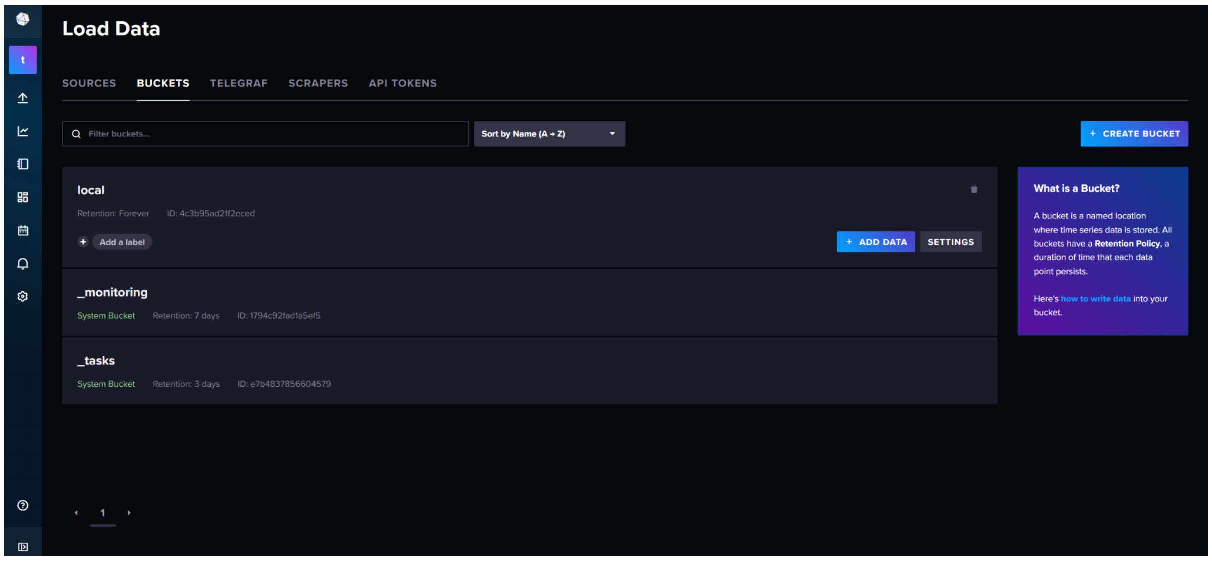
# TSDB- INFO5A - P1G2 groupe A
Mamisoa RANDRIANARIMANANA - Xianxiang ZHANG

## Q1. Open the *InfluxDB 2.0 OSS Metrics* dashboard. How many buckets are defined in the system.


Load Data

SOURCES BUCKETS TELEGRAF SCRAPERS API TOKENS

Filter buckets... | Sort by Name (A → Z) | + CREATE BUCKET

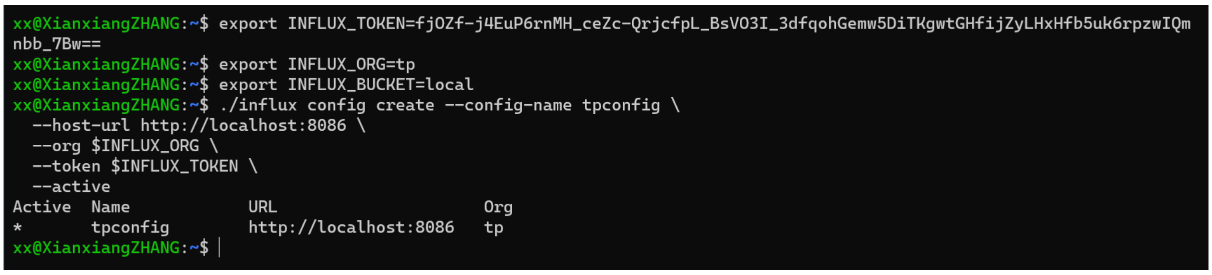
local
Retention: Forever ID: 4c3b95ad21f2eced
+ Add a label | + ADD DATA | SETTINGS

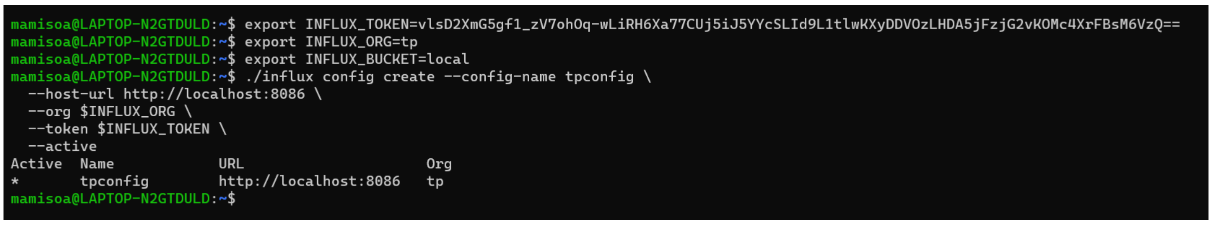
_monitoring
System Bucket Retention: 7 days ID: 1794c92fad1a5ef5

_tasks
System Bucket Retention: 3 days ID: e7b4837856604579

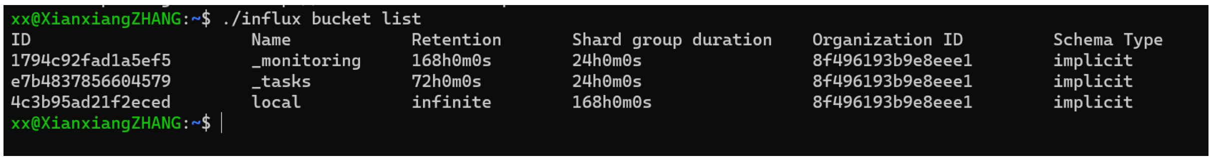
What is a Bucket?
A bucket is a named location where time series data is stored. All buckets have a **Retention Policy**, a duration of time that each data point persists.
Here's how to write data into your bucket.

3 buckets.


```
xx@XianxiangZHANG:~$ export INFLUX_TOKEN=fjOZf-j4EuP6rnMH_ceZc-QrjcfpL_BsVO3I_3dfqohGemw5DiTKgwtGHfijZyLHxHfb5uk6rpzwIQm
nbb_7Bw==
xx@XianxiangZHANG:~$ export INFLUX_ORG=tp
xx@XianxiangZHANG:~$ export INFLUX_BUCKET=local
xx@XianxiangZHANG:~$ ./influx config create --config-name tpconfig \
  --host-url http://localhost:8086 \
  --org $INFLUX_ORG \
  --token $INFLUX_TOKEN \
  --active
Active  Name            URL                     Org
*       tpconfig        http://localhost:8086   tp
xx@XianxiangZHANG:~$ |
```

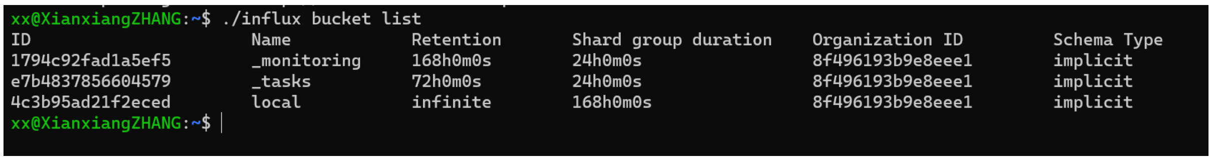

```
mamisoa@LAPTOP-N2GTDULD:~$ export INFLUX_TOKEN=vlsD2XmG5gf1_zV7ohOq-wLiRH6Xa77CUj5iJ5YYcSLId9L1tlwKXyDDVOzLHDA5jFzjG2vKOMc4XrFBsM6VzQ==
mamisoa@LAPTOP-N2GTDULD:~$ export INFLUX_ORG=tp
mamisoa@LAPTOP-N2GTDULD:~$ export INFLUX_BUCKET=local
mamisoa@LAPTOP-N2GTDULD:~$ ./influx config create --config-name tpconfig \
  --host-url http://localhost:8086 \
  --org $INFLUX_ORG \
  --token $INFLUX_TOKEN \
  --active
Active  Name            URL                     Org
*       tpconfig        http://localhost:8086   tp
mamisoa@LAPTOP-N2GTDULD:~$
```

## Q2. What is the influx command that allows you to list the available buckets?


```
xx@XianxiangZHANG:~$ ./influx bucket list
ID                      Name            Retention       Shard group duration    Organization ID         Schema Type
1794c92fad1a5ef5        _monitoring     168h0m0s        24h0m0s                 8f496193b9e8eee1        implicit
e7b4837856604579        _tasks          72h0m0s         24h0m0s                 8f496193b9e8eee1        implicit
4c3b95ad21f2eced        local           infinite        168h0m0s                8f496193b9e8eee1        implicit
xx@XianxiangZHANG:~$ |
```

./influx bucket list

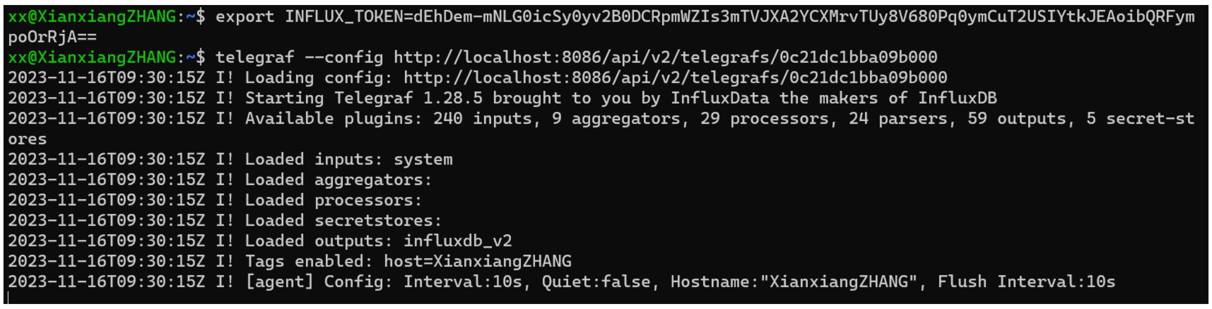
**Q3.** What are the names of the available buckets?



```
xx@XianxiangZHANG:~$ ./influx bucket list
ID                      Name            Retention       Shard group duration    Organization ID         Schema Type
1794c92fad1a5ef5        _monitoring     168h0m0s        24h0m0s                 8f496193b9e8eee1        implicit
e7b4837856604579        _tasks          72h0m0s         24h0m0s                 8f496193b9e8eee1        implicit
4c3b95ad21f2eced        local           infinite        168h0m0s                8f496193b9e8eee1        implicit
xx@XianxiangZHANG:~$ |
```

_monitoring,   _tasks and  local

## Monitor system resources



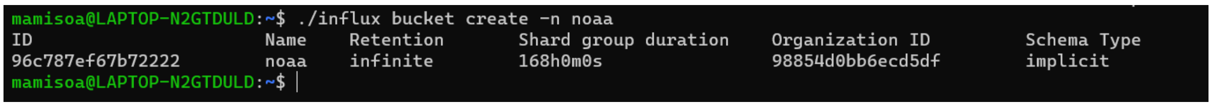
```
xx@XianxiangZHANG:~$ export INFLUX_TOKEN=dEhDem-mNLG0icSy0yv2B0DCRpmWZIs3mTVJXA2YCXMrvTUy8V680Pq0ymCuT2USIYtkJEAoibQRFym
poOrRjA==
xx@XianxiangZHANG:~$ telegraf --config http://localhost:8086/api/v2/telegrafs/0c21dc1bba09b000
2023-11-16T09:30:15Z I! Loading config: http://localhost:8086/api/v2/telegrafs/0c21dc1bba09b000
2023-11-16T09:30:15Z I! Starting Telegraf 1.28.5 brought to you by InfluxData the makers of InfluxDB
2023-11-16T09:30:15Z I! Available plugins: 240 inputs, 9 aggregators, 29 processors, 24 parsers, 59 outputs, 5 secret-st
ores
2023-11-16T09:30:15Z I! Loaded inputs: system
2023-11-16T09:30:15Z I! Loaded aggregators:
2023-11-16T09:30:15Z I! Loaded processors:
2023-11-16T09:30:15Z I! Loaded secretstores:
2023-11-16T09:30:15Z I! Loaded outputs: influxdb_v2
2023-11-16T09:30:15Z I! Tags enabled: host=XianxiangZHANG
2023-11-16T09:30:15Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"XianxiangZHANG", Flush Interval:10s
```

Q.4

Using the command-line interface, create a new bucket named `noaa` and import the NOAA data into it (`influx` binary, `write` command). **Important** Do not forget to specify the precision (time unit) used in your file.

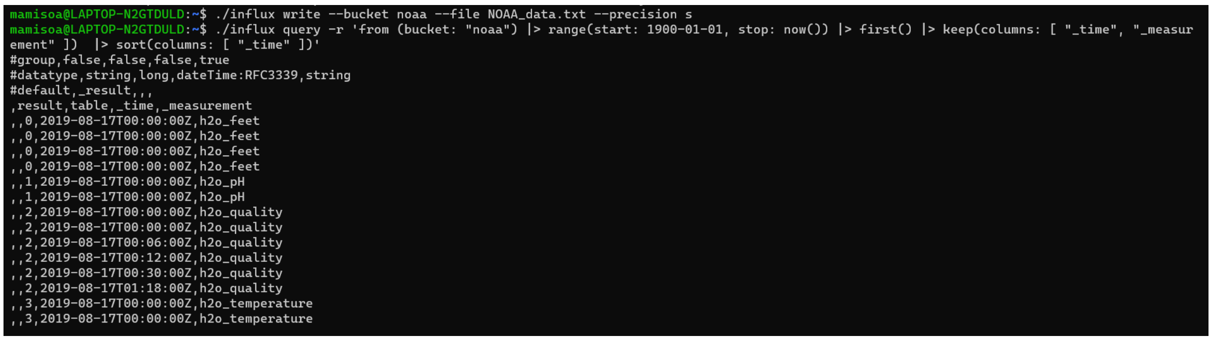
- **[Question]** Give the command-line used to import the data file.

https://www.influxdata.com/time-series-platform/telegraf/



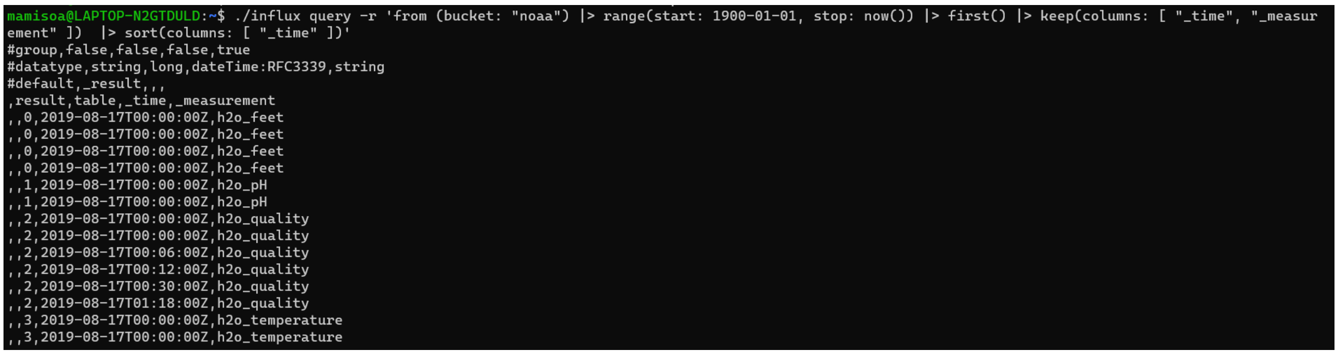
```
mamisoa@LAPTOP-N2GTDULD:~$ ./influx bucket create -n noaa
ID                      Name    Retention       Shard group duration    Organization ID         Schema Type
96c787ef67b72222        noaa    infinite        168h0m0s                98854d0bb6ecd5df        implicit
mamisoa@LAPTOP-N2GTDULD:~$ |
```

./influx write --bucket noaa --file NOAA_data.txt --precision s



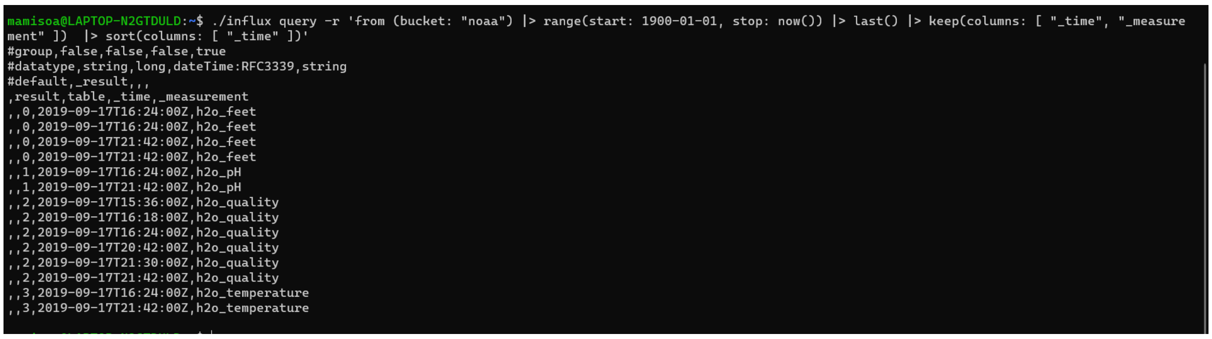
```
mamisoa@LAPTOP-N2GTDULD:~$ ./influx write --bucket noaa --file NOAA_data.txt --precision s
mamisoa@LAPTOP-N2GTDULD:~$ ./influx query -r 'from (bucket: "noaa") |> range(start: 1900-01-01, stop: now()) |> first() |> keep(columns: [ "_time", "_measur
ement" ])  |> sort(columns: [ "_time" ])'
#group,false,false,false,true
#datatype,string,long,dateTime:RFC3339,string
#default,_result,,,
,result,table,_time,_measurement
,,0,2019-08-17T00:00:00Z,h2o_feet
,,0,2019-08-17T00:00:00Z,h2o_feet
,,0,2019-08-17T00:00:00Z,h2o_feet
,,0,2019-08-17T00:00:00Z,h2o_feet
,,1,2019-08-17T00:00:00Z,h2o_pH
,,1,2019-08-17T00:00:00Z,h2o_pH
,,2,2019-08-17T00:00:00Z,h2o_quality
,,2,2019-08-17T00:00:00Z,h2o_quality
,,2,2019-08-17T00:06:00Z,h2o_quality
,,2,2019-08-17T00:12:00Z,h2o_quality
,,2,2019-08-17T00:30:00Z,h2o_quality
,,2,2019-08-17T01:18:00Z,h2o_quality
,,3,2019-08-17T00:00:00Z,h2o_temperature
,,3,2019-08-17T00:00:00Z,h2o_temperature
```

**[Question]** What is the beginning date of measurements? Note: you will need this information in the next questions



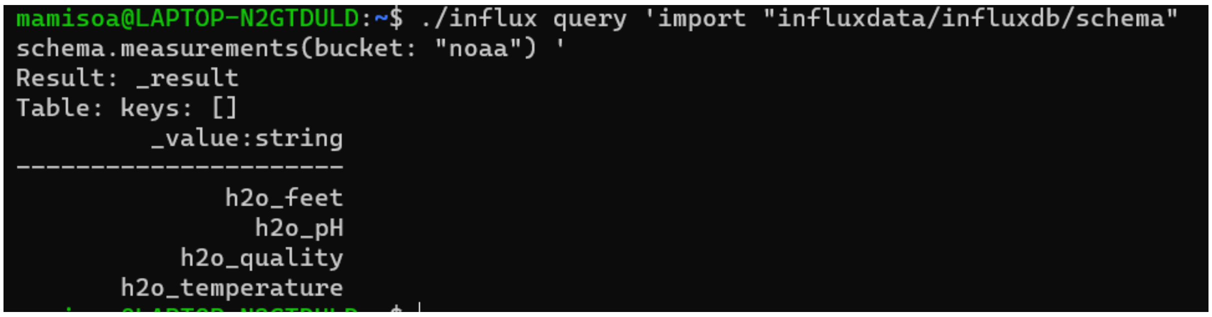
```
mamisoa@LAPTOP-N2GTDULD:~$ ./influx query -r 'from (bucket: "noaa") |> range(start: 1900-01-01, stop: now()) |> first() |> keep(columns: [ "_time", "_measur
ement" ])  |> sort(columns: [ "_time" ])'
#group,false,false,false,true
#datatype,string,long,dateTime:RFC3339,string
#default,_result,,,
,result,table,_time,_measurement
,,0,2019-08-17T00:00:00Z,h2o_feet
,,0,2019-08-17T00:00:00Z,h2o_feet
,,0,2019-08-17T00:00:00Z,h2o_feet
,,0,2019-08-17T00:00:00Z,h2o_feet
,,1,2019-08-17T00:00:00Z,h2o_pH
,,1,2019-08-17T00:00:00Z,h2o_pH
,,2,2019-08-17T00:00:00Z,h2o_quality
,,2,2019-08-17T00:00:00Z,h2o_quality
,,2,2019-08-17T00:06:00Z,h2o_quality
,,2,2019-08-17T00:12:00Z,h2o_quality
,,2,2019-08-17T00:30:00Z,h2o_quality
,,2,2019-08-17T01:18:00Z,h2o_quality
,,3,2019-08-17T00:00:00Z,h2o_temperature
,,3,2019-08-17T00:00:00Z,h2o_temperature
```

**[Question]** What is the new query, and what is the end date of measurements?

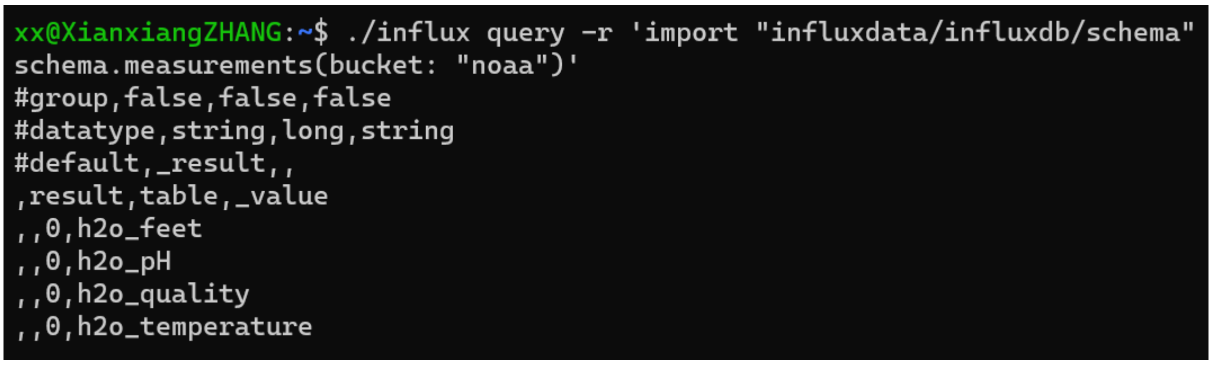


```
mamisoa@LAPTOP-N2GTDULD:~$ ./influx query -r 'from (bucket: "noaa") |> range(start: 1900-01-01, stop: now()) |> last() |> keep(columns: [ "_time", "_measure
ment" ])  |> sort(columns: [ "_time" ])'
#group,false,false,false,true
#datatype,string,long,dateTime:RFC3339,string
#default,_result,,,
,result,table,_time,_measurement
,,0,2019-09-17T16:24:00Z,h2o_feet
,,0,2019-09-17T16:24:00Z,h2o_feet
,,0,2019-09-17T21:42:00Z,h2o_feet
,,0,2019-09-17T21:42:00Z,h2o_feet
,,1,2019-09-17T16:24:00Z,h2o_pH
,,1,2019-09-17T21:42:00Z,h2o_pH
,,2,2019-09-17T15:36:00Z,h2o_quality
,,2,2019-09-17T16:18:00Z,h2o_quality
,,2,2019-09-17T16:24:00Z,h2o_quality
,,2,2019-09-17T20:42:00Z,h2o_quality
,,2,2019-09-17T21:30:00Z,h2o_quality
,,2,2019-09-17T21:42:00Z,h2o_quality
,,3,2019-09-17T16:24:00Z,h2o_temperature
,,3,2019-09-17T21:42:00Z,h2o_temperature
```

- **[Question]** What measurements are present in the *noaa* bucket? What is the query to list them?

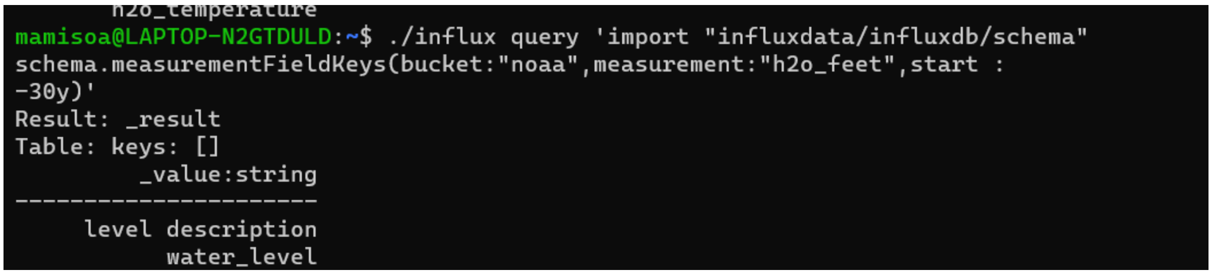


```
mamisoa@LAPTOP-N2GTDULD:~$ ./influx query 'import "influxdata/influxdb/schema"
schema.measurements(bucket: "noaa") '
Result: _result
Table: keys: []
         _value:string
----------------------
              h2o_feet
                h2o_pH
           h2o_quality
       h2o_temperature
```

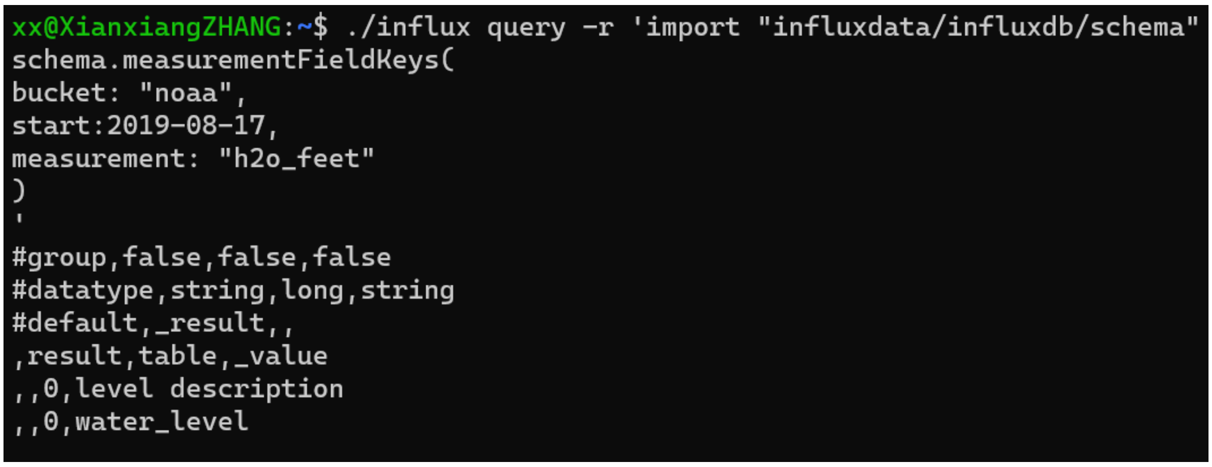


```
xx@XianxiangZHANG:~$ ./influx query -r 'import "influxdata/influxdb/schema"
schema.measurements(bucket: "noaa")'
#group,false,false,false
#datatype,string,long,string
#default,_result,,
,result,table,_value
,,0,h2o_feet
,,0,h2o_pH
,,0,h2o_quality
,,0,h2o_temperature
```

- **[Question]** What are the fields defined in the `h2o_feet` measurements, and the query to find them (hint: you have to specify a `predicate` parameter to filter against the appropriate measurement)?

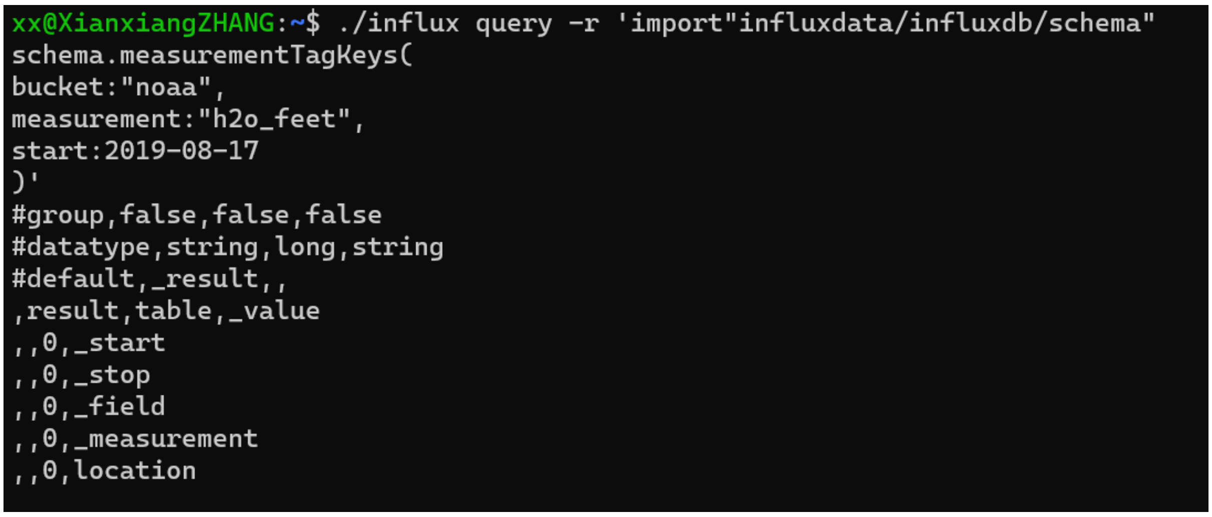


```
       h2o_temperature
mamisoa@LAPTOP-N2GTDULD:~$ ./influx query 'import "influxdata/influxdb/schema"
schema.measurementFieldKeys(bucket:"noaa",measurement:"h2o_feet",start :
-30y)'
Result: _result
Table: keys: []
         _value:string
----------------------
     level description
           water_level
```



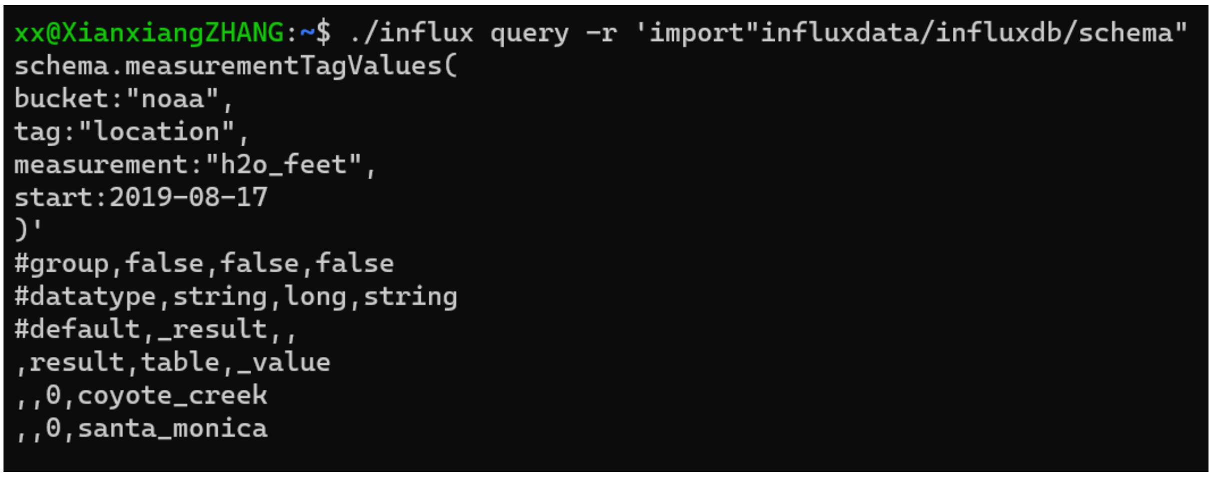
```
xx@XianxiangZHANG:~$ ./influx query -r 'import "influxdata/influxdb/schema"
schema.measurementFieldKeys(
bucket: "noaa",
start:2019-08-17,
measurement: "h2o_feet"
)
'
#group,false,false,false
#datatype,string,long,string
#default,_result,,
,result,table,_value
,,0,level description
,,0,water_level
```

- **[Question]** What are the tags keys defined in the `h2o_feet` measurements, and the query to find them?



```
xx@XianxiangZHANG:~$ ./influx query -r 'import"influxdata/influxdb/schema"
schema.measurementTagKeys(
bucket:"noaa",
measurement:"h2o_feet",
start:2019-08-17
)'
#group,false,false,false
#datatype,string,long,string
#default,_result,,
,result,table,_value
,,0,_start
,,0,_stop
,,0,_field
,,0,_measurement
,,0,location
```

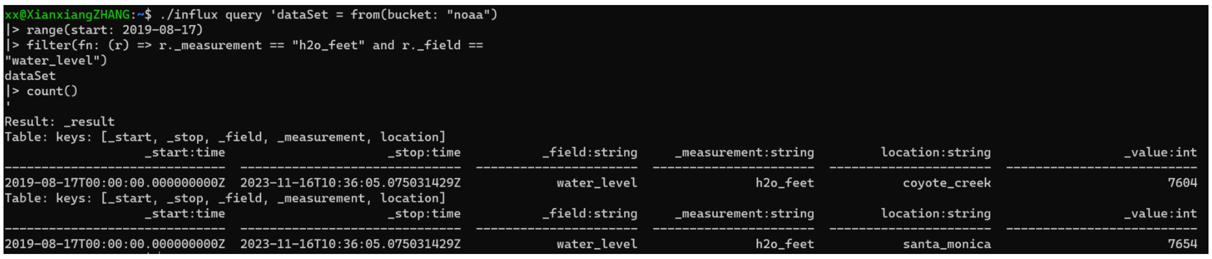
- **[Question]** What are the different values for the `location` tag in `h2o_feet` measurements, and the query to find them?



```
xx@XianxiangZHANG:~$ ./influx query -r 'import"influxdata/influxdb/schema"
schema.measurementTagValues(
bucket:"noaa",
tag:"location",
measurement:"h2o_feet",
start:2019-08-17
)'
#group,false,false,false
#datatype,string,long,string
#default,_result,,
,result,table,_value
,,0,coyote_creek
,,0,santa_monica
```

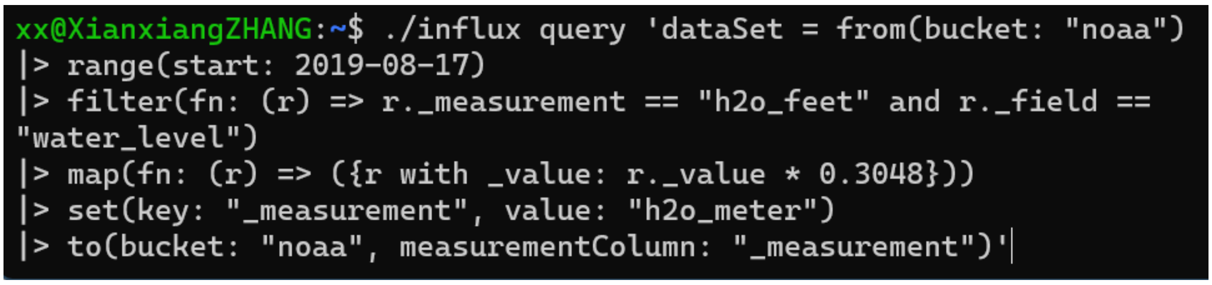
## Data exploration

- **[Question]** How many measurement points of **water level** are there in the `h2o_feet` measurement for each location? (hint: `count`). Give the query used.



```
xx@XianxiangZHANG:~$ ./influx query 'dataSet = from(bucket: "noaa")
|> range(start: 2019-08-17)
|> filter(fn: (r) => r._measurement == "h2o_feet" and r._field ==
"water_level")
dataSet
|> count()
'
Result: _result
Table: keys: [_start, _stop, _field, _measurement, location]
                   _start:time                      _stop:time           _field:string     _measurement:string          location:string                    _value:int
------------------------------  ------------------------------  ----------------------  ----------------------  ----------------------  --------------------------
2019-08-17T00:00:00.000000000Z  2023-11-16T10:36:05.075031429Z             water_level                h2o_feet            coyote_creek                        7604
Table: keys: [_start, _stop, _field, _measurement, location]
                   _start:time                      _stop:time           _field:string     _measurement:string          location:string                    _value:int
------------------------------  ------------------------------  ----------------------  ----------------------  ----------------------  --------------------------
2019-08-17T00:00:00.000000000Z  2023-11-16T10:36:05.075031429Z             water_level                h2o_feet            santa_monica                        7654
```

- **[Question]** Convert the `h2o_feet` measurement into a new `h2o_meter` measurement by converting the `water_level` values to meters (hint: `map` for mapping values, `set` for updating measurement name, `to` for storing the new measurement into `noaa`). Give the query used.



```
xx@XianxiangZHANG:~$ ./influx query 'dataSet = from(bucket: "noaa")
|> range(start: 2019-08-17)
|> filter(fn: (r) => r._measurement == "h2o_feet" and r._field ==
"water_level")
|> map(fn: (r) => ({r with _value: r._value * 0.3048}))
|> set(key: "_measurement", value: "h2o_meter")
|> to(bucket: "noaa", measurementColumn: "_measurement")'
```

## Using the Giraffe GUI

**[Question]**

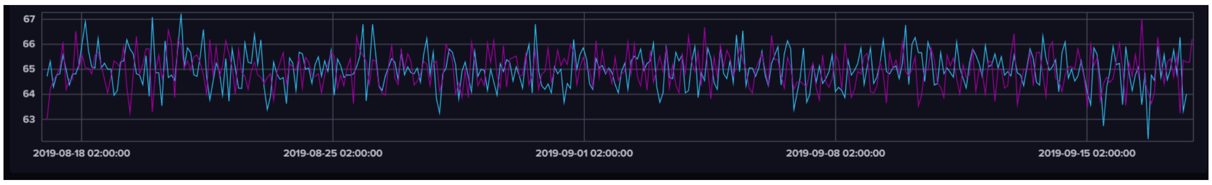
- Display the `h2o_temperature` graph. Its values are in Farenheit degrees, do a conversion (hint: map) to get a display in celsius degrees. Give the used query.

```
from(bucket: "noaa")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "h2o_temperature")
  |> filter(fn: (r) => r["_field"] == "degrees")
  |> map(fn: (r) => ({r with _value: (float(v: r._value) - 32.0) * 
5.0 / 9.0} ))
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: 
false)
  |> yield(name: "mean")
```

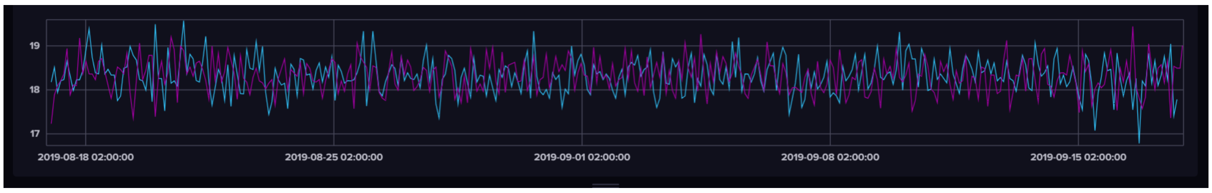
- Dashboard building: create a new Dashboard where you will define 2 timeline views: one for the water level, and the other for the average temperature.

**[Question]** Screenshot of visualisation

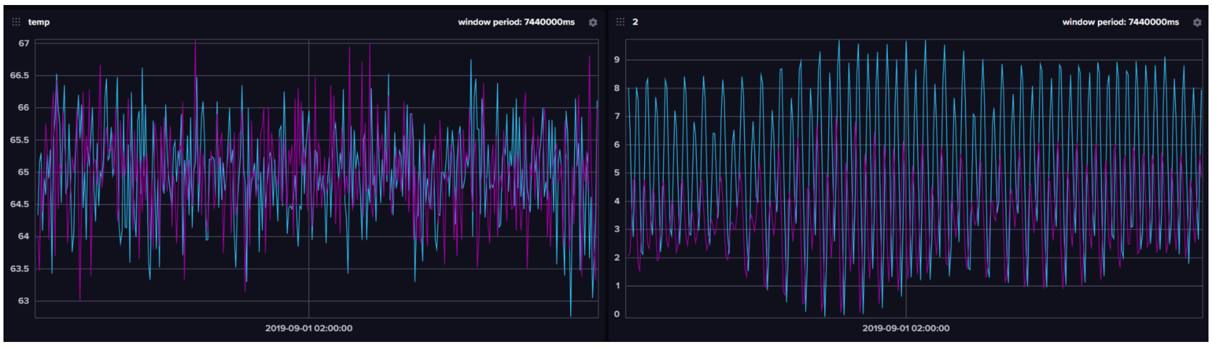
Visualisation for the Farenheit degrees



Visualisation for Celcius Degree



Visualisation for the Dashboard



## Injecting data

**[Question]** Provide the relevant source code extract that you had to implement.

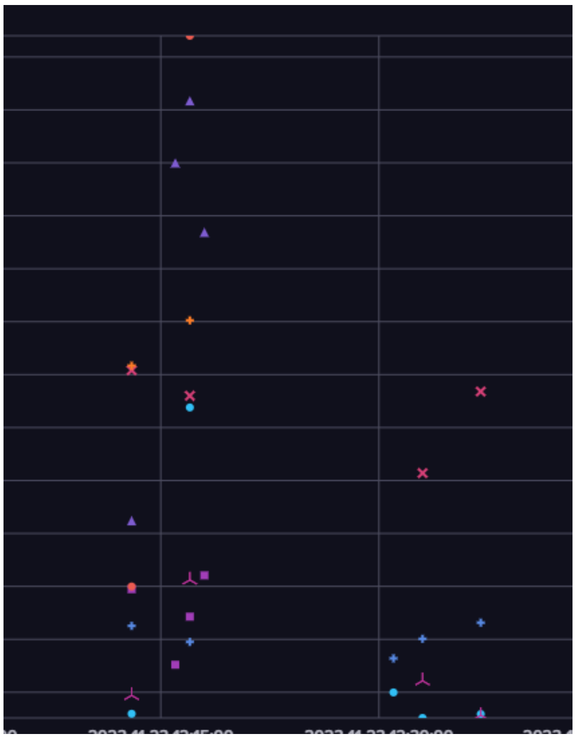
```
const influxurl = 'http://localhost:8086';
    /** InfluxDB authorization token */
    const token =
'V6UHTIPKxfBDr1h6L4i4bJP4uq98QPTph2N3vbmHU36L4F6sgMErX8iFMFT5Bms5vjzkeTT2baR
J7skjTF7rlw=='
    /** InfluxDB organization */
    const org = 'tp';
    /** InfluxDB bucket used for onboarding and write requests. */
    const bucket = 'mouse';

    const influxdb = new InfluxDB({url:influxurl, token:token});
    /* TODO: look at the doc to know how to connect to the appropriate
database */

    const writeApi = influxdb.getWriteApi(org, bucket);
    /* TODO: look at the doc to know how to get a write api endpoint */
```

```
var mouse_monitor = function(e) {
        var widget = e.target.id || "unknown";
        /* TODO: Send event with fields x (e.pageX) and y (e.pageY) and tags
url and widget */
        writeApi.writePoint(
          new Point('mouse_point')
          .floatField('x', e.pageX)
          .floatField('y', - e.pageY)
          .tag('url', url)
          .tag('widget', widget)
        );
        writeApi.flush(true);
    }
```

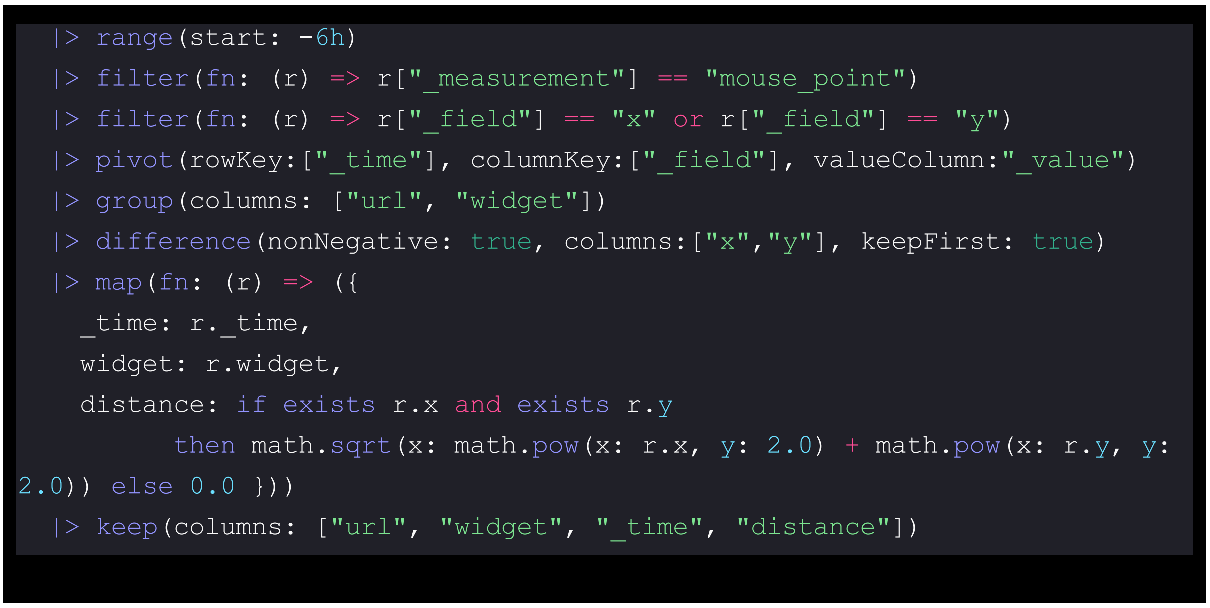
**[Question]** Use the Scatter plot to display mouse trajectories in a meaningful way, illustrating the captured moves. You will have to `pivot` the data. Give the Flux query and a screenshot.



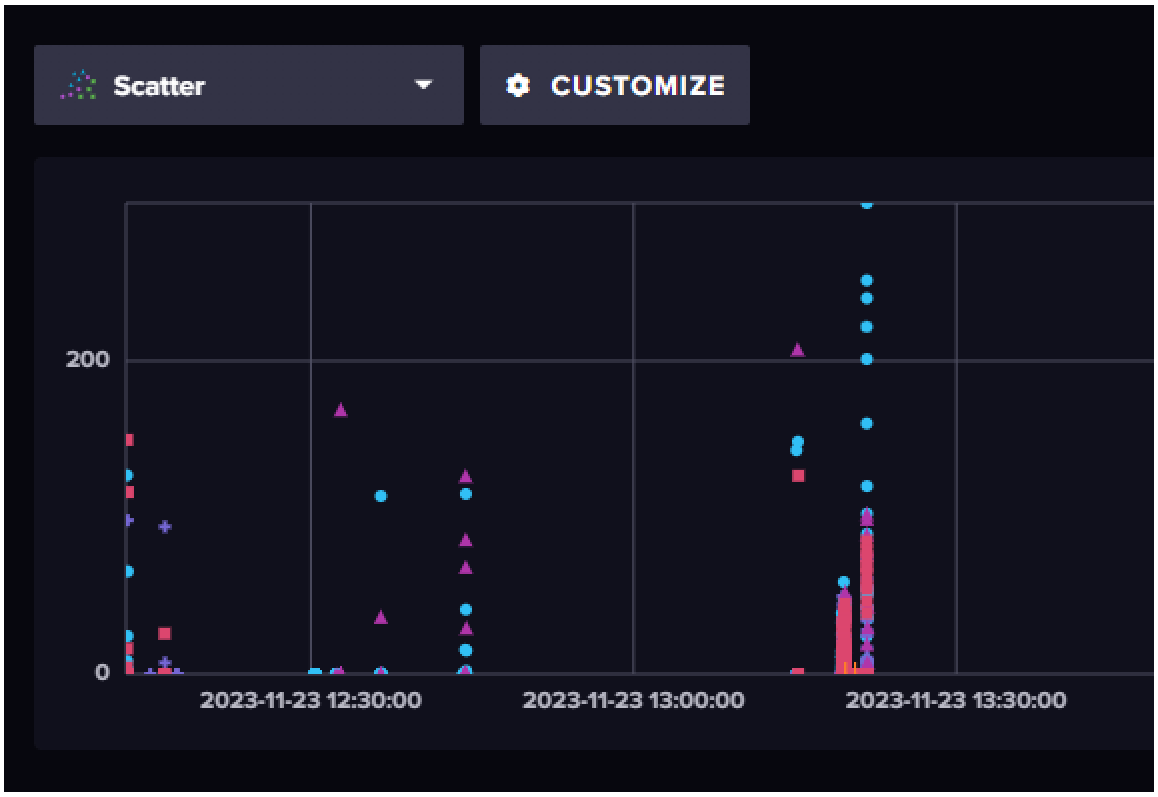
- Determine the global distance covered by the mouse in each of the squares (hint: it involves `pivot`, `difference`, and `math.sqrt`.

**[Question]** Give the query used to obtain the result

```
import "math"
from(bucket: "mouse")
```

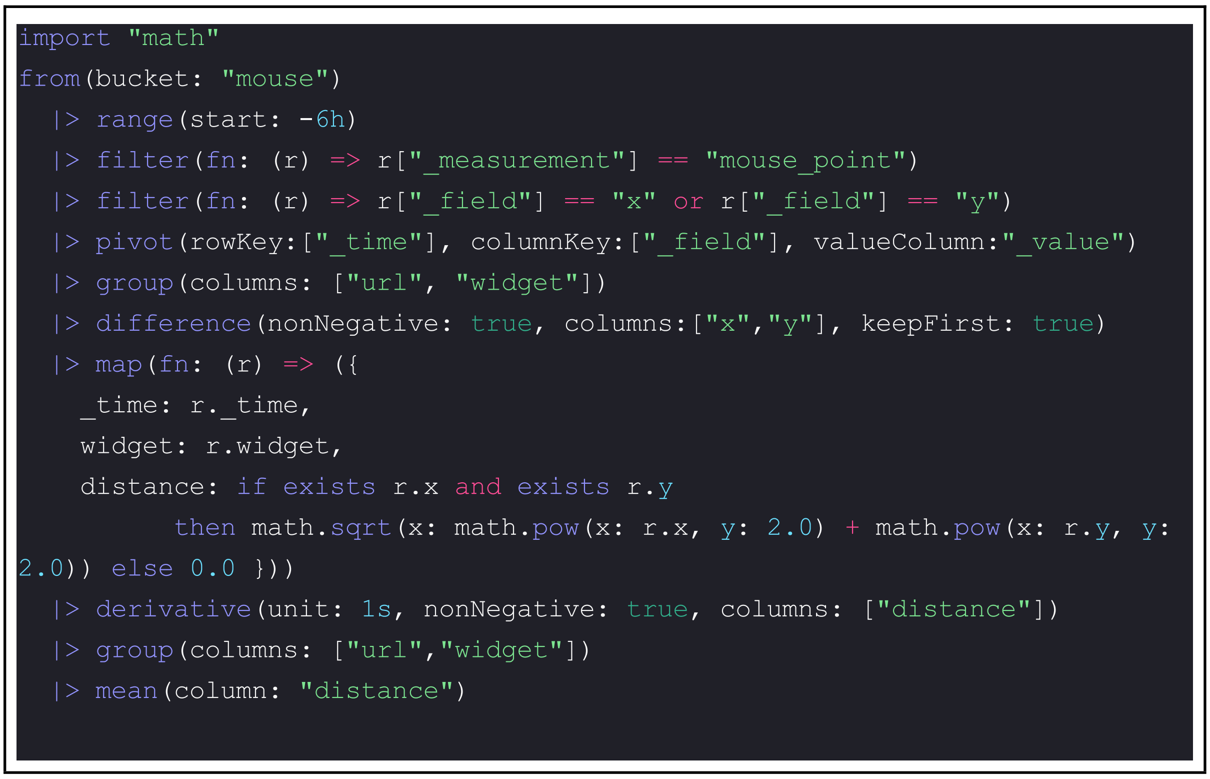


```
  |> range(start: -6h)
  |> filter(fn: (r) => r["_measurement"] == "mouse_point")
  |> filter(fn: (r) => r["_field"] == "x" or r["_field"] == "y")
  |> pivot(rowKey:["_time"], columnKey:["_field"], valueColumn:"_value")
  |> group(columns: ["url", "widget"])
  |> difference(nonNegative: true, columns:["x","y"], keepFirst: true)
  |> map(fn: (r) => ({
    _time: r._time,
    widget: r.widget,
    distance: if exists r.x and exists r.y
          then math.sqrt(x: math.pow(x: r.x, y: 2.0) + math.pow(x: r.y, y:
2.0)) else 0.0 }))
  |> keep(columns: ["url", "widget", "_time", "distance"])
```

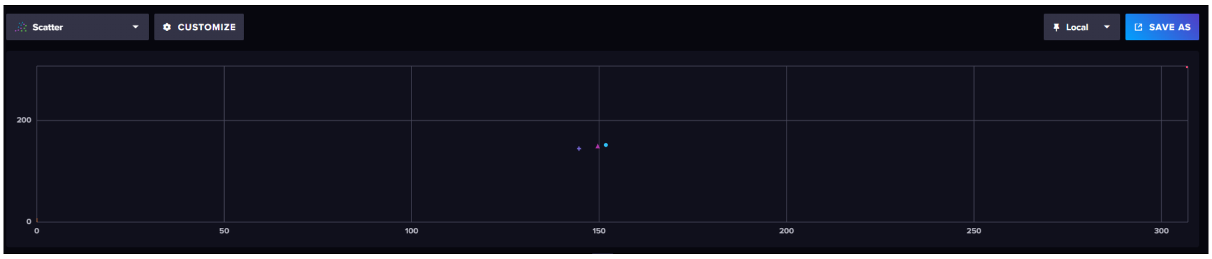


- Extending the previous query, determine the mean speed (in pixels/seconds) of the mouse for each of the squares (hint: it involves `derivative`)

**[Question]** Give the query used to obtain the result



```
import "math"
from(bucket: "mouse")
  |> range(start: -6h)
  |> filter(fn: (r) => r["_measurement"] == "mouse_point")
  |> filter(fn: (r) => r["_field"] == "x" or r["_field"] == "y")
  |> pivot(rowKey:["_time"], columnKey:["_field"], valueColumn:"_value")
  |> group(columns: ["url", "widget"])
  |> difference(nonNegative: true, columns:["x","y"], keepFirst: true)
  |> map(fn: (r) => ({
    _time: r._time,
    widget: r.widget,
    distance: if exists r.x and exists r.y
          then math.sqrt(x: math.pow(x: r.x, y: 2.0) + math.pow(x: r.y, y:
2.0)) else 0.0 }))
  |> derivative(unit: 1s, nonNegative: true, columns: ["distance"])
  |> group(columns: ["url","widget"])
  |> mean(column: "distance")
```


Scatter
CUSTOMIZE
Local
SAVE AS
200
0
0
50
100
150
200
250
300