Napredne arhitekture informacionih sistema

Baze podataka vremenskih nizova

Izvođači nastave:
dr Marko Vještica
Elena Akik
Sanja Radić

DAIS
Data Science and Information Systems Group

Sadržaj

Softverska podrška

- **Lokalno** – *InfluxDB*
  - *Link: https://docs.influxdata.com/influxdb/v2/install/*
- **Preko *Docker*-a** – pomoću pokretanja datoteke *docker-compose* u kojoj se nalazi *InfluxDB*
  - *InfluxDB-u* moguće je direktno pristupiti na adresi: *http://localhost:8086*

Sadržaj

Uvod u baze podataka vremenskih nizova

- **Baze podataka vremenskih nizova** su vrsta baza podataka dizajnirana za efikasno skladištenje, upravljanje i analizu podataka koji su **organizovani po vremenu**
- Pogodne su za podatke koji se generišu u **vremenskim intervalima**, kao što su podaci sa senzora, logovi servera i finansijski podaci
- Organizuju podatke po **vremenskim oznakama** (engl. *Timestamp*) i omogućavaju efikasno čuvanje, upravljanje i pretragu takvih podataka
- Koriste različite **tehnike kompresije i indeksiranja** kako bi smanjili prostor koji zauzimaju i ubrzali pristup podacima
- **Primeri** baza podataka vremenskih nizova:
  - InfluxDB, Prometheus, OpenTSDB, KairosDB

Sadržaj

Baza podataka *InfluxDB*

- ***InfluxDB*** je pionir u oblasti upravljanja podataka u formi vremenskih nizova, razvijen od strane tima u kompaniji InfluxData
- Uz razvoj baze podataka *InfluxDB*, stvoren je jezik ***Flux*** – namenski jezik (engl. *Domain Specific Language*) koji omogućava **naprednu analizu i manipulaciju vremenskim podacima**
  - Jezik *Flux* se ističe sposobnošću za brzu **analizu podataka u realnom vremenu**
  - Fleksibilnost jezika omogućava prilagođavanje analize podataka **različitim zahtevima** aplikacija
- Kako bi bio omogućen lakši prelaz iz sveta relacionih baza podataka u svet vremenskih baza podataka kreiran je ***InfluxQL***, koji omogućava takođe pretragu podataka, ali sa znatno manje mogućnosti

Struktura podataka u bazi podataka *InfluxDb*

- **Baket (engl. *Bucket*):** Imenovani prostor gde se čuvaju podaci vremenskih nizova
- Baket može sadržati različite grupe podataka, kao što su:
  - **Merenje (engl. *Measurement*)**: Logičko grupisanje podataka vremenskih nizova
    - Svi podaci unutar jednog merenja treba da dele iste karakteristike
    - Merenje obuhvata različite podatke i atribute
  - **Oznake (engl. *Tags*)**: Parovi ključ-vrednost koji definišu dodatne informacije o svakoj tački podataka
    - Omogućavaju dodavanje meta-podataka, odnosno konteksta, kao što su lokacija ili izvor podataka
  - **Polja (engl. *Fields*)**: Parovi ključ-vrednost koji predstavljaju osnovne podatke vremenskih nizova
    - Često predstavljaju numeričke ili tekstualne vrednosti koje se menjaju tokom vremena, kao što su temperatura, pritisak ili cena akcija
  - **Vremenska oznaka (engl. *Timestamp*)**: Vremenski indeks koji se povezuje sa svakom tačkom podataka
    - Prisutnost vremenske oznake omogućava uređivanje i organizaciju podataka po vremenskom redosledu prilikom upita ili čuvanja na disku

Struktura podataka u bazi podataka *InfluxDb*

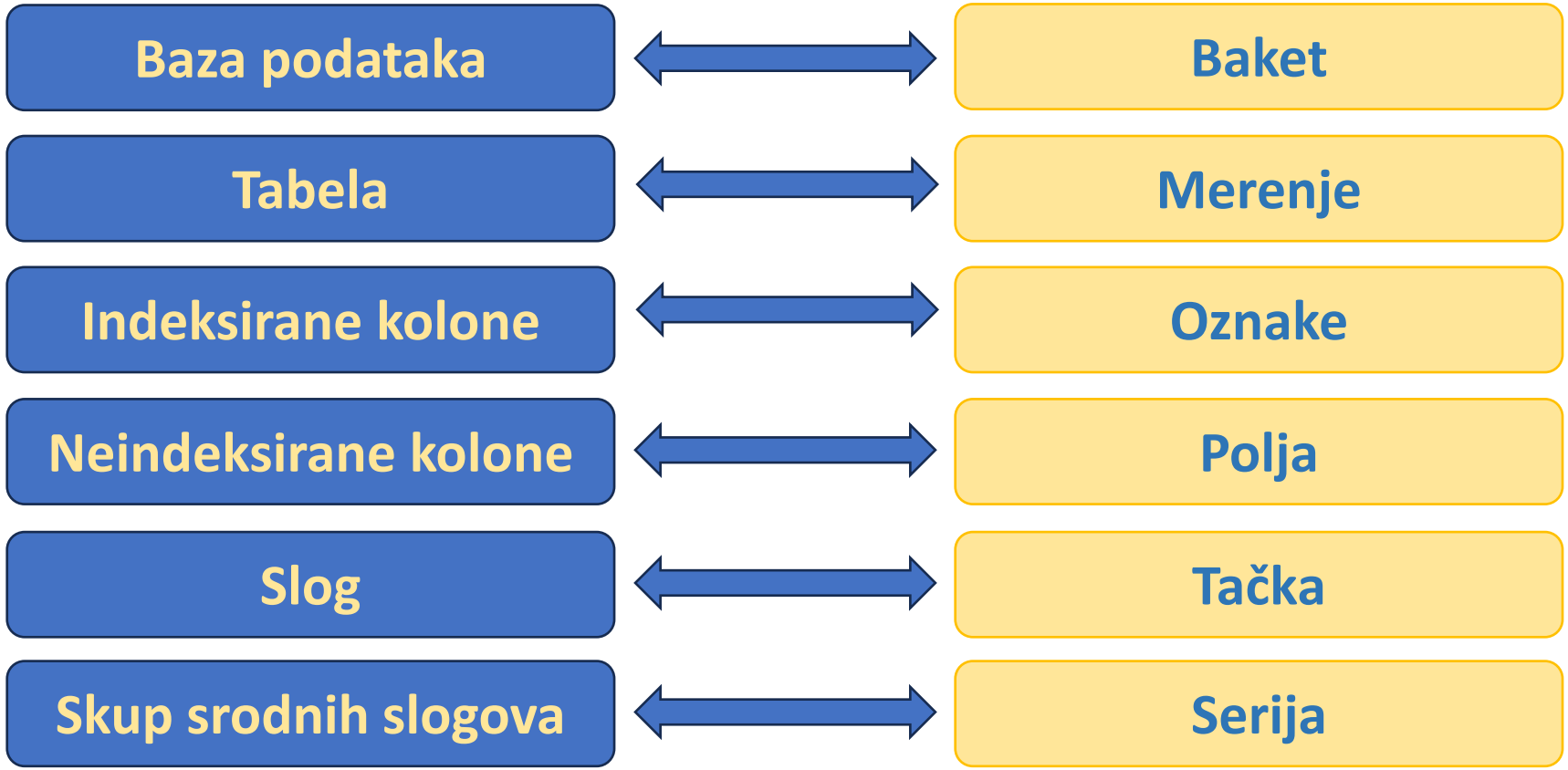
- ***Tačka* (engl. *Point*):** Jedan podatak identifikovan po svom merenju, oznakama, poljima i vremenskoj oznaci
- ***Serija* (engl. *Series*):** Grupa tačaka sa istim vrednostima merenja, oznaka i polja

time (Tačka)	_measurement (Merenje)	city (Oznake)	country (Oznake)	_field (Polja)	_value (Vrednosti)
2022-01-01T12:00:00Z	weather	London	UK	temperature	12.0
2022-02-01T12:00:00Z	weather	London	UK	temperature	12.1
2022-03-01T12:00:00Z	weather	London	UK	temperature	11.5
2022-04-01T12:00:00Z	weather	London	UK	temperature	5.9

SERIJA

Poređenje relacione i baze podataka *InfluxDB*

- Organizacija **relacionih** baza
- Organizacija baze podataka ***InfluxDB***



Kako se podaci čuvaju?

- Da bi se omogućila **efikasnost u skladištenju i pretraživanju podataka**, *InfluxDB* koristi **tri tehnike**:
  - **Grupisanje i uređivanje vrednosti po ključu serije:**
    - ***Storage engine*** prvo grupiše vrednosti po ključu serije
    - Zatim ih raspoređuje po vremenu unutar **TSM** datoteka
  - **Format podataka – stablo TSM (engl. *Time-Structured Merge tree*) datoteka:**
    - TSM datoteke čuvaju podatke serija koristeći **kompresiju i kolonarni format**
    - Čuvaju samo **razlike između vrednosti** radi efikasnosti pretraživanja i skladištenja podataka
  - **Kompaktiranje podataka:**
    - Nakon što se polja uspešno skladište u TSM datotekama, privremeni zapisi (poznati i kao Write Ahead Log, ili WAL) i keš memorija se brišu
    - Proces kompaktiranja stvara TSM datoteke koji su **brzi za čitanje**

time	City	_field	_value
2024-03-01T12:00:00Z	London	temperature	12,0
2024-03-01T12:05:00Z	London	temperature	12,1
2024-03-01T12:00:00Z	London	co2	0,65
2024-03-01T12:05:00Z	London	co2	0,60

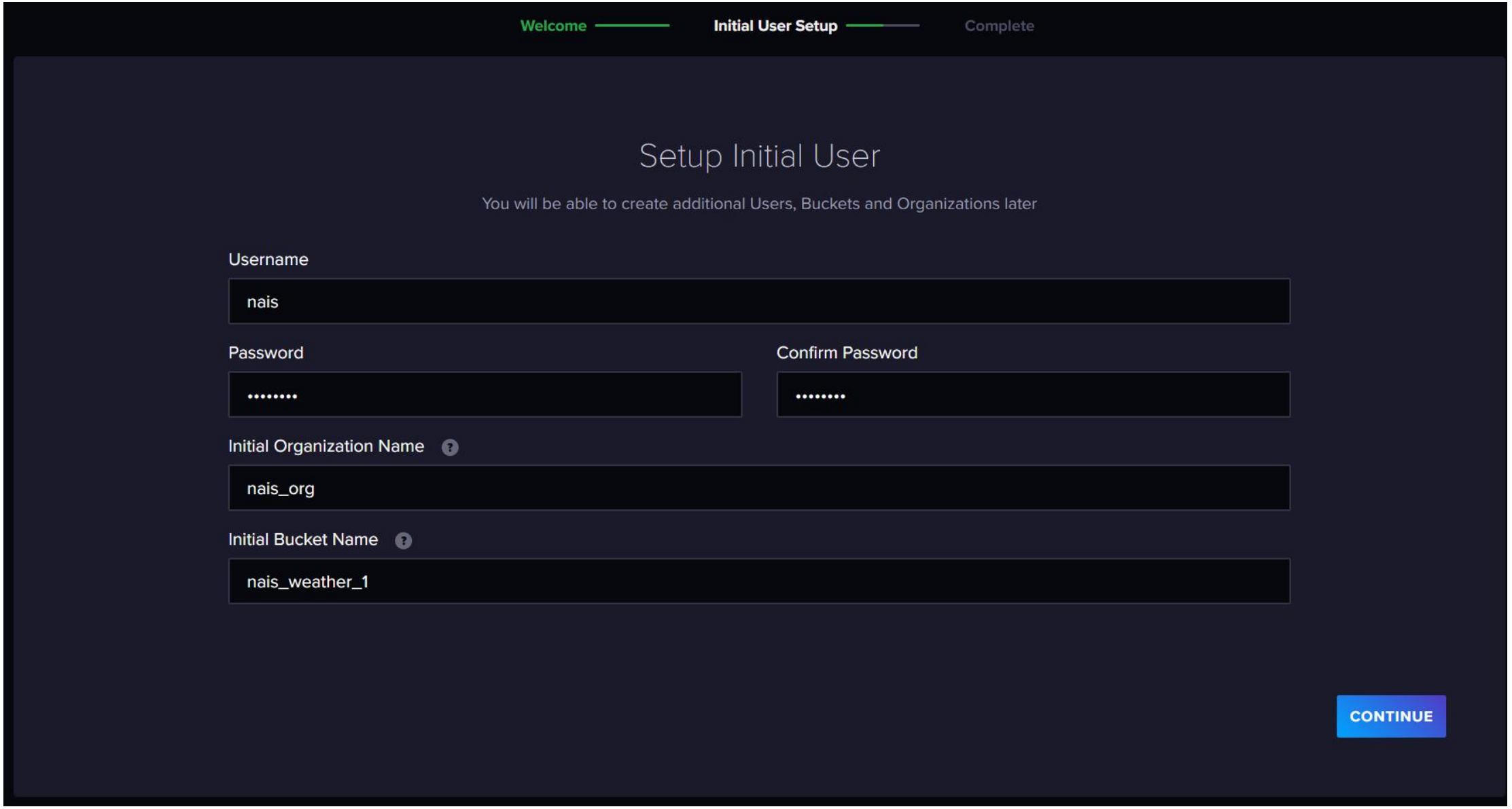
MERENJE

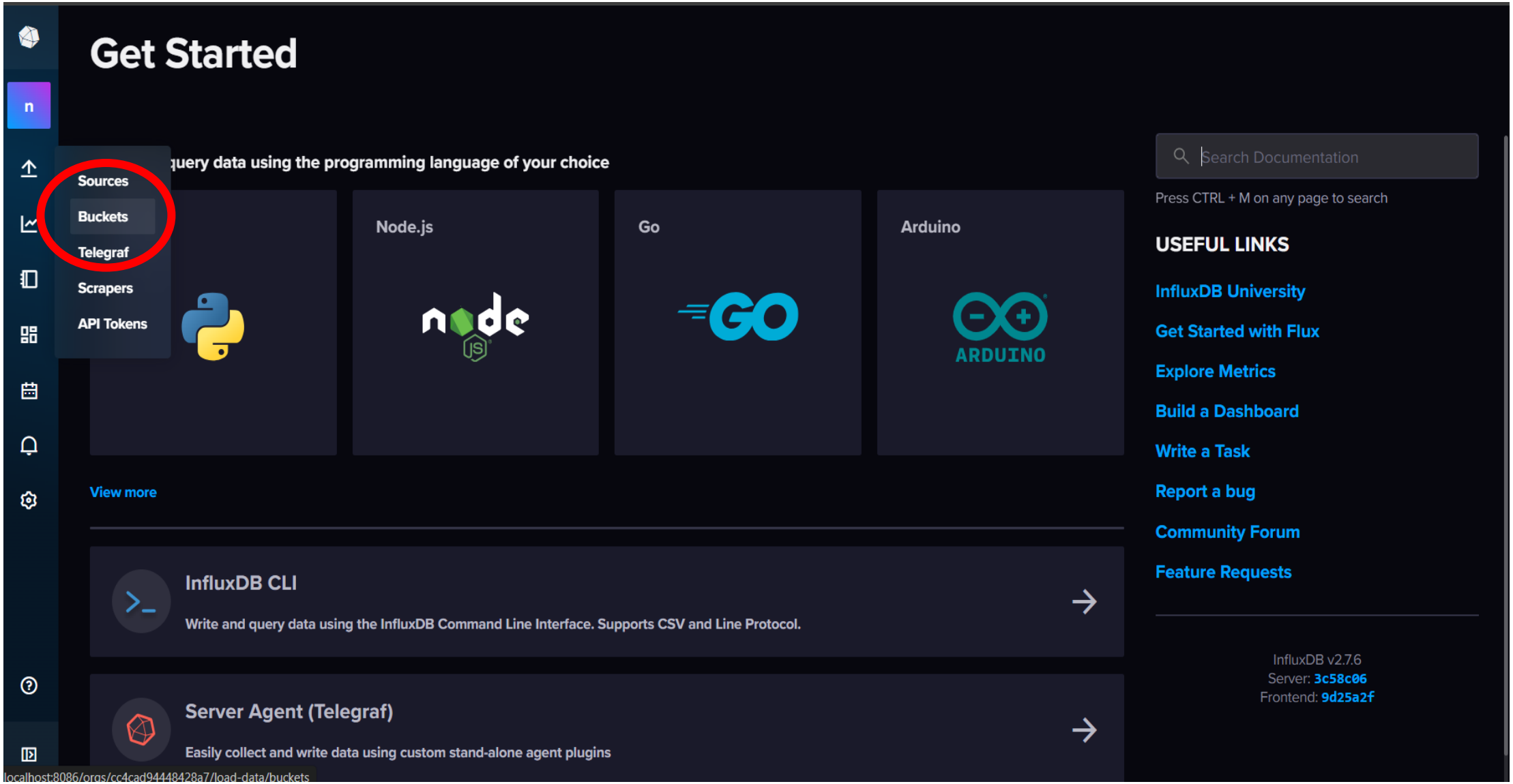
Podaci su razdvojeni po **ključu serije** (merenje + oznake + polja) koji su skladišteni u masovnoj memoriji u okviru **TSM datoteka**

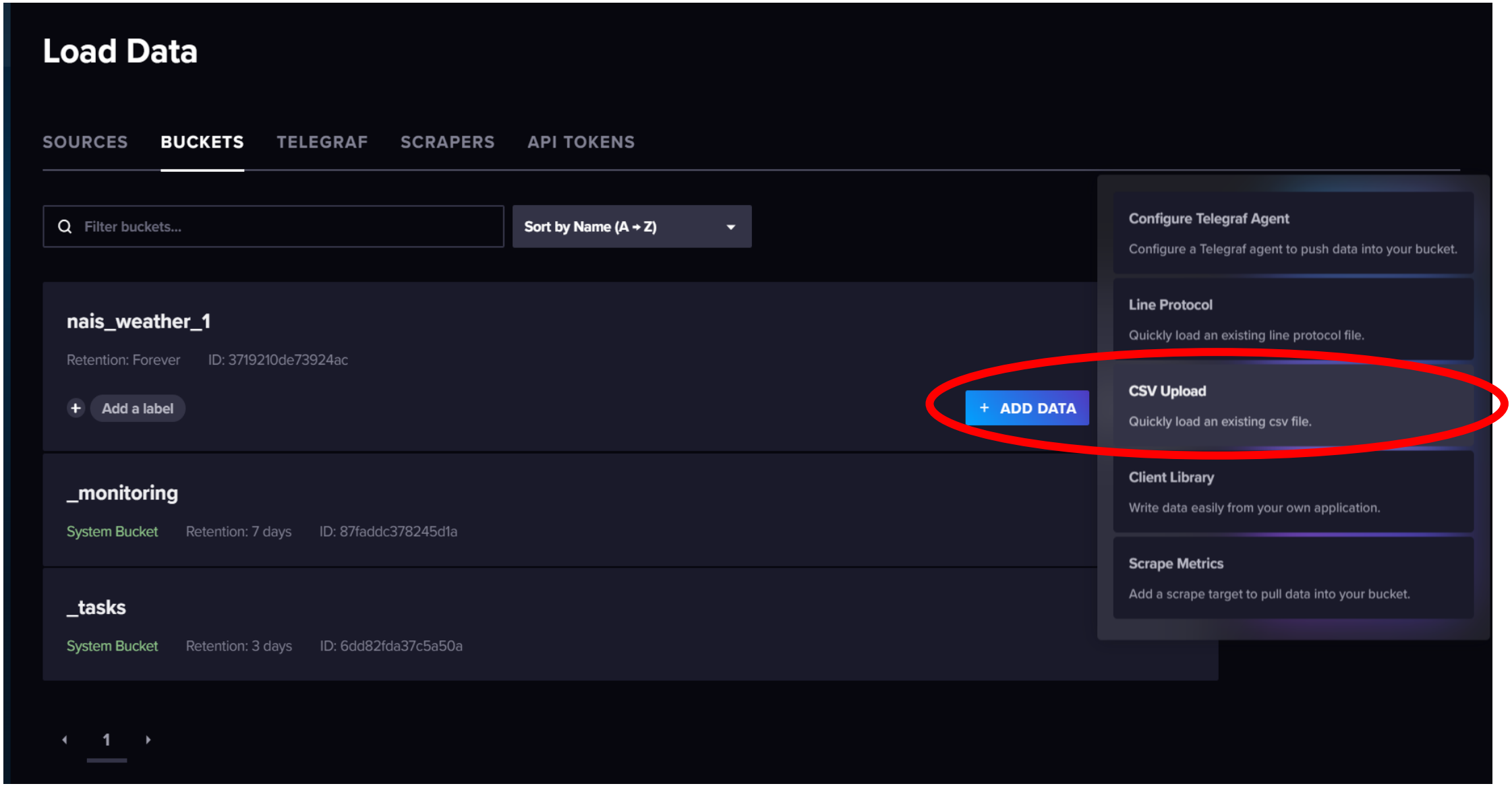
time	City	_field	_value
2024-03-01T12:00:00Z	London	temperature	12,0
2024-03-01T12:05:00Z	London	temperature	+0,1

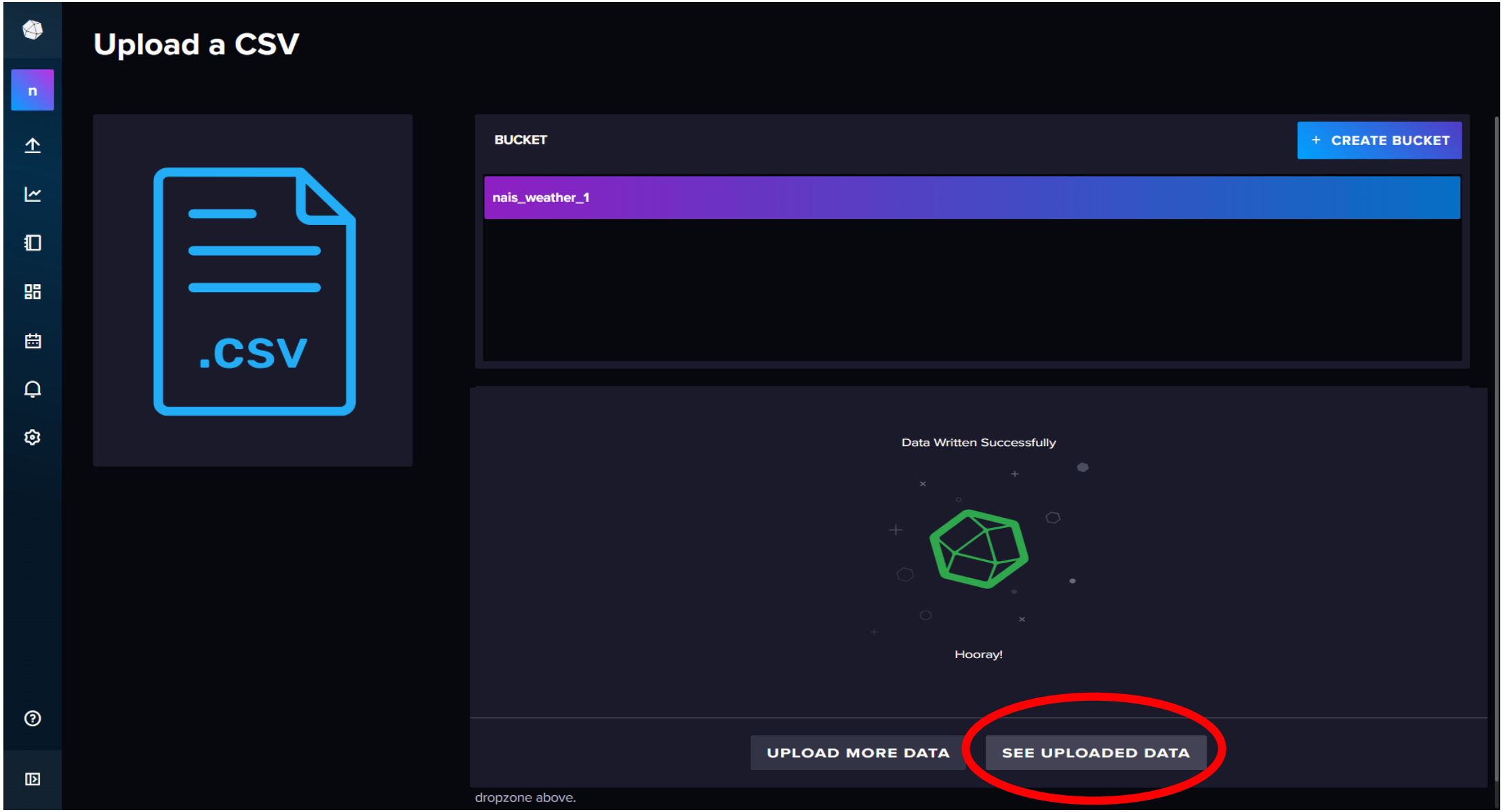
time	City	_field	_value
2024-03-01T12:00:00Z	London	co2	0,65
2024-03-01T12:05:00Z	London	co2	-0,05

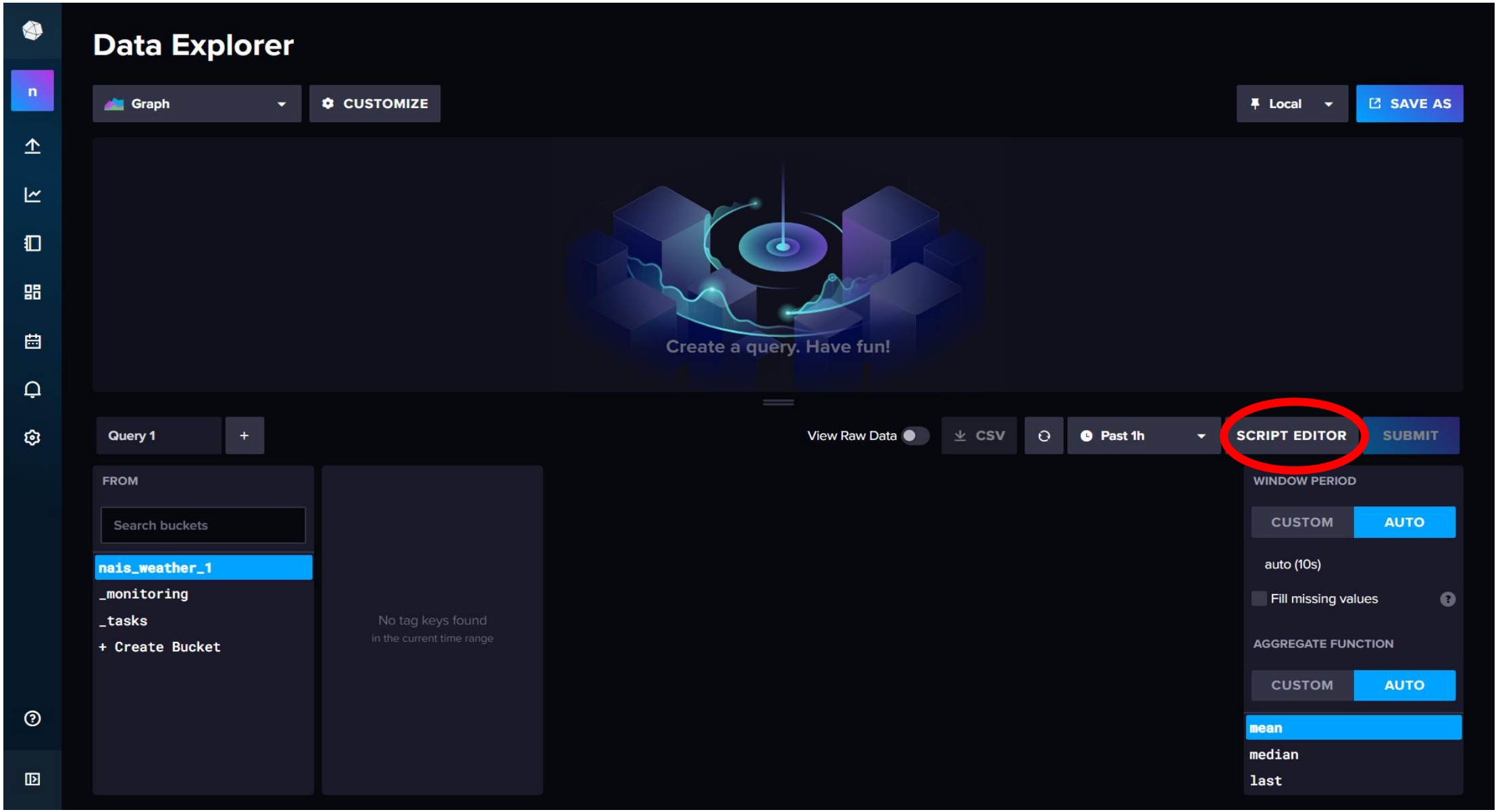
TSM

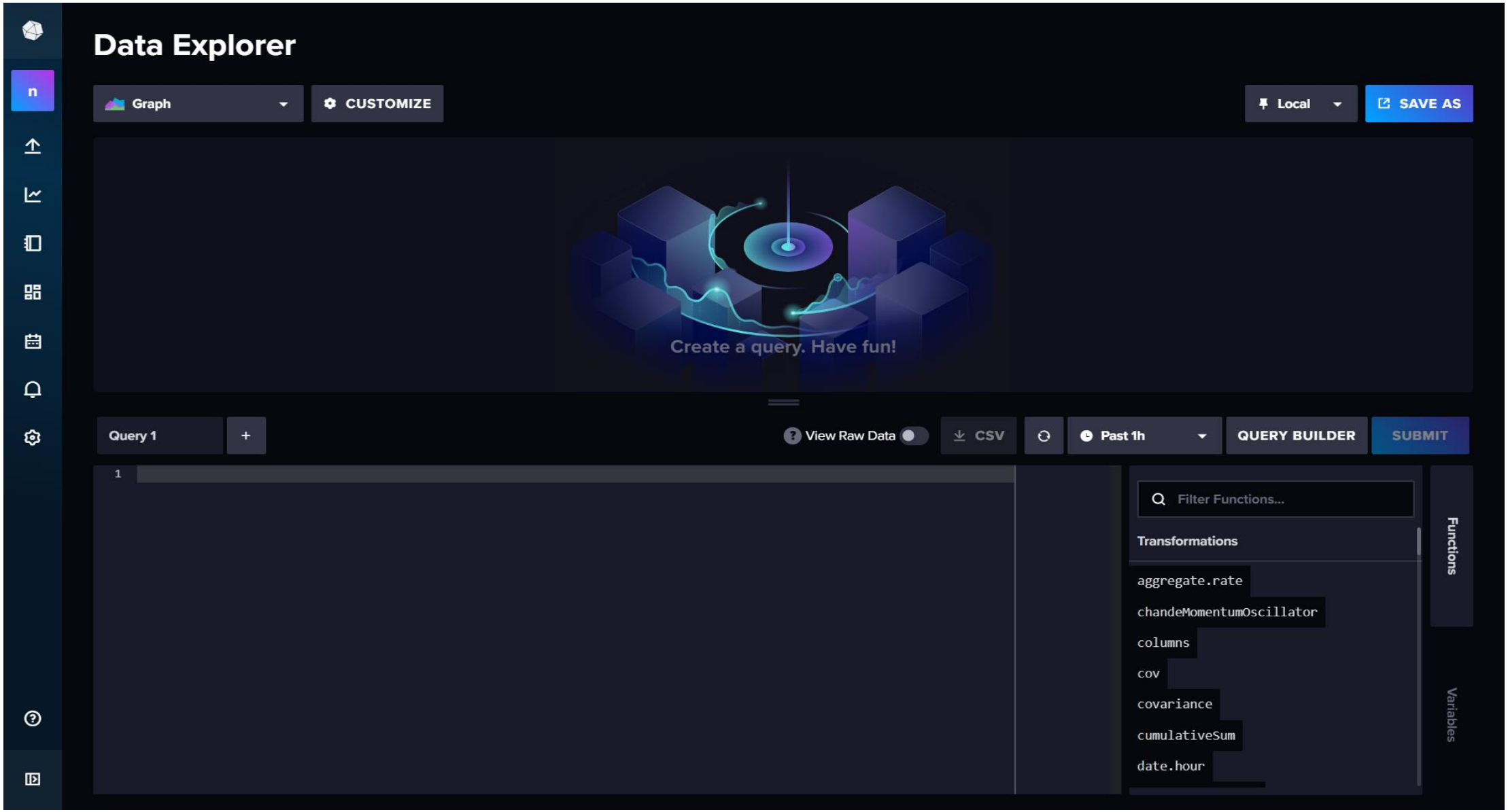

Welcome
Initial User Setup
Complete
Setup Initial User
You will be able to create additional Users, Buckets and Organizations later
Username
nais
Password
Confirm Password
Initial Organization Name
nais_org
Initial Bucket Name
nais_weather_1
CONTINUE


Get Started
Sources
Buckets
Telegraf
Scrapers
API Tokens
query data using the programming language of your choice
Node.js
Go
Arduino
ARDUINO
View more
InfluxDB CLI
Write and query data using the InfluxDB Command Line Interface. Supports CSV and Line Protocol.
Server Agent (Telegraf)
Easily collect and write data using custom stand-alone agent plugins
Search Documentation
Press CTRL + M on any page to search
USEFUL LINKS
InfluxDB University
Get Started with Flux
Explore Metrics
Build a Dashboard
Write a Task
Report a bug
Community Forum
Feature Requests
InfluxDB v2.7.6
Server: 3c58c06
Frontend: 9d25a2f
localhost:8086/orgs/cc4cad94448428a7/load-data/buckets


Load Data
SOURCES
BUCKETS
TELEGRAF
SCRAPERS
API TOKENS
Filter buckets...
Sort by Name (A → Z)
nais_weather_1
Retention: Forever
ID: 3719210de73924ac
Add a label
+ ADD DATA
_monitoring
System Bucket
Retention: 7 days
ID: 87faddc378245d1a
_tasks
System Bucket
Retention: 3 days
ID: 6dd82fda37c5a50a
1
Configure Telegraf Agent
Configure a Telegraf agent to push data into your bucket.
Line Protocol
Quickly load an existing line protocol file.
CSV Upload
Quickly load an existing csv file.
Client Library
Write data easily from your own application.
Scrape Metrics
Add a scrape target to pull data into your bucket.


Upload a CSV
.CSV
BUCKET
+ CREATE BUCKET
nais_weather_1
Data Written Successfully
Hooray!
UPLOAD MORE DATA
SEE UPLOADED DATA
dropzone above.


Data Explorer
Graph
CUSTOMIZE
Local
SAVE AS
Create a query. Have fun!
Query 1
View Raw Data
CSV
Past 1h
SCRIPT EDITOR
SUBMIT
FROM
Search buckets
nais_weather_1
_monitoring
_tasks
+ Create Bucket
No tag keys found
in the current time range
WINDOW PERIOD
CUSTOM
AUTO
auto (10s)
Fill missing values
AGGREGATE FUNCTION
CUSTOM
AUTO
mean
median
last


Data Explorer
Graph
CUSTOMIZE
Local
SAVE AS
Create a query. Have fun!
Query 1
View Raw Data
CSV
Past 1h
QUERY BUILDER
SUBMIT
1
Filter Functions...
Transformations
aggregate.rate
chandeMomentumOscillator
columns
cov
covariance
cumulativeSum
date.hour
Functions
Variables

Sadržaj

Primer upita

```
1 from(bucket: "naziv_baketa") // Izvor podataka
2 |> range([start: početak] [stop: kraj]) // Filtriranje vremena
3 [|> filter(fn: (r) => (r.<naziv_kolone> <relacioni_operator> vrednost)] //
  Filtriranje po nazivu kolone – koristi se predikatska funkcija
4 [|> group(columns: ["<naziv_kolone_1> [, "<naziv_kolone_2> …"]])] //
  oblikovanje podataka
5 [|> mean()] // procesiranje podataka
```

Filtiranje podataka **Operator** koji omogućava prenos rezulatat na sledecu liniju upita

- ***range***(): Filtrira podatke na osnovu vremena
  - Moguće je koristiti start i stop parametre koji označavaju početak vremenskog niza, odnosno kraj
- ***filter***(): Filtrira podatke na osnovu vrednosti kolona
  - Funkcija filter() koristi predikatsku funkciju definisanu u parametru *fn* kako bi filtrirala ulazne redove
  - Svaki red se prosleđuje u predikatsku funkciju kao zapis *r*, koji sadrži parove ključ-vrednost za svaku kolonu u tački

Napomena: Termin **kolona** obuhvata sve grupe podataka (merenja, oznake, polja i vremenske oznake), kao i dodatna polja koja mogu nastati kao rezultat upita

Primer upita

```
1 from(bucket: "naziv_baketa") // Izvor podataka
2 |> range(start: početak [stop: kraj]) // Filtriranje vremena
3 [|> filter(fn: (r) => (r.<naziv_kolone> <relacioni_operator> vrednost)] //
  Filtriranje po nazivu kolone – koristi se predikatska funkcija
4 [|> group(columns: ["<naziv_kolone_1> [, "<naziv_kolone_2> …"]])] //
  oblikovanje podataka
5 [|> mean()] // procesiranje podataka
```

- ***Oblikovanje podataka***
  - ***group***(): modifikuje ključeve grupa
  - ***window***(): modifikuje vrednosti _start i _stop tačaka kako bi grupisao podatke po vremenu
  - ***pivot***(): rotira kolone u redove
  - ***drop***(): uklanja određene kolone
  - ***keep***(): zadržava određene kolone i uklanja sve ostale

Primer upita

```
1 from(bucket: "naziv_baketa") // Izvor podataka
2 |> range(start: početak [stop: kraj]) // Filtriranje vremena
3 [|> filter(fn: (r) => (r.<naziv_kolone> <relacioni_operator> vrednost)] //
  Filtriranje po nazivu kolone – koristi se predikatska funkcija
4 [|> group(columns: ["<naziv_kolone_1> [, "<naziv_kolone_2> …"]])] //
  oblikovanje podataka
5 [|> mean()] // procesiranje podataka
```

- ***Procesiranje podataka***
  - **Agregacija podataka**: Agregira sve redove ulazne tabele u jedan red
  - **Selektovanje specifičnih redova**: Vraćanje određenih redova iz svake ulazne tabele
    - Npr. vraćanje prvog ili poslednjeg reda, ili reda sa najvišom ili najnižom vrednošću
  - **Transformacija redova**: Koristi se *map()* funkcija kako bi bio prepravljen svaki ulazni red
    - Npr. transformisanje vrednosti matematičkim operacijama, obrada stringova ili dinamičko dodavanje nove kolone
  - **Slanje obaveštenja**: Evaluacija podataka i korišćenje funkcije *Flux* za slanje obaveštenja eksternim servisima

Filtriranje podataka – Primer

- Prikažite podatke koji su zabeleženi u vremenskom opsegu od „2024-03-25T14:11:30Z“ do „2024-03-30T14:11:30Z“. Ovi podaci treba da su isključivo u vezi sa merenjima koja su identifikovana kao *'airSensors*', a koja su izvršena za merenje vlažnosti vazduha (*humidity*). Dodatno, podaci treba da budu dobijeni sa senzora čiji je identifikator 'TLM0100'.

```
1  from(bucket: "nais_weather_1") // Izvor podataka
2  |> range(start: 2024-03-25T14:11:30Z, stop: 2024-03-30T14:11:30Z) // Filtriranje
   vremena
3  |> filter(fn: (r) => r["_measurement"] == "airSensors")
4  |> filter(fn: (r) => r["_field"] == "humidity")
5  |> filter(fn: (r) => r["sensor_id"] == "TLM0100")
```

Napomena: Pre početka rada na upitu kreirati baket sa nazivom nais_weather_1 i koristii modified_air-sensor-1.csv kao izvor podataka

Filtriranje podataka – Zadatak

- Prikažite podatke koji su zabeleženi u vremenskom opsegu od „2024-04-01T00:00:00Z“ do „2024-04-10T00:00:00Z“. Ovi podaci treba da su isključivo u vezi sa merenjima (*_measurement*) koja su identifikovana kao '*airSensors*', a koja su izvršena za merenje temperature (*_field*) koja je iznad vrednosti (*_value*) od 75,5 Farenhajta.

```
1 from(bucket: "nais_weather_1")
2 |> range(start: 2024-04-01T00:00:00Z, stop: 2024-04-10T00:00:00Z)
3 |> filter(fn: (r) => r["_measurement"] == "airSensors")
4 |> filter(fn: (r) => r["_field"] == "temperature")
5 |> filter(fn: (r) => r["_value"] > 75.5)
```

Oblikovanje podataka i agregacija – Primer

- Prikažite podatke o vremenskim merenjima između "2024-04-01T00:00:00Z" i "2024-04-10T00:00:00Z" iz baketa "*nais_weather_1*". Ovi podaci treba da su grupisani po identifikatoru senzora (*sensor_id*) i tipu merenja (*_field*), a za svaku grupu izračunata je prosečna vrednost merenja.

```
1  from(bucket: "nais_weather_1")
2  |> range(start: 2024-04-01T00:00:00Z, stop: 2024-04-10T00:00:00Z )
3  |> group(columns: ["sensor_id", "_field"])
4  |> mean(column: "_value")
```

Oblikovanje podataka i agregacija – Primer

- Prikažite maksimalne vrednosti merenja između "2024-04-09T00:00:00Z" i "2024-04-10T00:00:00Z" iz baketa "*nais_weather_1*", podeljene po svakom satu. Ovi podaci pružaju informacije o najvišim vrednostima merenja tokom svakog sata u datom vremenskom intervalu. U izlaznom rezultatu ostaviti polja koja sadrže informacije o senzorima (*sensor_id*), vrsti merenja (*_field*) i vrednosti (*_value*).

```
1  from(bucket: "nais_weather_1")
2  |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z )
3  |> window(every: 1h)
4  |> max(column: "_value")
5  |> keep(columns: ["sensor_id", "_field", "_value"])
```

```
1  from(bucket: "nais_weather_1")
2  |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z )
3  |> aggregateWindow(every: 1h, fn:max)
4  |> keep(columns: ["sensor_id", "_field", "_value"])
```

aggregateWindow – omogućava kombinaciju window funkcije i bilo koje druge funckije agregacije

Oblikovanje podataka i agregacija – Primer

- Prikažite podatke o merenjima između "2024-04-09T00:00:00Z" i "2024-04-10T00:00:00Z" iz baketa "*nais_weather_1*", organizovane po vremenu (*_time*) i tipu merenja (*_field*), dok su vrednosti raspoređene prema različitim senzorima (*sensor_id*).

```
1 from(bucket: "nais_weather_1")
2 |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z )
3 |> pivot(rowKey: ["_time", "_field"], columnKey: ["sensor_id"], valueColumn:
  "_value")
```

Oblikovanje podataka i agregacija – Zadatak

- Prikažite prosečne vrednosti merenja između 2024-04-09T00:00:00Z i 2024-04-10T00:00:00Z iz baketa "*nais_weather_1*", gde su merenja izvršena pomoću senzora sa identifikatorima "TLM0100" ili "TLM0101". Podaci su grupisani po tipu merenja (*_field*) i identifikatoru senzora (*sensor_id*), a zatim predstavljeni u intervalima po satu kako bi se izračunala prosečna vrednost merenja za svaki sat. U izlaznom rezultatu zadržati polja koja sadrže informacije o senzorima (*sensor_id*), vrsti merenja (*_field*) i vrednosti (*_value*).

```
1 from(bucket: "nais_weather_1")
2 |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z )
3 |> filter(fn: (r) => r["sensor_id"] == "TLM0100" or r["sensor_id"] ==
  "TLM0101")
4 |> group(columns: ["_field", "sensor_id"])
5 |> window(every: 1h)
6 |> mean(column: "_value")
7 |> keep(columns: ["sensor_id", "_field", "_value"])
```

Selektovanje određenih redova – Primer

- Prikažite samo poslednje merenje temperature za svaki sensor između '2024-04-09T00:00:00Z' i '2024-04-10T00:00:00Z' iz baketa '*nais_weather_1*'.

```
1 from(bucket: "nais_weather_1")
2 |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z)
3 |> filter(fn: (r) => r["_field"] == "temperature")
4 |> last()
```

Procesiranje podataka *map*() – Primer

- Prikažite samo merenja temperature između '2024-04-09T00:00:00Z' i '2024-04-10T00:00:00Z' iz baketa '*nais_weather_1*'. Zatim, konvertujte dobijene temperature iz Fahrenheit u Celsius i prikažite rezultate sa novim atributom '*temperature_celsius*'.

```
1  from(bucket: "nais_weather_1")
2  |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z)
3  |> filter(fn: (r) => r["_field"] == "temperature")
4  |> map(fn: (r) => ({
5    r with temperature_celsius:
6    if r._value != 0
7    then (r._value - 32.0) * (5.0/9.0)
8    else 0.0
     }))
```

Procesiranje podataka *map*() – Zadatak

- Prikažite samo merenja temperature između '2024-04-10T18:00:00Z' i '2024-04-10T19:00:00Z' iz baketa '*nais_weather_1*'. Nakon toga, dodajte novi atribut '*location*' koji ima vrednost „*Indoor*“ ukoliko su u pitanju senzori TLM0100, TLM0101 i TLM0102, a za ostale senzore ima vrednost „*Outdoor*“.

```
1  from(bucket: "nais_weather_1")
2  |> range(start: 2024-04-10T18:00:00Z, stop: 2024-04-10T19:00:00Z)
3  |> filter(fn: (r) => r["_field"] == "temperature")
4  |> map(fn: (r) => ({
5    r with location:
6    if r.sensor_id == "TLM0100" or r.sensor_id == "TLM0101" or r.sensor_id ==
   "TLM0102"
7    then "Indoor"
8    else "Outdoor"
9    }))
```

Spajanje baketa *join*() – Primer

- Definišite dva skupa podataka, s1 i s2, koji prikazuju poslednje merenje koncentracije gasa CO za svaki senzor iz dva različita baketa ('*nais_weather_1*' i '*nais_weather_2*') u periodu od '2024-04-09T00:00:00Z' do '2024-04-10T00:00:00Z'. Nakon toga, spojite ova dva skupa podataka na osnovu identifikatora senzora. Za svaki senzor, izračunajte odstupanje između poslednjih merenja koncentracije CO iz s1 i s2, a te rezultate prikažite u novoj tabeli sa dodatom kolonom *deviation* koja sadrži izračunato odstupanje

Napomena: Kako bi bio kreiran upit koji spaja dva baketa potrebno je ispratiti korake pri kreiranju prvog baketa i napraviti novi baket „*nais_weather_2*“ i dodati sledeće podatke „*modified_air-sensor-data_2*“.

Spajanje baketa *join*() – Primer

```
 1  s1 = from(bucket: "nais_weather_1")
 2  |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z)
 3  |> filter(fn: (r) => r._field == "co")
 4  |> last()
 5  |> keep(columns: ["_value", "sensor_id"])
 6
 7  s2 = from(bucket: "nais_weather_2")
 8  |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z)
 9  |> filter(fn: (r) => r._field == "co")
10  |> last()
11  |> keep(columns: ["_value", "sensor_id"])
12
13  join(tables: {setup1: s1, setup2: s2}, on: ["sensor_id"])
14  |> map(fn: (r) => ({r with deviation: r._value_setup1 - r._value_setup2}))
```

Spajanje baketa *join*() – Zadatak

- Imate dva baketa podataka, *nais_weather_1* i *nais_weather_2*, koji sadrže podatke o vlažnosti vazduha za isti vremenski period. Potrebno je izračunati prosečnu vlažnost vazduha za svaki senzor iz oba baketa u prozoru od jednog sata i prikazati razliku između prosečnih vlažnosti vazduha za svaki senzor. Koristite vremenski period od '2024-04-09T00:00:00Z' do '2024-04-10T00:00:00Z'. Rezultate prikažite u obliku tabele sa informacijama o senzoru, vremenu merenja, prosečnoj vlažnosti vazduha iz nais_weather, prosečnoj vlažnosti vazduha iz nais_weather_2 i razlici između prosečne vlažnosti vazduha za svaki senzor.

Spajanje baketa *join*() – Rešenje

```
 1 // Definisanje skupa podataka za merenja vlažnosti vazduha iz prvog baketa
 2 humidity_nais_weather_1 = from(bucket: "nais_weather_1")
 3 |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z)
 4 |> filter(fn: (r) => r._field == "humidity")
 5 |> group(columns: ["sensor_id"])
 6 |> aggregateWindow(every: 1h, fn: mean)
 7 |> keep(columns: ["_time", "_value", "sensor_id"])
 8
 9 // Definisanje skupa podataka za merenja vlažnosti vazduha iz drugog baketa
10 humidity_nais_weather_2 = from(bucket: "nais_weather_2")
11 |> range(start: 2024-04-09T00:00:00Z, stop: 2024-04-10T00:00:00Z)
12 |> filter(fn: (r) => r._field== "humidity")
13 |> group(columns: ["sensor_id"])
14 |> aggregateWindow(every: 1h, fn: mean, createEmpty: false)
15 |> keep(columns: ["_time", "_value", "sensor_id"])
16
```

```
17 // Spajanje podataka oba baketa na osnovu vremena merenja i senzora
18 joined_humidity_data = join(tables: {humidity_nais_weather_1:
   humidity_nais_weather_1, humidity_nais_weather_2: humidity_nais_weather_2},
   on: ["sensor_id", "_time"])
19 // Prosečne vlažnosti vazduha za svaki senzor iz oba baketa
20 average_humidity = joined_humidity_data
21 |> map(fn: (r) => ({ r with
22   average_humidity_nais_weather_1: r._value_humidity_nais_weather_1,
23   average_humidity_nais_weather_2: r._value_humidity_nais_weather_2
24   }))
25 //Razlike između prosečne vlažnosti vazduha iz oba baketa za svaki senzor
26 deviation_data = average_humidity
27 |> map(fn: (r) => ({ r with
28   deviation: r.average_humidity_nais_weather_1 -
   r.average_humidity_nais_weather_2
29   }))
30 deviation_data
31 |> keep(columns: ["sensor_id", "_time", "average_humidity_nais_weather_1",
32 "average_humidity_nais_weather_2", "deviation"])
```

Sadržaj

Primeri upotrebe *InfluxDB* u svetu finansija

- **Analiza trgovanja:**
  - Pohrana podataka o cenama hartija od vrednosti, obimu trgovanja i vremenskim pečatima
  - Brza obrada vremenskih serija podataka omogućava identifikaciju trendova i analizu obrazaca
  - Omogućava investitorima da donose informisane odluke o investicijama na osnovu analize tržišta
- **Obrada transakcija:**
  - Neprekidno praćenje i upravljanje finansijskim transakcijama i tokovima novca
  - Efikasno beleženje podataka o transakcijama u realnom vremenu
  - Omogućava institucijama da identifikuju i reaguju na potencijalne pretnje ili nepravilnosti u tokovima novca
- **Izveštavanje o usklađenosti:**
  - Skladištenje podataka o transakcijama, korisnicima i tržištima u skladu sa regulatornim zahtevima
  - Generisanje tačnih i potpunih izveštaja koji zadovoljavaju regulatorne standarde
  - Omogućava finansijskim institucijama da ispunjavaju regulatorne obaveze vezane za izveštavanje i usklađenost

Analiza dnevnih promena cene Bitcoin-a

- Definišite skup podataka koji prikazuje istorijske cene Bitcoina denominovane u evrima (EUR) iz bucket-a 'bitcoin_historical' u periodu od '2024-03-23T09:33:51Z'. Zatim, izračunajte prosečnu dnevnu promenu cene Bitcoina na osnovu podataka sa CoinDesk-a.

Napomena: Kako bi bio kreiran ovaj potrebno je ispratiti korake pri kreiranju prvog baketa i napraviti novi baket „bitcoin_historical“ i dodati sledeće podatke „ bitcoin-historical.csv“.

Analiza dnevnih promena cene Bitcoin-a

```
// Učitaj podatke o ceni Bitcoina
bitcoin_cene = from(bucket: "bitcoin_historical")
|> range(start: 2024-03-23T09:33:51Z) // Podesi vremenski opseg po potrebi
|> filter(fn: (r) => r["_measurement"] == "coindesk" and r["code"] == "EUR")
|> aggregateWindow(every: 1d, fn: mean)

//Izračunaj dnevne promene
bitcoin_promene = bitcoin_cene
|> derivative(unit: 1d, columns: ["_value"])
|> map(fn: (r) => ({ r with dnevni_prosek: r._value }))
|> drop(columns: ["_value"])

//Izračunaj prosečnu dnevnu promenu
prosecna_promena = bitcoin_promene
|> mean(column: "dnevni_prosek")

// Prikaži rezultat
prosecna_promena
```

Identifikacija ekstremnih promena cena *Bitcoina* – Zadatak

- Imamo baket podataka *bitcoin_historical* koji sadrži informacije o cenama *Bitcoina* za poslednjih 30 dana. Vaš zadatak je identifikovati ekstremne promene cena *Bitcoina* tokom ovog perioda. Zadaci:
  - **Učitavanje podataka**: Učitajte podatke o cenama *Bitcoina* iz baketa "*bitcoin_historical*".
  - **Filtriranje podataka**: Filtrirajte podatke kako biste uzeli u obzir samo merenja sa "*coindesk*" merenjem i "*USD*" kodom valute.
  - **Izračunavanje ekstrema**: Koristite upite *Flux* za izračunavanje ekstremnih promena cena *Bitcoina* tokom poslednjih 30 dana. Definišite ekstremne promene kao one koje prelaze 10% promene u odnosu na prethodnu vrednost.
  - **Vizualizacija rezultata**: Kreirajte grafikone ili dijagrame koji jasno prikazuju identifikovane ekstremne promene cena *Bitcoina*.

Sadržaj

Korisni linkovi

- **InfluxDB – zvanična dokumentacija**
  - *https://docs.influxdata.com/*

Napredne arhitekture informacionih sistema

Baze podataka vremenskih nizova

Pitanja?

Izvođači nastave:
dr Marko Vještica
Elena Akik
Sanja Radić

DAIS
Data Science and Information Systems Group