INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Applications for data manipulation and storage

Worksheet 3

Name:	Number:
Bikal Bista	a43323
Dibya Raj Khatri	a57096

1. (*) Explain what it is: (i) InfluxDB, (ii) Time-series database and (iii) Flux?

(i) InfluxDB is a specialized database system designed to handle time-series data, which consists of data points collected or recorded at specific time intervals. It is particularly well-suited for applications that generate large amounts of data over time, such as IoT systems, server monitoring, financial data analysis, and energy consumption tracking. InfluxDB is optimized for fast writing and querying of time-stamped data, making it ideal for real-time analytics. It also provides a built-in user interface for data visualization and supports alerts and dashboards to monitor changes over time.

(ii) A time-series database is a type of database that is built specifically to handle time-stamped data. Unlike general-purpose databases, time-series databases are optimized for efficiently storing, retrieving, and analyzing data that changes over time. Each entry in a time-series database typically consists of a timestamp, one or more values (like temperature or humidity), and optional metadata (called tags). This kind of structure is very useful when analyzing trends, detecting anomalies, or generating summaries over a period of time, such as hourly averages or daily maximums.

(iii) Flux is a powerful and flexible query language used in InfluxDB, particularly in version 2.0 and above. It was designed specifically for working with time-series data and allows users to perform advanced operations such as filtering data based on time ranges or field values, calculating averages, detecting changes, joining multiple datasets, and even performing mathematical or statistical computations. Flux queries resemble a pipeline of operations, where each step transforms or filters the data before passing it to the next step. This makes it both powerful and easy to use for time-based analytics.

3.(c) (*) Using Flux (documentation at: https://tinyurl.com/mwkeewae) in the “Script Editor” section, use the range() function to filter data based on time limits. In this case, consider the following range start: 2019-08-22T00:00:00Z to stop: 2019-08-25T00:00:00Z. **Note:** delete the script for item b first.

In this given figure, range() function is used to filter data based on time limits(**2019-08-22T00:00:00Z to 2019-08- 25T00:00:00Z).**

(d). (*) Use the filter() function to filter data based on the location of stations. In this case, only show Santa Monica station data in the previously defined range (location = santa_monica).

Given figure uses the filter**() function** to filter data based on the location of stations santa_monica.

(e) (*) Use the filter() function to filter data based on the measured parameter. In this case, only show temperature data in the previously defined range and location (_measurement = average_temperature).

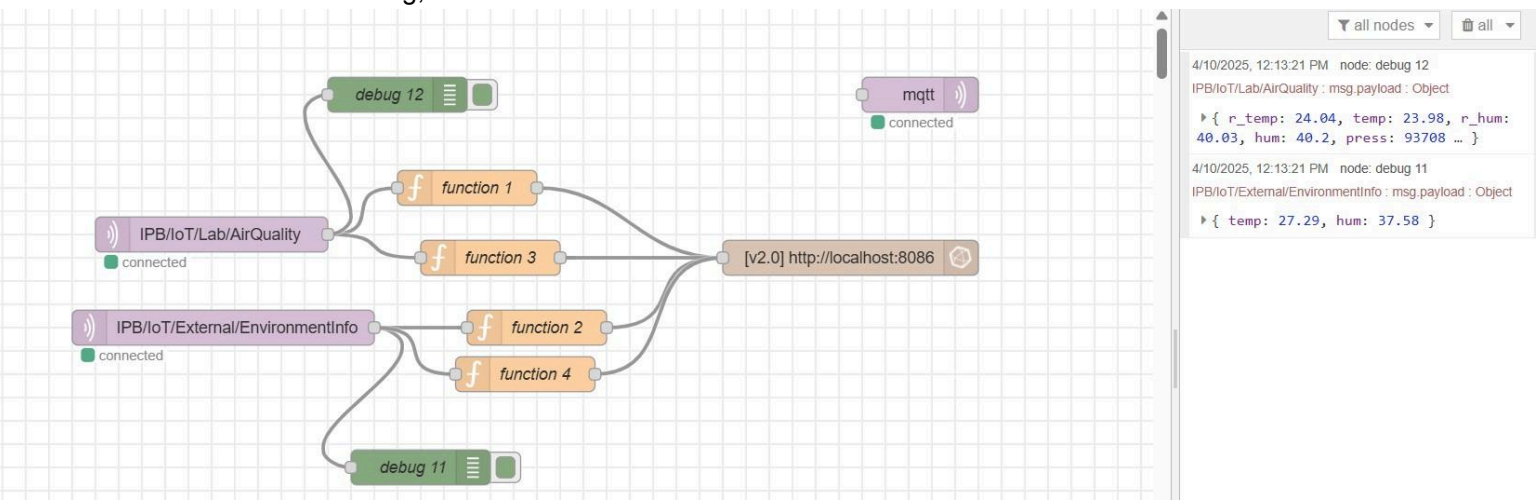
The filter() function to filter data based on the measure parameters **(_measurement = average_temperature)**

(f) (*) Use the aggregateWindow() function to aggregate data. In this case, based on what was done in the previous items, show the maximum values (fn:max) of temperature at each one hour interval (every:1h).

The **aggregateWindow() function** is used to show the max value of temp at each hour interval.

7. (*) Write the temperature and humidity values in InfluxDB. Consider that for each value to be recorded in the DB, it is necessary to define the following fields: fields (temperature and humidity) and tags (location and sensorID). The following is an example of how to perform this configuration in Node-RED. For the values received in the IPB/IoT/Lab/AirQuality topic, consider location = internal_area and for the IPB/IoT/External/EnvironmentInfo topic, consider location = external_area. msg . p a y l o a d = [{

```
measurement : " <... >" , fields:{

} ];

r e t u r n msg ;

t em p e r a t u r e / humidy : msg . p a y l o a d . <... > ,

},

tags:{
s e n s o rI D : " <... >" ,

l o c a t i o n : " <... >"

}

}];

return msg;
```



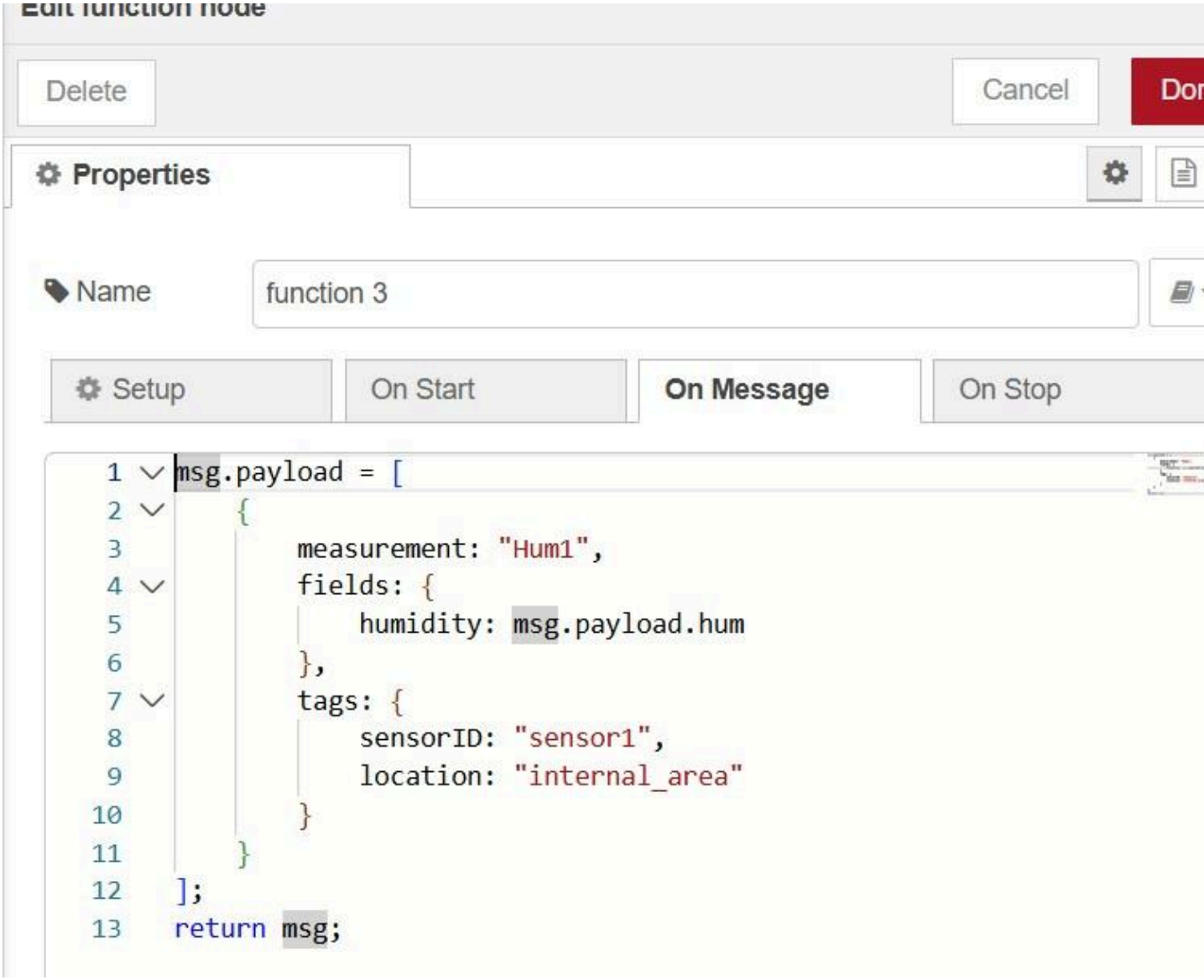
The MQTT-IN nodes in the given figure subscribe to the topics -IPB/IoT/Lab/AirQuality and IPB/IoT/Lab/External/Environmentinfo. The payload is then forwarded to the functions, which take the temperature and humidity values from the payload and store them in influxdb.

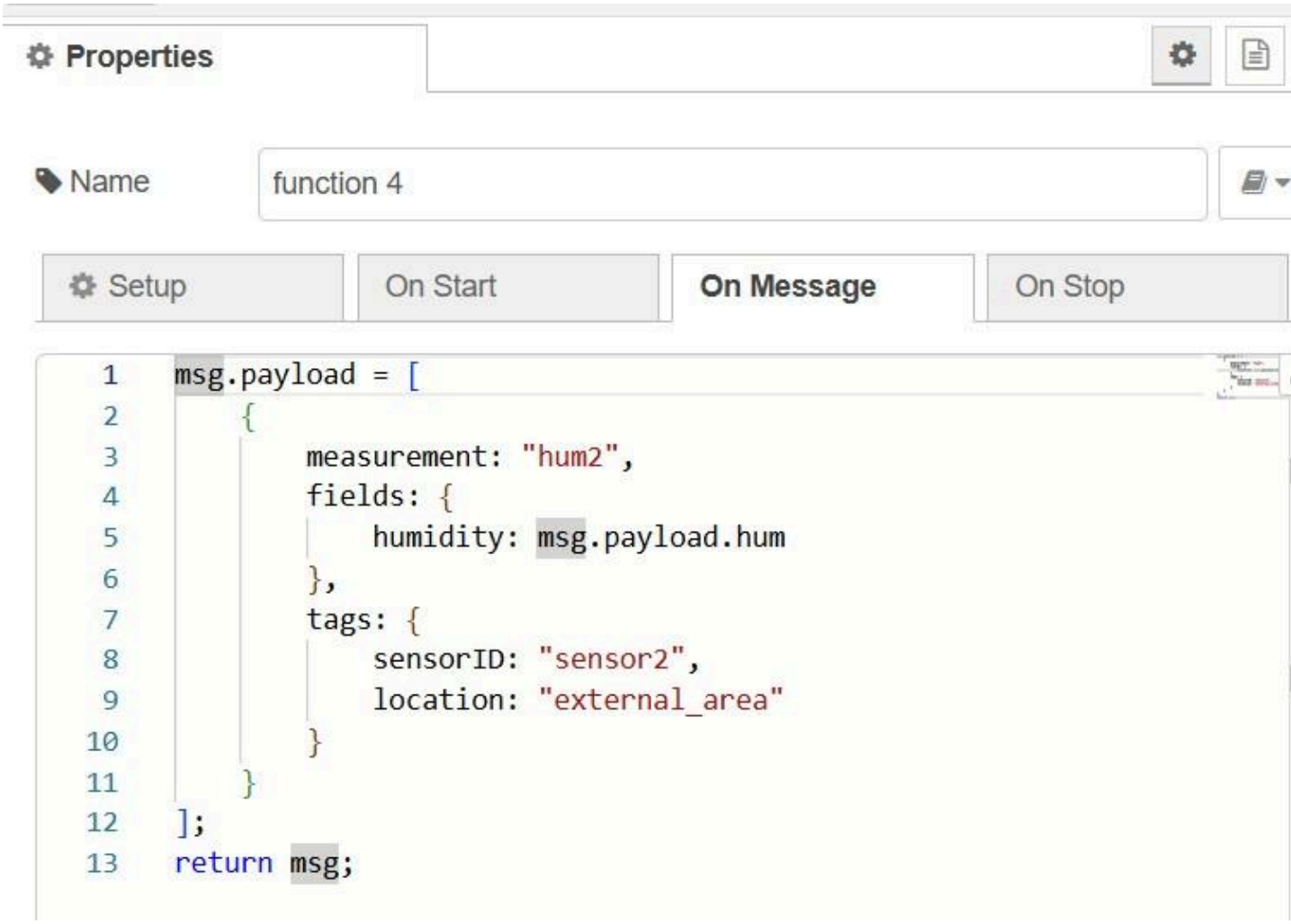
Properties

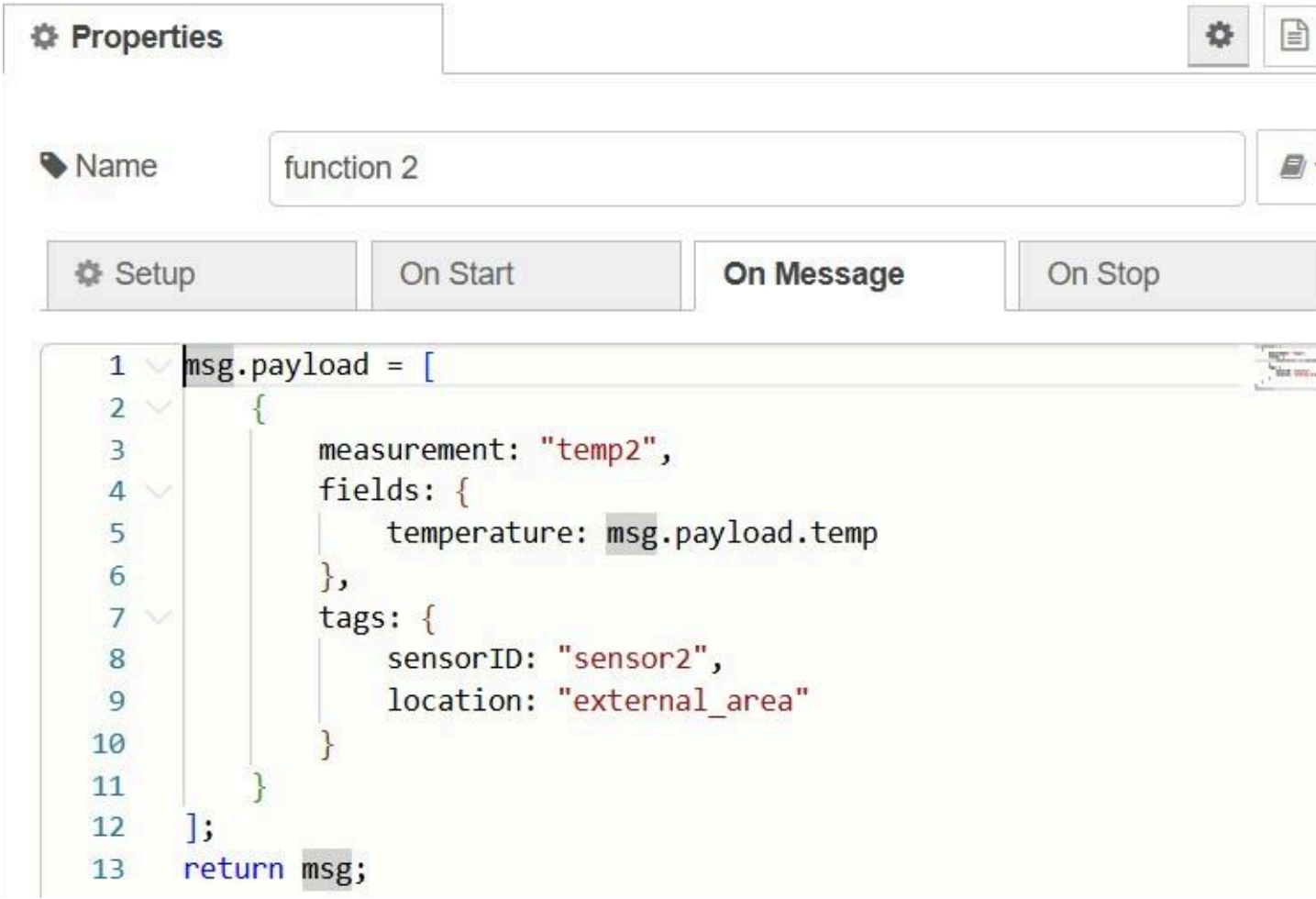
Name function 1

Setup | On Start | **On Message** | On Stop

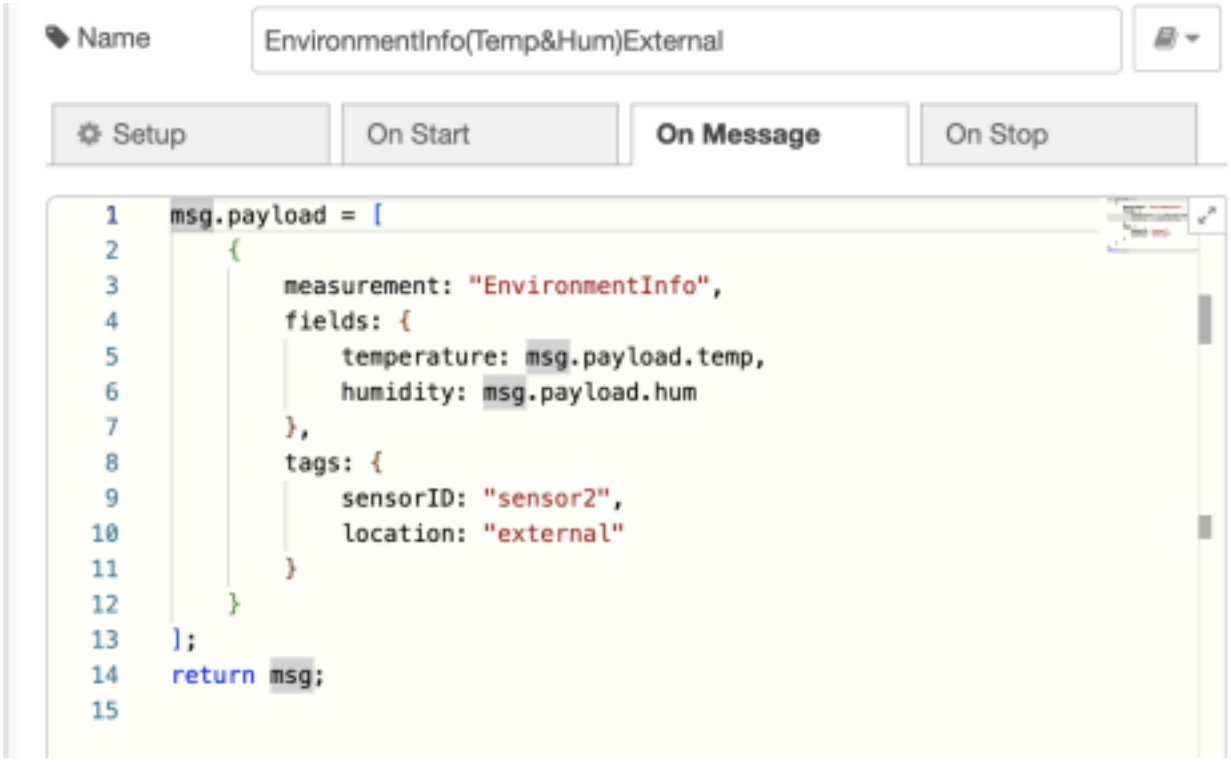
```
1  msg.payload = [
2      {
3          measurement: "temp1",
4          fields: {
5              temperature: msg.payload.temp
6          },
7          tags: {
8              sensorID: "sensor1",
9              location: "internal_area"
10         }
11     }
12 ];
13 return msg;
```


Edit function node
Delete
Cancel
Properties
Name
function 3
Setup
On Start
On Message
On Stop
msg.payload = [
{
measurement: "Hum1",
fields: {
humidity: msg.payload.hum
},
tags: {
sensorID: "sensor1",
location: "internal_area"
}
}
];
return msg;


Properties
Name
function 4
Setup
On Start
On Message
On Stop
msg.payload = [
{
measurement: "hum2",
fields: {
humidity: msg.payload.hum
},
tags: {
sensorID: "sensor2",
location: "external_area"
}
}
];
return msg;



Code snippet of the function for **AirQuality for internal environment** for both, temperature and humidity. We chose to use it in a single function node to increase the efficiency.



Code snippet of the function for **EnvironmentInfo for external environment** for both, temperature and humidity. We chose again to use it in a single function node to increase the efficiency.

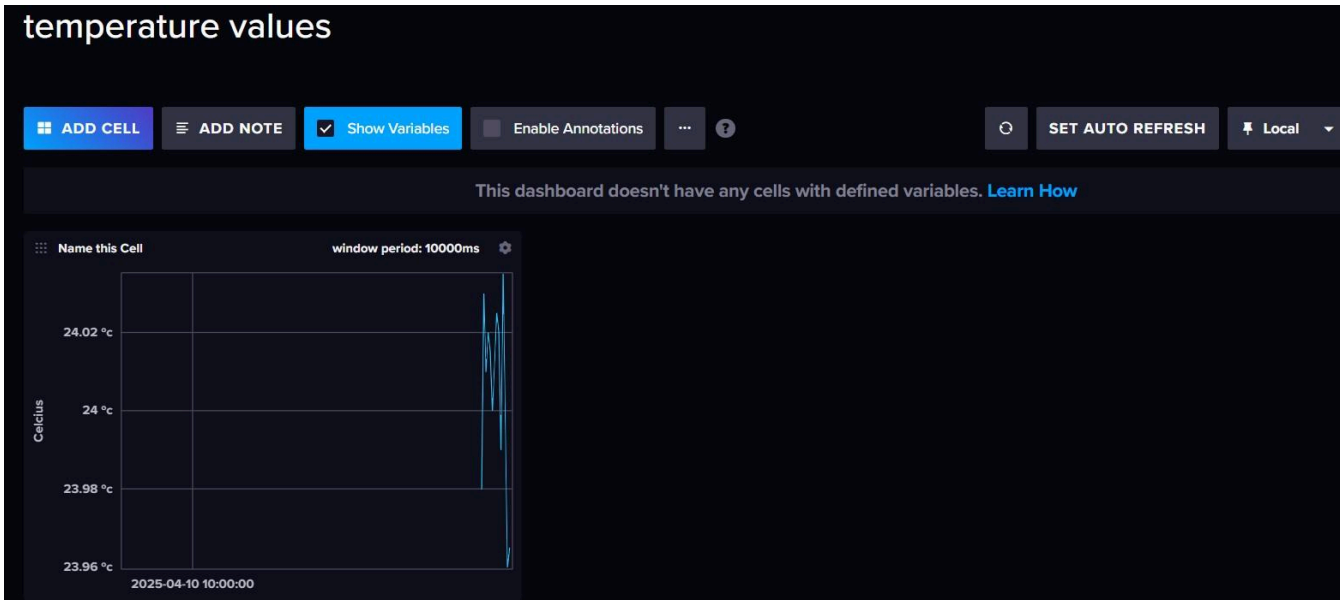
8. (*) In InfluxDB UI, create a dashboard containing:

• Chart showing temperature values.
• Chart showing the humidity values.
• Chart that shows the average temperature for each hour.

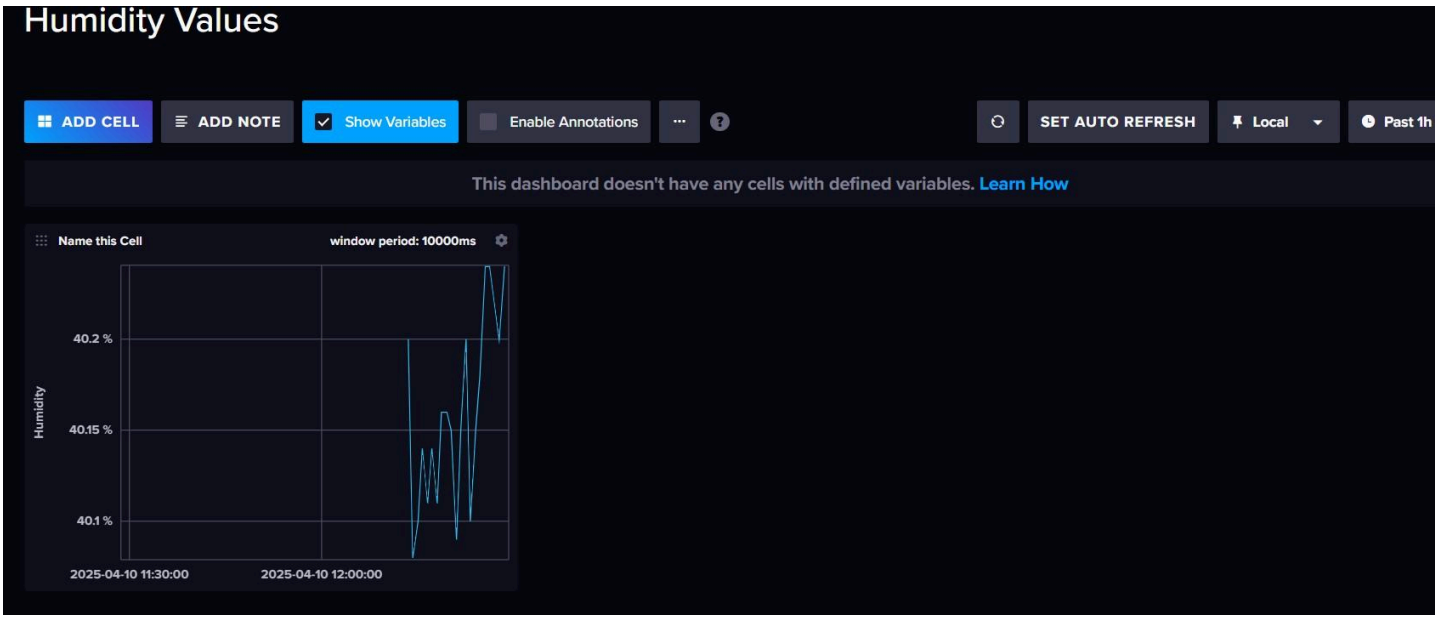
• Chart showing the average humidity every hour.

• Single stats that show the maximum and minimum values for temperature and humidity for the last one hour.

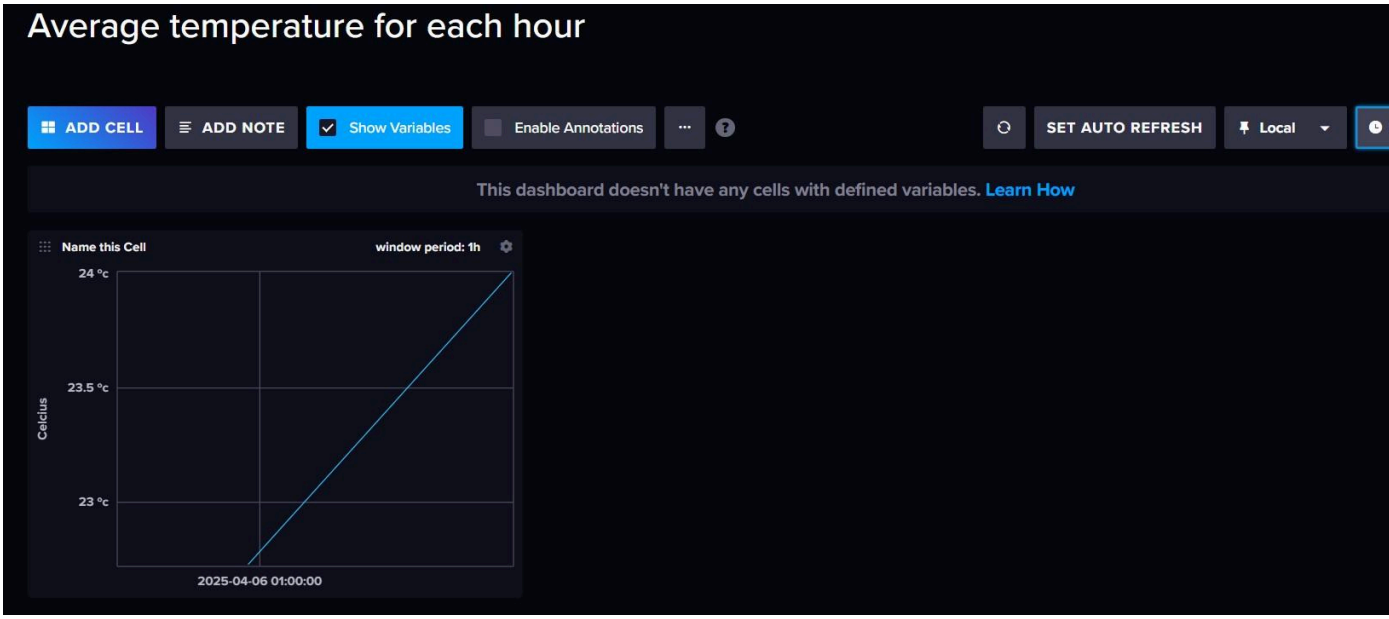
- **Chart showing temperature values.**



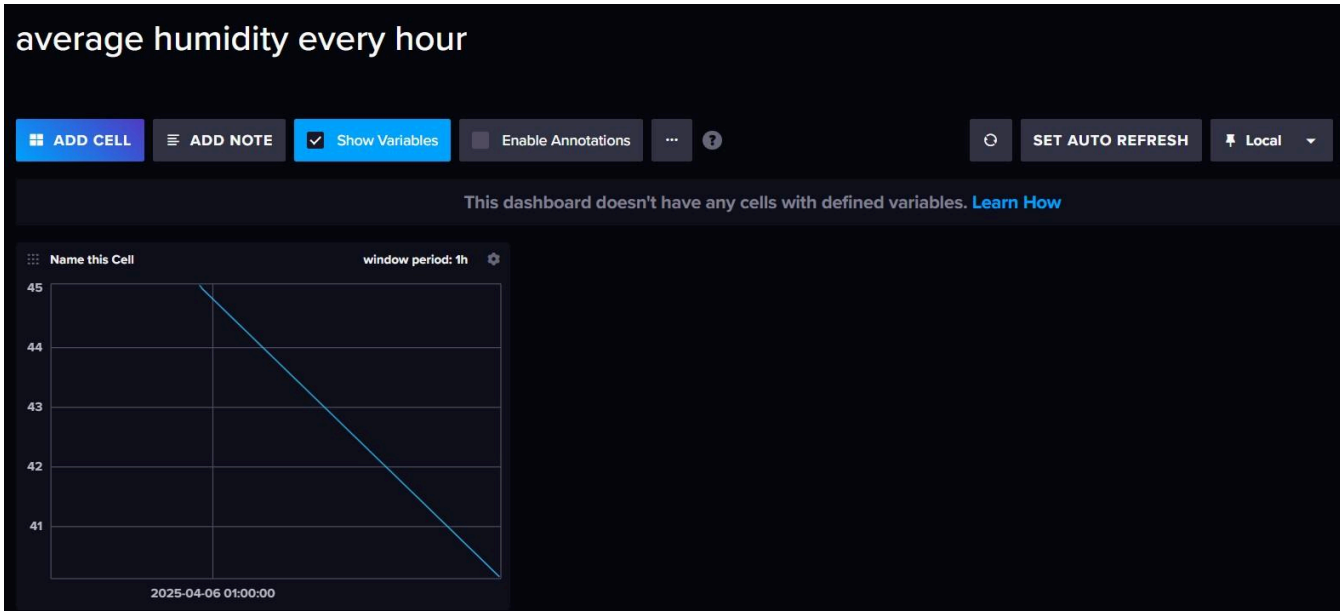
- **Chart showing the humidity values.**



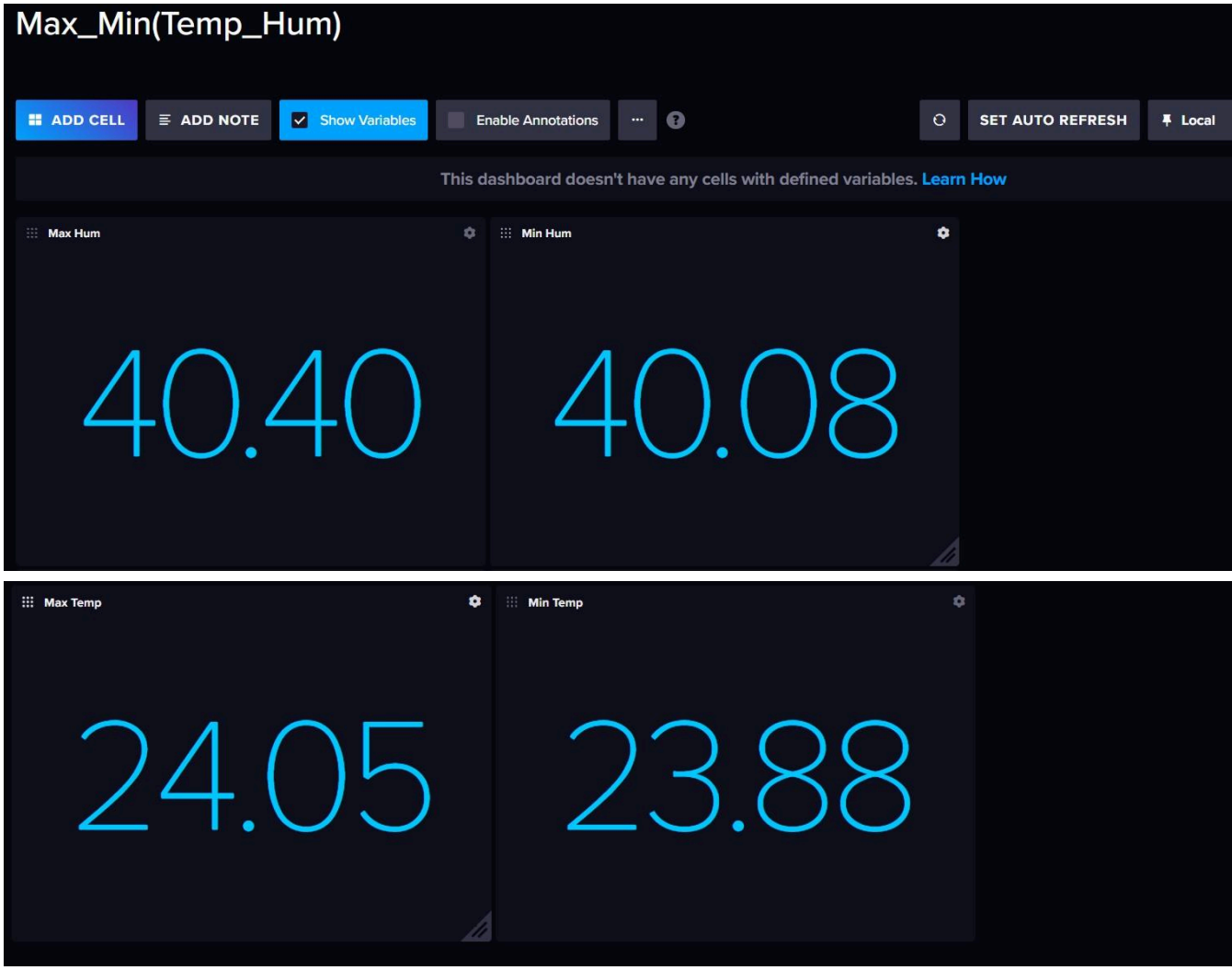
- Chart that shows the average temperature for each hour.



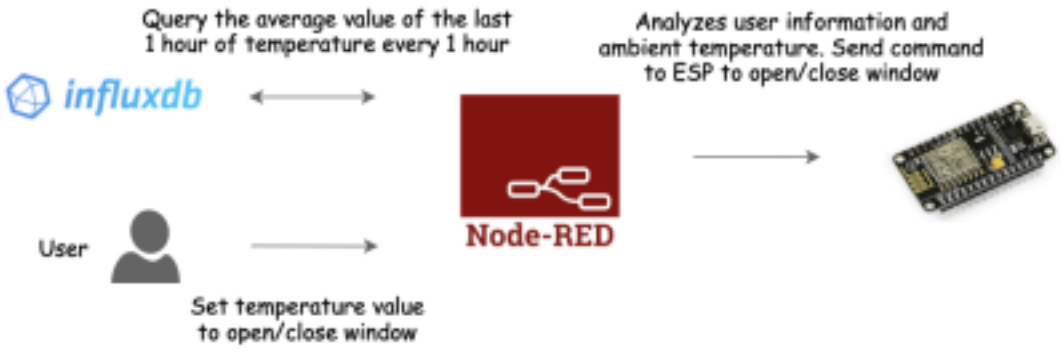
- Chart showing the average humidity every hour.

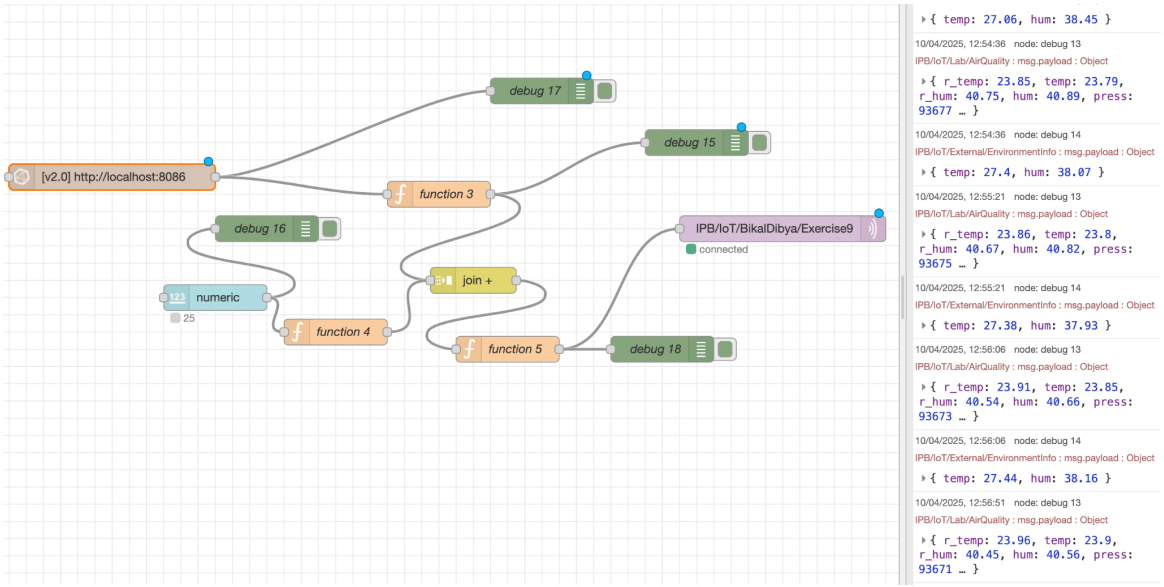


- Single stats that show the maximum and minimum values for temperature and humidity for the last one hour.



9. (*) Consider the following problem: you want to automatically control a window based on the average ambient temperature of the last one hour and the temperature selected by a user. For this, consider the temperature data stored in InfluxDB (performed in previous exercises). Based on the figure below, develop an application that solves the problem described. Simulate the opening/closing of the window using the ESP8266 internal LED.





In the flow shown, a query is sent to InfluxDB to retrieve temperature data. The result is passed to Function 3, where the payload is assigned a topic named "`avg`". Simultaneously, a numeric input node allows the user to input a desired temperature. This value is then processed by Function 4, which sets its topic to "`input`".Both payloads — the average temperature and the user-defined input — are then routed to a Join node, which combines them into a single message object based on their topics.The merged payload is passed to Function 5, which compares the average temperature (`avg`) with the user input (`input`). Based on the result of this comparison, it returns either `1` (if average > input) or `0` (otherwise). This output is finally published to the MQTT topic:
`IPB/IoT/BikalDibya/Exercise9`

`Function 3:`

```
msg.topic = "avg";
```

```
return msg;
```

Function 4:

```
msg.topic = "input";


return msg;
```

Function 5:

```
let avgTemp = msg.payload.avg[0]._value;

let userTemp = msg.payload.input;




if (avgTemp > userTemp) {

   msg.payload = {

       status: "Off",  // Window closed
```

```
        ledStatus: 0    // LED off

    };

} else {

    msg.payload = {

        status: "On",   // Window open

        ledStatus: 1    // LED on

    };

}



return msg;
```

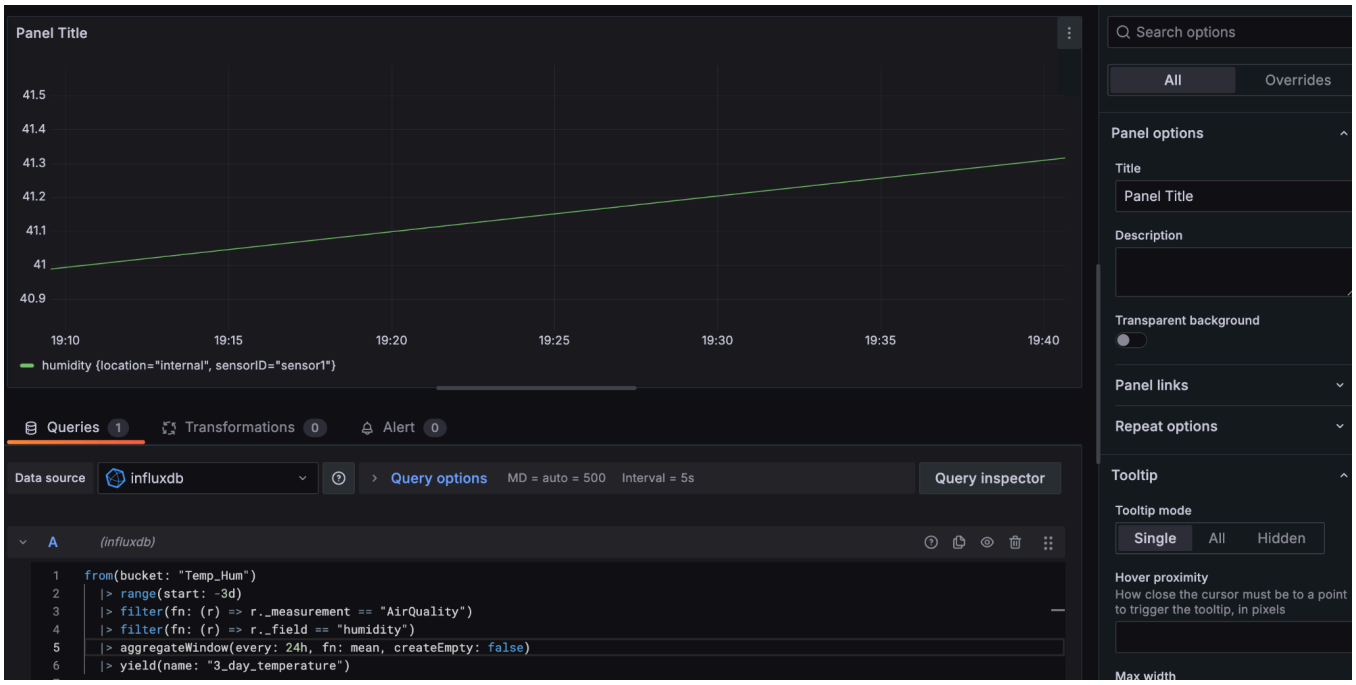
Query in the influxdb node!

11. (*) What is Grafana ?

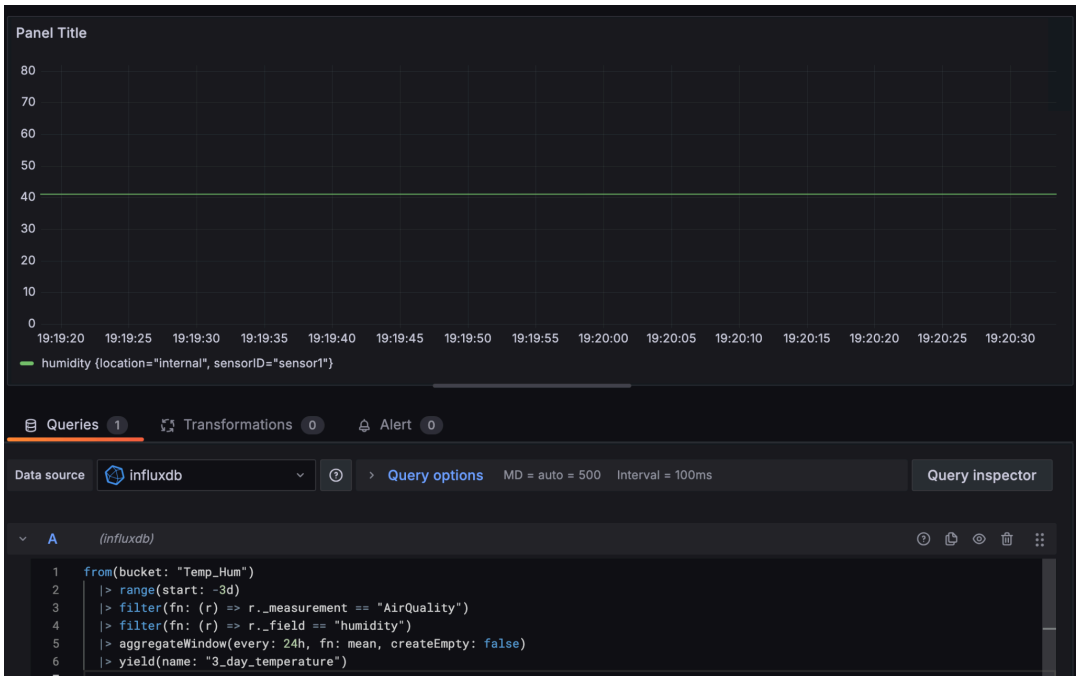
Grafana is an open-source platform for data visualization and monitoring that enables users to create interactive, real-time dashboards for visualizing data from various sources. It supports a wide range of data sources, including time-series databases like InfluxDB, Prometheus, and others. Grafana allows users to design custom dashboards with powerful features like templating, variables, and dynamic filtering. It also provides real-time monitoring, alerting, and notifications based on defined thresholds, making it ideal for tracking system performance, IoT data, and business metrics. With its intuitive interface and robust data exploration capabilities, Grafana is widely used for visualizing complex data and gaining insights across various industries.

12. Considering the database built in exercise 7, use Grafana to:

(a) (*) Present in a graph for each location the variation of the last three days of the temperature, obtained from the IoT nodes installed in the respective locations. The graph must have a resolution of 24 hours and show 3 time series (each represents 1 day);



(b) (*) Present in a graph for each location the variation of the last three days of the **humidity**, obtained from the IoT nodes installed in the respective locations. The graph must have a resolution of 24 hours and show 3 time series (each represents 1 day);



(c) (*) Present a screen capture with all this data. Based on the variation over the last 3 days, what can you conclude about the climate inside the indoor area? What about the outside area? Comment on the difference or similarity between them.

The temperature and humidity data show significant differences between the areas based on their location tags. In the outdoor area, temperature fluctuates widely, with noticeable peaks during the day and lower values at night, which is typical due to direct exposure to weather conditions such as sunlight and wind. In contrast, the indoor area showed more stable temperatures, with smaller, gradual fluctuations, indicating insulation or climate control, such as air circulation systems or walls and windows buffering external temperature changes. Regarding humidity, the outdoor area exhibited higher variability, with spikes in the early mornings and nights and lower levels during the hotter parts of the day, reflecting natural atmospheric moisture behavior influenced by temperature and weather. Meanwhile, the indoor humidity remained relatively consistent with fewer fluctuations, suggesting a more controlled or isolated environment, likely influenced by air conditioning or dehumidifiers.