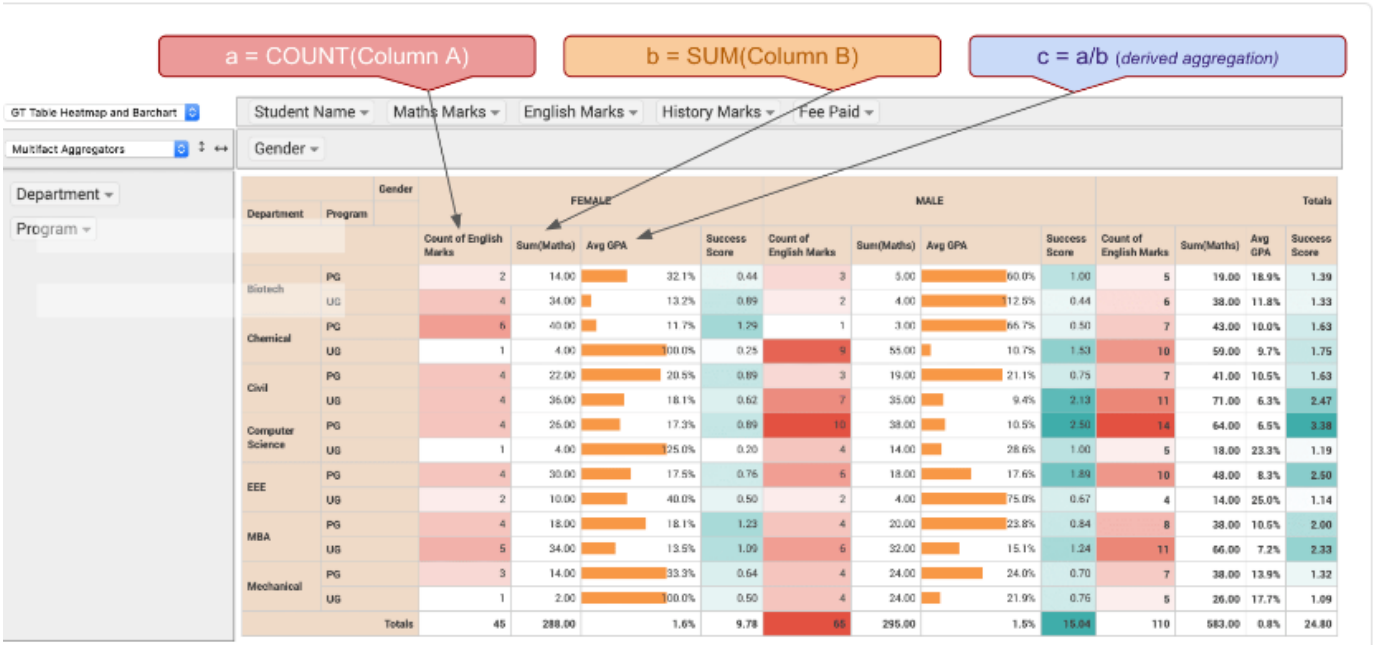
Multifact

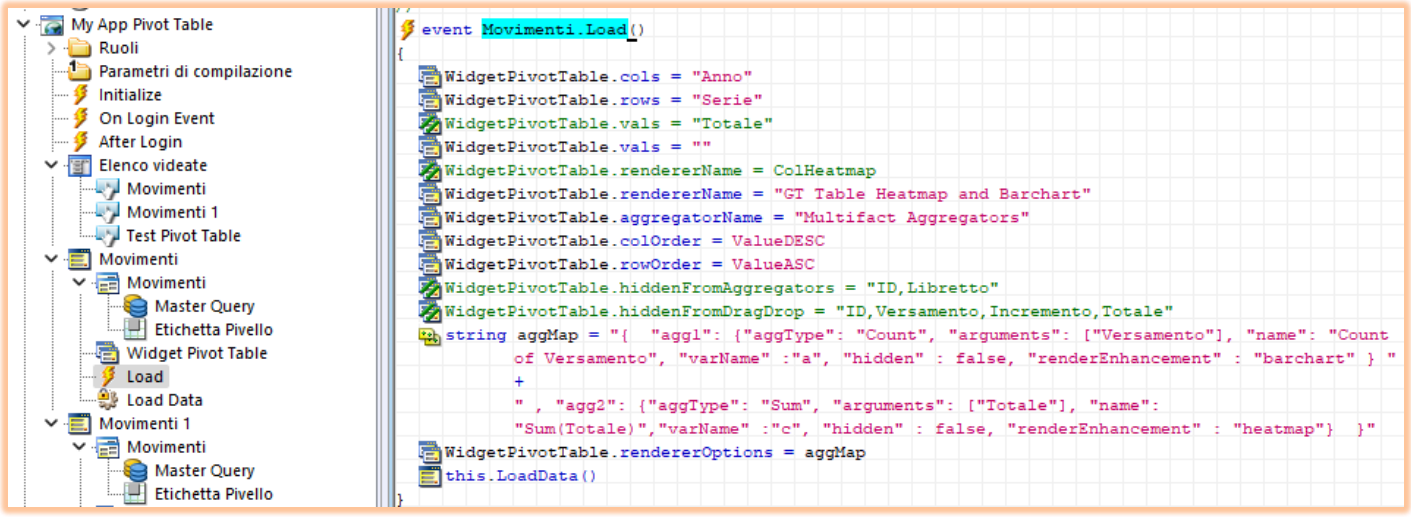
Ho preso le informazioni da:
http://pranjalgoswami.in/multifact-pivottable/examples/1_multiple-aggregations.html

It renders multiple aggregations simultaneously along with dynamic aggregations based on expressions



Le modifiche da effettuare sull'esempio del webinar di Giannelli

Nella load della form ho aggiunto la proprietà di configurazione aggMap come nell'esempio di pranjal-goswami:



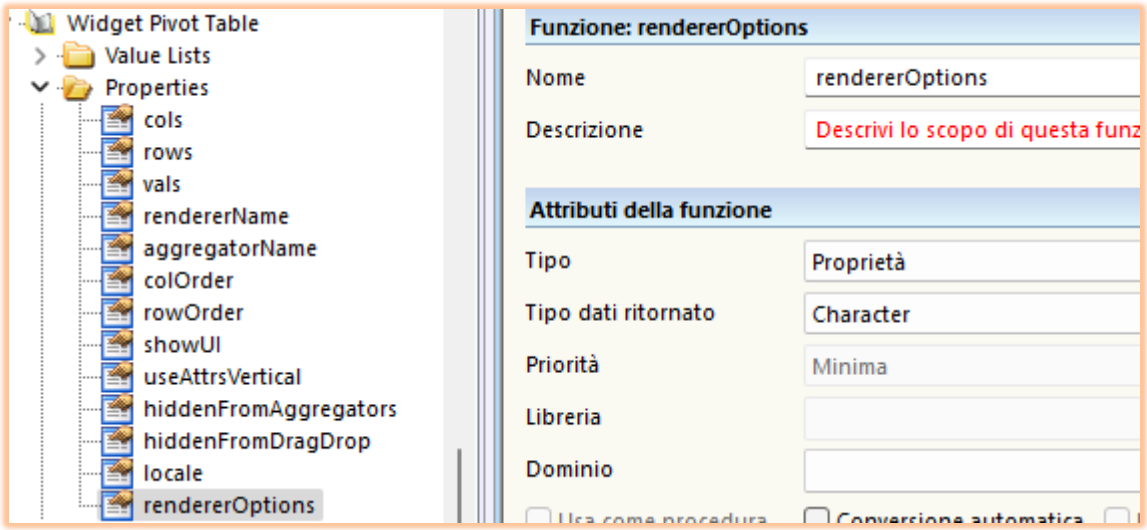
In formato testo:

Event Mdm.Load(){
WidgetPivotTable.cols = "Libretto"
WidgetPivotTable.rows = "Anno"
WidgetPivotTable.vals = ""
WidgetPivotTable.rendererName = "GT Table Heatmap and Barchart"
WidgetPivotTable.aggregatorName = "Multifact Aggregators" (con le maisucole altrimenti va in crash)
WidgetPivotTable.colOrder = ValueDESC

```
WidgetPivotTable.rowOrder = ValueASC
string aggMap = "{  "agg1": {"aggType": "Count", "arguments": ["Versamento"], "name": "Count
      of Versamento", "varName" :"a", "hidden" : false, "renderEnhancement" : "heatmap" } "
      +
      " , "agg2": {"aggType": "Sum", "arguments": ["Totale"], "name":
      "Sum(Totale)","varName" :"c", "hidden" : false, "renderEnhancement" : "heatmap"}  }"
WidgetPivotTable.rendererOptions = aggMap
this.LoadData()}
```

Ho perso ore per capire perché non funzionasse, esaminando il debug, ho scoperto di aver scritto: "Multifact aggregators" l'iniziale di aggregators in minuscolo.

Ho aggiunto la proprietà rendererOptions a cui assegno aggMap:



Per quel che riguarda **Inde** non ho dovuto aggiungere altro rispetto all'esempio del webinar.

Però per semplificare aggMap nel mio esempio è fisso. Ma si può prevedere una videata in cui si possono operare delle scelte e una procedura che crei la stringa JSON.

Occorre aggiungere al filelist.txt e alla cartella "…JSComponents\PivotTable" **multifact-pivottable.js** scaricabile qui https://github.com/pranjal-goswami/multifact-pivottable

Modifica di WidgetPivotTable.js

```
var calJS = [
    "JSComponents/PivotTable/jquery.min.js",
    "JSComponents/PivotTable/jquery.ui.touch-punch.min.js",
    "JSComponents/PivotTable/jquery-ui.min.js",
    "JSComponents/PivotTable/pivot.js",
    "JSComponents/PivotTable/plotly-basic-latest-min.js",
    "JSComponents/PivotTable/plotly_renderers.min.js",
    "JSComponents/PivotTable/export_renderers.min.js",
    "JSComponents/PivotTable/multifact-pivottable.js"];
```

Ho aggiunto l'ultima riga.

Ho definito la nuova proprietà rendererOptions:

```
// rendererOptions
Object.defineProperty(WidgetPivotTable.prototype, "rendererOptions", {
  get: function () {
    return this._rendererOptions;
  },
  set: function (value) {
    this._rendererOptions = JSON.parse(value);
    this.updateTable();
  }
});
```

ho aggiunto rendererOptions (che deriva da aggMap) come nell'esempio dell'autore:

```
let config = {
    renderers: renderers,
    cols: this.arrayToLabel(this.cols),
    rows: this.arrayToLabel(this.rows),
    vals: this.arrayToLabel(this.vals),
    rowOrder: this.rowOrder,
    colOrder: this.colOrder,
    showUI: this.showUI,
    rendererName: rn,
    rendererOpts : this.rendererOptions,
    rendererOptions : {
      aggregations : {
          defaultAggregations : this.rendererOptions,
      }
    },
```

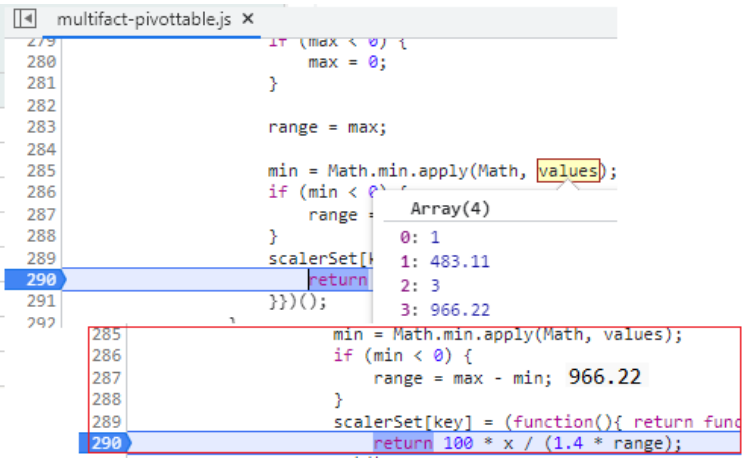
concludo come nell'esempio dell'autore:

```
  var customAggs = {};
    customAggs['Multifact Aggregators'] =
$.pivotUtilities.multifactAggregatorGenerator(config.rendererOpts,[]);

  $.pivotUtilities.customAggs = customAggs;

  config.aggregators = $.extend($.pivotUtilities.aggregators,
$.pivotUtilities.customAggs);

  config.renderers = $.extend($.pivotUtilities.renderers, $.pivotUtilities.gtRenderers);


  let id = this.pivotDiv.id;
  id = id.replace(/=/g, "\\=");
  $("#" + id).pivotUI(this.pivotData, config, true, this.locale);
};
```

Aggiungo alcune cose utili

Se seguite il programma allegato in debug sul js

Serie	Anno	2018		2019	
		Count of Versamento	Sum(Totale)	Count of Versamento	Sum(Totale)
12		1	483.11	3	966.22
13		2	43.30		
14		1	907.78		
15		1	61.49	3	122.98
16		10	12,343.57	10	3,987.58
17		89	85,177.13	52	39,184.11
18		184	88,187.66	104	26,166.00
Totals		288	187,204.04	172	70,426.89



fa un calcolo molto empirico:
1)determina il minimo e il massimo di capre e cavoli e ricava un range
2)calcola la % della colonna di sinistra rispetto al range moltiplicato per 1,4 e lo usa usa come larghezza della barra.
Si capisce bene il funzionamento seguendolo in debug nel punto di interruzione che ho usato io.
Al punto di interruzione controllate *values* e poi calcolate la formula scalarSet.

Nella prima riga ad esempio, nel caso di Count, nella prima colonna la %, con quel sistema, vale 0,07.
Nella terza colonna viene fuori una % pari a 0,22.
Stranamente il rapporto è giusto ¼ contro ¾. Però graficamente non è apprezzabile con valori così piccoli.
Se il calcolo venisse effettuato correttamente i valori sarebbero 25% e 75%.
Non so se usarlo così o “correggerlo”.