Exploratory data analysis on Azure Databricks

Exploratory data analysis (EDA) includes methods for exploring data sets to summarize their main characteristics and identify any problems with the data. Using statistical methods and visualizations, you can learn about a data set and analysis and inform what techniques to apply for data preparation.

What is EDA?

Exploratory Data Analysis (EDA) is the first important step in data science. It means looking at and visualizing data to:

- See its main features.
- Find patterns and trends.
- Spot anything unusual.
- Understand how different variables are related.

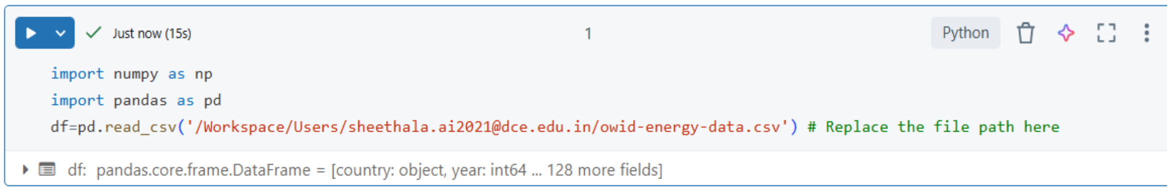
EDA gives you a clear understanding of your dataset so you can decide what to do next, like which statistical tests or machine learning models to use.**In Azure Databricks**, you can do EDA in notebooks using common Python libraries like:

- **NumPy** – For working with numbers, arrays, and math functions.
- **pandas** – For handling and analyzing structured data in DataFrames.
- **Plotly** – For making interactive and good-looking charts.
- **Matplotlib** – For making static, animated, or interactive plots.

Databricks notebooks also have built-in tools to explore data directly—like filtering tables, searching within them, zooming in on charts, and even using **Databricks Assistant** to help you write EDA code.

EDA Techniques

1. **Create a new notebook**
2. **Load CSV file**

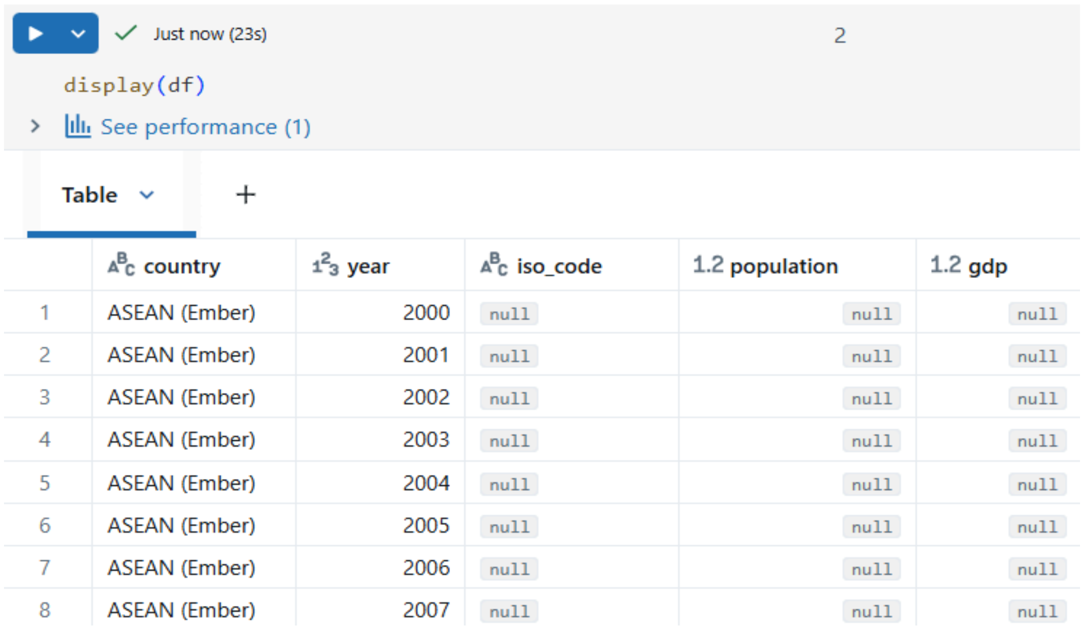


```
import numpy as np
import pandas as pd
df=pd.read_csv('/Workspace/Users/sheethala.ai2021@dce.edu.in/owid-energy-data.csv') # Replace the file path here
```

```
df:  pandas.core.frame.DataFrame = [country: object, year: int64 ... 128 more fields]
```

3.Understand the data

Understanding the basics of the dataset is crucial for any data science project. It involves familiarizing oneself with the structure, types, and quality of the data at hand. In a Azure Databricks notebook, you can use the `display(df)` command to display the dataset.

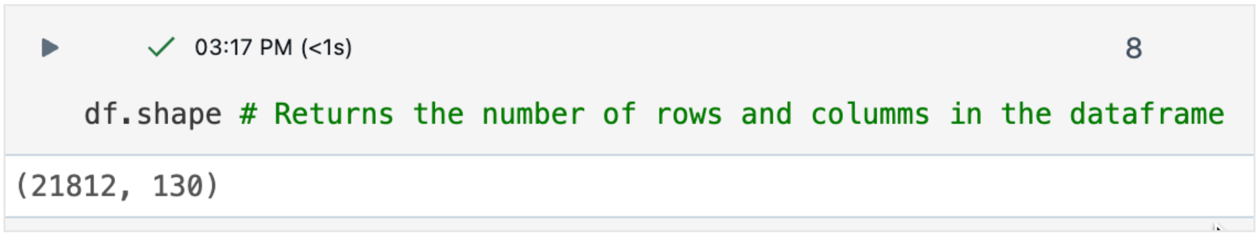


```
display(df)
```

	country	year	iso_code	population	gdp
1	ASEAN (Ember)	2000	null	null	null
2	ASEAN (Ember)	2001	null	null	null
3	ASEAN (Ember)	2002	null	null	null
4	ASEAN (Ember)	2003	null	null	null
5	ASEAN (Ember)	2004	null	null	null
6	ASEAN (Ember)	2005	null	null	null
7	ASEAN (Ember)	2006	null	null	null
8	ASEAN (Ember)	2007	null	null	null

Use pandas for data insights

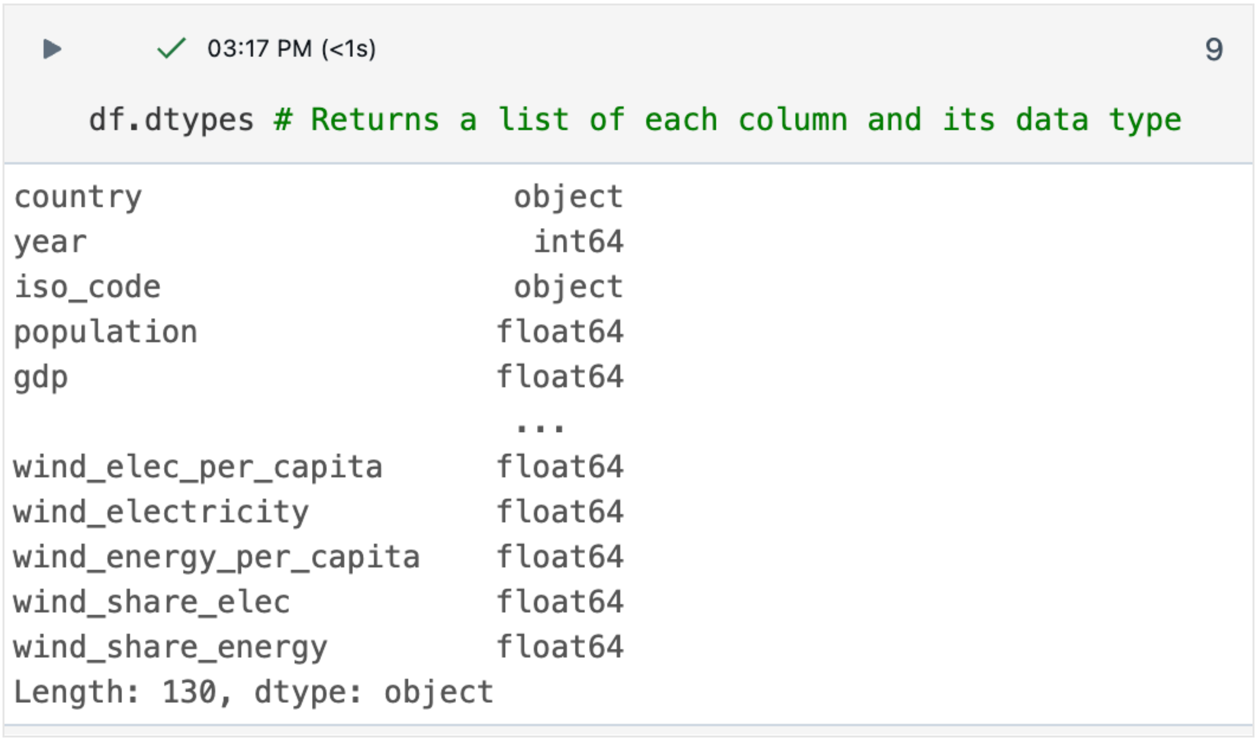
- To understand your dataset effectively, use the following pandas commands:
  The `df.shape` command returns the dimensions of the DataFrame, giving you a quick overview of the number of rows and columns.



```
df.shape # Returns the number of rows and columms in the dataframe
```

```
(21812, 130)
```

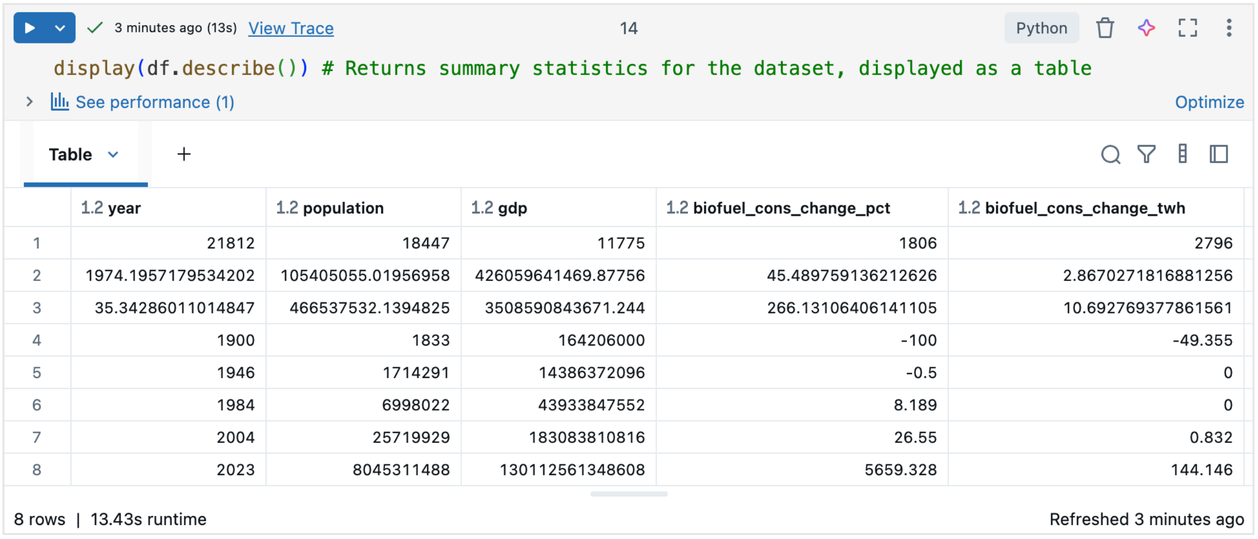
- The `df.dtypes` command provides the data types of each column, helping you understand the kind of data you are dealing with. You can also see the data type for each column in the results table.



```
df.dtypes # Returns a list of each column and its data type
```

```
country                   object
year                       int64
iso_code                  object
population               float64
gdp                      float64
                           ...
wind_elec_per_capita     float64
wind_electricity         float64
wind_energy_per_capita   float64
wind_share_elec          float64
wind_share_energy        float64
Length: 130, dtype: object
```

- The `df.describe()` command generates descriptive statistics for numerical columns, such as mean, standard deviation, and percentiles, which can help you identify patterns, detect anomalies, and understand the distribution of your data.
- Use it with `display()` to see summary statistics in a tabular format you can interact with. See Explore the data using the Databricks notebook output table.



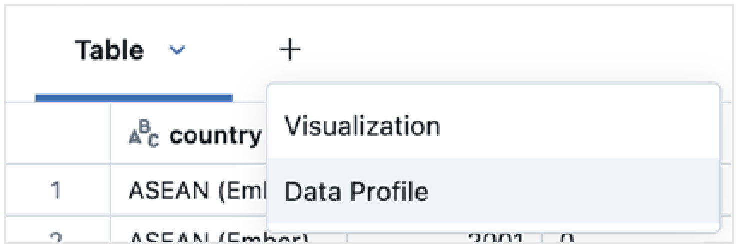
```
display(df.describe()) # Returns summary statistics for the dataset, displayed as a table
```

	1.2 year	1.2 population	1.2 gdp	1.2 biofuel_cons_change_pct	1.2 biofuel_cons_change_twh
1	21812	18447	11775	1806	2796
2	1974.1957179534202	105405055.01956958	426059641469.87756	45.489759136212626	2.8670271816881256
3	35.34286011014847	466537532.1394825	3508590843671.244	266.13106406141105	10.692769377861561
4	1900	1833	164206000	-100	-49.355
5	1946	1714291	14386372096	-0.5	0
6	1984	6998022	43933847552	8.189	0
7	2004	25719929	183083810816	26.55	0.832
8	2023	8045311488	130112561348608	5659.328	144.146

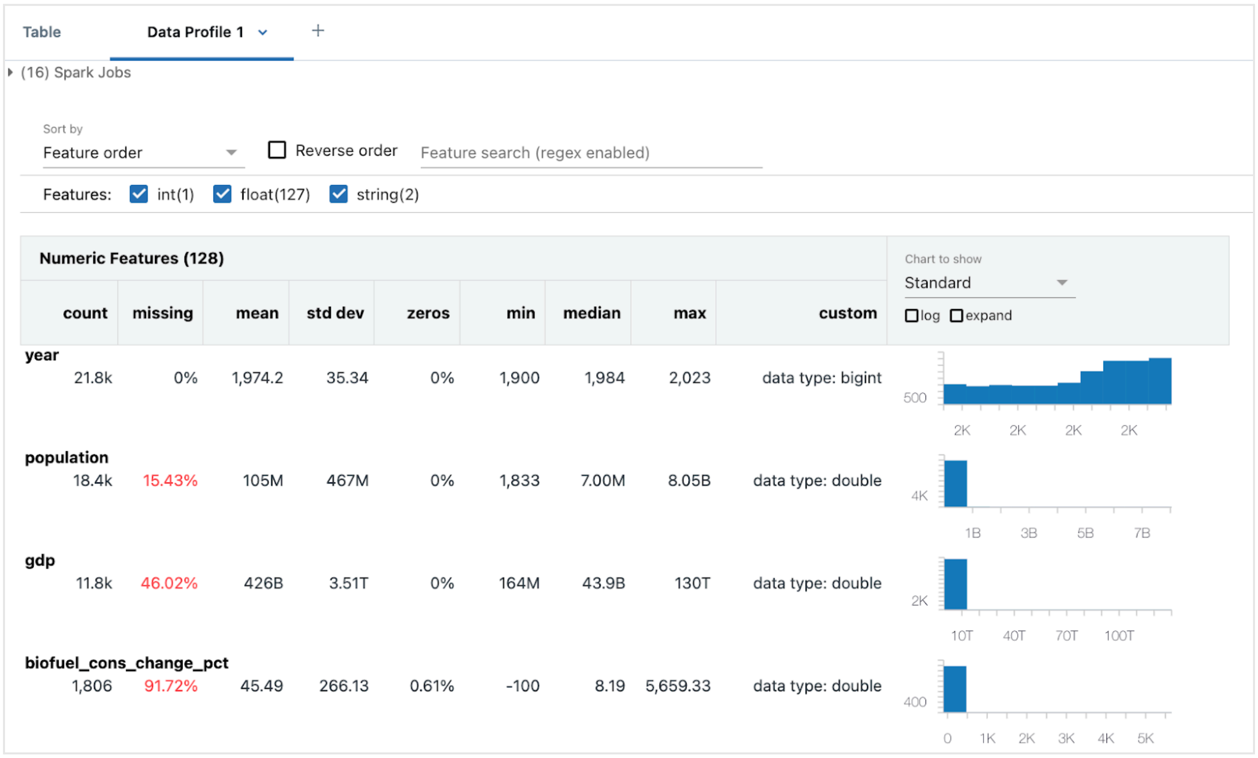
8 rows | 13.43s runtime — Refreshed 3 minutes ago

Generate a data profile

Azure Databricks notebooks include built-in data profiling capabilities. When viewing a DataFrame with the Azure Databricks display function, you can generate a data profile from the table output.



Click + > Data Profile next to the Table in the output. This runs a new command that generates a profile of the data in the DataFrame.



Sort by: Feature order
Reverse order
Feature search (regex enabled)

Features: int(1) float(127) string(2)

Numeric Features (128)

Chart to show: Standard — log, expand

	count	missing	mean	std dev	zeros	min	median	max	custom
year	21.8k	0%	1,974.2	35.34	0%	1,900	1,984	2,023	data type: bigint
population	18.4k	15.43%	105M	467M	0%	1,833	7.00M	8.05B	data type: double
gdp	11.8k	46.02%	426B	3.51T	0%	164M	43.9B	130T	data type: double
biofuel_cons_change_pct	1,806	91.72%	45.49	266.13	0.61%	-100	8.19	5,659.33	data type: double

The data profile includes summary statistics for numeric, string, and date columns as well as histograms of the value distributions for each column

4. Clean the data

Cleaning data is an important part of EDA because it makes sure your dataset is correct, consistent, and ready for analysis.

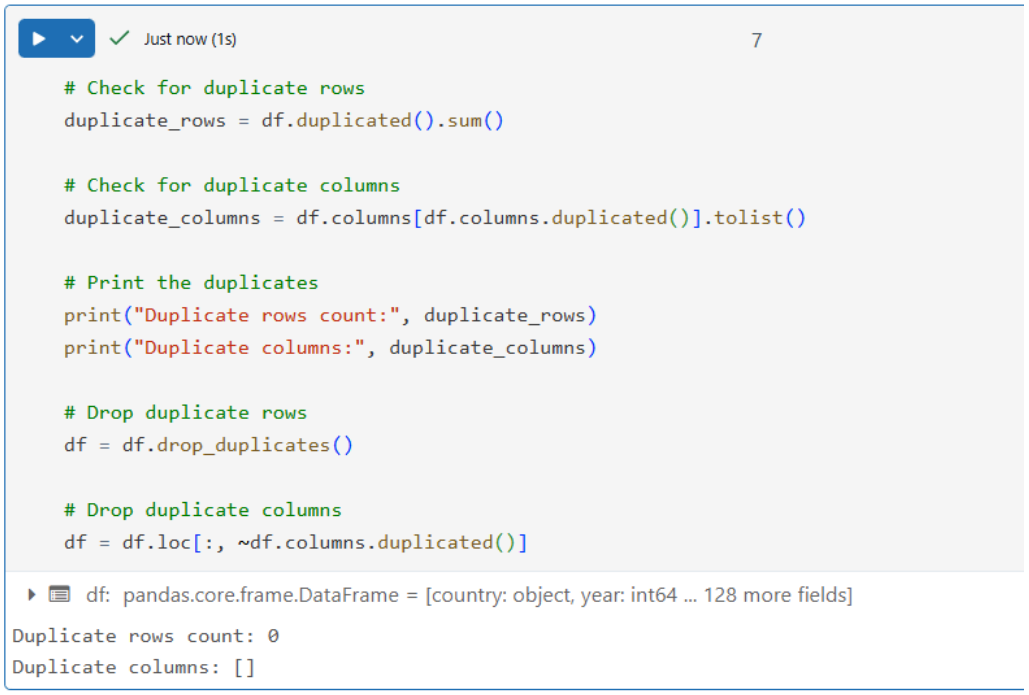
Key cleaning steps include:

1. **Removing duplicates** – Find any repeated rows or columns and delete them.
2. **Handling missing values** –
   - Fill them in with a specific value, or
   - Remove the rows with missing data.

3. **Standardizing data types** – For example:
   - Change text dates into real date formats.
   - Convert data into formats that are easier to work with.

Cleaning the data improves quality and reliability, which leads to better and more accurate analysis.

Remove duplicate data : Check if the data has any duplicate rows or columns. If so, remove them.



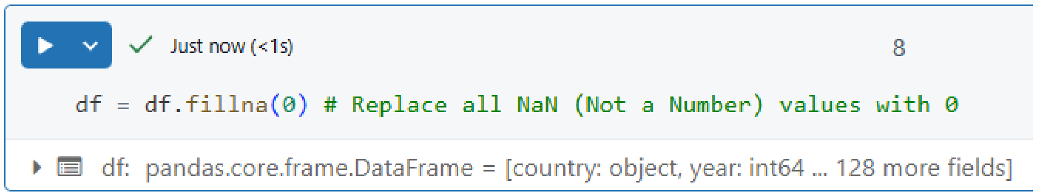
Just now (1s) 7

```python
# Check for duplicate rows
duplicate_rows = df.duplicated().sum()

# Check for duplicate columns
duplicate_columns = df.columns[df.columns.duplicated()].tolist()

# Print the duplicates
print("Duplicate rows count:", duplicate_rows)
print("Duplicate columns:", duplicate_columns)

# Drop duplicate rows
df = df.drop_duplicates()

# Drop duplicate columns
df = df.loc[:, ~df.columns.duplicated()]
```

```
df: pandas.core.frame.DataFrame = [country: object, year: int64 ... 128 more fields]
Duplicate rows count: 0
Duplicate columns: []
```

Handle null or missing values

A common way to treat NaN or Null values is to replace them with 0 for easier mathematical processing.This ensures that any missing data in the DataFrame is replaced with 0, which can be useful for subsequent data analysis or processing steps where missing values might cause issues.



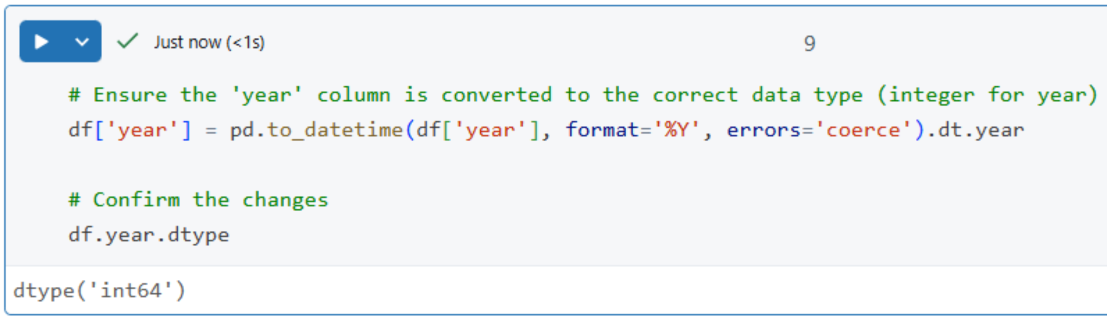
```
df = df.fillna(0) # Replace all NaN (Not a Number) values with 0
```

```
df:  pandas.core.frame.DataFrame = [country: object, year: int64 ... 128 more fields]
```

Reformat Dates

Why reformat dates?
 Dates in datasets can come in many formats — actual date objects, strings like `"2021-01-01"`, or even numbers.
 For analysis, it's often easier to make them all **consistent**.

Example: If your dataset has a `year` column, you can make sure it's stored as an **integer** (e.g., `2021`) instead of a string or mixed type.



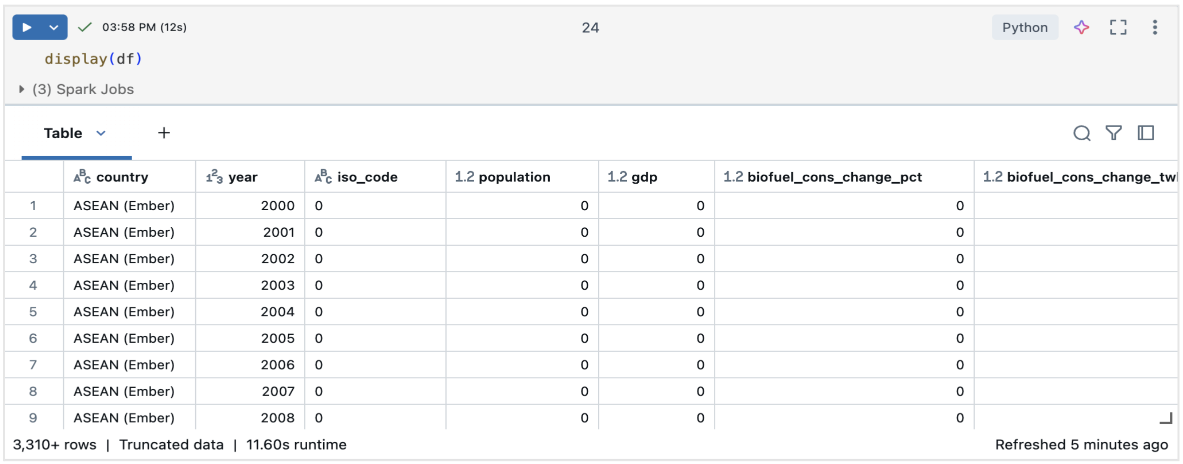
```
# Ensure the 'year' column is converted to the correct data type (integer for year)
df['year'] = pd.to_datetime(df['year'], format='%Y', errors='coerce').dt.year

# Confirm the changes
df.year.dtype
```

```
dtype('int64')
```

5. Explore the data using the Databricks notebook output table

Azure Databricks provides built-in features to help you explore your data using the output table.

In a new cell, use `display(df)` to display the dataset as a table.



```
display(df)
```

(3) Spark Jobs

	country	year	iso_code	population	gdp	biofuel_cons_change_pct	biofuel_cons_change_tw
1	ASEAN (Ember)	2000	0	0	0	0	
2	ASEAN (Ember)	2001	0	0	0	0	
3	ASEAN (Ember)	2002	0	0	0	0	
4	ASEAN (Ember)	2003	0	0	0	0	
5	ASEAN (Ember)	2004	0	0	0	0	
6	ASEAN (Ember)	2005	0	0	0	0	
7	ASEAN (Ember)	2006	0	0	0	0	
8	ASEAN (Ember)	2007	0	0	0	0	
9	ASEAN (Ember)	2008	0	0	0	0	

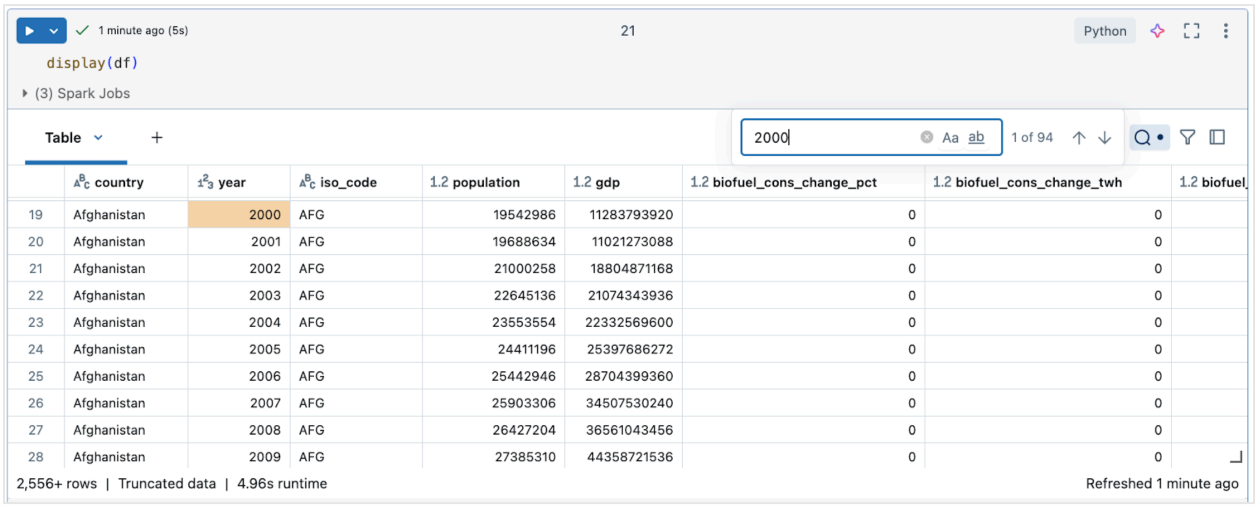
3,310+ rows | Truncated data | 11.60s runtime   Refreshed 5 minutes ago

Using the output table, you can explore your data in several ways:

- Search the data for a specific string or value
- Filter for specific conditions
- Create visualizations using the dataset

Search the data for a specific string or value

Click the search icon on the top right of the table and enter your search.



```
display(df)
```

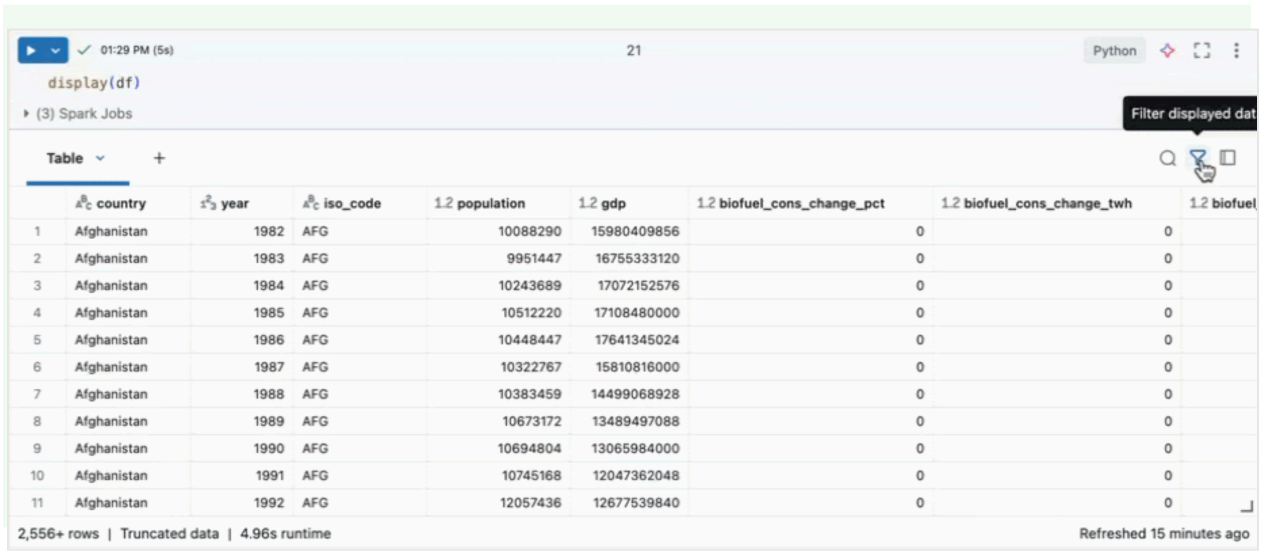
	country	year	iso_code	population	gdp	biofuel_cons_change_pct	biofuel_cons_change_twh	biofuel_
19	Afghanistan	2000	AFG	19542986	11283793920	0	0	
20	Afghanistan	2001	AFG	19688634	11021273088	0	0	
21	Afghanistan	2002	AFG	21000258	18804871168	0	0	
22	Afghanistan	2003	AFG	22645136	21074343936	0	0	
23	Afghanistan	2004	AFG	23553554	22332569600	0	0	
24	Afghanistan	2005	AFG	24411196	25397686272	0	0	
25	Afghanistan	2006	AFG	25442946	28704399360	0	0	
26	Afghanistan	2007	AFG	25903306	34507530240	0	0	
27	Afghanistan	2008	AFG	26427204	36561043456	0	0	
28	Afghanistan	2009	AFG	27385310	44358721536	0	0	

2,556+ rows | Truncated data | 4.96s runtime

Refreshed 1 minute ago

Filter for specific conditions

You can use built-in table filters to filter your columns for specific conditions. There are several ways to create a filter.



```
display(df)
```

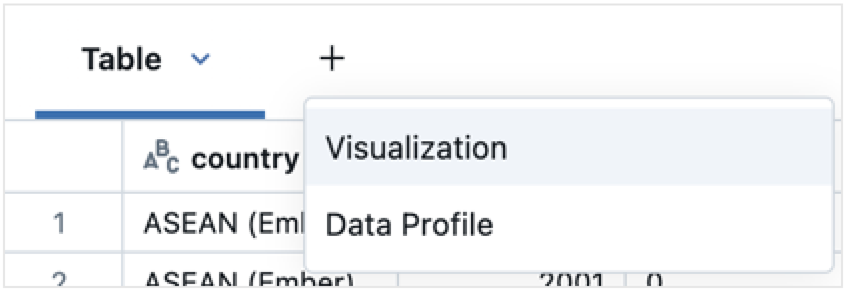
	country	year	iso_code	population	gdp	biofuel_cons_change_pct	biofuel_cons_change_twh	biofuel
1	Afghanistan	1982	AFG	10088290	15980409856	0	0	
2	Afghanistan	1983	AFG	9951447	16755333120	0	0	
3	Afghanistan	1984	AFG	10243689	17072152576	0	0	
4	Afghanistan	1985	AFG	10512220	17108480000	0	0	
5	Afghanistan	1986	AFG	10448447	17641345024	0	0	
6	Afghanistan	1987	AFG	10322767	15810816000	0	0	
7	Afghanistan	1988	AFG	10383459	14499068928	0	0	
8	Afghanistan	1989	AFG	10673172	13489497088	0	0	
9	Afghanistan	1990	AFG	10694804	13065984000	0	0	
10	Afghanistan	1991	AFG	10745168	12047362048	0	0	
11	Afghanistan	1992	AFG	12057436	12677539840	0	0	

2,556+ rows | Truncated data | 4.96s runtime

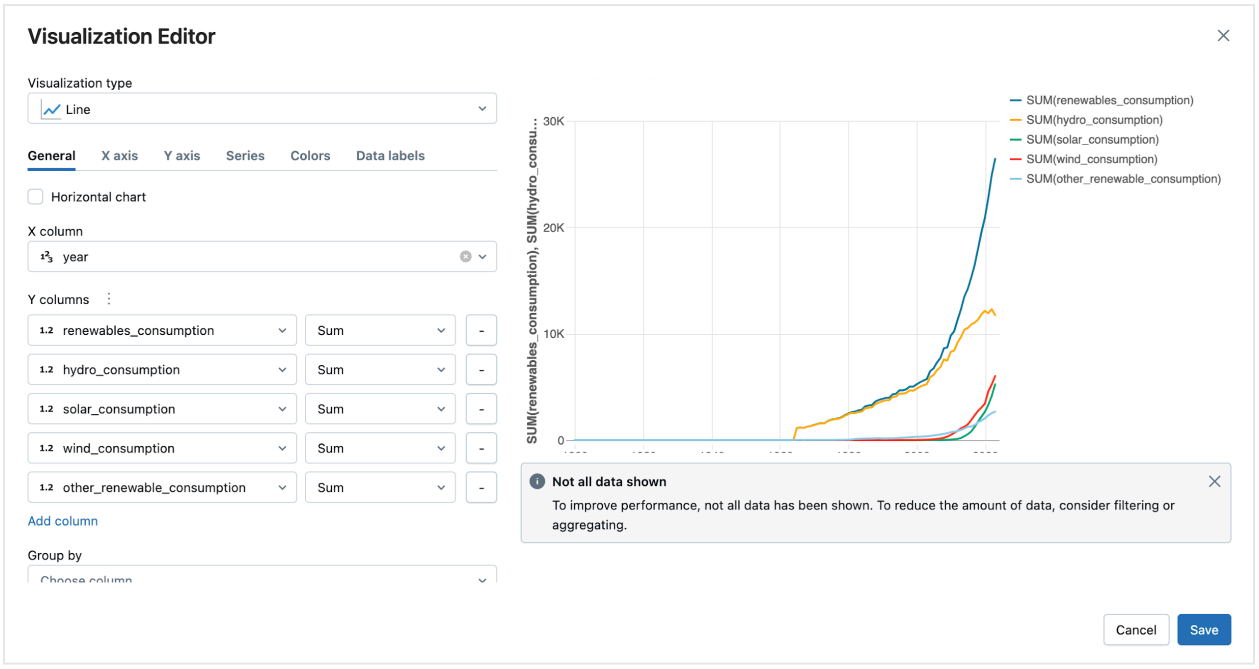
Refreshed 15 minutes ago

Create visualizations using the dataset

At the top of the output table, click + > Visualization to open the visualization editor.



Select the visualization type and columns you'd like to visualize. The editor displays a preview of the chart based on your configuration. For example, the image below shows how to add multiple line charts to view the consumption of various renewable energy sources over time.


Visualization Editor

Visualization type: Line

General | X axis | Y axis | Series | Colors | Data labels

Horizontal chart

X column: year

Y columns

renewables_consumption	Sum
hydro_consumption	Sum
solar_consumption	Sum
wind_consumption	Sum
other_renewable_consumption	Sum

Add column

Group by: Choose column

SUM(renewables_consumption)
SUM(hydro_consumption)
SUM(solar_consumption)
SUM(wind_consumption)
SUM(other_renewable_consumption)
30K
20K
10K
0
SUM(renewables_consumption), SUM(hydro_consu...

Not all data shown

To improve performance, not all data has been shown. To reduce the amount of data, consider filtering or aggregating.

Click Save to add the visualization as a tab in the cell output.

6.Explore and visualize the data using Python libraries

In EDA, charts and graphs are super helpful because they make it easier to spot patterns, trends, and relationships in your data — things you might miss if you only looked at numbers.

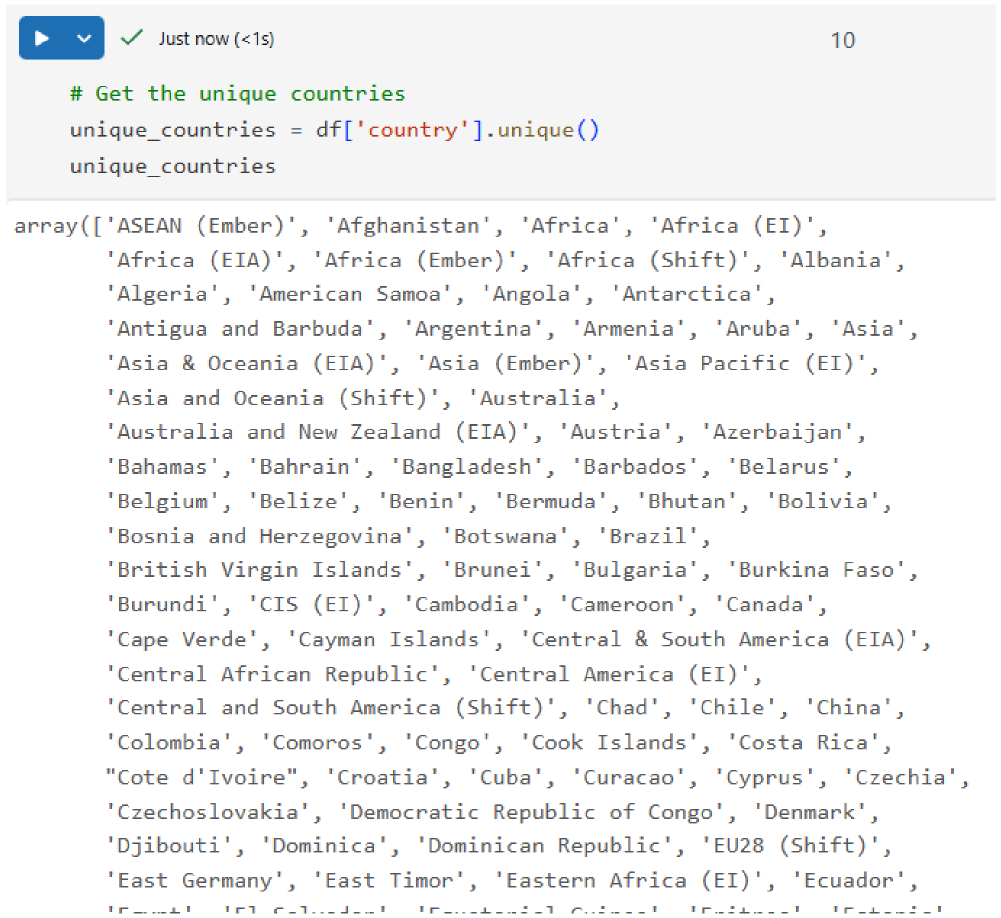
Using libraries like **Plotly** or **Matplotlib**, you can make:

- **Scatter plots** → good for finding outliers.
- **Bar charts** → compare values between categories.
- **Line graphs / Time series plots** → see trends over time.
- **Histograms** → understand the distribution of your data.

6.1 – Create an array of unique countries

We start by checking which countries are in our dataset. This also shows if there are non-country entries like “World” or “High-income countries” that may need filtering.

Python code:



```python
# Get the unique countries
unique_countries = df['country'].unique()
unique_countries
```

```
array(['ASEAN (Ember)', 'Afghanistan', 'Africa', 'Africa (EI)',
       'Africa (EIA)', 'Africa (Ember)', 'Africa (Shift)', 'Albania',
       'Algeria', 'American Samoa', 'Angola', 'Antarctica',
       'Antigua and Barbuda', 'Argentina', 'Armenia', 'Aruba', 'Asia',
       'Asia & Oceania (EIA)', 'Asia (Ember)', 'Asia Pacific (EI)',
       'Asia and Oceania (Shift)', 'Australia',
       'Australia and New Zealand (EIA)', 'Austria', 'Azerbaijan',
       'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus',
       'Belgium', 'Belize', 'Benin', 'Bermuda', 'Bhutan', 'Bolivia',
       'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'British Virgin Islands', 'Brunei', 'Bulgaria', 'Burkina Faso',
       'Burundi', 'CIS (EI)', 'Cambodia', 'Cameroon', 'Canada',
       'Cape Verde', 'Cayman Islands', 'Central & South America (EIA)',
       'Central African Republic', 'Central America (EI)',
       'Central and South America (Shift)', 'Chad', 'Chile', 'China',
       'Colombia', 'Comoros', 'Congo', 'Cook Islands', 'Costa Rica',
       "Cote d'Ivoire", 'Croatia', 'Cuba', 'Curacao', 'Cyprus', 'Czechia',
       'Czechoslovakia', 'Democratic Republic of Congo', 'Denmark',
       'Djibouti', 'Dominica', 'Dominican Republic', 'EU28 (Shift)',
       'East Germany', 'East Timor', 'Eastern Africa (EI)', 'Ecuador',
       'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
```

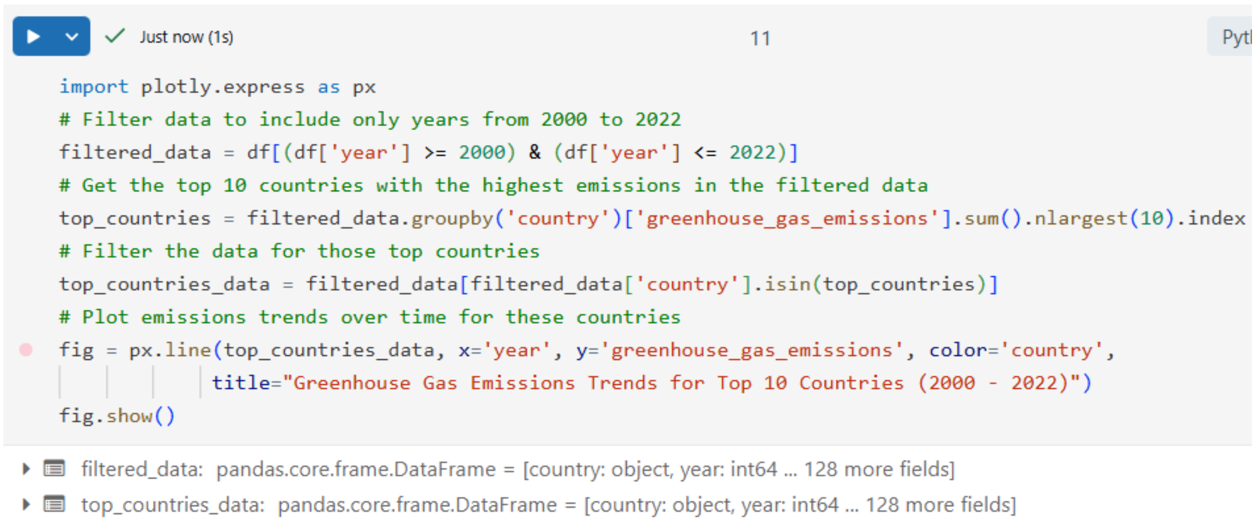
What happens:

- The `.unique()` method lists all distinct values in the `country` column.

Insight:
Some entries aren't actual single countries (like "World" or "Asia"), so for some analyses, it may be better to filter data by **region** or focus only on actual countries.

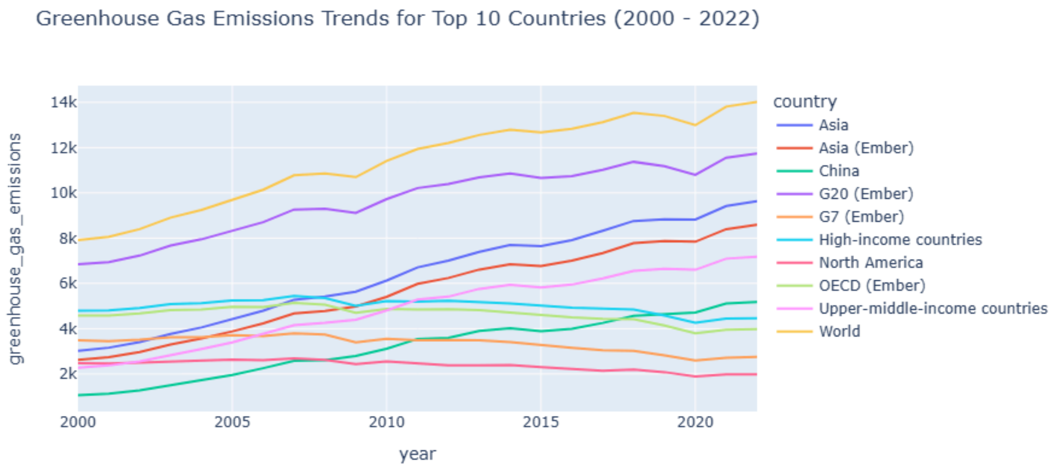
6.2 Emissions Trends for Top 10 Countries (2000–2022)

- Filter for **2000–2022**.
- Identify top 10 emitters by total emissions.
- Use `plotly.express.line()` to show trends over time.



```
import plotly.express as px
# Filter data to include only years from 2000 to 2022
filtered_data = df[(df['year'] >= 2000) & (df['year'] <= 2022)]
# Get the top 10 countries with the highest emissions in the filtered data
top_countries = filtered_data.groupby('country')['greenhouse_gas_emissions'].sum().nlargest(10).index
# Filter the data for those top countries
top_countries_data = filtered_data[filtered_data['country'].isin(top_countries)]
# Plot emissions trends over time for these countries
fig = px.line(top_countries_data, x='year', y='greenhouse_gas_emissions', color='country',
              title="Greenhouse Gas Emissions Trends for Top 10 Countries (2000 - 2022)")
fig.show()
```

```
filtered_data:  pandas.core.frame.DataFrame = [country: object, year: int64 ... 128 more fields]
top_countries_data:  pandas.core.frame.DataFrame = [country: object, year: int64 ... 128 more fields]
```



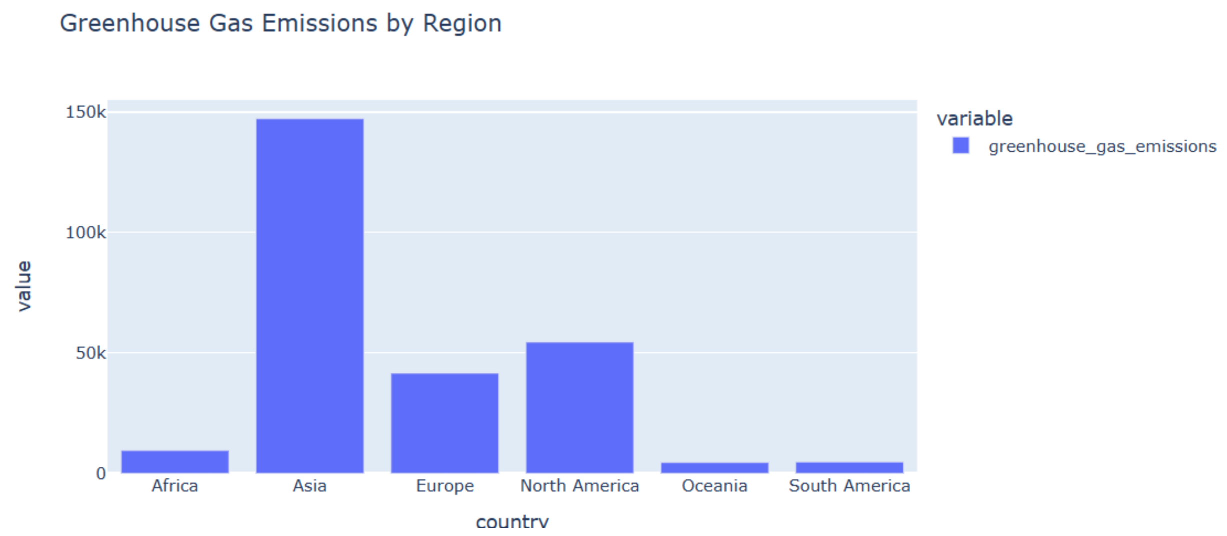
Key Insight:
Most top emitters show an upward emissions trend, except a few stable/declining ones.

6.3 Emissions by Region

- Keep only **major world regions**.
- Group & sum emissions.

- Use `plotly.express.bar()` to compare totals.

```
# Filter out regional entities
regions = ['Africa', 'Asia', 'Europe', 'North America', 'South America', 'Oceania']
# Calculate total emissions for each region
regional_emissions = df[df['country'].isin(regions)].groupby('country')['greenhouse_gas_emissions'].sum()
# Plot the comparison
fig = px.bar(regional_emissions, title="Greenhouse Gas Emissions by Region")
fig.show()
```

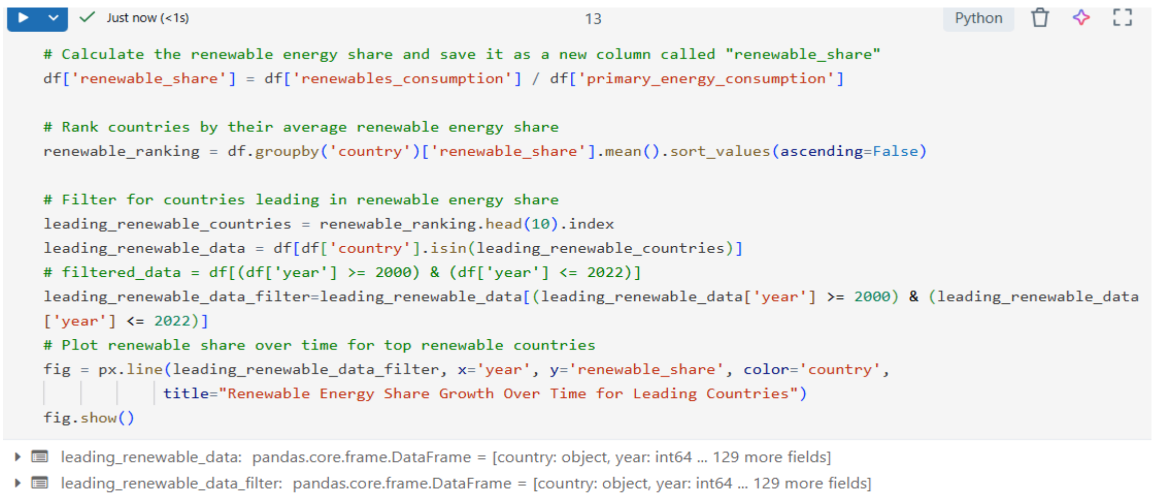


Key Insight:
Asia leads emissions by a large margin; Oceania & South America emit the least

6.4 Renewable Energy Share Growth

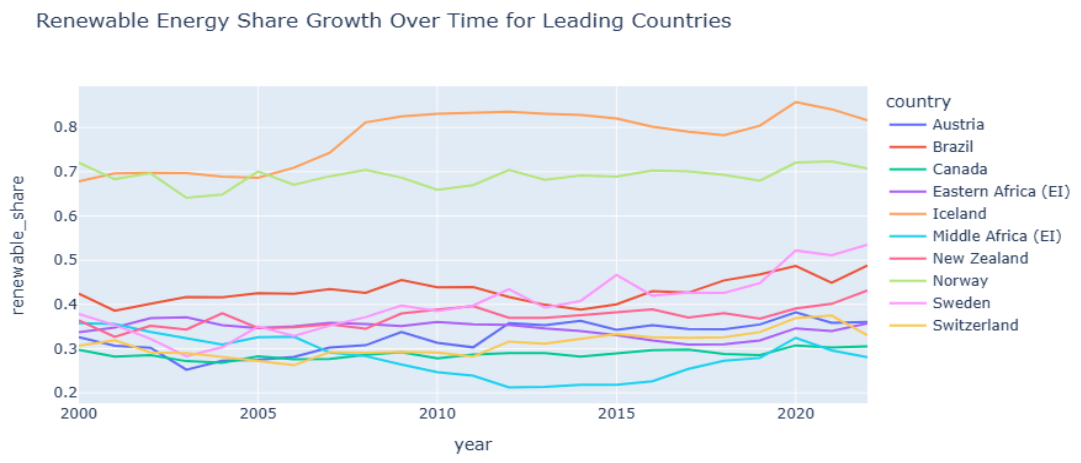
- Create `renewable_share` = `renewables_consumption / primary_energy_consumption`.

- Rank by **average share** and plot top 10 over time.



```python
# Calculate the renewable energy share and save it as a new column called "renewable_share"
df['renewable_share'] = df['renewables_consumption'] / df['primary_energy_consumption']

# Rank countries by their average renewable energy share
renewable_ranking = df.groupby('country')['renewable_share'].mean().sort_values(ascending=False)

# Filter for countries leading in renewable energy share
leading_renewable_countries = renewable_ranking.head(10).index
leading_renewable_data = df[df['country'].isin(leading_renewable_countries)]
# filtered_data = df[(df['year'] >= 2000) & (df['year'] <= 2022)]
leading_renewable_data_filter=leading_renewable_data[(leading_renewable_data['year'] >= 2000) & (leading_renewable_data
['year'] <= 2022)]
# Plot renewable share over time for top renewable countries
fig = px.line(leading_renewable_data_filter, x='year', y='renewable_share', color='country',
            title="Renewable Energy Share Growth Over Time for Leading Countries")
fig.show()
```

```
leading_renewable_data:  pandas.core.frame.DataFrame = [country: object, year: int64 ... 129 more fields]
leading_renewable_data_filter:  pandas.core.frame.DataFrame = [country: object, year: int64 ... 129 more fields]
```

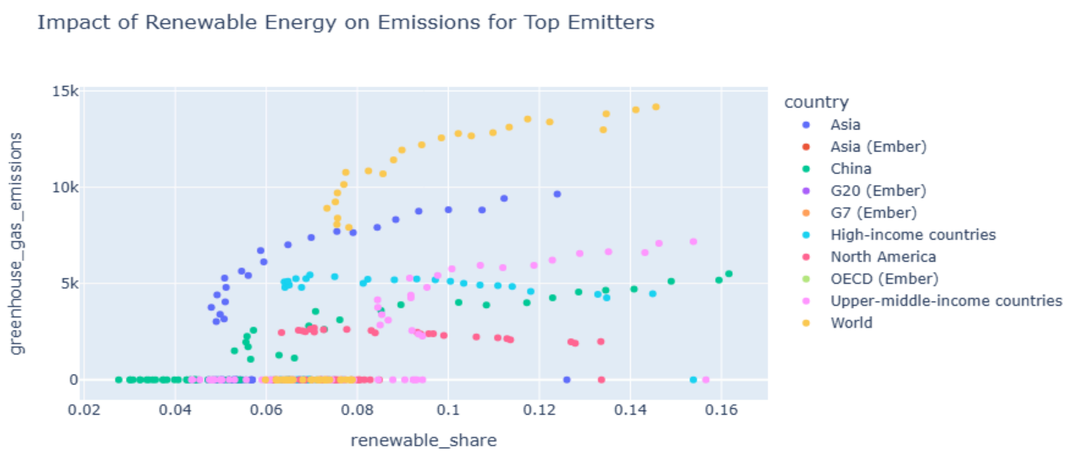


Key Insight:
Norway & Iceland lead with >50% renewable share. Growth is non-linear with occasional dips.

2.5 Scatter Plot — Renewables vs Emissions for Top Emitters

- Compare `renewable_share` to `greenhouse_gas_emissions`.
- Use `plotly.express.scatter()`.

```python
# Select top emitters and calculate renewable share vs. emissions
top_emitters = df.groupby('country')['greenhouse_gas_emissions'].sum().nlargest(10).index
top_emitters_data = df[df['country'].isin(top_emitters)]

# Plot renewable share vs. greenhouse gas emissions over time
fig = px.scatter(top_emitters_data, x='renewable_share', y='greenhouse_gas_emissions',
                 color='country', title="Impact of Renewable Energy on Emissions for Top Emitters")
fig.show()
```



Key Insight:
For most top emitters, more renewables still coincide with more total emissions — likely due to higher overall energy use.

EDA in Databricks SQL = Create a dashboard

Requirements
Step 1. Create a dashboard
Step 2. Define datasets
Step 3. Add a visualization
Step 4. Configure your visualization
Step 5. Clone and modify a visualization
Step 6. Create a scatterplot
Step 7. Create dashboard filters
Step 8. Resize and arrange charts and filters
Step 9. Publish and share

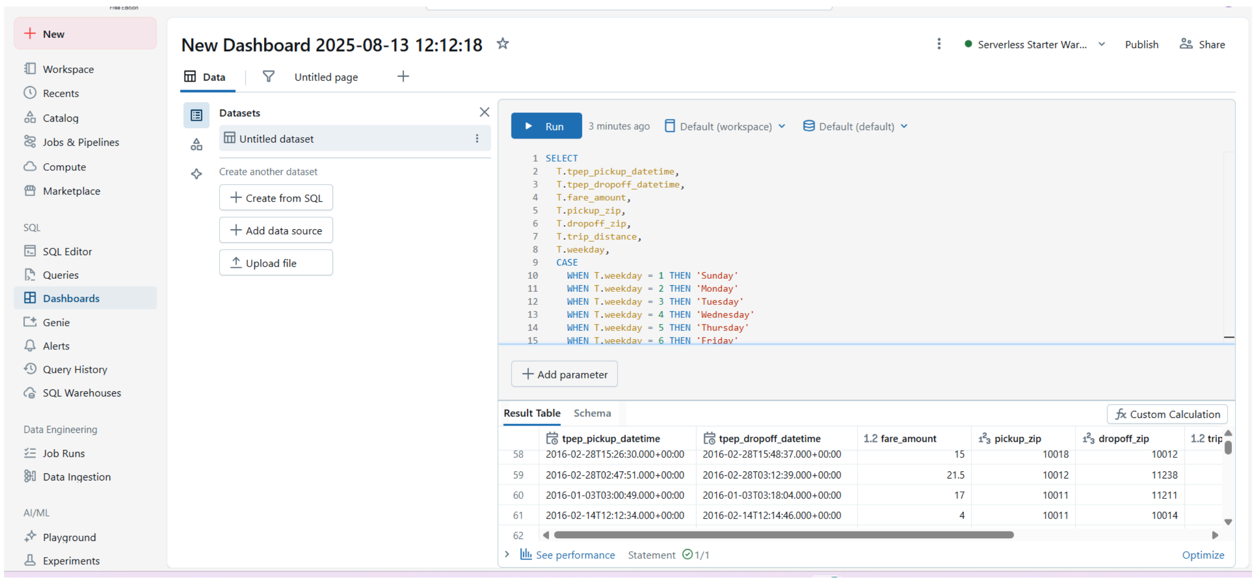
Step 1. Create a dashboard

Click ⊕ New in the sidebar and select Dashboard.

Step 2. Define datasets

The Canvas tab is for creating and editing widgets like visualizations, text boxes, and filters. The Data tab is for defining the underlying datasets used in your dashboard.

1. Click the Data tab.
2. Click Create form SQL.
3. Paste the following query into the editor. Then click Run to return a collection of records.



```
SELECT
  T.tpep_pickup_datetime,
  T.tpep_dropoff_datetime,
  T.fare_amount,
  T.pickup_zip,
  T.dropoff_zip,
  T.trip_distance,
  T.weekday,
  CASE
    WHEN T.weekday = 1 THEN 'Sunday'
    WHEN T.weekday = 2 THEN 'Monday'
    WHEN T.weekday = 3 THEN 'Tuesday'
    WHEN T.weekday = 4 THEN 'Wednesday'
    WHEN T.weekday = 5 THEN 'Thursday'
    WHEN T.weekday = 6 THEN 'Friday'
```

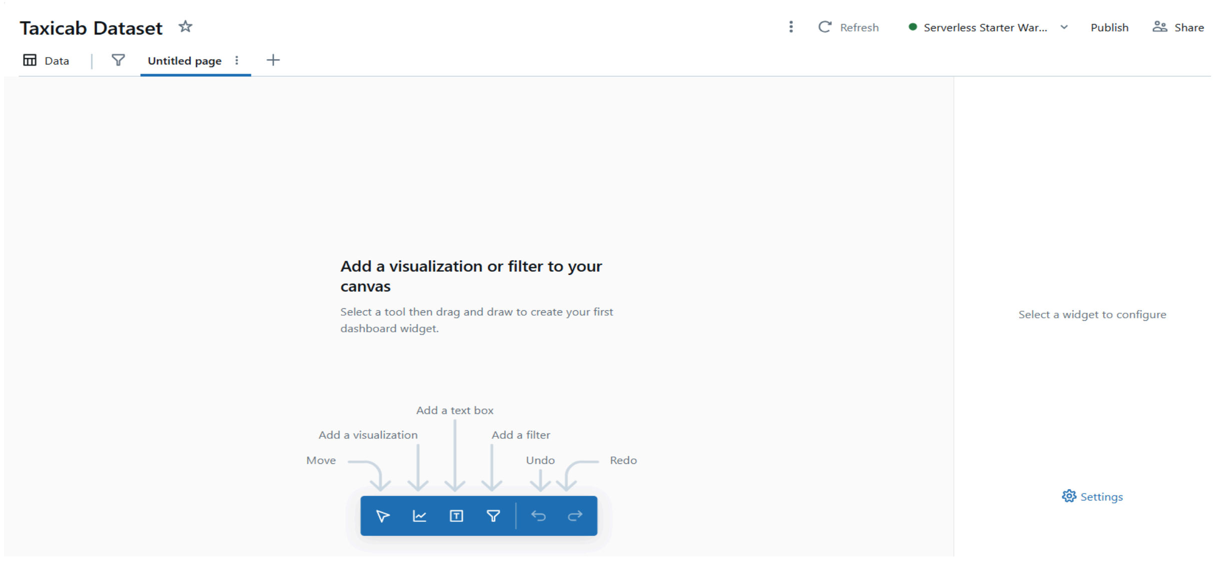
	tpep_pickup_datetime	tpep_dropoff_datetime	1.2 fare_amount	pickup_zip	dropoff_zip	1.2 trip
58	2016-02-28T15:26:30.000+00:00	2016-02-28T15:48:37.000+00:00	15	10018	10012	
59	2016-02-28T02:47:51.000+00:00	2016-02-28T03:12:39.000+00:00	21.5	10012	11238	
60	2016-01-03T03:00:49.000+00:00	2016-01-03T03:18:04.000+00:00	17	10011	11211	
61	2016-02-14T12:12:34.000+00:00	2016-02-14T12:14:46.000+00:00	4	10011	10014	
62						

1. Inspect your results. The returned records appear in the Result table when the query is finished running.
2. Change the name of your query. Your newly defined dataset is autosaved with the name, Untitled dataset. Double click on the title to rename it Taxicab data.

Step 3. Add a visualization

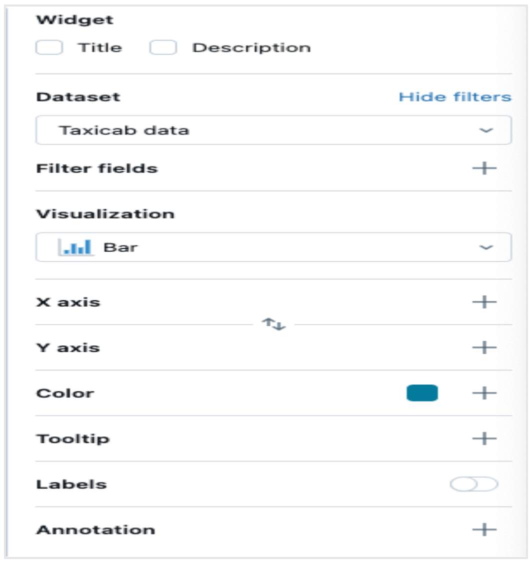
To create your first visualization, complete the following steps:

1. Click the Canvas tab.
2. Click - Add a visualization to add a visualization widget and use your mouse to place it on the canvas.


Taxicab Dataset
Refresh
Serverless Starter War...
Publish
Share
Data
Untitled page
Add a visualization or filter to your canvas
Select a tool then drag and draw to create your first dashboard widget.
Add a text box
Add a visualization
Add a filter
Move
Undo
Redo
Select a widget to configure
Settings

Step 4. Configure your visualization

When a visualization widget is selected, you can use the configuration panel on the right side of the screen to display your data. As shown in the following image, only one Dataset has been defined, and it is selected automatically.



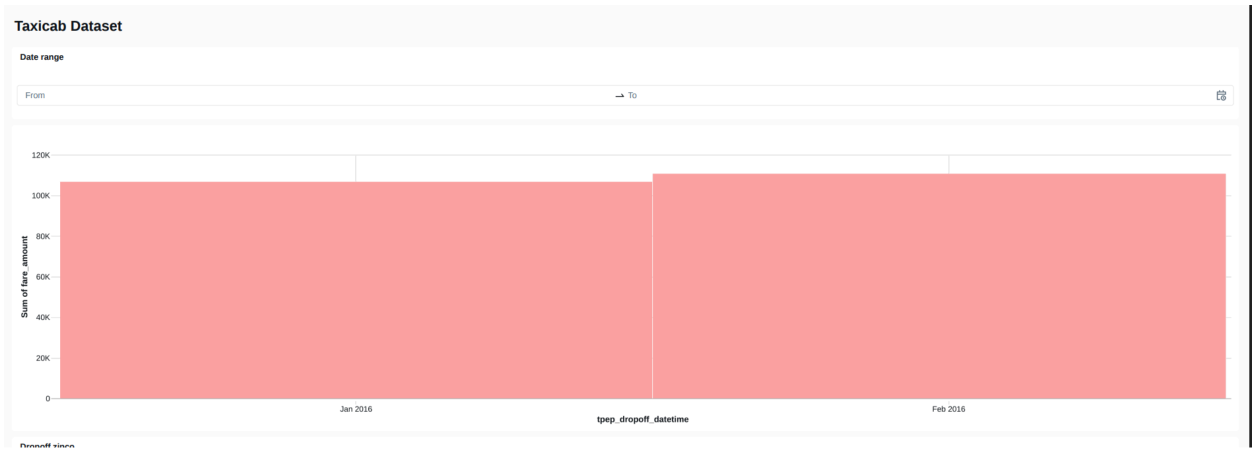
Setup the X-axis

1. If necessary, select Bar from the Visualization drop-down menu.

2. Click the plus icon to choose the data presented along the X-axis. You can use the search bar to search for a field by name. Select tpep_dropoff_datetime.
3. Click the field name you selected to view additional configuration options.
    - As the Scale Type, select Continuous.
    - For the Transform selection, choose HOURLY.

Setup the Y-axis

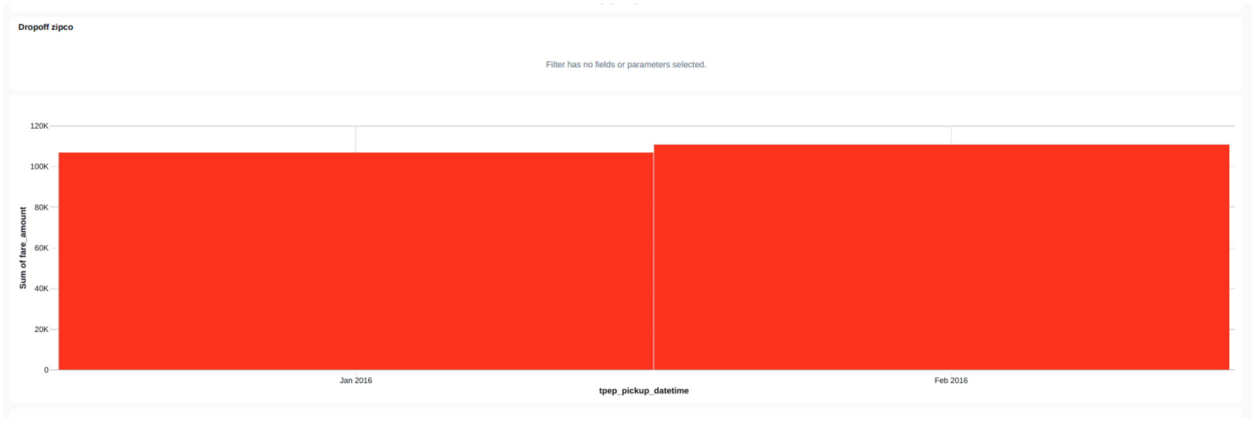
1. Click the plus icon next to the Y-axis to select the fare_amount for the data presented along the y-axis.
2. Click the field name you selected to view additional configuration options.
    - As the Scale Type, select Continuous.
    - For the Transform selection, choose AVG.



Step 5. Clone and modify a visualization

You can clone an existing chart to create a new visualization.

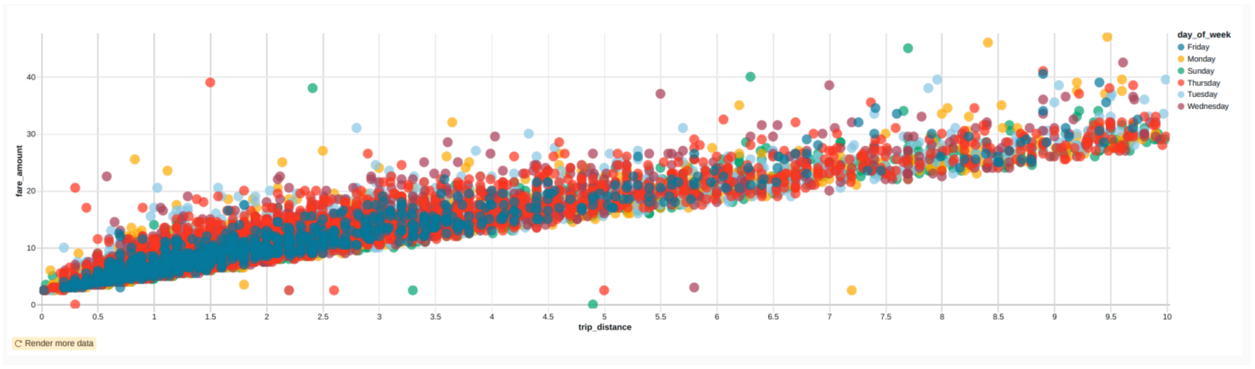
1. Right-click on your existing chart and then click Clone.
2. With your new chart selected, use the configuration panel to change the X-axis field to tpep_pickup_datetime. If necessary, choose HOURLY under the Transform type.
3. Use the Color selector to choose a new color for your new bar chart.



Step 6. Create a scatterplot

Create a new scatterplot with colors differentiated by value. To create a scatterplot, complete the following steps:

1. Click the Create a visualization icon to create a new visualization widget.
2. Configure your chart by making the following selections:
    - Dataset: Taxicab data
    - Visualization: Scatter
    - X axis: trip_distance
    - Y axis: fare_amount
    - Color: Click the plus icon > day_of_week



Step 7. Create dashboard filters see 1st bar graph (top)

You can use filters to make your dashboards interactive. In this step, you create filters on three fields.

Create a date range filter

1. Click Add a filter (field/parameter) to add a filter widget.
2. Click Widget title and enter Date range to retitle your filter.
3. From the Filter drop-down menu in the configuration panel, select Date range picker.
4. Click the plus icon next to the Fields menu. Click tpep_pickup_datetime from the drop-down menu.

Create a single-select drop-down filter

1. Click Add a filter (field/parameter) to add a filter widget.
2. Click Widget title and enter Dropoff zip code to retitle your filter.
3. From the Filter drop-down menu in the configuration panel, select Dropdown (single-select).
4. Select the Title checkbox to create a title field on your filter. Click on the placeholder title and type Dropoff zip code to retitle your filter.
5. From the Fields menu, select dropoff_zip.

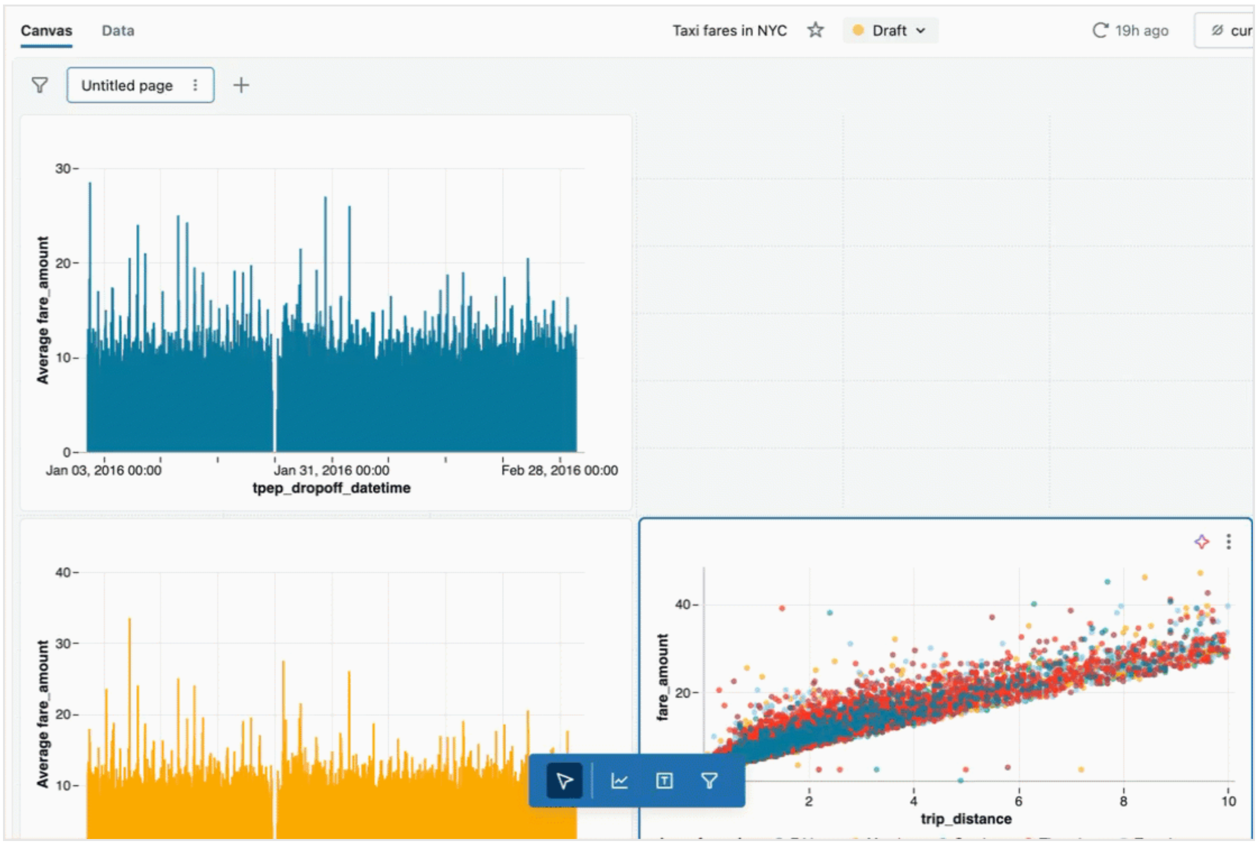
Clone a filter

1. Right-click on your Dropoff zip code filter. Then, click Clone.
2. Double-click on the title and enter Pickup zip code to retitle the cloned widget.
3. Click the to remove the current field. Then, select pickup_zip to filter on that field.

Step 8. Resize and arrange charts and filters

Use your mouse to arrange and resize your charts and filters.

The following image shows one possible arrangement for this dashboard.



Step 9. Publish and share

While you develop a dashboard, your progress is saved as a draft. To create a clean copy for easy consumption, publish your dashboard.

1. Click Publish in the upper-right corner of the dashboard.
2. Click Embed credentials (default).
3. Click Publish. A Sharing dialog opens.
4. Add users, groups, or service principals that you want to share with. Set permission levels as appropriate. See Share a dashboard and AI/BI dashboard ACLs to learn more about permissions and rights.
5. Click Copy link and paste it in a new tab to go to your published dashboard.

Visualizations in Databricks notebooks and SQL editor

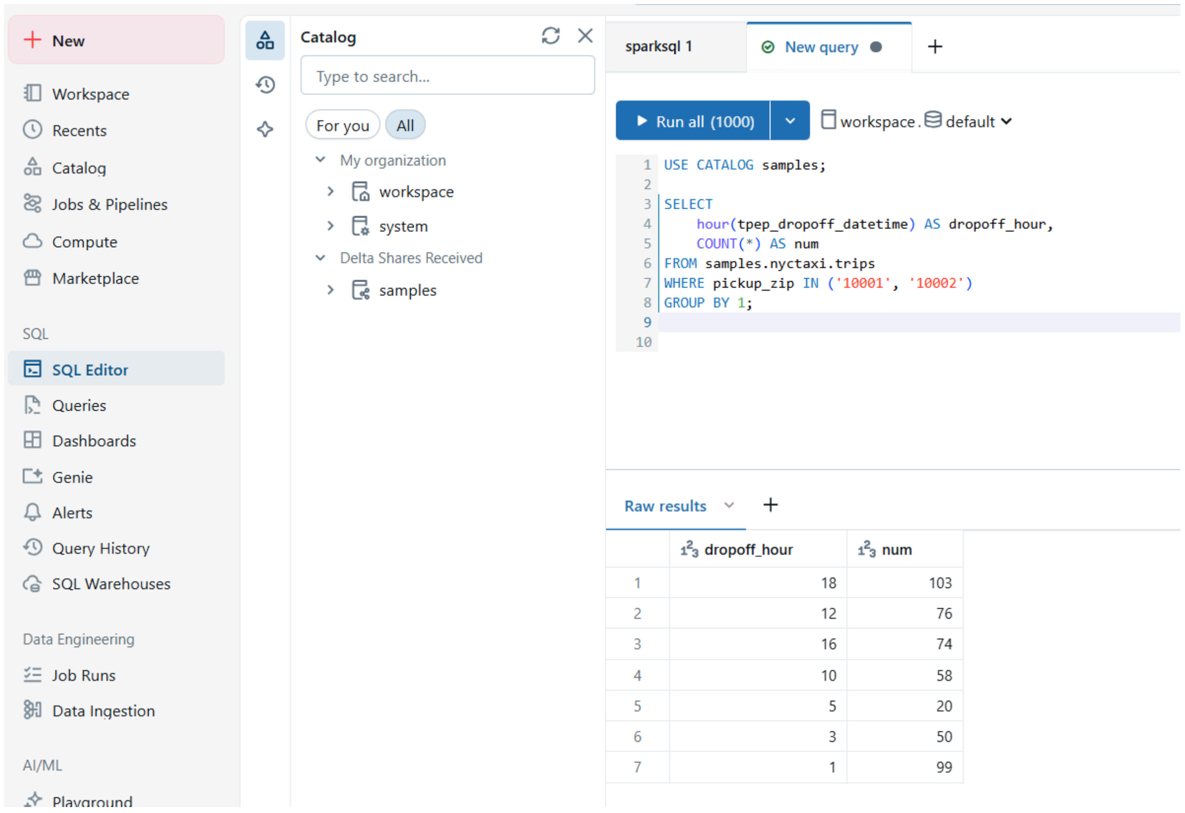
Azure Databricks has powerful, built-in tools for creating charts and visualizations directly from your data when working with notebooks or the SQL editor done in both.

Generate a result set to visualize

To generate the result set used on this page, use the following code:

SQL

Run the following query in the SQL editor.

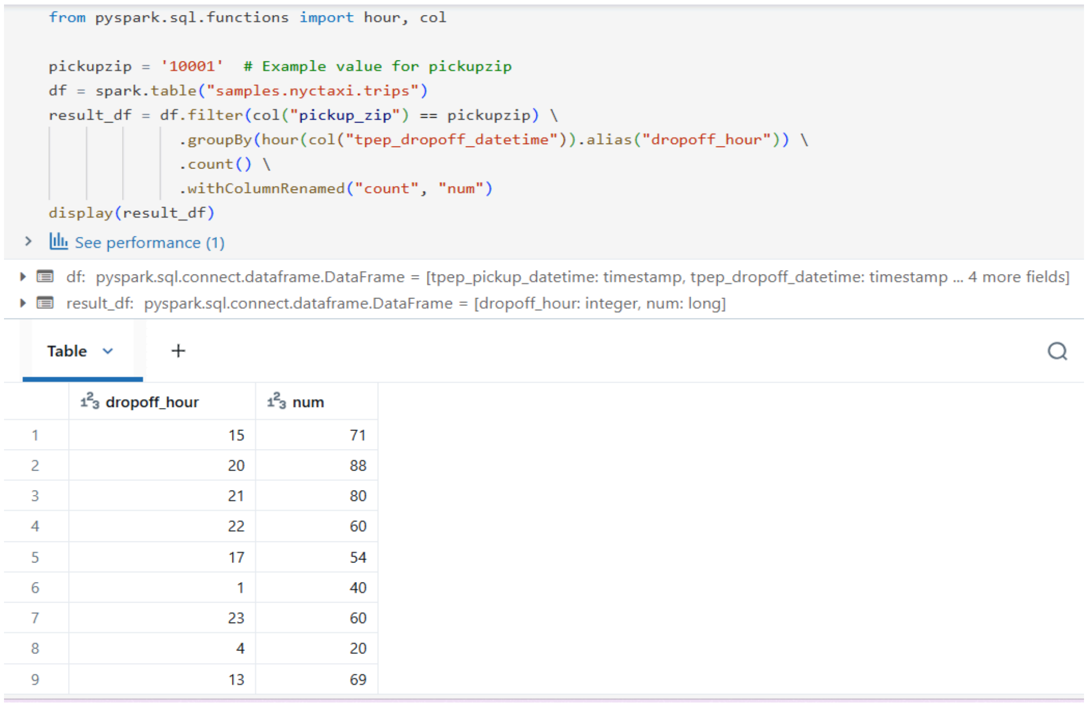


```sql
USE CATALOG samples;

SELECT
    hour(tpep_dropoff_datetime) AS dropoff_hour,
    COUNT(*) AS num
FROM samples.nyctaxi.trips
WHERE pickup_zip IN ('10001', '10002')
GROUP BY 1;
```

	dropoff_hour	num
1	18	103
2	12	76
3	16	74
4	10	58
5	5	20
6	3	50
7	1	99

Python

Run the following code from a Python cell in a notebook.



```python
from pyspark.sql.functions import hour, col

pickupzip = '10001'  # Example value for pickupzip
df = spark.table("samples.nyctaxi.trips")
result_df = df.filter(col("pickup_zip") == pickupzip) \
              .groupBy(hour(col("tpep_dropoff_datetime")).alias("dropoff_hour")) \
              .count() \
              .withColumnRenamed("count", "num")
display(result_df)
```

See performance (1)

df: pyspark.sql.connect.dataframe.DataFrame = [tpep_pickup_datetime: timestamp, tpep_dropoff_datetime: timestamp ... 4 more fields]
result_df: pyspark.sql.connect.dataframe.DataFrame = [dropoff_hour: integer, num: long]

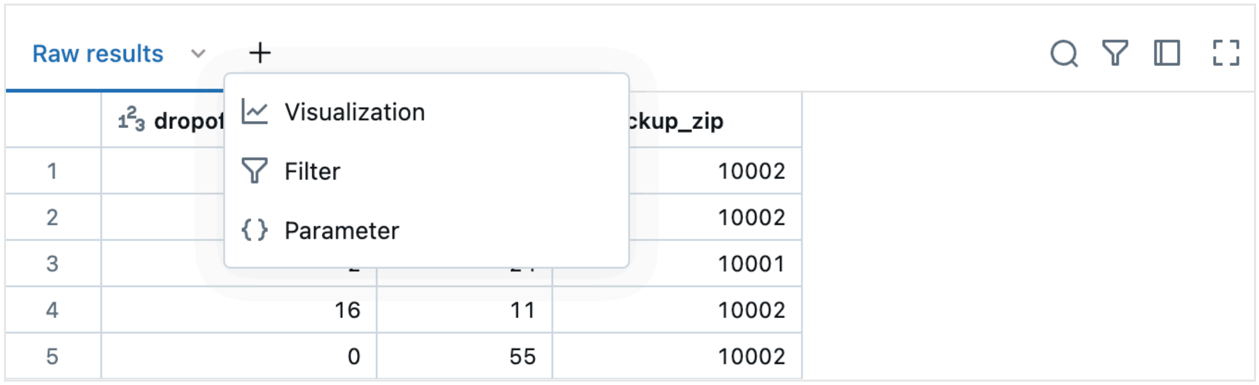
	dropoff_hour	num
1	15	71
2	20	88
3	21	80
4	22	60
5	17	54
6	1	40
7	23	60
8	4	20
9	13	69

Create a new visualization

You can create visualizations in the same UI where the results table appears. If you're working in a notebook, you can also generate a data profile, which provide summary statistics and visual insights for DataFrames and tables. To learn more about data profiles, see Generate a data profile.

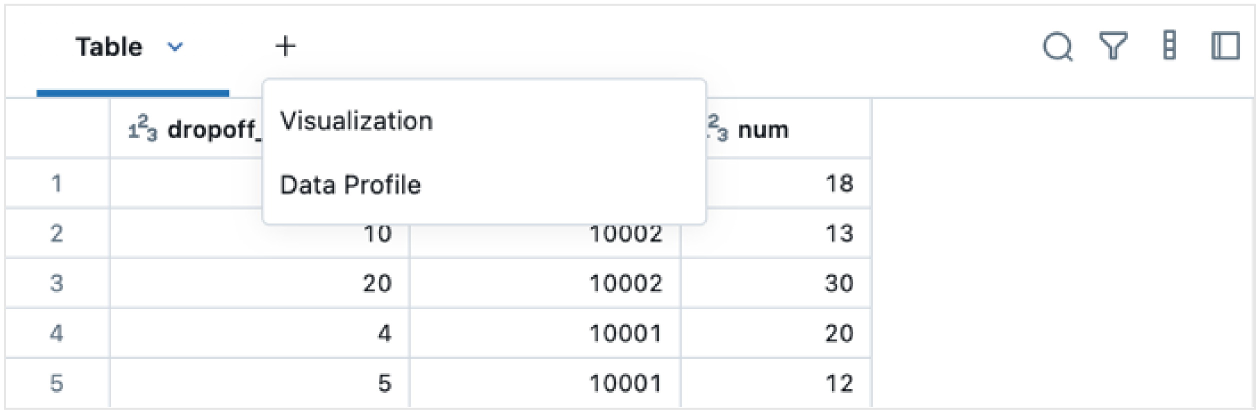
1. To create a visualization, click + above a result and select Visualization to open the visualization editor.

SQL editor



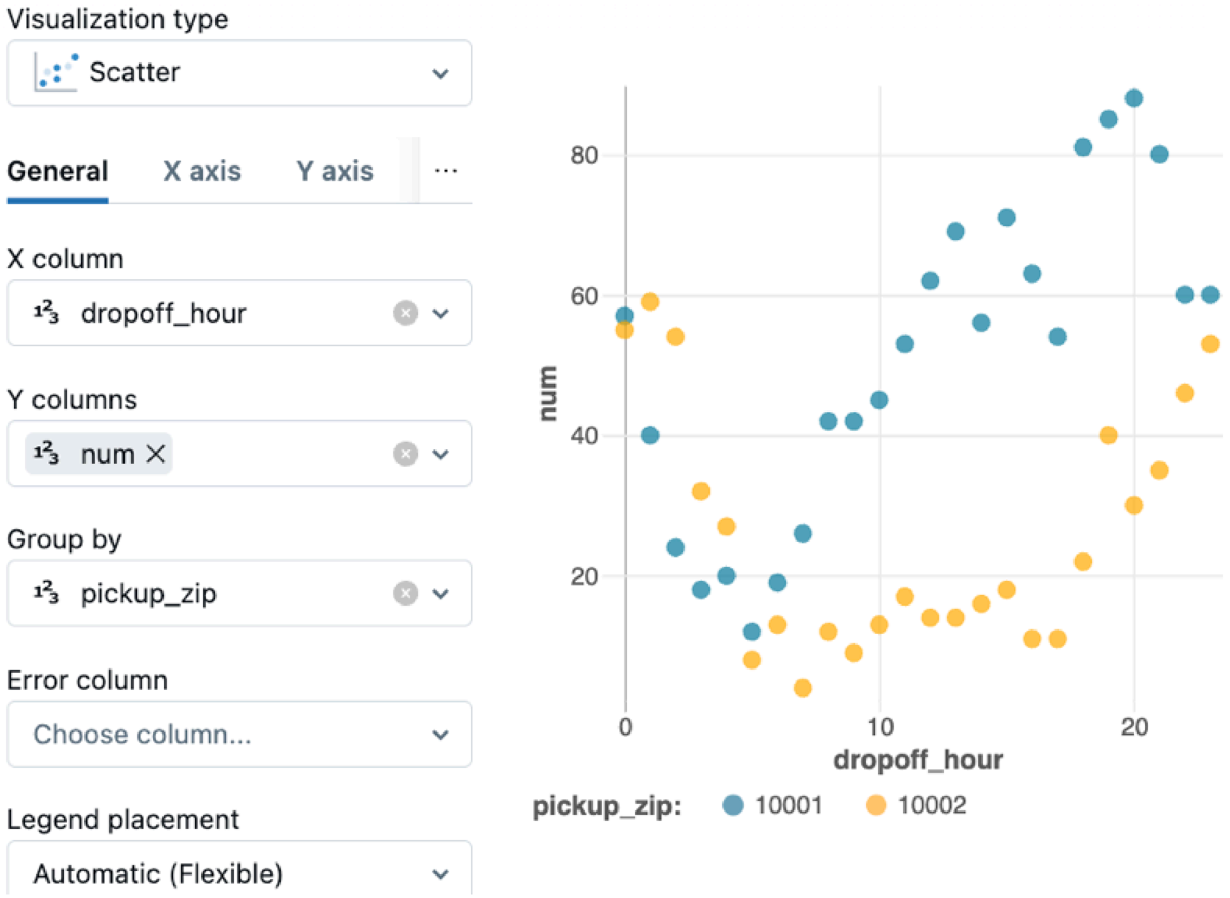
	dropof		ckup_zip
1			10002
2			10002
3			10001
4	16	11	10002
5	0	55	10002

Notebook



	dropoff_		num
1			18
2	10	10002	13
3	20	10002	30
4	4	10001	20
5	5	10001	12

2. In the Visualization Type drop-down, choose a type. Then, select the data to appear in the visualization.



Visualization type: Scatter

X column: dropoff_hour

Y columns: num

Group by: pickup_zip

Error column: Choose column...

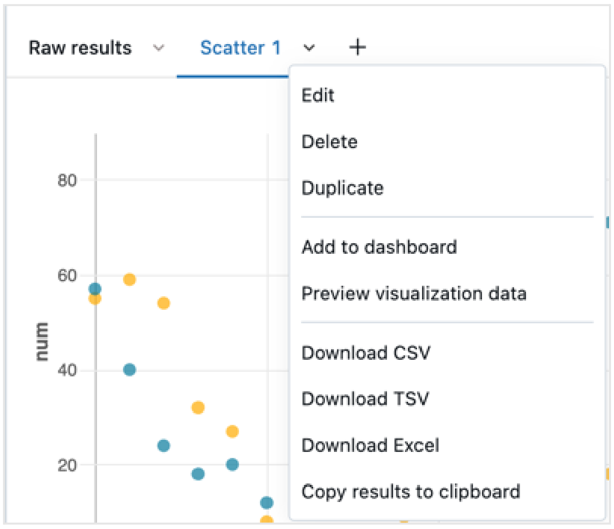
Legend placement: Automatic (Flexible)

3. After making configuration choices, click Save.

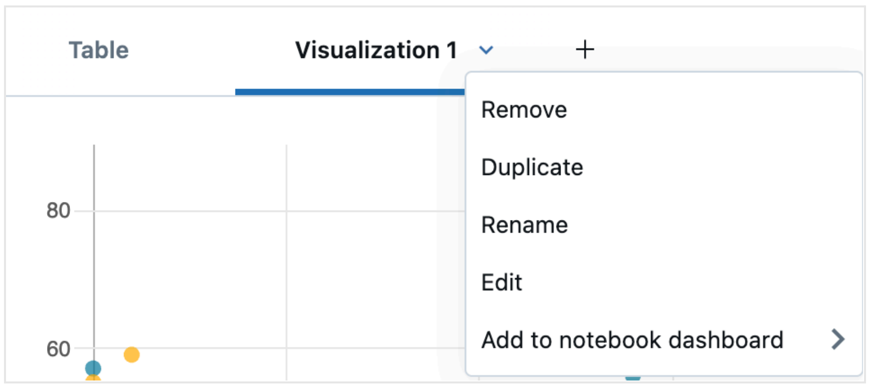
Remove, duplicate, or edit a visualization

To remove, duplicate, or edit a visualization or data profile, click the downward pointing arrow at the right of the tab name. You can also create a dashboard from the menu.

SQL editor



Notebook



You can also rename the tab by clicking directly on the name and editing the name in place.

Edit a visualization

To edit a visualization:

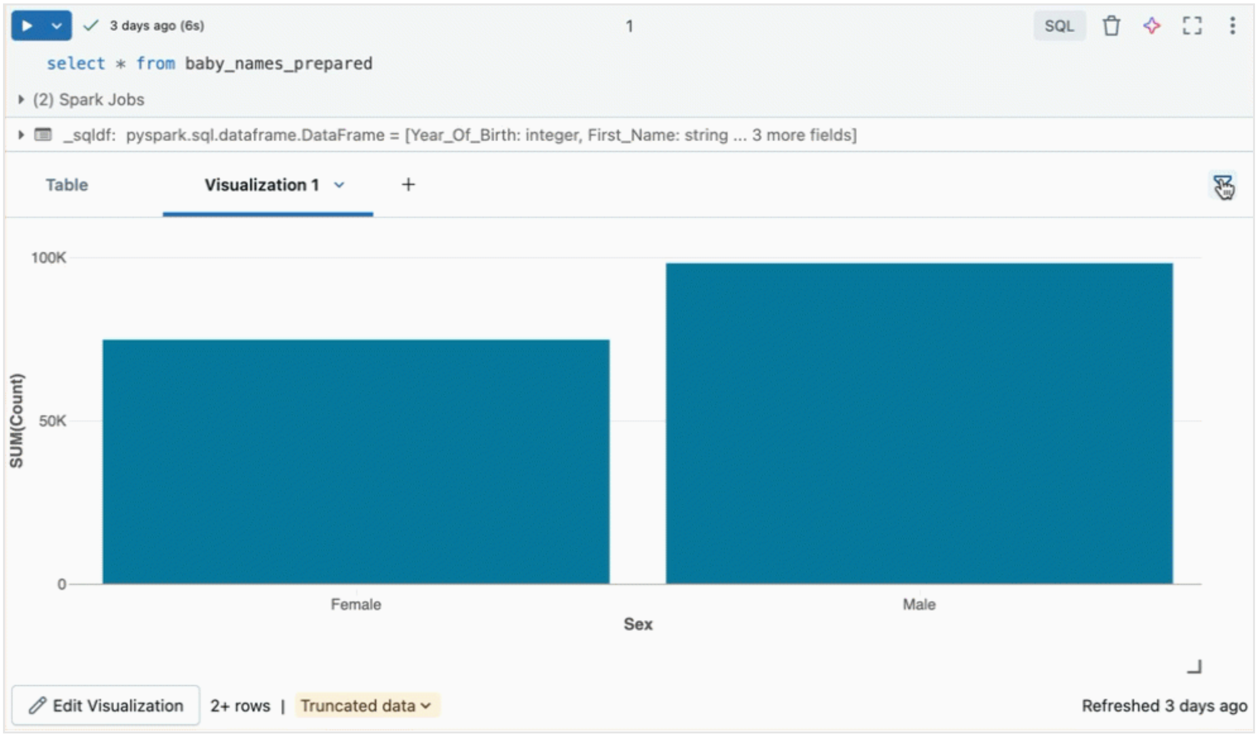
1. Click the downward pointing arrow in the visualization tab. Then, click Edit.

2. Use the tabs in the Visualization Editor to access and edit different parts of the chart.

Filter a visualization

To apply a filter on a visualization, click in the upper right corner and enter the filter conditions to apply.

Filter(s) applied on a visualization will also apply to the results table. Filter(s) applied on the results table will also apply to the visualization.



```
select * from baby_names_prepared
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Year_Of_Birth: integer, First_Name: string ... 3 more fields]

Clone a visualization

To clone a visualization, click the downward pointing arrow in the visualization tab. Then click Duplicate.

Enable aggregation in a visualization

When you make a **chart** (bar, line, pie, etc.) in Databricks, you can do the **totaling, counting, or averaging inside the chart settings** — you don't have to change your SQL query.

Why this is useful:

- You don't touch or rewrite the query — less chance of mistakes.
- You can quickly test different "what if" scenarios without editing code.
- The chart can work with the **entire dataset**, not just the first 64,000 rows that the table shows.

Where you can do this:

- Line chart
- Bar chart
- Area chart
- Pie chart
- Heatmap
- Histogram

Note: You can't use aggregation if your chart mixes two types (like line + bar together).

How to do aggregation on the Y-axis in a chart:

1. **Open the chart editor** — either make a new chart or edit one you already have.
    - If you see a message saying the chart uses an "old configuration," you'll need to **make a new chart** first to use aggregation.

2. **Find the Y-axis settings** in the editor.
    - There will be a dropdown where you can choose how to combine the numbers (**aggregation type**).

3. **Pick your aggregation type**:
    - If your Y-axis values are **numbers**, you can choose:
        - **Sum** (add them up) — *default*
        - **Average**
        - **Count** (how many values there are)
        - **Count Distinct** (how many different values there are)
          Max (largest value)
        - **Min** (smallest value)
        - **Median** (middle value)

    - If your Y-axis values are **text/strings**, you can choose:

        - **Count**

            - **Count Distinct**

4. **Click Save** — your chart will now show the aggregated values, and you'll see how many rows it used for the calculation.

Edit visualization colors

You can customize a visualization's colors when you create the visualization or by editing it.

1. Create or edit a visualization.
2. Click Colors.
3. To modify a color, click the square and select the new color by doing one of the following:
    - Click it in the color selector.
    - Enter a hex value.
4. Click anywhere outside the color selector to close it.
5. Click Save in the Visualization Editor to save the changes.

Temporarily hide or show a series

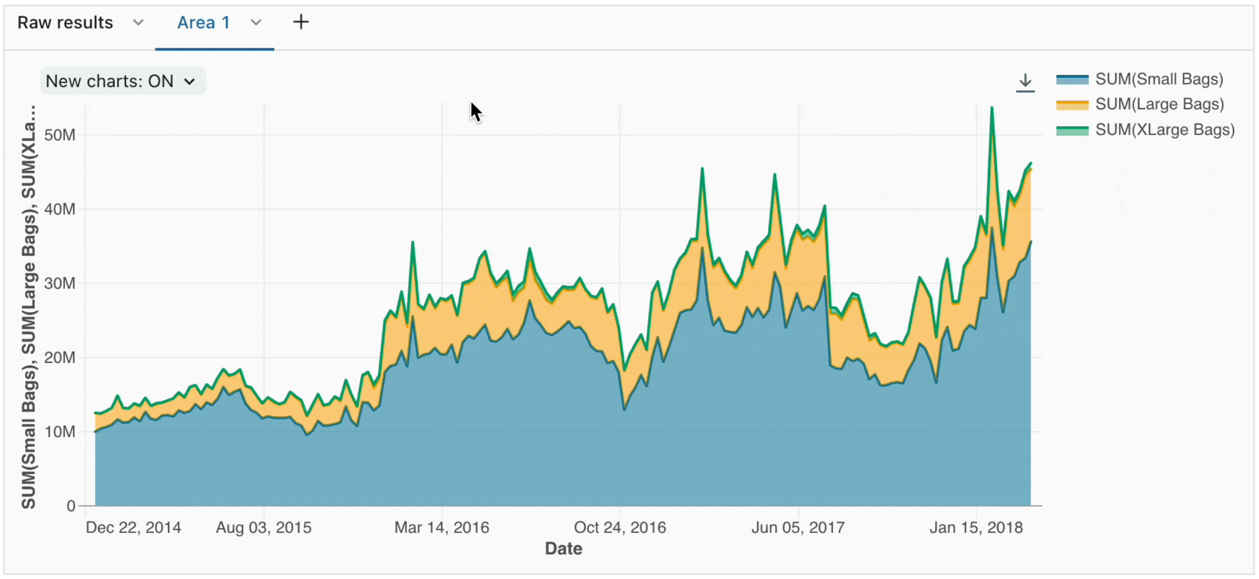
To hide a series in a visualization, click the series in the legend. To show the series again, click it again in the legend.

To show only a single series, double-click the series in the legend. To show other series, click each one.

Series selection

To select a specific series to analyze on a chart, use the following commands:

- Click on a single legend item to select that series
- Cmd/Ctrl + click on a legend item to select or deselect multiple series

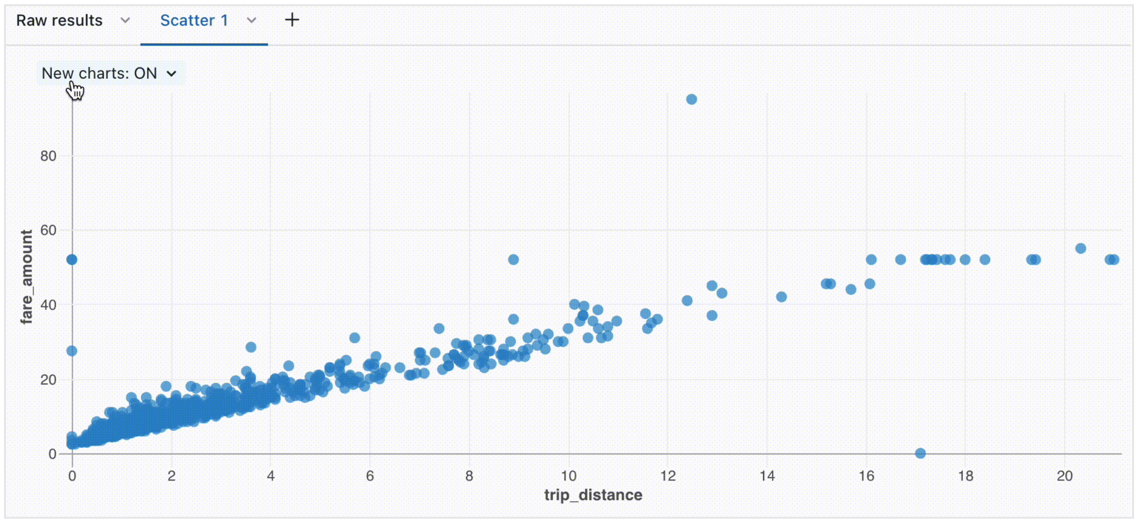


Sorted tooltips

Use tooltips on line charts and unstacked bar charts, ordered by magnitude, for easier analysis.

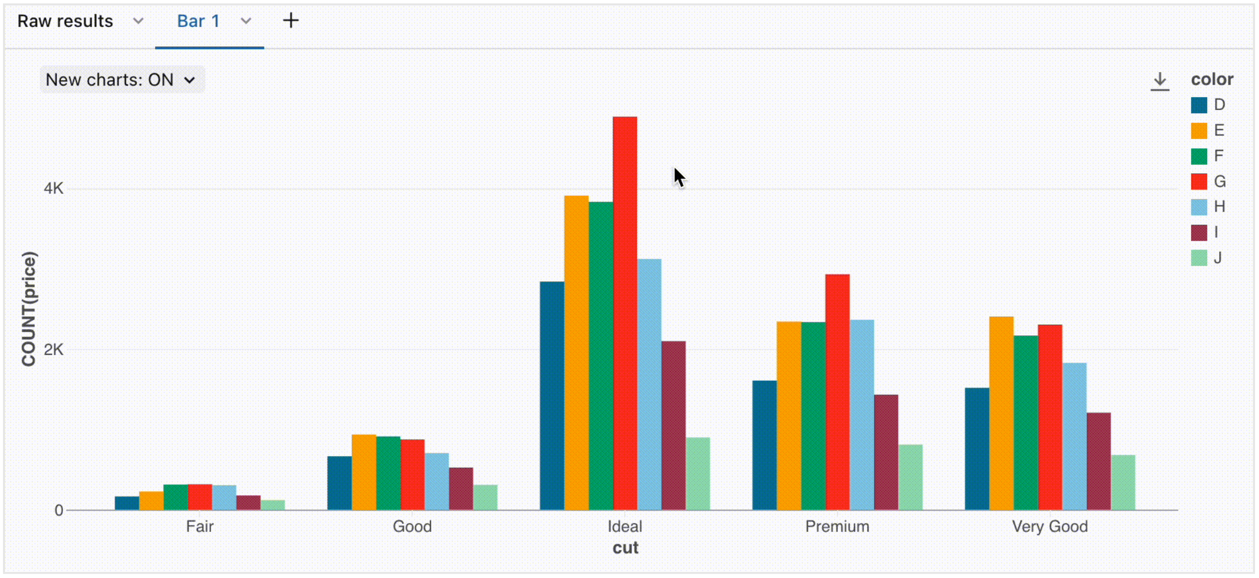
Zoom

For data-dense charts, zooming in on individual data points can be helpful to investigate details and to crop outliers. To zoom in a chart, click and drag on the canvas. To clear the zoom, hover over the canvas and click the Clear zoom button in the upper right corner of the visualization.



Download a visualization as a PNG file

To dowload a visualization as a PNG file, hover over the canvas and click the download icon in the upper-right corner.



A png file is downloaded to your device.

Visualization Types:

https://learn.microsoft.com/en-us/azure/databricks/visualizations/visualization-types