Highcharts Library (Waterfall Chart)

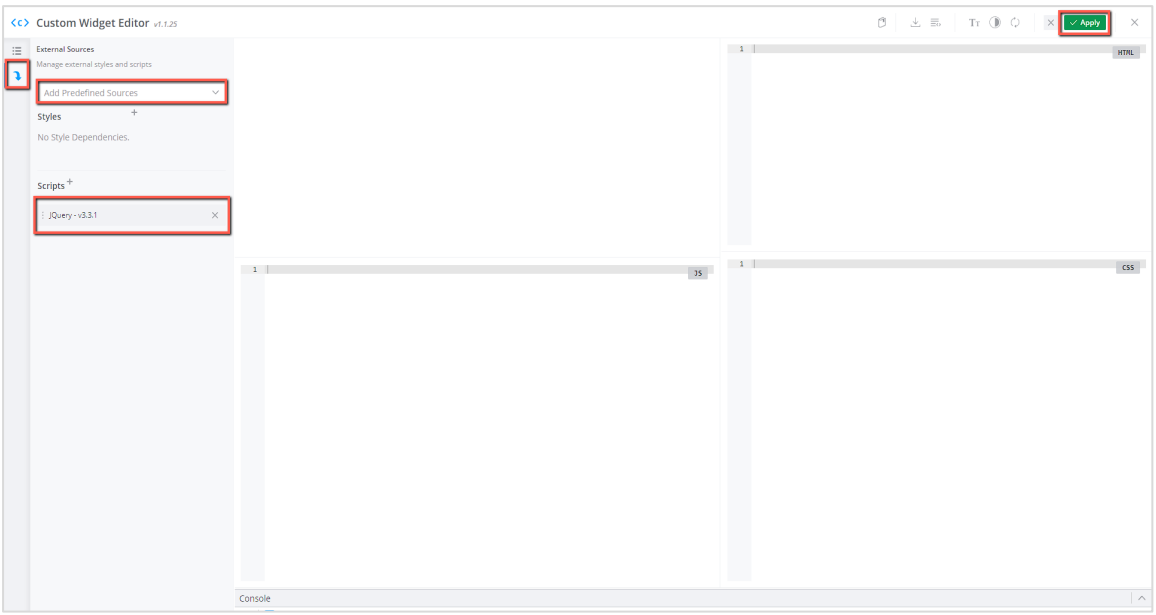
We will be creating a waterfall chart using the HighCharts library in DTR custom widget.

https://platform.datorama.com/external/dashboard?embedpage=1ea882a8-a529-472b-910a-d111902a3fd8

Add HighCharts library to DTR custom widget

1. We start by adding a custom widget to our DTR dashboard page.
2. Click on "Custom Widget Editor". Next, click on the "External Sources" icon.
3. HighCharts library is a standalone framework and does not require any additional frameworks or plugins to work. But since we are using jQuery functions to help us with other parts of the code, we require to import the jQuery library before importing the HighCharts library as it has a dependency on it.

   Select jQuery from "Add Predefined Sources" the DTR-supported version is (jQuery - v3.3.1).



4. Next, import the HighCharts library CDN, which also imports the Utility functions for the waterfall chart. We can refer to these links to know more about other charts.
   https://www.highcharts.com/docs/getting-started/installation

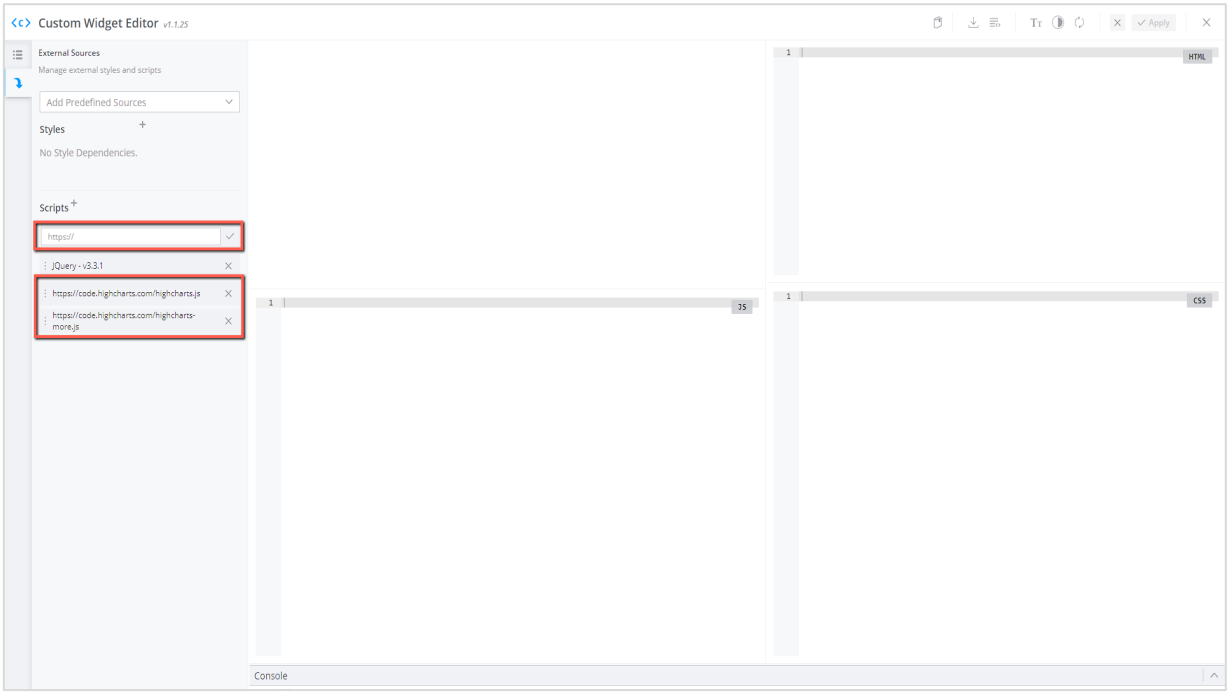
https://api.highcharts.com/highcharts/

CDNs to import in custom widget external sources (Import after jQuery library)

a. [jQuery – import]

b. https://code.highcharts.com/highcharts.js

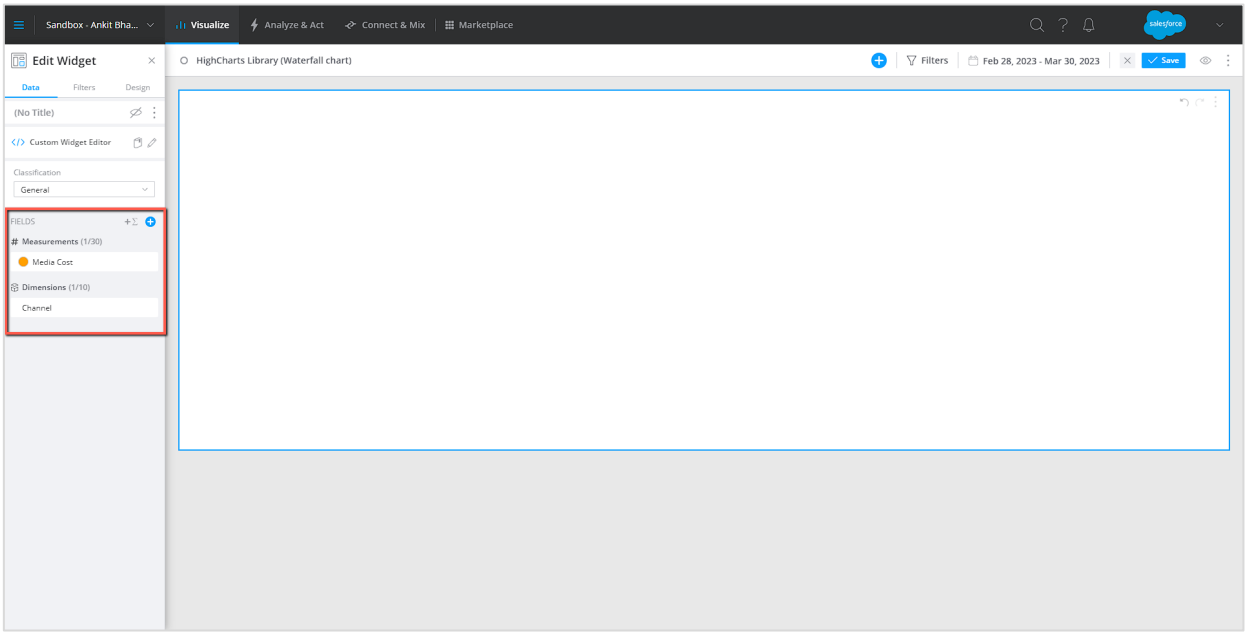
c. https://code.highcharts.com/highcharts-more.js



Add measurements & dimensions

1. Go to "Edit Widget", then "Data" section to add one measurement and one dimension.

(Depending on our use case and the JS code is customized accordingly).



Add Waterfall chart JS code

1. According to our use case we require a waterfall chart that defines Media Cost by Channel. (The JS code is dynamic so we can select any one measurement & one dimension for our chart).
2. We need to add an HTML block with an id in the custom widget editor in the HTML section, that can be identified by our JS code to render the chart into this HTML block."

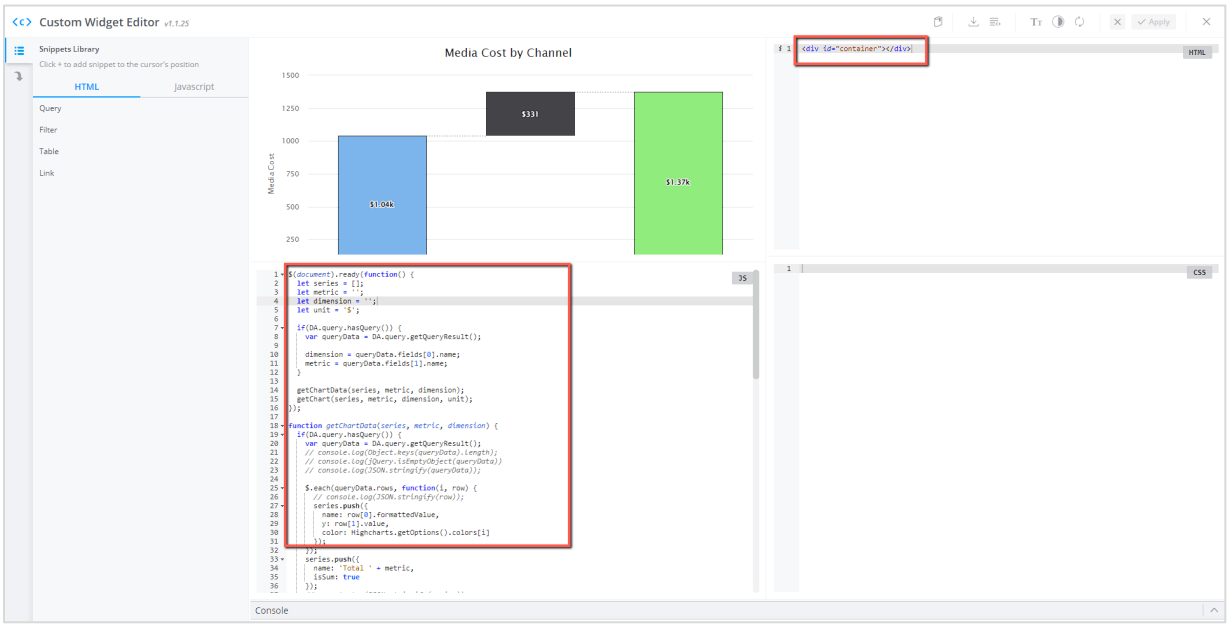
   ```
   <div id="container"></div>
   ```

3. We can fetch the underlying measurement and dimension data using the DTR snippet library – Query API and transform the data into the format required by Highcharts to render our chart.

   ```
   DA.query.getQueryResult();
   ```

4. Once we have the data transformed, we are ready to use the Highcharts function to create our chart using this data. We can write our JS code in the editor JS section.

   ```
   Highcharts.chart('container', {
   <Options in JSON format which needs to be passed to customize our chart>
   })
   ```



We can refer to this link for more details on Highcharts options

https://www.highcharts.com/demo/waterfall

Complete JS code which highly depends on your use case

```
$(document).ready(function() {
  let series = [];
  let metric = '';
  let dimension = '';
  let unit = '$';

  if(DA.query.hasQuery()) {
    var queryData = DA.query.getQueryResult();

    dimension = queryData.fields[0].name;
    metric = queryData.fields[1].name;
  }

  getChartData(series, metric, dimension);
  getChart(series, metric, dimension, unit);
});

function getChartData(series, metric, dimension) {
  if(DA.query.hasQuery()) {
    var queryData = DA.query.getQueryResult();
    $.each(queryData.rows, function(i, row) {
      series.push({
        name: row[0].formattedValue,
        y: row[1].value,
        color: Highcharts.getOptions().colors[i]
      });
    });
    series.push({
      name: 'Total ' + metric,
      isSum: true
    });
  }
}

function getChart(series, metric, dimension, unit) {
  Highcharts.chart('container', {
    chart: {
      type: 'waterfall'
    },
    accessibility: {
      enabled: false
    },
    title: {
      text: metric + ' by ' + dimension
    },
    credits: {
      enabled: false
    },
    xAxis: {
      type: 'category'
    },
    yAxis: {
      title: {
        text: metric
      }
    },
    legend: {
      enabled: false
    },

    tooltip: {
      pointFormatter: function() {
        if(this.y < 1000)
```

```
          return unit + parseFloat(Highcharts.numberFormat(this.y, 2, '.', ','));
        else
          return unit + parseFloat(Highcharts.numberFormat(this.y / 1000, 2, '.', ',')) + 'k';
      }
    },

    plotOptions: {
      series: {
        colorByPoint: true
      }
    },

    series: [{
      upColor: Highcharts.getOptions().colors[2],
      color: Highcharts.getOptions().colors[3],
      pointPadding: 0,
      data: series,
      dataLabels: {
        enabled: true,
        formatter: function () {
          if(this.y < 1000)
            return unit + parseFloat(Highcharts.numberFormat(this.y, 2, '.', ','));
          else
            return unit + parseFloat(Highcharts.numberFormat(this.y / 1000, 2, '.', ',')) + 'k';
        },
        style: {
          fontWeight: 'bold'
        }
      }
    }]
  });
}
```