Cheatography

FluxLang 101 Cheat Sheet

by rictomm via cheatography.com/165543/cs/34657/

A Table of Measure

Measurement	Sensor	Temperature	Timestamp
iot-oven	S1	290	1633953600000000000
iot-oven	S2	105	1633953615000000000
iot-oven	S1	305	1633953660000000000
iot-oven	S2	120	1633953675000000000

Basic Filtering

```
from(bucket: "training")
    |> range( start: v.time Ran geS tart, stop:
v.time Ran geStop)
    |> filter(fn: (r) => r._mea sur ement == " -
iot -ov en")
    |> filter(fn: (r) => r._field == " tem per -
atu re")
    |> filter(fn: (r) => r.sensor == " S2")
    |> filter(fn: (r) => r._value < 100)
```

The **range** clause allows to filter by time creating a time windodw
The **filter** clause reduces the ammount of records and can be applied on the measurements, fields, tags, and field/tag keys.

Functions

Flux transformations take a stream of tables as input, transform the data in some way, and output a stream of tables.

Aggregation

```
from(bucket: "training")
    |> range( start: v.time Ran geS tart, stop:
v.time Ran geStop)
    |> filter(fn: (r) => r._mea sur ement == " -
iot -ov en")
    |> group( col umns: ["_f iel d"])
    |> mean()
```

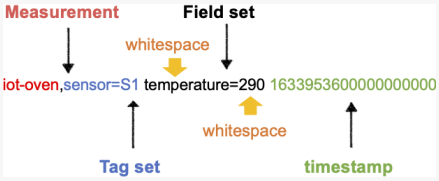
Flux aggregate functions are transformations aggregate values from input tables in some way.

Selector

```
from(bucket: "training")
    |> range( start: v.time Ran geS tart, stop:
v.time Ran geStop)
    |> filter(fn: (r) => r._mea sur ement == " -
iot -ov en")
    |> filter(fn: (r) => r.sensor == " S1")
    |> group( col umns: ["_f iel d"])
    |> last()
```

Flux selector functions are transformations that return one or more record per input table.

Line Protocol



Advanced Windowing

```
from(bucket: "training")
    |> range( start: v.time Ran geS tart, stop:
v.time Ran geStop)
    |> filter(fn: (r) => r._mea sur ement == " -
iot -ov en")
    |> filter(fn: (r) => r._field == " tem per -
atu re")
    |> filter(fn: (r) => r.sensor == " S1")
    |> aggreg ate Win dow (every: 2m, fn: mean)
```

aggregateWindow() downsamples data by grouping data into fixed windows of time and applying an aggregate or selector function to each window.

NOTE The flag `create Empty: false`can be used to consider only the windows that contains data (its default value is `true`)

Map and Custom Functions

```
from(bucket: "training") ...
                |> map(fn: (r) => ({ r with
correc tValue: r._value - 5.0 }))
```

Note the `r with` clause maintains alla the original columns and adds the new one.

Custom pipe forwardable function

```
adjValues = (tables=<-, x) =>
    tables
        |> map(fn: (r) => ({ r with correc tValue:
r._value + x}))
from(b ucket: " tra ini ng")
    |> range( start: v.time Ran geS tart, stop:
v.time Ran geStop)
    |> filter(fn: (r) => r._mea sur ement == " -
iot -ov en")
    |> adjVal ues (x: -5.0)
```

Most Flux functions manipulate data piped-forward into the function. In order for a custom function to process piped-forward data, one of the function parameters must capture the input tables using the `<-` pipe-receive expression.

Joins

Conditional Expressions

```
... |> map(fn: (r) => ({
           r with _value:
                                    if r._value
== true then 1
                                    else 0 }))
```

Conditional expressions evaluate a boolean-valued condition. If the result is true, the expression that follows the then keyword is evaluated and returned.

```
from(bucket: "training")
...
join(t ables: {s1: hs1, s2: hs2}, on: ["_t -
ime "], method: " inn er")
```

The join() function merges two or more input streams, whose values are equal on a set of common columns, into a single output stream.

By **rictomm**
cheatography.com/rictomm/

Published 13th October, 2022.
Last updated 13th October, 2022.