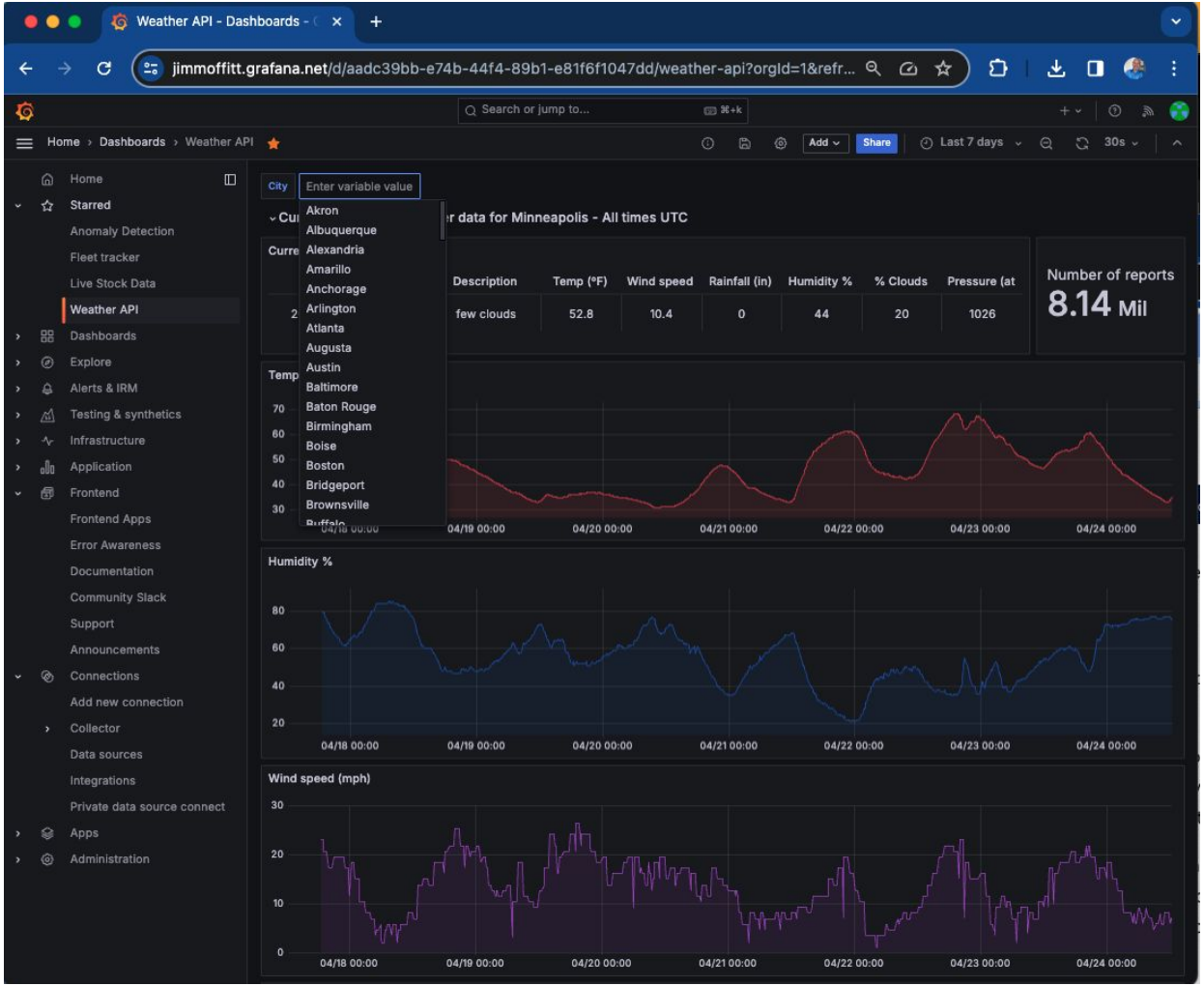
May 21, 2024

tinybird

Building Real-time Anomaly Detection Systems

Workshop (and part live coding session)

Jim Moffitt - Developer Advocate


Weather API - Dashboards -
jimmoffitt.grafana.net/d/aadc39bb-e74b-44f4-89b1-e81f6f1047dd/weather-api?orgId=1&refr...
Search or jump to...
Home › Dashboards › Weather API
Add
Share
Last 7 days
30s
Home
Starred
Anomaly Detection
Fleet tracker
Live Stock Data
Weather API
Dashboards
Explore
Alerts & IRM
Testing & synthetics
Infrastructure
Application
Frontend
Frontend Apps
Error Awareness
Documentation
Community Slack
Support
Announcements
Connections
Add new connection
Collector
Data sources
Integrations
Private data source connect
Apps
Administration
City
Enter variable value
Akron
Albuquerque
Alexandria
Amarillo
Anchorage
Arlington
Atlanta
Augusta
Austin
Baltimore
Baton Rouge
Birmingham
Boise
Boston
Bridgeport
Brownsville
data for Minneapolis - All times UTC
Description
Temp (°F)
Wind speed
Rainfall (in)
Humidity %
% Clouds
Pressure (at
few clouds
52.8
10.4
0
44
20
1026
Number of reports
8.14 Mil
Humidity %
Wind speed (mph)
04/18 00:00
04/19 00:00
04/20 00:00
04/21 00:00
04/22 00:00
04/23 00:00
04/24 00:00

Session resources

- Workshop resources doc
- Anomaly detection Github repository
  - Anomaly detection tutorials
  - Tinybird data project

Session goals

- Set the stage by reviewing what Tinybird is, what are its core objects, and what its datastore “layer” brings to anomaly detection.
- Review anomaly types and detection methods.
- Add anomaly detection to an existing weather monitoring system.

Why?

- You already have data sources set up and you want to start scanning it for anomalies.
- You have a use case where anomalous events are of interest to you ;)
- You have ‘sensors’ that report time-series data ;)

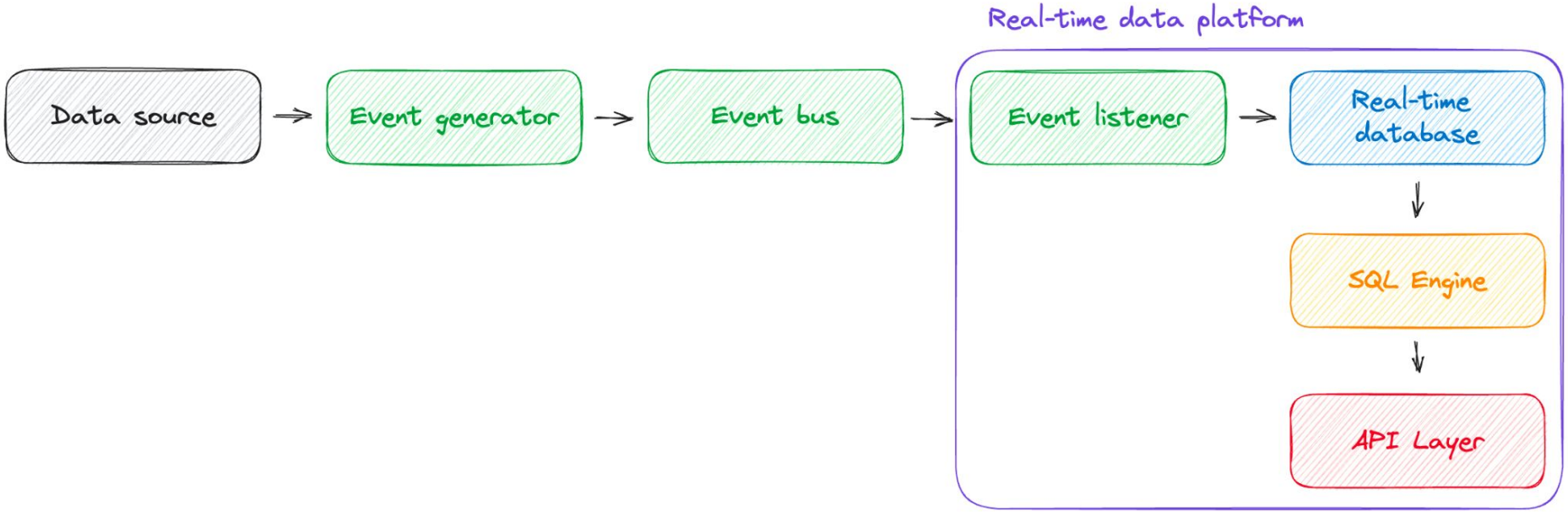
Tinybird core characteristics

What is Tinybird?

- Has an OLAP database as its data layer.
- A data platform for unifying data sources.
- A data analysis platform.
- A place to design and deploy API-based data products.

What is Tinybird?

Tinybird is a real-time data platform



Topics

- Anomaly types - algorithm descriptions
- Implementing algorithms with SQL
- Building detection Pipes
- Publishing API endpoints

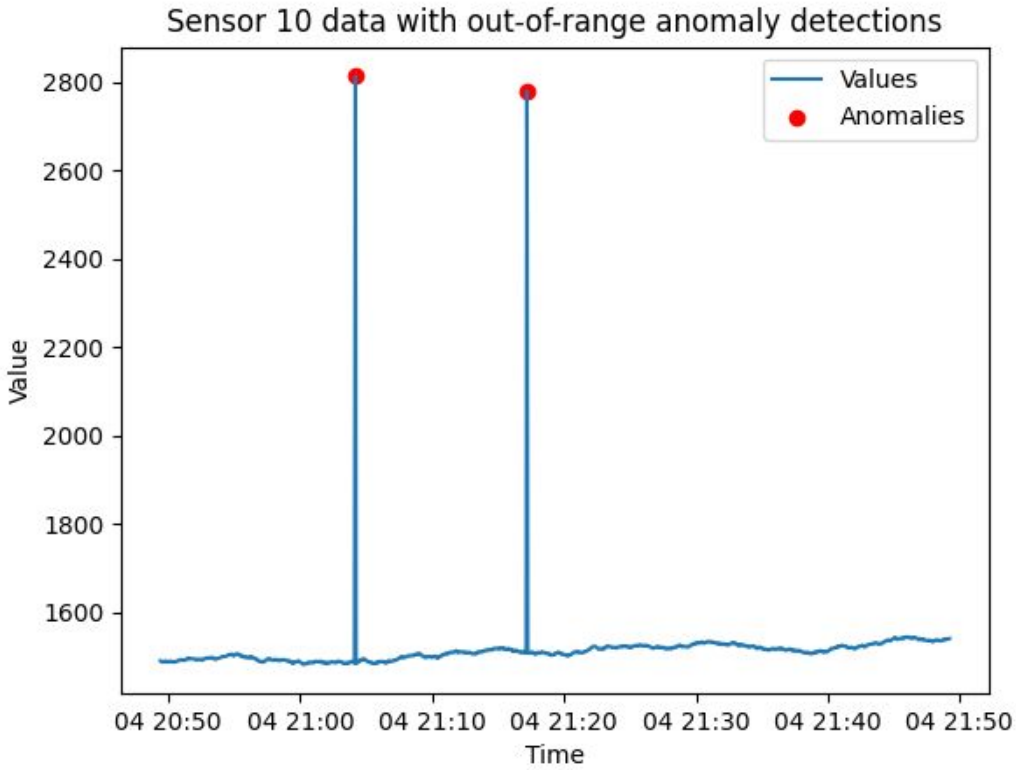
Anomaly Types

- Out-of-range
- Timeout
- Rate-of-change
- Interquartile Range (IQR)
- Z-score

Out-of-range anomalies

- For sensors and data that are expected to report within an expected valid range.

```
SELECT *
FROM incoming_data
WHERE value < min_value OR value > max_value
```


Sensor 10 data with out-of-range anomaly detections
Values
Anomalies
2800
2600
2400
2200
2000
1800
1600
Value
04 20:50
04 21:00
04 21:10
04 21:20
04 21:30
04 21:40
04 21:50
Time

Timeout anomalies

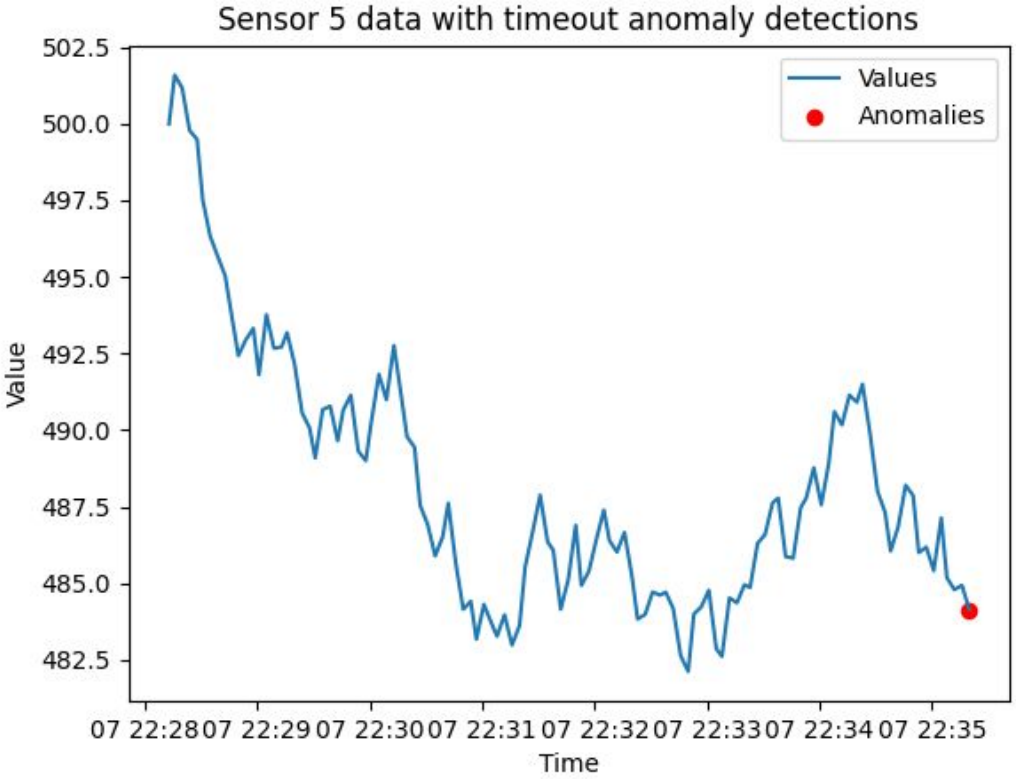
- Most sensors have an interval they are expected to report on.
- Two steps 1) Look up most recent sensor reports and 2) test against a **timeout duration**.

Timeout anomalies

- The SQL challenge is retrieving a set of most recent reports.
- Our first example of how ClickHouse's SQL dialect helps

```
SELECT *
FROM incoming_data
ORDER BY timestamp DESC
LIMIT 1 BY id
```

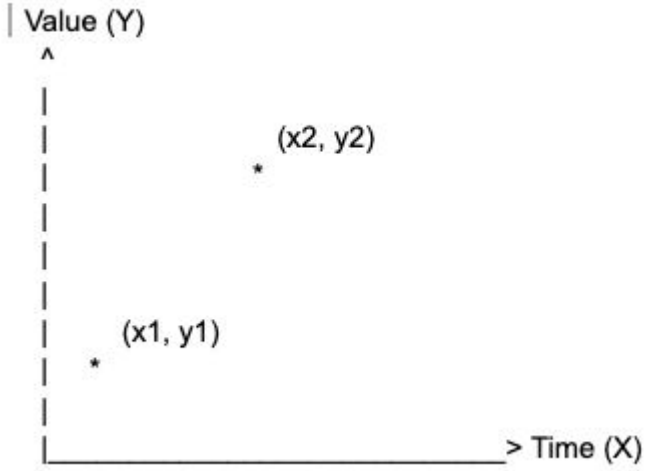
```
SELECT ie.id, ie.value, ie.timestamp
FROM incoming_data ie
JOIN (
    SELECT id, MAX(timestamp) AS max_timestamp
    FROM incoming_data
    GROUP BY id
) AS max_timestamps
ON ie.id = max_timestamps.id
AND ie.timestamp = max_timestamps.max_timestamp
ORDER BY timestamp DESC
```


Sensor 5 data with timeout anomaly detections
Values
Anomalies
502.5
500.0
497.5
495.0
492.5
490.0
487.5
485.0
482.5
Value
07 22:28
07 22:29
07 22:30
07 22:31
07 22:32
07 22:33
07 22:34
07 22:35
Time

Rate-of-change anomalies

- Simple concept
- Rate-of-change = Slope

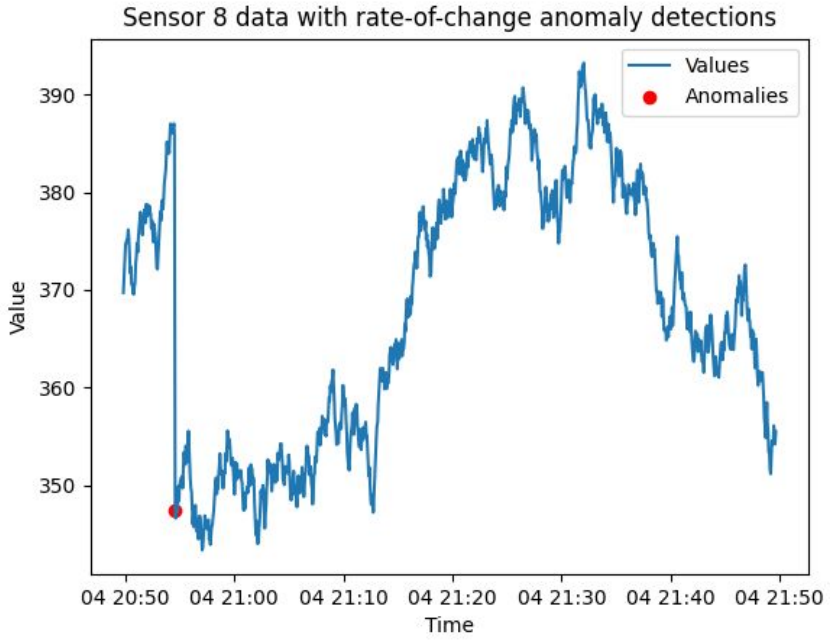
  = (y2 – y1)/(x2 – x1)

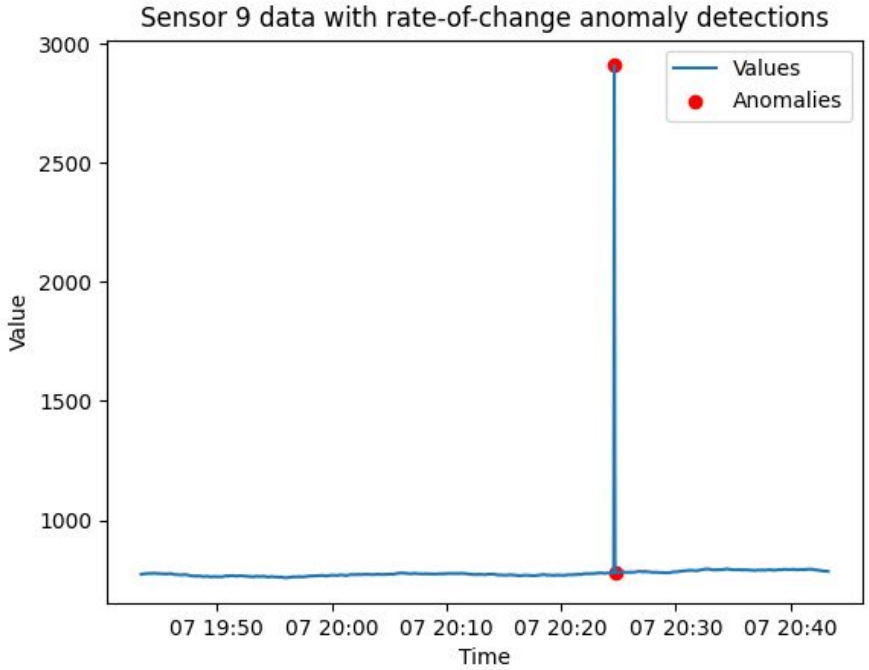


Rate-of-change anomalies

- Simple concept, **more complex SQL**

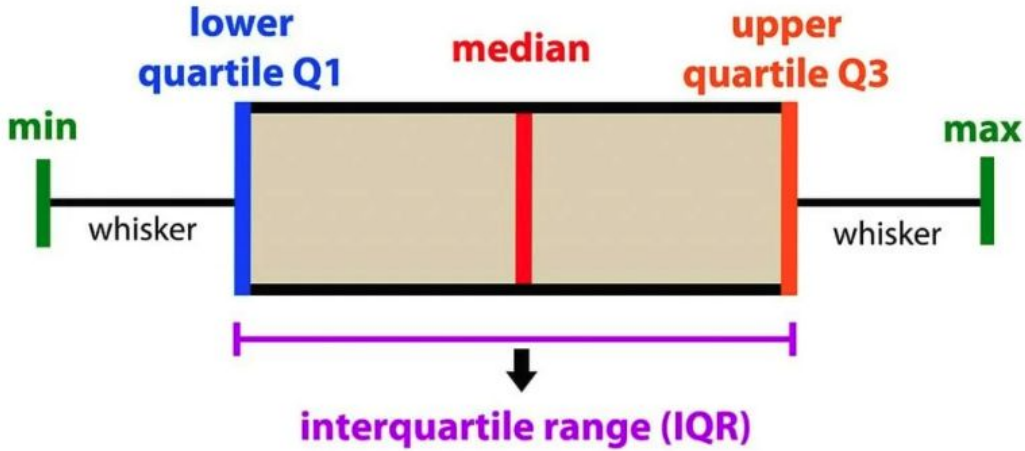
```
lagInFrame(timestamp, 1)
OVER (PARTITION BY id ORDER BY timestamp ASC ROWS
BETWEEN 1 PRECEDING AND 1 PRECEDING) AS previous_timestamp,

lagInFrame(value, 1)
OVER (PARTITION BY id ORDER BY timestamp ASC ROWS
BETWEEN 1 PRECEDING AND 1 PRECEDING) AS previous_value,
```


Sensor 8 data with rate-of-change anomaly detections
Values
Anomalies
390
380
370
360
350
Value
04 20:50
04 21:00
04 21:10
04 21:20
04 21:30
04 21:40
04 21:50
Time


Sensor 9 data with rate-of-change anomaly detections
Values
Anomalies
3000
2500
2000
1500
1000
Value
07 19:50
07 20:00
07 20:10
07 20:20
07 20:30
07 20:40
Time

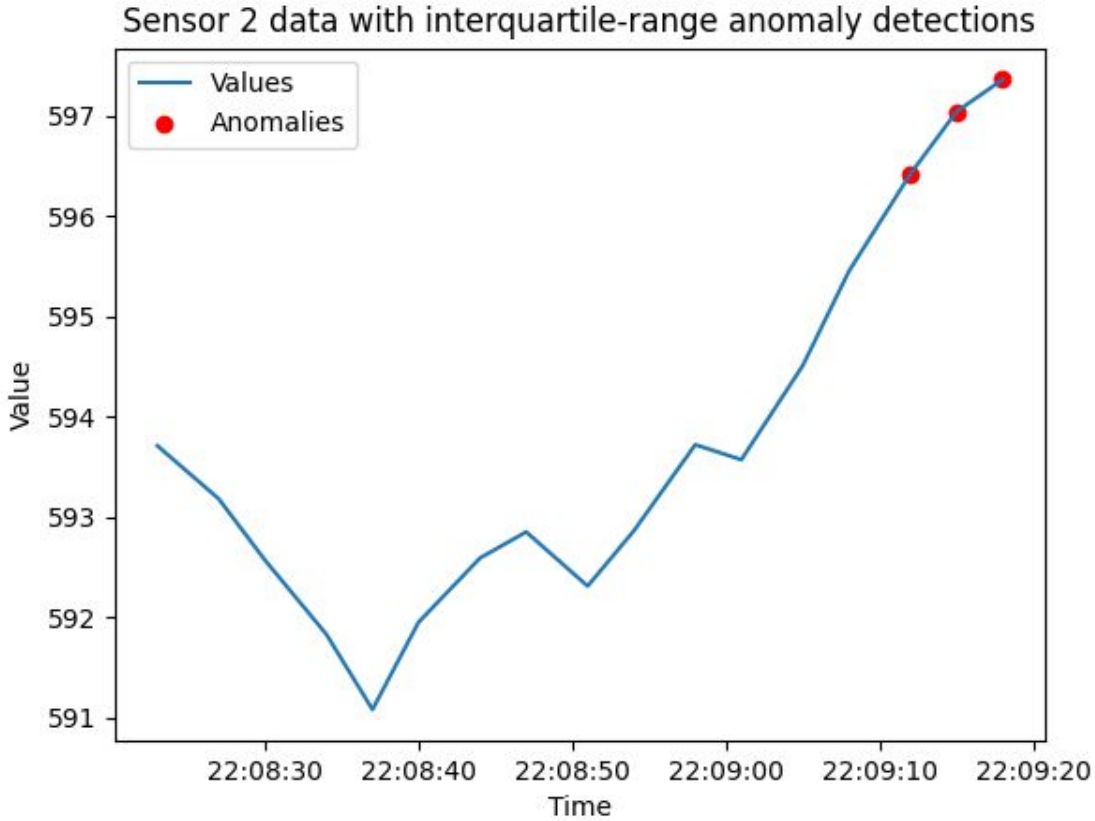
Interquartile Range (IQR) anomalies

- Introduction of a “stats window” for calculating first and third quartiles of data from that time period.
- IQR = Q3 - Q1
- Define a ‘valid` range:
  - values > Q1 - (IQR * 1.5)
  - values < Q3 + (IQR * 1.5)
- The power of SQL CTEs.



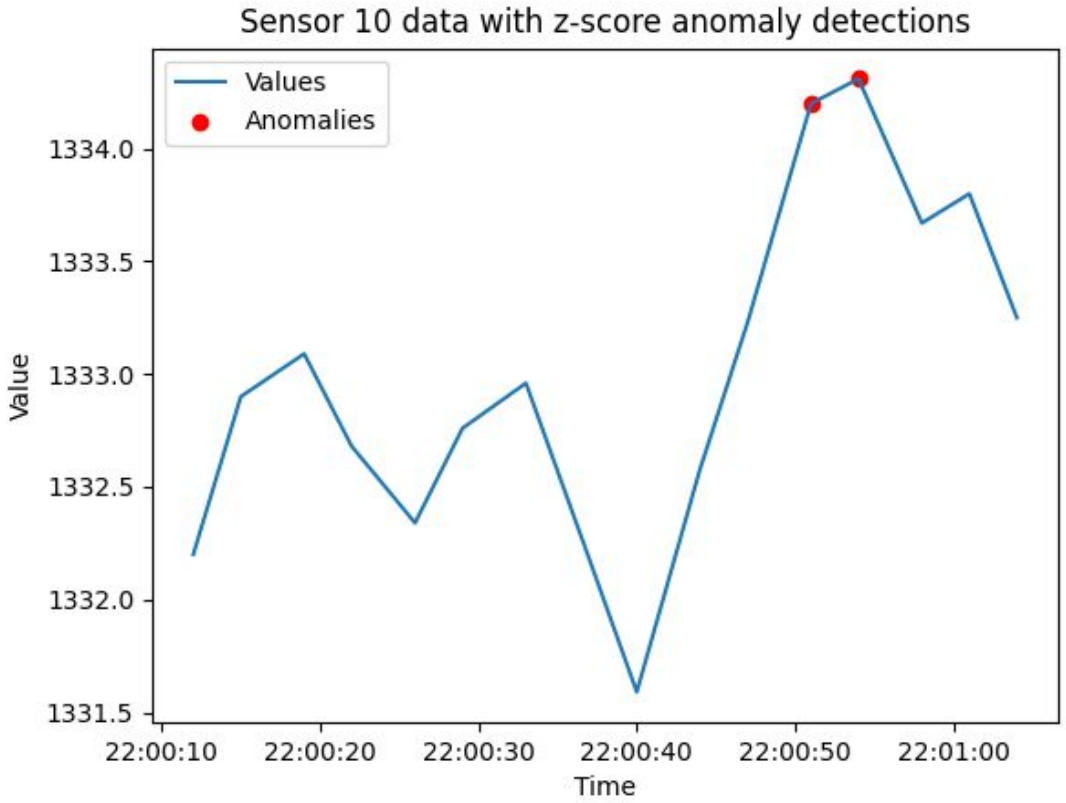
Calculating quartiles and IQR

```
WITH stats AS (
    SELECT id,
        quantileExact(0.25) (value) AS Q1,
        quantileExact(0.75) (value) AS Q3,
        (Q3 - Q1) * 1.5 AS IQR
    FROM incoming_data
    WHERE timestamp BETWEEN (NOW() - INTERVAL 30 MINUTE) AND NOW()
    GROUP BY id
)
SELECT stats.IQR
FROM incoming_data
JOIN stats ON incoming_data.id = stats.id
```


Sensor 2 data with interquartile-range anomaly detections
Values
Anomalies
597
596
595
594
593
592
591
Value
22:08:30
22:08:40
22:08:50
22:09:00
22:09:10
22:09:20
Time

Z-score anomalies

- Similar CTE structure as IQR.
- “stats window” for calculating average and standard deviations of data from that time period.
- Z-score = (value - average) / stddev
- Z-scores higher than a threshold are marked as an anomaly.
- Previous blog post


Sensor 10 data with z-score anomaly detections
Values
Anomalies
1334.0
1333.5
1333.0
1332.5
1332.0
1331.5
Value
22:00:10
22:00:20
22:00:30
22:00:40
22:00:50
22:01:00
Time

Let’s see these implemented in Tinybird

Let’s go build a anomaly detection system

id Int16
timestamp DateTime
value Float32

site_name String
timestamp DateTime
temp_f Float32

Next steps

- Review repository content
  - github.com/tinybirdco/use-case-real-time-anomaly-detection
- Add algorithms to your own data
- Join Slack community
  https://www.tinybird.co/docs/community

Thank you.