



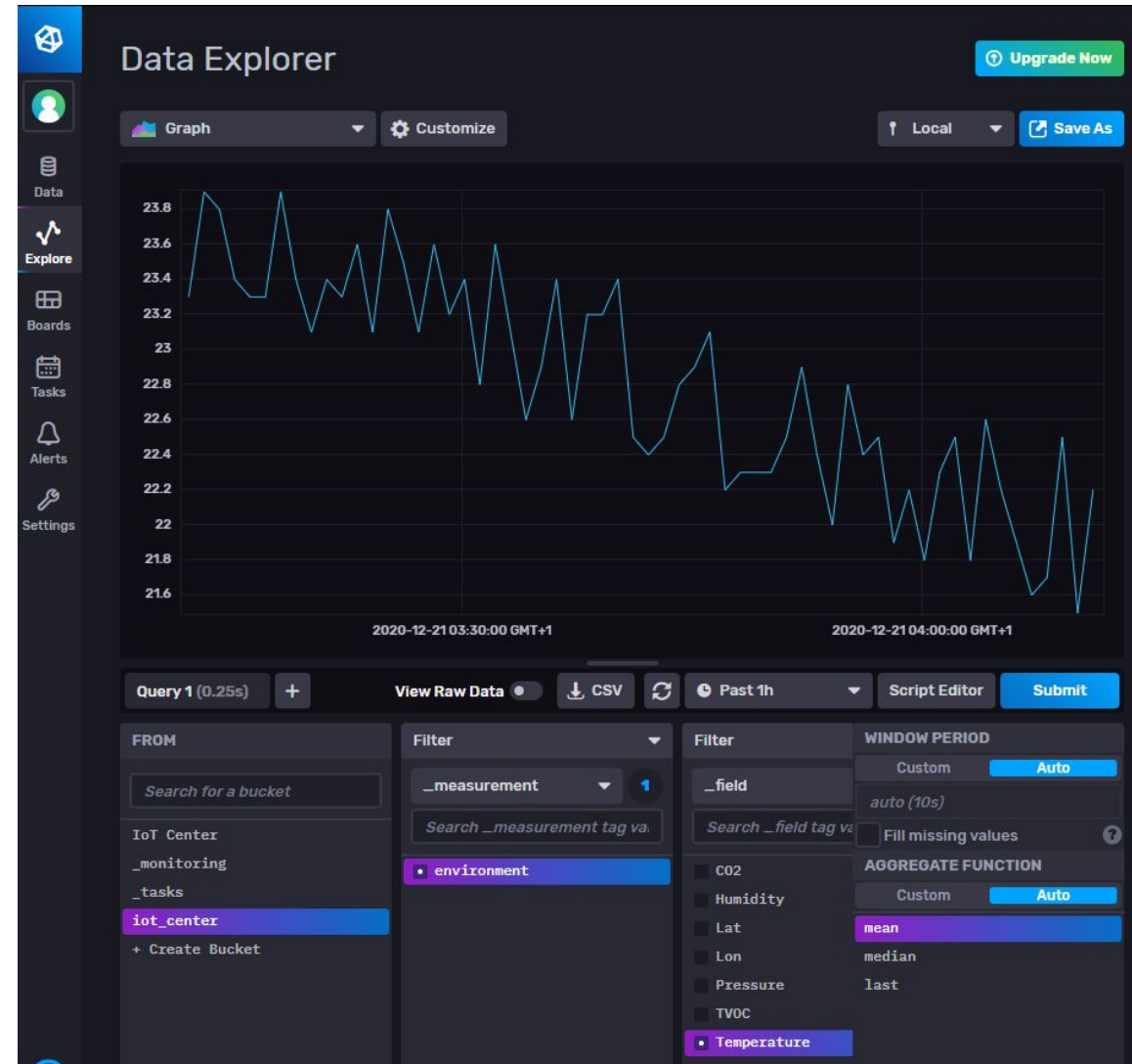
InfluxDB Data Explorer

Workshop



Agenda

- InfluxDB Cloud Login
- Introduce Flux Query Language
- Data Explorer
- Exercise: Query GEO Data



Login into Influxdb Cloud 2

Open Web Browser

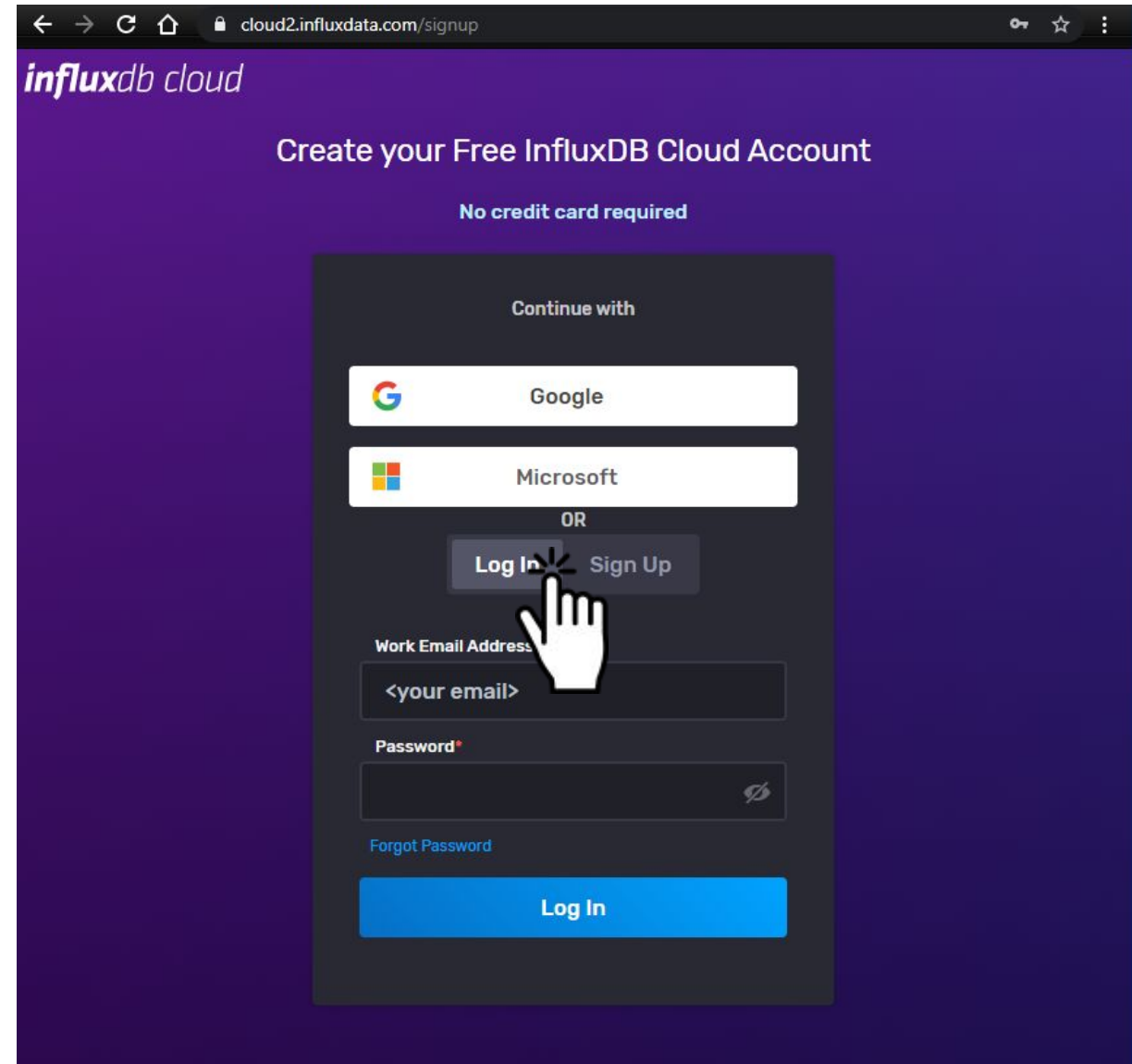
<https://cloud2.influxdata.com/>

Click to **Log In** option

Three options available:

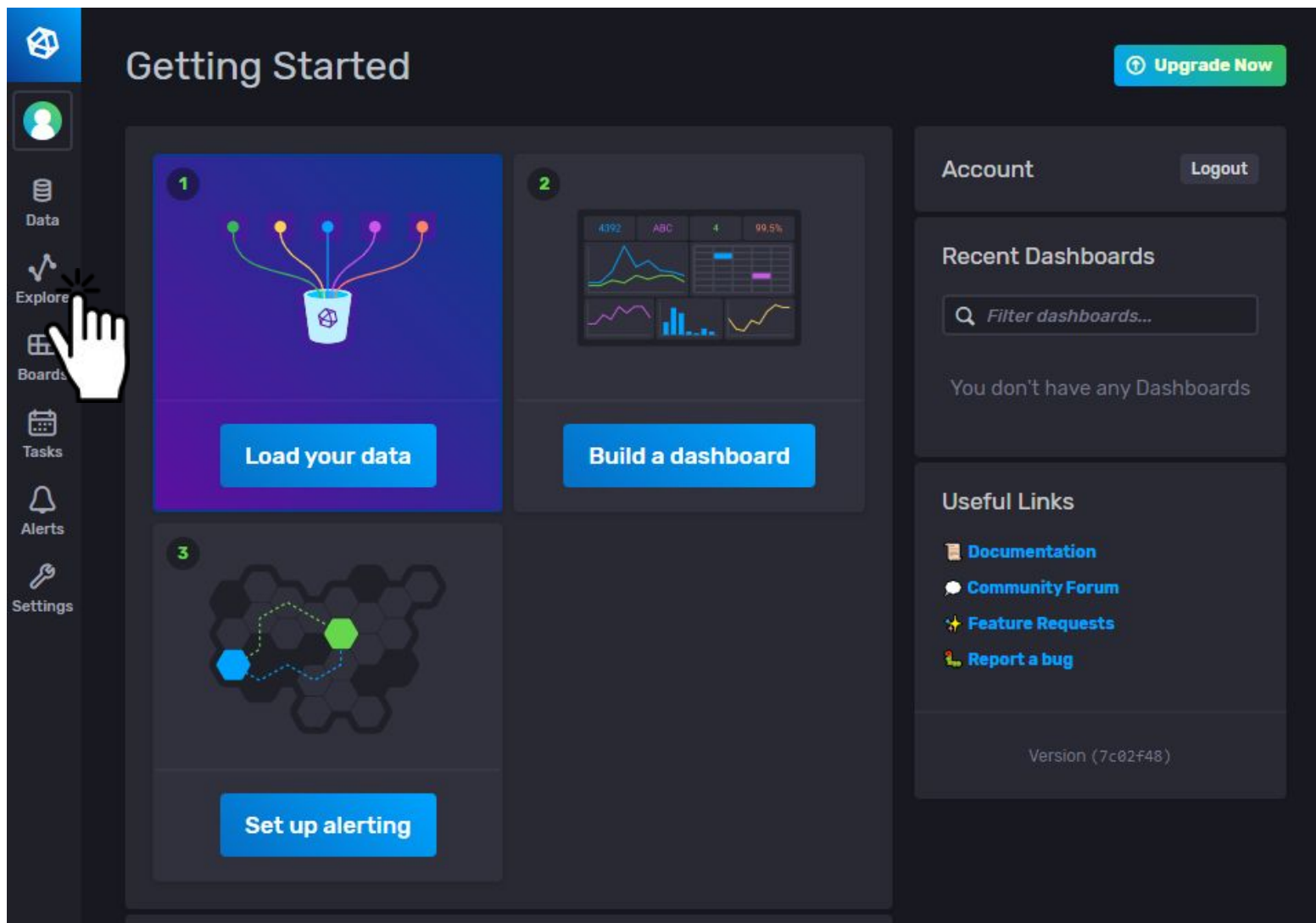
1. Google account
2. Microsoft account
3. Own email

Click to **Log In** option



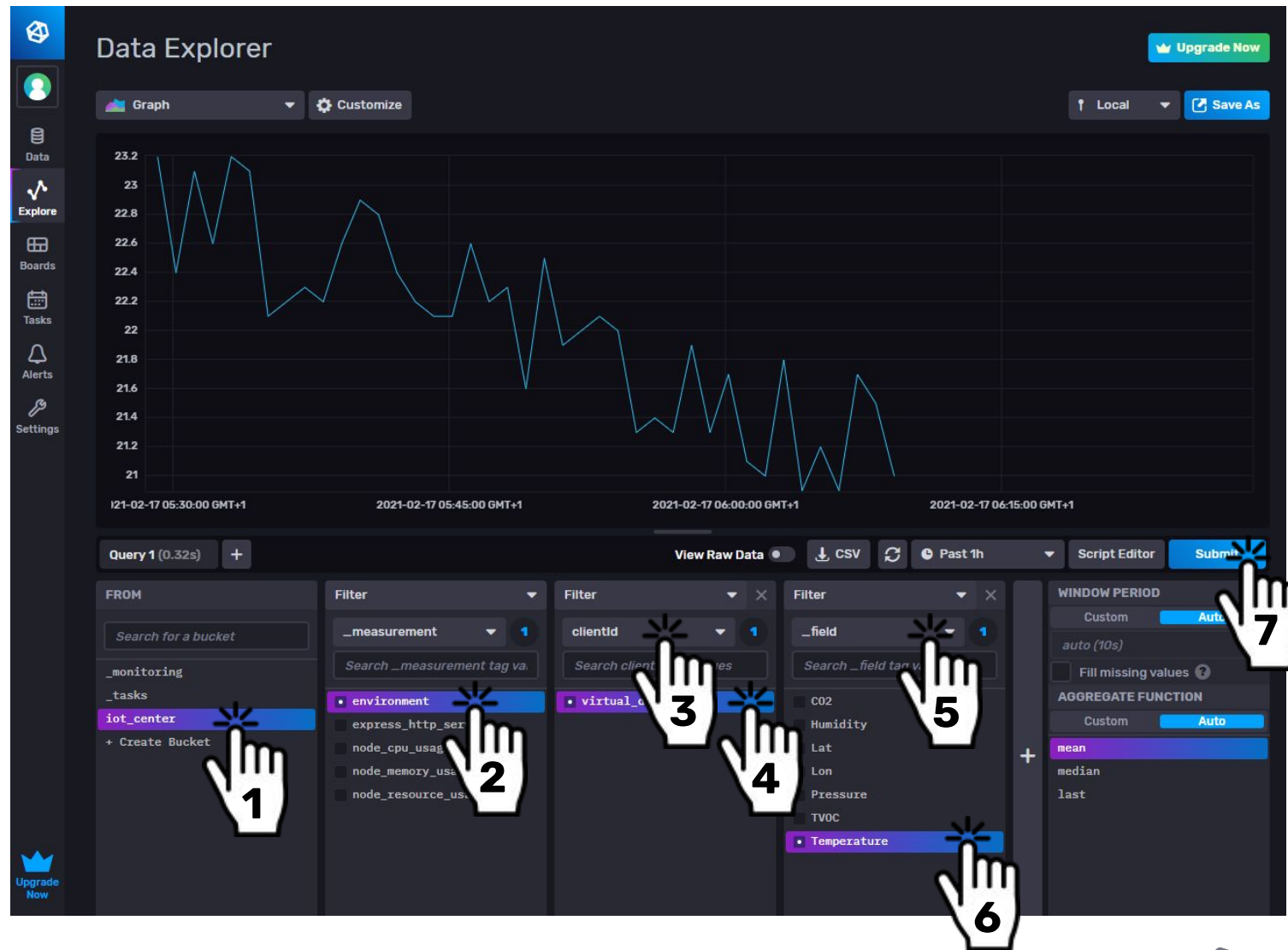
Start Explorer

Click on Explore Icon



Query InfluxDB

- Select bucket
 - **iot_centrum**
- Select measurement
 - **environment**
- Select field
 - **Temperature**
- Keep aggregation
 - **Mean**
- Set Time
 - **Past 1h**

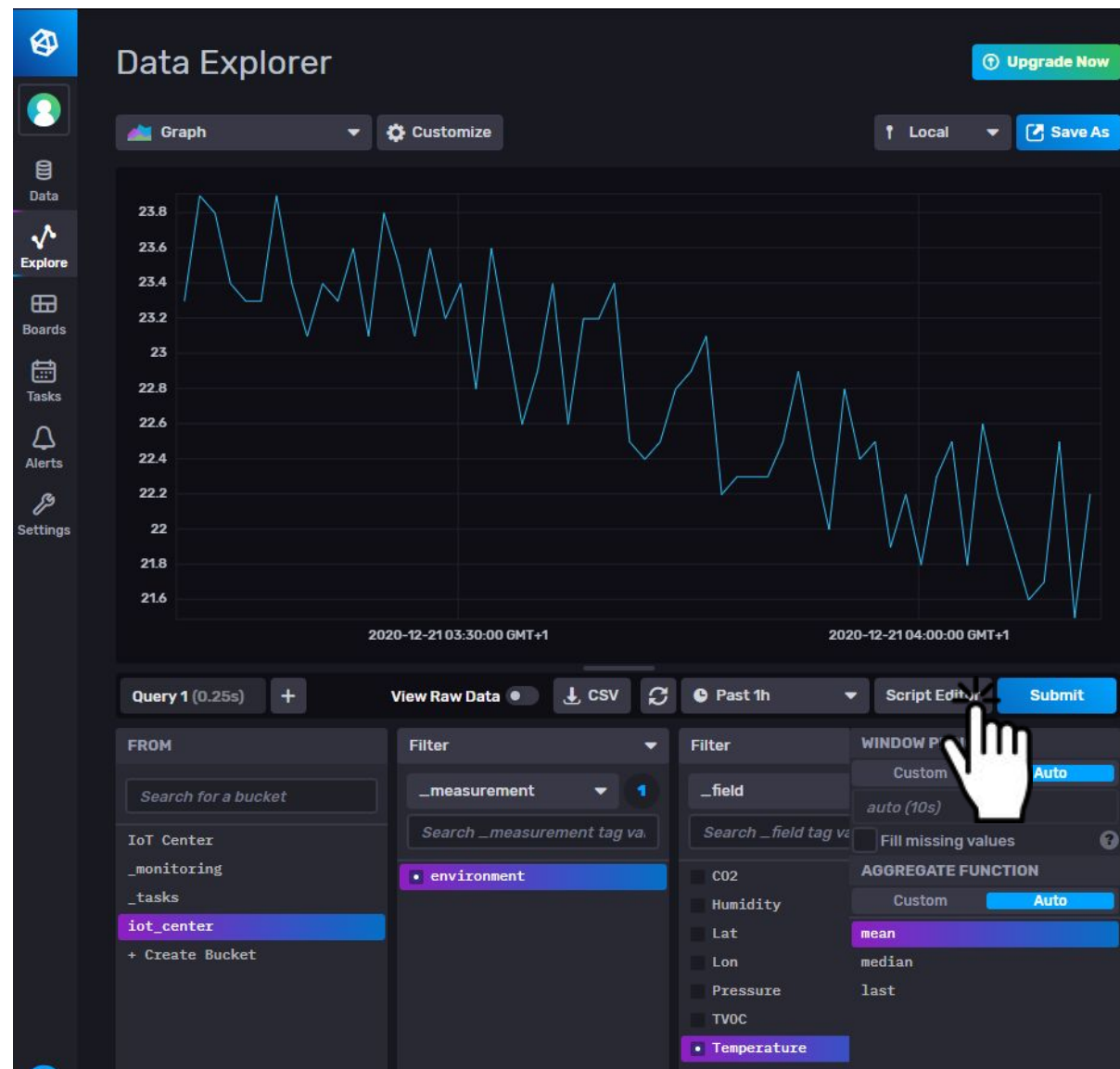


Switch to the Script Editor

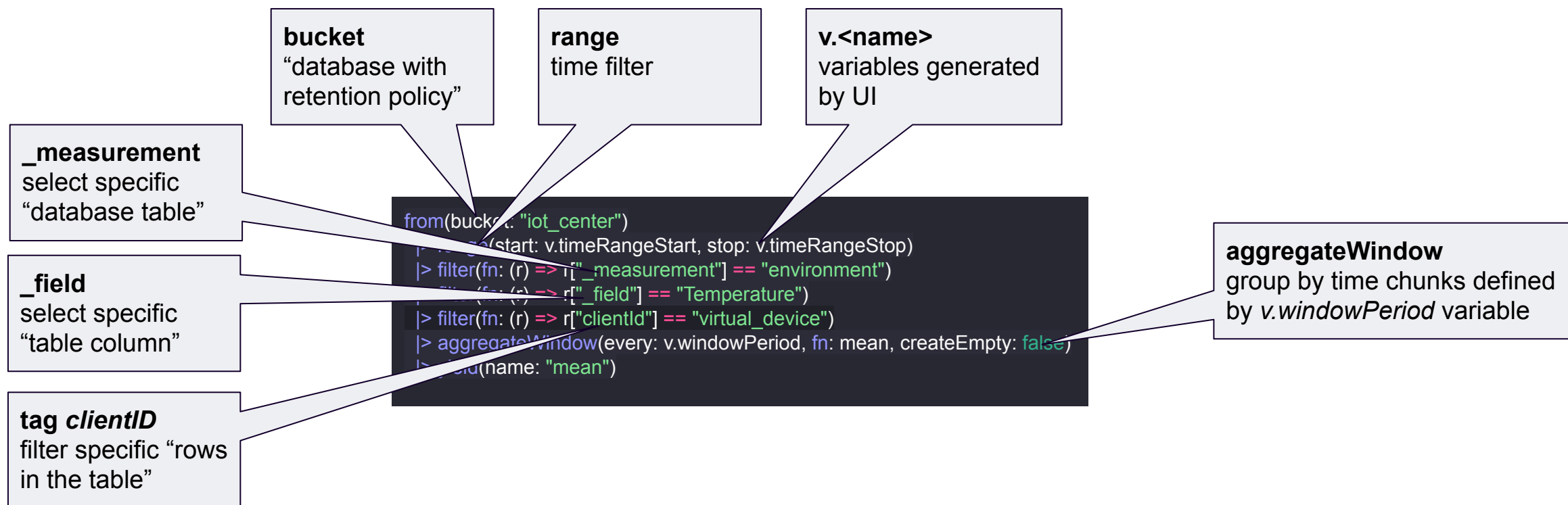
Click to Script Editor Button

Show **Flux Query**

```
from(bucket: "iot_center")
  > range(start: v.timeRangeStart, stop: v.timeRangeStop)
  > filter(fn: (r) => r["_measurement"] == "environment")
  > filter(fn: (r) => r["clientId"] == "virtual_device")
  > filter(fn: (r) => r["_field"] == "Temperature")
  > aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  > yield(name: "mean")
```



Basic Flux Query Structure



Show Raw Query Data

Click to **View Raw Data**

- Table with all the columns

The screenshot shows the InfluxDB Data Explorer interface. At the top, there's a 'Data Explorer' header with a 'Graph' dropdown and a 'Customize' button. On the right, there's an 'Upgrade Now' button and a 'Local' dropdown with a 'Save As' button. The main area displays a table of raw query data. The table has four columns: 'true', 'true', 'false', and 'false'. The first two columns are labeled 'dateTime: RFC3339' and the last two are labeled 'double'. The table contains 10 rows of data, each with a timestamp and a value. A hand icon points to the 'View Raw Data' button in the bottom right corner of the interface. Below the table, there's a query editor with a query for 'Query 1 (0.26s)' and a 'Submit' button. The query is as follows:

```
1 from(bucket: "iot_center")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r["_measurement"] == "environment")
4   |> filter(fn: (r) => r["_field"] == "Temperature")
5   |> filter(fn: (r) => r["_clientId"] == "virtual_device")
6   |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
7   |> yield(name: "mean")
```

On the right side of the interface, there's a 'Functions' panel with a search bar and a list of functions: 'aggregate.rate', 'chandeMomentumOscillator', and 'columns'.



Task: How to Get GPS Coordinates?

We need this query for the GEO widget

Virtual device provides GPS coordinates

- Fields **Lat** and **Lon**
- How to adjust the query?
- Two options

```
from(bucket: "iot_center")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["measurement"] == "environment")
  |> filter(fn: (r) => r["_field"] == "Temperature")
  |> filter(fn: (r) => r["clientId"] == "virtual_device")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```



1/2 Using Query Builder

The screenshot displays the InfluxDB Data Explorer interface. At the top, the title "Data Explorer" is visible, along with an "Upgrade Now" button. Below the title, there are tabs for "Graph" and "Customize", and a "Local" dropdown menu. A "Save As" button is also present. The main area shows a table of data with columns: `true`, `dateTime:RFC3339`, `false`, and `double`. The table contains 10 rows of data, with the first row showing `true`, `dateTime:RFC3339`, `false`, and `double`. The subsequent rows show timestamps and values for `_start`, `_stop`, `_time`, and `_value`.

Below the table, the query builder is visible. It includes a "Query 1 (0.24s)" label, a "View Raw Data" toggle, a "CSV" download button, a "Past 24h" time range selector, a "Script Editor" button, and a "Submit" button. The query builder is divided into several sections: "FROM" (with a search bar and a list of buckets including `_monitoring`, `_tasks`, and `iot_center`), "Filter" (with a dropdown for `_measurement` and a search bar), "Filter" (with a dropdown for `_field` and a search bar), and "WINDOW PERIOD" (with a dropdown for `clientid` and a search bar). The "AGGREGATE FUNCTION" section is also visible, with a dropdown for `virtual_device` and a search bar.



2/2 Using Script Editor

Add Lat, Lon
select
"database columns"

```
from(bucket: "iot_center")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "environment")
  |> filter(fn: (r) => r["_field"] == "Lat" or r["_field"] == "Lon")
  |> filter(fn: (r) => r["clientId"] == "virtual_device")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

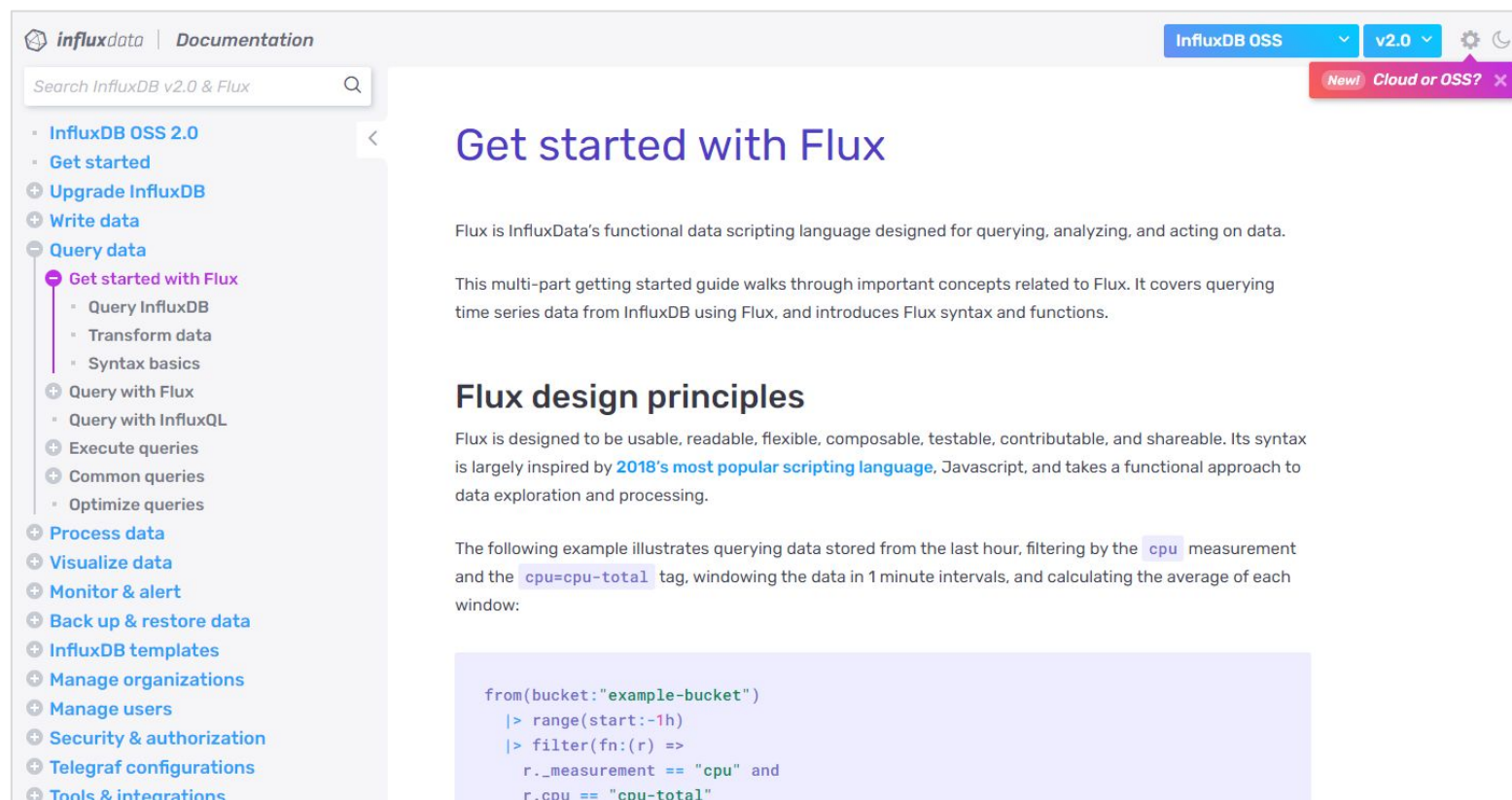
_start	_stop	_time	_value
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T15:36:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T15:40:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T15:44:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T15:48:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T15:52:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T15:56:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T16:00:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T16:04:00.000Z	14.4071543
2020-12-21T15:31:52.862Z	2020-12-22T15:31:52.862Z	2020-12-21T16:08:00.000Z	14.4071543



Documentation

More information:

<https://docs.influxdata.com/influxdb/v2.0/query-data/get-started/>



The screenshot shows the InfluxData documentation website. The header includes the InfluxData logo, the word 'Documentation', and navigation links for 'InfluxDB OSS' and 'v2.0'. A search bar is located on the left. The left sidebar contains a table of contents with categories like 'InfluxDB OSS 2.0', 'Get started', 'Upgrade InfluxDB', 'Write data', 'Query data', 'Process data', 'Visualize data', 'Monitor & alert', 'Back up & restore data', 'InfluxDB templates', 'Manage organizations', 'Manage users', 'Security & authorization', 'Telegraf configurations', and 'Tools & integrations'. The 'Query data' category is expanded, showing 'Get started with Flux' as the selected item. The main content area is titled 'Get started with Flux' and contains the following text:

Flux is InfluxData's functional data scripting language designed for querying, analyzing, and acting on data.

This multi-part getting started guide walks through important concepts related to Flux. It covers querying time series data from InfluxDB using Flux, and introduces Flux syntax and functions.

Flux design principles

Flux is designed to be usable, readable, flexible, composable, testable, contributable, and shareable. Its syntax is largely inspired by [2018's most popular scripting language](#), Javascript, and takes a functional approach to data exploration and processing.

The following example illustrates querying data stored from the last hour, filtering by the `cpu` measurement and the `cpu=cpu-total` tag, windowing the data in 1 minute intervals, and calculating the average of each window:

```
from(bucket:"example-bucket")
  |> range(start:-1h)
  |> filter(fn:(r) =>
    r._measurement == "cpu" and
    r.cpu == "cpu-total")
```

