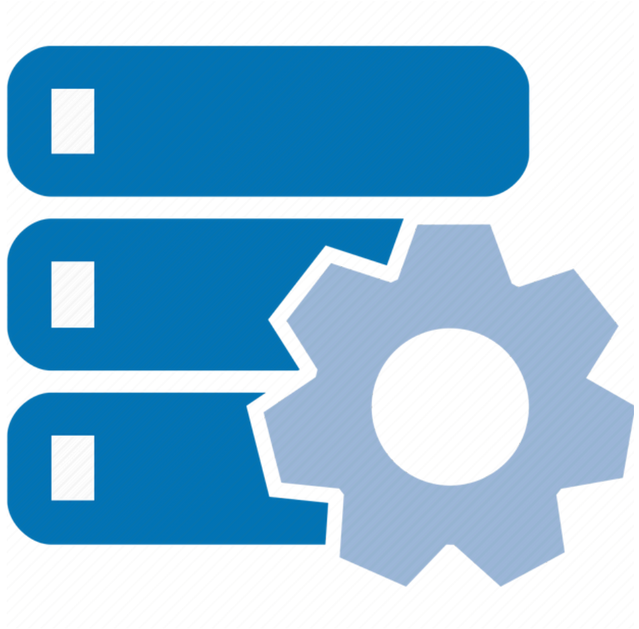
Administriranje baza podataka

Time-series baza podataka



Smerovi: SRT, KOT

Predmetni profesor: Dr Dušan Stefanović, prof. struk. studija

Predmetni asistenti: Nevena Minić, Nikola Vukotić

ODSEK NIŠ
AKADEMIJA
TEHNIČKO-VASPITAČKIH
STRUKOVNIH STUDIJA

Cilj vežbe

1. **Razumevanje koncepta vremenskih serija i modelovanja podataka:** Upoznavanje sa osnovnim principima vremenskih serija, strukturiranjem podataka po *timestamp*-u, i modeliranjem podataka koji se prikupljaju kontinuirano kroz vreme.
2. **Kreiranje i upravljanje time-series bazama i merenjima:** Razumevanje kako kreirati time-series bazu podataka, organizovati podatke kroz merenja (measurements), tagove i vrednosti (fields), kao i osnovne operacije sa vremenskim podacima.
3. **Razumevanje prednosti i kada koristiti time-series baze podataka:** Upoznavanje sa slučajevima kada je time-series baza pogodnija od klasičnih relacionih baza, posebno za aplikacije koje zahtevaju praćenje događaja kroz vreme (npr. senzori, server logovi, metrika performansi), kao i razumevanje optimizacija koje nude za skladištenje i analizu vremenskih podataka.

Vremenske serije podataka

- Promenljiva čije se vrednosti mere u određenim tačkama u vremenu se naziva **vremenskom serijom podataka (time series)**.
- Vremenski intervali između tačaka su često jednaki.
- Vremenske serije podataka imaju trostruku namenu:
  - **Razumevanje** pojava i struktura koje su proizvele određenu vremensku seriju podataka
  - **Nadgledanje (monitoring)** tih istih pojava i struktura
  - **Predviđanje** budućih vrednosti

- Široku praktičnu primenu vremenske serije podataka imaju u oblastima kao što su: ekonomska prognoza, vremenska prognoza, analiza budžeta, kontrola procesa i kvaliteta...
- U prirodi postoje razne veličine i pojave čije se vrednosti mogu meriti u vremenu u određenim vremenskim intervalima. Primeri: temperatura, količina padavina, prinos neke biljke…
- U računarstvu i informatici, vremenske serije podataka se često koriste za praćenje određenih performansi sistema, kako softvera, tako i hardvera.
- Beleže se i skladište podaci iz različitih izvora kako bi se stekla slika o performansama sistema i blagovremeno primetili i otklonili potencijalni problemi u radu.
- Takođe, primenjuju se i u oblastima kao što su mašinsko učenje i data mining.
- Sa napretkom različitih tehnologija gde je potrebno raditi sa vremenskim serijama, npr. „Internet Stvari“ (IoT – *Internet of Things*), količina podataka je sve veća i veća (*big data*) tako da primena vremenskih serija u računarstvu i informatici postaje sve veći izazov za koji je potrebno razvijati posebne alate i servise.

Time-series baze podataka

- **Time series databases (TSDB)** - baze podataka za rad sa vremenskim serijama podataka
- To su baze podataka koje su posebno optimizovane i namenjene za rad sa vremenskim serijama i često imaju ugrađene u sebe određene metode za analizu serija, kao i druge pogodnosti.
- Sama arhitektura različitih TSDB se razlikuje od implementacije do implementacije.
- Neke koriste NoSQL pristup, dok se druge baziraju na korišćenju tehnologije relacionih baza podataka u pozadini.
- Neke koriste već postojeće DBMS-ove i dodaju sloj funkcionalnosti za rad sa vremenskim serijama, dok druge koriste svoja rešenja i razvijaju i svoj upitni jezik (*query language*).

- Postoje dva fundamentalna zahteva koje baza podataka rad sa vremenskim serijama podataka mora da zadovolji:
  - **Ko-lokacija podataka**
  - **Efikasan pristup podacima u proizvoljnom intervalu**
- Sekvencijalni pristup podacima je i dalje dosta brži od nasumičnog pristupa podacima, čak i sa SSD-ovima (*solid-state drive*).
- Sistem koji smešta sortirane podatke u sekvencijalnom redosledu (ko-lokacija) na istim mestima, što ide prirodno uz vremenske serije podataka, će učiniti te podatke dostupnim za brzo i efikasno čitanje.
- Forsiraju se CR- a ne CRUD operacije

Time-series baze podataka

- **Performanse i skalabilnost**
  - Dobro projektovana TDSB omogućava lako skaliranje sistema tako da on može da podrži i milione tačaka u vremenskim serijama u kontinualnom toku podataka i da izvrši analize u realnom vremenu.
- **Kompakcija podataka (*data compaction*)**
  - Kako podaci neke vremenske serije podataka stare, to su manje bitne individualne tačke.
  - Gotovo svaka TSDB ima određene mehanizme za degradiranje i združavanje podataka.
  - To znači da se uglavnom visoko precizni podaci kratko čuvaju u bazi (npr. vrednost neke promenljive u intervalima od jedne sekunde), tim podacima se konstantno smanjuje rezolucija (*downsample*) i raznim matematičkim funkcija agregacije (*aggregate*) se objedinjuju u tačke sa većim periodima (npr. sada je vremenska serija aproksimirana sa tačkama između kojih je interval od jednog minuta
- **Niži troškovi**
  - Zbog optimizovanosti za rad sa vremenskim serijama, TSDB direktno utiču na smanjenje troškova zato što troše manje računarskih resursa pri radu sa vremenskim serijama podataka.
- **Bolje poslovne odluke**
  - Time što se podaci mogu analizirati u realnom vremenu, organizacije mogu da prave bolje i preciznije odluke, kao i da detektuju i reaguju na potencijalne probleme na vreme.

InfluxDB

InfluxDB je open-source **time-series** baza podataka, posebno razvijena za **čuvanje**, **upisivanje** i **analizu** vremenskih serija podataka kao što su metričke vrednosti, logovi i događaji. Umesto tradicionalnih tabela i redova, organizuje podatke oko vremenskih pečata (*timestamps*).

Karakteristike:

- 🛠️ Razvila kompanija **InfluxData**
- 📅 Godina nastanka: **2013**
- 📍 Sedište: San Francisco, SAD
- 🧪 Implementirana korišćenjem **programskog jezika Go**
- 📈 Native **Time Series Database (TSDB)** rešenje
- 🛠️ Deo **TICK stack-a** (*Telegraf, InfluxDB, Chronograf, Kapacitor*) za kompletno upravljanje vremenskim podacima
- 💾 Postoji u **besplatnoj open-source** i **komercijalnoj verziji** (InfluxDB Cloud, Enterprise)
- 📄 **Optimizovana za čitanje i pisanje vremenskih serija,** a ne za klasične CRUD operacije
- ❗ **Ne preporučuje se** za aplikacije koje zahtevaju intenzivno ažuriranje i brisanje podataka, zbog potencijalnog pada performansi
- 🌐 Fokusirana na **brzu akviziciju**, **skladištenje** i **analizu vremenski orijentisanih podataka** (npr. senzori, IoT uređaji, sistemski logovi, metrički monitoring)
- 🌐 Zvanični sajt: https://www.influxdata.com
- 📚 Dokumentacija: https://www.docs.influxdata.com/influxdb

InfluxDB

- **Model podataka:**
  - **Measurement -** Kao “tabela” u tradicionalnim bazama - grupiše slične podatke (npr. temperature)
  - **Tag -** Indeksirano polje - koristi se za filtriranje i grupisanje (npr. ime uređaja, lokacija)
  - **Field -** Neindeksirano polje - stvarna merenja, konkretne vrednosti koje se mere (npr. temperatura, vlažnost)
  - **Timestamp -** Vremenski pečat - obavezan za svaki zapis, označava kada je merenje nastalo

Primer podataka o temperaturi prikupljenih sa senzora

timestamp	measurement	tag_key	tag_value	field_key	field_value
2025-04-26T08:00:00Z	temperature	device	sensor_1	value	22.5
2025-04-26T08:00:00Z	temperature	device	sensor_2	value	23.0
2025-04-26T08:01:00Z	temperature	device	sensor_1	value	22.7

Linijski format - način na koji InfluxDB interno zapisuje podatke

```
temperature,device=sensor_1 value=22.5 1714118400000000000
temperature,device=sensor_2 value=23.0 1714118400000000000
temperature,device=sensor_1 value=22.7 1714118460000000000
```

- **temperature -** measurement
- **device=sensor_1 -** tag key i value
- **value=22.5 -** field key i value
- **1714118400000000000 -** timestamp u nanosekundama, Unix

Zadatak - Monitoring temperature u gradovima

Potrebno je projektovati i kreirati sistem za prikupljanje i analizu podataka o temperaturi sa senzora postavljenih na različitim lokacijama (gradovima). Za svaki zapis čuva se:

- Timestamp (vreme merenja)
- Device ID (senzor koji šalje podatke)
- City (grad u kome je senzor)
- Temperature (merenja u °C)

Primer zapisa podataka za monitoring temperature u gradovima

timestamp	device_id	city	temperature
2025-04-26T08:00:00Z	sensor_1	Belgrade	22.5
2025-04-26T08:00:00Z	sensor_2	Novi Sad	23.0
2025-04-26T09:00:00Z	sensor_1	Belgrade	23.2
2025-04-26T09:00:00Z	sensor_2	Novi Sad	22.8
2025-04-26T10:00:00Z	sensor_3	Niš	24.1

REŠENJE: Korišćenje time-series baze podataka - *InfluxDB:*

- **Svaki podatak ima vremenski pečat (timestamp)**
  - U SQL bazi mora se svaki put ručno definisati kolona timestamp, i često pisati kompleksne WHERE upiti za filtriranje po vremenu:.

```sql
SELECT * FROM measurements WHERE timestamp >= '2025-04-26 00:00:00' AND timestamp <= '2025-04-26 23:59:59';
```

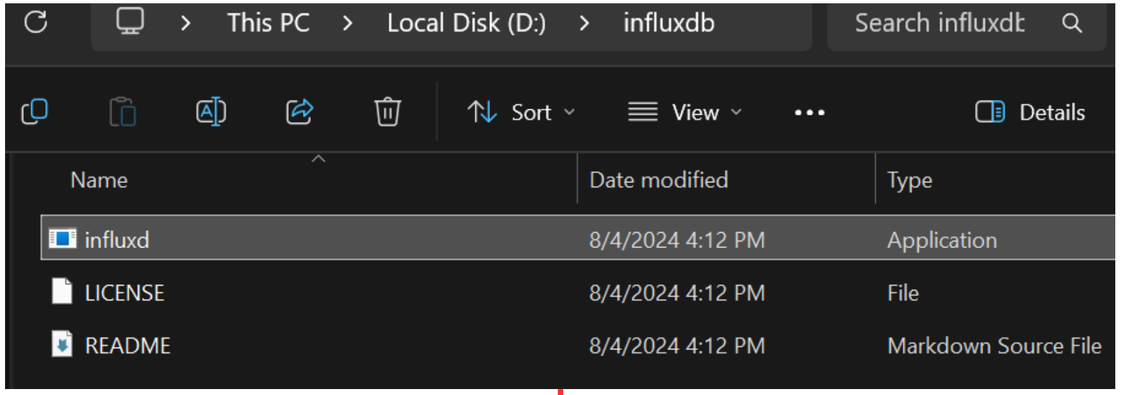
  - U time-series bazama (kao InfluxDB), timestamp je centralni deo podataka, automatski optimizovan za upite kroz vreme.
- **Podaci se stalno upisuju i gotovo nikada ne menjaju**
  - SQL baze (MySQL, PostgreSQL) su optimizovane za često menjanje i ažuriranje podataka (UPDATE, DELETE).
  - Time-series baze kao InfluxDB su optimizovane za brz upis i čuvanje novih vremenskih merenja — bez potrebe za editovanjem.
- **Potrebne su brze agregacije kroz vreme**
  - U SQL bazama mora se koristiti grupisanje (GROUP BY) uz manipulaciju datumima, što može biti sporo na velikim skupovima podataka:

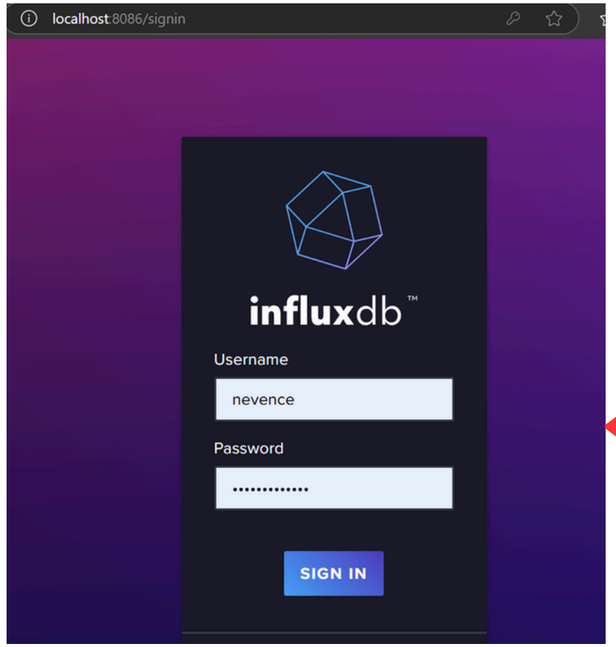
```sql
SELECT DATE(timestamp), AVG(temperature)
FROM measurements
GROUP BY DATE(timestamp);
```

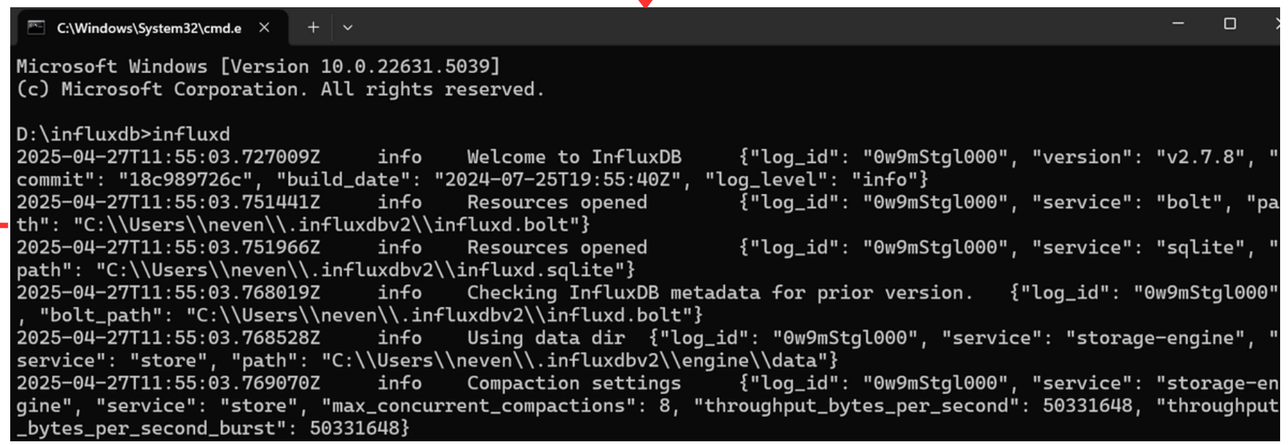
  - Ovo može biti sporo za milionske tabele bez posebnih optimizacija (particionisanje, indeksiranje).
  - InfluxDB već automatski grupiše podatke po vremenskim intervalima (hourly, daily, weekly) vrlo brzo i efikasno.

Zadatak - Monitoring temperature u gradovima

- **Instalacija InfluxDB:** https://docs.influxdata.com/influxdb/v2/install/
- **Pokretanje InfluxDB-a:**
  - Otvoriti folder sa *influxd* izvršnim fajlom, i u njemu otvoriti Command prompt
  - Pokrenuti server komandom: *influxd*
  - Otvoriti web browser i otići na adresu http://localhost:8086 za kreiranje početnog naloga:
    - Username (korisničko ime)
    - Password (lozinka)
    - Organization (naziv organizacije, može biti bilo šta)
    - Bucket (default baza za podatke)



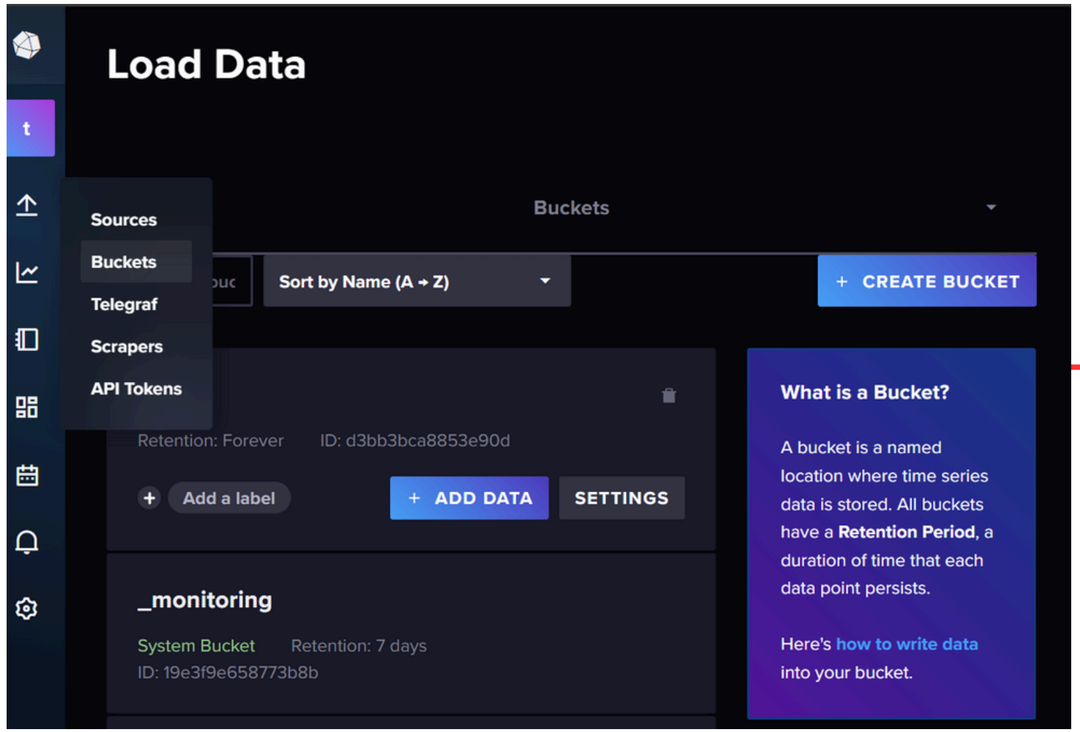


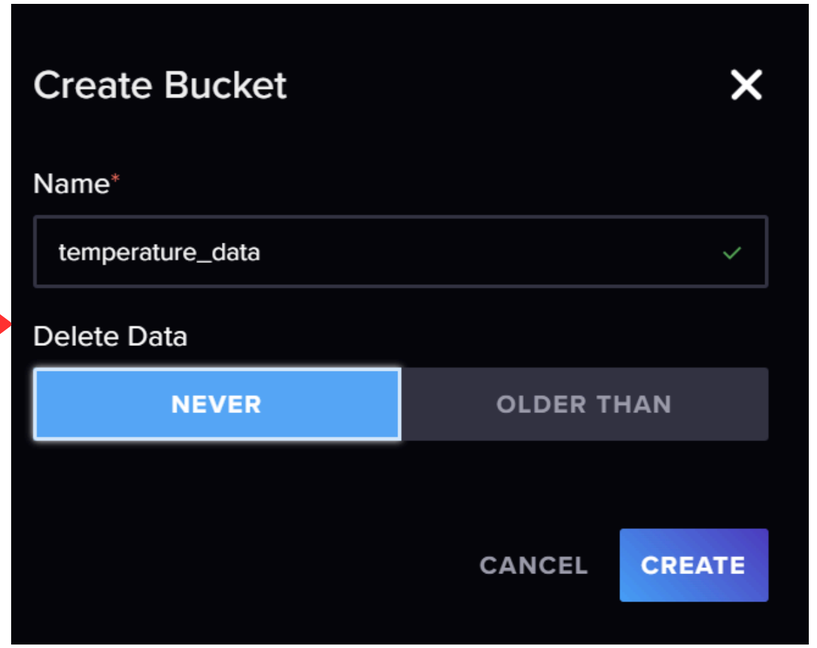


Zadatak - Monitoring temperature u gradovima

1) Kreiranje bucket-a

1. Ulogovati se u InfluxDB na http://localhost:8086.
2. Sa leve strane klik na "Buckets" (kanta ikonice).
3. Klik na "Create Bucket".
4. Uneti:
    a. Name: npr. temperature_data
    b. Retention: može se staviti "Forever" (da se podaci ne brišu automatski) ili neki vremenski period (npr. 30 dana).
5. Klik na "Create".

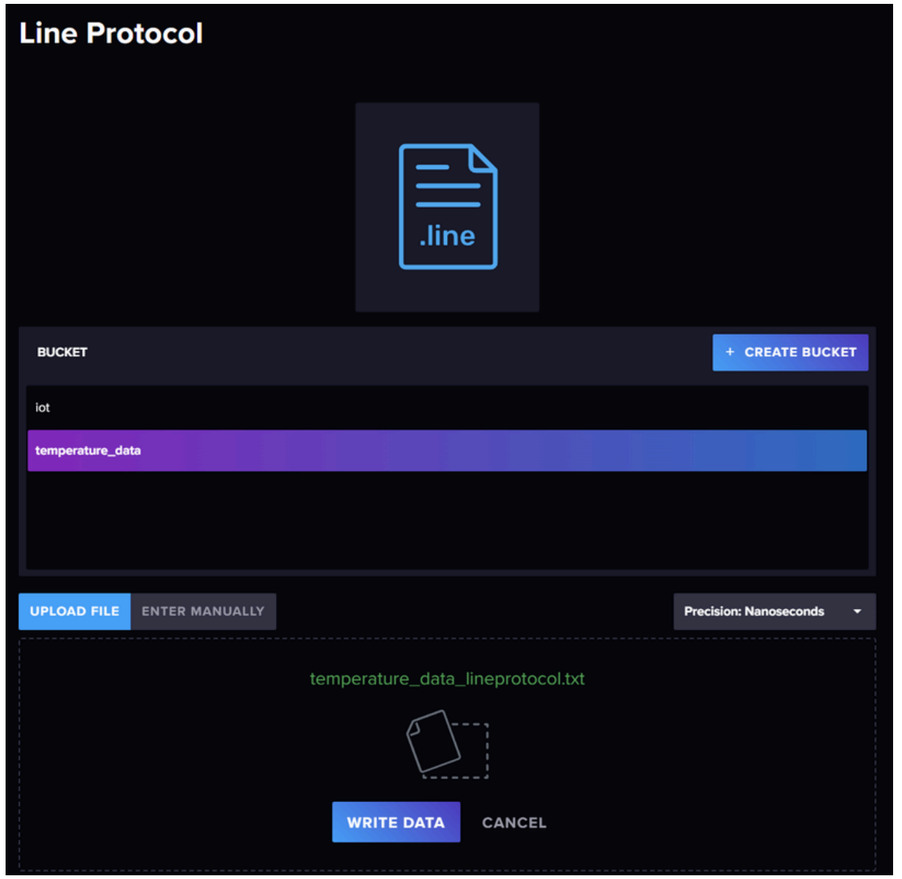




Zadatak - Monitoring temperature u gradovima

2) Import podataka preko UI - Line Protocol

1. Klik na "Data" → "Load Data" → "Buckets".
2. Na bucketu *temperature_data* klik na "Upload Data".
3. Izabrati opciju "Upload Line Protocol".
4. Učitati .txt fajl (temperature_data_lineprotocol.txt):
5. Klik na Continue i Import.



Zadatak - Monitoring temperature u gradovima

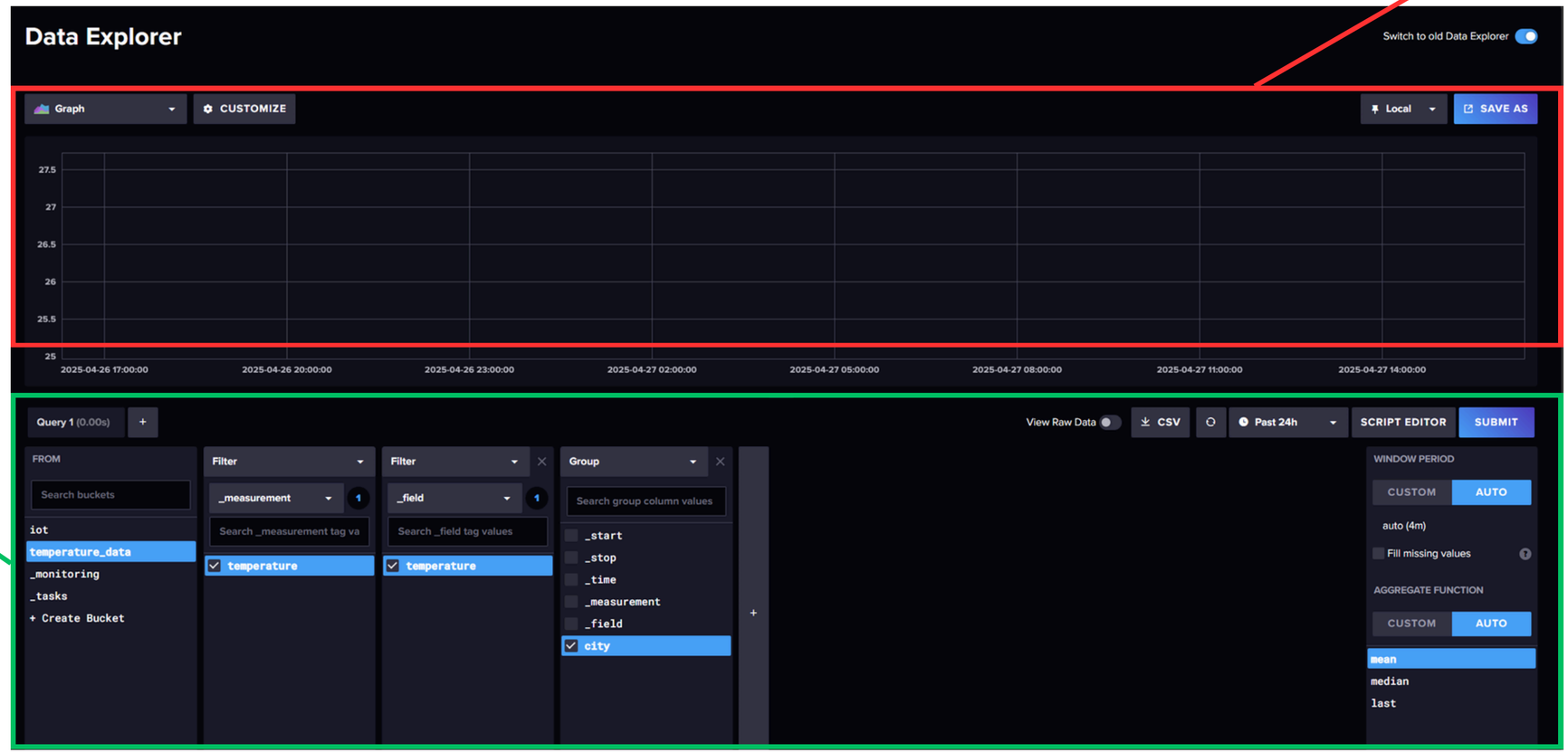
3) Obrada podataka preko Data explorer-a

1. U Data explorer-u nalazi se UI prozor za *Query build (izradu upita)*
2. UI *Query builder* automatski generiše sam upit u *Flux* jeziku.
3. Kreirani upit se može dalje ručno modifikovati po potrebi u *Script Editor-u*
4. Rezultati upita se odmah prikazuju u gornjem delu ekrana, gde se može izabrati prikaz rezultata kao:
    - Graph (grafikon),
    - Table (tabela sa podacima),
    - Simple Table (pojednostavljena tabela bez dodatnih metapodataka).

Na ovaj način moguće je brzo analizirati podatke vizuelno ili tabelarno, zavisno od potreba korisnika.

Prozor za prikaz rezultujućeg upita u željenom formatu - grafikon, tabela...

Query Builder prozor - odabir vremenskog opsega podataka, polja, grupisanje i agregacione funkcije za analizu i obradu podataka.



Zadatak - Monitoring temperature u gradovima

Flux je **funkcionalni jezik** razvijen od strane InfluxData kompanije, specijalno za rad sa **vremenskim serijama podataka** u InfluxDB bazi.

Koristi se za:

- **upite** (čitanje podataka iz baze),
- **obradu** i transformaciju vremenskih serija,
- **agregacije** (prosek, zbir, minimum, maksimum),
- **filterisanje** i **grupisanje** podataka,
- **vizuelizaciju trendova**,
- **integraciju sa spoljnim sistemima** (HTTP API, CSV fajlovi i slično).

Flux zamenjuje stari *SQL-like* jezik u InfluxDB-u jer omogućava fleksibilnije i kompleksnije analize.

Primer Flux upita

```
from(bucket: "temperature_data")
  |> range(start: -24h)
  |> filter(fn: (r) => r["_measurement"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> mean()
```

- Čita podatke iz bucketa *temperature_data*
- Uzima samo poslednja 24 sata (*range*)
- Filtrira podatke koji pripadaju merenju "*temperature*"
- Računa prosečnu vrednost temperature (*mean*)

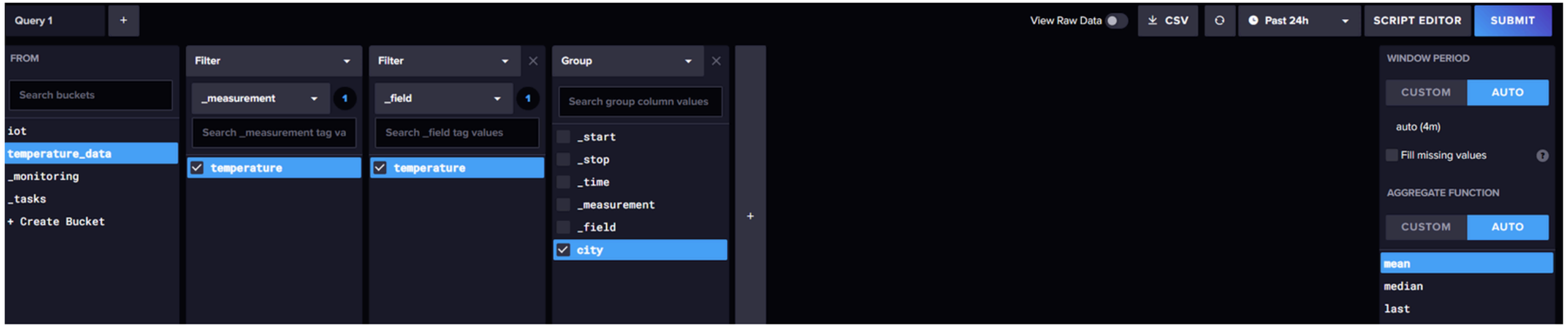
Osnovne Flux funkcije

Funkcija	Opis
`from(bucket: "...")`	Čitanje podataka iz izabranog bucketa
`range(start: ..., stop: ...)`	Definisanje vremenskog raspona podataka
`filter(fn: (r) => ...)`	Filtriranje redova na osnovu uslova
`mean()`	Računanje prosečne vrednosti
`sum()`	Sabira sve vrednosti
`count()`	Broji broj redova
`min()`	Nalazi minimalnu vrednost
`max()`	Nalazi maksimalnu vrednost
`group(columns: [...])`	Grupisanje podataka po kolonama
`aggregateWindow(every: ..., fn: ...)`	Agregacija u vremenske prozore (za grafove)
`yield(name: "...")`	Vraća rezultat upita

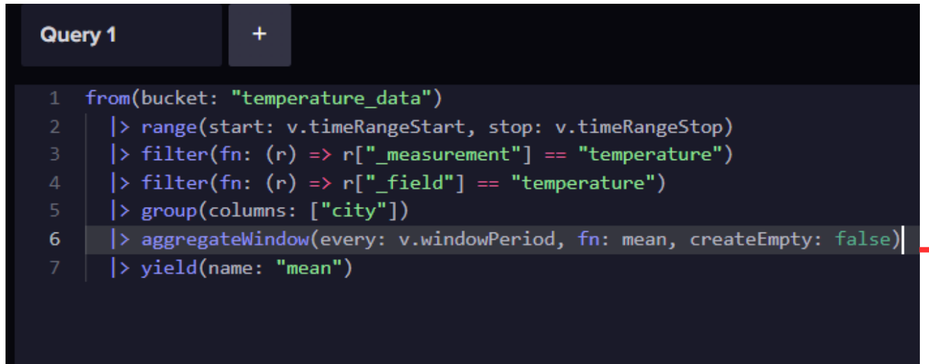
Zadatak - Monitoring temperature u gradovima

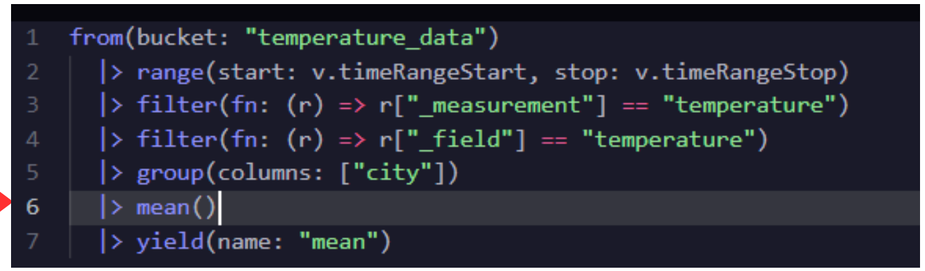
Zadatak 1) Prikazati prosečnu temperaturu za svaki grad u poslednjih 24h

- U Query builderu odabarti:
  - **Bucket → temperature_data** - Biranje iz kog izvora podataka (bucketa) se čitaju podaci
  - **_measurement → temperature** - Filtriranje samo onih podataka koji pripadaju merenju temperature (zadatku monitoringa temperature u gradovima)
  - **_field → temperature** - Odabir konkretnog polja temperature koje se analizira
  - **Group → city** - Grupisanje podataka po gradu, kako bi se dobio prosek po svakom gradu
  - **Time range → Past 24h** - Postavljanje vremenskog raspona za uvažavanje podataka samo iz poslednja 24 sata



- Zatim odabrati Script Editor:
  - Potrebno je funkciju *aggregateWindow*() zameniti sa *mean*()
  - UI automatski generiše upit koji koristi *aggregateWindow*(), ali pošto je potrebno da izračunati prosečnu temperaturu za ceo period, a ne kroz vremenske prozore, potrebno je izmeniti upit. Nakon izmene kliknuti na Submit



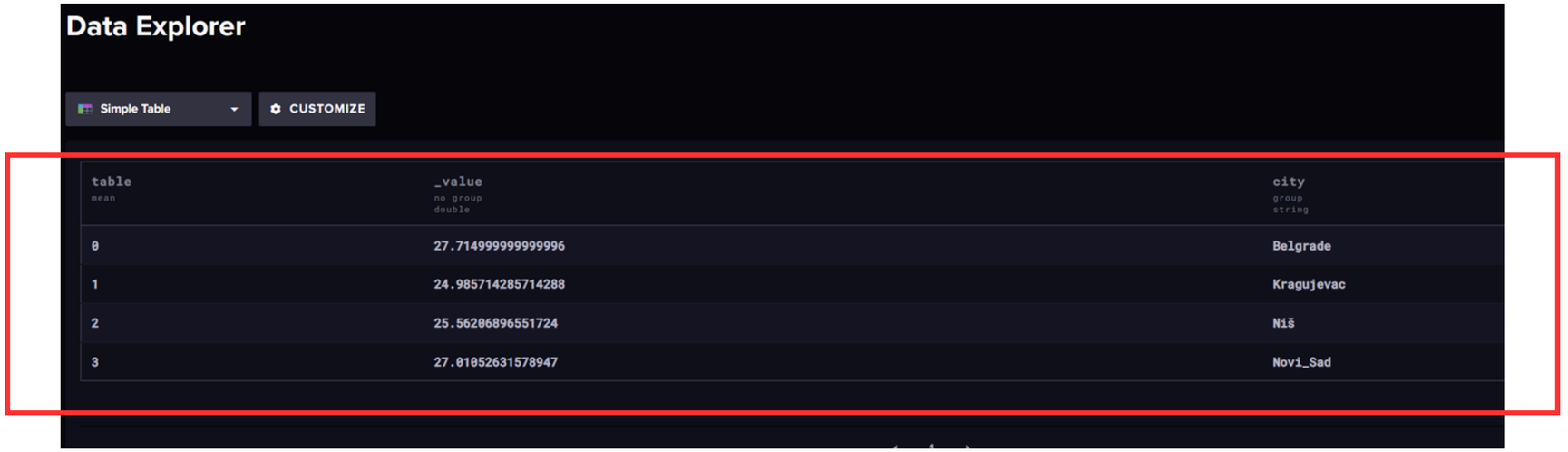


Zadatak - Monitoring temperature u gradovima

Zadatak 1) Prikazati prosečnu temperaturu za svaki grad u poslednjih 24h

- Rezultujući upit je moguće predstaviti na više načina: *Heatmap, Graph, Gauge, Scatter, Simple table, Table*

Simple table:



Graph:

Zadatak - Monitoring temperature u gradovima

Zadatak 2) Pronaći maksimalnu temperaturu za svaki senzor u toku jednog dana

- U Query builderu odabarti:
  - **Bucket → temperature_data**
  - **_measurement → temperature**
  - **_field → temperature** - Odabir konkretnog polja temperature koje se analizira
  - **Group → device_id**
  - **Chart → Simple table**
  - **Time range → Past 24h**

- Zatim odabrati Script Editor:
  - Potrebno je funkciju *aggregateWindow*() zameniti sa *max*() - potrebna je najveća temperatura

```
from(bucket: "temperature_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> group(columns: ["device_id"])
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

```
from(bucket: "temperature_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> group(columns: ["device_id"])
  |> max()
  |> yield(name: "max")
```

- Rezultat:

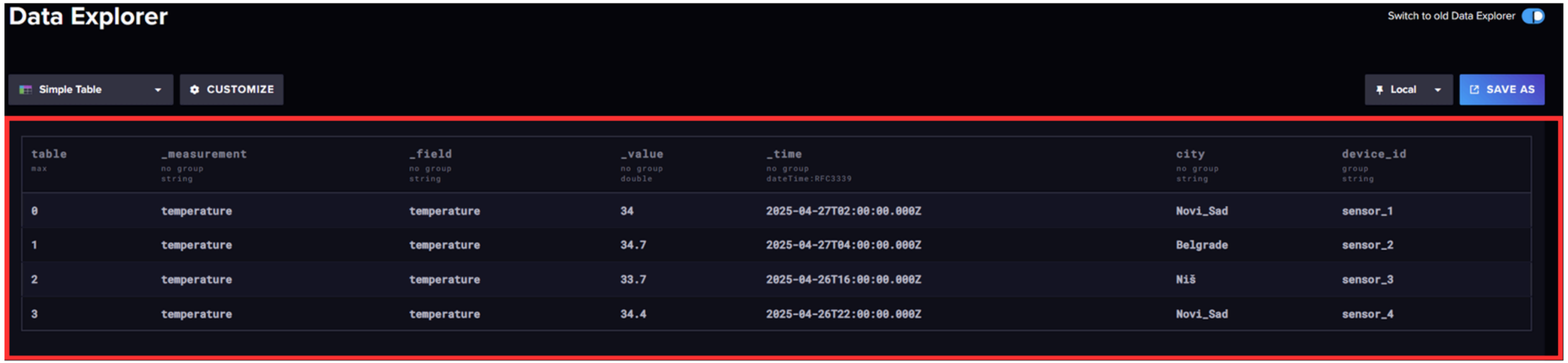


table max	_measurement no group string	_field no group string	_value no group double	_time no group dateTime:RFC3339	city no group string	device_id group string
0	temperature	temperature	34	2025-04-27T02:00:00.000Z	Novi_Sad	sensor_1
1	temperature	temperature	34.7	2025-04-27T04:00:00.000Z	Belgrade	sensor_2
2	temperature	temperature	33.7	2025-04-26T16:00:00.000Z	Niš	sensor_3
3	temperature	temperature	34.4	2025-04-26T22:00:00.000Z	Novi_Sad	sensor_4

Zadatak - Monitoring temperature u gradovima

Zadatak 3) Izračunati minimalnu temperaturu na svakom senzoru u gradu Beogradu u poslednjih 7 dana

- U Query builderu odabarti:
  - **Bucket → temperature_data**
  - **_measurement → temperature**
  - **_field → temperature** - Odabir konkretnog polja temperature koje se analizira
  - **Filter → city=Belgrade**
  - **Chart → Simple table**
  - **Time range → Past 7d**

- Zatim odabrati Script Editor:
  - Potrebno je funkciju *aggregateWindow*() zameniti sa *min*() - potrebna je najmanja temperatura

```
from(bucket: "temperature_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> filter(fn: (r) => r["city"] == "Belgrade")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

→

```
from(bucket: "temperature_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> filter(fn: (r) => r["city"] == "Belgrade")
  |> min()
  |> yield(name: "mean")
```

- Rezultat:

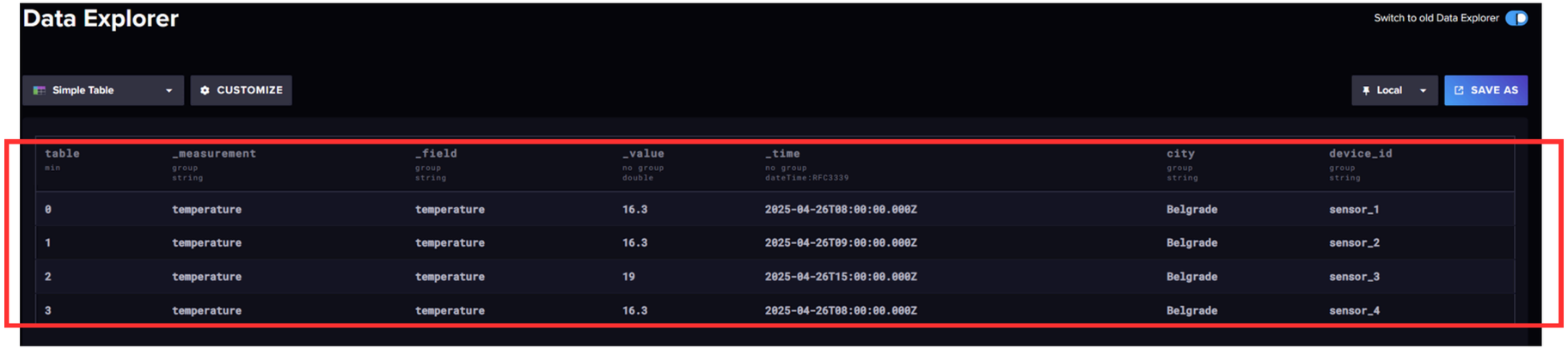


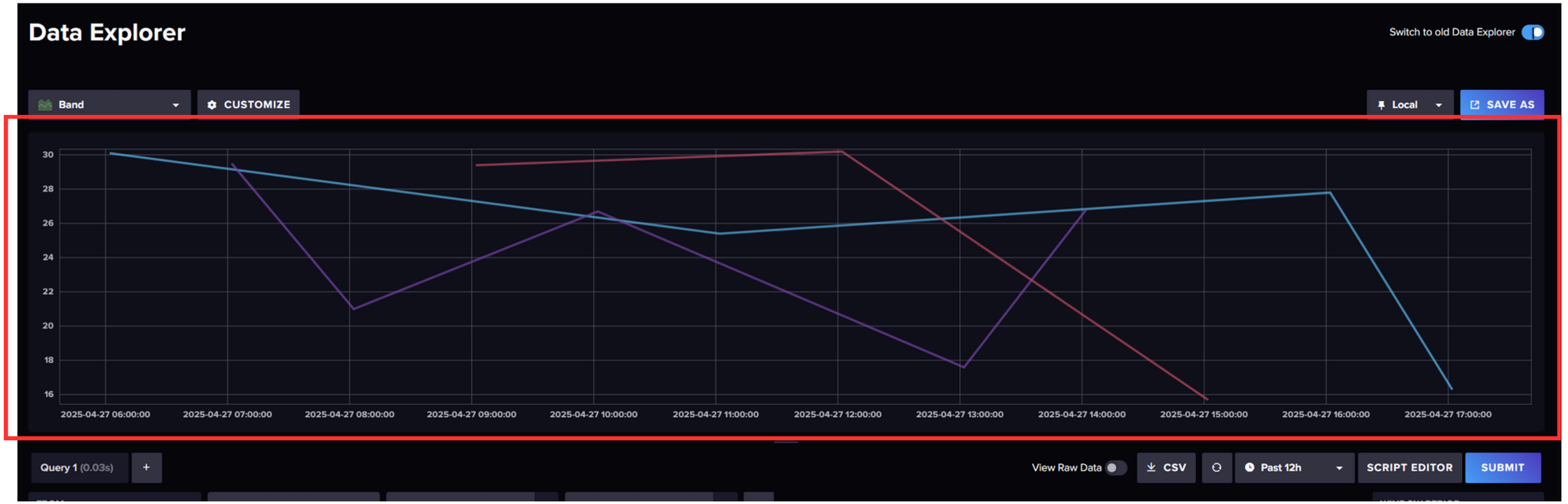
table	_measurement	_field	_value	_time	city	device_id
min	group string	group string	no group double	no group dateTime:RFC3339	group string	group string
0	temperature	temperature	16.3	2025-04-26T08:00:00.000Z	Belgrade	sensor_1
1	temperature	temperature	16.3	2025-04-26T09:00:00.000Z	Belgrade	sensor_2
2	temperature	temperature	19	2025-04-26T15:00:00.000Z	Belgrade	sensor_3
3	temperature	temperature	16.3	2025-04-26T08:00:00.000Z	Belgrade	sensor_4

Zadatak - Monitoring temperature u gradovima

Zadatak 4) Prikazati trend temperature u svakom gradu za senzor sensor_1 u zadnjih 12 sati

- U Query builderu odabarti:
  - **Bucket → temperature_data**
  - **_measurement → temperature**
  - **_field → temperature** - Odabir konkretnog polja temperature koje se analizira
  - **Filter → device_id=sensor_1**
  - **Chart → Band**
  - **Time range → Past 12h**

- Rezultat:



Zadatak - Monitoring temperature u gradovima

Zadatak 5) Izračunati ukupan broj merenja za svaki grad koje je poslao sensor_2 u poslednjih 30 dana

- U Query builderu odabarti:
  - **Bucket → temperature_data**
  - **_measurement → temperature**
  - **_field → temperature** - Odabir konkretnog polja temperature koje se analizira
  - **Filter → device_id=sensor_2**
  - **Chart → Simple table**
  - **Time range → Past 30d**

- Zatim odabrati Script Editor:
  - Potrebno je funkciju *aggregateWindow*() zameniti sa *count*() - brojanje

```
from(bucket: "temperature_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> filter(fn: (r) => r["device_id"] == "sensor_2")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

```
from(bucket: "temperature_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> filter(fn: (r) => r["device_id"] == "sensor_2")
  |> count()
  |> yield(name: "mean")
```

- Rezultat:

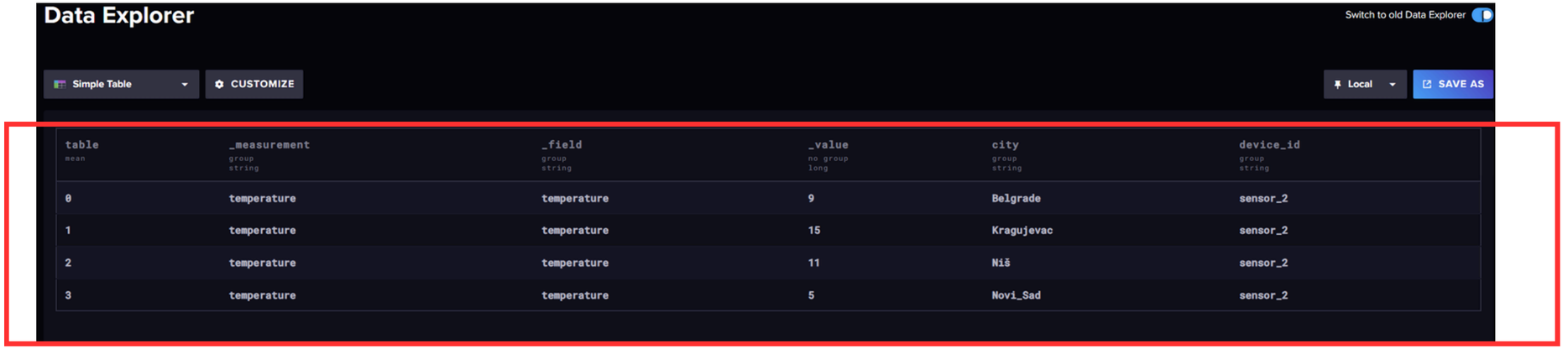


table mean	_measurement group string	_field group string	_value no group long	city group string	device_id group string
0	temperature	temperature	9	Belgrade	sensor_2
1	temperature	temperature	15	Kragujevac	sensor_2
2	temperature	temperature	11	Niš	sensor_2
3	temperature	temperature	5	Novi_Sad	sensor_2