# Flux@Cisco Training - Day 1

## Authors

```
Marco Balduini – marco.balduini@quantiaconsulting.com
Emanuele Della Valle – emanuele.dellavalle@quantiaconsulting.com
```

## Use Case Description - Linear Pizza Oven

We have a linear oven to continuously cook pizza.

The cooking operation has two main steps:

- the cooking of the pizza base, and
- the mozzarella melting area.

There are two sensors:

- S1 measures the temperature and the relative humidity of the pizza base cooking area.
- S2 measures the temperature and the relative humidity of the mozzarella melting area.

Both sensors send a temperature measurement every minute, but are not synchronised.

## Simple Data Modelling

doc

**Note**: At this time we consider only the temperature measurements from the sensors

**1 - Model temperature and humidity data from two sensors**

| measurement | sensor | temperature | humidity | ts |
|---|---|---|---|---|
| iot-oven | S1 | 290 | 30 | 1595332800000000000 |
| iot-oven | S2 | 105 | 55 | 1595332815000000000 |
| iot-oven | S1 | 305 | 38 | 1595332860000000000 |
| iot-oven | S2 | 120 | 65 | 1595332875000000000 |

Solution

```
iot-oven,sensor=S1 temperature=290,humidity=30 1595332800000000000
iot-oven,sensor=S2 temperature=105,humidity=55 1595332815000000000
iot-oven,sensor=S1 temperature=305,humidity=38 1595332860000000000
iot-oven,sensor=S2 temperature=120,humidity=65 1595332875000000000
```

Complete set of data to be loaded into the training bucket

```
iot-oven,sensor=S1 temperature=290,humidity=30 1595332800000000000
iot-oven,sensor=S2 temperature=105,humidity=55 1595332815000000000
iot-oven,sensor=S2 temperature=110,humidity=60 1595332845000000000
iot-oven,sensor=S1 temperature=305,humidity=38 1595332860000000000
iot-oven,sensor=S2 temperature=120,humidity=65 1595332875000000000
iot-oven,sensor=S2 temperature=115,humidity=60 1595332905000000000
iot-oven,sensor=S1 temperature=280,humidity=45 1595332920000000000
iot-oven,sensor=S2 temperature=110,humidity=67 1595332935000000000
iot-oven,sensor=S2 temperature=115,humidity=72 1595332965000000000
iot-oven,sensor=S1 temperature=280,humidity=22 1595332980000000000
iot-oven,sensor=S2 temperature=95,humidity=65 1595332995000000000
iot-oven,sensor=S2 temperature=90,humidity=60  1595333025000000000
iot-oven,sensor=S1 temperature=285,humidity=32 1595333040000000000
iot-oven,sensor=S2 temperature=100,humidity=55 1595333055000000000
iot-oven,sensor=S2 temperature=105,humidity=60 1595333085000000000
```

**Note**

- Start Time: 2020-07-21 12:00:00 GMT (2020-06-07T12:00:00Z)
- End Time: 2020-07-21 12:05:00 GMT (2020-06-07T12:05:00Z)

# First Query (DEMO)

```
from(bucket: "training")
  |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
  |> filter(fn: (r) => r._measurement == "iot-oven")
  |> filter(fn: (r) => r._field == "temperature")
  |> filter(fn: (r) => r.sensor == "S2")
  |> filter(fn: (r) => r._value < 100)
```

# 02 - Range and Tables (DEMO)

doc

**Extract all the measurements in a given range**

**Absolute: range(start: 2019-10-01T00:00:00Z, stop: 2019-10-01T00:10:00Z)**

```
from(bucket: "training")
  |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
```

**Relative: range(start: -36h)**

```
from(bucket: "training")
  |> range(start: -36h)
```

## 03 - Filter By Tag

**Extract the temperature data from the cooking base area (sensor S1)**

doc

```
from(bucket: "training")
  |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
  |> filter(fn: (r) => r._measurement == "iot-oven")
  |> filter(fn: (r) => r._field == "temperature")
  |> filter(fn: (r) => r.sensor == "S1")
```

## 04 - Filter By Value

Extract the measurements from the cooking base area (sensor S1) with a temperature under 300°

doc

```
from(bucket: "training")
  |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
  |> filter(fn: (r) => r._measurement == "iot-oven")
  |> filter(fn: (r) => r._field == "temperature")
  |> filter(fn: (r) => r.sensor == "S1")
  |> filter(fn: (r) => r._value < 300)
```

## 05 - Functions

**Note**

- Start Time: 2020-07-21 12:00:00 GMT (2020-06-07T12:00:00Z)
- End Time: 2020-07-21 12:05:00 GMT (2020-06-07T12:05:00Z)

Grouping + Aggregator (mean) - Extract the average temperature and the average humidity along the different stages of the linear pizza oven

doc

```
temp = from(bucket: "training")
  |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
  |> filter(fn: (r) => r._measurement == "iot-oven")
  |> filter(fn: (r) => r._field == "temperature")
```

```
    |> group()
    |> mean()

  hum = from(bucket: "training")
    |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
    |> filter(fn: (r) => r._measurement == "iot-oven")
    |> filter(fn: (r) => r._field == "humidity")
    |> group()
    |> mean()
```

Selector (last) - Extract the last humidity measurements from the cooking base area

[doc](#)

```
  from(bucket: "training")
    |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
    |> filter(fn: (r) => r._measurement == "iot-oven")
    |> filter(fn: (r) => r._field == "humidity")
    |> filter(fn: (r) => r.sensor == "S1")
    |> last()
```

## 06 - aggregateWindow

Extract the moving average temperature observed in the cooking base area over a window of 2 minutes (DEMO)

[doc](#)

```
  from(bucket: "training")
    |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
    |> filter(fn: (r) => r._measurement == "iot-oven")
    |> filter(fn: (r) => r._field == "temperature")
    |> filter(fn: (r) => r.sensor == "S1")
    |> aggregateWindow(every: 2m, fn: mean)
```

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

```
  from(bucket: "training")
    |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
    |> filter(fn: (r) => r._measurement == "iot-oven")
    |> filter(fn: (r) => r._field == "temperature")
    |> filter(fn: (r) => r.sensor == "S1")
    |> aggregateWindow(every: 2m, fn: mean, createEmpty: false)
```

Extract the moving average temperature observed by S2 over a window of 3 minutes (hands-on)

```
from(bucket: "training")
  |> range(start: 2020-07-21T12:00:00Z, stop: 2020-07-21T12:05:00Z)
  |> filter(fn: (r) => r._measurement == "iot-oven")
  |> filter(fn: (r) => r._field == "temperature")
  |> filter(fn: (r) => r.sensor == "S2")
  |> aggregateWindow(every: 3m, fn: mean, createEmpty: false)
```

## BC1 - City Water Tank dashboard queries (part 1)

Needed input files:

- fill-level-lp.txt
- flow-rate-lp.txt
- pump-speed-lp.txt

The fill level of the city water tank

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "fill_level")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.tank == "city_water")
```

The speed of the pump that refills the city water tank

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "pump_speed")
  |> filter(fn: (r) => r._field == "value")
```

The fill level of each tank down sampled to 1 min

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "fill_level")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.tank == "A1")
  |> aggregateWindow(every: 1m, fn: mean, createEmpty: false)
```

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
```

```
|> filter(fn: (r) => r._measurement == "fill_level")
|> filter(fn: (r) => r._field == "value")
|> filter(fn: (r) => r.tank == "A2")
|> aggregateWindow(every: 1m, fn: mean, createEmpty: false)
```

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "fill_level")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.tank == "B1")
  |> aggregateWindow(every: 1m, fn: mean, createEmpty: false)
```

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "fill_level")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.tank == "B2")
  |> aggregateWindow(every: 1m, fn: mean, createEmpty: false)
```

The flow rate of each valve

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "flow_rate")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.valve == "V1")
  |> last()
```

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "flow_rate")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.valve == "V2")
  |> last()
```

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "flow_rate")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.valve == "V3")
  |> last()
```

```
from(bucket: "training")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "flow_rate")
  |> filter(fn: (r) => r._field == "value")
  |> filter(fn: (r) => r.valve == "V4")
  |> last()
```