

Suemy Inagaki

# Data Engineer Challenge

## Apache Beam

### What is Apache Beam

Apache Beam is open source and is a unified model for batch and streaming data processing pipelines that simplifies large-scale data processing. It provides powerful abstraction, flexibility to execute the same code for both batch and streaming data pipelines, Cross-language Capabilities, freedom to choose between various execution engines and also other benefits.

### Why use Apache Beam

The reason to use Apache Beam is related to the many benefits it provides, as stated above. It allows you to build robust pipelines with data reliability, you can unify the code for batch and streaming data, flexibility to change the runner using the same code, and supports the use in pipelines that work with an intense volume of data. Furthermore, It runs the code through a distributed computing framework and allows efficient parallelization of extensive processes without knowing how Apache Beam handles data between the cluster.

## Challenge

Create a data pipeline, using Apache Beam, that reads the two input files (EstadosIBGE.csv and Vendas\_por\_dia.csv) and as a result generates the following two files:

### 1st file

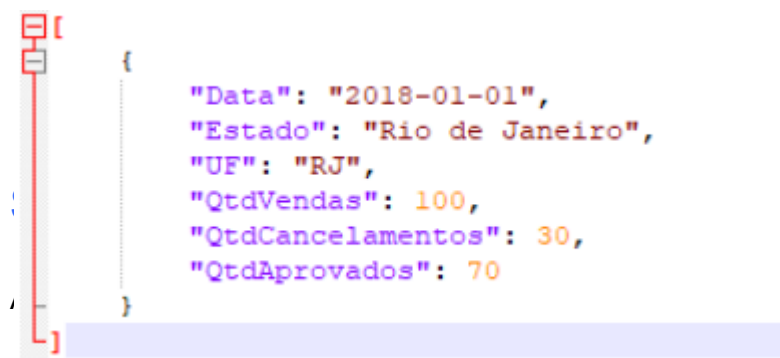
Aggregate information by state.

Format: CSV

Information: Data, Estado, UF, QtdVendas, QtdCancelamentos, QtdAprovados

### 2nd file

Based on the same results generated in the 1st file, generate valid a .json file where each column of the previous file is a key within that JSON



```
[{"Data": "2018-01-01", "Estado": "Rio de Janeiro", "UF": "RJ", "QtdVendas": 100, "QtdCancelamentos": 30, "QtdAprovados": 70}]
```

## What is expected

Code in a public GitHub repository, with the script developed, the output files and a README.md explaining how the script was developed and how to run it.

Differentials will be considered:

- Use of good development practices
- Continuous versioning of code on GitHub
- Code and project documentation
- Explanations on why Apache Beam is used, demonstrating mastery over the framework.

## Solution

Some PCollections were created to help the development of the solution:

**ibgeData:** read and format IBGE file. The PCollection generated here has the state code and the UF as below.

```
('12', 'Acre')
('27', 'Alagoas')
('16', 'Amapá')
('13', 'Amazonas')
('29', 'Bahia')
...
```

**vendasData:** read and format Vendas file. The result is a PCollection grouped by state, where each primary key is the state code (to be able to merge with ibgeData). The value of each key is a list of elements that represent sales data for a given date and state.

```
('31', [['31', '41883', '', 'MG', 2, 0, 2], ['31', ... ]])
('35', [['35', '41884', '', 'SP', 5, 4, 1], ['35', ... ]])
('25', [['25', '41885', '', 'PB', 1, 1, 0], ['25', ... ]])
('27', [['27', '41901', '', 'AL', 1, 1, 0], ['27', ... ]])
('33', [['33', '41886', '', 'RJ', 8, 2, 6], ['33', '41887', '', 'RJ', 3,
1, 2], ['33', '41889', '', 'RJ', 4, ... ]])
...
```

**mergedData:** merge the IBGE and Vendas data to relate the UF with the state name. The resulting PCollection adds the state name to the list of elements mentioned above.

```
('12', (['Acre'], []))
('27', (['Alagoas'], [[['27', '41901', '', 'AL', 1, 1, 0], ['27',
'41907', '', 'AL', 1, ... ]]]))
('16', (['Amapá'], []))
...
```

It is now possible to manipulate the data to format both CSV and JSON.

**outputData:** is a PCollection with the formatted output data

```
[ '41901', 'Alagoas', 'AL', 1, 1, 0 ]  
[ '41907', 'Alagoas', 'AL', 1, 0, 1 ]  
[ '41895', 'Alagoas', 'AL', 2, 1, 1 ]  
...  
[ '41896', 'Sergipe', 'SE', 1, 1, 0 ]  
[ '41888', 'Sergipe', 'SE', 1, 0, 1 ]
```

**csvData:** format data and write to CSV file

**jsonData:** format data and write to JSON file

## How to run

First, you need to install the libraries required to run the program. To do this, run the code below in the terminal

```
pip install -r requirements.txt
```

Then you can run the code below to run the program.

```
python script.py
```

The final file will be named "output-HH-MM-SS-00000-of-00001" .csv and .json.

output-21-28-56-00000-of-00001.json

output-21-28-56-00000-of-00001.csv

You can see an example of output in the output directory.

## References

<https://medium.com/hurb-engineering/streaming-pipelines-using-dataflow-and-apache-beam-3000aab1c04d>

<https://medium.com/hurb-engineering/complex-tasks-orchestration-at-hurb-with-apache-airflow-dcb423c4dee6>

<https://medium.com/hurb-engineering/data-platform-architecture-at-hurb-com-8c472c051fa2>

<https://beam.apache.org/about/#about-apache-beam-project>