How to Use Effectively and Efficiently

Superset

v2.1.0

Created by: Boris Menshikov

Workflow with Superset

Query Configuration

↓

Creating and Configuring a Dataset

↓

Dashboard Creation

↓

Selection and Creation of Visualizations Based on the Dataset

↓

Implementation of panel and cross filters

↓

Dashboard Appearance Configuration

↓

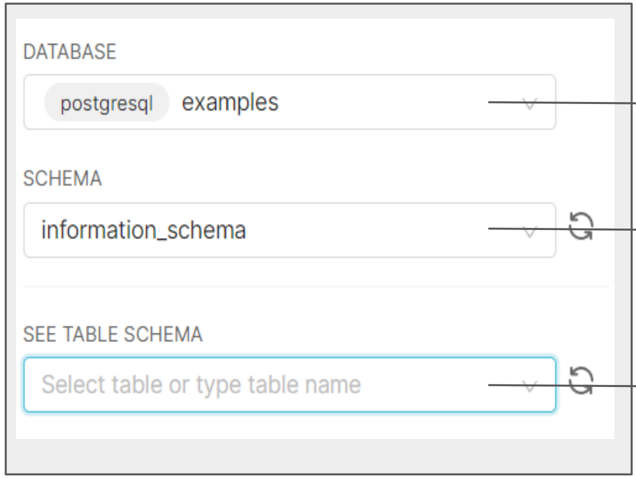
Final Check →

Query Configuration

- Creation of a Table for the Virtual Dataset
- Aggregatable Columns
- Selection of Required Attributes
- Logic of Filtering in Superset
- Life Hacks and Useful Features

Query Configuration | SQL Lab

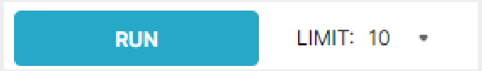
Autocomplete will only work if the corresponding schema is selected in the left panel.



To start working with the query, you need to select a database

Choosing a schema allows you to quickly type table names (or column names inside tables) in that schema using the TAB key

Allows you to preview the table you need to work with. Does not affect the query



Shows a preview of the table, limited by the size specified in the 'limit' field. Does not affect data availability for visualization based on this query.

Useful features:

Necessity of use:

Save Query

The basics of why this feature is needed and how to use it with maximum efficiency. It allows you to save time, nerves and speed up the process of creating queries of the same structure for different purposes:

Deferred configuration	The request needs to be finalized
Share sample queries	Make it easier for colleagues to work
Personal Use	Optimization, etc

Useful features:

Necessity of use:

Query History

The basics of why this feature is needed and how to use it with maximum efficiency. It allows one to find shortcomings, see the difference and determine the best query options. When using it, one can subtract the strengths and weaknesses of work requests.

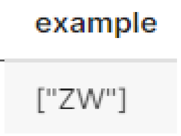
Restoring Queries	If the needed queries are lost
Optimization and monitoring	Shows the difference in execution speed
Result and syntax	Attempts analysis

Useful features:

Work with
Hidden Attributes

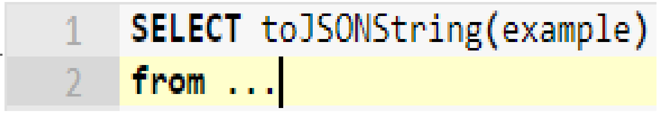
When analyzing data, it is sometimes necessary to work with attribute values that are not clearly visible or easily accessible. In this case, we encounter non-obvious attributes with no name in the data table

In the example column there is a non-obvious attribute "ZW"



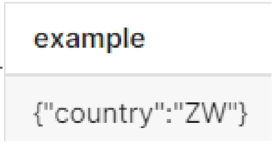
example
["ZW"]

In order to find out the name of the attribute, one needs to refer to the column using toJSONString



```
SELECT toJSONString(example)
from ...
```

As a result, we will get the name of all attributes and sub-attributes, if there are any



example
{"country":"ZW"}

Parameters:

Types of parameters and their purposes:

SQL lab parameters

Narrow scope of application, limited functionality, cannot be automated

Parameters with custom attributes

Parameters using other datasets

Filter Parameters

Convenient when creating a multi-level query, there are no restrictions on use

Time Parameter

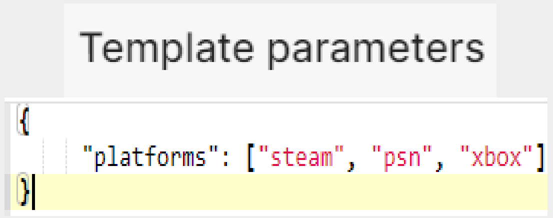
Value Parameter

Parameters with custom attributes:

Jinja Templating parameters

are used to filter the query by using the WHERE clause

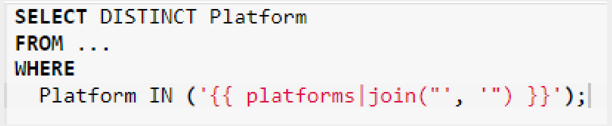
Specifying the parameters



Template parameters

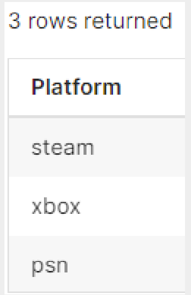
```
{
    "platforms": ["steam", "psn", "xbox"]
}
```

Referring to the parameter in the query



```
SELECT DISTINCT Platform
FROM ...
WHERE
  Platform IN ('{{ platforms|join("', '") }}');
```

The result



3 rows returned

Platform
steam
xbox
psn

* When using these parameters, it is important to understand that their structure does not change automatically, they must be updated if necessary

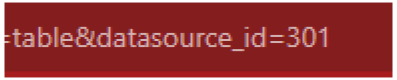
Parameters using other datasets

Parameters from other datasets

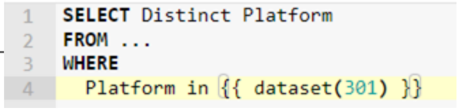
They are needed to use columns and attributes from already existing datasets.

Find out the path to the dataset in which the future parameter will be

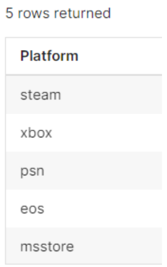
Go to the dataset and look at the link, in this case number 301


=table&datasource_id=301

Accessing the dataset using where clause


```
SELECT Distinct Platform
FROM ...
WHERE
  Platform in {{ dataset(301) }}
```

As a result, we select only those platforms that are in the dataset 301


5 rows returned

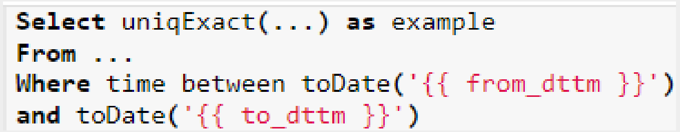
Platform
steam
xbox
psn
eos
msstore

Filter Parameters:

Working with
Time Filter Parameter

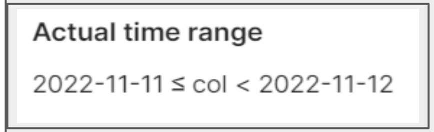
Function: Query data based on the selected time interval.
Special condition:
1) A Time Range filter has been created and a time period has been selected
2) The visualization to which the dataset is linked is linked to the filter using Scope

From - is the beginning to - the end of the period



```
Select uniqExact(...) as example
From ...
Where time between toDate('{{ from_dttm }}')
and toDate('{{ to_dttm }}')
```

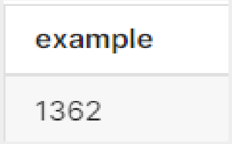
Select a period in the Time Range filter



Actual time range

2022-11-11 ≤ col < 2022-11-12

Result of the executed request



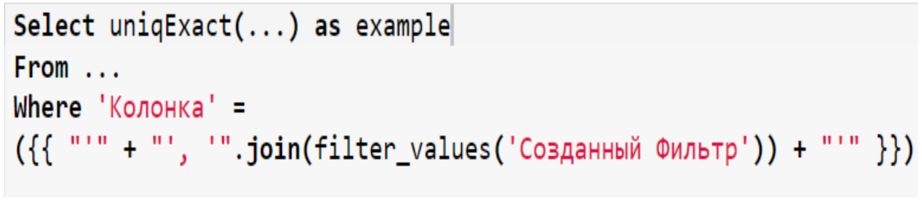
example
1362

Filter Parameters:

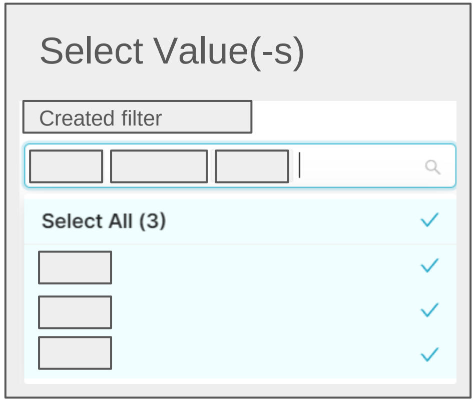
Working with the Value Filter Parameter

Function: Query data based on the selected attribute in the column.
Special condition:
1) The Value filter has been created and the attribute(s) have been selected
2) The visualization to which the dataset is linked is linked to the filter using Scope

Refer to the filter in the request



```
Select uniqExact(...) as example
From ...
Where 'Колонка' =
({{ "'" + "', '".join(filter_values('Созданный Фильтр')) + "'" }})
```



Creating and configuring a dataset

- Saving the Original Query Table
- Creating persistent metrics
- Configuring the assigned columns

Types of datasets:

Table Dataset

Dataset created from a table
(It is advisable to use already aggregated tables)

One can aggregate data using Saved Metrics & Calculated Columns

Takes a lot of time to process raw data

Virtual Dataset

Dataset created using a query
(One can aggregate data at the query level)

One can aggregate data in any way possible

Processing depends directly on the speed of the request

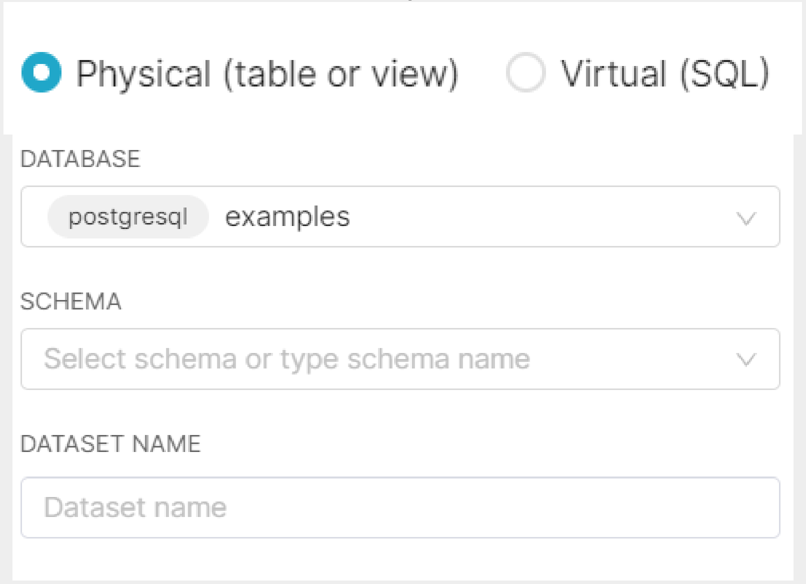
Dataset configuration

1. Editing a query in Source
2. Creating permanent metrics in Metrics
3. Selecting Column Structures in Columns
4. Creating Aggregated Columns in Calculated Columns
5. General dataset settings in Setting

Editing a query in Source

Main functions:

Change the dataset type:
And select a table, Database



[✓ Physical (table or view)] Virtual (SQL)

DATABASE

postgresql examples

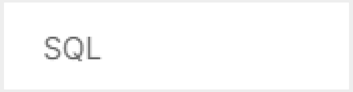
SCHEMA

Select schema or type schema name

DATASET NAME

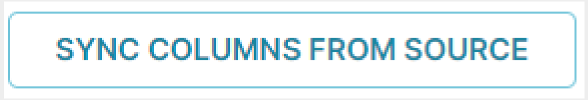
Dataset name

Modify the query directly in the Source (If it is virtual)



SQL

When modifying, sync the dataset



Metrics:

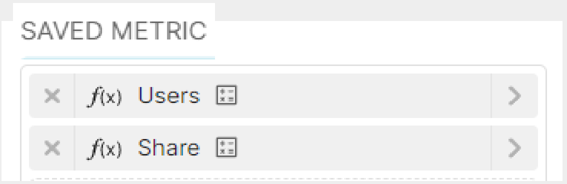
Creating permanent metrics

Metrics will be available when one creates visualizations

To simplify the creation of visualizations, one can create the main ones, then use them for their intended purpose, for specifics: a dataset is created, and then all the metrics with which the work will be performed are saved and used later

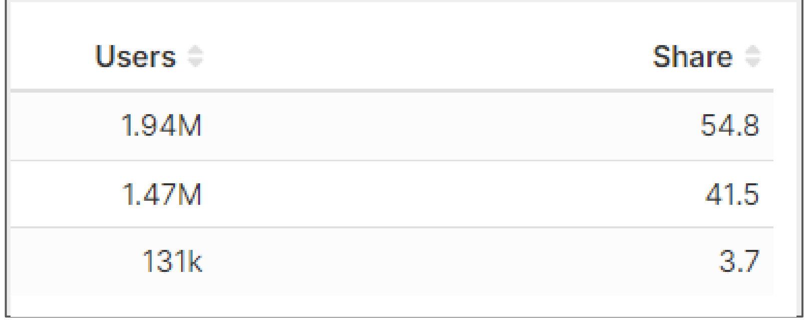
Creating complex metrics (Metric for Metric Aggregations)

```
Users as SUM(Users),
Share as
round(100*(Users/sum(Users) over ()), 1)
```


SAVED METRIC

- f(x) Users
- f(x) Share

Application in the table chart



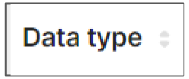
Users	Share
1.94M	54.8
1.47M	41.5
131k	3.7

Columns:

Columns Configuration

Columns will be available when the charts are configured

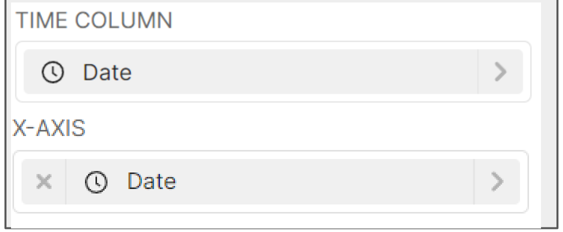
One can see the data type of the columns in the table


Data type

Their parameters can be configured:

Default datetime

Mainly used for:


TIME COLUMN

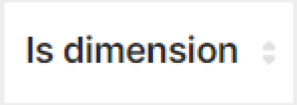
Date

X-AXIS

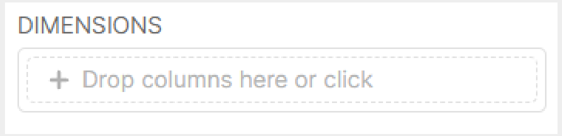
Date

Is filterable

Whether the column will be filtered


Is dimension

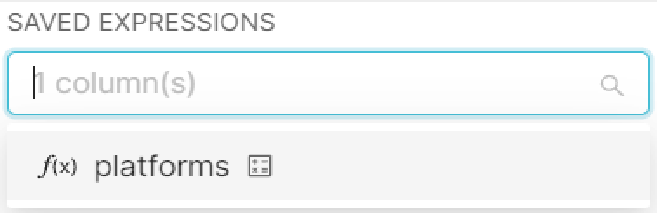
Mainly used for:


DIMENSIONS

Calculated columns:

Create permanent columns

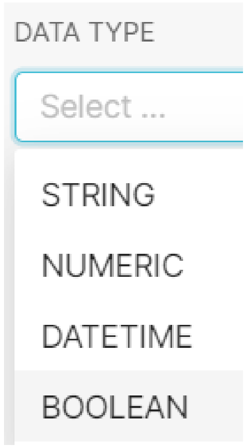
Columns will be available when creating visualizations in:


SAVED EXPRESSIONS

1 column(s)

f(x) platforms

Using Sql expression, one can customize the purpose of the column, as well as the type of data


DATA TYPE

Select ...

STRING

NUMERIC

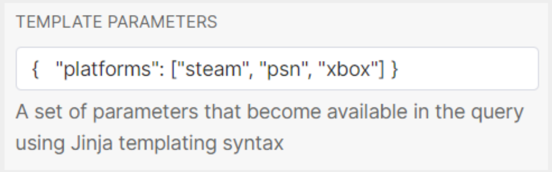
DATETIME

BOOLEAN

Settings:

Template parameters

Parameters will only work if they are specified


TEMPLATE PARAMETERS

```
{   "platforms": ["steam", "psn", "xbox"] }
```

A set of parameters that become available in the query using Jinja templating syntax

Owners

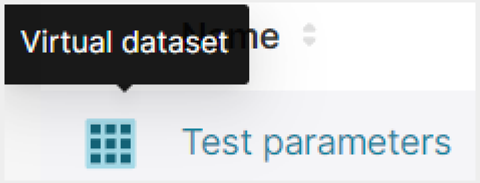
Users who can configure the dataset

Charts creation:

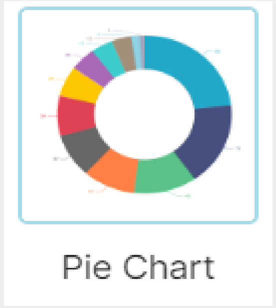
1. Creating a dashboard
2. Choosing a dataset
3. Choosing a chart type
4. Chart configuration
5. Saving the chart on the dashboard

Selecting and creating a visualization:

Choosing a dataset

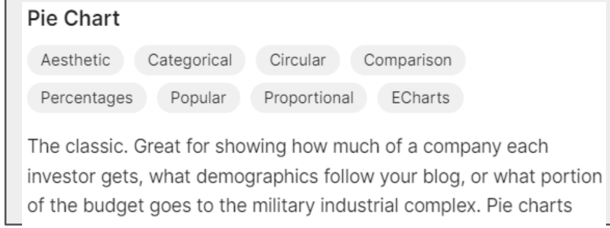


Choosing the chart we need



Description

(The purpose of the chart is known)



Pie Chart

Aesthetic Categorical Circular Comparison Percentages Popular Proportional ECharts

The classic. Great for showing how much of a company each investor gets, what demographics follow your blog, or what portion of the budget goes to the military industrial complex. Pie charts

Moving to the chart configuration space

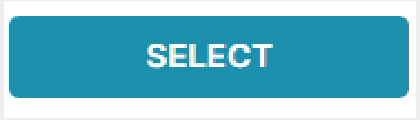
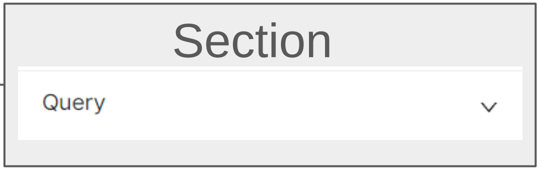


Chart configuration:

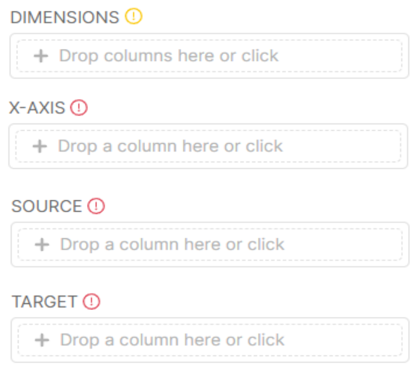
To create a chart, one basically needs:


Section

Query

You can constantly repeat the same thing in every chart, over and over again, with a chance to make a mistake, create and not save objects in the Query section, or you can academically and without errors use the saved resources, such as:

Calculated columns:


DIMENSIONS

+ Drop columns here or click

X-AXIS

+ Drop a column here or click

SOURCE

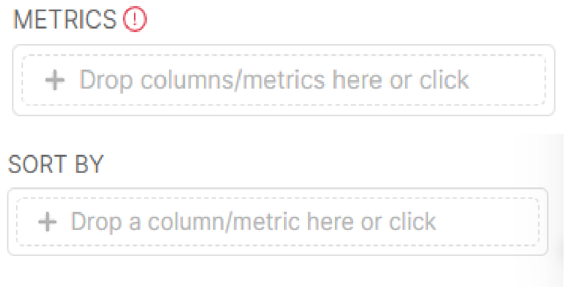
+ Drop a column here or click

TARGET

+ Drop a column here or click

&

Metrics:

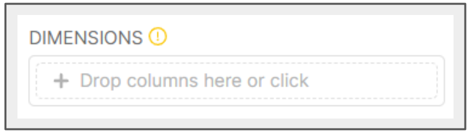

METRICS

+ Drop columns/metrics here or click

SORT BY

+ Drop a column/metric here or click

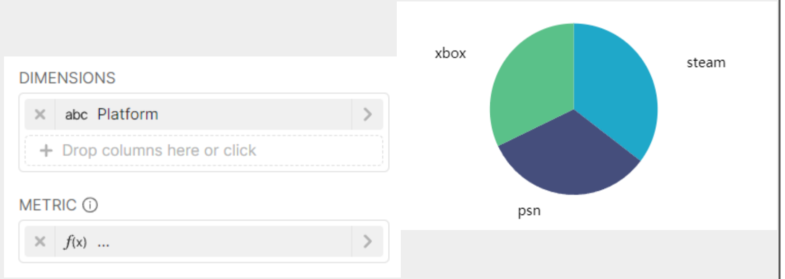
Examples of configurations by Cal. Columns:


DIMENSIONS

+ Drop columns here or click

Change of the metric based on the selected space

Example on the pie chart

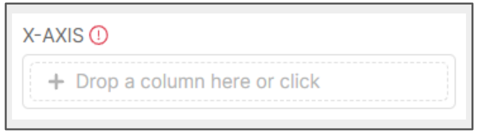

DIMENSIONS

abc Platform

+ Drop columns here or click

METRIC

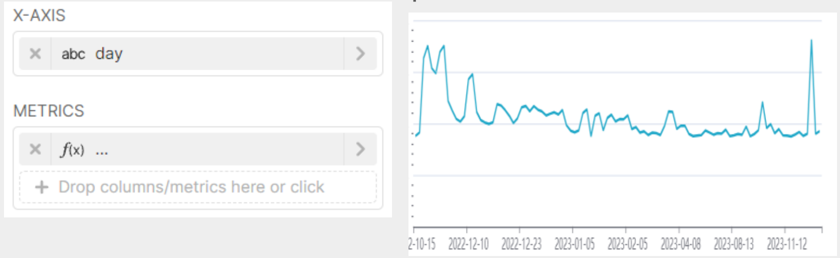
$f(x)$ ...


X-AXIS

+ Drop a column here or click

Changing a metric based on an axis attribute

Example on the Line Chart


X-AXIS

abc day

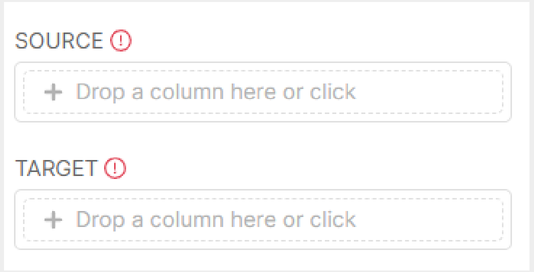
METRICS

$f(x)$ ...

+ Drop columns/metrics here or click

Examples of configurations by Cal. Columns:

Used for column hierarchy in the chart



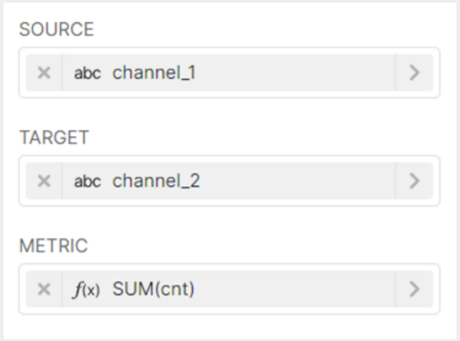
SOURCE ⓘ

+ Drop a column here or click

TARGET ⓘ

+ Drop a column here or click

Example: Chord diagram



SOURCE

× abc channel_1 >

TARGET

× abc channel_2 >

METRIC

× f(x) SUM(cnt) >

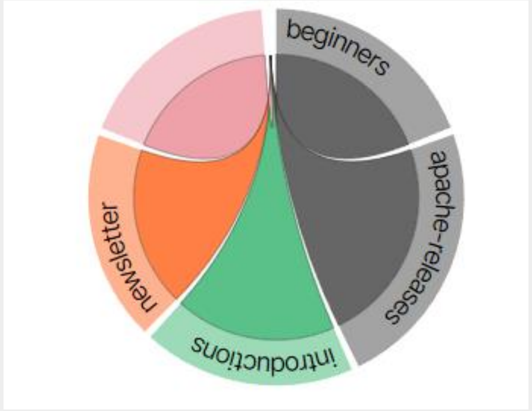
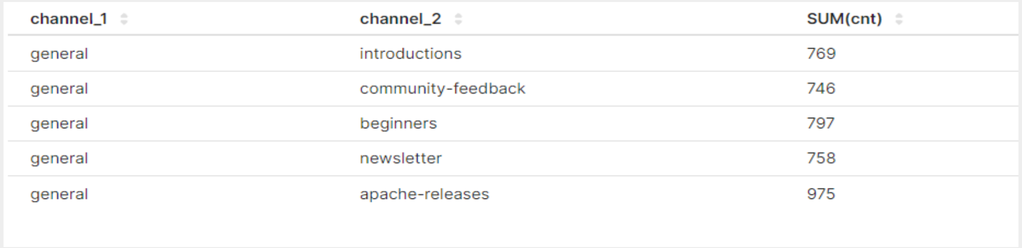
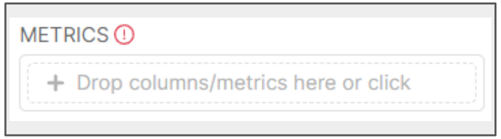


Table with the result

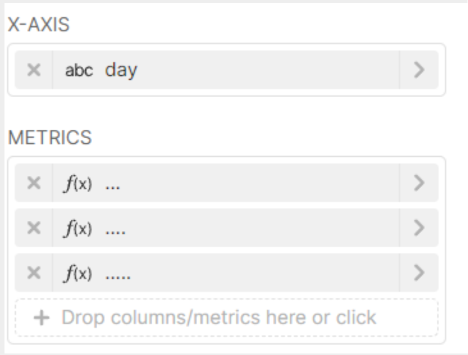


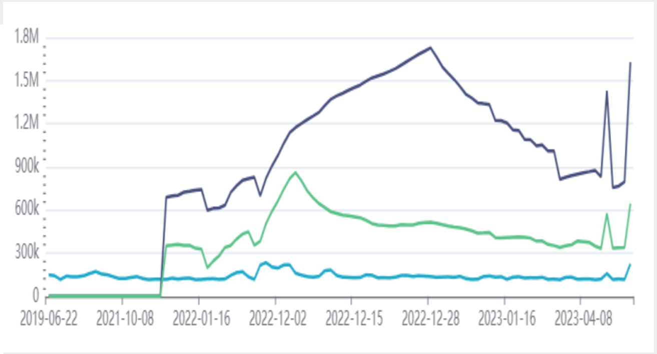
channel_1	channel_2	SUM(cnt)
general	introductions	769
general	community-feedback	746
general	beginners	797
general	newsletter	758
general	apache-releases	975

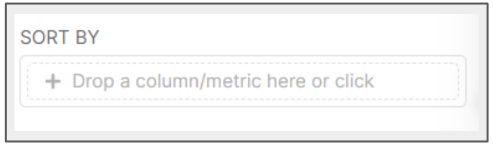
Examples of configurations by Metrics:

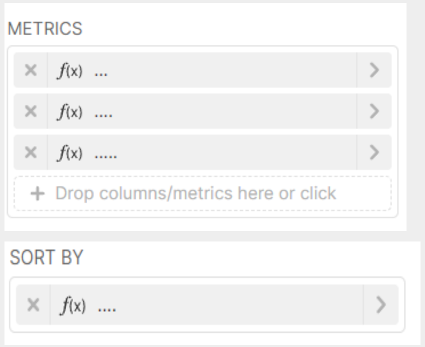

METRICS
Drop columns/metrics here or click

Example on the line chart

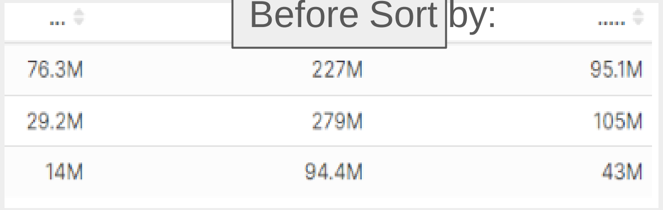

X-AXIS
abc day
METRICS
f(x) ...
f(x) ....
f(x) .....
Drop columns/metrics here or click




SORT BY
Drop a column/metric here or click

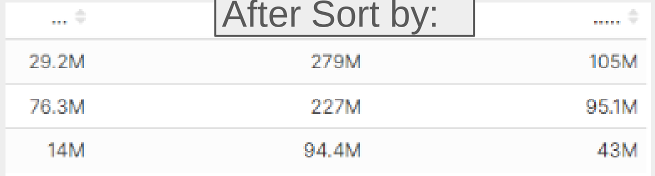

METRICS
f(x) ...
f(x) ....
f(x) .....
Drop columns/metrics here or click
SORT BY
f(x) ....

Before Sort by:



...		.....
76.3M	227M	95.1M
29.2M	279M	105M
14M	94.4M	43M

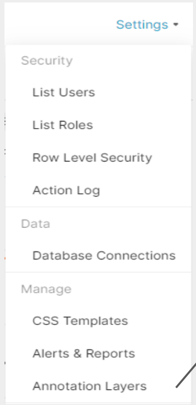
After Sort by:



...		.....
29.2M	279M	105M
76.3M	227M	95.1M
14M	94.4M	43M

Annotations on the charts:

For the appearance of time legends that show the reason for the abrupt change in the data, one can use annotations.

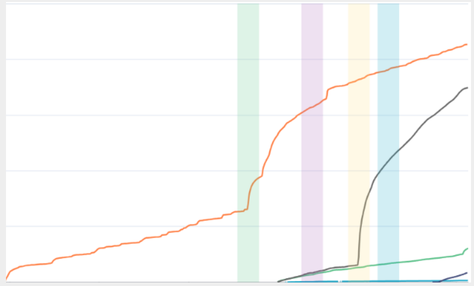


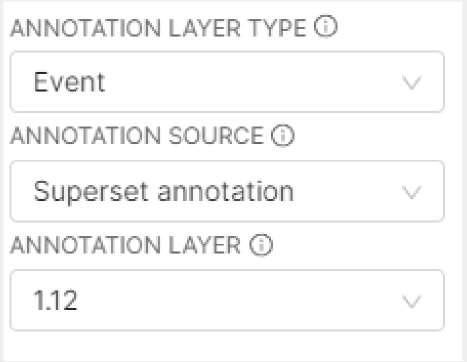
Moving on to the annotation space

Creating an annotation container

Creating annotations

Choosing a container for annotations and all events (in the form of annotations) appear on the chart





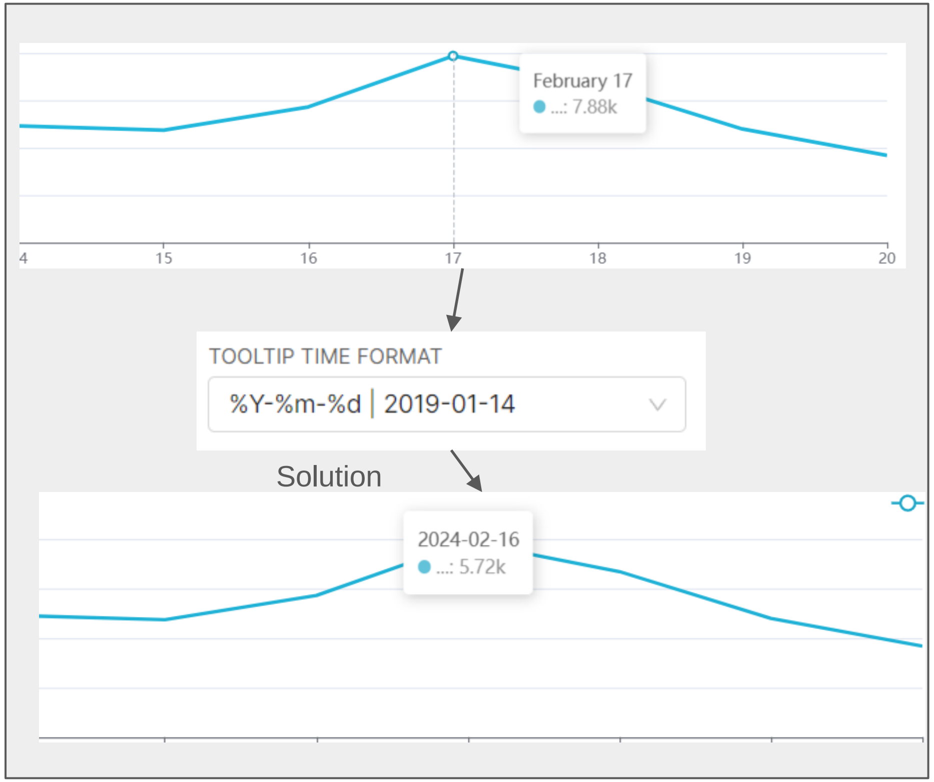
ANNOTATION LAYER TYPE: Event

ANNOTATION SOURCE: Superset annotation

ANNOTATION LAYER: 1.12

Main functions of Customise:

Date formatting

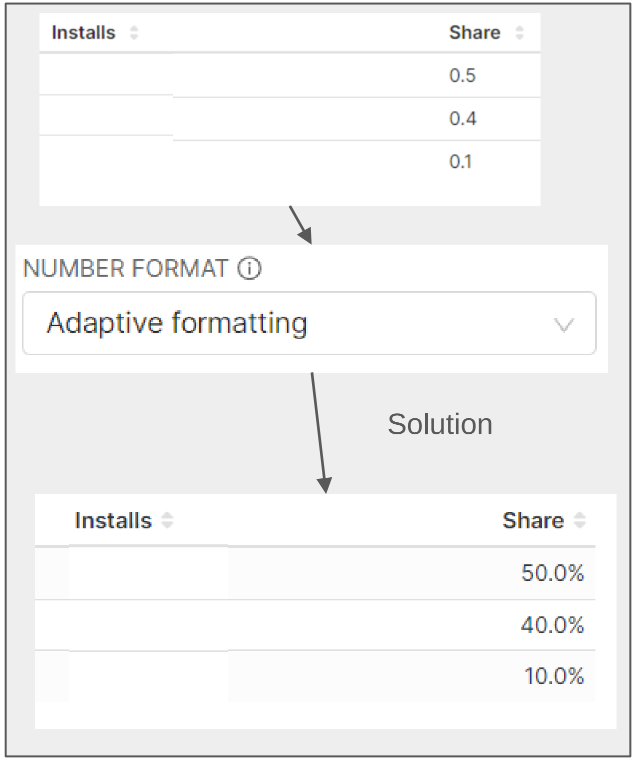


TOOLTIP TIME FORMAT

%Y-%m-%d | 2019-01-14

Solution

Formatting numbers



Installs	Share
	0.5
	0.4
	0.1

NUMBER FORMAT

Adaptive formatting

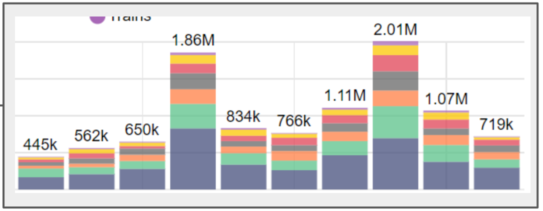
Solution

Installs	Share
	50.0%
	40.0%
	10.0%

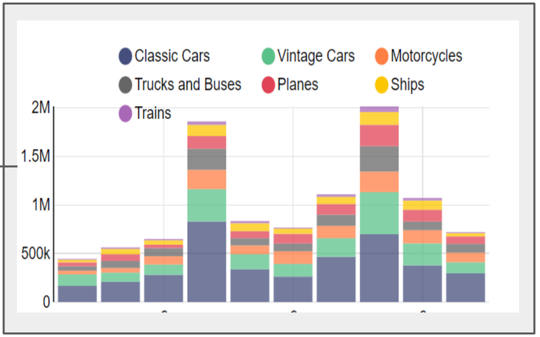
Main functions of Customise:

Useful features:

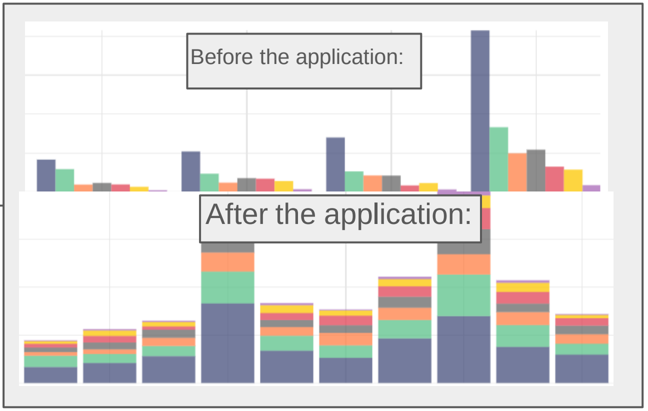
✓ SHOW VALUE



✓ LEGEND



✓ STACKED BARS



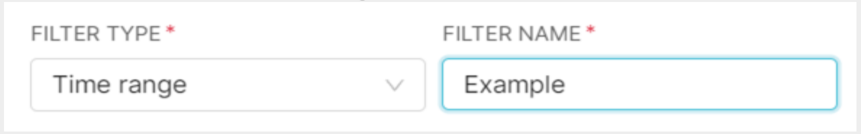
Implementation of panel filters:

1. Using Dataset to create a filter
2. Selection of a column that has attributes to filter

Implementation of panel filters:

Time Range

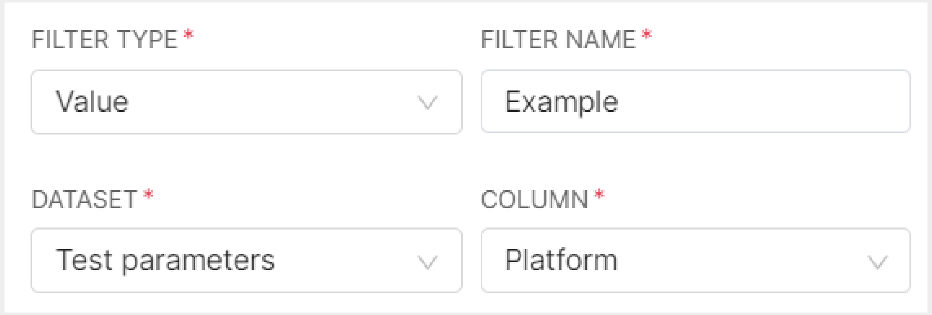
Creating a Filter


FILTER TYPE * Time range
FILTER NAME * Example

Value filter

Before creating, make sure that the dataset has the required attribute

Creating a Filter


FILTER TYPE * Value
FILTER NAME * Example
DATASET * Test parameters
COLUMN * Platform

SCOPING

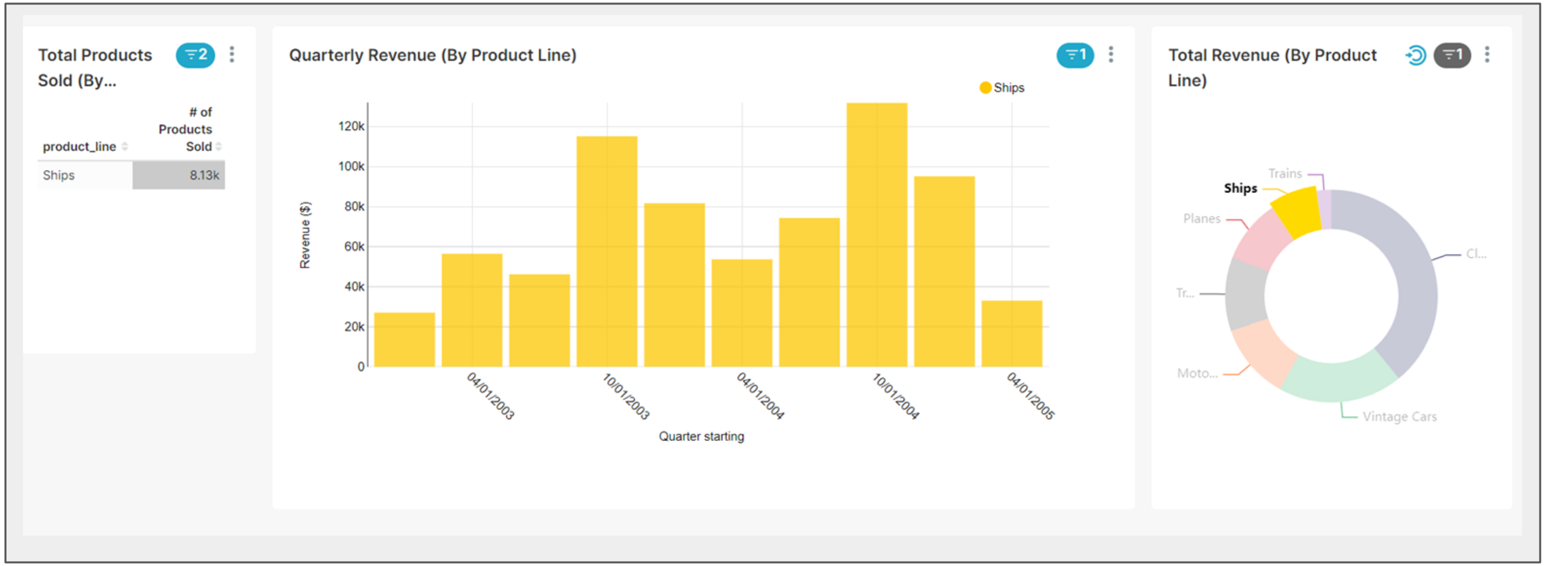
Needed to bind filters to visualizations

Implementation of cross-filters:

They are created automatically when you create a chart, mainly the Dimension tab

Cross-filters affect the entire dashboard, which sometimes breaks visualizations when attributes are missing

Some types of charts do not have cross-filters



product_line	# of Products Sold
Ships	8.13k

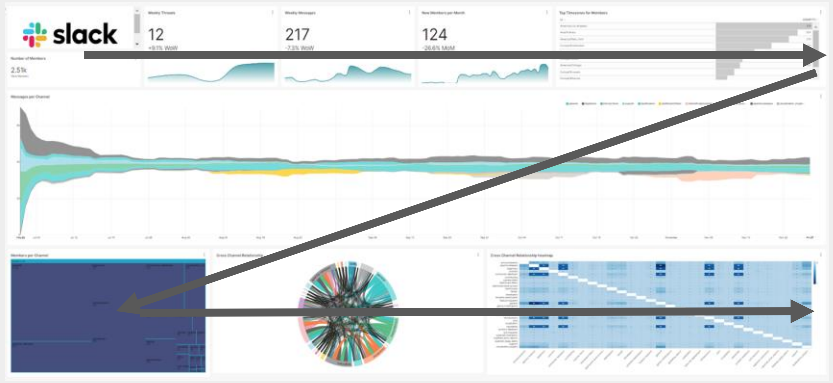
Dashboard appearance changes:

1. Choosing the type of layout
2. Logical division of visualization types into sections, if necessary, pages
3. CSS customization, Individualization of specific attributes

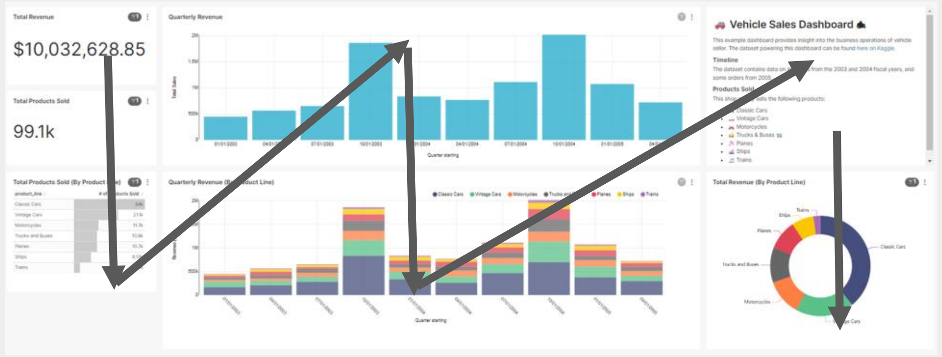
Dashboard appearance configuration:

Choosing the type of layout

Reading order from left to right



Newspaper layout


Total Revenue
$10,032,628.85
Total Products Sold
99.1k
Total Products Sold (By Product Line)
Quarterly Revenue
Quarterly Revenue (By Product Line)
Vehicle Sales Dashboard
Total Revenue (By Product Line)

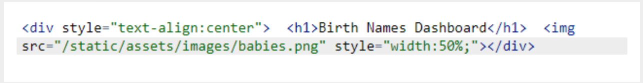
Layout element :

Text

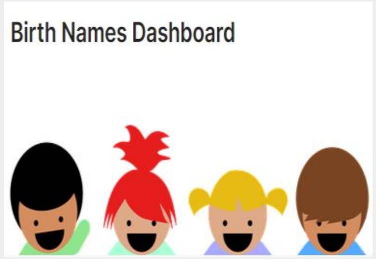
To format text in a given Layout, one can use: Markdown and HTML

Needed to indicate data loss issues or to explain attributes

One can also use it this way:



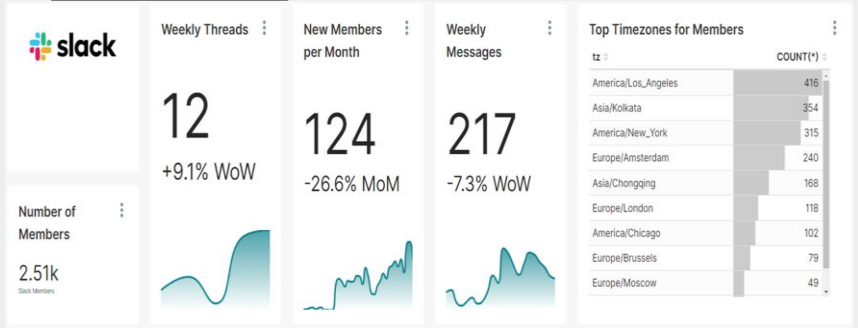
```
<div style="text-align:center">  <h1>Birth Names Dashboard</h1>  <img
src="/static/assets/images/babies.png" style="width:50%;"></div>
```



Layout element Tab:

Example of pagination with visualizations

Whole page

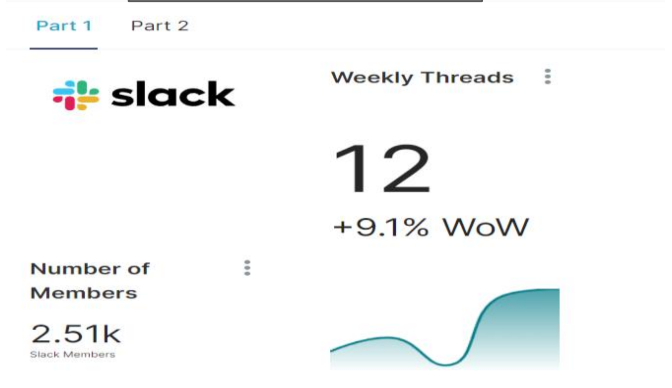


Top Timezones for Members

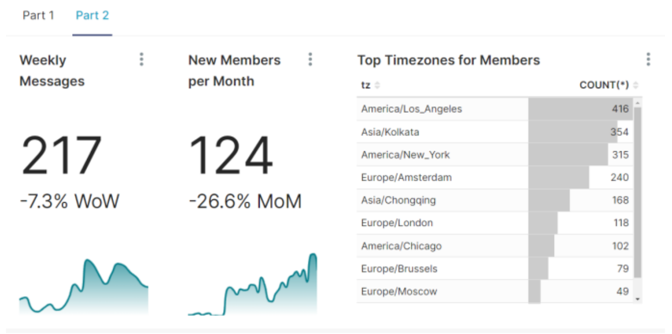
tz	COUNT(*)
America/Los_Angeles	416
Asia/Kolkata	354
America/New_York	315
Europe/Amsterdam	240
Asia/Chongqing	168
Europe/London	118
America/Chicago	102
Europe/Brussels	79
Europe/Moscow	49

One can save space, and put less important information on lower levels using Tab

First tab



Second tab

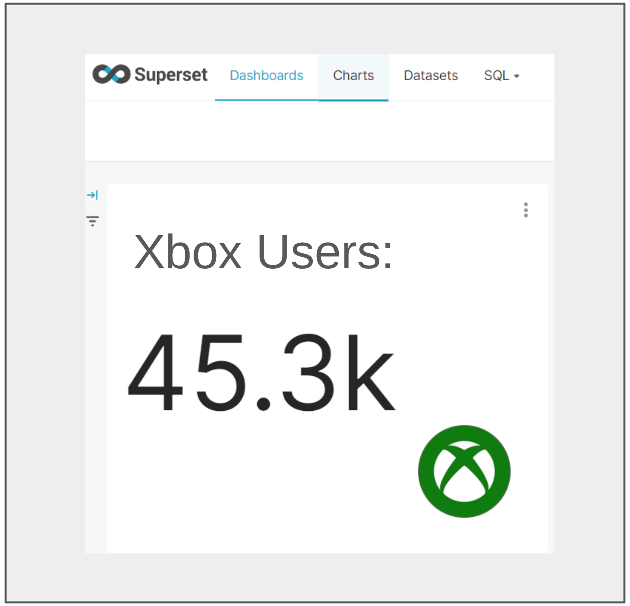


Top Timezones for Members

tz	COUNT(*)
America/Los_Angeles	416
Asia/Kolkata	354
America/New_York	315
Europe/Amsterdam	240
Asia/Chongqing	168
Europe/London	118
America/Chicago	102
Europe/Brussels	79
Europe/Moscow	49

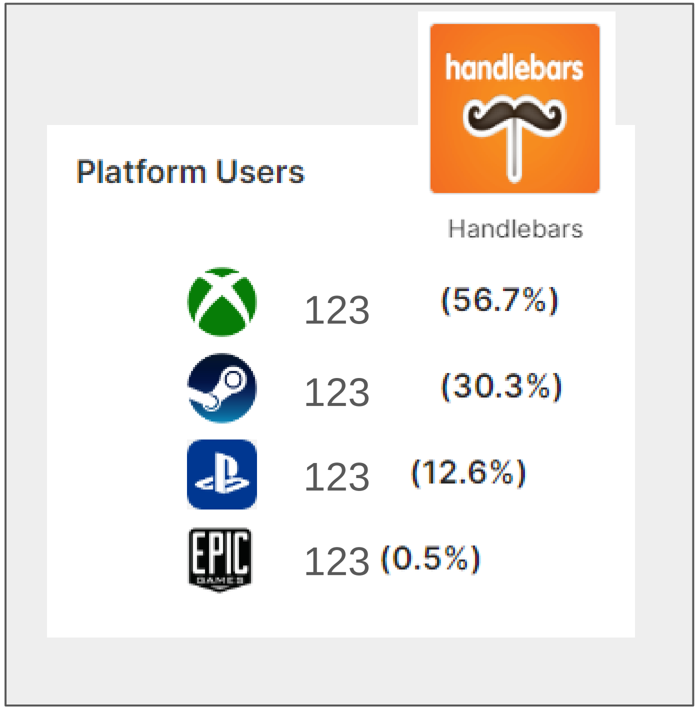
CSS and attribute customization:

It's inconvenient and doesn't work well



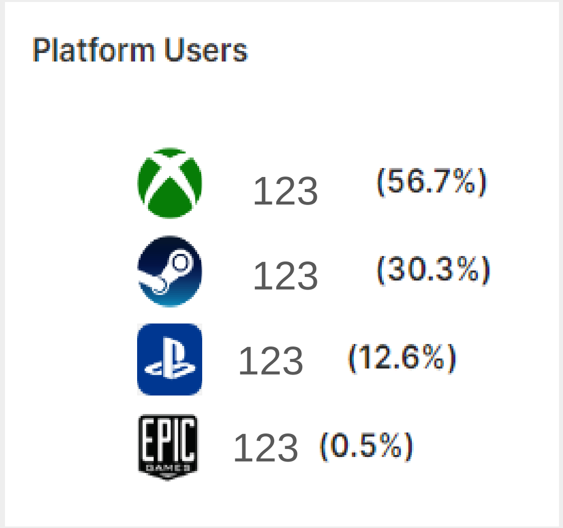
Big Number + CSS

The best solution



Handlebars + CSS

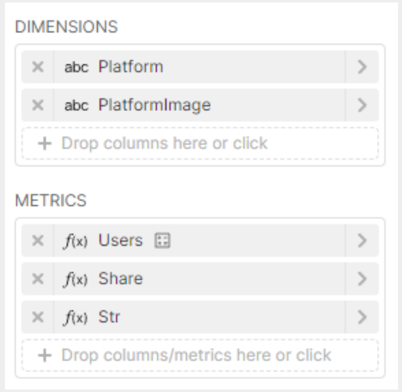
CSS and attribute customization:



Platform Users

	123	(56.7%)
	123	(30.3%)
	123	(12.6%)
	123	(0.5%)

Create an Image Url column and link them to platforms



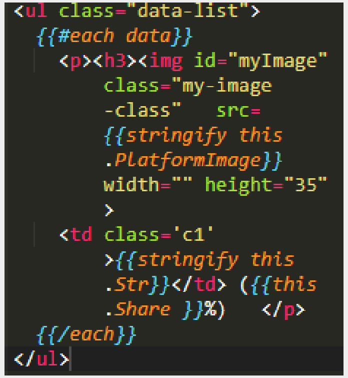
DIMENSIONS

- abc Platform
- abc PlatformImage
- + Drop columns here or click

METRICS

- f(x) Users
- f(x) Share
- f(x) Str
- + Drop columns/metrics here or click

Output using css via stringify this Metrics and columns

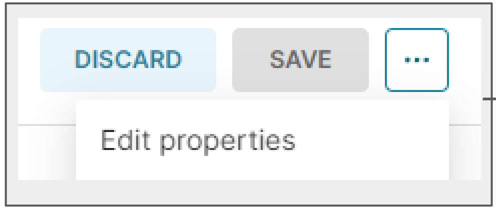


```
<ul class="data-list">
  {{#each data}}
    <p><h3><img id="myImage"
        class="my-image
        -class"   src=
        {{stringify this
        .PlatformImage}}
        width="" height="35"
        >
    <td class='c1'
        >{{stringify this
        .Str}}</td> ({{this
        .Share }}%)   </p>
  {{/each}}
</ul>
```

After all the manipulations are made, one will get an attractive visualization that is easy to work with.

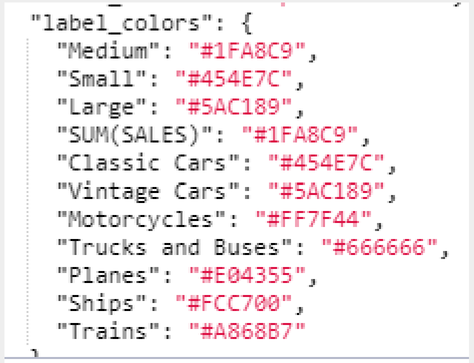
CSS and attribute customization:

How do you make sure that all the Legends in different charts are the same color?

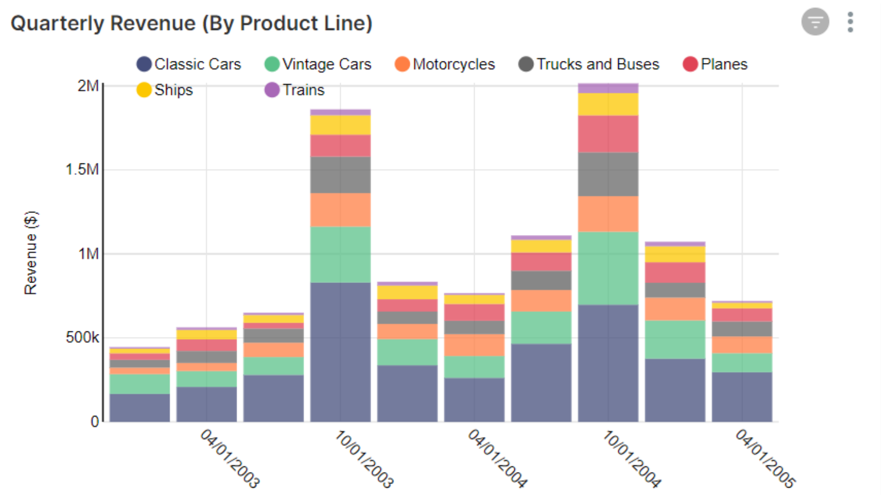


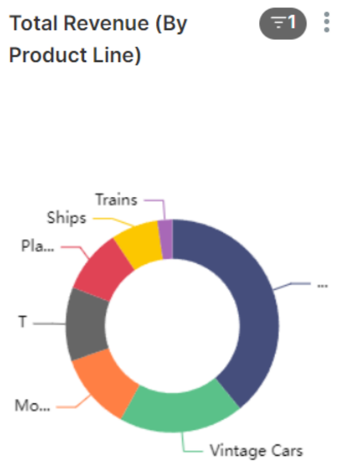
ADVANCED

Specify the name and the colour of each attribute here:



```
"label_colors": {
  "Medium": "#1FA8C9",
  "Small": "#454E7C",
  "Large": "#5AC189",
  "SUM(SALES)": "#1FA8C9",
  "Classic Cars": "#454E7C",
  "Vintage Cars": "#5AC189",
  "Motorcycles": "#FF7F44",
  "Trucks and Buses": "#666666",
  "Planes": "#E04355",
  "Ships": "#FCC700",
  "Trains": "#A868B7"
}
```





The end

Thank you for your attention!