Transforming JSON to CSV with the help of Flatten task in Azure Data Factory

datanrg.blogspot.com/2020/03/transforming-json-to-csv-with-help-of.html

(2020-Mar-19) Recently, Microsoft introduced a new Flatten task to the existing set of powerful transformations available in the Azure Data Factory (ADF) Mapping Data Flows - https://docs.microsoft.com/en-us/azure/data-factory/data-flow-flatten.

What this new task does it helps to transform/transpose/flatten your JSON structure into a denormalized flatten datasets that you can upload into a new or existing flat database table.

2020-Mar-26 Update:
Part 2: Transforming JSON to CSV with the help of Flatten task in Azure Data Factory - Part 2 (Wrangling data flows)

I like the analogy of the Transpose function in Excel that helps to rotate your vertical set of data pairs (**name : value**) into a table with the column **name**s and **value**s for corresponding objects. And when this vertical JSON structural set contains several similar sets (array) then ADF Mapping Data Flows Flatten does a really good job by transforming it into a table with several rows (records).

Let's use this JSON data file as an example

```
{
 "id": "0001",
 "type": "donut",
 "name": "Cake",
 "ppu": 0.55,
 "batters":
  {
   "batter":
    [
     { "id": "1001", "type": "Regular" },
     { "id": "1002", "type": "Chocolate" },
     { "id": "1003", "type": "Blueberry" },
     { "id": "1004", "type": "Devil's Food" }
    ]
  },
 "topping":
  [
   { "id": "5001", "type": "None" },
   { "id": "5002", "type": "Glazed" },
   { "id": "5005", "type": "Sugar" },
   { "id": "5007", "type": "Powdered Sugar" },
   { "id": "5006", "type": "Chocolate with Sprinkles" },
   { "id": "5003", "type": "Chocolate" },
   { "id": "5004", "type": "Maple" }
  ]
}
```

and create a simple ADF mapping data flow to Flatten this JSON file into a CSV sink dataset.

On a high level my data flow will have 4 components:
1) Source connection to my JSON data file
2) Flatten transformation to transpose my Cake to Toppings
3) Further Flattent transformation to transpose my Cake > Toppings to Batters
4) Sink output Flatten result in a CSV file

(1) Source connection to my JSON data file

Connection to my JSON file is simple, however, it's interesting to see how the output of my consumed JSON file is shown in the Data Preview tab, which shows one row with several array objects.

(2) Flatten transformation to transpose my Cake to Toppings

My next Flatten transformation task transposes the JSON Topping array with 2 objects (id, type) into 7 flatten rows, where JSON Batter objects are now visible as individual arrays.

(3) Further Flattent transformation to transpose my Cake > Toppings to Batters

All 7 records that came out from the previous Flatten transformation task can now be

used as input for my further Flatten transformation. This helps to convert (unroll) 2 additional fields from the Batter JSON subset (id, type).

(4) Sink output Flatten result in a CSV file

Data Preview option in my Sink doesn't get changed from its sibling in the previous task, so I thought to challenge the Data Factory and replaced my initial JSON file that contained one object with another file that would contain several similar objects.


```
[
 {
  "id": "0001",
  "type": "donut",
  "name": "Cake",
  "ppu": 0.55,
  "batters":
   {
    "batter":
     [
      { "id": "1001", "type": "Regular" },
      { "id": "1002", "type": "Chocolate" },
      { "id": "1003", "type": "Blueberry" },
      { "id": "1004", "type": "Devil's Food" }
     ]
   },
  "topping":
   [
    { "id": "5001", "type": "None" },
    { "id": "5002", "type": "Glazed" },
    { "id": "5005", "type": "Sugar" },
    { "id": "5007", "type": "Powdered Sugar" },
    { "id": "5006", "type": "Chocolate with Sprinkles" },
    { "id": "5003", "type": "Chocolate" },
    { "id": "5004", "type": "Maple" }
   ]
 },
 {
  "id": "0002",
  "type": "donut",
  "name": "Raised",
  "ppu": 0.55,
  "batters":
   {
    "batter":
     [
      { "id": "1001", "type": "Regular" }
     ]
   },
  "topping":
   [
    { "id": "5001", "type": "None" },
    { "id": "5002", "type": "Glazed" },
    { "id": "5005", "type": "Sugar" },
    { "id": "5003", "type": "Chocolate" },
```

```
    { "id": "5004", "type": "Maple" }
   ]
  },
  {
   "id": "0003",
   "type": "donut",
   "name": "Old Fashioned",
   "ppu": 0.55,
   "batters":
    {
     "batter":
      [
       { "id": "1001", "type": "Regular" },
       { "id": "1002", "type": "Chocolate" }
      ]
    },
   "topping":
    [
     { "id": "5001", "type": "None" },
     { "id": "5002", "type": "Glazed" },
     { "id": "5003", "type": "Chocolate" },
     { "id": "5004", "type": "Maple" }
    ]
  }
]
```

Source file Data Preview correctly showed me 3 rows. All next Flatten transformation tasks' outputs were tripled in their results, and my final Output file contained all 41 expected records!

Well done, Microsft team!

I really like this visual way to transform (flatten) a sourcing JSON stream into a CSV file.