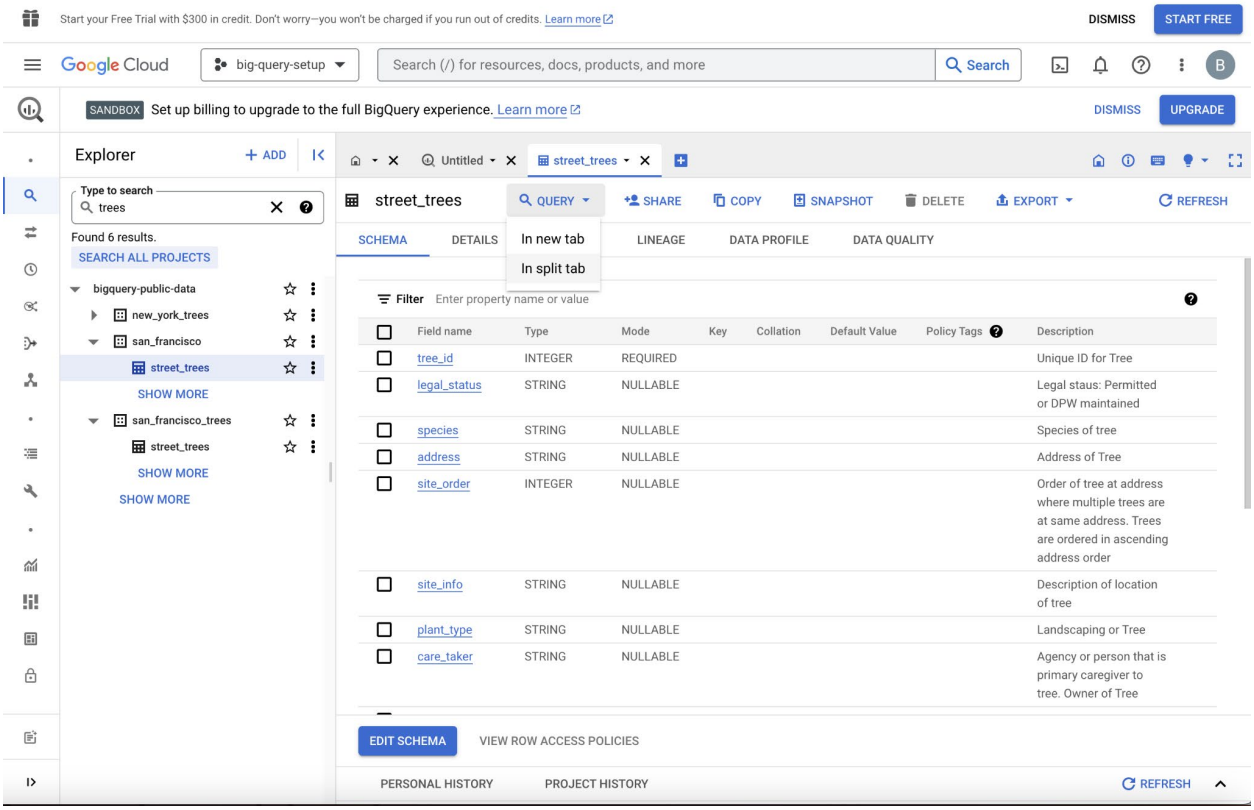
The quickstart guide will guide you through the same steps as those presented to you here.

From the “Welcome to your SQL Workspace!” page, click the “Compose a new query” button.

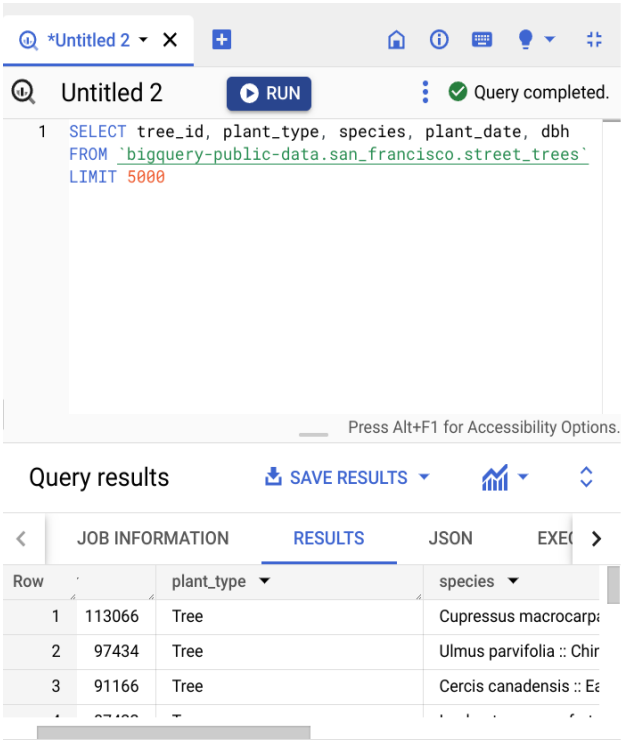


Click into the search bar in the Explorer on the left side of the page. For example, you can search for “trees.” Initially, this will return zero results. However, click “Search all projects” and it will return applicable datasets from the biquery-public-data project and premade tables from those datasets.

Click the street_trees table in the san_francisco dataset. The metadata for this table will appear in a panel to the right. Then, click “Query” from the menu at the top of the metadata panel. You can opt to query in a new tab or in a split-pane of the current window.



Now, you can query the table using SQL. For example, the query in the following screenshot selects 5,000 rows with columns of tree_id, plant_type, species, plant_date, and dbh – defined as "depth, height."



Once you are satisfied with your query, click the “Run” button at the top of the SQL query panel. The results will display below, and there is a button to “Save results,” which allows you to save the resulting table in different locations and formats. From there, you can read the data into your notebook.

Using notebooks within BigQuery

Another way to access data on BigQuery is by using the tools within the BigQuery platform itself. This workflow more closely resembles what data professionals would use when working with very large datasets stored in the cloud. Essentially, you set up a virtual machine on BigQuery. A virtual machine is a computer that has its own CPU, memory, software, etc., just like any other computer, only it does not have its own dedicated hardware; they most often exist as a partition on a server. You can work in a Jupyter notebook on the virtual machine on the BigQuery platform, from which you can query and pull in data directly.

This process requires you to set up a payment method. However, new users get a $300 credit, and a ML instance is only a few cents per minute, so you’ll get approximately 2,000 hours of free usage before incurring any charges. There are a lot of great tutorials for setting this up. For instance, if you search for “How to use Jupyter notebook in Google Cloud AI,” you’ll find a number of useful videos and blogs on the topic.

Using notebooks outside of BigQuery

It’s also possible to query data on BigQuery from notebooks that are not on the BigQuery platform. However, the details of this process are dependent on a number of factors, including the platform that is hosting the notebook, the operating environment, and the specific location of the data being accessed. Therefore, we will not go into depth on this method. Feel free to explore this on your own, though. You’ll find many helpful online resources that are just a search away.

Key takeaways

There are lots of different kinds of data, which means there are numerous ways to import data. Learning several methods to import data, whether it be from a data file or a database, will build a solid foundation for your career as a data professional.

Resources for more information

To learn more about importing data into Python, you can refer to the following links:

- An overview of importing data in Python
- How to connect to BigQuery from a Colab

Citations:

#	Title	Link
1.	An Overview of importing data in Python	**https://towardsdatascience.com/an-overview-of-importing-data-in-python-ac6aa46e0889**
2.	Downloading BigQuery data to pandas using the BigQuery Storage API	**https://cloud.google.com/bigquery/docs/bigquery-storage-python-pandas**

Python functions for the discovery of a dataset

Python reference guide for EDA: Discovering

Use the following Python Pandas functions to help you learn about a dataset when you encounter it for the first time.

DataFrame.head(X)

- The head() function will display the number of dataset rows you input in the argument field.
- For the "X" in the argument field, input the number of rows you want displayed in a Python notebook. The default is 5 rows.
- Once executed, the head() function looks like this:

  df.head(10)

n/a	Date	number of strikes	center point geom
0	2018-01-03	194	POINT(-75 27)
1	2018-01-03	41	POINT(-78.4 29)
2	2018-01-03	33	POINT(-73.9 27)
3	2018-01-03	38	POINT(-73.8 27)
4	2018-01-03	92	POINT(-79 28)
5	2018-01-03	119	POINT(-78 28)
6	2018-01-03	35	POINT(-79.3 28)

7	2018-01-03	60	POINT(-79.1 28)
8	2018-01-03	41	POINT(-78.7 28)
9	2018-01-03	119	POINT(-78.6 28)

Note: In a Python notebook, the results of head() will not include a table with visible grid lines.

DataFrame.info(X)

- The info() function will display a summary of the dataset, including the range index, dtypes, column headers, and memory usage.
- Leaving the argument field blank will return a full summary. As an option, in the argument field you can type in "show_counts=True," which will not return any null fields.
- Once executed, the info() function looks like this:

```
df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex:3401012 entries, 0 to 3401011

Data columns (total 3 columns):

#    Column              Dtype

--   ----                -----

0    date                object

1    number_of_strikes   int64

2    center_point_geom   object

Dtypes: int64(1), object(2)

Memory usage 77.8+ MB
```

Dataframe.describe(X)

- The describe() function will return descriptive statistics of the entire dataset, including total count, mean, minimum, maximum, dispersion, and distribution.
- Leaving the argument field blank will default to returning a summary of the data frame's statistics. As an option, you can use "include=[X]" and "exclude=[X]" which will limit the results to specific data types, depending on what you input in the brackets.
- Once executed, the describe() function looks like this:

  df_joined.describe()

N/A	longitude	latitude	number_of_strikes_x	number_of_strikes_y
count	717530.00	717530.00	717530.00	323700.00000
mean	-90.875445	33.328572	21.637081	25.410587
std	13.648429	7.938831	48.02952	57.421824
min	-133.9000	16.600000	1.00000	1.000000
25%	-102.80000	26.900000	3.00000	3.000000
50%	-90.300000	33.200000	6.00000	8.000000
75%	-80.900000	39.400000	21.00000	24.000000
max	-43.800000	51.700000	2211.00000	2211.000000

***Note**: In a Python notebook, the results of describe() will not include a table with visible grid lines.*

DataFrame.shape

- 'Shape' returns a tuple representing the dimensions of the dataset by number of rows and columns. The code will look something like this:

Df.shape

(3401012, 3)

Key takeaways

Head(),**info()**,**describe()**, and **shape** are all Python functions that data scientists can use to understand a dataset at a high level. The information learned from running these functions will serve to inform the remainder of your EDA work when you use Python to analyze data throughout your career.

Resources for more information

For more information on the EDA discovering functions above and others like it, you can use the online Pandas reference guide:

A list of Pandas dataframe functions

Manipulating datetime strings in Python

Below, you will find a table with the datetime functions you can use to help you manipulate datetime objects in different ways.

Code	Format	Example
%a	Abbreviated workday	Sun
%A	Weekday	Sunday
%b	Abbreviated month	Jan
%B	Month name	January
%c	Date and time	Sun Jan 1 00:00:00 2021
%d	Day (leading zeros)	01 to 31

%H	24 hours	00 to 23
%I	12 hours	01 to 12
%j	Day of year	001 to 366
%m	Month	01 to 12
%M	Minute	00 to 59
%p	AM or PM	AM/PM
%S	Seconds	00 to 61
%U	Week number (Sun)	00 to 53
%W	Week number (Mon)	00 to 53
%w	Weekday	0 to 6
%x	Locale’s appropriate date representation	08/16/88 (None); 08/16/1988 (en_US); 16.08.1988 (de_DE)
%X	A locale’s appropriate time representation	21:30:00 (en_US); 21:30:00 (de_DE)
%y	Year without century	00 to 99
%Y	Year	2022
%z	Offset	+0900
%Z	Time zone	EDT/JST/WET etc (GMT)

● Datetime functions to remember.

- All of the following date string manipulations require the datetime package to be imported first.

Code	Input Type	Input Example	Output Type	Output Example
datetime.strptime(“25/11/2022”, “%d/%m/%Y”)	string	“25/11/2022”	DateTime	“2022-11-25 00:00:00”
datetime.strftime(dt_object, “%d/%m/%Y”)	DateTime	“2022-11-25 00:00:00”	string	“25/11/2022”
dt_object = datetime.strptime(“25/11/2022”, “%d/%m/%Y”) datetime.timestamp(dt_object)	string	“25/11/2022”	float (UTC timestamp in seconds)	1617836400.0
datetime.strptime(“25/11/2022”, “%d/%m/%Y”).strftime(“%Y-%m-%d”)	string	“25/11/2022”	string	“2022-11-25”
datetime.fromtimestamp(1617836400.0)	float (UTC timestamp in seconds)	1617836400.0	DateTime	datetime.datetime(2021, 4, 7, 23, 0)
datetime.fromtimestamp(1617836400.0).strftime(“%d/%m/%Y”)	float (UTC timestamp in seconds)	1617836400.0	string	“'07/04/2021'”
from pytz import timezone ny_time = datetime.strptime(“25-11-2022 09:34:00-0700”, “%d-%m-%Y %H:%M:%S%z”) Tokyo_time = ny_time.astimezone(timezone(‘Asia/Tokyo’))	string	NewYork timezone “25-11-2022 09:34:00-0700”	DateTime	Tokyo timezone 2022, 11, 26, 1, 34, JST+9:00:00 STD>
datetime.strptime(“20:00”, “%H:%M”).strftime(“%I:%M %p”)	string	“20:00”	string	“08:00 PM”
datetime.strptime(“08:00 PM”, “%I:%M %p”).strftime(“%H:%M”)	string	“08:00 PM”	string	“20:00”

Datetime in NumPy and pandas

- A preface regarding terminology in the following section: datetime refers to the specific module of that name in the Python standard library or to the specific class within that module. Datetime (or uncapitalized, datetime) refers to any date/time-related object from any library or language.

- You've learned that the datetime module in Python's standard library contains a number of classes used to work with time data, including date, time, datetime, timedelta, timezone, and tzinfo. Remember, modules are similar to libraries, in that they are groups of related classes and functions, but they are generally subcomponents of libraries. Classes are data types that bundle data and functionality together.

- NumPy and pandas have their own datetime classes that offer significant performance boosts when working with large datasets. Pandas datetime classes, like the rest of the pandas library, are built on NumPy. These classes have very similar (and in many cases identical) functionality to Python's native datetime classes, but they run more efficiently due to NumPy and pandas' vectorization capabilities. Therefore, although you *can* use datetime data in pandas, it's generally better to use NumPy or pandas datetime objects when working in pandas, if possible.

- NumPy's datetime classes include, most notably, datetime64 and timedelta64. Like datetime objects, datetime64 objects contain date and time information in a single data structure; and, like timedelta objects, timedelta64 objects contain information pertaining to spans of time.

- Pandas' datetime classes include Timestamp, Timedelta, Period, and DateOffset.
- Because these classes are efficient and dynamic in their capabilities, you often don't need to import the datetime module when working with datetime data in pandas. Also, pandas will automatically recognize datetime-like data and convert it to the appropriate class when possible. Here's an example:

```
data = ['2023-01-20', '2023-04-27', '2023-06-15']
my_series = pd.Series(data)
my_series
0    2023-01-20
1    2023-04-27
2    2023-06-15
dtype: object
```

- This series contains string data, but it can be converted to datetime64 data using the pd.to_datetime() function:

```
my_series = pd.to_datetime(my_series)
my_series


0   2023-01-20
1   2023-04-27
2   2023-06-15
dtype: datetime64[ns]
```

- Refer to the pandas to_datetime() documentation for more information about this function.
- When a Series object contains datetime data, you can use dt to access various properties of the data. For example:

```
print(my_series.dt.year)
print()
print(my_series.dt.month)
print()
print(my_series.dt.day)

0    2023
1    2023
2    2023
dtype: int64

0    1
1    4
2    6
dtype: int64

0    20
1    27
2    15
dtype: int64
```

- Note that it's not uncommon to import the datetime module from Python's standard library as dt. You may have encountered this yourself. In such case, dt is being used as an alias. The pandas dt

Series accessor (as demonstrated in the last example) is a different thing entirely. Refer to the pandas dt accessor documentation for more information.

Key takeaways

- Use reference guides like the tables above throughout your career to help remind you of the different ways to manipulate datetime objects. Even experts in the field use reference guides, rather than memorizing all this information. Getting familiar with guides like these will be beneficial because you will be using them throughout your career as a data professional.

Reference guide: Python functions for structuring a dataset

Pandas structuring reference guide

As you've learned, there are far too many Python functions to memorize all of them. That's why, as every data professional will tell you, you'll be using reference sheets and coding libraries nearly every day in your data analysis work.

The following reference guide will help you identify the most common Pandas tools used for structuring data. Note that this is just for reference. For detailed information on how each method works, including explanations of every parameter and examples, refer to the linked documentation.

Combine data

Note that for many situations that require combining data, you can use a number of different functions, methods, or approaches. Usually you're not limited to a single "correct" function. So if these functions and methods seem very similar, don't worry! It's because they are! The best way to learn them, determine what works best for you, and understand them is to use them!

df.merge()

- A method available to the DataFrame class.
- Use df.merge() to take columns or indices from other dataframes and combine them with the one to which you're applying the method.
- Example:

  df1.merge(df2, how='inner', on=['month','year'])

pd.concat()

- A pandas function to combine series and/or dataframes

- Use pd.concat() to join columns, rows, or dataframes along a particular axis
- Example:

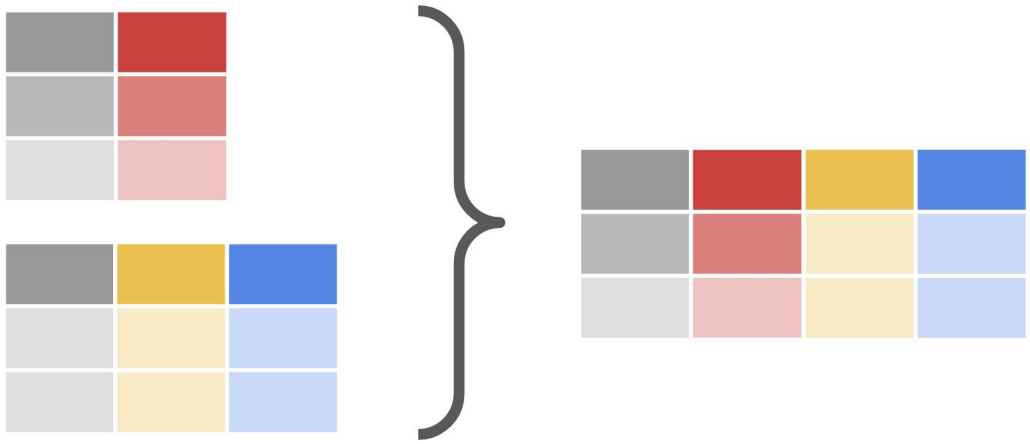
```
df3 = pd.concat([df1.drop(['column_1','column_2'], axis=1), df2])
```

df.join()

- A method available to the DataFrame class.
- Use df.join() to combine columns with another dataframe either on an index or on a key column. Efficiently join multiple DataFrame objects by index at once by passing a list.
- Example:

```
df1.set_index('key').join(df2.set_index('key'))
```

Visual representation of a combination:



Extract or select data

df[[columns]]

- Use **df[[columns]]** to extract/select columns from a dataframe.
- Example:

```
df[['animal', 'legs']]
```

df.select_dtypes()

- A method available to the DataFrame class.
- Use df.select_dtypes() to return a subset of the dataframe's columns based on the column dtypes (e.g., float64, int64, bool, object, etc.).
- Example:

```
df2 = df.select_dtypes(include=['int64'])
```

Visual representation of extraction:

Filter data

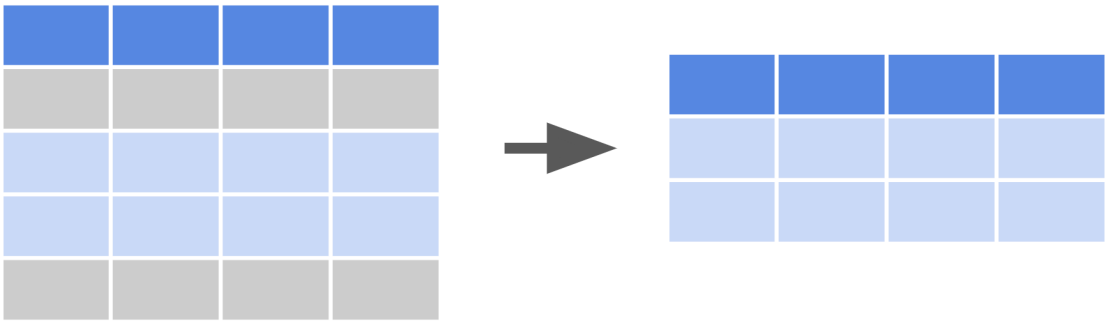
Recall from Course 2 that Boolean masks are used to filter dataframes.

df[condition]

- Use df[condition] to create a Boolean mask, then apply the mask to the dataframe to filter according to selected condition.
- Example:

```
df[df['class']=='Aves']
```

Visual representation of filtering:



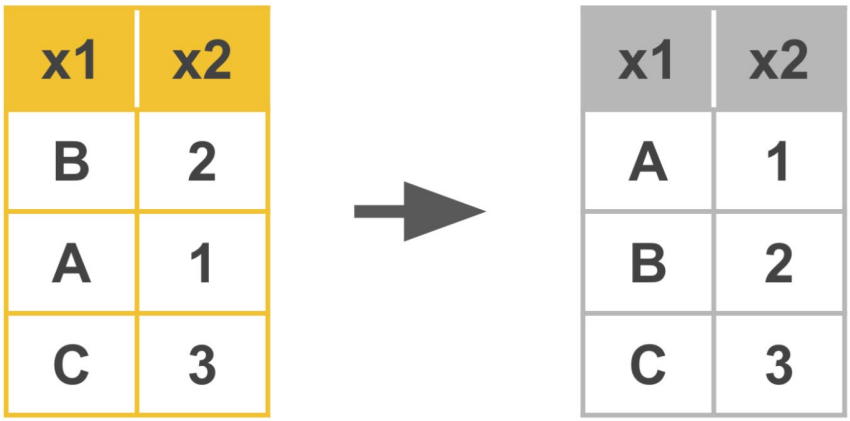
Sort data

df.sort_values()

- A method available to the DataFrame class.
- Use df.sort_values() to sort data according to selected parameters.
- Example:

```
df.sort_values(by=['legs'], ascending=False)
```

Visual representation of sorting:



Slice data

df.iloc[]

- Use df.iloc[] to slice a dataframe based on an integer index location.

- Examples:

  df.iloc[5:10, 2:] → selects only rows 5 through 9, at columns 2+
  df.iloc[5:10] → selects only rows 5 through 9, all columns
  df.iloc[1, 2] → selects value at row 1, column 2
  df.iloc[[0, 2], [2, 4] → selects only rows 0 and 2, at columns 2 and 4

df.loc[]

- Use df.loc[] to slice a dataframe based on a label or Boolean array.
- Example:

  df.loc[:, ['color', 'class']]

Key takeaways

The tools in this reference guide are foundational to structuring data, including filtering, sorting, merging, and slicing. You will find yourself using them throughout your career as a data professional.

Resources for more information

Refer to these links for more details on Python functions and their various parameters.

- Pandas documentation to describe parameters in Python functions
- W3schools provides explanations for Python functions in an easy-to-understand way

Histograms

As you've been learning, the purpose of exploratory data analysis (EDA) is just what its name says: explore and analyze the data. As a data professional, you'll almost always begin with a guiding question or objective, such as, "Where are the highest emitters of carbon dioxide located?" or "Determine the characteristics of people most likely to buy product X." Reflecting on this often throughout your process creates a driving force that keeps you on track.

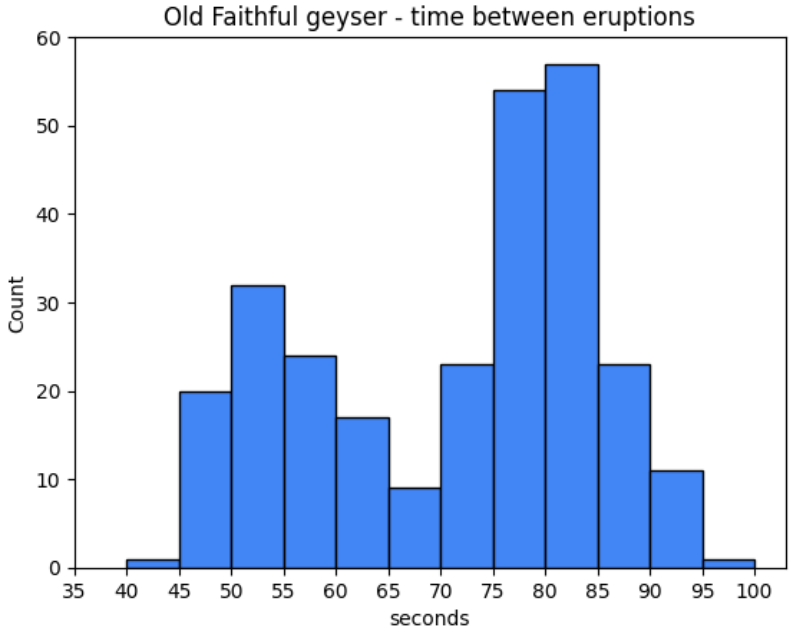
One of the most important tools at your disposal when exploring data is the **histogram**. A histogram is a graphical representation of a frequency distribution, which shows how frequently each value in a dataset or variable occurs. It's essential for data professionals to understand the distributions of their data, because this

knowledge drives many downstream decisions around experiment design, modeling, and further analysis. In this reading, you'll learn about histograms, what they are, how to make them, and how to interpret them.

Introduction to histograms

Histograms are commonly used to illustrate the shape of a distribution, including the presence of any outliers, the center of the distribution, and the spread of the data. Histograms are typically represented by a series of bars, where each bar represents a range of values. Bar height represents the frequency or count of the data points within that range.

The following example is a histogram of the number of seconds between eruptions of the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.



The x-axis represents the number of seconds between eruptions. The y-axis represents the eruption count. So, as indicated by the second bar in the graph, there are 20 eruptions that occurred after a wait time of 45-49 seconds.

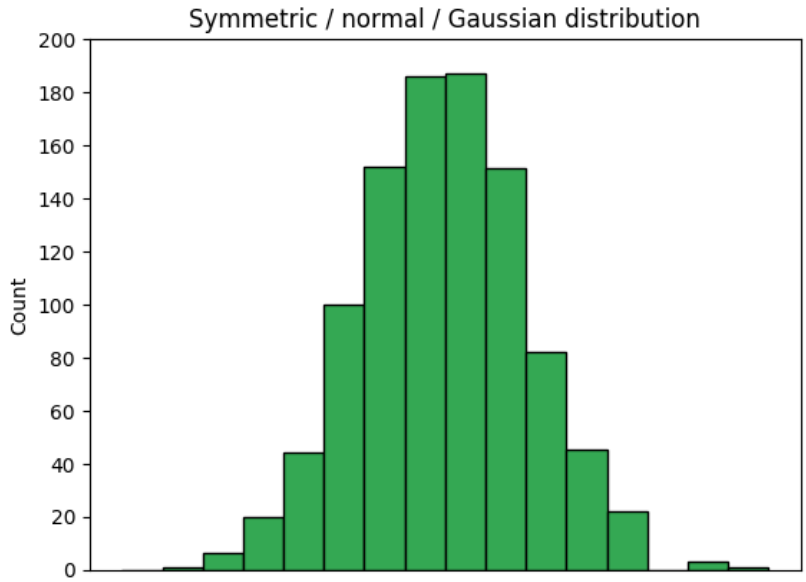
The importance of histograms

Histograms are an essential tool for understanding the characteristics of a dataset. They provide a visual representation of the data's distribution and enable data professionals to identify patterns, trends, or outliers within the data. Histograms can also help data professionals choose appropriate statistical tests and models for the data and determine whether the data meets any assumptions required for the analysis. Histograms are widely used in any field and any situation that requires any kind of data analysis, including finance, health care, engineering, and social sciences.

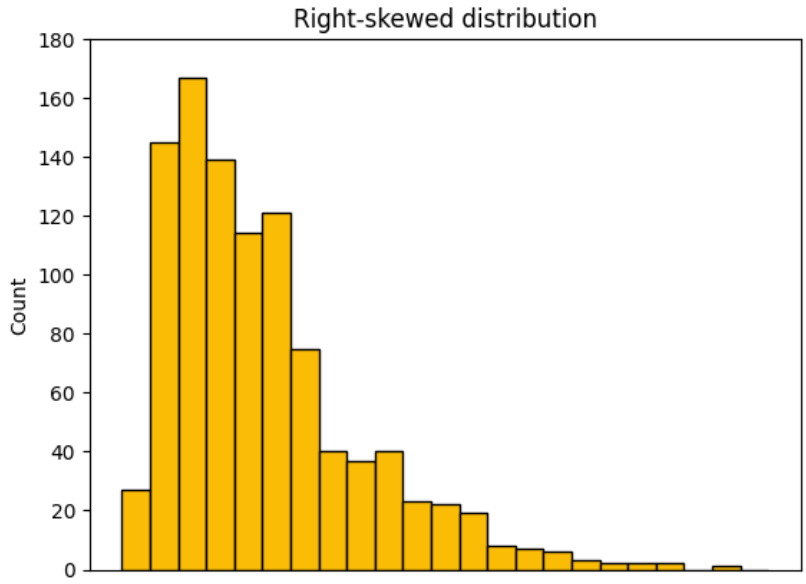
How to interpret histograms

Interpreting histograms involves understanding the shape, center, and spread of the distribution. There are several common shapes of histograms, including:

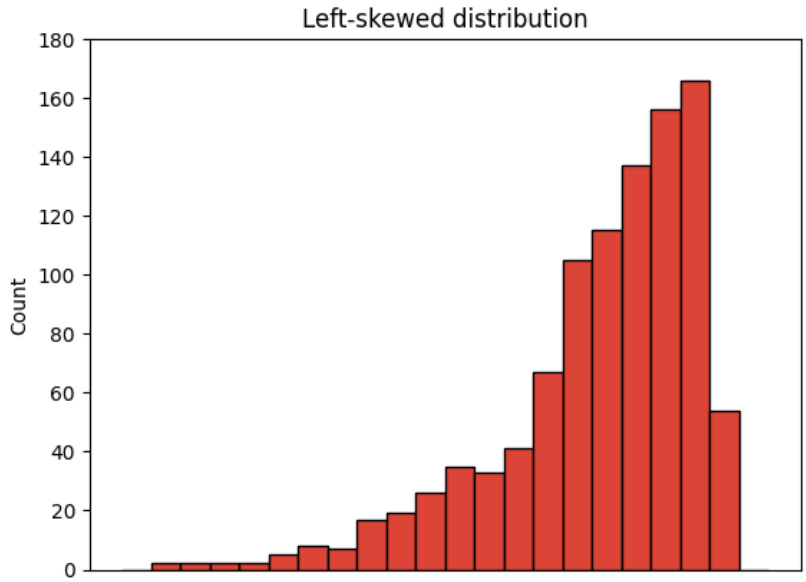
1. Symmetric: A symmetric histogram has a bell-shaped curve with a peak in the middle, indicating that the data is evenly distributed around the mean. This is also known as a normal, or Gaussian, distribution.



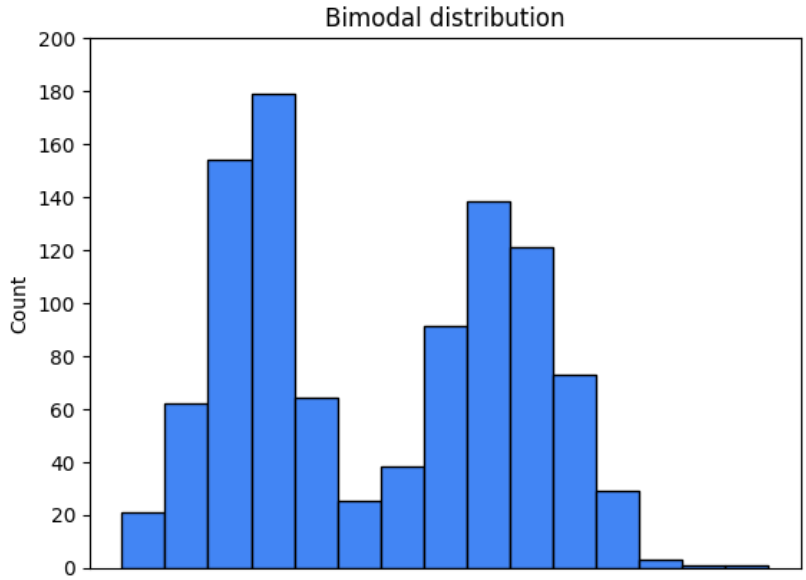
2. Skewed: A skewed histogram has a longer tail on one side than the other. A right-skewed histogram has a longer tail on the right side, indicating that there are more data points on the left side of the histogram.



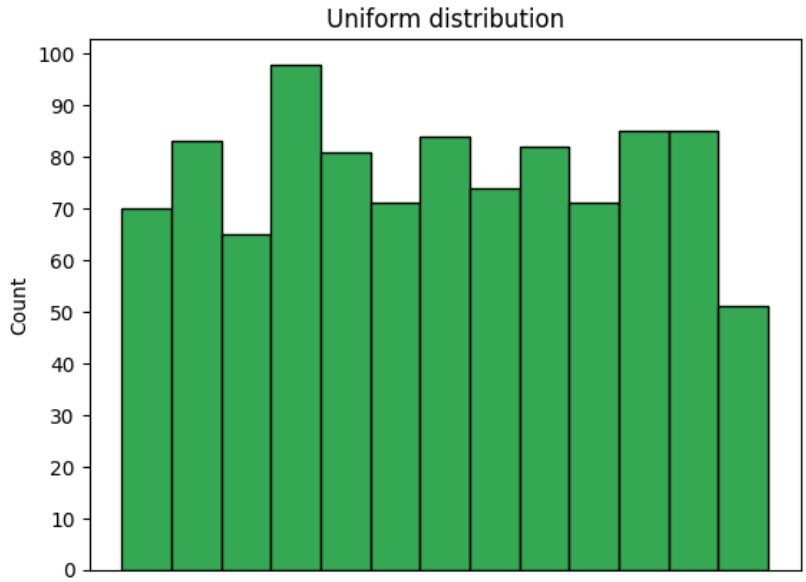
A left-skewed distribution has a longer tail on the left side, indicating more data points on the right side.



3. Bimodal: A bimodal histogram has two distinct peaks, indicating that the data has two modes.



4. Uniform: A uniform histogram has a flat distribution, indicating that all data points are evenly distributed.



The examples provided are not the only distributions you'll encounter, but they are some of the most common. Soon, you will learn more about distributions.

Now, return to the Old Faithful geyser histogram at the beginning of this reading. Ask yourself: what type of distribution is represented by that graph? In addition to the shape, it's important to understand the center and spread. The center of the distribution is typically represented by the mean or median, while the spread is represented by the standard deviation or range of the data. The center and spread can provide insights into data concentration and variability.

How to create histograms

Python's seaborn and matplotlib libraries provide simple and powerful options to create histograms.

plt.hist(x, bins=10, …)

To generate a histogram in matplotlib, use the **hist()** function in the pyplot module. The function can take many different arguments, but the primary ones are:

- x: A sequence of values representing the data you want to plot. It can be a list, tuple, NumPy array, pandas series, and so on.
- bins: The number of bins you want to sort your data into. The default value is 10, but this parameter can be an int, sequence, or string. If you use a sequence, it defines the bin edges, including the left edge of the first bin and the right edge of the last bin. In other words, if bins = [1, 3, 5, 7], then the first bin is [1–3) (including 1, but excluding 3) and the second [3–5). The last bin, however, is [5–7], which includes 7. A string refers to a predefined binning strategy supported by numpy. Refer to the documentation for more information.

The following example demonstrates how to generate the Old Faithful geyser histogram from the beginning of this reading using the **plt.hist()** function.

```
# Plot histogram with matplotlib pyplot
```

```
plt.hist(df['seconds'], bins=range(40, 101, 5))

plt.xticks(range(35, 101, 5))

plt.yticks(range(0, 61, 10))

plt.xlabel('seconds')

plt.ylabel('count')

plt.title('Old Faithful geyser - time between eruptions')

plt.show();
```

RunReset

In this case, the data being plotted is the seconds column of the dataframe. The bins begin at 40 seconds and go to 100 seconds in steps of five, for a total of 12 bins.

sns.histplot(x, bins, binrange, binwidth …)

One way to generate a histogram in seaborn is to use the **sns.histplot()** function. Like the matplotlib function, **sns.histplot()** can take many arguments. Here are some important ones:

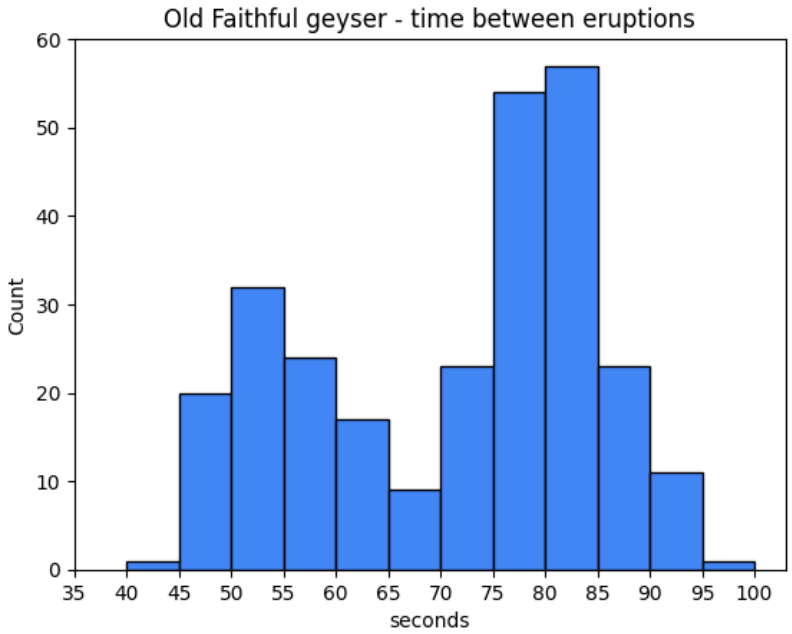
- x: The data sequence. Same as **plt.hist()**
- bins: Same as **plt.hist()**
- binrange: Lowest and highest value for bin edges; can be used either with **bins** or **binwidth**; defaults to data extremes
- binwidth: Width of each bin, overrides **bins** but can be used with **binrange**

The following example is the code used to generate the Old Faithful geyser histogram using the seaborn **histplot()** function. It uses all of the previously mentioned parameters. Run this code block to generate a histogram.

Notice in this case that **binrange** is defined from 40 to 100 and **binwidth** is set to 5. This produces the same results as setting **bins=range(40, 101, 5)**. This example also makes use of a couple of style parameters by specifying a particular color using hex code notation and setting the color saturation level to 100%, as indicated by the **alpha** parameter.

Note: The following code block is not interactive.

```
# Plot histogram with seaborn

ax = sns.histplot(df['seconds'], binrange=(40, 100), binwidth=5, color='#4285F4', alpha=1)

ax.set_xticks(range(35, 101, 5))

ax.set_yticks(range(0, 61, 10))

plt.title('Old Faithful geyser - time between eruptions')

plt.show();
```



Key takeaways

Histograms help data professionals understand the frequency distributions of their dataset and variables. Knowledge of the shape and type of data distribution will affect important downstream decisions, such as statistical tests and model architecture selection. Additionally, knowing the shape of your data gives valuable insights into the story that your data is telling you by helping you understand its distributional trends.

Box plot: A data visualization that depicts the locality, spread, and skew of groups of values within quartiles

CSV file: A simple text file that can be easy to import or store in other softwares, platforms, and databases

Database (DB) file: A file type used to store data, often in tables, indexes, or fields

Data source: The location where data originates

Extracting: The process of retrieving data out of data sources for further data processing or storage

Filtering: The process of selecting a smaller part of a dataset based on specified values and using it for viewing or analysis

First-party data: Data that was gathered from inside your own organization

Grouping: The process of aggregating individual observations of a variable into groups

Hypothesis: A theory or an explanation, based on evidence, that has not yet been refuted

Info(): Gives the total number of entries, along with the data types—called Dtypes in pandas—of the individual entries

Int64: A standard integer data type, representing numbers somewhere between negative nine quintillion and positive nine quintillion

JSON file: A data storage file that is saved in a JavaScript format

Merging: A method to combine two (or more) different data frames along a specified starting column(s)

Second-party data: Data that was gathered outside your organization but directly from the original source

Slicing: A method for breaking information down into smaller parts to facilitate efficient examination and analysis from different viewpoints

Sorting: The process of arranging data into a meaningful order for analysis

String: A sequence of characters and punctuation that contains textual information

Third-party data: Data gathered outside your organization and aggregated

Data deduplication with Python

As you've learned, the data cleaning and validating practices include several different steps, including handling missing data, outliers, and label encoding; checking for misspellings; and, handling duplicates. As a data professional, it will be your task to know how best to handle data values in those categories. In this reading, you'll learn more about handling duplicates. You will also learn to identify and decide whether deduplication is the right strategy for a dataset. In addition, you will learn some common Python functions for handling duplicates.

Identifying duplicates

Before we make any decisions on whether to remove duplicate values or not, we should first determine if duplicate values are present in our dataset.

A simple way to identify duplicates is to use the **duplicated()** function from Pandas. **duplicated()** is a method of the **DataFrame** class.

This function returns a series of "true/false" outputs, with "true" indicating the data value is a duplicate, and "false" indicating it is a unique value.

Here's an example of a five-row dataframe:

df

Using the **duplicated()** function, the result is that one has been marked "True," indicating it is a duplicate.

```
print(df)

print()

print(df.duplicated())
```

RunReset

Identifying duplicates for an entire dataframe will be different than a single column or index. Be sure when you use the **duplicated()** function for an entire dataframe. The **duplicated()** function will only return *entire rows* that have exactly matching values, not just individual matching values found within a column. If you wish to identify duplicates for only one column or a series of columns within a dataframe, you will need to include that in the "subset" portion of the argument field of the **duplicated()** function. Going further, if you'd like to specify which of the duplicates to keep as the "original" as opposed to the duplicate, you can specify that in the **keep** portion of the argument field.

Below is an example of identifying duplicates in only one column (subset) of values and labeling the last duplicates as "false," so that they are "kept":

```
print(df)

print()

print(df.duplicated(subset=['type'], keep='last'))
```

Decision time: To drop or not to drop?

As you've learned, every dataset is unique and you cannot treat every dataset the same. When you are making the decision on whether to eliminate duplicate values or not, think deeply about the **dataset itself** and about **the objective you wish to achieve**. What impact will dropping duplicates have on your dataset and your objective?

1. Deciding to drop

You should **drop or eliminate** duplicate values if duplicate values are clearly mistakes or will misrepresent the remaining unique values in the dataset.

House address	Latest price
567432 Pickled Puppeteer Pike	275,300
10009 Al B Kerxy St.	199,999
984 Fortitudnal Fort Rd	298,342
6743 Believed Blvd	245,654
14573 S Match Rd	203,778
32 South Uhvdabor Dur	299,444
27000 N Umberland St	270,008
~~14573 S Match Rd~~	~~203,778~~

For example, you can be reasonably sure that a data professional will (in most cases) eliminate duplicate values of a dataset containing house addresses and house prices. Counting the same house twice will (in most cases) misrepresent any conclusions drawn from the dataset as a whole, such as average house price, total house price, or even total number of houses. In a case like this, a data professional would almost certainly eliminate the duplicate data so as to fairly represent the remaining data during analysis and visualization.

2. Deciding to NOT drop

You should **keep** duplicated data in your dataset if the duplicate values are clearly **not** mistakes and should be taken into account when representing the dataset as a whole.

Throw / Attempt No.	Distance (m)
1	12.3
2	13.6
3	12.3
4	12.7
5	12.9
6	13.4
7	12.9
8	13.2

For example, a dataset marking the number of throws and distances of an Olympic shot-put athlete in training will likely include several duplicate distances; just by nature of number of attempts and the limits a person can have a weighted ball, there will be duplicate values—particularly if the distance measurements are labeled to only 1 or 2 decimal places. In a case like this, a data professional would almost certainly keep all of the data to fairly represent it as a whole during analysis and visualization.

Don’t be duped — How to do deduplication

Before we get back into Python and learn how to eliminate duplicates, let’s first define the term “deduplication”:

- **Deduplication:** The elimination or removal of matching data values in a dataset.

There are a number of different libraries, functions, and methods in Python you could use to remove matching data values.

One of the more common functions to use is in Pandas: **drop_duplicates()**

drop_duplicates() is another **DataFrame** method. It's used to create a new dataframe with all of the duplicate rows removed.

For example, use a dataframe from earlier in this reading:

df

Now apply the drop duplicates function:

```
df.drop_duplicates()
```

You'll notice in the resulting output that the duplicate row of data was removed, leaving the remaining unique values intact.

Note: Keep in mind that the **drop_duplicates()** function as written above will only drop duplicates of exact matches of **entire rows of data**. If you wish to drop duplicates within a single column, you will need to specify which columns to check for duplicates using the **subset** keyword argument.

This example drops all rows that have duplicate values in the **style** column (except for the first occurrence):

```
print(df)

df = df.drop_duplicates(subset='style')

print()

print(df)
```

RunReset

And this example drops all rows (except the first occurrence) that have *duplicate values in both* the **style** and **rating** columns:

```
print(df)

df = df.drop_duplicates(subset=['style', 'rating'])

print()

print(df)
```

RunReset

Key Takeaways

Identifying duplicate data values in a dataset is an important part of EDA (or "Exploratory Data Analysis") practices, specifically cleaning and validating. After identifying duplicates, think about the impact to the dataset and your analysis objective when choosing to eliminate duplicates or not eliminate duplicates.

Additional Resources

Want to learn more about duplicates and deduplication? Check out the following additional links.

- Look at Pandas documentation to learn more about the parameters of the argument field
- W3 Schools: Pandas - removing duplicates

Protect the people behind the data

We've all been there…

Whether at work or at school, there's a moment of realization about an essay or a project you've been working on—you suddenly realize you've made a mistake, and it needs to be fixed.

"But think of all of the trouble that will cause," you think. "It will make the project late, and everyone will find out I made a mistake."

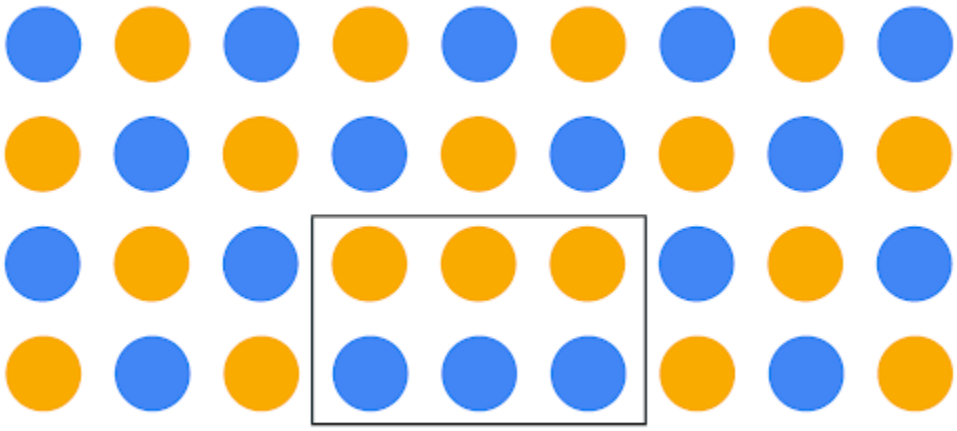
Is it really worth the time and effort to stop the process and fix it?

For data professionals, the answer needs to always be ***YES***.

The big picture

Even seasoned data professionals can fail to see the big picture. As intuitive as our brains can be, we often fail to see how a small error, a tiny change, or a seemingly insignificant choice about data analysis can have significant implications on a process, a business, or even a whole population.

To help illustrate this, imagine a grid of alternating orange and blue circles. Now, imagine there is a person who can only see a small section of six circles. What if this person decides that it makes more sense for the circles to be arranged by groups, with orange circles on top and blue on bottom? You'll find that even though the circles within the rectangle may be perceived as orderly, the reality is the change has actually disrupted the larger pattern. Now imagine the rectangle of six circles is a department and the bigger grid is an entire company.



An ethical mindset

Hopefully, this illustration helps you understand the importance of context and scope when dealing with data. One principle that can help data professionals keep data in context is ethics and developing an ethical mindset.

Here are three important concepts to help you foster an ethical mindset as you seek to tell stories with data:

- Models have lives beyond code blocks and data strings.
- Data science needs regulatory compliance.
- Customers should choose how their data is used.

Models have lives beyond the code blocks and data strings

There can be a tendency to think of data as strictly numbers and math. The reality is that data consists of people-driven inputs. Data professionals should never lose sight of the fact that analysis in all forms—visualizations, models, decisions, and strategies—impacts people.

The end goal of all data analysis and data science work should be to improve the lives of individuals and groups. Next time you are designing a data dashboard or coding a complex algorithm, take a moment to consider how the data might impact human beings, and whether the decisions you make alter that impact for the better. Remember that your own biases may prevent you from being able to anticipate many impacts, so gathering input from a diverse group of team members helps you mitigate your personal biases.

Here is a hypothetical data career example to help you see how this concept may someday impact your work:

- Imagine a data professional is analyzing traffic flow data in order to help a city planner for a major metropolitan area decide where to focus on expanding roads. At first glance, using data for road construction appears to be a purely financial decision between a city and a road construction company. The professional's analysis first led them to select a more hilly, uneven terrain as the most cost-effective place for the expansion of roads. But after analyzing the data with concern for the citizens and families who will be driving on those roads in all seasons, the data professional's analysis determined the hilly area to have a higher accident risk, and decided to recommend a flatter location for road expansion.

Data science needs regulatory compliance

If you are unfamiliar with the laws and regulations of the industry you work in, your job will not only be much more difficult, but also could land you in legal trouble. A data professional should always remain up to date and in compliance with all regulations in their particular field. As an example, when you begin a new job in the data career space, you can do the following:

- Ask your manager for compliance guidance.
- Take time to research the data governance policies.
- Take time to research the regulatory body of your particular industry and its relevant policy documents.

Keep in mind that the data governance field is quite young and that regulations can and will change at a quick pace. Be sure to keep up with the changes!

It's important to follow the rules … but not ***just*** for rules' sake. Here are a few important benefits of remaining compliant with regulatory bodies and data governance policies:

- Keeps client and company data safe from security threats
- Bolsters trust with clients, peer groups, companies, and public
- Lessens likelihood of lost or mismanaged data
- Ensures business critical data remains usable, accessible, and available

Customers should choose how their data is used

With each passing day, technology gets more advanced and capable. As data becomes increasingly involved in our day-to-day lives, it stands to reason the data itself becomes more and more important and valuable. Because of this, data privacy and security is critical to a company's clients and a government's citizens.

As data professionals, it is our ethical responsibility to treat client and customer data with respect and dignity. Companies that store, analyze, and utilize data as part of their business processes should make sure those processes are transparent and compliant. Customers who divulge confidential information in order to procure a service should be able to trust that the data will remain confidential and not compromised due to breaches or cyber security threats.

You'll find that many companies have begun to treat customer data with more internal care and security. More companies are providing extensive employee training regarding the handling and securing of customer data. Many are adding multi-factor authentication systems and auditing their third-party vendors, requiring high levels of digital security systems and platforms.

As an example, because data breaches like Home Depot in 2014, Uber in 2017, and Instagram in 2020 have been commonplace for decades, corporations are increasingly ramping up cybersecurity, employee and vendor training, and transparency to customers and the public regarding data gathering and data use methods. Specifically, Home Depot has added data security into its risk management plan, and built a data security web page that describes what data they gather and how they use it.

Data transparency is not only good for business, it is the right thing to do—the ethical thing to do. Data professionals need to understand that more than anyone.

Key Takeaways

Data ethics is an expanding field and remains an integral part of a data professional's daily work. While manipulating, analyzing, or using data in any capacity, remember that data models have lives beyond code blocks and data strings. Data science needs regulatory compliance and customers should always get to choose how their data is used.

Resources for more information

Data governance and data ethics are a growing field of work and study. Here are some resources should you like to learn more:

- Harvard Business School: 5 principles of Data Ethics for Business
- Dataversity: Data Governance and Data Quality

How to handle outliers

Previously, you watched two videos about how to detect outliers and why handling outliers can be an important part of data cleaning. At this point, you likely have a good understanding of this. It is important to not only detect outliers, but also to have a plan for them.

That is precisely what you will review in this reading. Once you've detected outliers in your dataset—whether global, contextual, or collective—how do you handle them? When it comes to exploratory data analysis, or EDA, there are essentially three main ways to handle outliers: delete, reassign, or leave them in.

Whether you keep outliers as they are, delete them, or reassign values is a decision that you make on a dataset-by-dataset basis. To help you make the decision, you can start with these general guidelines:

- **Delete them:** If you are sure the outliers are mistakes, typos, or errors and the dataset will be used for modeling or machine learning, then you are more likely to decide to delete outliers. Of the three choices, you'll use this one the least.

- **Reassign them:** If the dataset is small and/or the data will be used for modeling or machine learning, you are more likely to choose a path of deriving new values to replace the outlier values.

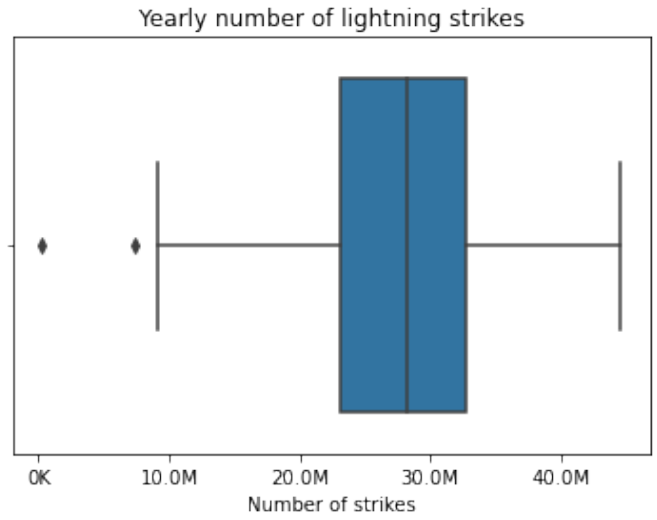
- **Leave them:** For a dataset that you plan to do EDA/analysis on and nothing else, or for a dataset you are preparing for a model that is resistant to outliers, it is most likely that you are going to leave them in.

The videos discussing outliers went into detail on how to handle outliers when you leave them in the dataset. In this reading, you will learn about some techniques for deleting and reassigning outliers.

1. Delete them

For one way to delete outlier values, recall the coding you saw in the walkthrough video on outliers. In that video, the instructor coded a box plot to help you visualize two different outliers, as shown here:

```
box = sns.boxplot(x=df['number_of_strikes'])
g = plt.gca()
box.set_xticklabels(np.array([readable_numbers(x) for x in g.get_xticks()]))
plt.xlabel('Number of strikes')
plt.title('Yearly number of lightning strikes');
```



The instructor then used the following code to find the lower limit—8.6M.

```
# Calculate 25th percentile of annual strikes
percentile25 = df['number_of_strikes'].quantile(0.25)

# Calculate 75th percentile of annual strikes
percentile75 = df['number_of_strikes'].quantile(0.75)

# Calculate interquartile range
iqr = percentile75 - percentile25

# Calculate upper and lower thresholds for outliers
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
```

```
print('Lower limit is: ', lower_limit)
print(upper_limit)
```

```
Lower limit is:  8585016.625
47356671.625
```

Next, a Boolean mask was used to filter the dataframe so it only contained rows where the number of strikes was less than the lower limit.

```
print(df[df['number_of_strikes'] < lower_limit])
```

```
   number_of_strikes  year
1             209166  2019
33           7378836  1987
```

Once you know the cutoff points for outliers, if you want to delete them, you can use a Boolean mask to select all rows such that: lower limit ≤ values ≤ upper limit.

```
mask = (df['number_of_strikes'] >= lower_limit) & (df['number_of_strikes']
        <=upper_limit)
df = df[mask].copy()
print(df)
```

```
number_of_strikes  year
0            15620068  2020
2            44600989  2018
3            35095195  2017
4            41582229  2016
5            37894191  2015
6            34919173  2014
7            27600898  2013
8            28807552  2012
9            31392058  2011
10           29068965  2010
11           30100585  2009
12           29790934  2008
13           30529064  2007
14           33292382  2006
15           38168699  2005
```

16	40023951	2004
17	39092327	2003
18	29916767	2002
19	25470095	2001
20	26276135	2000
21	27758681	1999
22	28802221	1998
23	26986915	1997
24	26190094	1996
25	22763540	1995
26	25094010	1994
27	24206929	1993
28	16371876	1992
29	16900934	1991
30	15839052	1990
31	14245186	1989
32	9150440	1988

Next, you'll consider reassigning outliers by deriving new values that are a better fit for the dataset.

2. Reassign them

Instead of deleting outliers, you can always reassign them, that is, change the values to ones that fit within the general distribution of the dataset. There are two common ways to do this, but many different ways can be used, depending on your use case:

1. Create a floor and ceiling at a quantile: For example, you could place walls at the 90th and 10th percentile of the distribution of data values. Any value above the 90% mark or below the 10% mark are changed to fit within the walls you set. Here is an example of what that code might look like:

```
tenth_percentile = np.percentile(df['number_of_strikes'], 10)
ninetieth_percentile = np.percentile(df['number_of_strikes'], 90)
df['number_of_strikes'] = df['number_of_strikes'].apply(lambda x: (
        tenth_percentile if x < tenth_percentile
```

```
        else ninetieth_percentile if x > ninetieth_percentile
        else x))
```

```
0     15620068.0
1     14657650.6
2     38815238.6
3     35095195.0
4     38815238.6
5     37894191.0
6     34919173.0
7     27600898.0
8     28807552.0
9     31392058.0
10    29068965.0
11    30100585.0
12    29790934.0
13    30529064.0
14    33292382.0
15    38168699.0
16    38815238.6
17    38815238.6
18    29916767.0
19    25470095.0
20    26276135.0
21    27758681.0
22    28802221.0
23    26986915.0
24    26190094.0
25    22763540.0
26    25094010.0
27    24206929.0
28    16371876.0
29    16900934.0
```

```
30    15839052.0
31    14657650.6
32    14657650.6
33    14657650.6
Name: number_of_strikes, dtype: float64
```

2. Impute the average: In some cases, it might be best to reassign all outlier values to match the median or mean value. This will ensure that your median and distribution are based solely on the non-outlier values, leaving the original outliers excluded. The actual imputation or reassigning of values can be pretty simple if you've already found the outliers. The following code block calculates the median of the values greater than the lower limit. Then it imputes the median where values are lower than the lower limit.

```
median = np.median(df['number_of_strikes'][df['number_of_strikes'] >=
        lower_limit])

df['number_of_strikes'] = np.where(df['number_of_strikes'] < lower_limit,
        median, df['number_of_strikes'] )
```

Note: Outside of EDA, machine learning and regression modeling have more complex variations on dealing with outliers. You will learn more about those topics later.

Key Takeaways

After detecting the outliers in a dataset, it is essential that you determine a strategy for how to handle them. Because every dataset and data-based problem is different, your strategy will vary. For the most part, you will be choosing between deleting, reassigning, or leaving outliers.

Categorical data: Data that is divided into a limited number of qualitative groups

Collective outliers: A group of abnormal points, following similar patterns and isolated from the rest of the population

Contextual outliers: Normal data points under certain conditions but become anomalies under most other conditions

Data ethics: Well-founded standards of right and wrong that dictate how data is collected, shared, and used

Data governance: A process for ensuring the formal management of a company's data assets

Deduplication: The elimination or removal of matching data values in a dataset

Docstring: (Refer to **documentation string**)

Documentation string: A group of text that explains what a method or function does; also referred to as a "docstring"

Dummy variables: Variables with values of 0 or 1 that indicate the presence or absence of something

Global outliers: Values that are completely different from the overall data group and have no association with any other outliers

Heatmap: A type of data visualization that depicts the magnitude of an instance or set of values based on two colors

Input validation: The practice of thoroughly analyzing and double-checking to make sure data is complete, error-free, and high quality

Joining: The process of augmenting data by adding values from other datasets; one of the six practices of EDA

Label encoding: Data transformation technique where each category is assigned a unique number instead of a qualitative value

Missing data: A data value that is not stored for a variable in the observation of interest

Non-null count: The total number of data entries for a data column that are not blank

One-hot encoding: A data transformation technique that turns one categorical variable into several binary variables

Outliers: Observations that are an abnormal distance from other values or an overall pattern in a data population

Tableau Public overview

As you have been learning, Tableau is a powerful data visualization tool used by data professionals around the world. If you have taken the Google Data Analytics Professional Certificate, then you should already be familiar with Tableau. If you haven't completed the Data Analytics Certificate, you can review resource materials below and linked in other videos. The Tableau software is available for free through its browser version, which allows learners like you to test out the capabilities of the software in a limited capacity. In this reading, you will be given an overview of Tableau Public, the free use, basic version of this visualization software.

Reviewing the fundamentals of Tableau Public

In this reading, you will learn about the basic structure of the **data source** and **design** screens featured in Tableau Public. The data source page is used for inputting or connecting to the data, and the design page is used for plotting and creating data visualizations. Both are needed to successfully design impactful and compelling data visualizations.

Note: To review the Tableau Public setup process, refer to the reading about how to sign on to Tableau Public.

Data source page

Before you can start designing visualizations, you'll first need to upload your data. Since you've already set up your Tableau Public profile, all you need to do is log in and select **Web Authoring** under **Create** in the navigation bar.

Note: Everything required for Tableau in this course can be completed with Web Authoring; you are *not* required to download the Tableau software.

Tableau Public Web Authoring

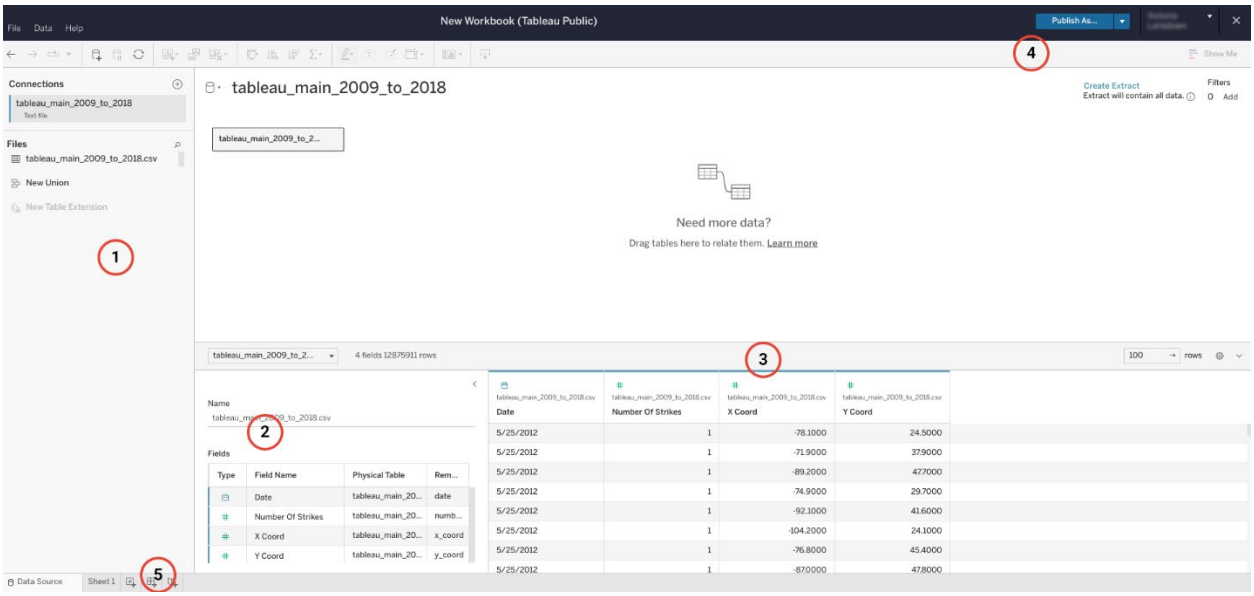
Web authoring allows you to create visualizations directly from a web browser. Can you create a viz without downloading any software? Yes! Since you've already set up your Tableau Public profile, all you need to do is log in and select **Web Authoring** under **Create** in the navigation bar. For the purposes of this certificate program, you can perform everything you need in Tableau Public. The instructions in the following resources refer to Tableau Public.

Tableau Desktop Public Edition

You can also download the software directly to your Mac or PC. Select **Tableau Desktop Public Edition** under **Create** in the navigation bar on Public's website.

Reminder: Tableau Public should only be used for analyzing and sharing public data. *All workbooks and datasets published will be freely accessible to anyone.*

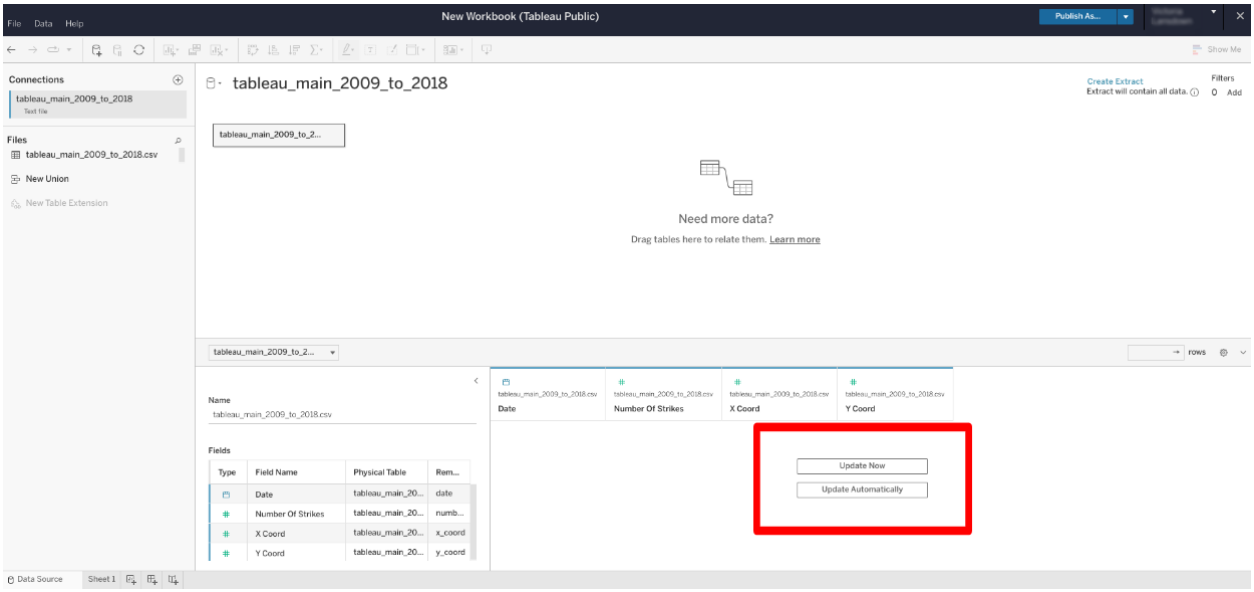
After you upload your dataset, you can perform the following steps outlined to match the circled numbers in the following image:



The following descriptions correspond to the image above.

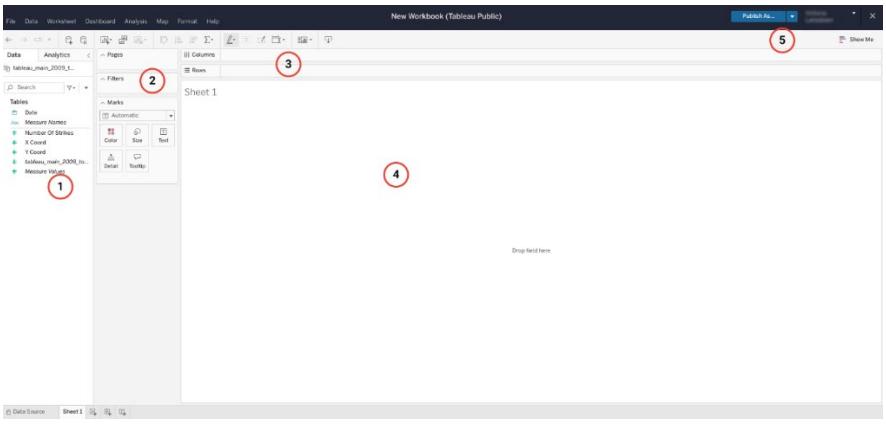
1. This left-hand pane includes your data connections and files. Here you will find all the files you upload in a list so that you can keep track of multiple files and/or multiple connections to different databases.
2. Just to the right of the data connections window is a list of all of the fields that Tableau Public has detected in a particular file. If you have multiple files uploaded, you can select the file from a dropdown to access each file's fields. As you'll learn in an upcoming video, Tableau's fields are acquired from the data columns in your file. Tableau automatically sorts these fields into dimensions or measures and discrete or continuous variables.
3. The biggest pane on the page, on the middle right, allows you to access all of the columns of your file as Tableau fields, including several rows of data. Unlike the pane to the left, this pane allows you to create new fields based on those already present, such as new calculation fields, groups, sets, or parameters (you will learn more about these features in upcoming videos). You may be prompted to select "update now" or "update automatically" in order to populate this pane with your data. If that's the case, it is good practice to update automatically to ensure you're consistently working with recent data. (For reference -- review the following image, after #5.)
4. The blue button "Publish" in the upper right of the screen acts as your "save" button. Because Tableau Public is a browser-based platform, anything you create and want to save will be published to your public account. There are ways to password lock or hide data sources and data visualizations if desired, but Tableau Public only offers the Publish field for saving your work. When you click on the 'Publish' button, you may automatically be navigated to your data design page, which may be blank depending on your design progress. Do not be alarmed. Your latest dataset uploads or data designs were still saved; just navigate back to where you were last and continue editing your visualization.

5. Lastly, you will use the collection of buttons at the bottom left of the page to navigate to your data design page. You'll find button options for creating a new worksheet, a new dashboard, and a new story. These elements will be introduced in the next section.



Data design page

The data design page is where your data visualizations will be built. To navigate to the data design page, click on 'Sheet 1' or create a new sheet as instructed in #5 on the previous corresponding image. When you first click to open a data design page, you may be prompted that Tableau is 'Creating Extract'. That means that Tableau is extracting the providing data to be used in visualizations. This process may take several minutes. Here you will move your data source fields to appropriate shelves to build the type of visualization you want. You can build data visualizations or entire interactive dashboards from this page.



The following numbered items correspond to the numbers displayed in the Tableau workbook image above.

1. In this pane on the far left, you will find your list of discrete and continuous dimensions and measures. You will move these variables to different panes on this page to build visualizations. You will learn more about these variables later.
2. In the next pane just to the right, you'll find "Pages," "Filters," and "Marks." You can move any dimension or measure to these different fields to manipulate the data visualization. You will learn how to use these features in upcoming videos.
3. At the top of the page, just under the menu bar, there are two empty rows that act as your main two shelves for moving your variable fields. The "Columns" and "Rows" shelves help you position your data visualization as desired. You'll also notice above these rows a toolbar and menu full of other options for manipulating your data visualization.
4. In the middle of the screen is the main viewing panel for your visualization. As you add elements and drag your dimensions and measures to different fields, you will notice the impact they have on your data visualization in this panel. In the upper right corner you will find your "Publish" button, which acts as the save button, and the "Show Me" dropdown. Under the "Show Me" dropdown, you will find a selection of data visualization types and guides for building each of them.
5. When you're ready to save and share your work, publish it to your Tableau Public profile. View the options for publishing your work by clicking on the down arrow next to the "Publish" button on the top navigation bar.

Tableau.com

By visiting Tableau.com, you'll notice multiple product offerings, everything from Tableau Public (which is free) to Tableau Desktop, Tableau Mobile, and Tableau Server. Each product has its own use and specialization, but the main elements for data visualization are the same. You can search through the Tableau Help page to find specific articles on just about any topic regarding data visualizations. There are a variety of training resources available from Tableau to help users learn the different features of their products. Tableau offers a variety of training resources for helping to learn their different products.

Key takeaways

Tableau is a powerful data visualization tool, but that means it takes a lot of practice and experience to use it proficiently. The two main pages you'll use are the data source and data design pages. There are also a large number of resources available to help you in each step of the process, including Tableau Help and Grow With Google Data Analytics Certificate Program.

Resources for more information

To help you troubleshoot or to learn more, you can use the following links:

- Use Tableau resource page to set up your data for success: Set up data sources
- Tableau Tools and Web Authoring Help: Design charts and analyze data
- The Tableau Public "Discover" page, which includes "Viz of the Day" and other beautiful vizzes designed on the platform: Welcome to Tableau Public
- Beginner's guide to using Tableau Public: A step-by-step guide to get you started on your own data viz journey

Getting Ready to Publish Your First Data Visualization:

How to sign on to Tableau Public

As you have been learning, Tableau is a powerful data visualization tool used by data professionals around the world. Tableau offers a free browser version of its software, Tableau Public, which allows learners like yourself to test out its capability in a limited capacity. Tableau Public is the software you will use throughout this course on data visualization. In this reading, you will be guided on how to sign up for Tableau Public, and you will be given troubleshooting resources for potential obstacles you may experience.

How to sign in to Tableau Public

Signing in with an existing Tableau.com account

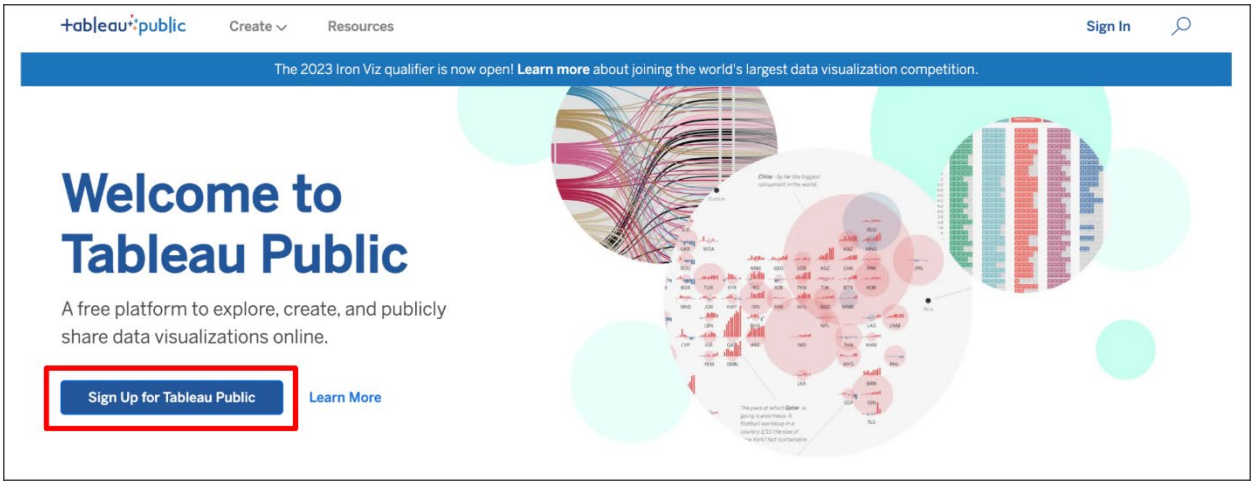
If you already have a Tableau.com account, you can use your existing login credentials to sign into Tableau Public. Click here to learn more. If this is your first time signing in to Tableau Public with your Tableau.com account, you can set your account password by resetting your password. To do so, click "reset password."

Signing in for the first time

The following instructions show you how to sign up for Tableau Public for the first time, without a pre-existing Tableau.com or Tableau Public account. First, go to the Tableau Public home page. If this link does not open to Tableau's homepage, type public.tableau.com in your browser's address bar.

Note: *Tableau Public works best on Chrome (Windows, Mac, Android), Edge (Windows), Firefox (Windows and Mac), Safari (Mac and IOS).*

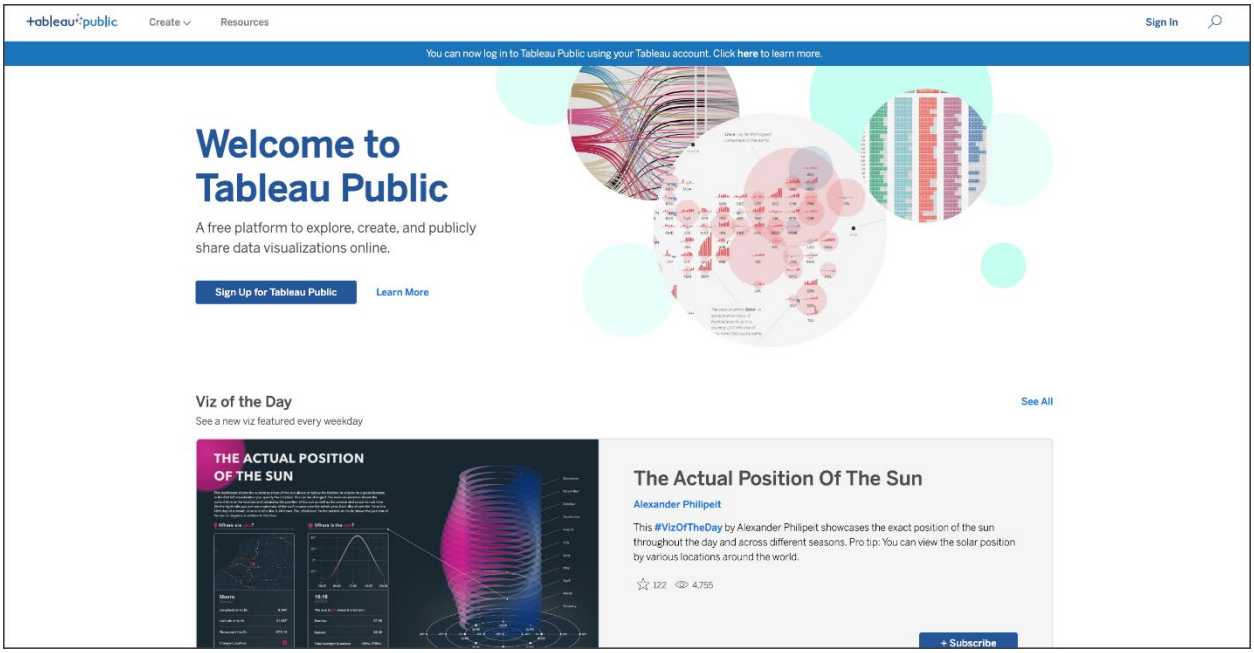
Next, at the top right of the page, click the "*Sign Up*" button.



Fill in all the required fields and click "*Create My Account*."

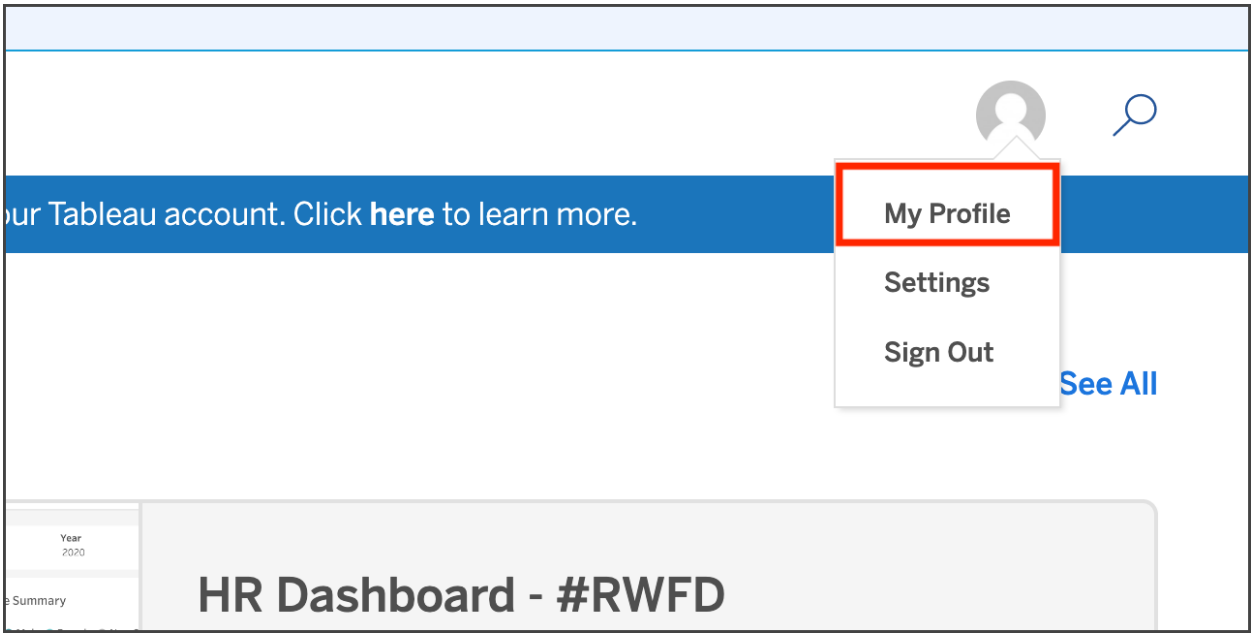
Navigating Tableau Public

The home page for Tableau Public contains resources and guides for helping data professionals learn more about and get inspired by data visualizations. You can explore any and all of the links on this landing page to help enhance your knowledge of Tableau. The most helpful place to start if you are new to Tableau is the Resources page, which has how-to videos, community resources and user forums.



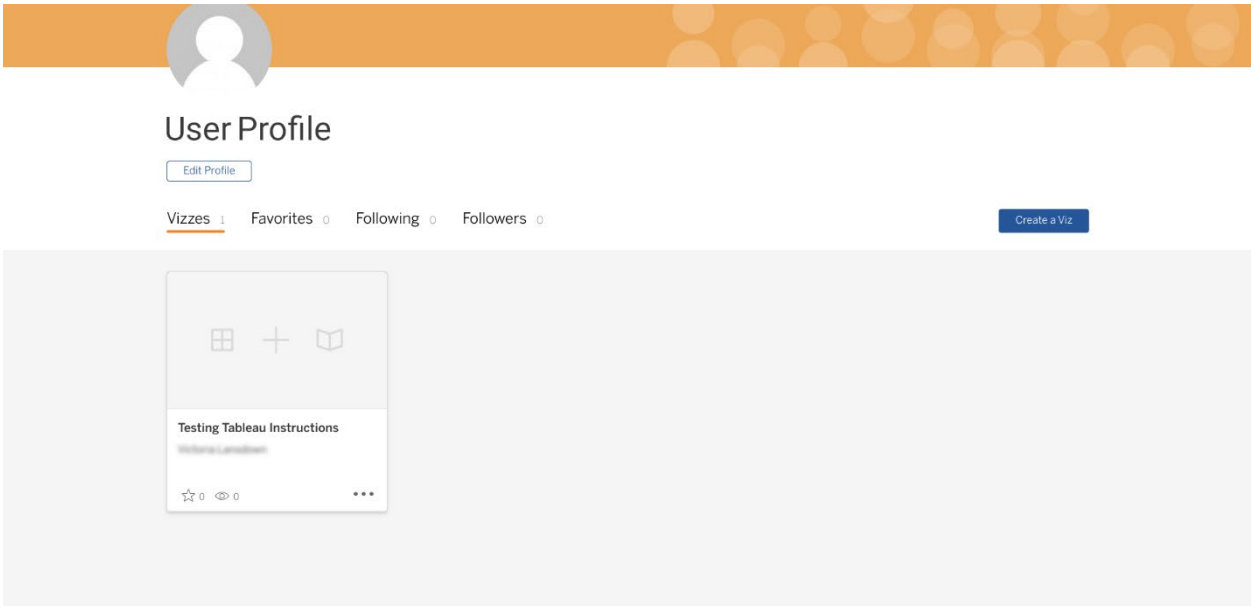
How to start creating your own visualizations

During the instruction videos, the instructor will ask you to log in to Public Tableau and follow along in the creation of a data visualization. Once you've set up your Tableau Public profile, all you need to do is log in and select Web Authoring under Create in the navigation bar.



Get Started

Within your Tableau Public profile, you'll find tabs for *Vizzes*, *Favorites*, *Following*, and *Followers*. At the instructor's prompt, click *Create a Viz*.



Key takeaways

Accessing Tableau Public and creating a profile will be your first step to learning how to design data visualizations like a data professional.

Resources for more information

To help you troubleshoot or to learn more, explore take a look at the following links:

- Tableau Public not working? Check out these Technical speculations and storage requirements
- The Tableau Public Discover page includes 'Viz of the Day' and other beautiful vizzes designed on the platform
- The Tableau Public Blog offers regular dispatches from the Tableau Public Team

Download your datasets and begin presenting with Tableau

Accessing and utilizing resources in this section

To help your learning in this part of the course, follow along with the instructor as they go through Tableau in each video.

Steps to complete:

1. Open Tableau Public.
2. Since you've already set up your Tableau Public profile, just log in and select Web Authoring under Create in the navigation bar.
3. Download the dataset files attached below and then upload them to Tableau. ***See additional instructions below.**
4. Open the Tableau follow-along guides linked below. These guides are labeled to correspond to video titles and concepts.

- Tableau follow-along guide: Work with Tableau, Part 1
- Tableau follow-along guide: Work with Tableau, Part 2
- Tableau follow-along guide: Craft compelling stories with Tableau
- Tableau follow-along guide: Present like a pro with Tableau

4. Follow along with the instructor in each video as they show the creation of the data visualizations in Tableau.

In this lesson's videos, you will need access to these datasets:

tableau_datasets

ZIP File

***Instructions for downloading the dataset files and uploading them to Tableau:**

You will need to download the provided .csv files and upload them to Tableau Public to follow along with the instructor in subsequent videos and interact with Tableau. To do so, follow these steps:

1. Download the attached .csv zip file to your device.
2. Extract the contents of the zip file once the download is finished.
    a. You'll need to upload the specific dataset files to Tableau. Do not upload the entire .zip folder.
    b. When you download the zip folder from this page, your computer will automatically download a .zip file folder. The .zip folder is automatically named with a series of letters and numbers.
    c. Open up that .zip folder, then save the individual dataset files. The two files are: tableau_main_2009_to_2018.csv and tableau_dataset.csv.

    d. Once you can see the individual dataset files, proceed to upload those to Tableau Public.
3. Upload the extracted .csv files to Tableau.
    a. Review the provided Tableau follow-along guide: Work with Tableau, Part 1, and/or follow along with the instructor in the video Work with Tableau, Part 1 for further instructions about uploading files to Tableau.
    b. There are two Tableau datasets: tableau_main_2009_to_2018.csv and tableau_dataset.csv. Be sure to check the dataset name shown on screen in the instructional video.
    c. When following along, make sure you've uploaded the same file so you can create the same visualizations.

Data dictionary

The datasets above represent lightning strike counts in the United States. They include four columns of data: latitude, longitude, date, and lightning strike counts. Each row represents a total lightning strike count on the specified date for a particular location.

Column name	Type	Description
number of strikes	int64	The total count of lightning strikes on a given day
X Coor	obj	Longitude
Y Coor	obj	Latitude
date	str	The recorded date (format: DD/MM/YYYY)

Follow-along guide: Work with Tableau, Part 1

This document includes detailed instructions for how to perform the data visualizations described in the video "Work with Tableau, Part 1."

The following guide points out areas of the video that may require adjustment. These reference guides can also serve as a set of usability reminders for you to recall when using Tableau in your future career.

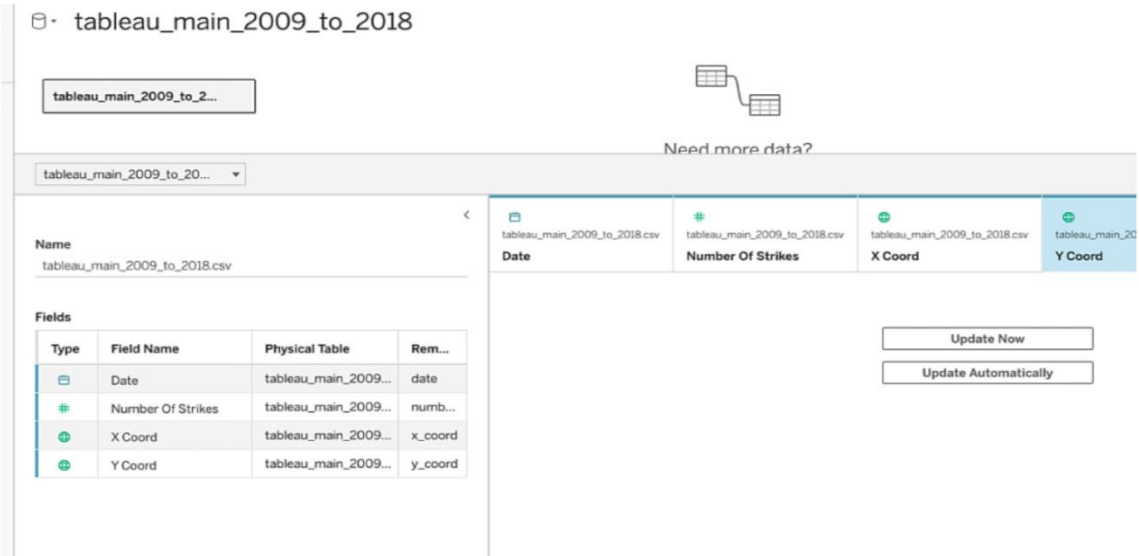
Instructions

- Since you've already set up your Tableau Public profile, all you need to do is log in and select **Web Authoring** under **Create** in the navigation bar.
- **Upload your dataset from your computer**. Select the appropriate CSV file provided in the instructions. The dataset you'll use with this instructional video is: tableau_main_2009_to_2018.csv.

(**Note:** Please allow several minutes for data upload into Tableau Public.)

Before you can start designing visualizations, you'll first need to upload your data. You'll need to upload the specific dataset files to Tableau. Do not upload the entire .zip folder. When you download the zip folder from this page, your computer will automatically download a .zip file folder. The .zip folder is automatically named with a series of letters and numbers. Open up that .zip folder, then save the individual dataset files. The two files are: tableau_main_2009_to_2018.csv and tableau_dataset.csv. Once you can see the individual dataset files, proceed to upload your dataset for this video to Tableau Public.

Notice on the data source tab that you can see all of your column headers and Tableau icons that help you determine data types. In this case, you'll see a calendar icon and pound signs indicating numbers or integers.

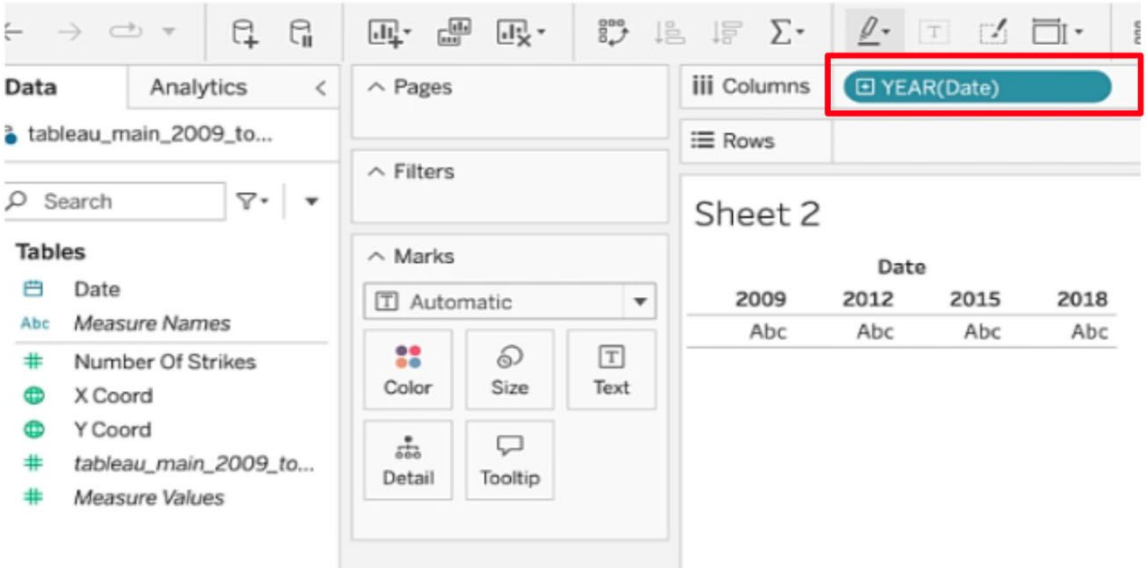


- Click on **New Worksheet**.

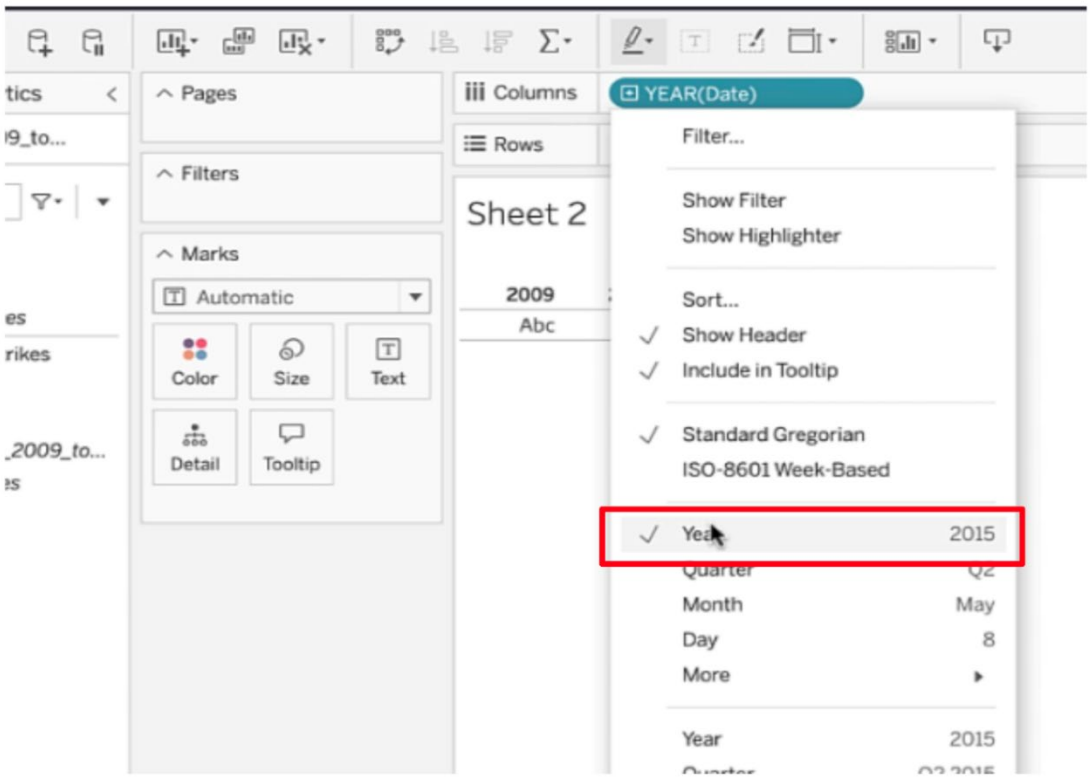
(**Note:** Please allow several minutes for data to import into a new worksheet.)

Notice the green and blue fields and the gray line dividing them. As you'll remember from the video, blue indicates a discrete field and green indicates a continuous field. The gray line divides the dimensions from the measures, with dimensions above and measures below the line.

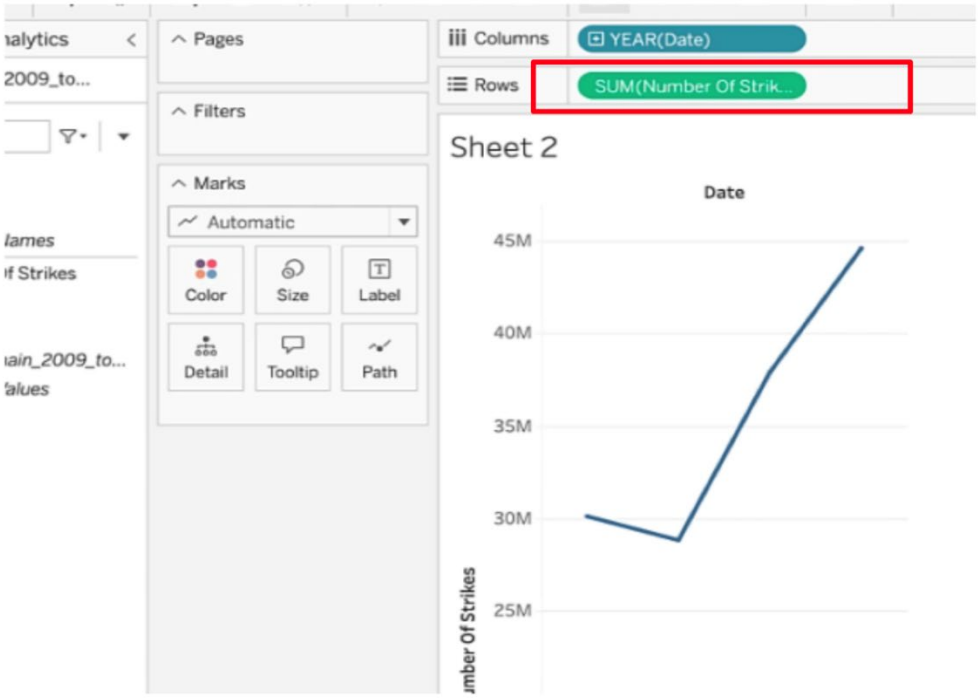
- Drag **Date** into Columns shelf.



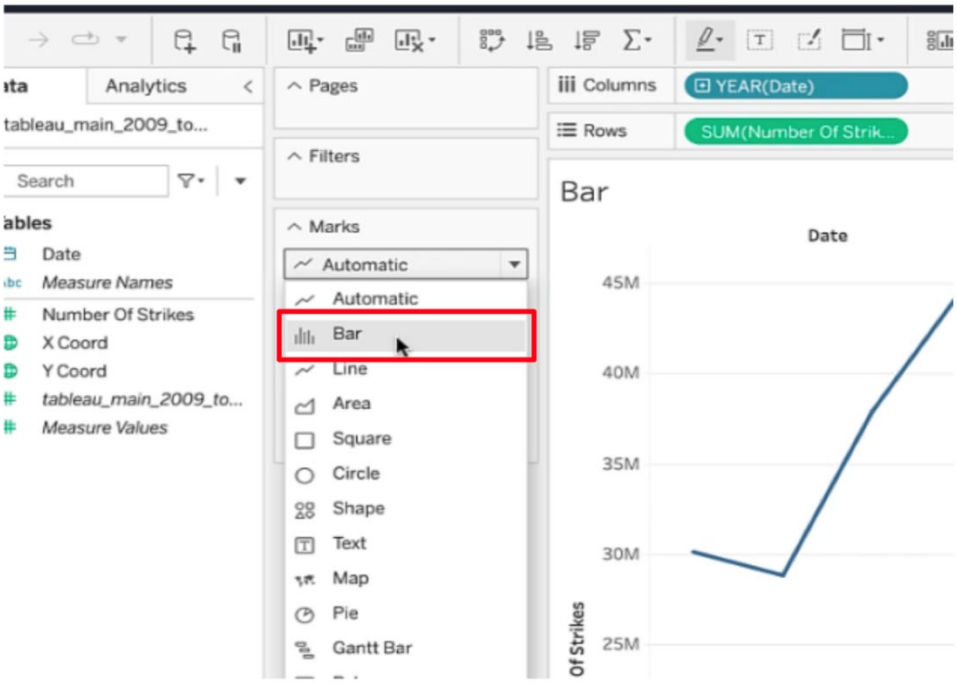
- Click on the drop down menu of the date field. Ensure there is a checkmark next to **Year**.



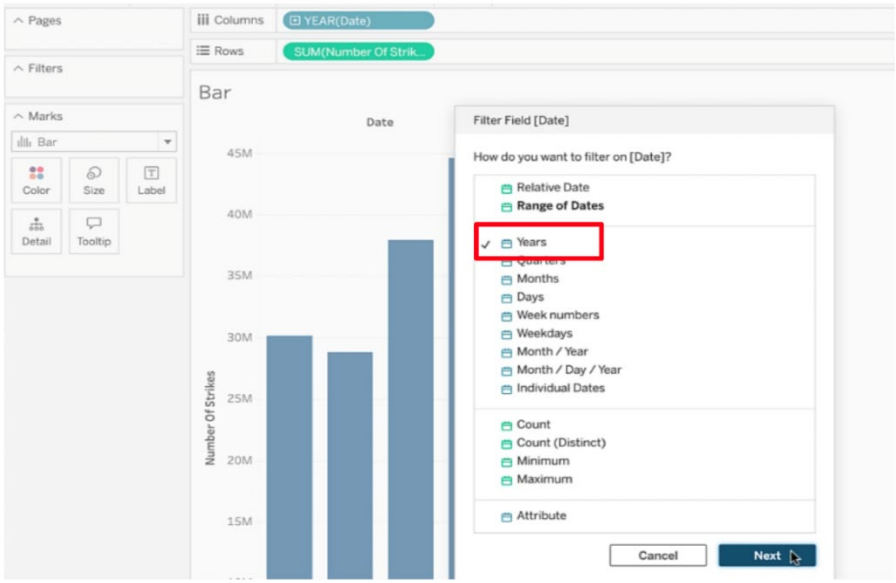
- Drag **Number Of Strikes** to the Row shelf. You've created a line graph.



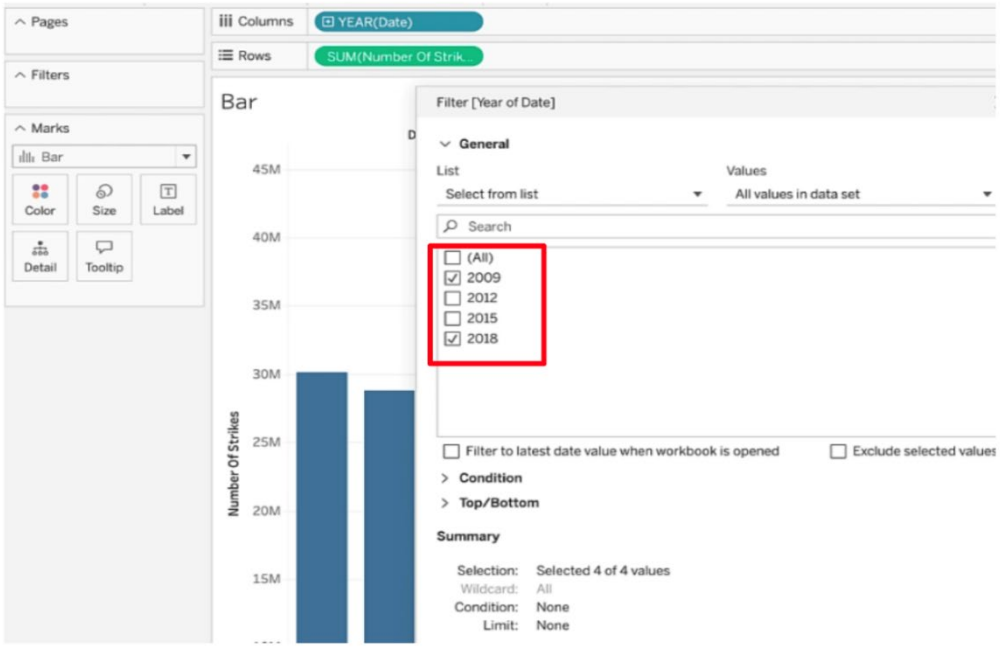
- Click on **Duplicate** worksheet in the toolbar menu.
- Click on the dropdown menu in the **Marks** field. Select **Bar**.



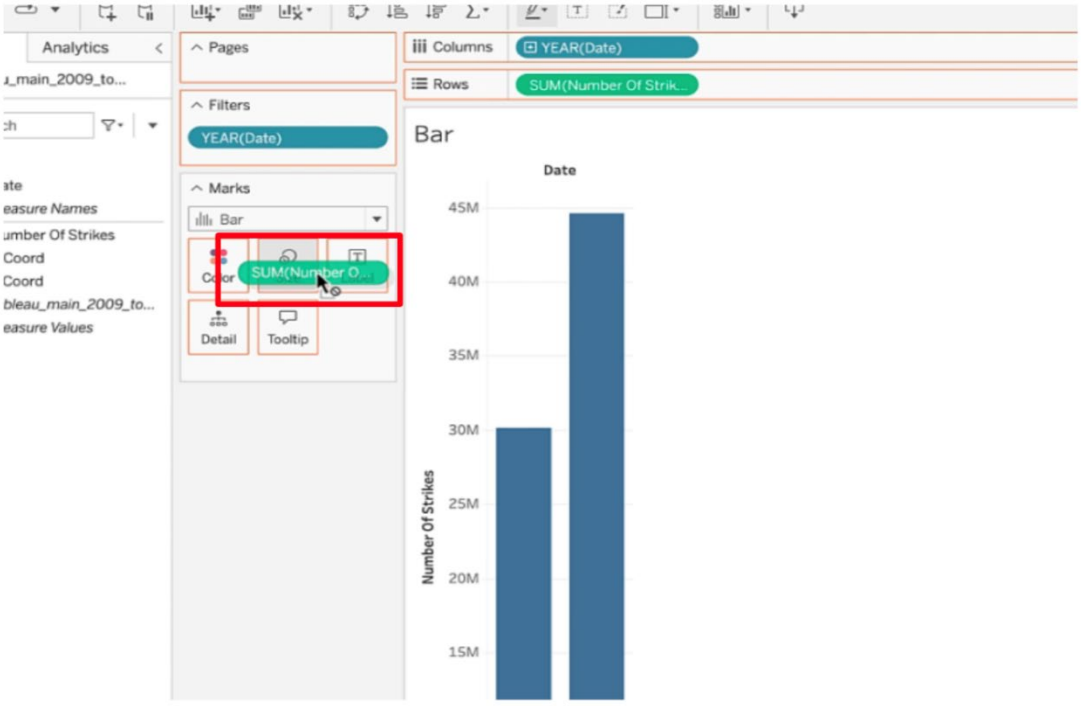
- Click **Filters**.
- Select **Years** when the pop-up window appears.
- Click **Next**.



- Select only **2009** and **2018** in the next pop-up window.
- Click **OK**.



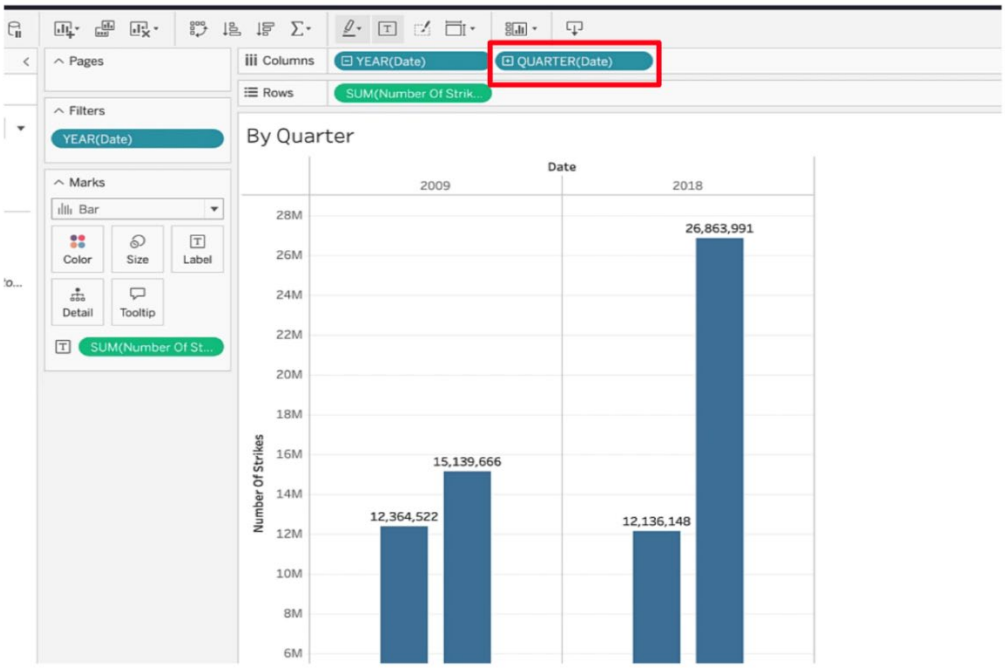
- Add labels to the bars by dragging **Number Of Strikes** to the **Label** field.



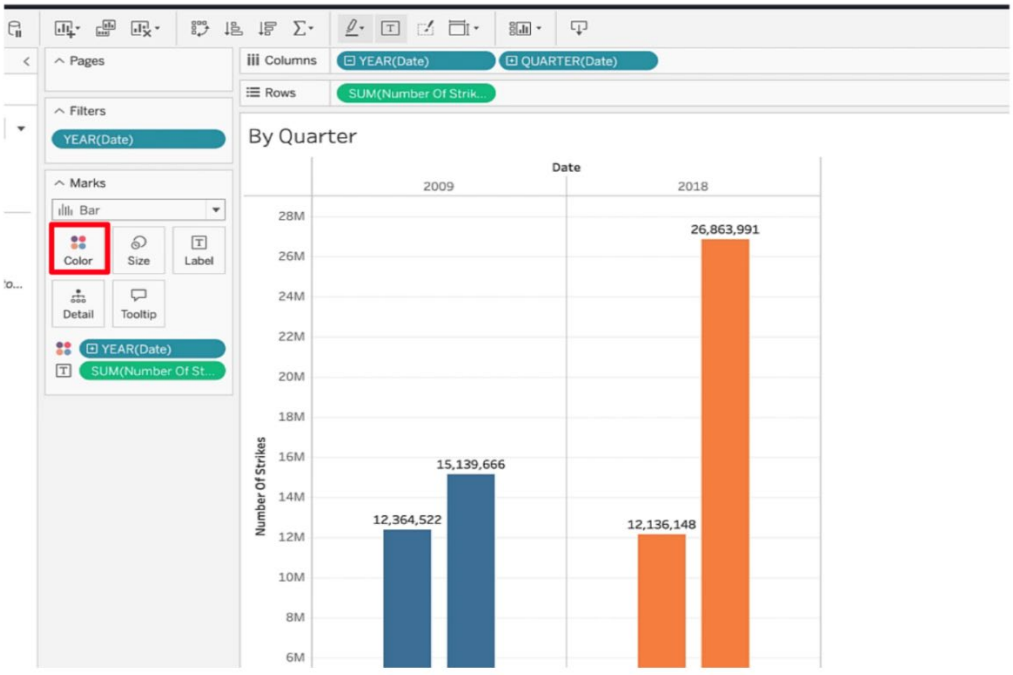
Notice the bars are labeled at the top with total number of strikes for 2009 and 2018.

- Click **Duplicate** worksheet in the toolbar menu.

- Drag **Date** to column shelf. This will automatically divide the number of strikes into quarters.



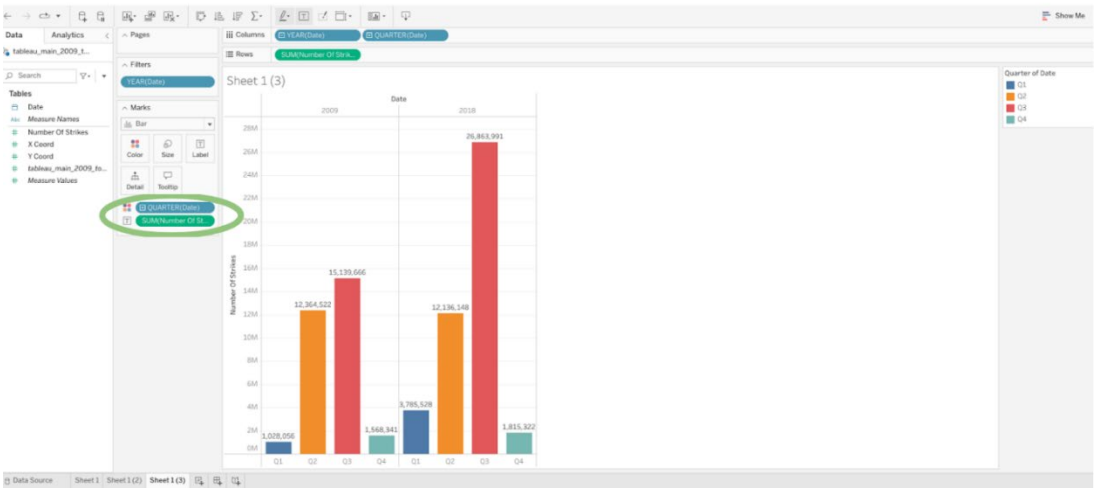
- Drag **Date** to **Color** square.



- Click on the blue drop down arrow of **YEAR(Date)** in the **Color** field and select **Quarter**.

You're done!

(**Note:** Be sure to save your work to your Tableau Public profile by clicking on "Publish.")


Sheet 1 (3)
Date
2009
2018
Number Of Strikes
1,028,056
12,364,522
15,139,666
1,668,341
3,785,528
12,136,148
26,863,991
1,815,322
Q1
Q2
Q3
Q4
Quarter of Date

Follow-along guide: Work with Tableau, Part 2

This document includes detailed instructions for how to perform the data visualizations described in the video "Work with Tableau, Part 2."

The following guide points out areas of the video that may require adjustment. These reference guides can also serve as a set of usability reminders for you to recall when using Tableau in your future career.

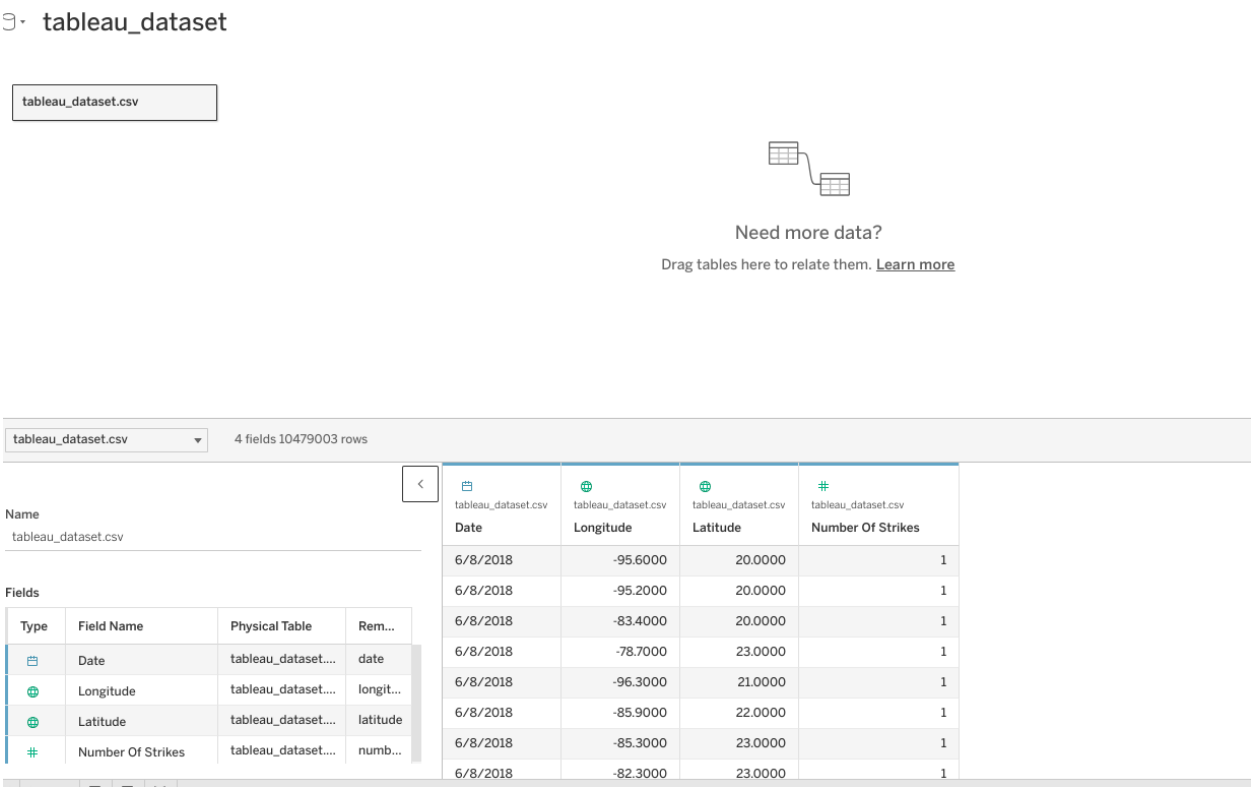
Instructions

- Go to https://public.tableau.com/s/
- Since you've already set up your Tableau Public profile, all you need to do is log in and select **Web Authoring** under **Create** in the navigation bar.
- Select the appropriate CSV file provided in the instructions. The dataset you'll use with this instructional video is: tableau_dataset.csv.
    - Please be aware that when you download the zip file folder provided, the computer automatically names that zip file folder with a long string of numbers and letters. You have to open that folder and then upload the individual files that are named correctly and match what's shown in the video.

(**Note:** Please allow several minutes for dataset upload.)

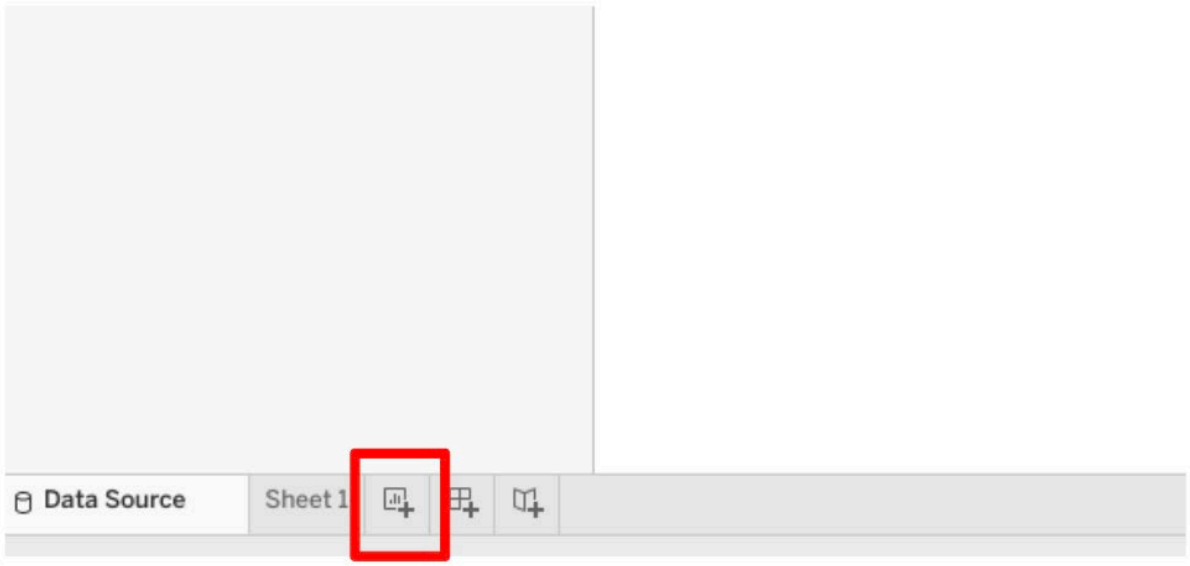
Before you can start designing visualizations, you'll first need to upload your data. You'll need to upload the specific dataset files to Tableau. Do not upload the entire .zip folder. When you download the zip folder from this page, your computer will automatically download a .zip file folder. The .zip folder is automatically named with a series of letters and numbers. Open that .zip folder, then save the individual dataset files. The two files are: tableau_main_2009_to_2018.csv and tableau_dataset.csv. Once you can see the individual dataset files, proceed to upload your dataset for this video to Tableau Public.

Notice on the data source tab that you can see all of your column headers and Tableau icons that help you determine data types. In this case, you'll see a calendar icon, globe icons, and pound signs.



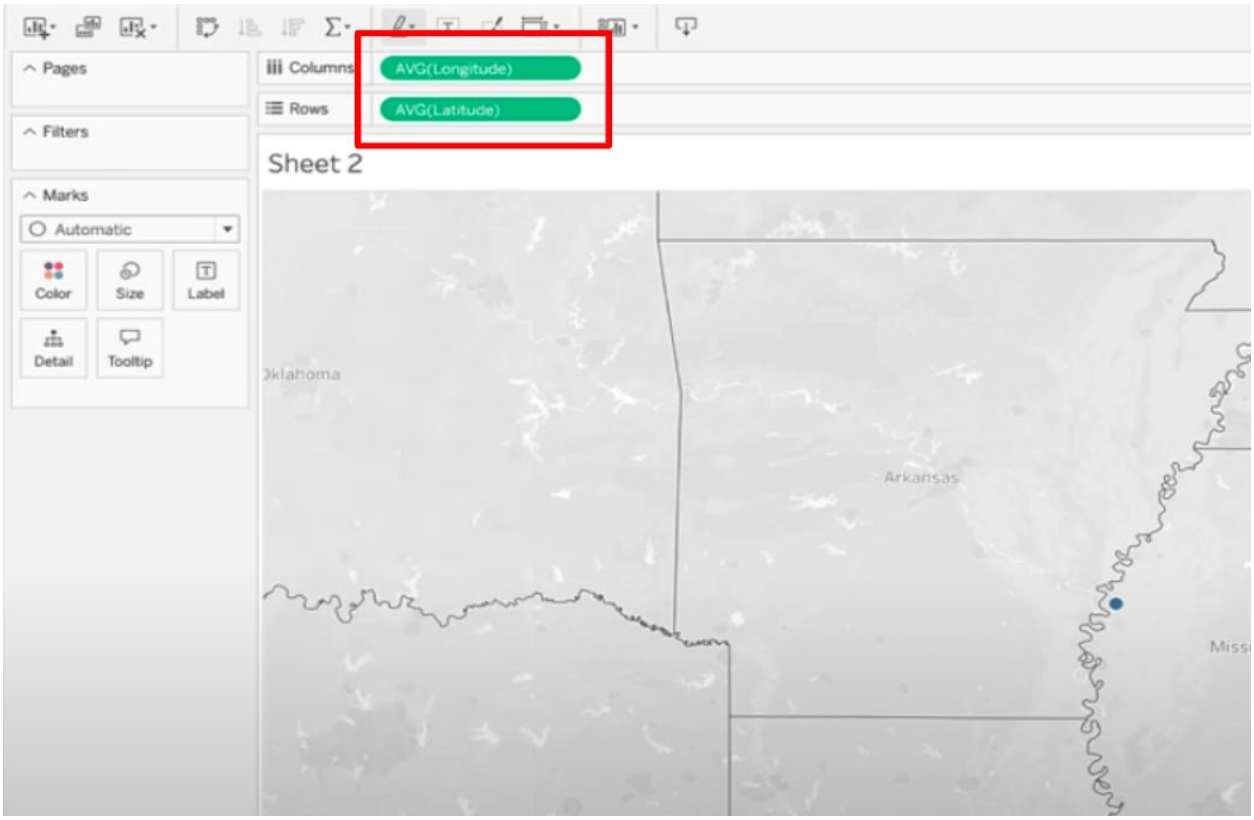
- Click on NEW WORKSHEET.

(Note: Please allow several minutes for data import into a new worksheet)

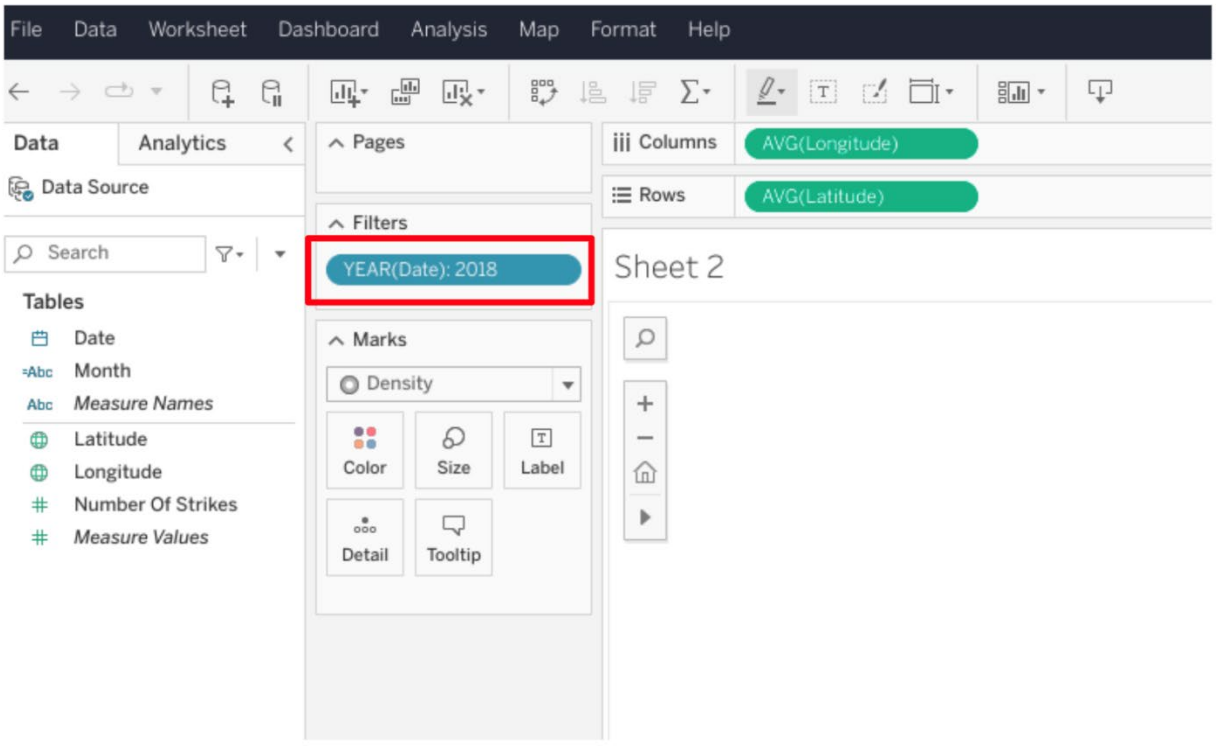


- Drag LONGITUDE into the column field. Drag LATITUDE into the rows field.

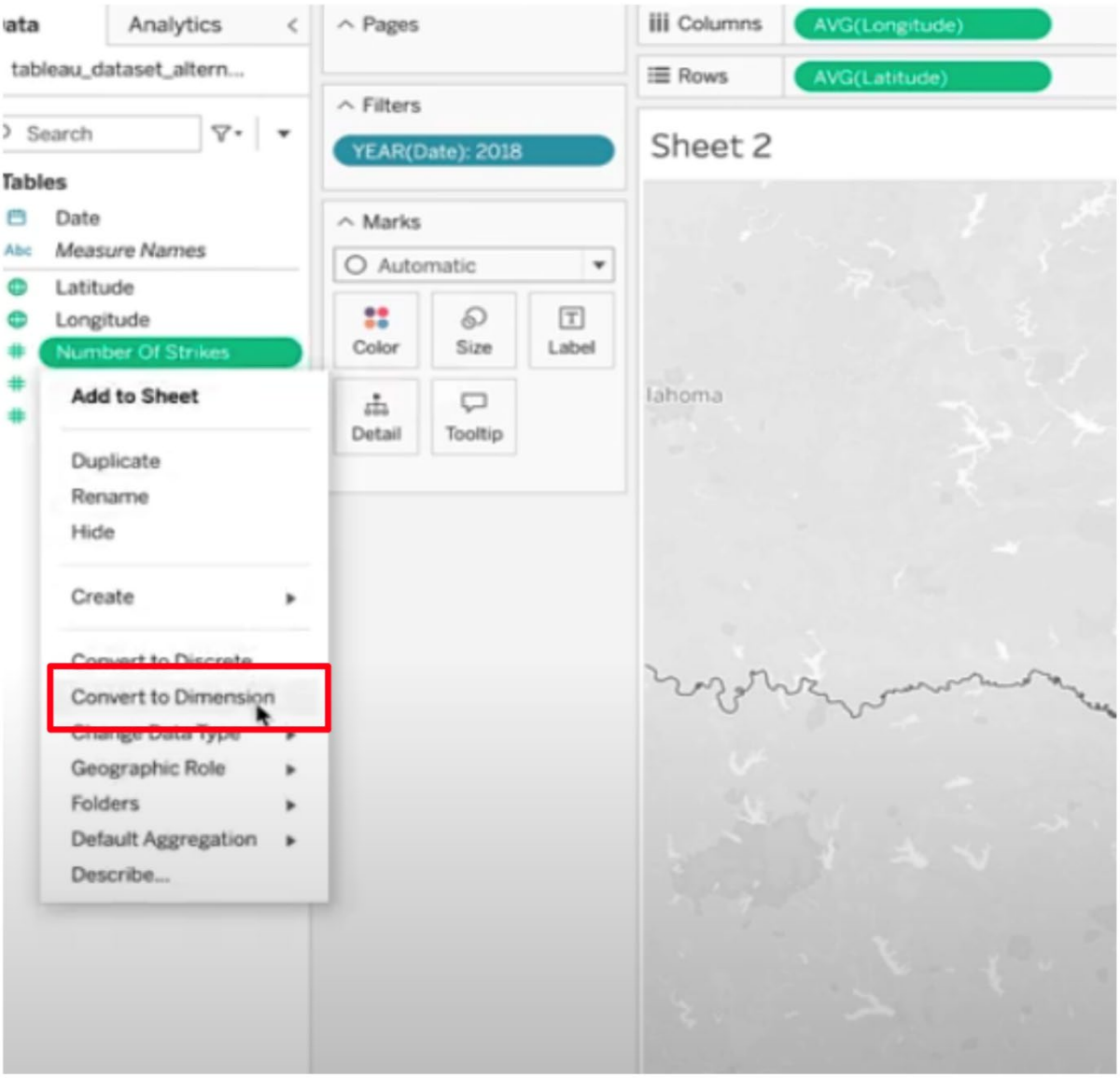
(**Note:** Make sure the latitude and longitude fields are set to continuous dimensions)



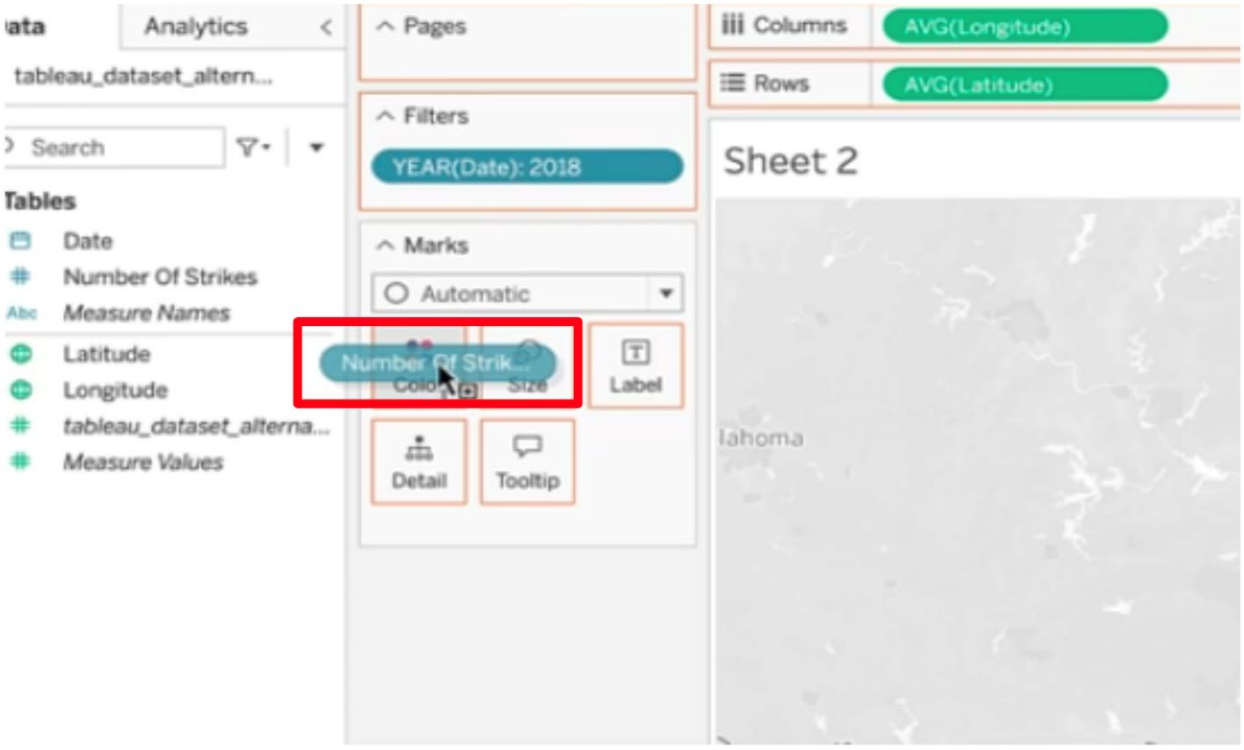
- Drag DATE into Filters field. Filter to only 2018.



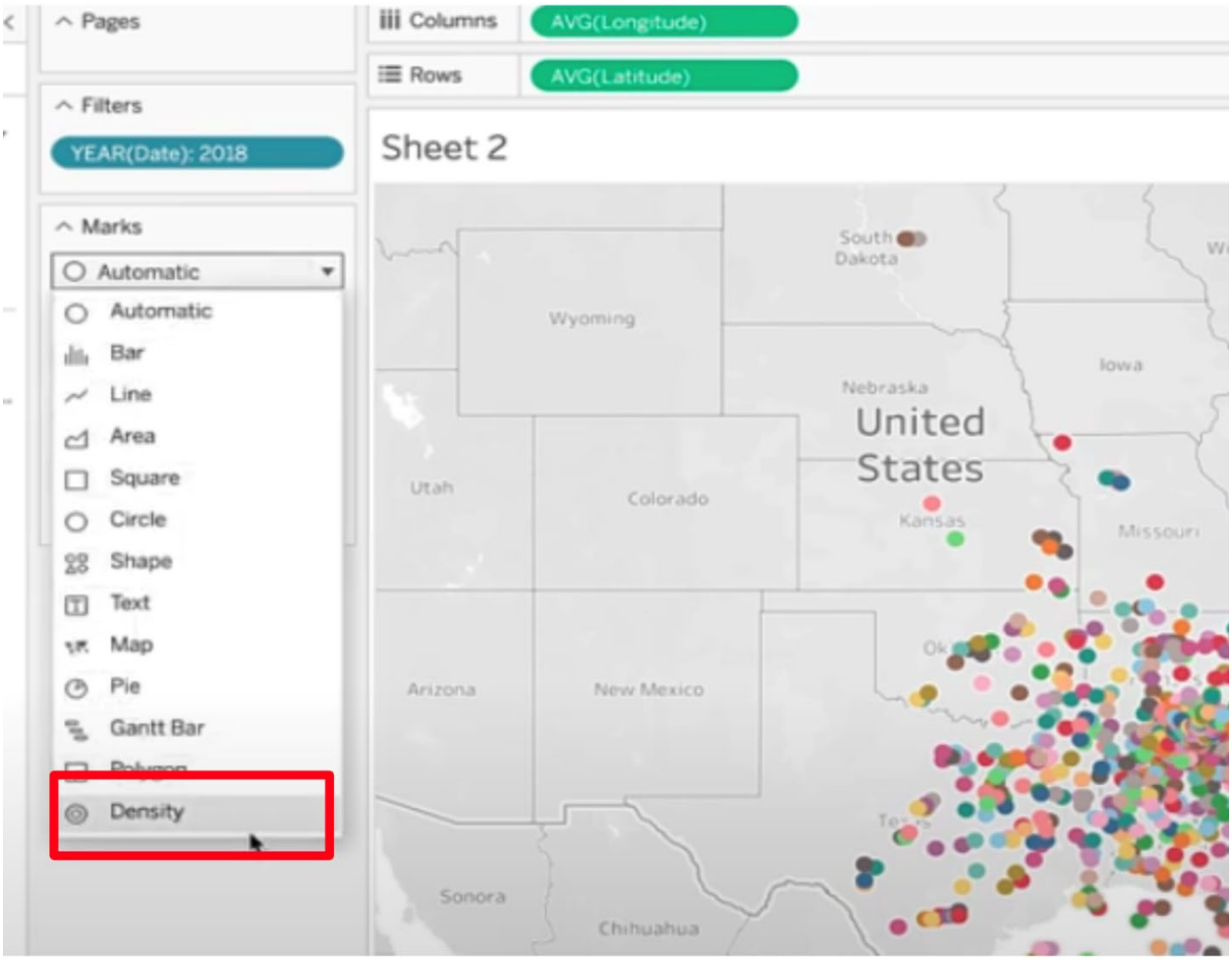
- Click on NUMBER OF STRIKES dropdown.
- Select “Convert to Dimension.”



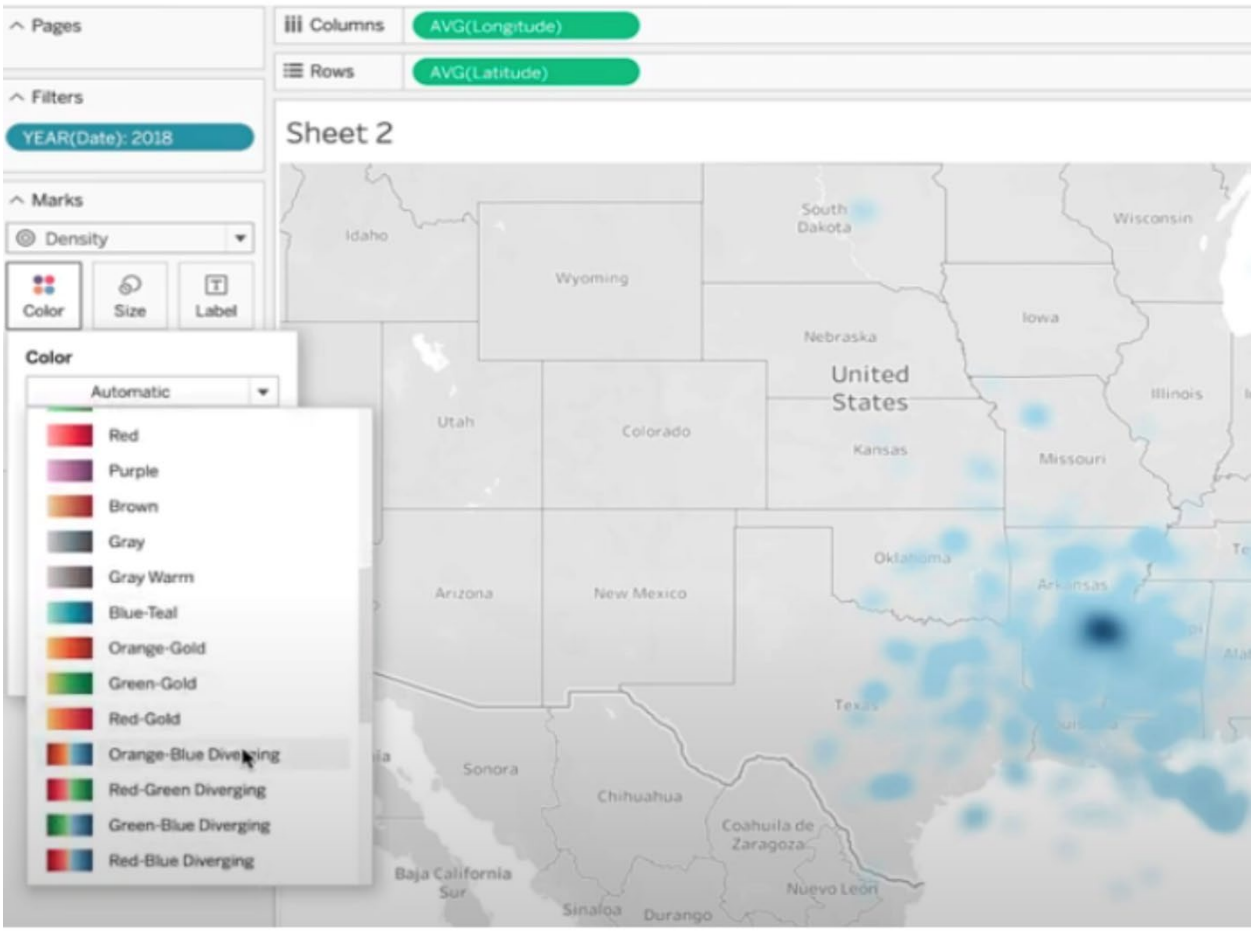
- Drag NUMBER of STRIKES to the box labeled COLOR.



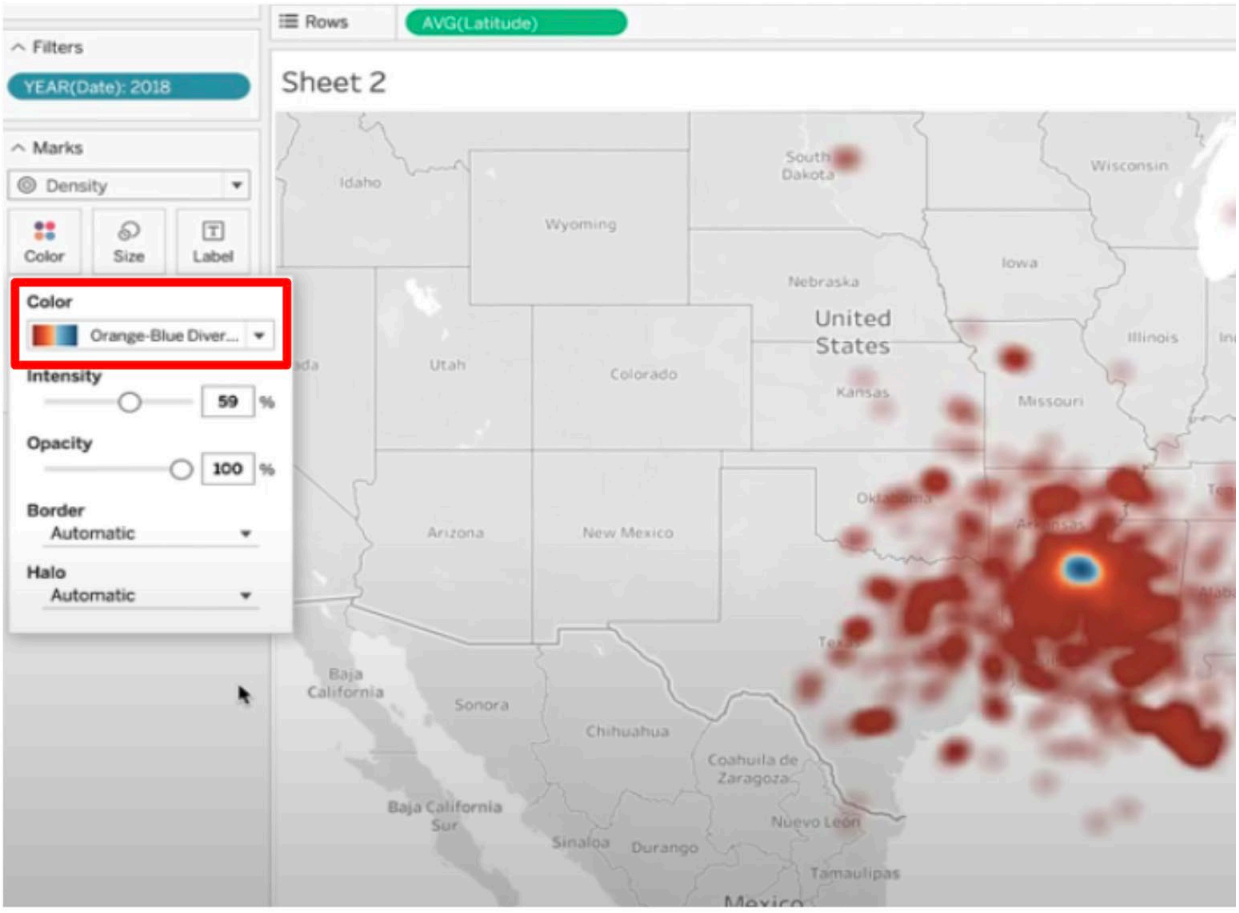
- Select Density in MARKS dropdown.



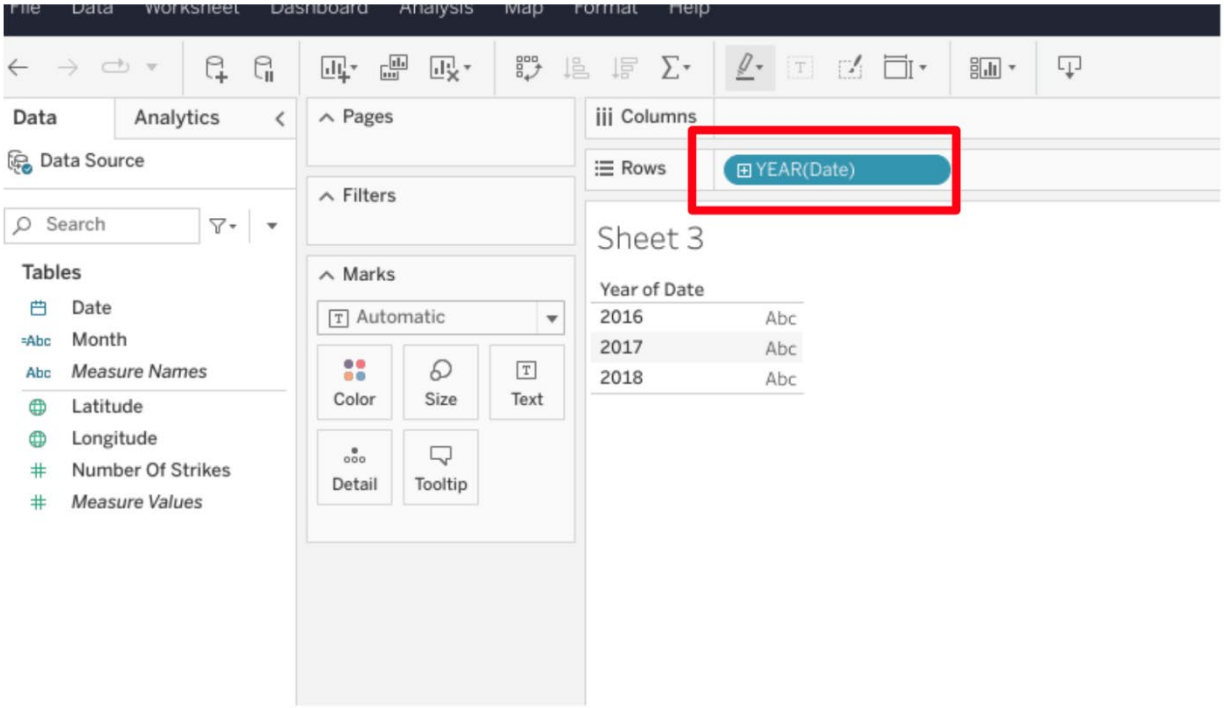
- Click on color and show the dropdown. Select any color you'd like.



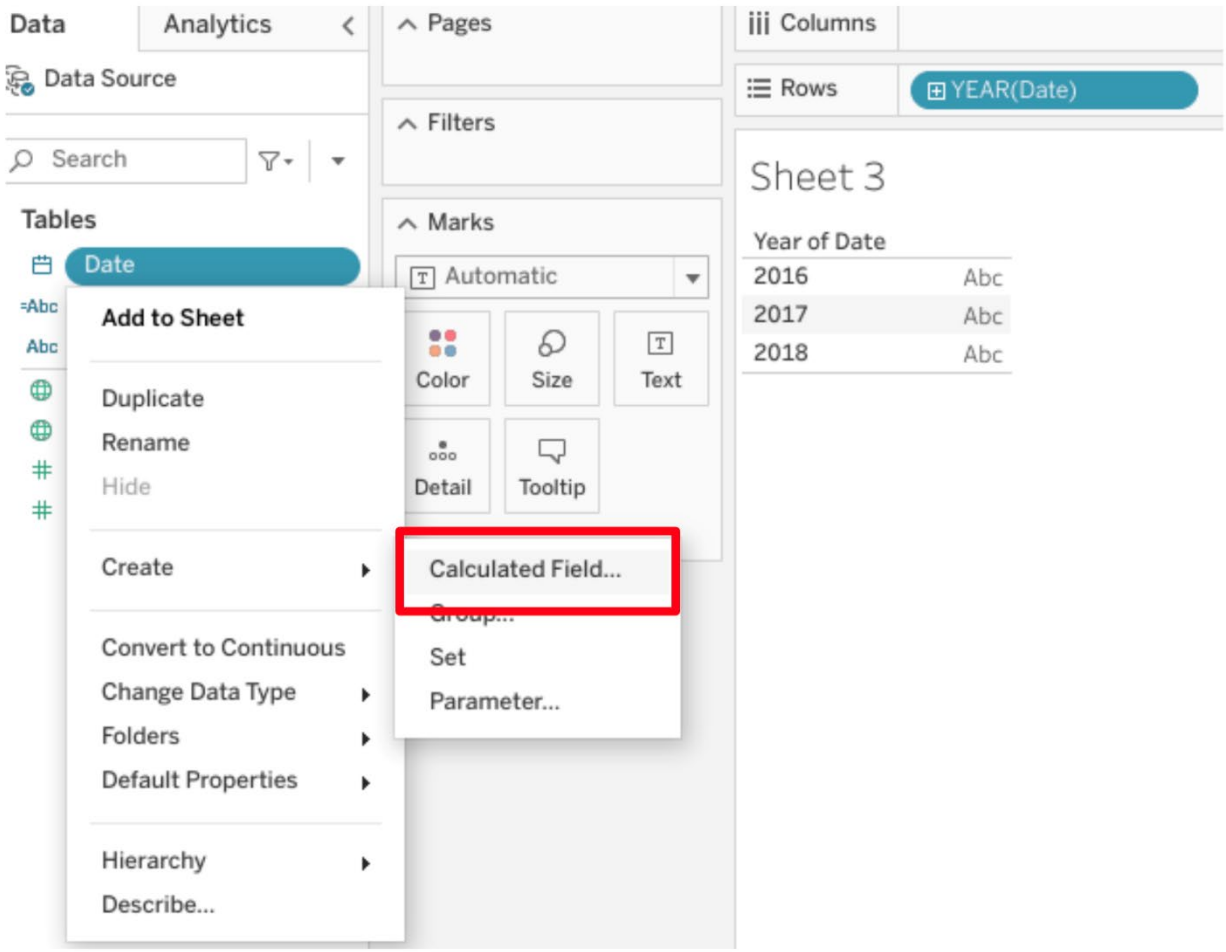
- And here is your geographic map of the location of lightning strikes in the U.S. in 2018.



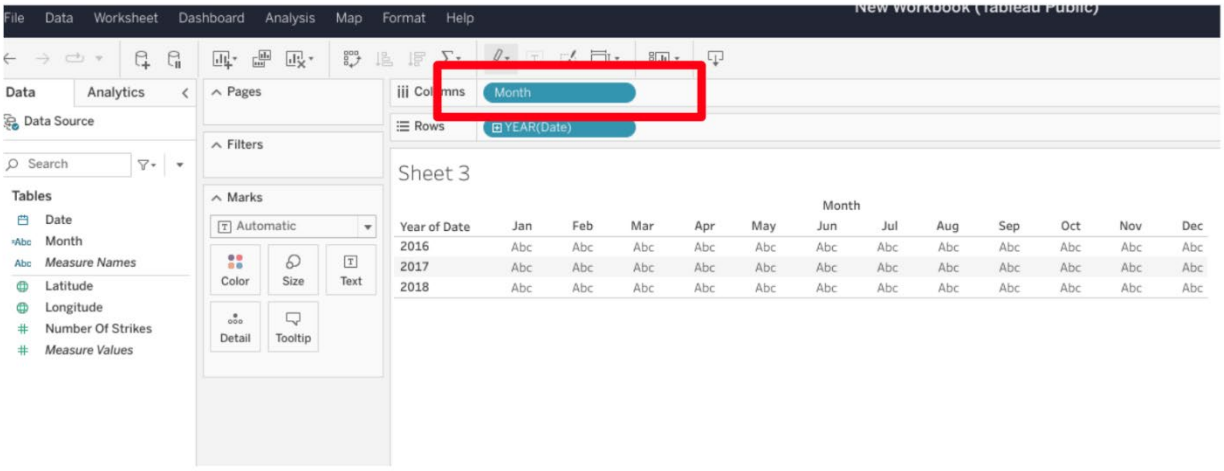
- Click on “new worksheet.”
- Drag DATE to the ROW field and select YEAR.



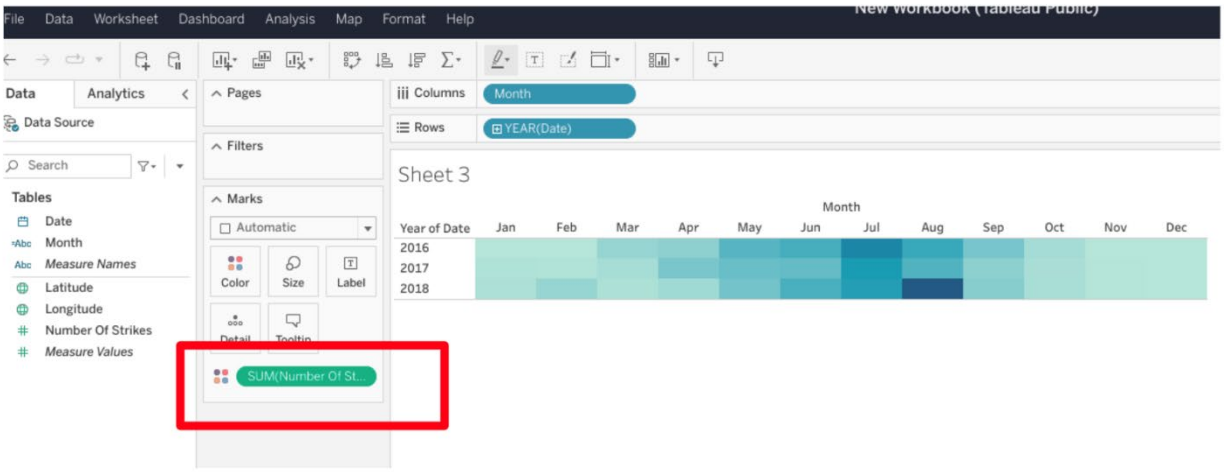
- Click on DATE dropdown in the “Tables” menu. Click CREATE and select Calculated Field.



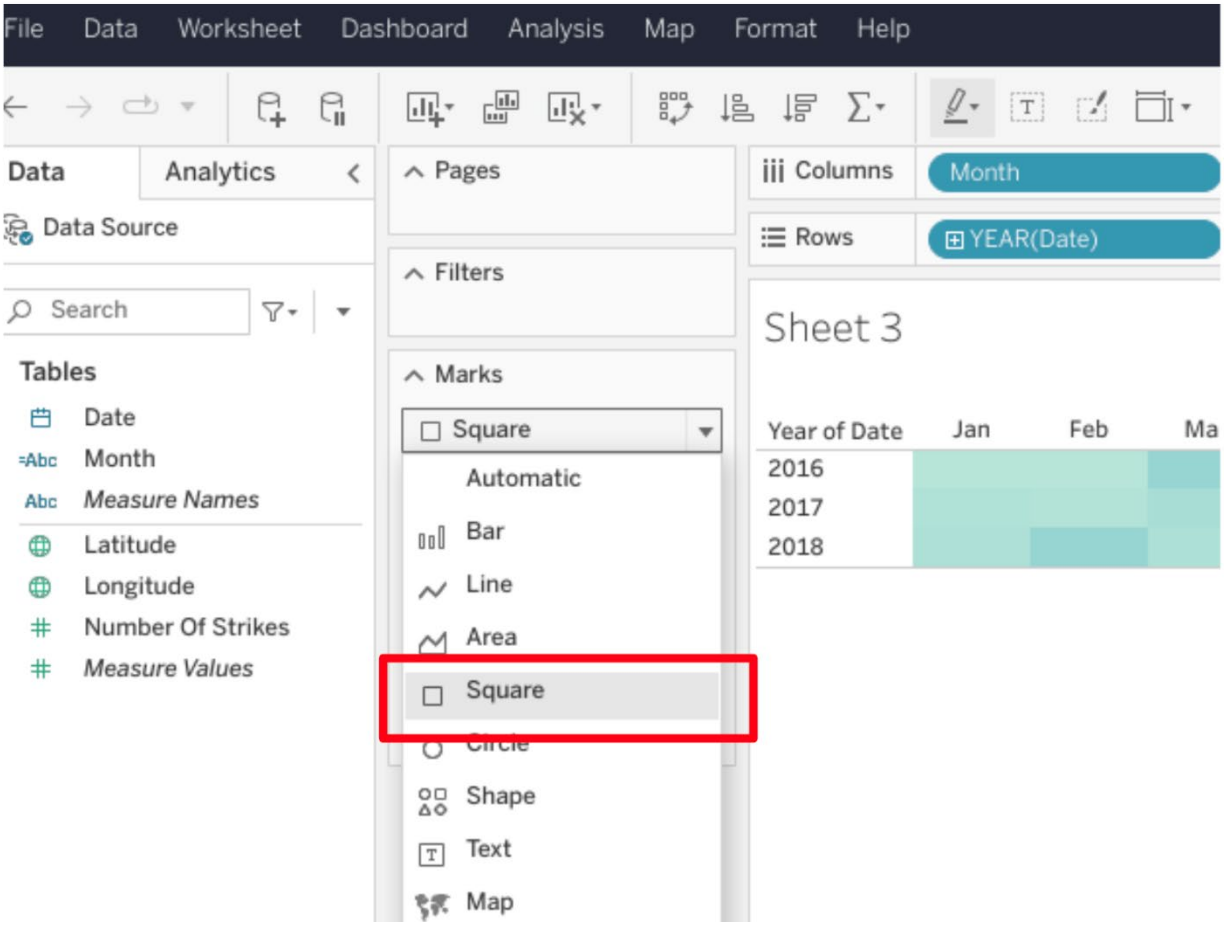
- Type in Month for Calculated Field name.
- Type into field:
  - **LEFT(DATENAME('month',[date]),3)**
  - Click Apply.
  - Click OK.
- Drag month to column field.



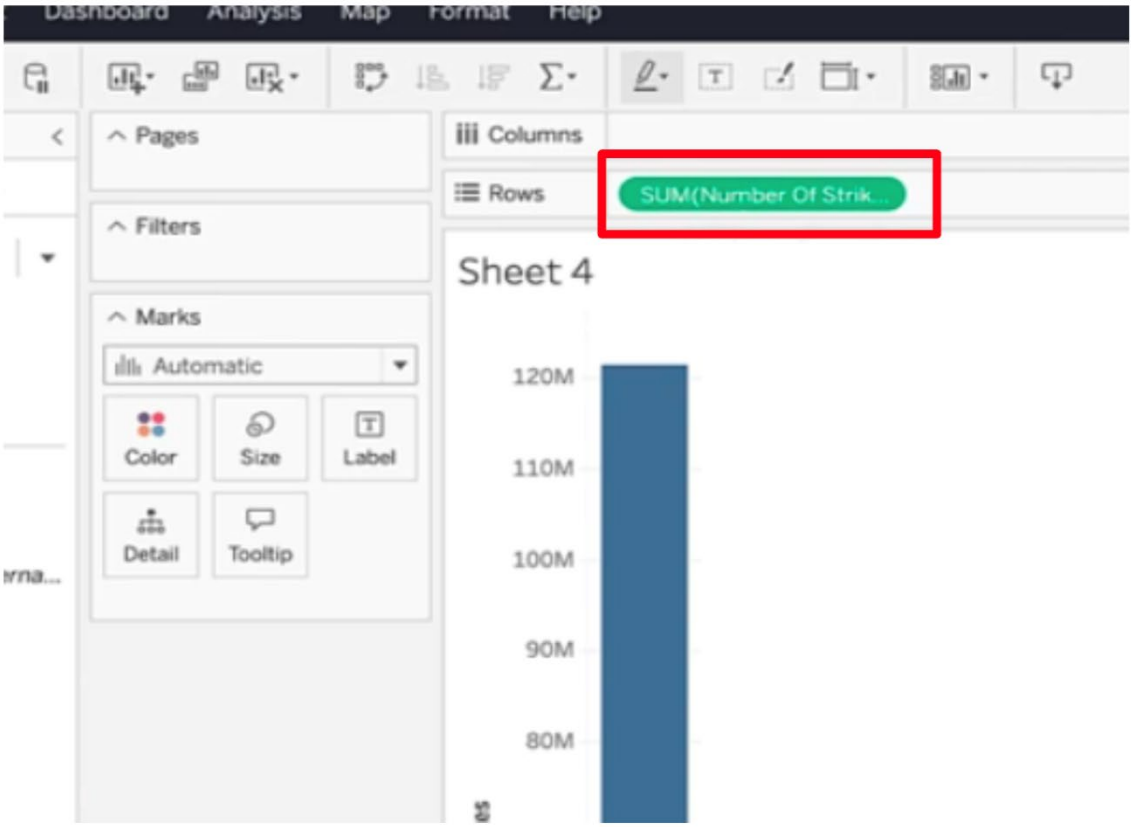
- Drag NUMBER of STRIKES to the color square under the MARKS field.
- Make sure NUMBER of STRIKES is converted to Measure. To do so, right click on Number of Strikes field and select 'Convert to Measure.'



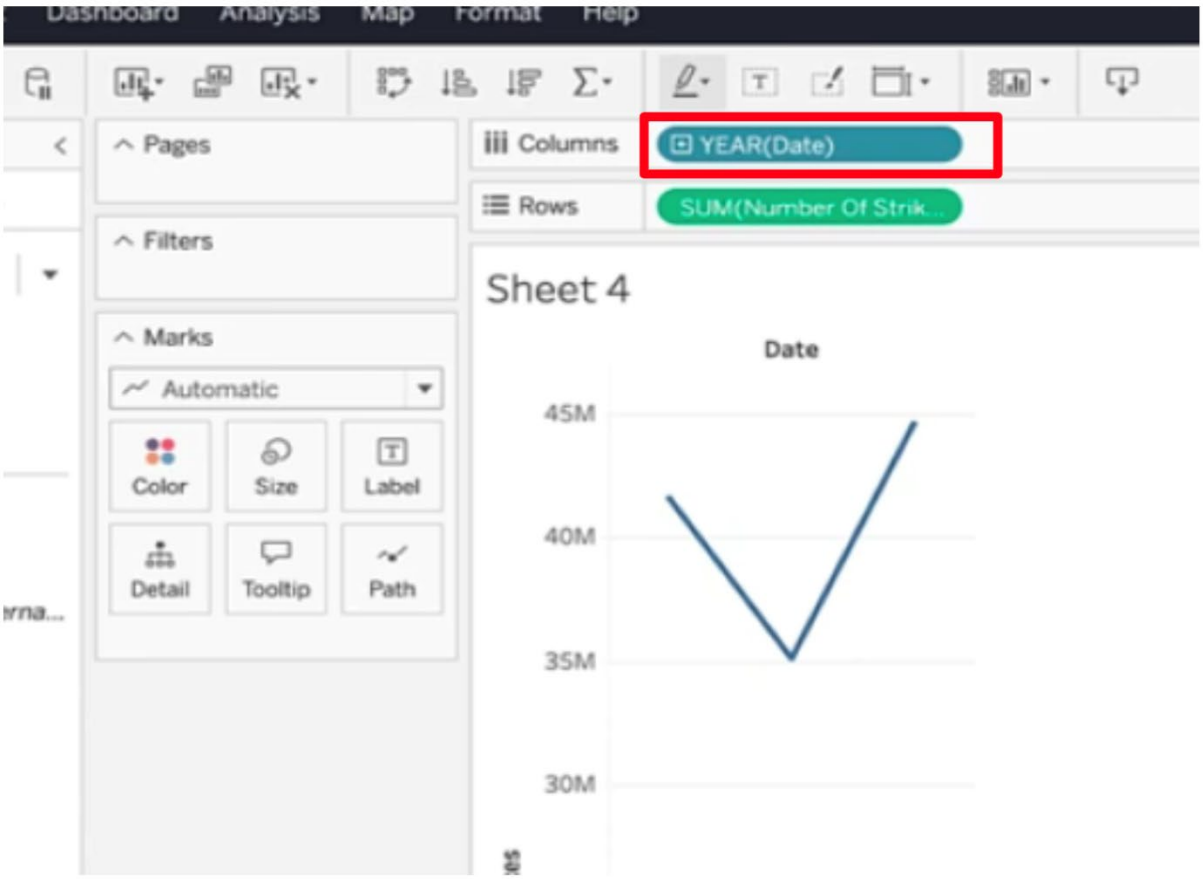
- In Marks dropdown, select SQUARE.



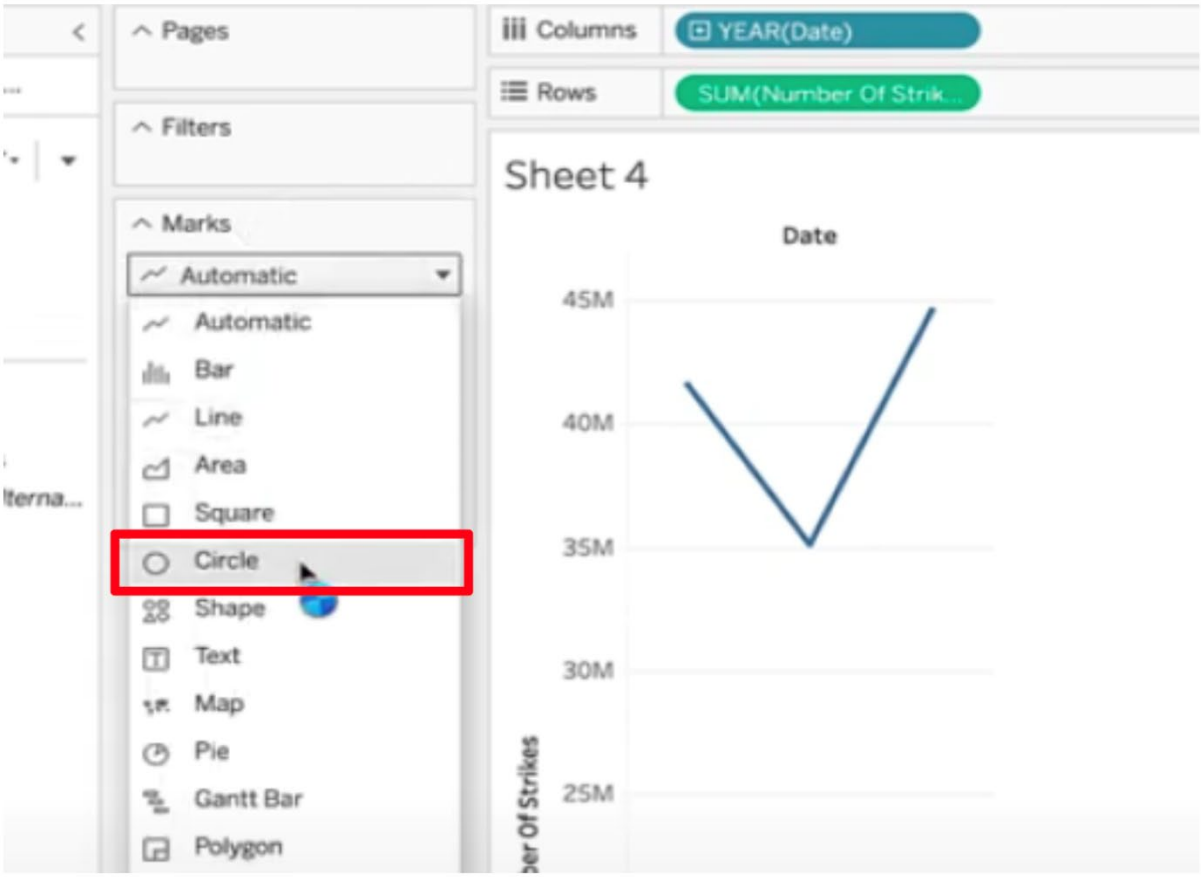
- Feel free to change the color. You've completed a heatmap.
- Click on NEW WORKSHEET.
- Drag NUMBER OF STRIKES to the rows field.



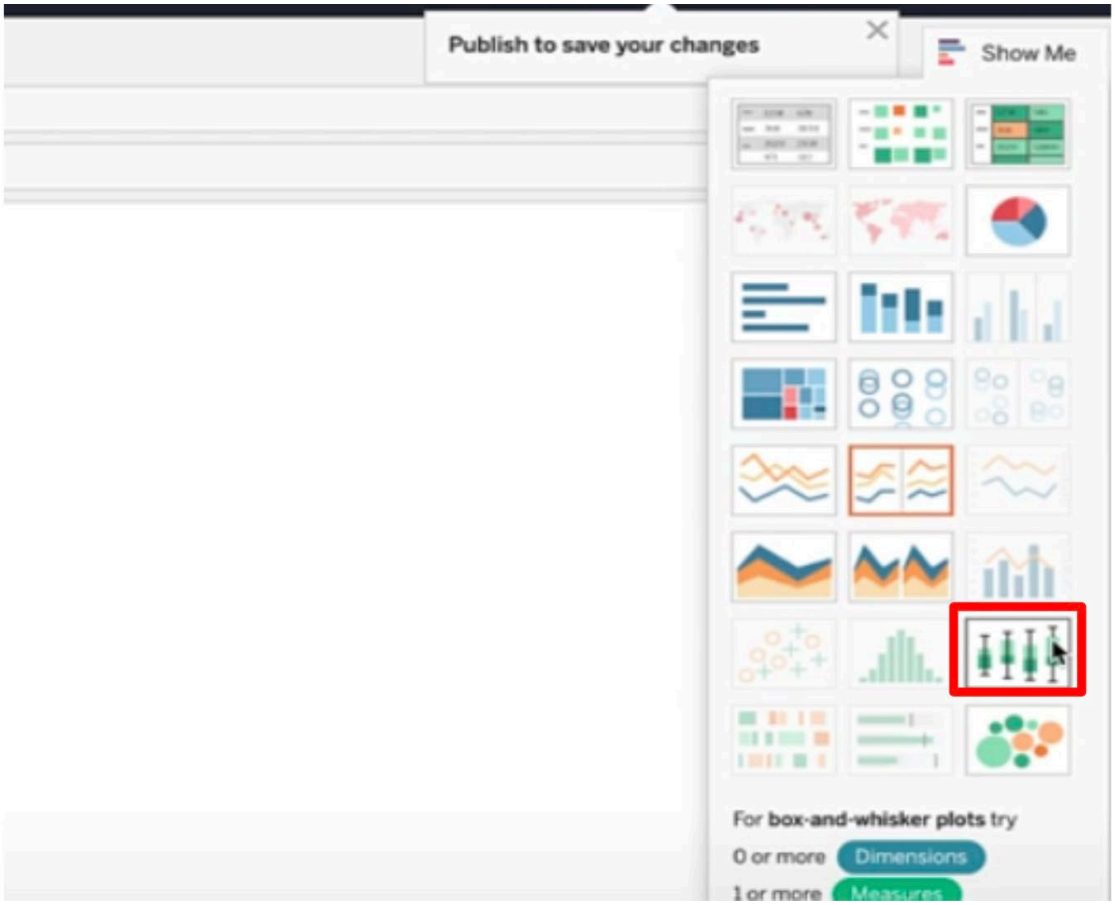
- Drag DATE to the column field and select YEAR.



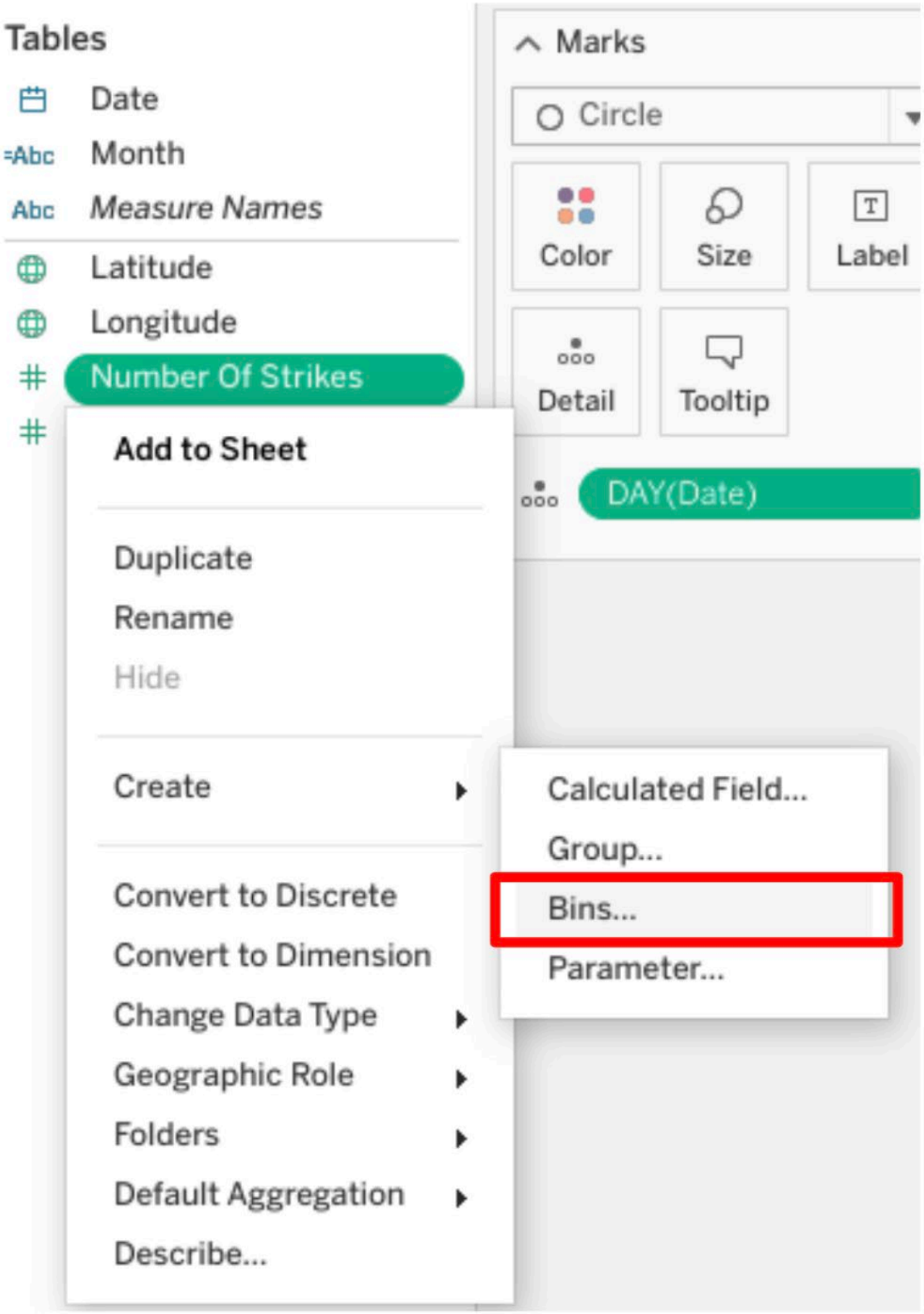
- **Note:** If nothing is shown in the Columns field, drag YEAR back to the Columns filter again.
- Select CIRCLE from MARKS dropdown.



- **Note**: If box plot isn't the default, select the BOX and WHISKER PLOT from the SHOW ME dropdown.
- Drag Date to the column field again, if that has been removed.

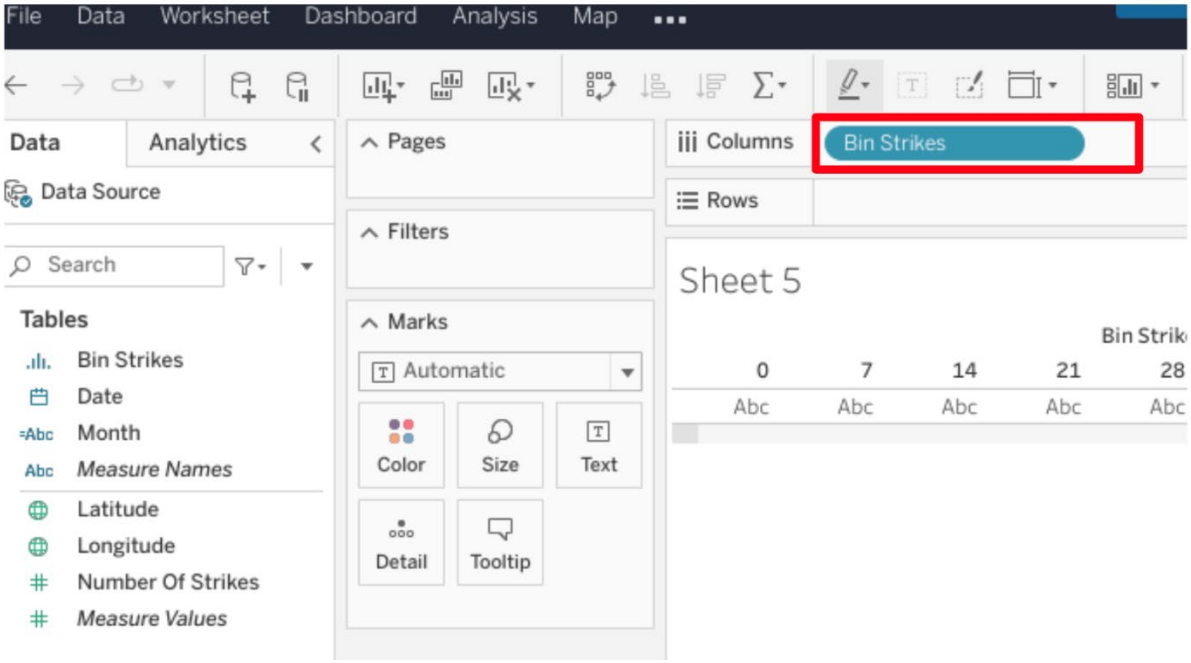


- Drag DATE into the detail square under the MARKS field, and select DAY. **Note:** When you do this, you'll see that Tableau changes your YEAR(Date) filter to QUARTER(Date) to show an additional level of detail. That change is shown in the instructional video, so please proceed.
- Now you've completed a boxplot.
- Click on NEW WORKSHEET.
- Click on the dropdown of NUMBER of STRIKES in Table menu, and select CREATE and then BINS.

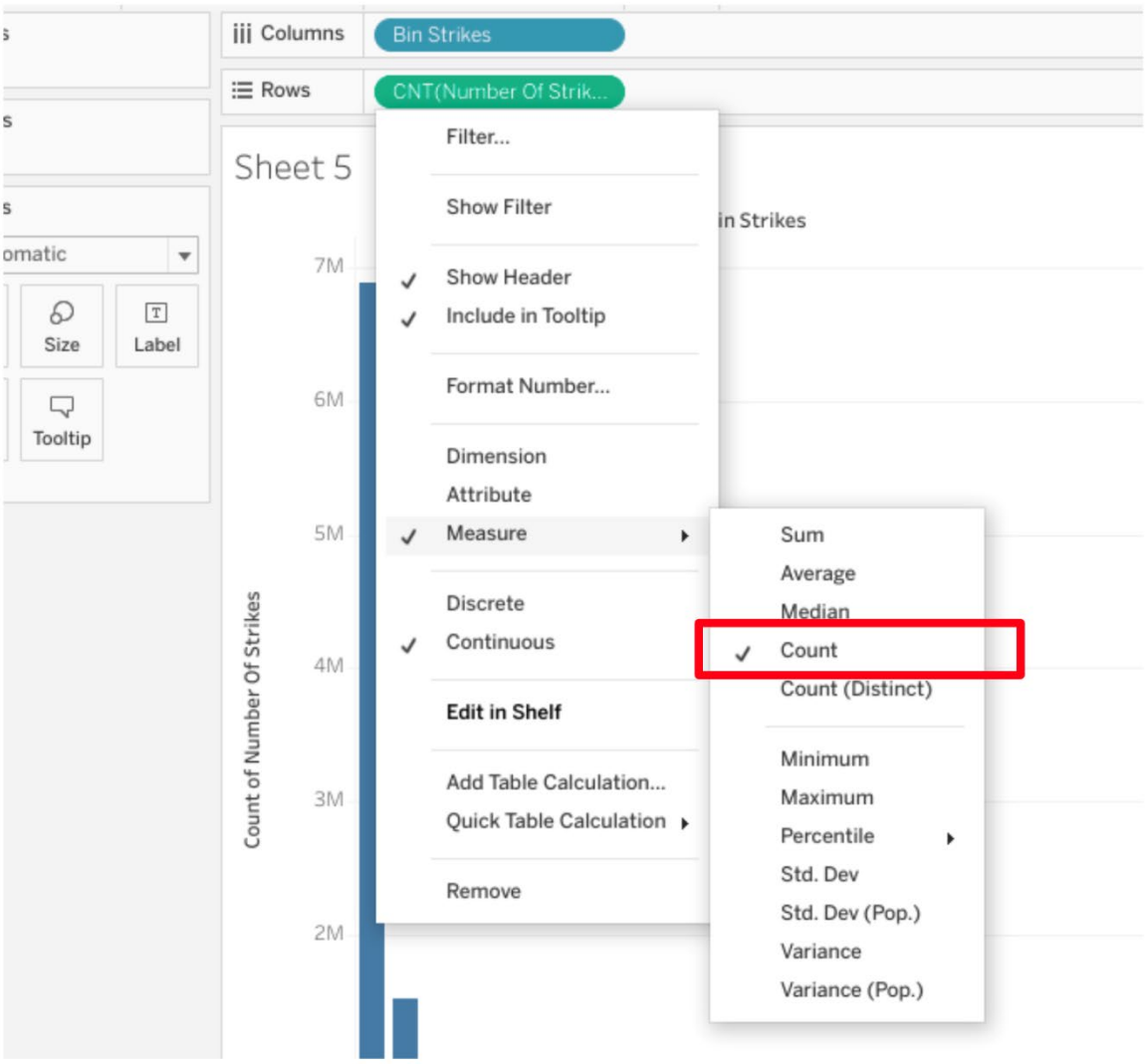


- Give bin a name: BIN STRIKES.

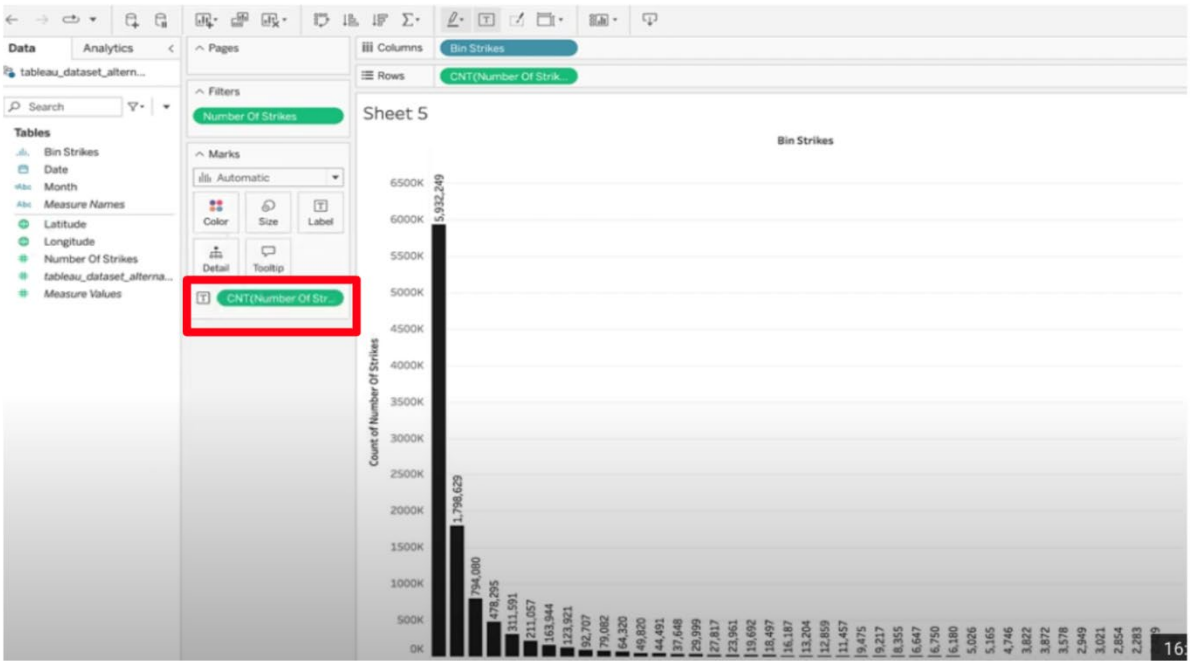
- Select a number between 5 and 10 for “Size of Bins” field.
- Drag BIN STRIKES to columns field.



- Drag NUMBER OF STRIKES to Row Field.
- In NUMBER of STRIKES dropdown, make sure COUNT is selected.



- Drag NUMBER of STRIKES to filter field.
- Limit the field numbers between 1 and 200.
- Drag NUMBER of STRIKES to the LABEL.
- Select COUNT.
- Change color as desired.
- Now you know how to create a histogram!


Data
Analytics
tableau_dataset_altern...
Search
Tables
Bin Strikes
Date
Month
Measure Names
Latitude
Longitude
Number Of Strikes
tableau_dataset_alterna...
Measure Values
Pages
Filters
Number Of Strikes
Marks
Automatic
Color
Size
Label
Detail
Tooltip
CNT(Number Of Str...
Columns
Bin Strikes
Rows
CNT(Number Of Strik...
Sheet 5
Bin Strikes
Count of Number Of Strikes
6500K
6000K
5500K
5000K
4500K
4000K
3500K
3000K
2500K
2000K
1500K
1000K
500K
0K
5,932,249
1,798,629
794,080
478,295
311,591
211,057
163,944
123,921
92,707
79,082
64,320
49,820
44,491
37,648
29,999
27,817
23,961
19,692
18,497
16,187
13,204
12,859
11,457
9,475
9,217
8,355
6,647
6,750
6,180
5,026
5,165
4,746
3,822
3,872
3,578
2,949
3,021
2,854
2,283

Activity Exemplar: Design a bar graph that tells a story in Tableau Public

Here is a completed exemplar along with an explanation of how the exemplar fulfills the expectations for the activity.

Completed Exemplar

To review the exemplar for this course item, click the link below.

Link to exemplar: Seasonal Seoul Average Bike Rentals on Weekdays in 2018

Assessment of Exemplar

Compare the exemplar to your completed activity. Review your work using each of the criteria in the exemplar. What did you do well? Where can you improve? Use your answers to these questions to guide you as you continue to progress through the course.

Note: *The exemplar represents one possible way to complete the data visualization. Yours will likely differ in certain ways. What's important is that your data visualization meets the business scenario criteria.*

How the exemplar meets the criteria

Notice how the exemplar posted to Tableau Public uses the detail from the given scenario to adhere to the following guidelines:

The data visualization includes only data from 2018.

- The "YEAR(Date)" variable is in the filter field and is check marked for 2018 only.

The data visualization includes only weekdays, Monday to Friday.

- The "WEEKDAY(Date)" variable is in the filter field and is check marked for Monday through Friday only.
- The "Hour" variable in the filter field is check marked on the hours 8 to 17.

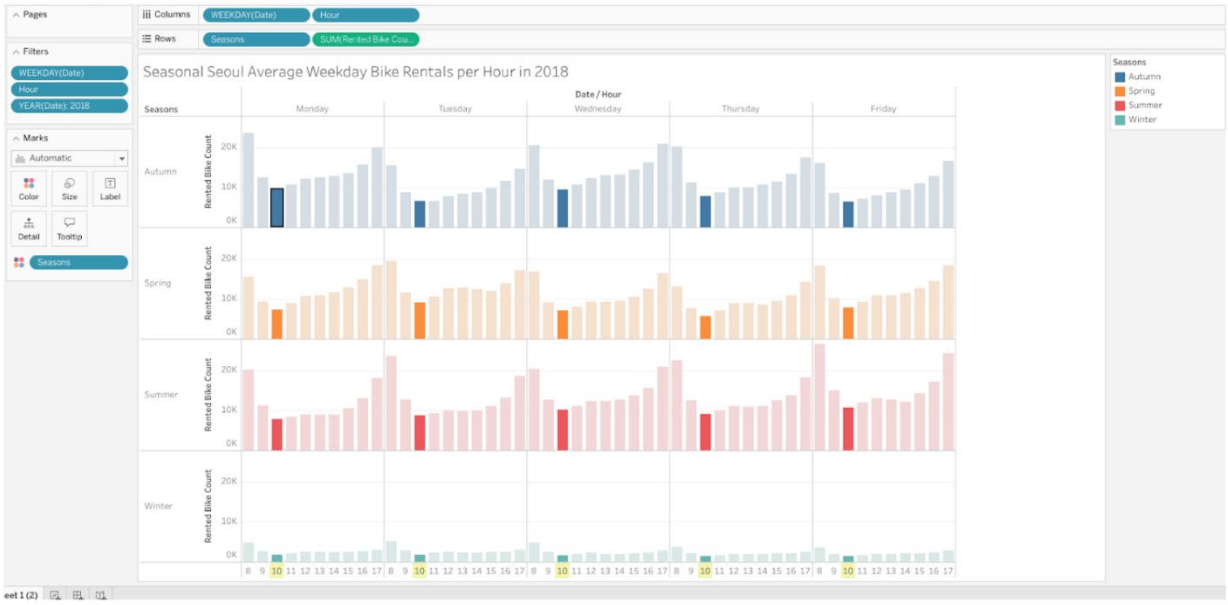
The data visualization type makes sense for the data shown and includes color, contrast, labeling, and emphasis elements that focus on the most relevant data.

- The chart selected was a bar chart. A scatter plot and box plot could be helpful, but would likely be too complex for the purpose of the visualization. A geographic map would not be helpful at all since there is no geographic data.
- The blue color bars represent viable times of day and dark orange are times in which bike rental traffic is too high for maintenance to be viable.
- Annotations are added to highlight the relevant information.

- A title was added.
- The "Seasons" variable was added to provide a more thorough answer.

The data visualization can be observed without scrolling out or zooming in.

- Because of the filtering mentioned above (2018 data, excluding weekends and non-business hours), the visualization fits on one screen or a printout.



Follow-along guide: Craft compelling stories with Tableau

This document includes detailed instructions for how to perform the data visualizations described in the video "Craft compelling stories with Tableau."

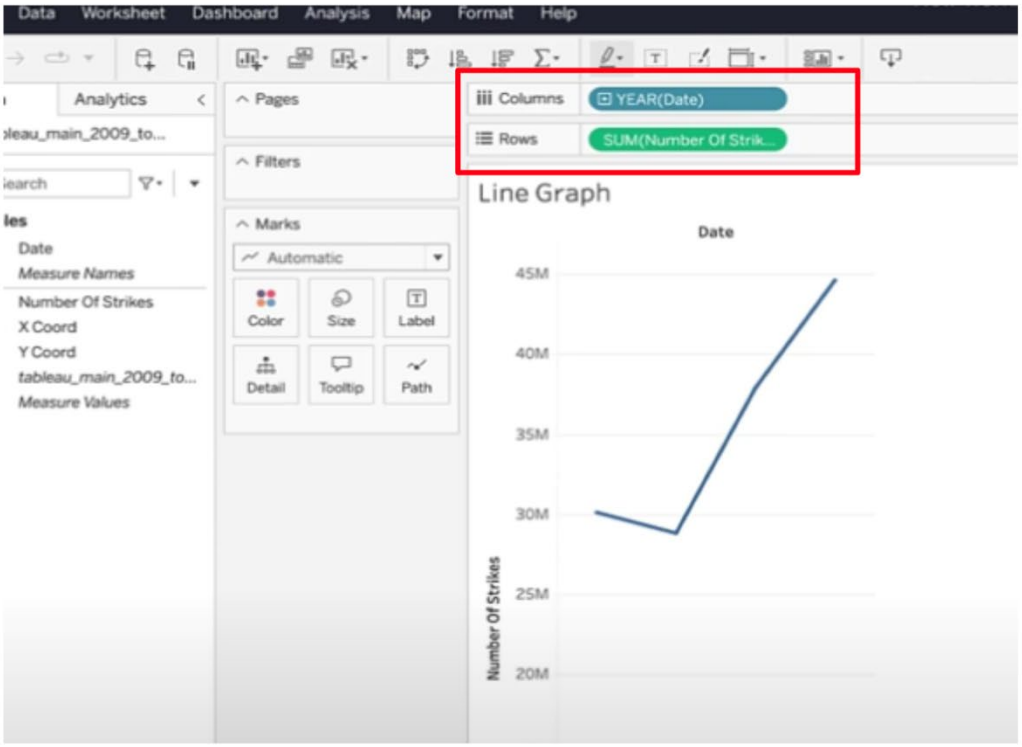
The following guide points out areas of the video that may require adjustment. These resource guides can also serve as a set of usability reminders for you to recall when using Tableau in your future career.

Instructions

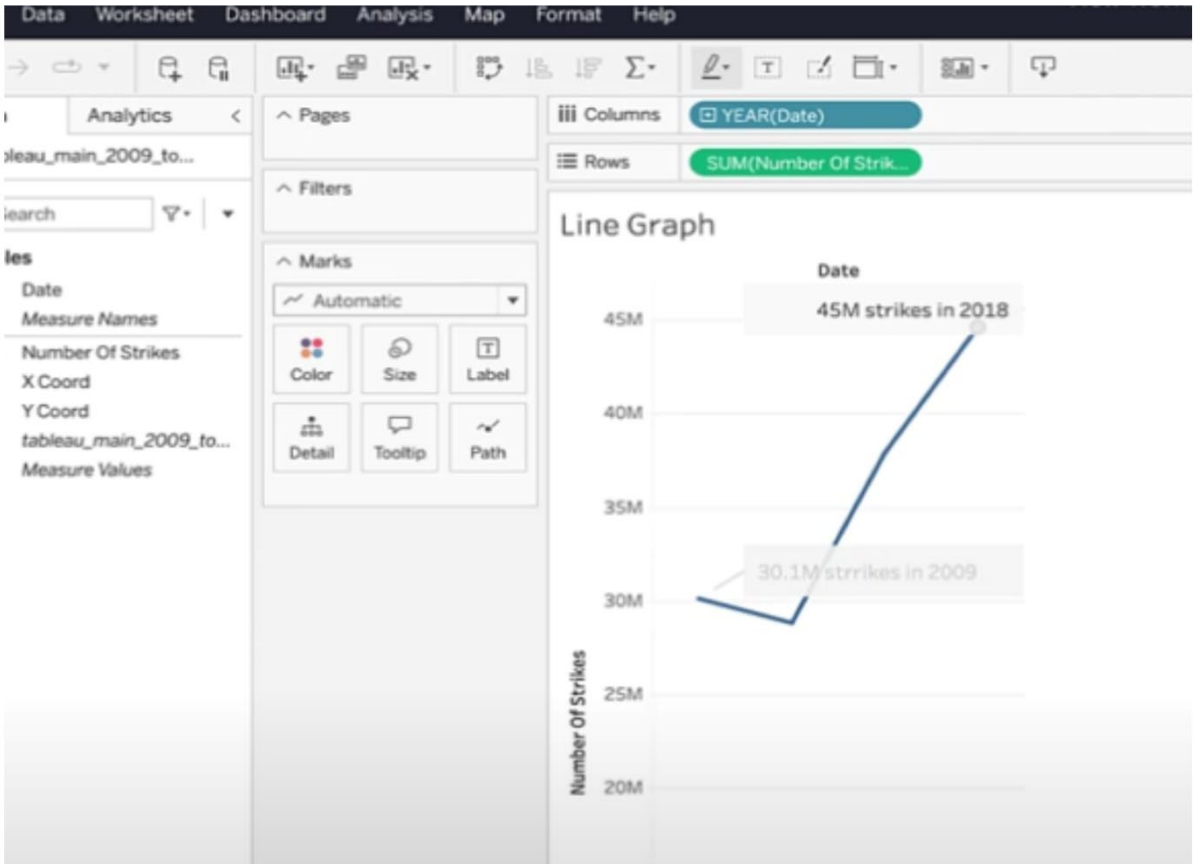
- Go to https://public.tableau.com/s/
- Since you've already set up your Tableau Public profile, all you need to do is log in and select **Web Authoring** under **Create** in the navigation bar.
- Select the appropriate CSV file provided in the instructions. The dataset you'll use with this instructional video is: tableau_main_2009_to_2018.csv.
    - Please be aware that when you download the zip file folder provided, the computer automatically names that zip file folder with a long string of numbers and letters. You have to open that folder and then upload the individual files that are named correctly and match what's shown in the video.
- Click on NEW WORKSHEET.

Note: Please allow several minutes for data import into a new worksheet.

- Move the blue Date field to the column field.
- Select YEAR from dropdown.
- Move NUMBER OF STRIKES to the row field.

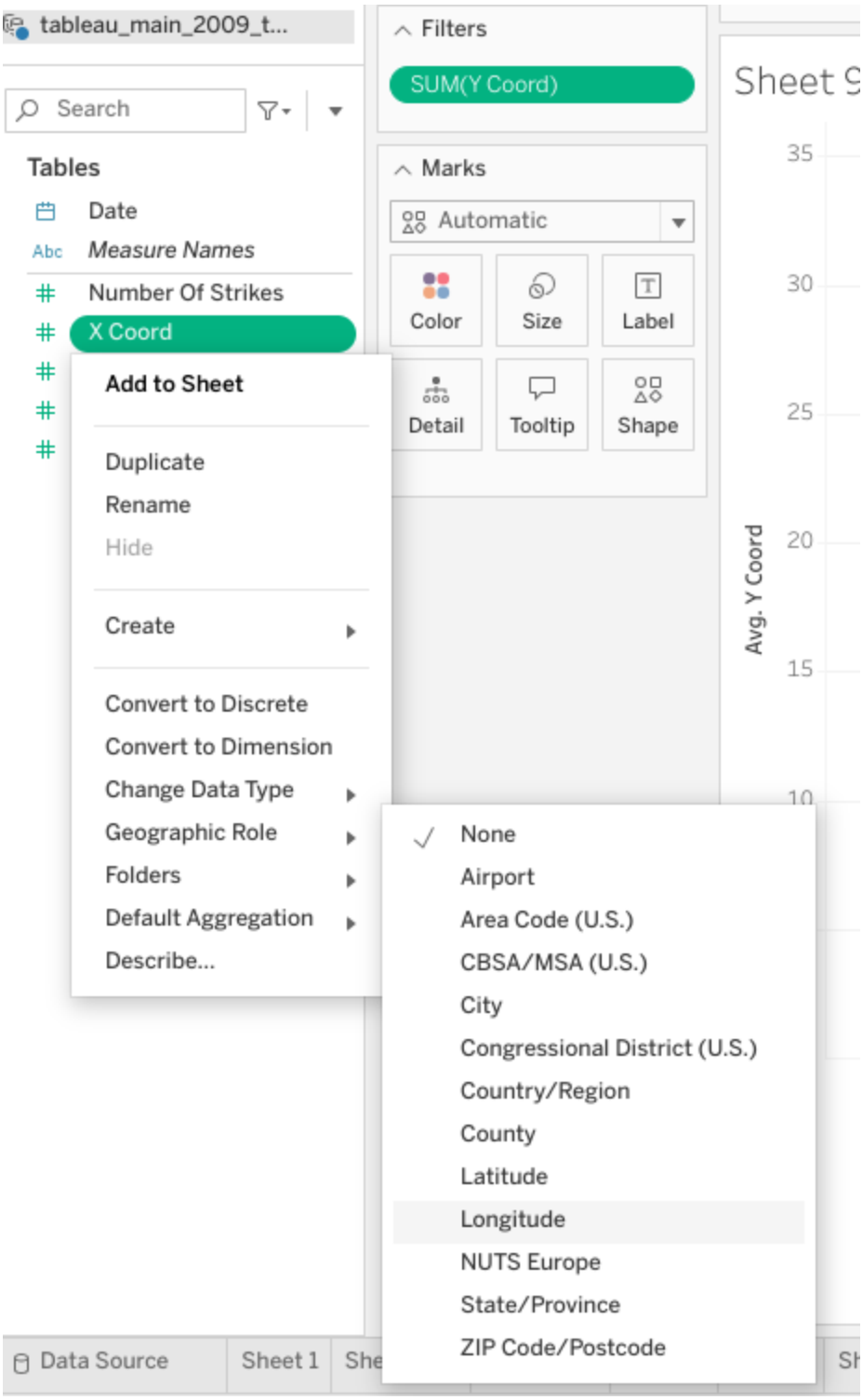


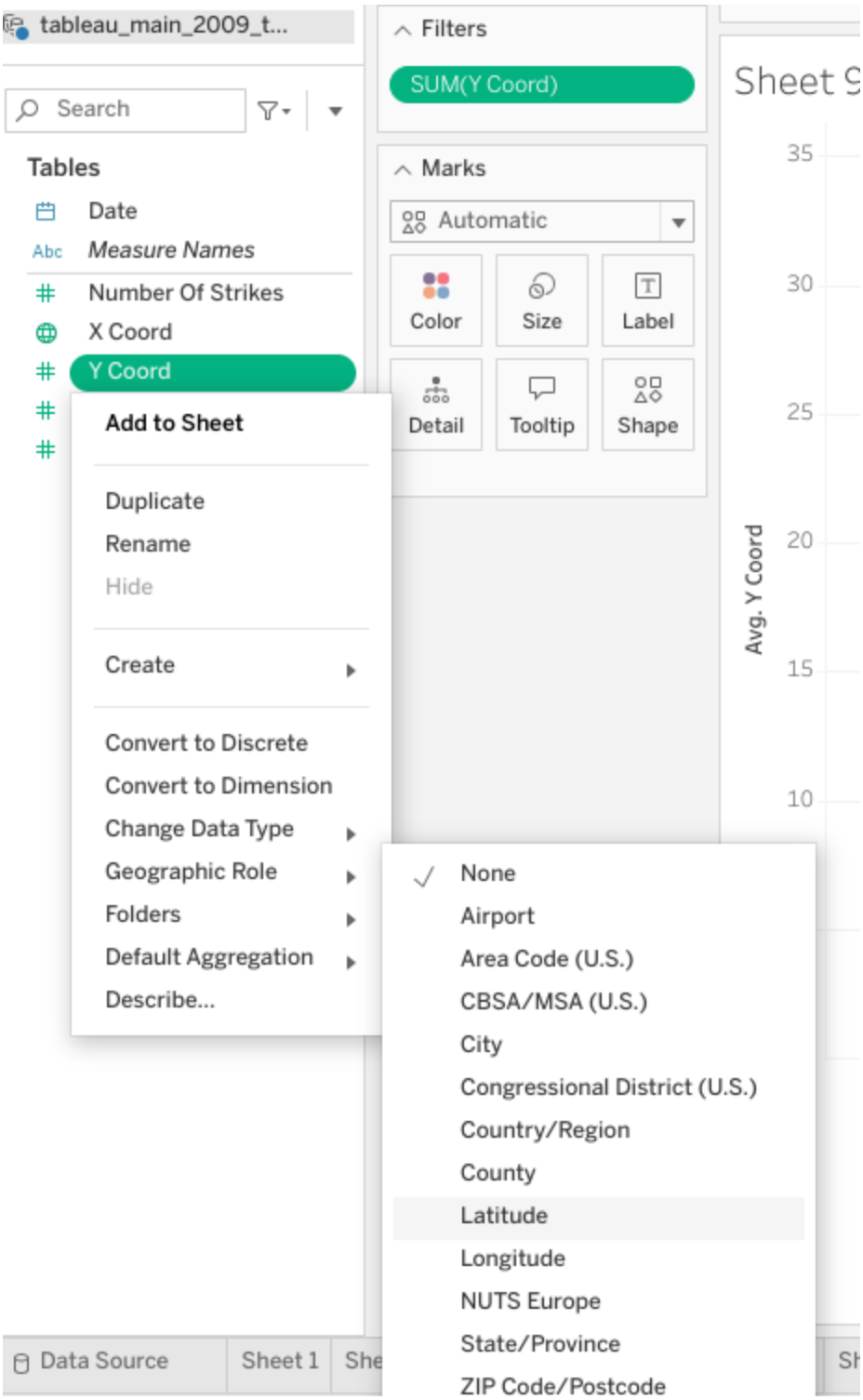
- If your visualization doesn't default to a line graph, click on "SHOW ME" in the upper right of the screen and select Lines (discrete).
- Create annotation by **right clicking** on each end of the line and selecting "Annotate mark."
- Type in "30.1M" for the Number of Strikes.
- Create annotation on the final mark in the same way.
- Type "45M" for number of strikes in 2018.



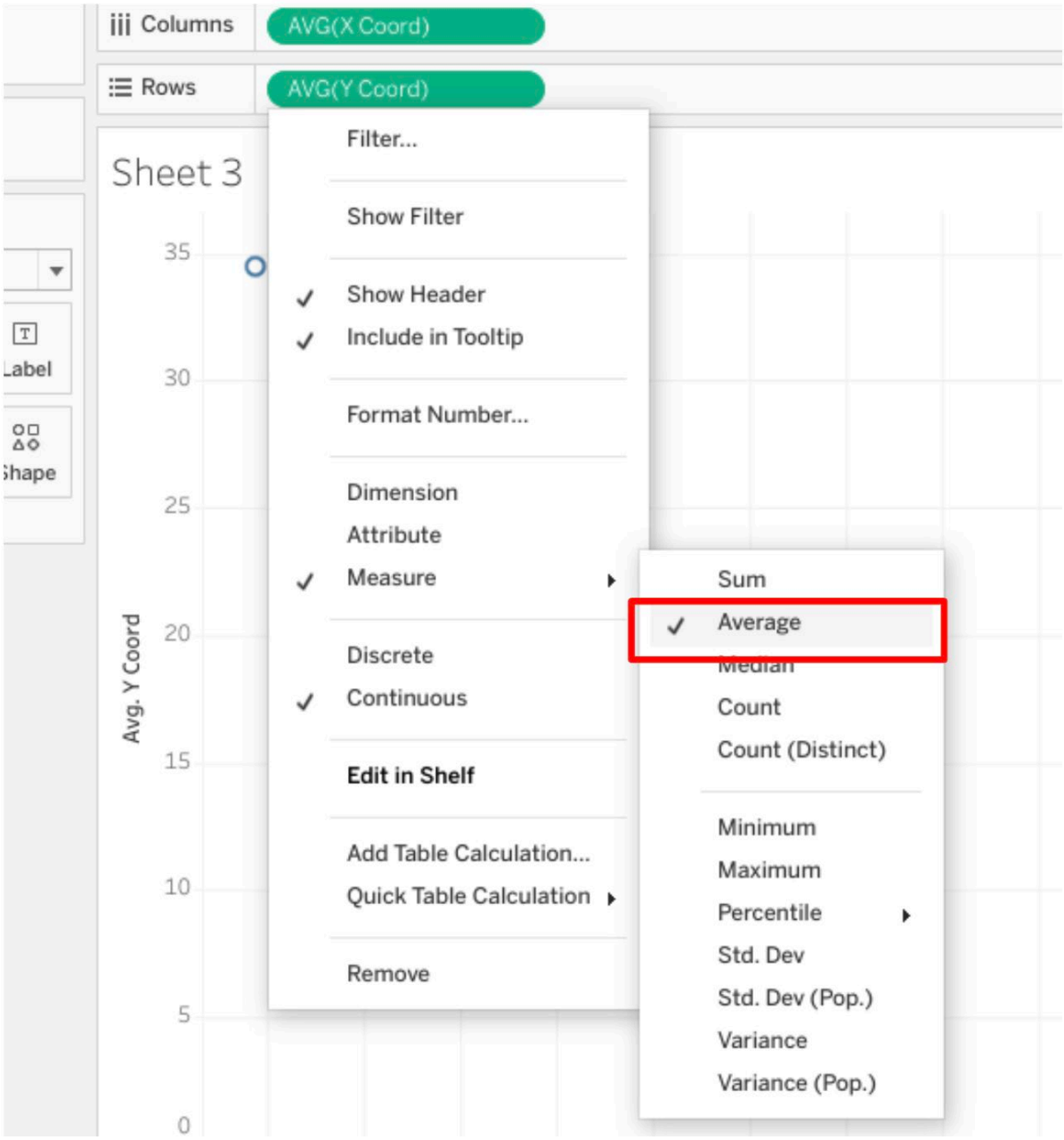
- Click on NEW WORKSHEET.
- Drag the X-coord to the column field. Drag the Y-coord to the Row field.

Note: Make sure the X and Y coordinates are continuous dimensions with Y coord geographic role set to latitude and X coord geographic role set to longitude.

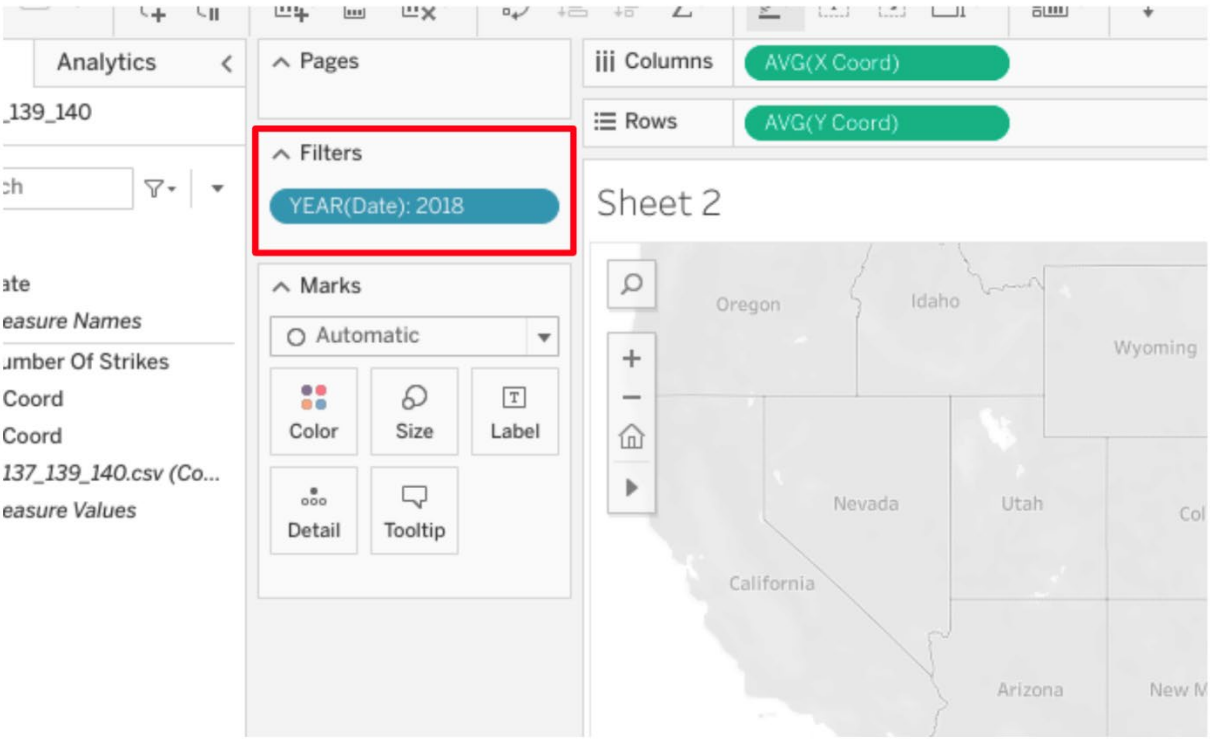

tableau_main_2009_t...
Search
Tables
Date
Measure Names
Number Of Strikes
X Coord
Add to Sheet
Duplicate
Rename
Hide
Create
Convert to Discrete
Convert to Dimension
Change Data Type
Geographic Role
Folders
Default Aggregation
Describe...
None
Airport
Area Code (U.S.)
CBSA/MSA (U.S.)
City
Congressional District (U.S.)
Country/Region
County
Latitude
Longitude
NUTS Europe
State/Province
ZIP Code/Postcode
Filters
SUM(Y Coord)
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Shape
Sheet 9
35
30
25
20
15
10
Avg. Y Coord
Data Source
Sheet 1



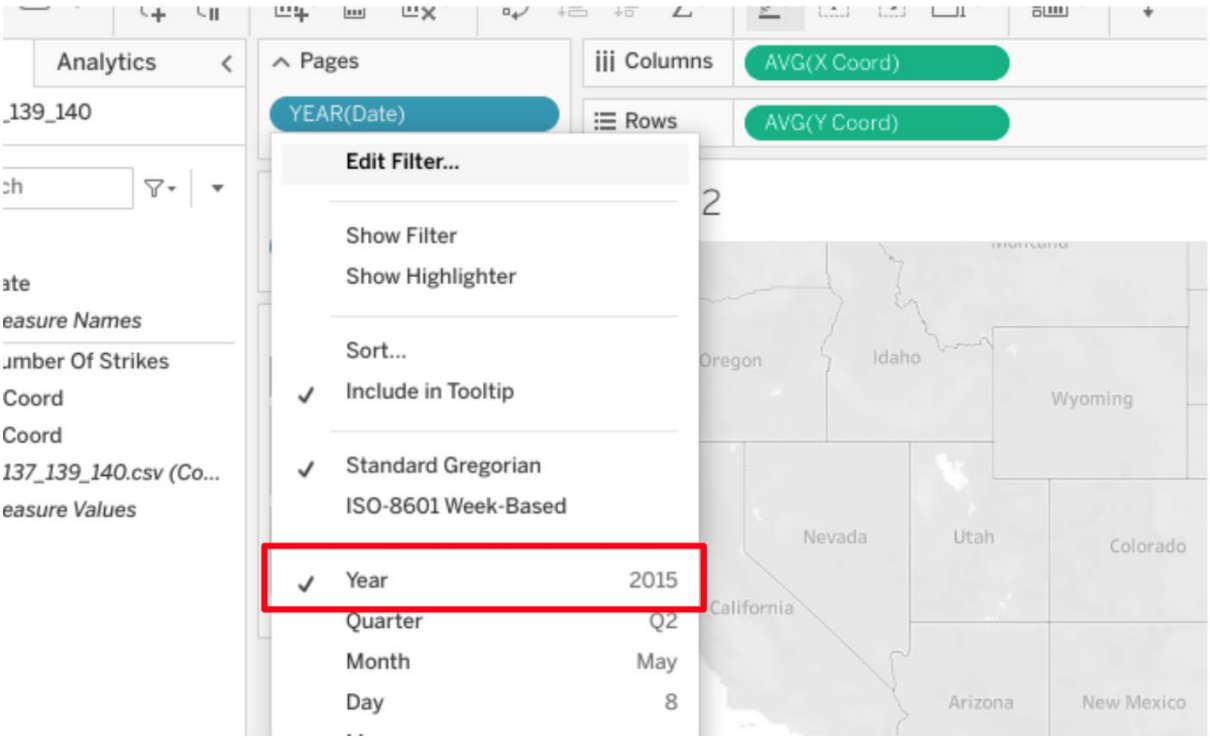
- Click on the dropdown for both the X and Y coordinates.
- Select MEASURE then AVERAGE.



- Drag DATE to filter field.

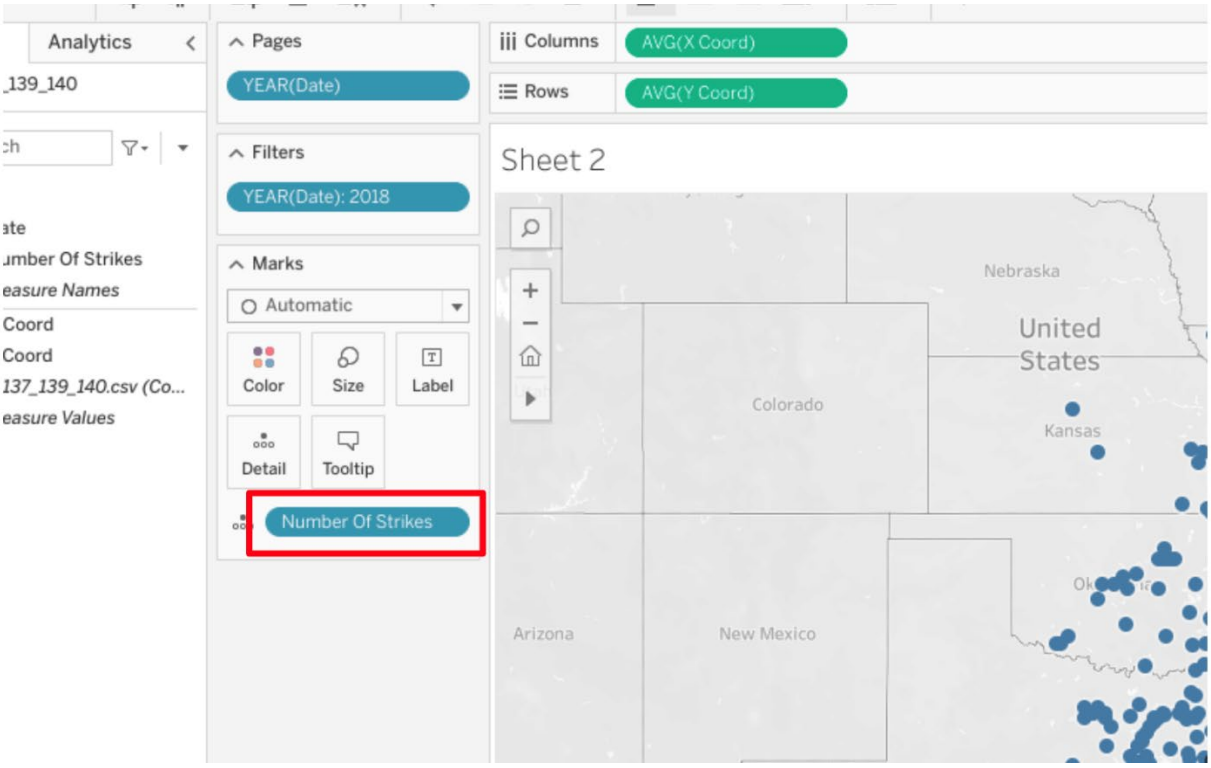


- Drag DATE to pages field, select year from dropdown.

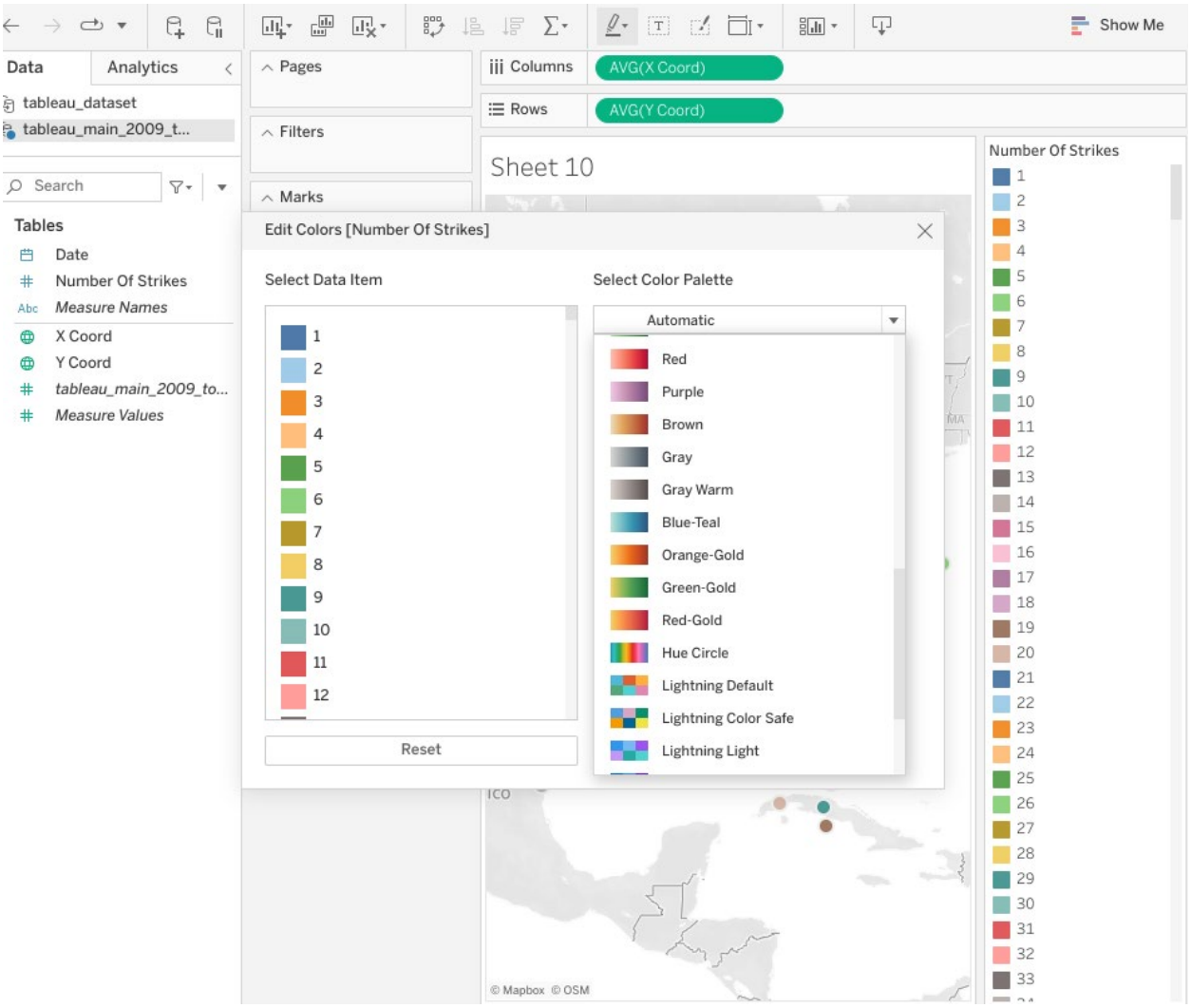


- Drag NUMBER of STRIKES to the detail field.

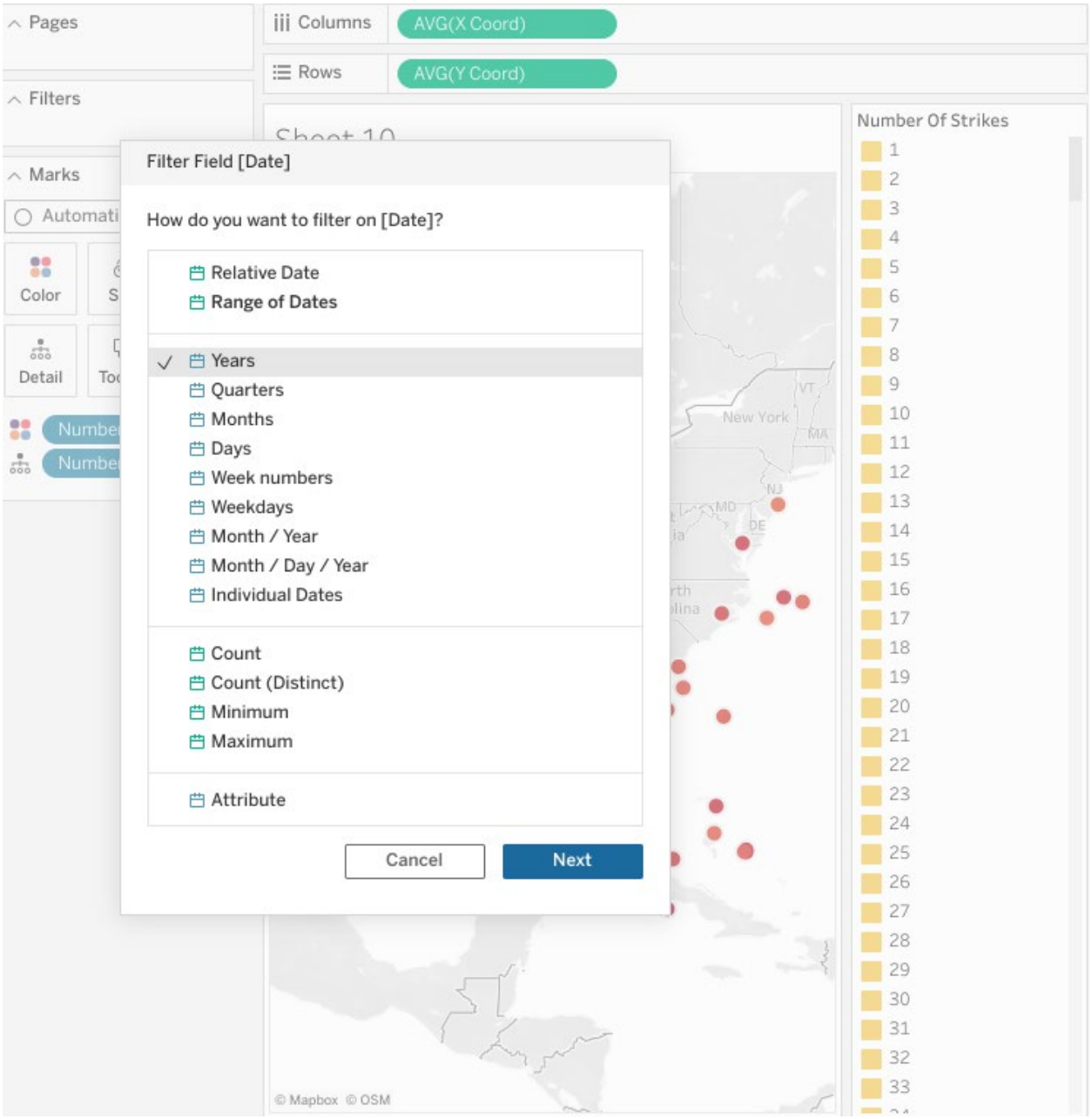
Note: Be sure to click the dropdown on "Number of Strikes" and convert it to a dimension.



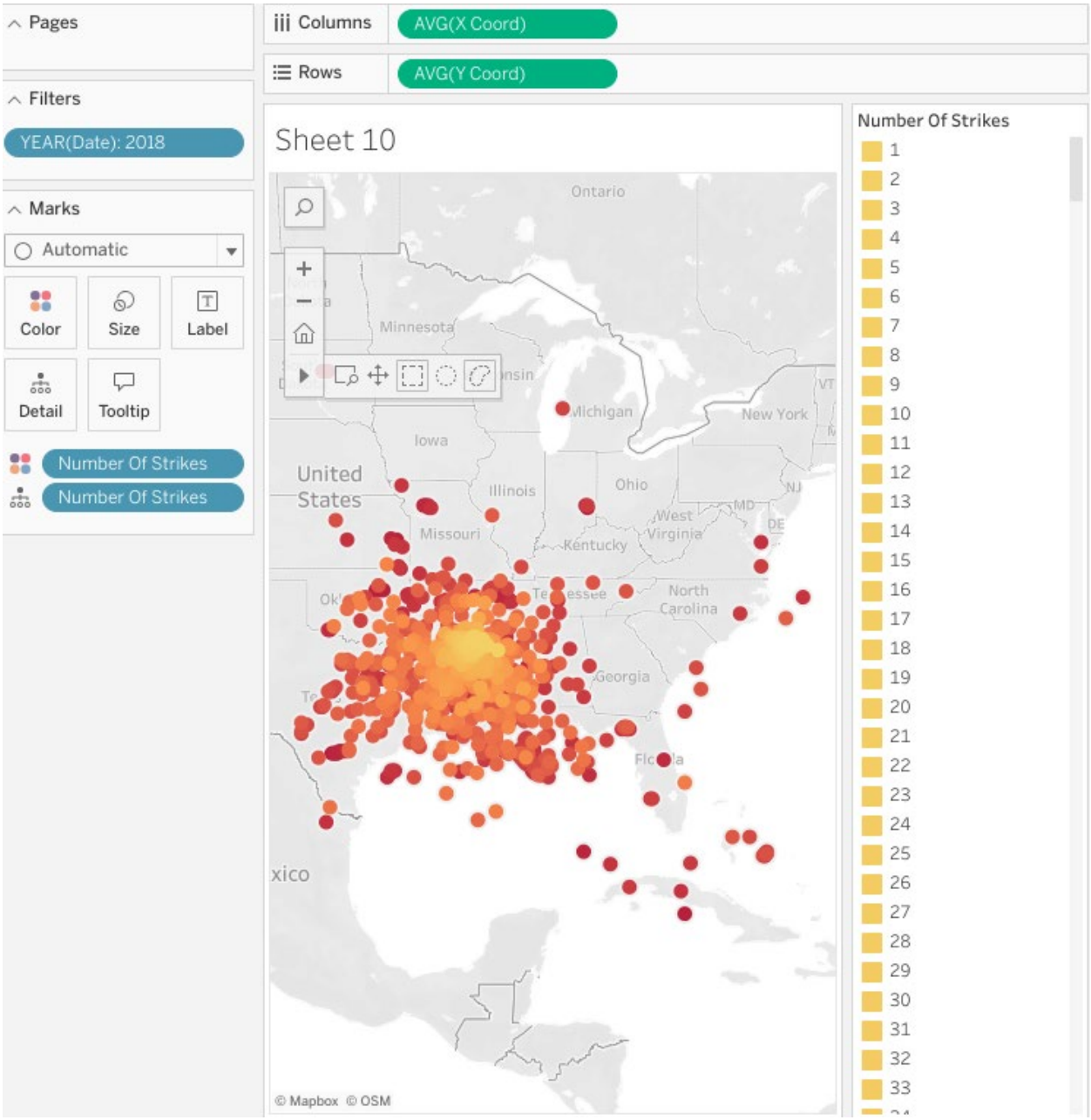
- Click on NEW WORKSHEET.
- Drag the X-coord to the column field. Drag the Y-coord to the Row field.
- Drag the Number of Strikes measure to the detail field.
- **Note:** Make sure the Number of Strikes measure is Discrete. To do so, right click Number of Strikes measure and click "Convert to Discrete."
- Add Number of Strikes to the color field and make the color coincide with the number of strikes. Select Red-Gold under "Automatic" drop down.



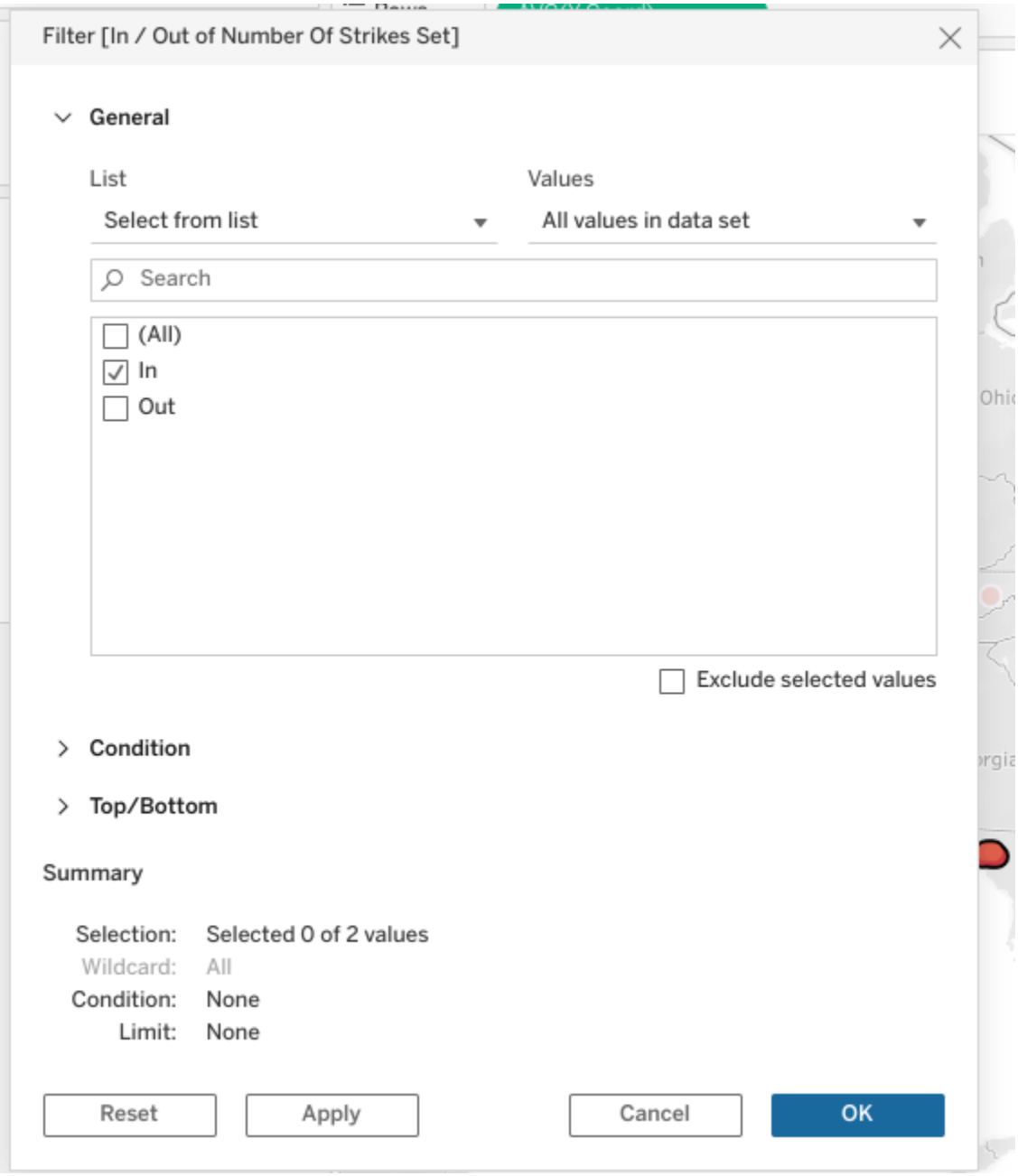
- Drop Date into the Filters field and select Year.



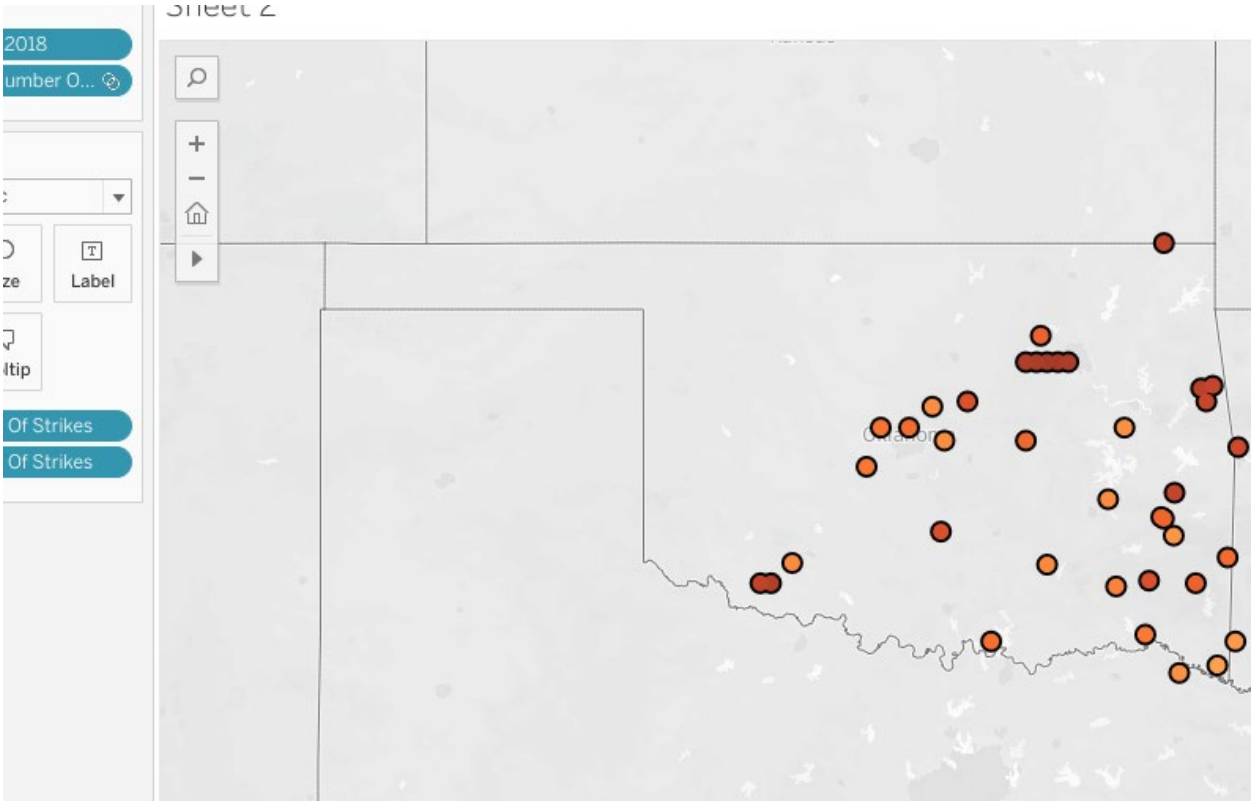
- Use the lasso tool. To do so, hover your cursor over your geographic map visualization and a small line of graphics will appear on the left side. Hover over the right arrow then click on the lasso image.

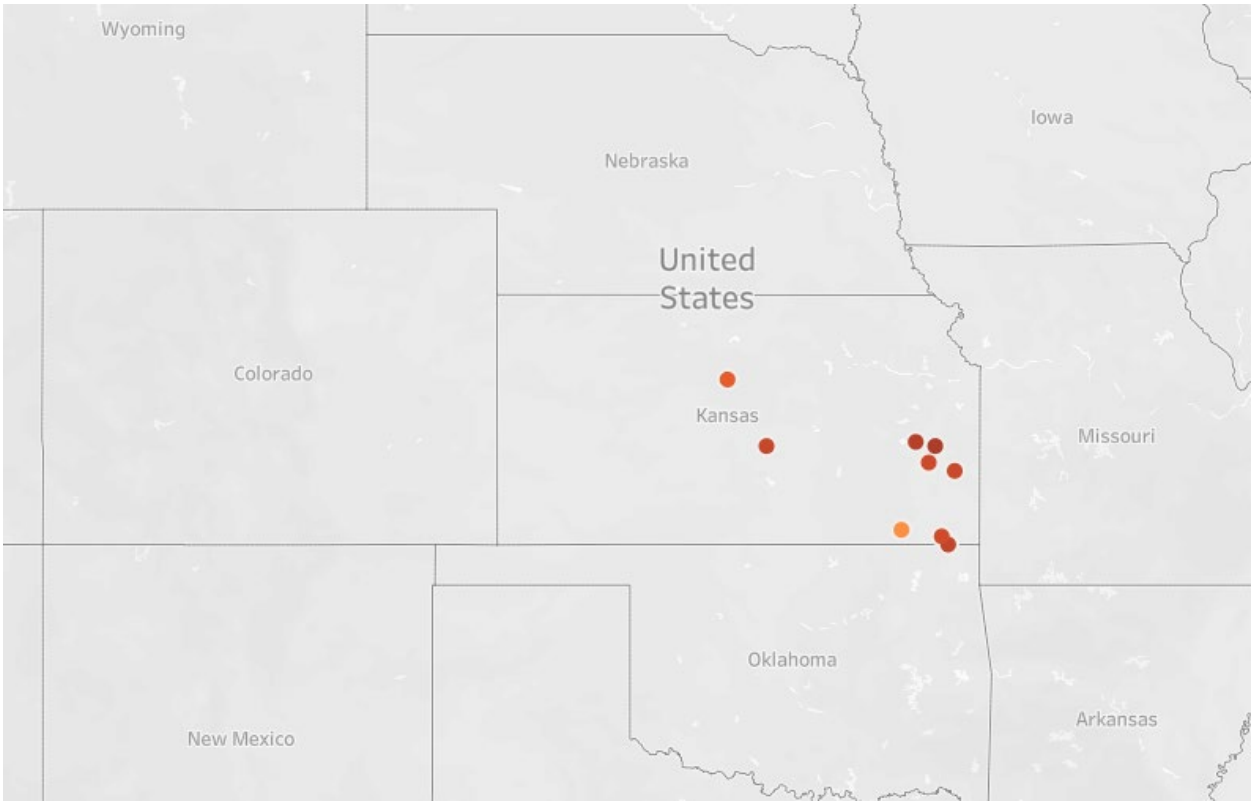


- Drag your cursor over the Texas state border, select Keep Only then click Create Set.
- Drag your new set to the Filters field and select ‘In’ from Filter.

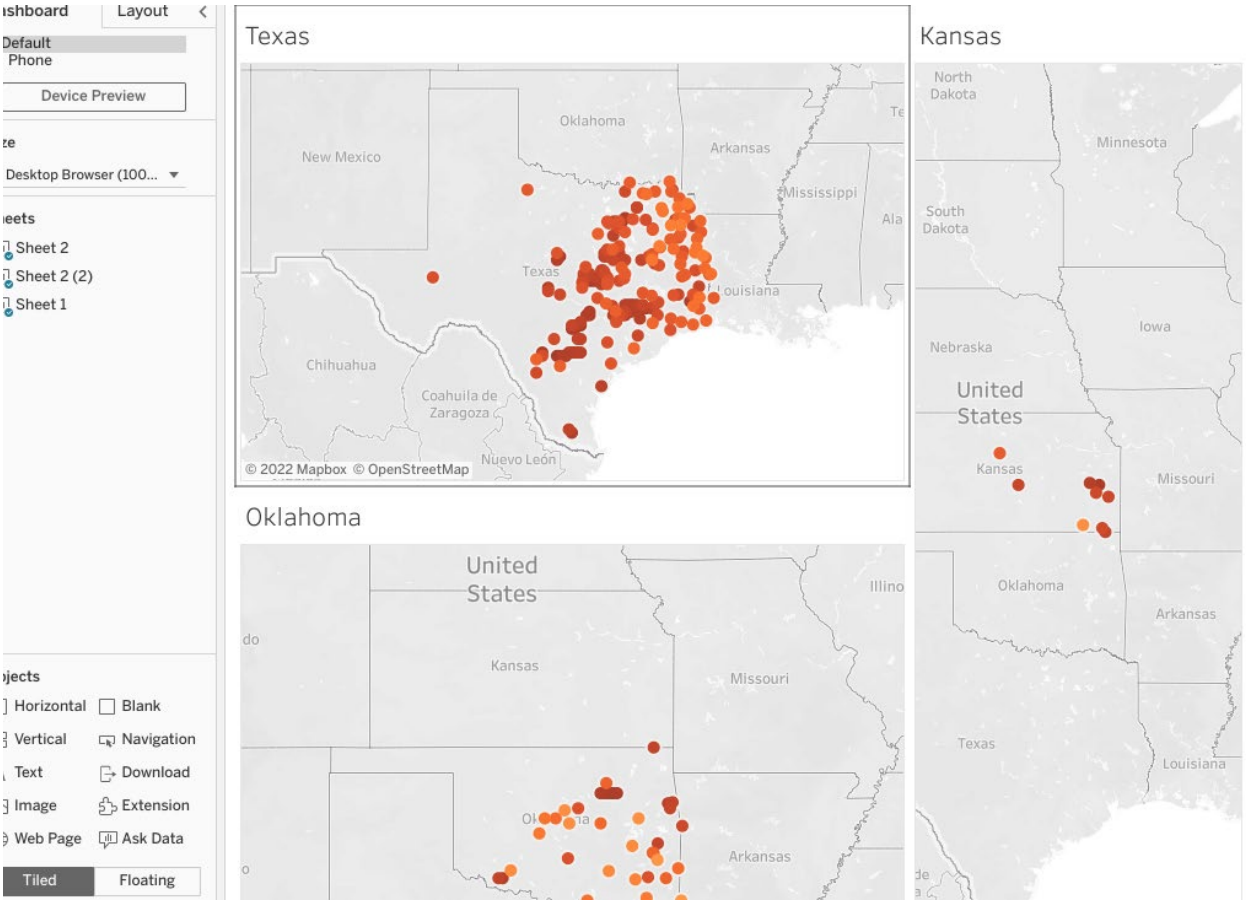


- **Note:** The dashboards section shown in the final part of this instructional video should be completed on your own. It is up to you to design the Oklahoma and Kansas visualizations by following the same steps as performed for Texas in the video. For detailed instructions to create the on-screen visuals shown in the remainder of this video, please follow the proceeding steps.
- Create two duplicates of this sheet and redo the same tracing of states process, but for Oklahoma and Kansas.

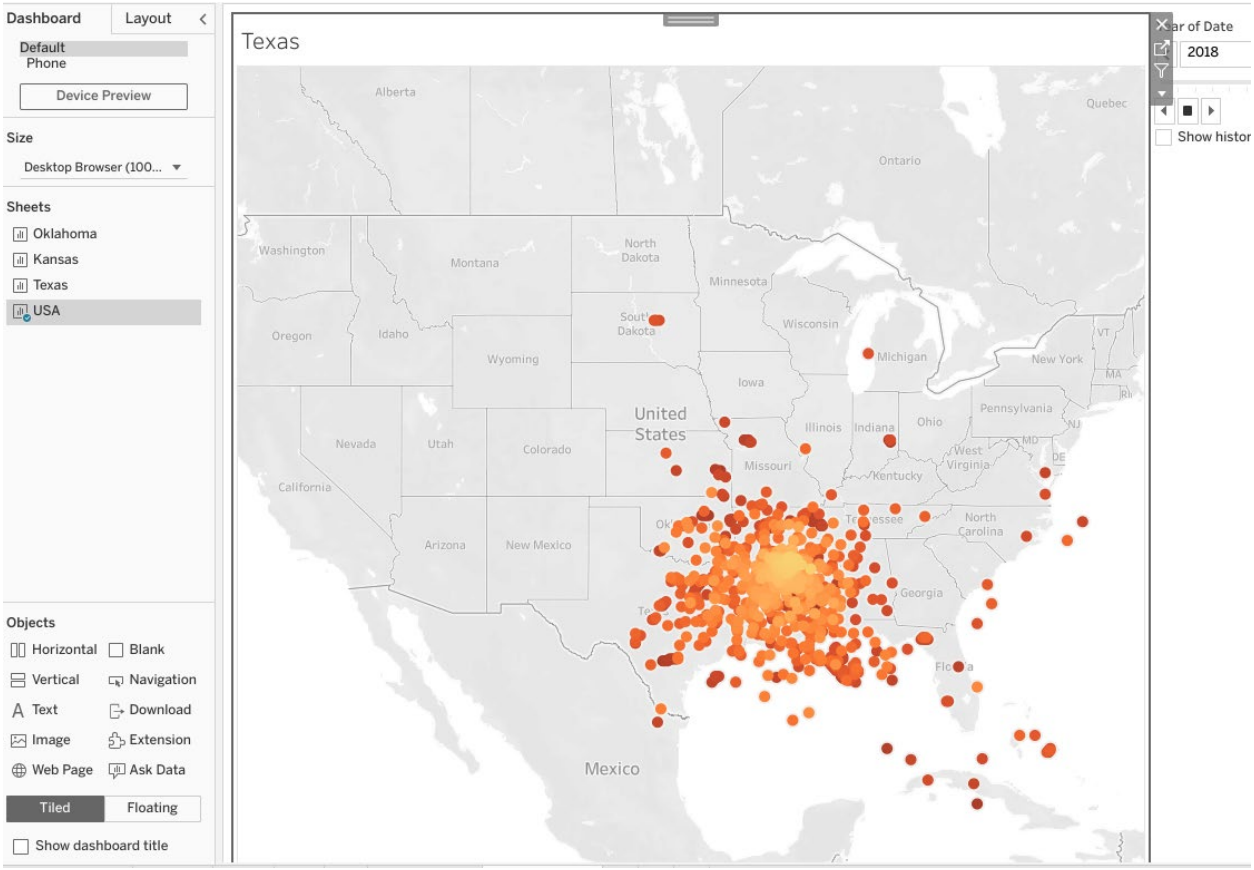




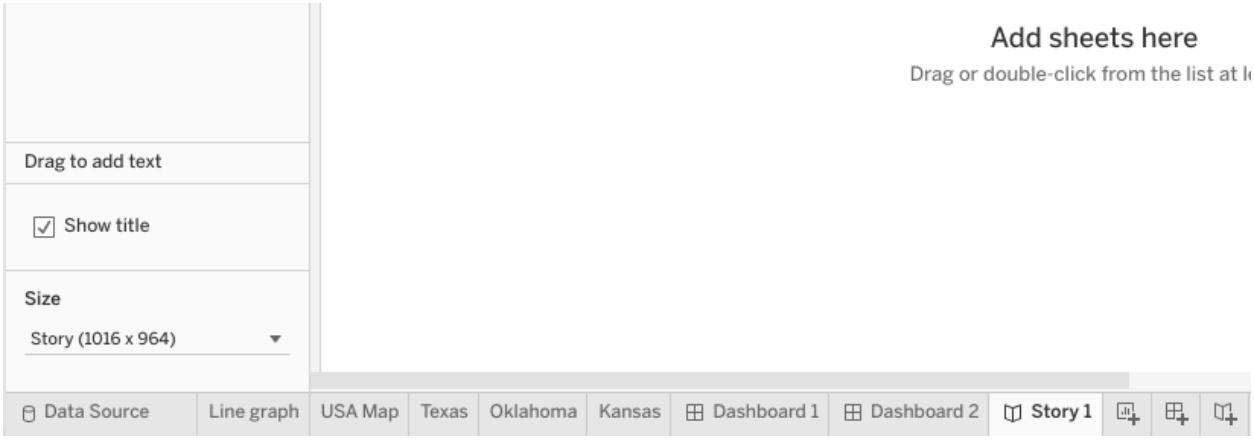
- Click on “New Dashboard” at bottom of page.
- In the dashboard, embed the three state worksheets.



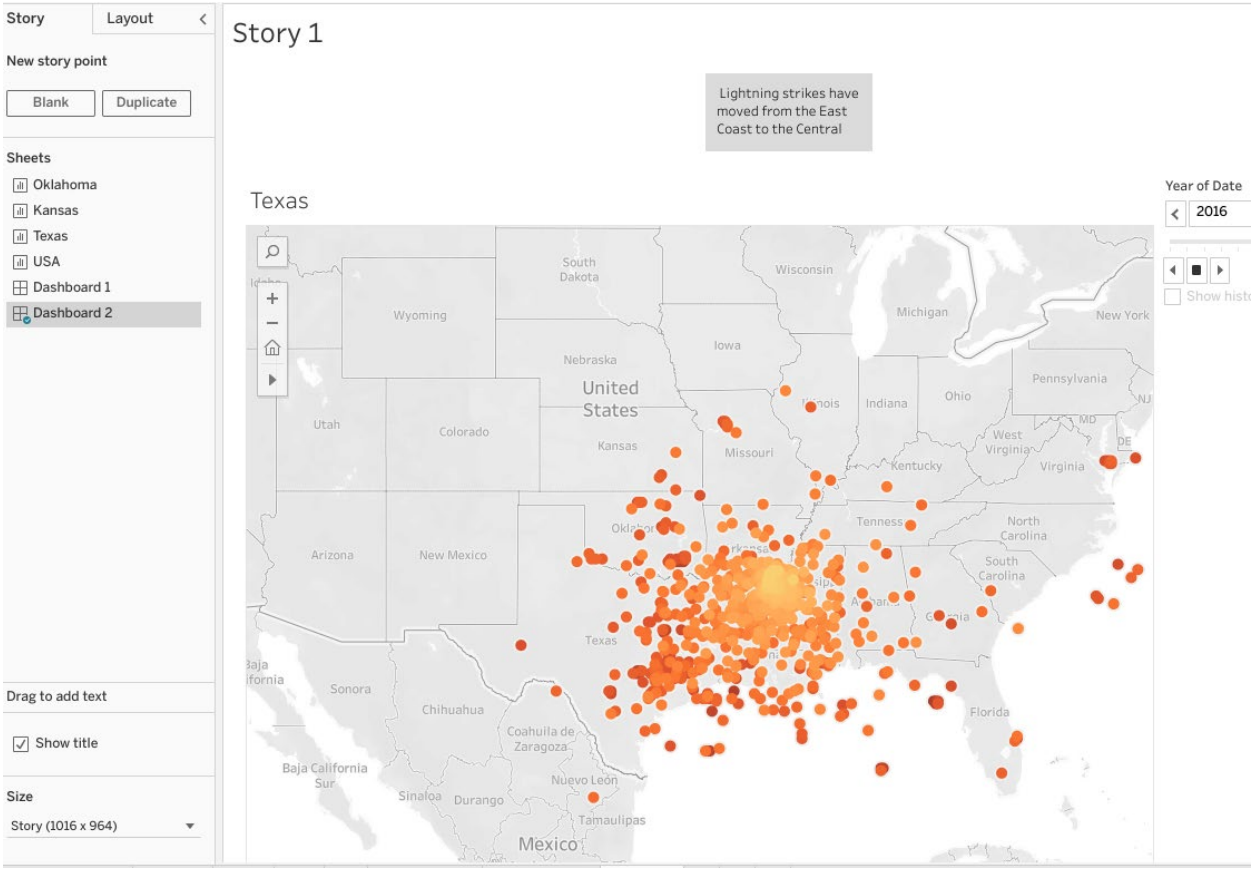
- Create another dashboard.
- Place USA map in it.



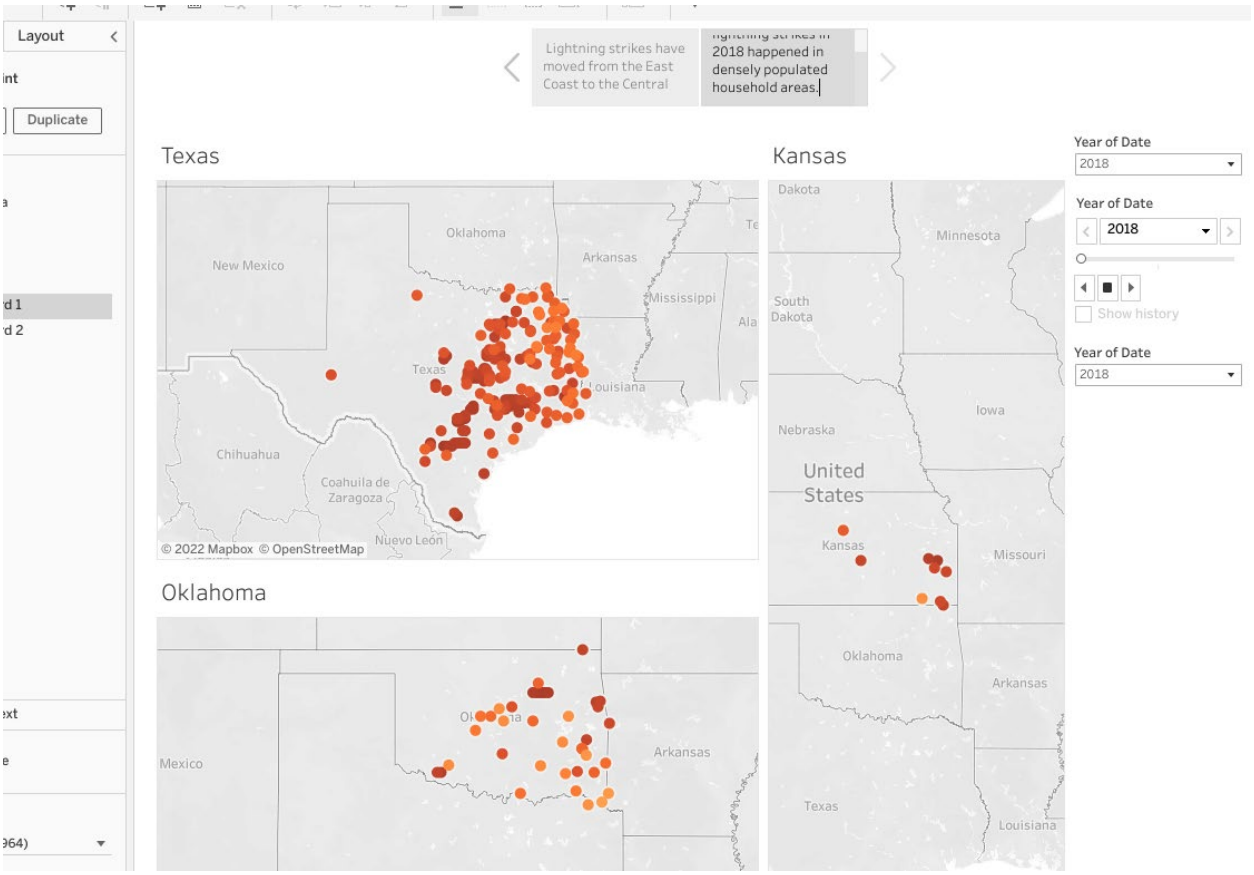
- Create a new STORY at the bottom of the Tableau page.



- On the first page, select the line graph dashboard.
- Fill in the caption with “Lightning strikes in the U.S. have increased 50% over the last decade.”
- Go to page 2 of the story.
- Select USA dashboard for Page 2.
- In caption type, “Lightning strikes have moved from the East Coast to the Central Mainlands over the last decade.”



- Go to page 3 of Story.
- Select the last dashboard with three states on it.
- In caption type, “Most number of lightning strikes in 2018 happened in densely populated household areas.”
- And now you’ve created a story using dashboards and worksheets as building blocks!


Layout
Duplicate
Lightning strikes have moved from the East Coast to the Central
2018 happened in densely populated household areas.
Texas
Kansas
Oklahoma
Year of Date
2018
Show history
New Mexico
Oklahoma
Arkansas
Mississippi
Texas
Louisiana
Chihuahua
Coahuila de Zaragoza
Nuevo León
© 2022 Mapbox © OpenStreetMap
Dakota
Minnesota
South Dakota
Iowa
Nebraska
United States
Kansas
Missouri
Oklahoma
Arkansas
Texas
Louisiana
Mexico

The top five data visualization resources
Top five data visualization resources for data professionals

You've gotten a lot of experience in data visualization by now, which will be instrumental in your career as a data professional. As you step into new roles in the data industry, it is important to keep up with data visualization trends by participating in visualization ("viz") communities and accessing viz resources.

Below you'll find the top five data viz resources for data professionals. At each site you'll find amazing training resources and bustling data viz communities ready to help you maintain and improve on the skills you've already acquired. Along with improving your skills, these sites and communities can be a useful resource for maintaining a working knowledge of data viz ethics and accessibility.

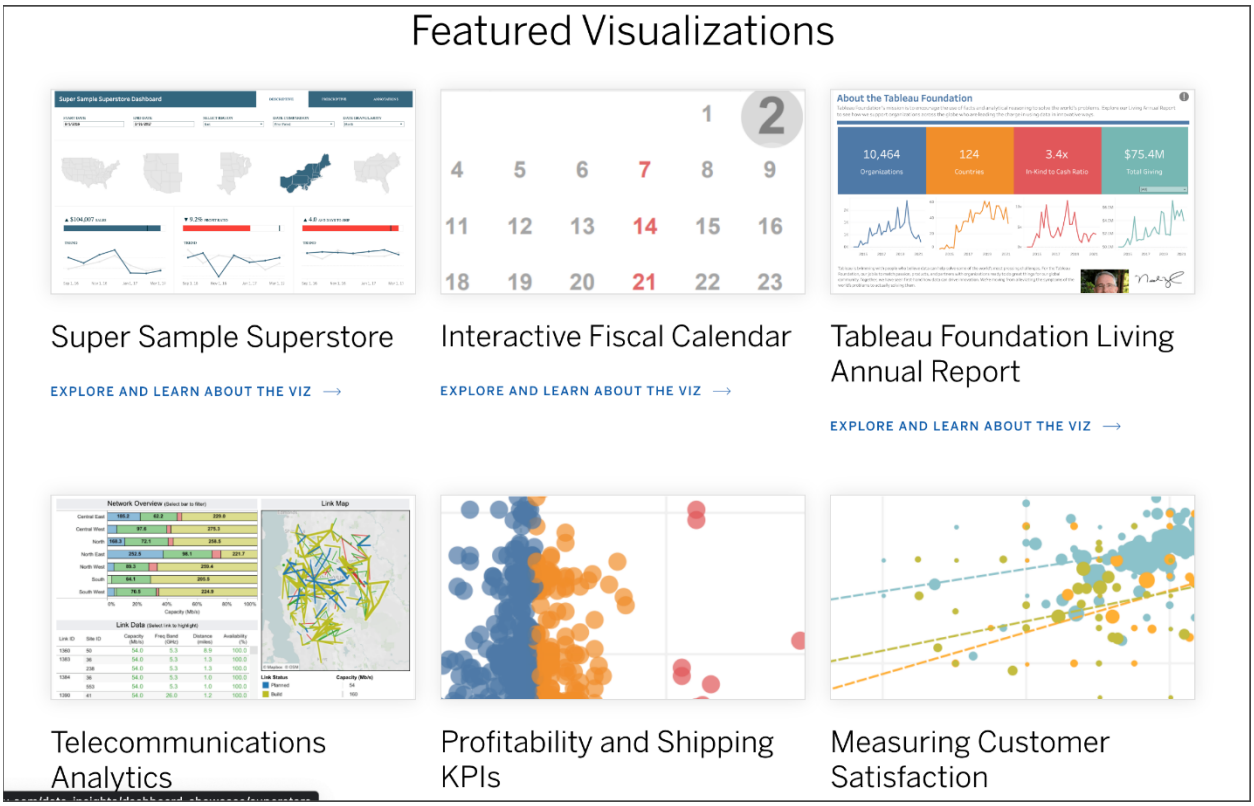
Take time to explore each resource. In particular, learn what they offer for training, education, and community engagement.

"Where can I find training material and sample visualizations to help me with my own data visualizations?"

Tableau

You're already quite familiar with Tableau's free browser software: **Tableau Public**. Of course, they offer a variety of software packages that appeal to every data professional, from the daily power user to the occasional presenter.

But did you know that Tableau also offers a comprehensive education and tutorial platform focused on training both software users and class leaders? In their Resources tab, Tableau offers video tutorials, e-learning courses, and webinars to help every level of user improve their data visualizations. Along with articles, blogs, and white papers, Tableau has a huge collection of sample visualizations (including Viz of the Day, which showcases some of the best Tableau vizzes) that is meant to encourage and empower data pros at every level to turn their dull datasets into effective storytelling tools.



"Where can I find a community of data professionals dedicated to building intelligent data visualizations for businesses? "

PowerBI

What PowerBI is probably best known for is its popular software product used by businesses and corporations worldwide for data visualization. Not only do they offer a free online version of their premium product, PowerBI, they also provide comprehensive guided learning and online workshops for teaching users of its products to create powerful data visualizations.

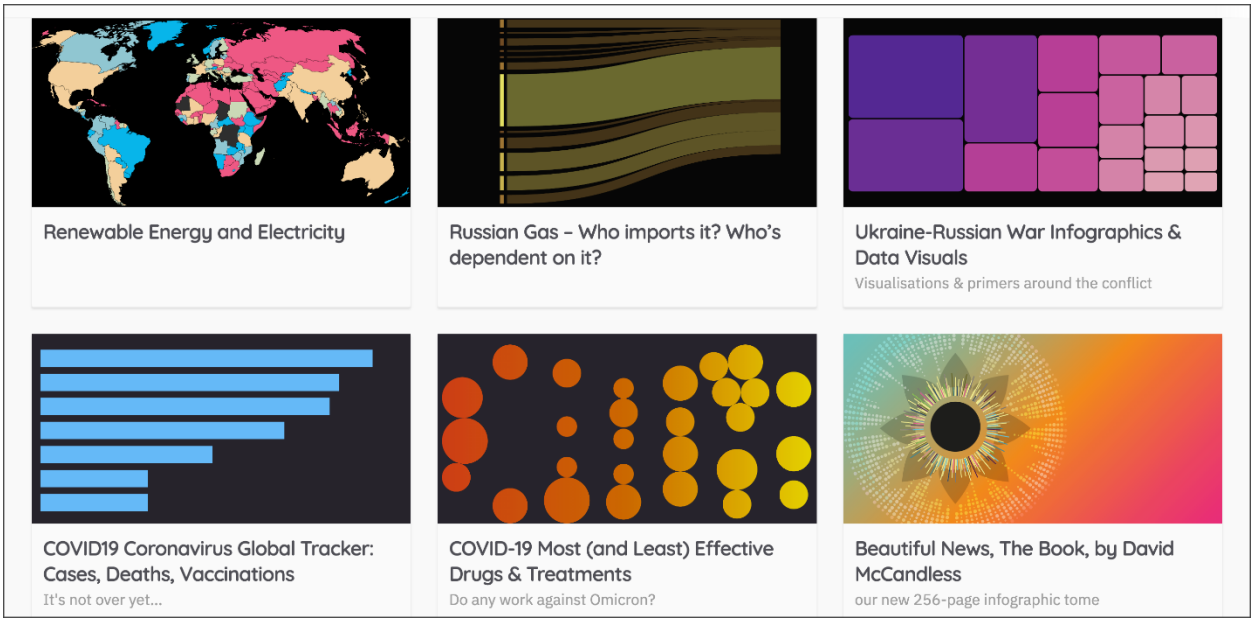
A growing number of data pros are also part of their PowerBI data visualization community. Here, users collaborate and share insight about the most effective ways to build data visualizations for businesses and organizations of every kind all over the world, like The Associated Press, Nokia, Real Madrid Football Club, Toyota, UNICEF, Galadari Brothers, and Eurobank, to name a few.

"Where can I find data visualizations paired with the datasets they are built from? "

Information is Beautiful

Founded by author and data visualization expert David McCandless, the Information is Beautiful platform is a gathering place for data pros who want to help people make clearer, more informed decisions in the world.

The site is a treasure trove of data visualizations on a host of topics. The Information is Beautiful team also has blogs, newsletters, and live and online workshops, which aim to help data professionals improve their craft. Beyond the powerful visualizations and graphs on their website, Information is Beautiful focuses on transparency by sharing all of the datasets on which they base their vizzes.

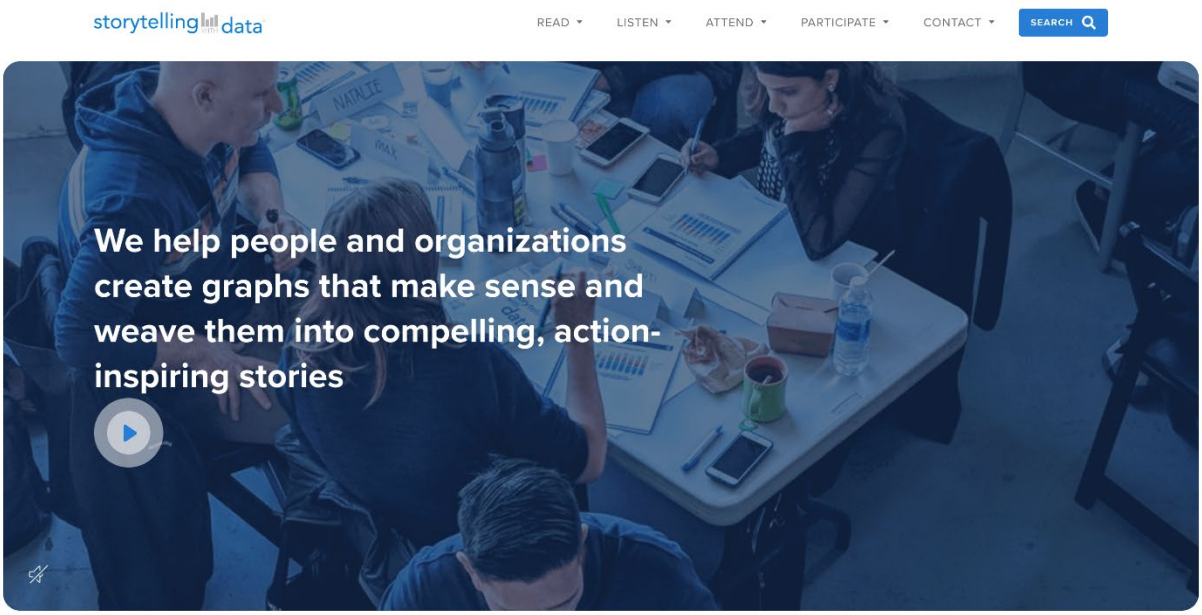


"Where can I go to learn more about communicating my data visualizations effectively?"

Storytelling with Data

As evidenced by their name, Storytelling with Data is an organization focused on communicating with concise data that informs change. They provide training, workshops, and tools for the craft that combines both data visualization and storytelling.

Storytelling with Data has a particularly engaged community, where data pros gather to learn, practice, ask for feedback, and help others. Though they don't have their own dedicated software for designing vizzes, they have a myriad of tiered resources for training, office hours, and consulting.



"Where can I go to find out more about data visualizations created in Python?"

Python Graph Gallery

If you are looking to improve your skill in plotting data visualizations in Python, the Python Graph Gallery is a fantastic website to visit. This site created by Yan Holtz includes a wide range of different types of visualization plots, like bar graphs, line charts, times series charts, geographic maps, and many more. Along with each visualization, you will also find the accompanying Python code used to plot the viz.

The site is organized first by plot type, but can also be sorted by the visualization Python package used, seaborn, matplotlib, or plotly. There is also a space for general knowledge tips and tricks, and a section titled Dataviz Inspiration, which can help you understand what Python is capable of.

Key Takeaways

Your data visualization education doesn't end when you complete this program. There are amazing resources available that are consistently being updated with the latest trends and most up-to-date principles regarding the design, ethics, accessibility, and communication of data visualizations.

Follow-along guide: Present like a pro with Tableau

This document includes detailed instructions for how to perform the data visualizations described in the video "Present like a pro with Tableau".

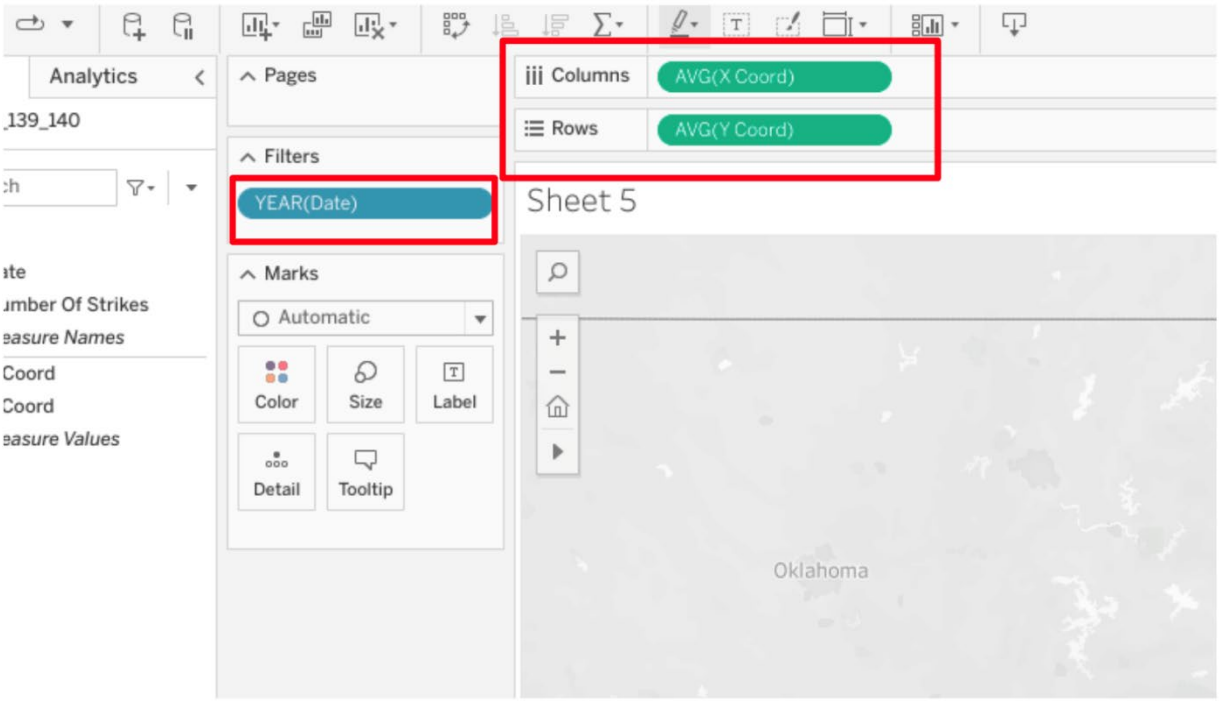
The following guide points out areas of the video that may require adjustment. These resource guides can also serve as a set of usability reminders for you to recall when using Tableau in your future career.

Instructions

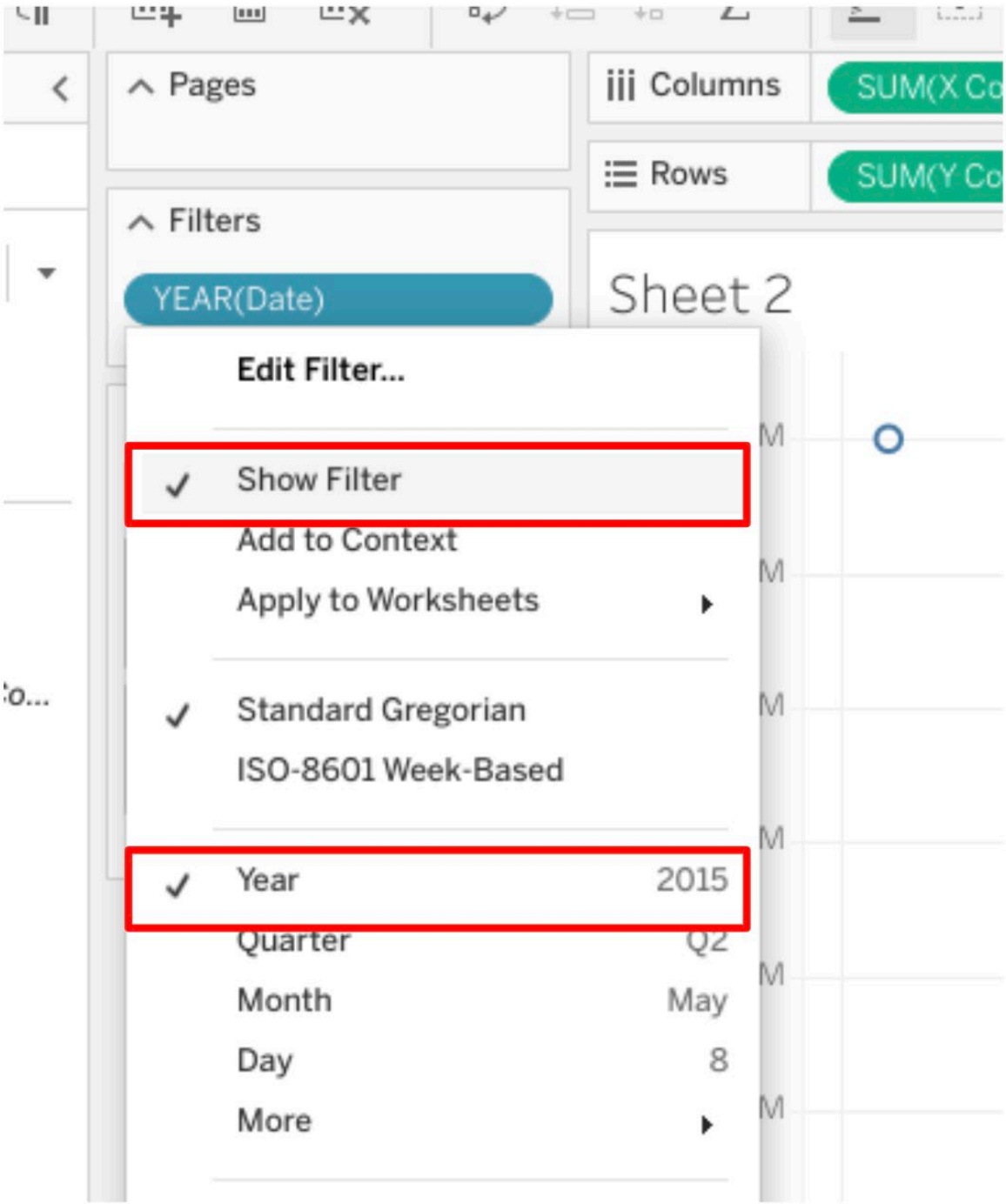
- Go to https://public.tableau.com/s/
- Since you've already set up your Tableau Public profile, all you need to do is log in and select **Web Authoring** under **Create** in the navigation bar.
- Select the appropriate CSV file provided in the instructions. Use the same data source as the previous video. The dataset you'll use with this instructional video is: tableau_main_2009_to_2018.csv
    - Please be aware that when you download the zip file folder provided, the computer automatically names that zip file folder with a long string of numbers and letters. You have to open that folder and then upload the individual files that are named correctly and match what's shown in the video
- Click on NEW WORKSHEET.

Note: Please allow several minutes for data import into a new worksheet.

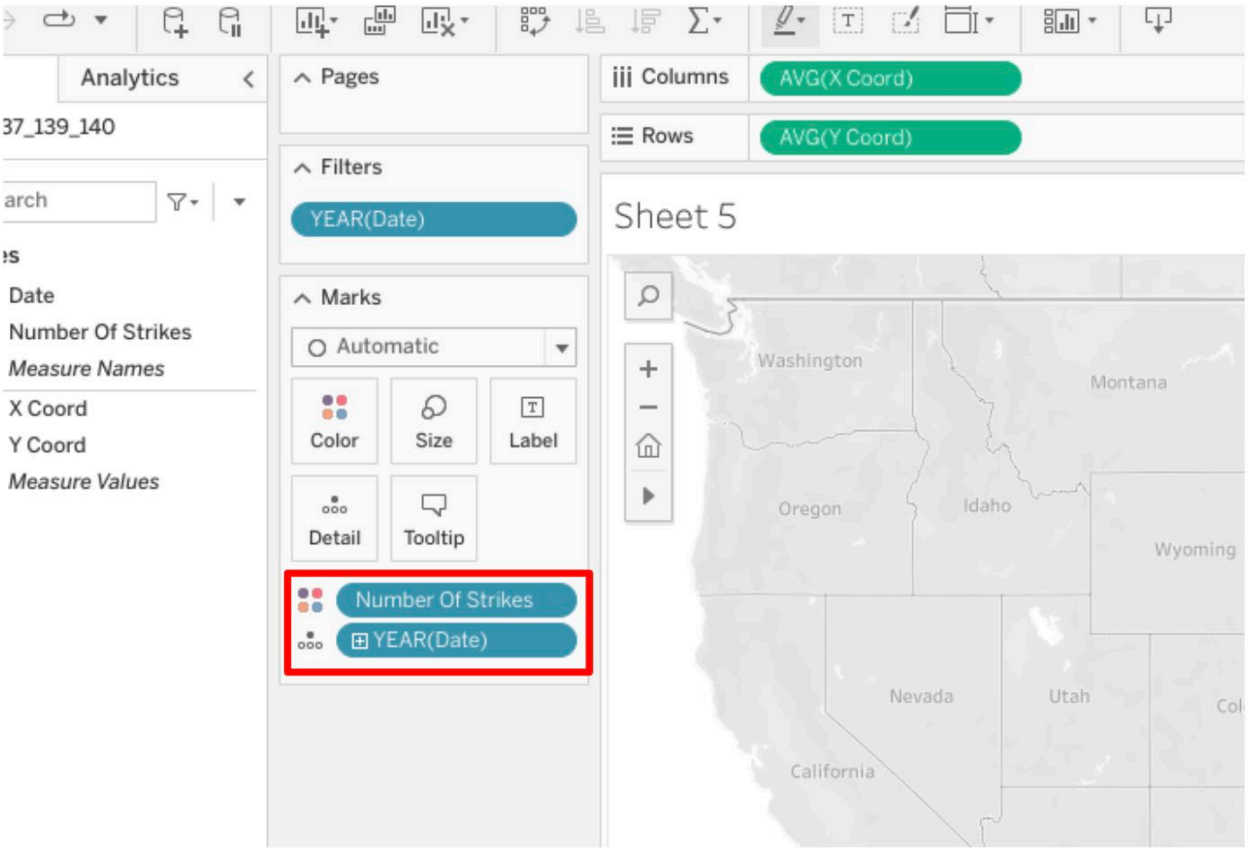
- Drag X coord to Column field. Drag Y coord to Row field.
- Drag DATE to FILTERS.



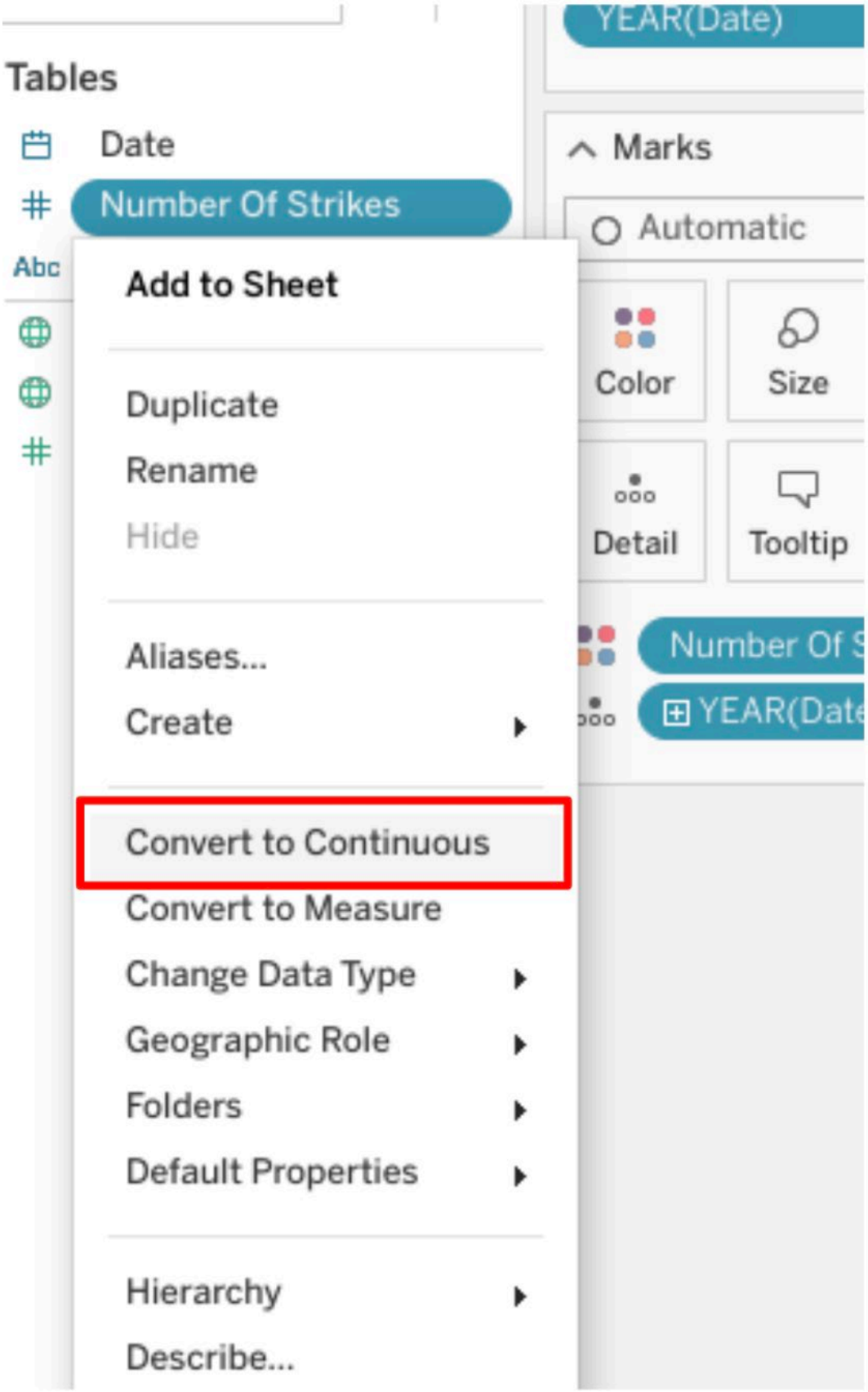
- Click on DATE dropdown, Select YEAR.
- Click on DATE dropdown, select SHOW FILTER.



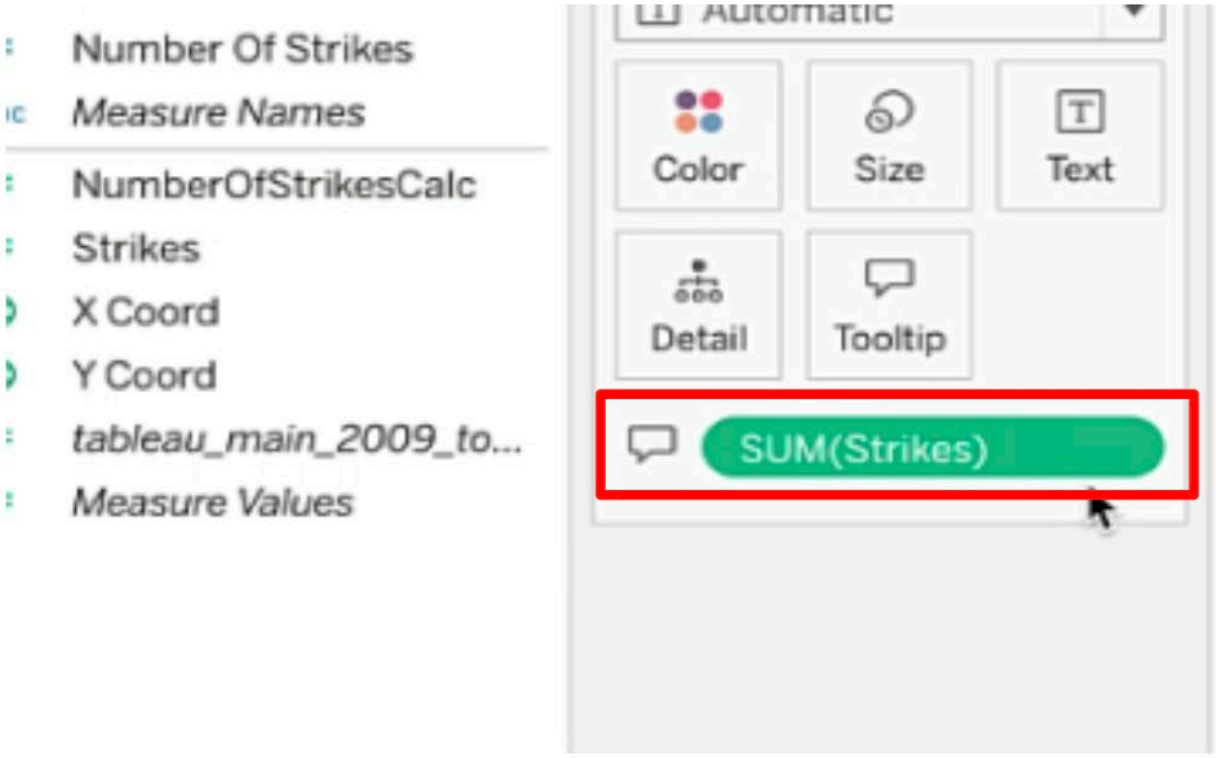
- Drag NUMBER of STRIKES into both the COLOR square and DETAIL square inside the MARKS box.



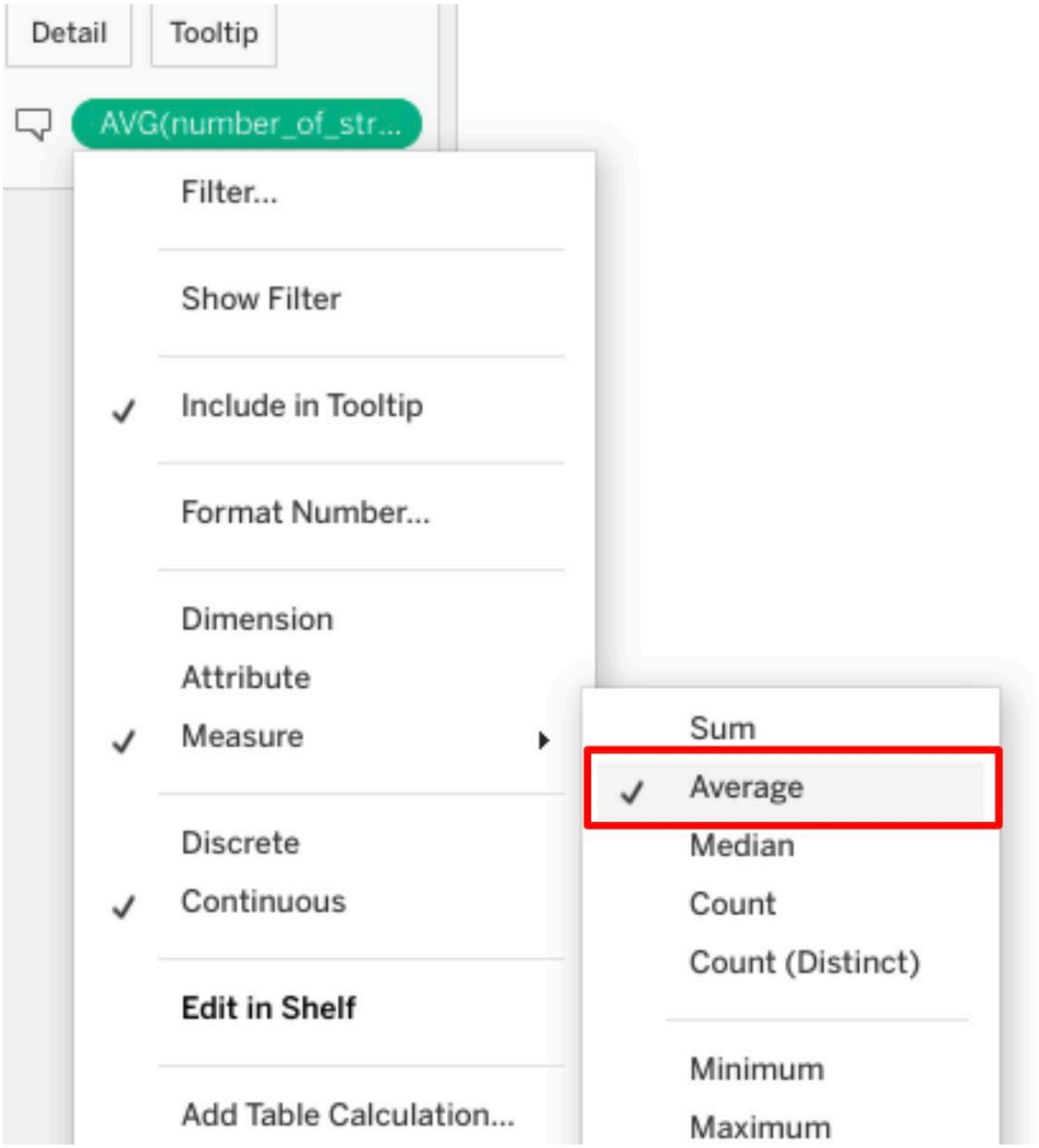
- Click on dropdown of NUMBER OF STRIKES, select CONVERT to CONTINUOUS.


YEAR(Date)
Tables
Date
Number Of Strikes
Abc
Marks
Automatic
Color
Size
Detail
Tooltip
Number Of S
YEAR(Date
Add to Sheet
Duplicate
Rename
Hide
Aliases...
Create
Convert to Continuous
Convert to Measure
Change Data Type
Geographic Role
Folders
Default Properties
Hierarchy
Describe...

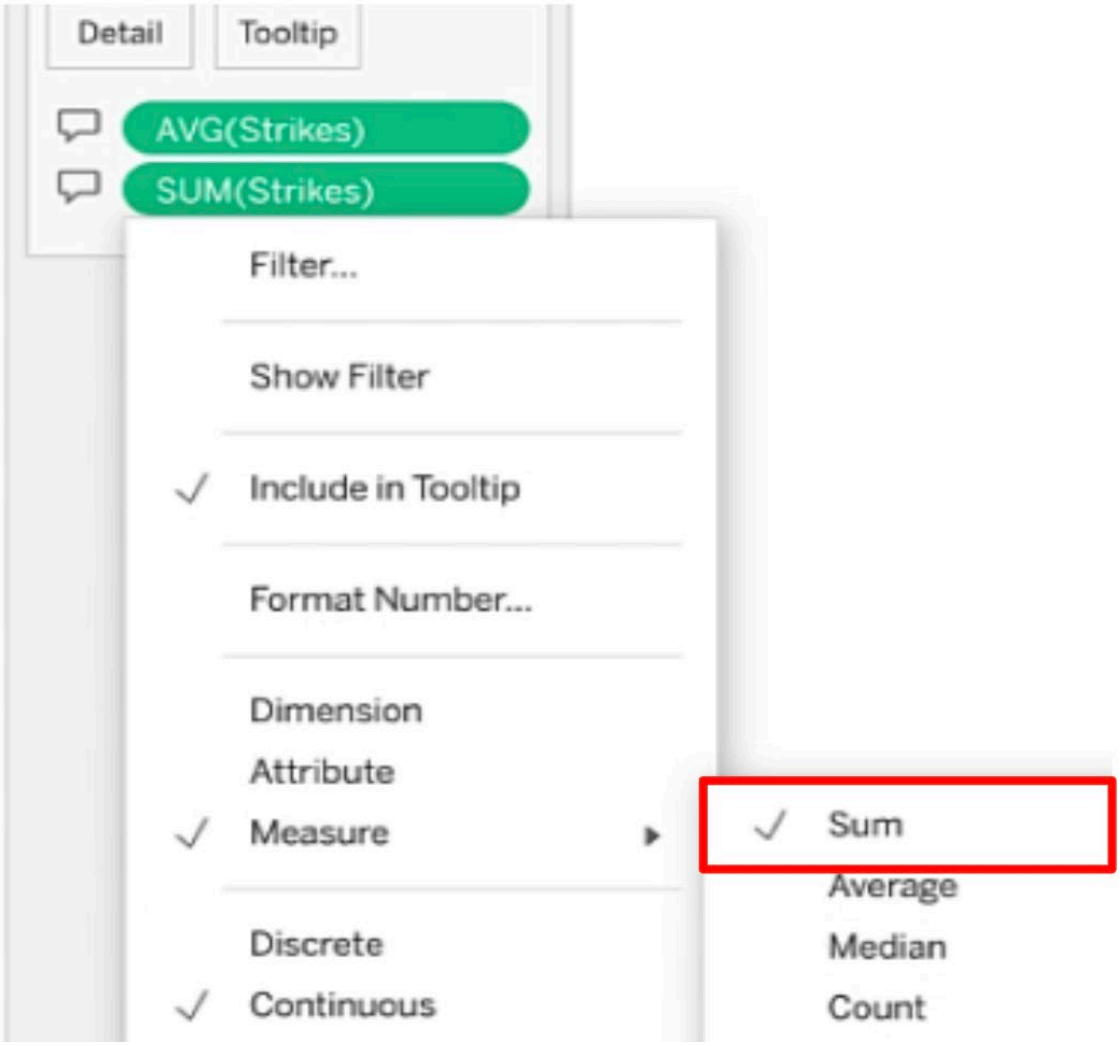
- Click on NEW WORKSHEET.
- Click dropdown of NUMBER of STRIKES field.
- Select CREATE and CALCULATED FIELD.
- TYPE “Strikes” for the title.
- Click OK.
- Drag calculation into TOOLTIP field.



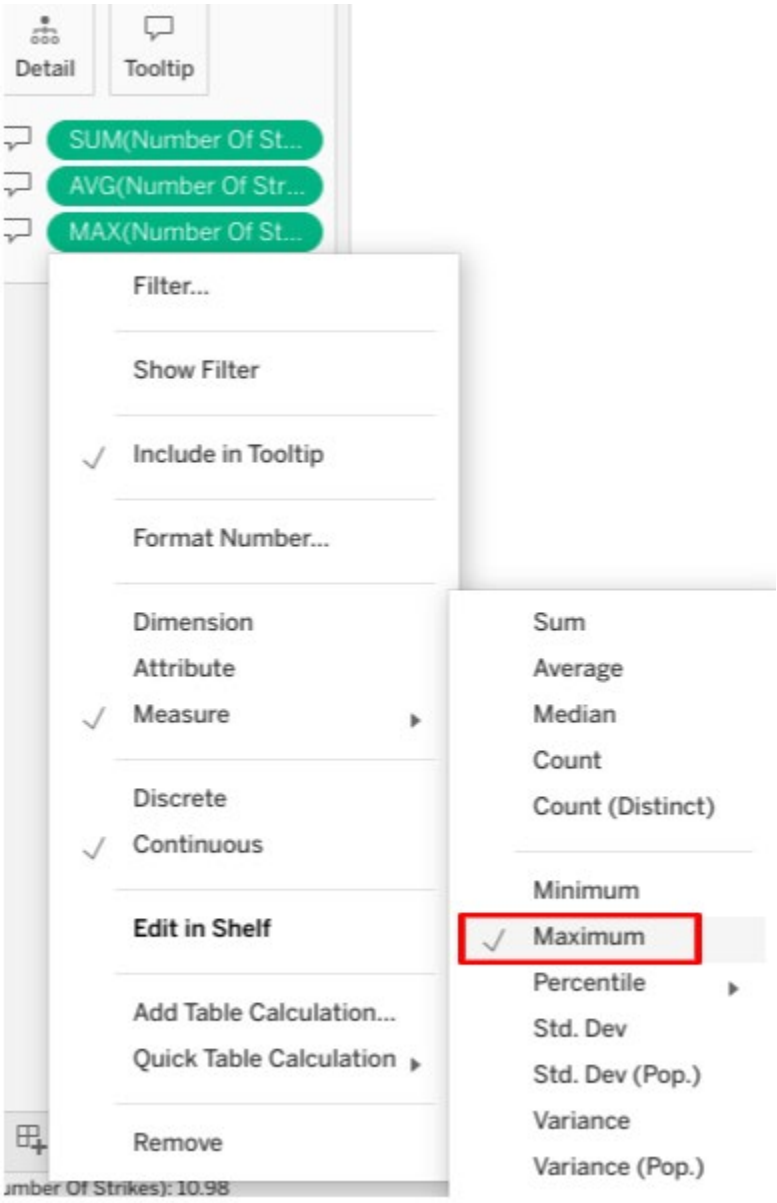
- In the Strikes drop down, Select MEASURE, then AVERAGE.



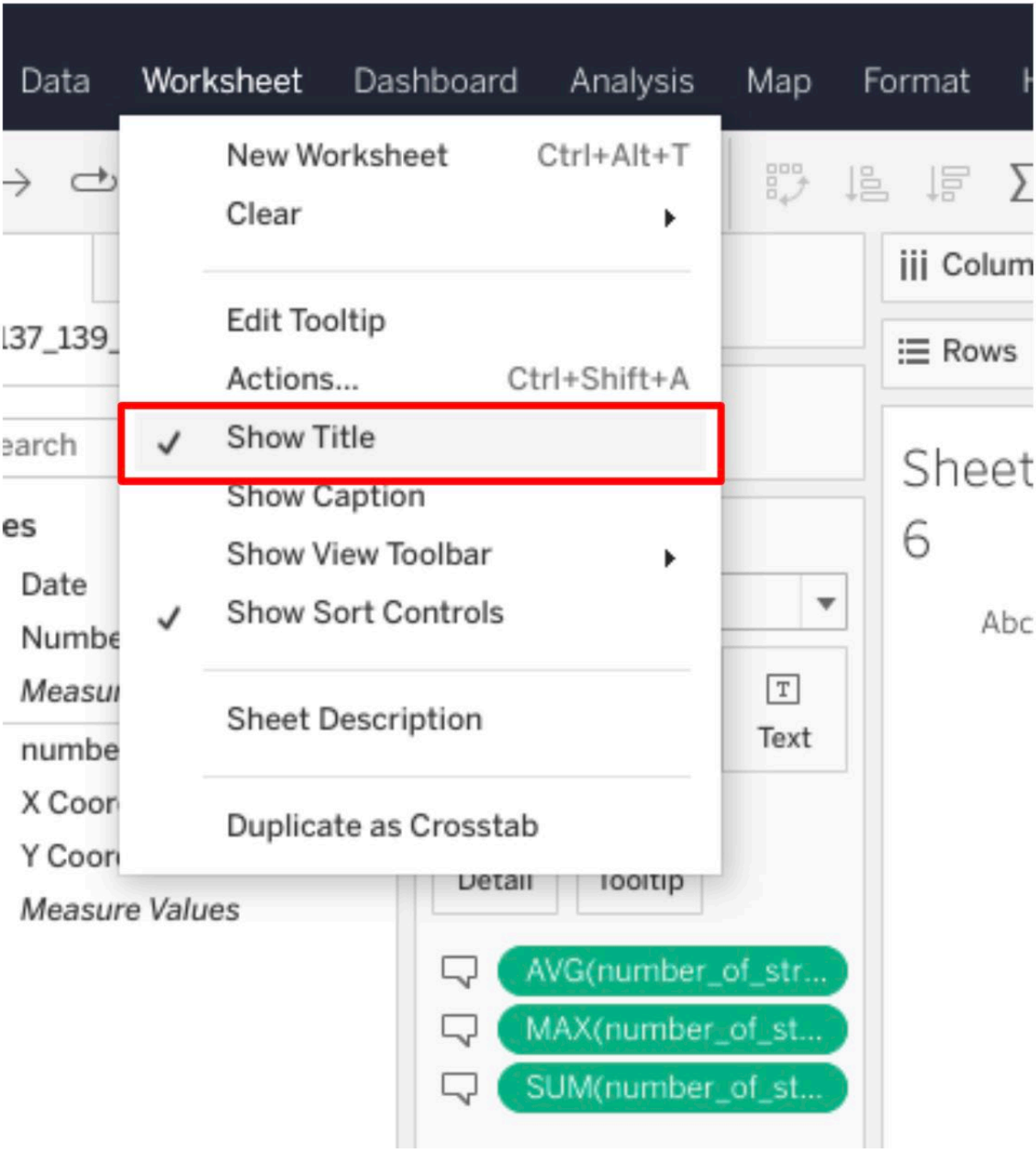
- Drag calculation field to TOOLTIP again. Select MEASURE, then SUM.



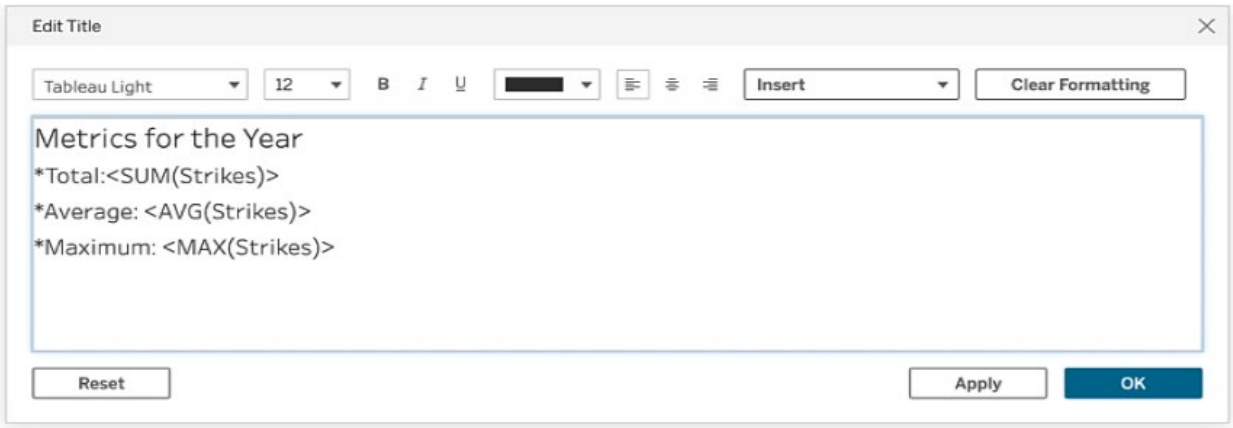
- Drag calculation to TOOLTIP, select MEASURE, then MAXIMUM.



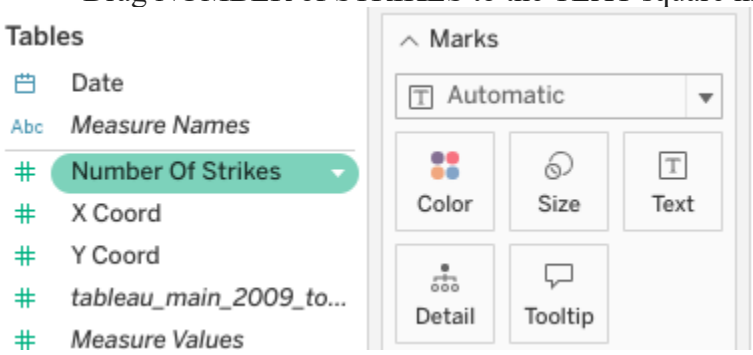
- Click on WORKSHEET and select SHOW TITLE.



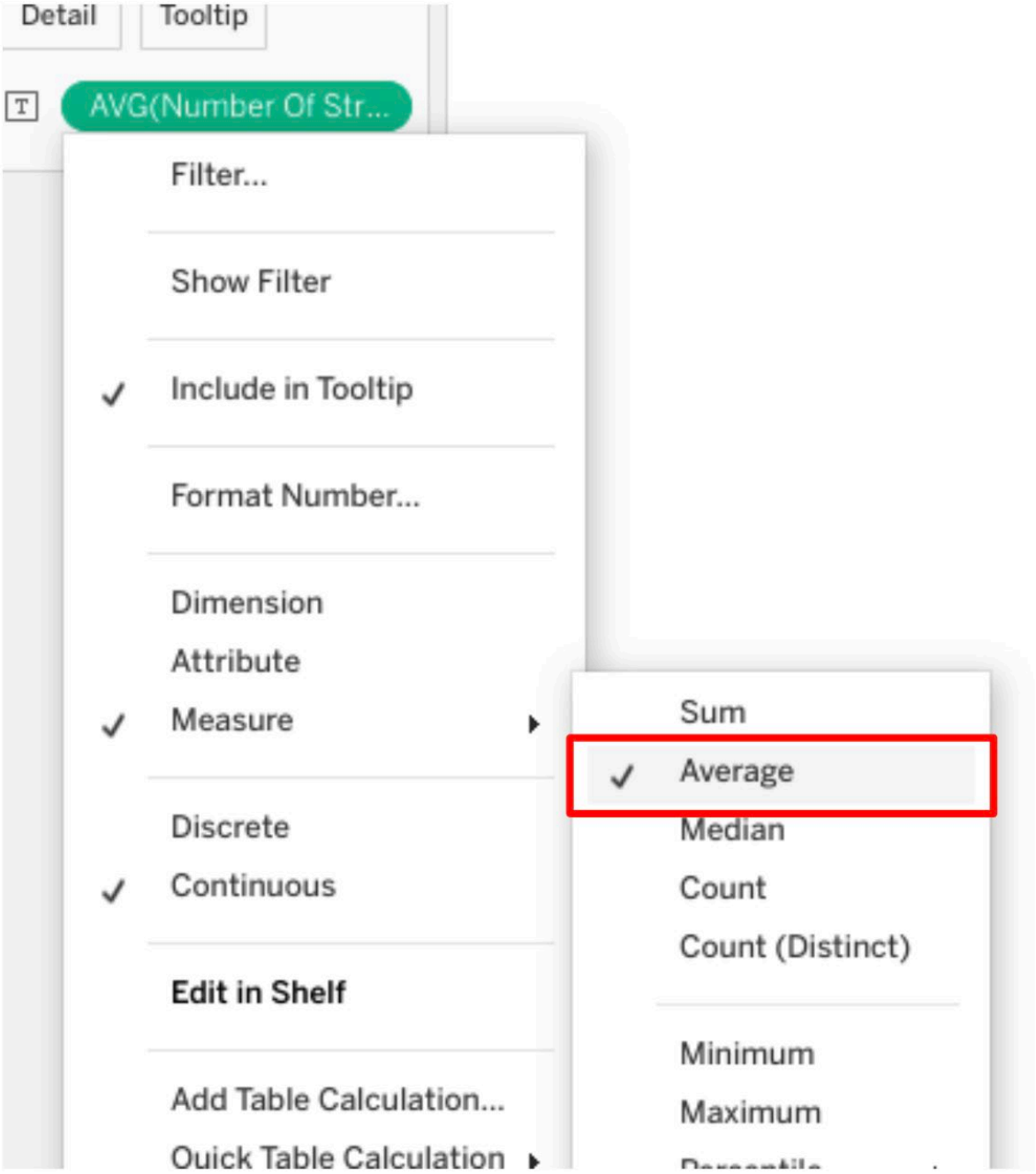
- Type in ‘Metrics for the Year’ in title field.
- Then add the following beneath it:
  - *Total: <SUM(Strikes)>
  - * Average: <AVG(Strikes)>
  - * Maximum: <MAX(Strikes)>’.
  - Click OK.



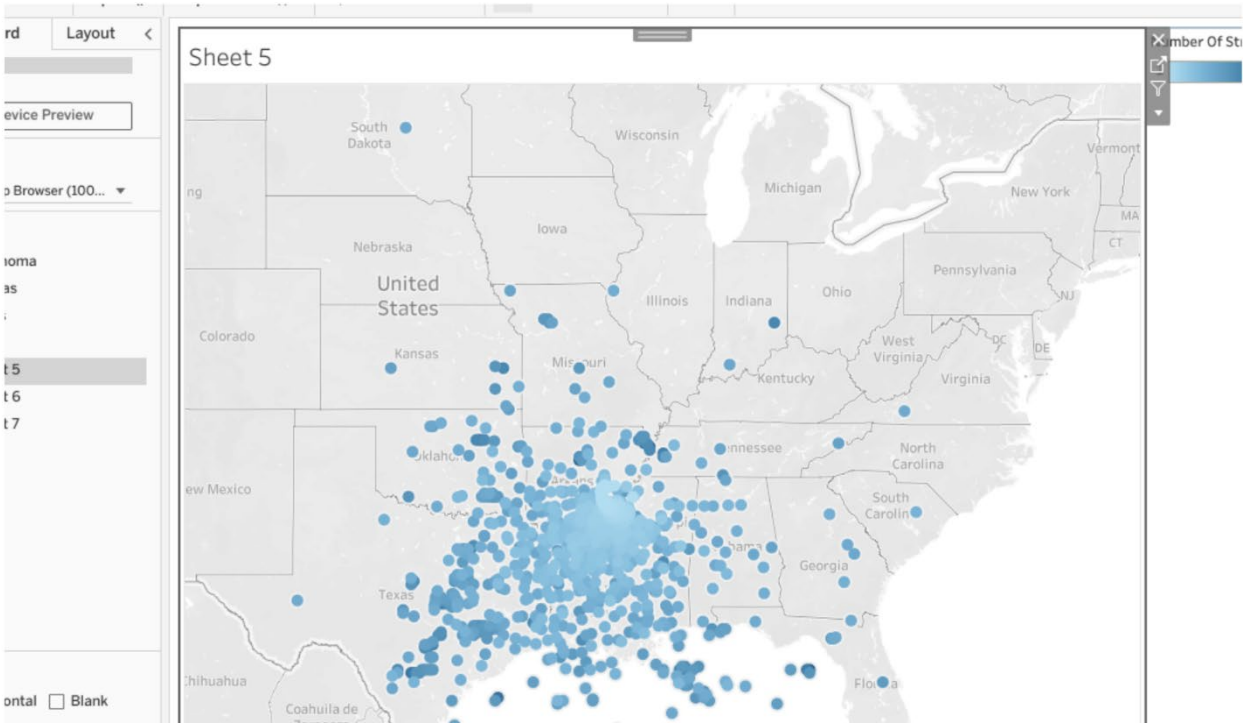
- Click on NEW WORKSHEET.
- Drag NUMBER of STRIKES to the TEXT square in the MARKS field.



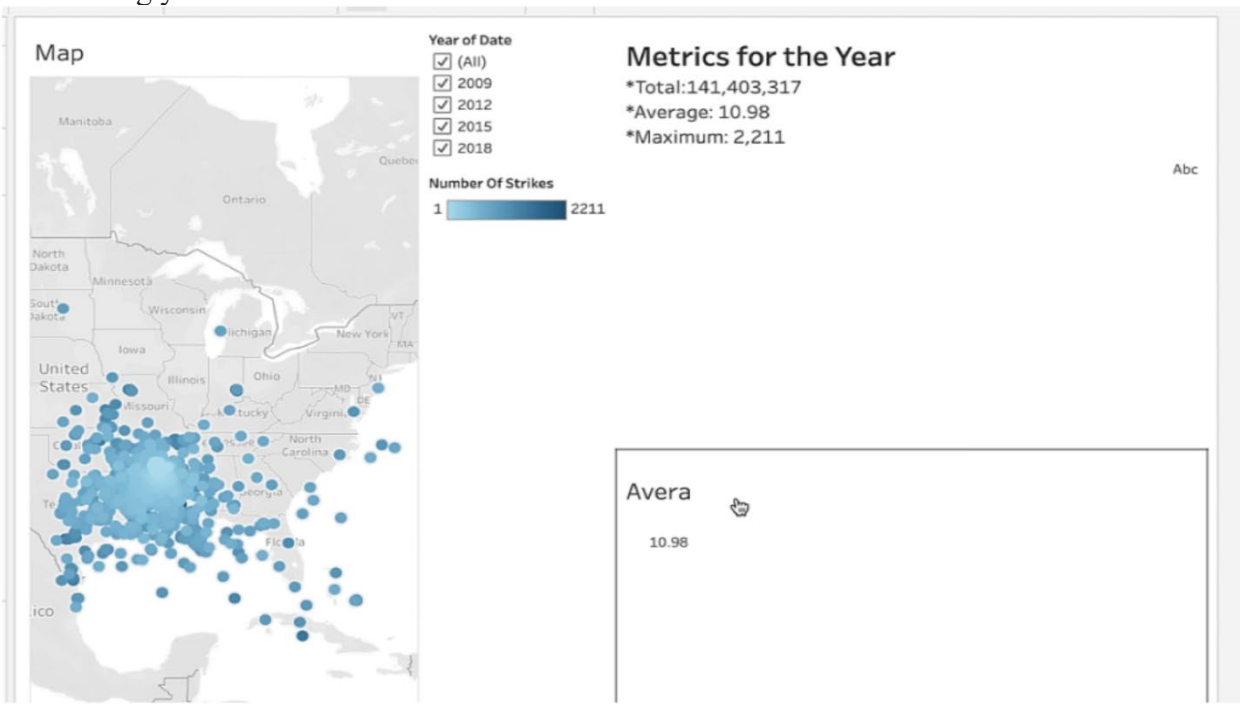
- From dropdown, select MEASURE and AVERAGE.



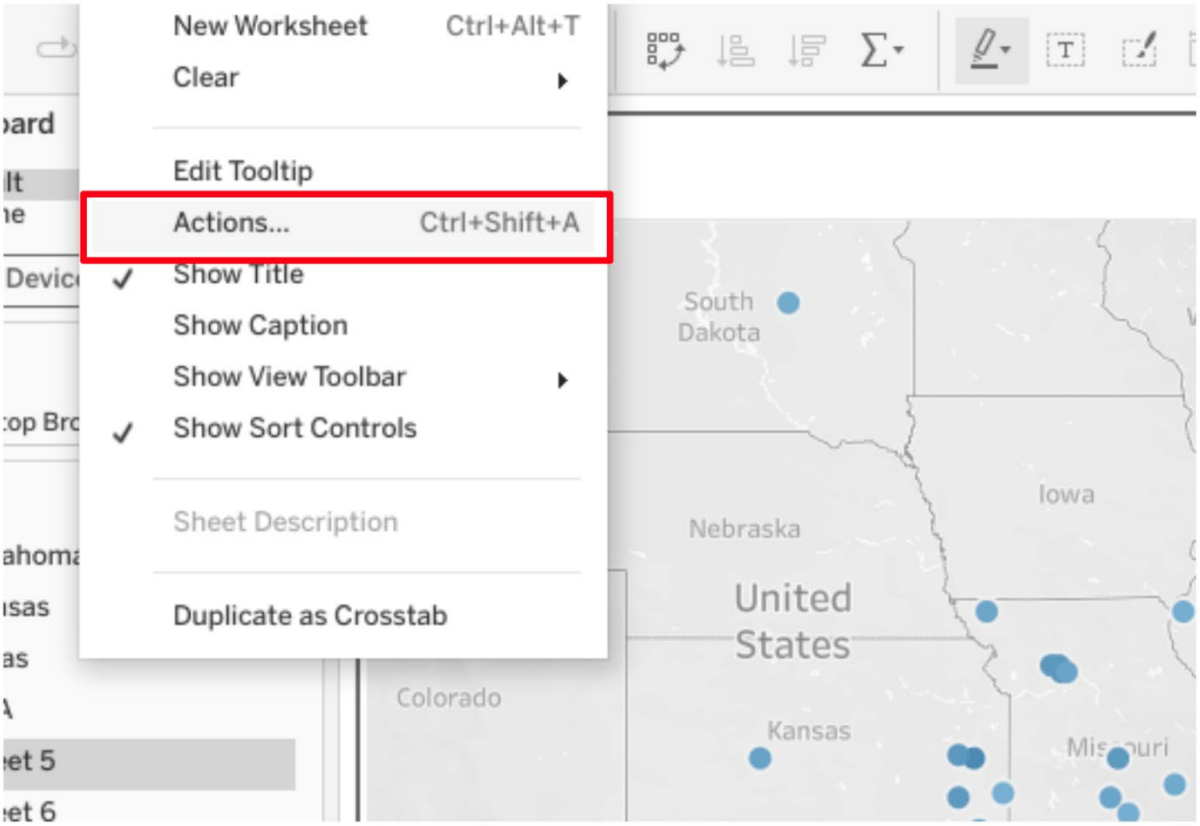
- Click on NEW DASHBOARD.
- Drag USA map worksheet to empty dashboard.



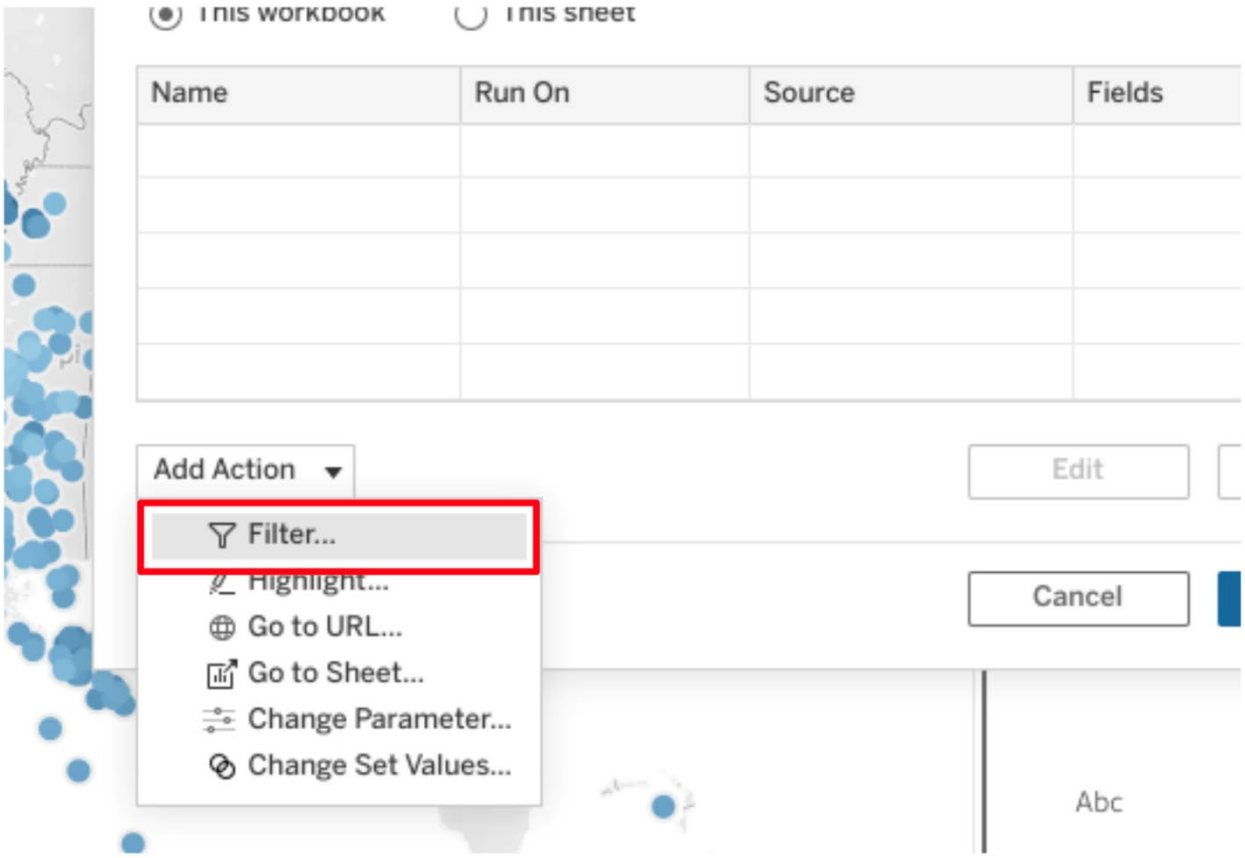
- Drag your other two worksheets to the dashboard.



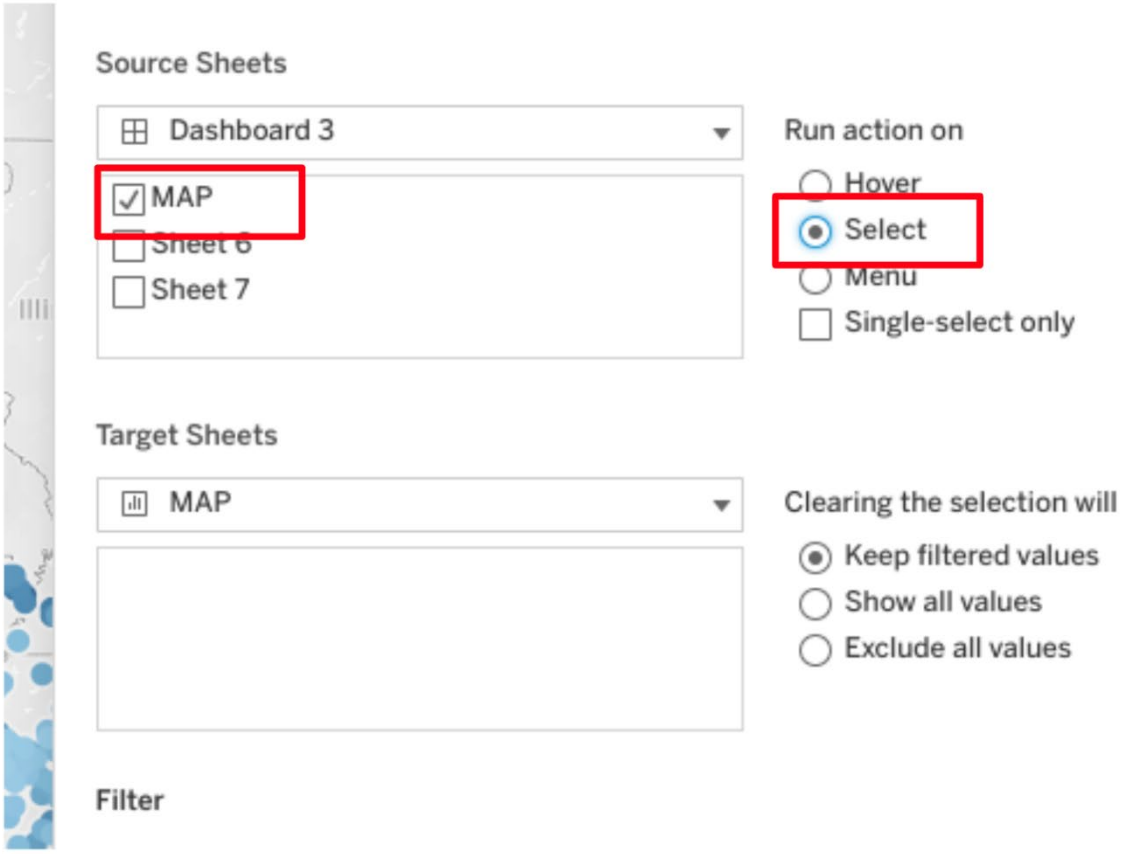
- Click map, then select WORKSHEET and ACTIONS.



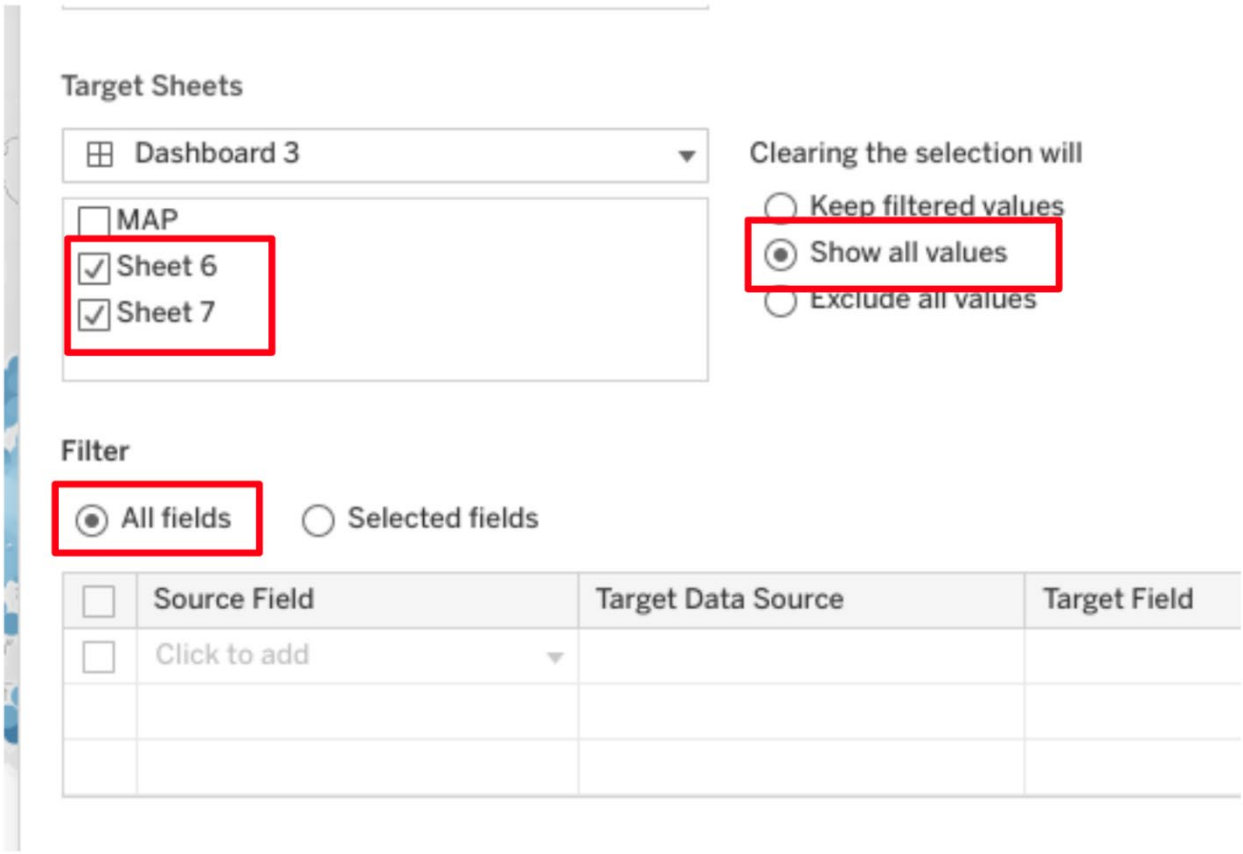
- Select ADD ACTION and FILTER.



- Under SOURCE SHEETS, select the dashboard you created from the dropdown.
- Check the box for the map worksheet, and then choose SELECT under the RUN ACTION ON list.



- Next, under the TARGET SHEETS list, select dashboard from the drop down, then check box for the other two worksheets you created: Map and Metrics.
- Select SHOW ALL VALUES.
- Select ALL FIELDS, under Filter.
- Click OK.



- Type in ‘Interactive Filter’ in Legend title field.
- Type in ‘Location Metrics’ in title field.
- Now you’ve created an interactive geographic dashboard that adjusts metrics each time a year or location is changed or selected. Well done!

Activity Exemplar: Build an interactive dashboard in Tableau Public

Here is a completed exemplar along with an explanation of how the exemplar fulfills the expectations for the activity.

Completed Exemplar

To review the exemplar for this course item, click the following link .

Link to exemplar: [Impact of Holidays on Seoul Bike Rentals in 2017 - 2018]

Assessment of Exemplar

Compare the exemplar to your completed activity. Review your work using each of the criteria in the exemplar. What did you do well? Where can you improve? Use your answers to these questions to guide you as you continue to progress through the course.

Note: *The exemplar represents one possible way to complete the data visualization. Yours will likely differ in certain ways. What's important is that your data visualization meets the business scenario criteria.*

How the exemplar meets the criteria

Notice how the exemplar posted to Tableau Public uses the detail from the given scenario to adhere to the following guidelines:

- The data dashboard includes two dynamic worksheets.
- The dashboard has divided the data into two different bar graphs: by Weekday and by Adjacent Day.
- One worksheet divides data into dates and months
- This worksheet is the "By Adjacent Day" worksheet.
- Another worksheet is divided into dates for each day of the week. This worksheet is the "By Weekday" worksheet.
- The dashboard, worksheets, and filters are properly labeled.
- Holidays are shown in a clearly contrasting color. Holiday data is in orange, while non-holiday data is in blue.
- The filters on the dashboard sidebar filter the desired data.
- There are two filters shown and one legend.
- One filter connects to the "By Adjacent Day" data and allows the data to be filtered by month.
- Another filter connects to the "By Weekday" data and allows the data to filter by day of the week.
- The legend shows that holidays are orange and non-holidays are blue.

Glossary

Advanced Data Analytics

Terms and definitions from Course 3

A

Action: A Tableau tool to help an audience interact with a visualization or dashboard by allowing control of selection

B

Bias: In data structuring, organizing data results in groupings, categories, or variables that are misrepresentative of the whole dataset

Bin: A segment of data that groups values into categories

Box plot: A data visualization that depicts the locality, spread, and skew of groups of values within quartiles

C

Categorical data: Data that is divided into a limited number of qualitative groups

Cleaning: The process of removing errors that might distort your data or make it less useful; one of the six practices of EDA

Collective outliers: A group of abnormal points, following similar patterns and isolated from the rest of the population

Contextual outliers: Normal data points under certain conditions but become anomalies under most other conditions

Continuous: A mathematical concept indicating that a measure or dimension has an infinite and uncountable number of outcomes

CSV file: A simple text file that can be easy to import or store in other softwares, platforms, and databases

D

Database (DB) file: A file type used to store data, often in tables, indexes, or fields
Data ethics: Well-founded standards of right and wrong that dictate how data is collected, shared, and used

Data governance: A process for ensuring the formal management of a company's data assets

Data source: The location where data originates
Data visualization: A graph, chart, diagram, or dashboard that is created as a representation of information

Deduplication: The elimination or removal of matching data values in a dataset

Dimensions: Qualitative data values used to categorize and group data to reveal details about it

Discovering: The process data professionals use to familiarize themselves with the data so they can start conceptualizing how to use it; one of the six practices of EDA

Discrete: A mathematical concept indicating that a measure or dimension has a finite and countable number of outcomes

Docstring: (Refer to **documentation string**)

Documentation string: A group of text that explains what a method or function does; also referred to as a "docstring"

Dummy variables: Variables with values of 0 or 1 that indicate the presence or absence of something

E

Exploratory data analysis (EDA): The process of investigating, organizing, and analyzing datasets and summarizing their main characteristics, often by employing data wrangling and visualization methods; the six main practices of EDA are: discovering, structuring, cleaning, joining, validating, and presenting

Extracting: The process of retrieving data out of data sources for further data processing

F

Filtering: The process of selecting a smaller part of a dataset based on specified values and using it for viewing or analysis
First-party data: Data that was gathered from inside your own organization

G

Global outliers: Values that are completely different from the overall data group and have no association with any other outliers

Grouping: The process of aggregating individual observations of a variable into groups

H

Heatmap: A type of data visualization that depicts the magnitude of an instance or set of values based on two colors

Histogram: A data visualization that depicts an approximate representation of the distribution of values in a dataset

Hypothesis: A theory or an explanation, based on evidence, that has not yet been refuted

I

Info(): Gives the total number of entries, along with the data types—called Dtypes in pandas—of the individual entries
Input validation: The practice of thoroughly analyzing and double-checking to make sure data is complete, error-free, and high quality

Int64: A standard integer data type, representing numbers somewhere between negative nine quintillion and positive nine quintillion

J

Joining: The process of augmenting data by adding values from other datasets; one of the six practices of EDA

JSON file: A data storage file that is saved in a JavaScript format

L

Label encoding: Data transformation technique where each category is assigned a unique number instead of a qualitative value

M

Measures: Numeric values that can be aggregated or placed in calculations

Merging: A method to combine two (or more) different data frames along a specified starting column(s)
Missing data: A data value that is not stored for a variable in the observation of interest

N

Non-null count: The total number of data entries for a data column that are not blank

O

One-hot encoding: A data transformation technique that turns one categorical variable into several binary variables

Outliers: Observations that are an abnormal distance from other values or an overall pattern in a data population

P

PACE: A workflow data professionals can use to remain focused on the end goal of any given dataset; stands for plan, analyze, construct, and execute

Presenting: The process of making a cleaned dataset available to others for analysis or further modeling; one of the six practices of EDA

S

Second-party data: Data that was gathered outside your organization but directly from the original source
Set: A Tableau term for a custom field of data created from a larger dataset based on custom conditions

Slicing: A method for breaking information down into smaller parts to facilitate efficient examination and analysis from different viewpoints
Sorting: The process of arranging data into a meaningful order for analysis
Story: A Tableau term for a group of dashboards or worksheets assembled into a presentation

String: A sequence of characters and punctuation that contains textual information

Structuring: The process of taking raw data and organizing or transforming it to be more easily visualized, explained, or modeled; one of the six practices of EDA

T

Tableau: A data visualization software primarily used for presenting data to inform and improve businesses

Third-party data: Data gathered outside your organization and aggregated

V

Validating: The process of verifying that the data is consistent and high quality; one of the six practices of EDA