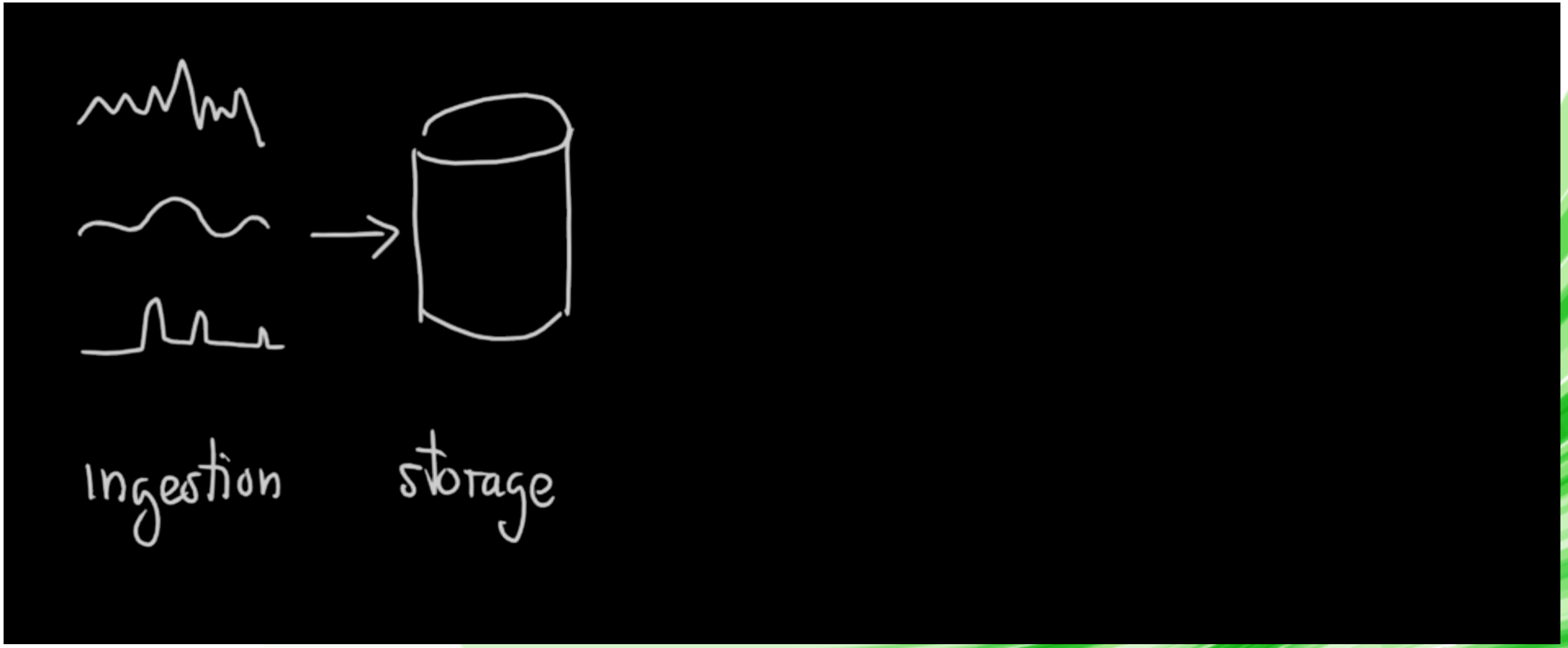
Data Analysis

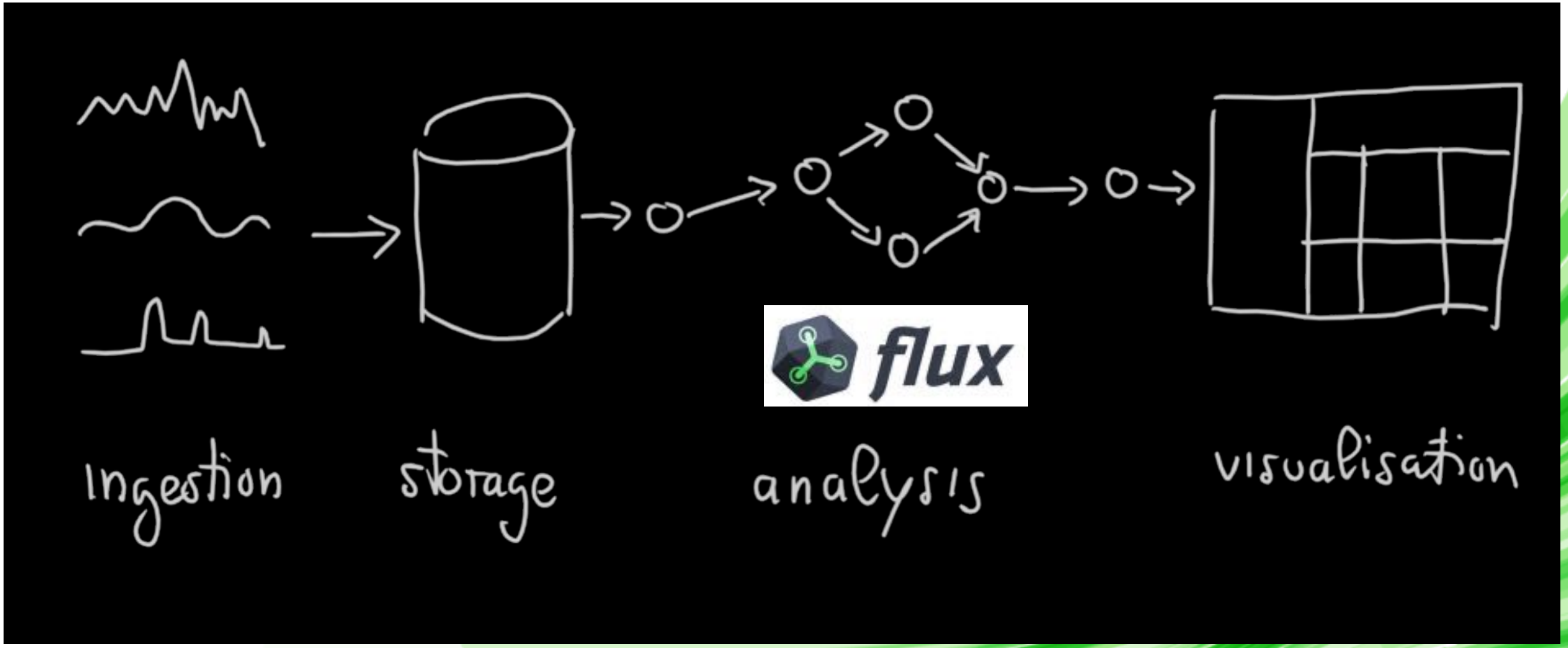
Emanuele Della Valle
Prof. @ Politecnico di Milano & Partner
@ Quantia Consulting
Marco Balduini
Founder & CEO @ Quantia Consulting

Introduction

Data Lifecycle



Data Lifecycle



Flux

What Is Flux?

Flux is a functional data scripting and query language

Written to be:

- Useable: easy to learn
- Readable: developers read more code than we write
- Composable: developers can build onto the language
- Testable: queries are code
- Contributable: open source contributions matter
- Shareable: developers read more code than we write

What is Flux? (cont.)

- Open Source (MIT)
- Data-first mentality
- Streaming Data
- Planner-optimized
- Compatible with multiple sources and sinks

Flux vs. InfluxQL

Flux is an alternative to InfluxQL with additional functionality like:

- Joins
- Math across measurements
- Sort by any column
- Pivot
- Histograms
- Covariance

Reference: https://v2.docs.influxdata.com/v2.0/query-data/

Basic Flux Language Elements

Every Flux query needs:

- data source
- time range
- data filters

Anonymous function

```
from(bucket:"telegraf/autogen")
  |> range(start:-1h)
  |> filter(fn: (r) =>
     r._measurement == "cpu" and
     r._field == "system" and
     r._value > 2000.0)
```

Pipe-forward operator:
chain operations together

Flux Data Model

An infinite stream…

of finite tables…

identified by the GroupKey

From bucket to tables

_time	host	_field	_value
1492…0	serverA	system	2342.2
1492…	serverA	system	477.4
GroupKey[host=serverA,_field=system]			

_time	host	_field	_value
1492…1	serverB	system	2.779
1492…	serverB	system	1274.5
GroupKey[host=serverB,_field=system]			

_time	host	_field	_value
1492…1	serverB	idle	1789.4
1492…	serverB	idle	2475.9
GroupKey[host=serverB,_field=idle]			

```
from(bucket:"telegraf/autogen")
  |> range(start:-1s)
```

_time	host	idle	system
1492…0 … 1492…1	serverA … serverB	1.667 … 1789.4	2342.2 … 2.779

Query Process

For Each Table:

1. Convert incoming rows into 0 or mo[illegible] outgoi[illegible]
2. Map outgoing rows to tables

_time	host	_field	_value
1492…0	serverA	system	2342.2
1492…	serverA	system	477.4
GroupKey[host=serverA,_field=system]			

_time	host	_field	_value
1492…1	serverB	system	2.779
1492…	serverB	system	1274.5
GroupKey[host=serverB,_field=system]			

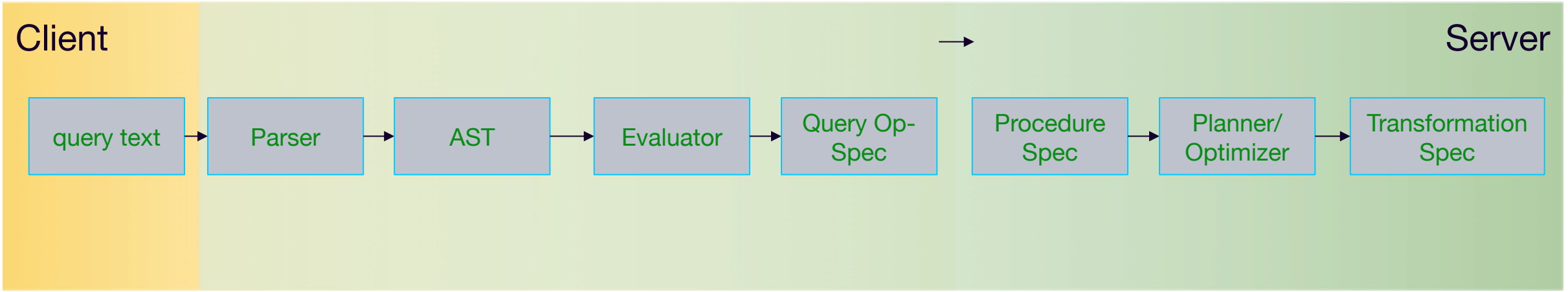
_time	host	_field	_value
1492…1	serverB	idle	1789.4
1492…	serverB	idle	2475.9
GroupKey[host=serverB,_field=idle]			

```
|> filter(fn: (r) =>
    r._measurement == "cpu" and
    r._field == "system" and
    r._value > 2000.0)
```

_time	host	_field	_value
1492…0	serverA	system	2342.2
GroupKey[host=serverA,_field=system]			

_time	host	_field	_value
1492…1	serverB	system	2.779
GroupKey[host=serverB,_field=system]			

The Life of a Flux Query



:)

Wrap up of Key terms and terminology

Bucket: Named data source with retention policy

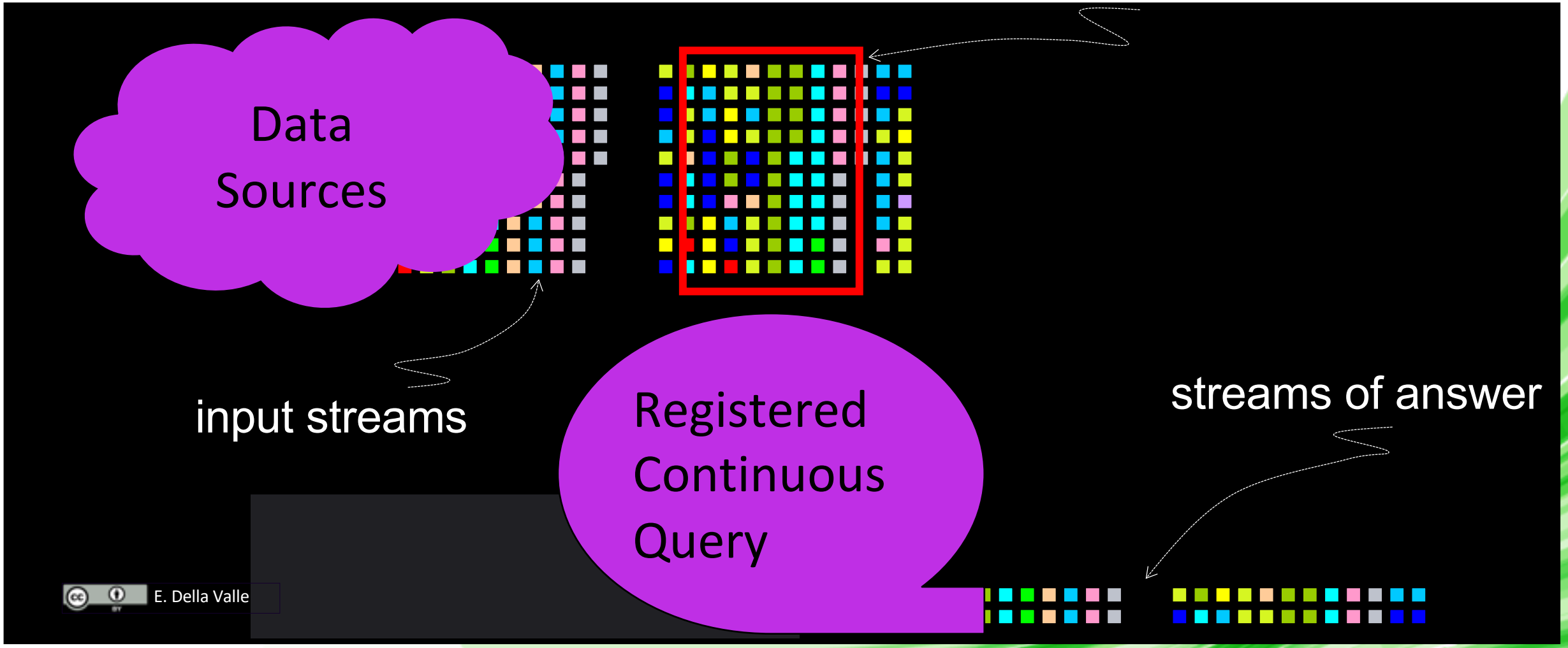
Pipe-forward operator |> : chain operations (e.g., time range and data filters) together

Table: data is returned in annotated CSVs, represented as tables

Group Key: list of columns for which every row in the table has the same value

Exploring Flux
Basics

Key concepts



Windows

Streams are infinite in nature

Processing can end only if the input is finite

A **window operator** selects a finite portion of a stream

- **Time range** is a window operator whose input is a time interval specified either stating an absolute start and end time

```
range(start: 2019-10-01T00:00:00Z,
    stop: 2019-10-01T00:05:00Z)
```

  or stating a relative time period relative

```
range(start:-2h, stop: -1h)
range(start:-1h)
```

Let’s get dirty!

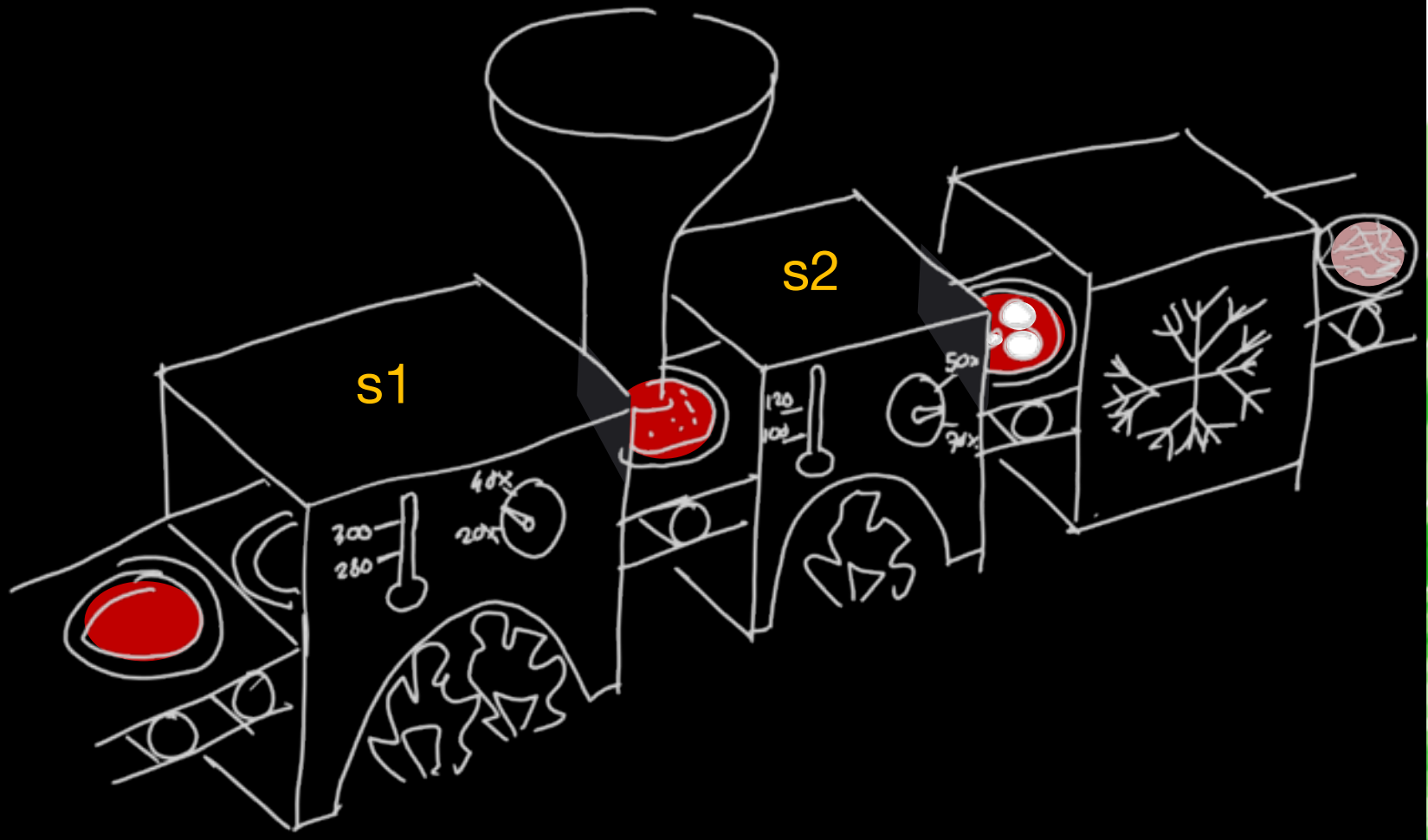
Continuous Linear Pizza Oven

Learning goals:

Data source

Window

Tables



Task

Extract all the measurements in a given range

Absolute: from 2019-10-01T00:00:00Z to 2019-10-01T00:10:00Z

Relative: in the last 24 hours

Take home message

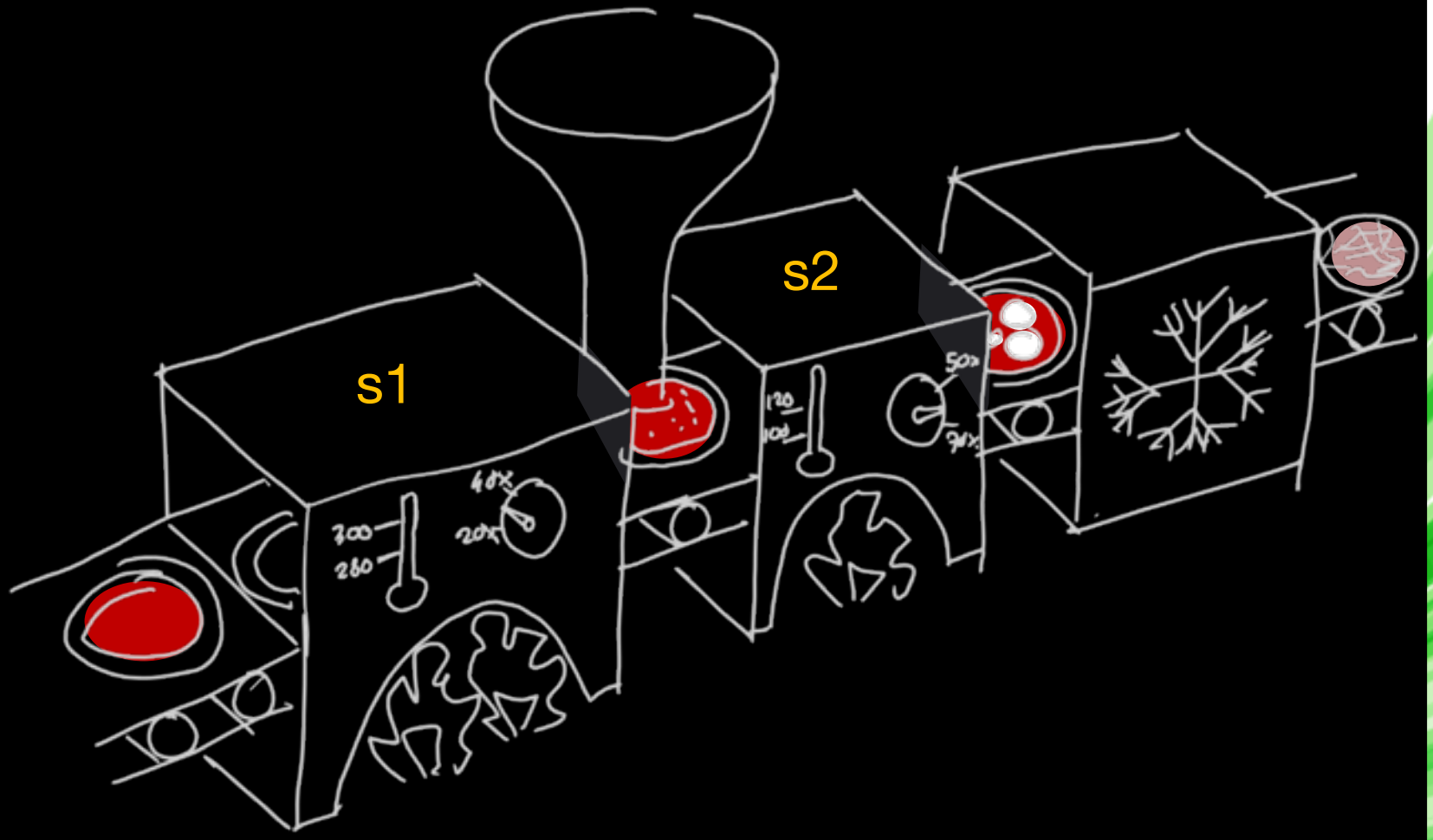
The range() function **extracts** from the named bucket all the possible **tables in the given time range**

Let’s get dirty!

Continuous Linear Pizza Oven

Learning goals:

Filter by tag



Task

Extract the temperature data from the cooking base area (sensor S1)

Take home message

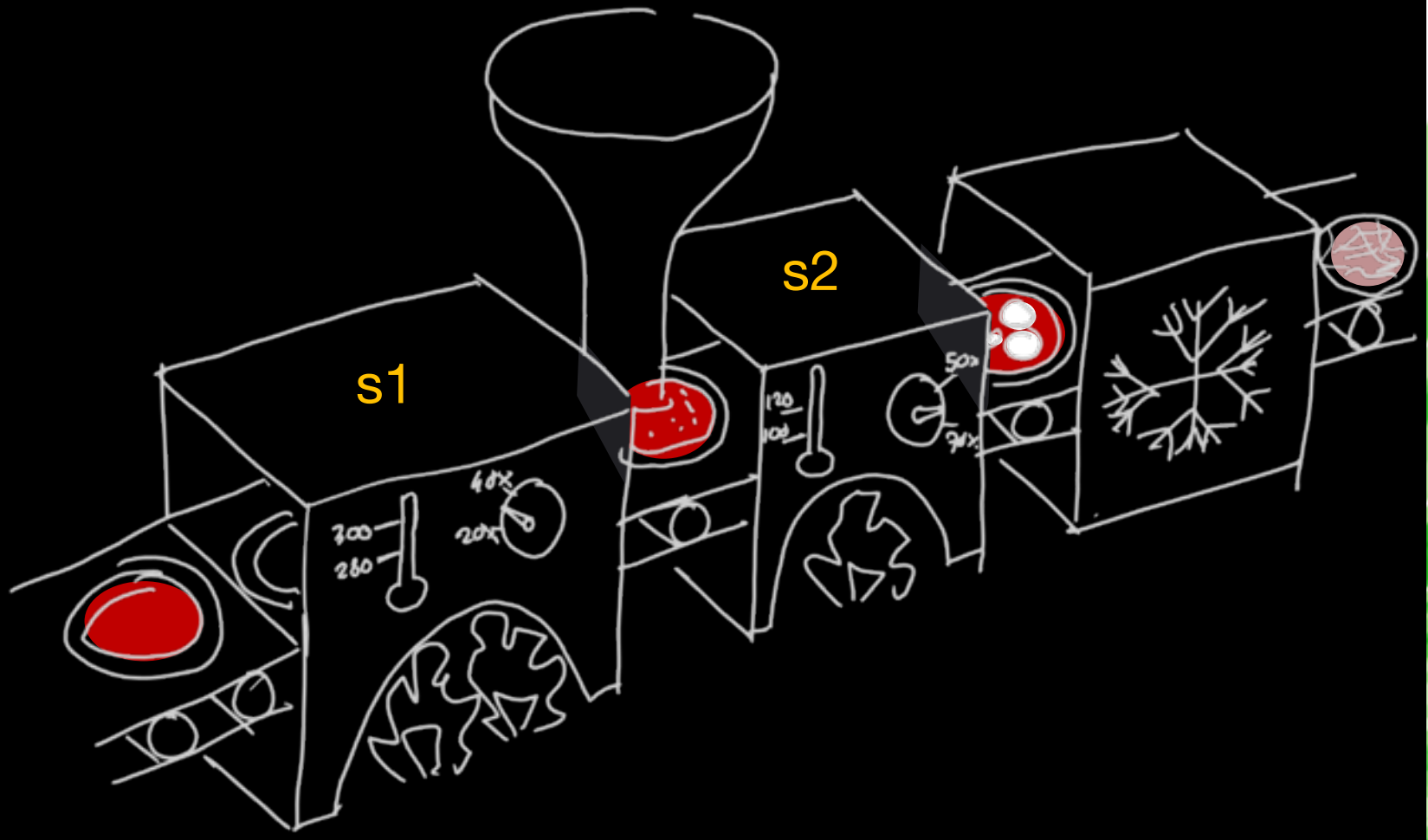
The filter() function **applied to tags selects** only the **table(s)** of your interest from all the possible tables

Let’s get dirty!

Continuous Linear Pizza Oven

Learning goals:

Filter by value



Task

Extract the measurements from the cooking base area (sensor S1) with a temperature under 300°

Take home message

The filter() function **applied to values selects** only the **row(s)** of your interest from all the possible tables

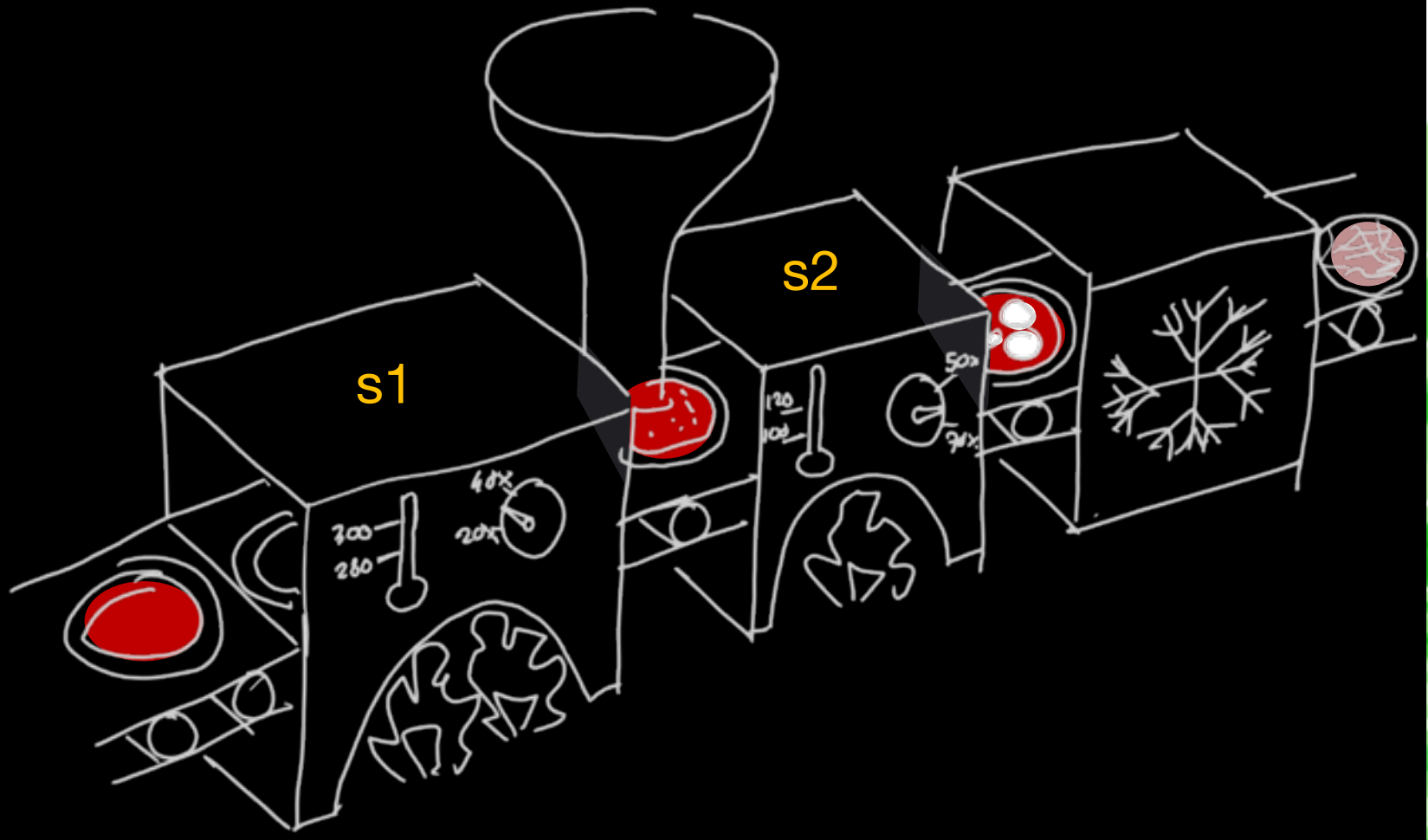
Let’s get dirty!

Continuous Linear Pizza Oven

Learning goals:

Function

- Aggregates
- Selectors



Task

Aggregator - Extract the average temperature in the cooking base area

Selector - Extract the last humidity measurements from the cooking base area

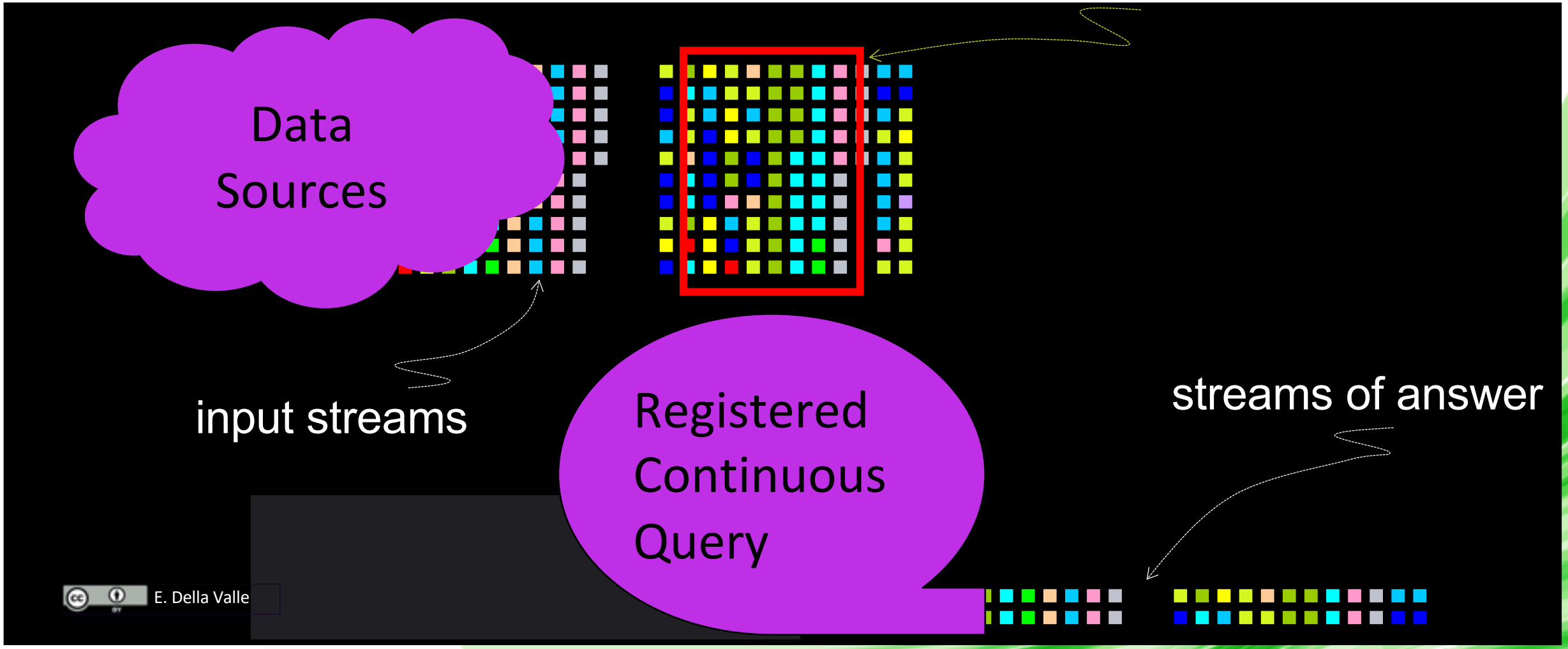
Take home message

Aggregators (e.g. mean) **compress** each table **in a row**

Selectors (e.g. last) select **specific row(s)** from each table

Exploring Flux
Windowing

Recall



Windows

Streams are infinite in nature

Processing can end only if the input is finite

A **window operator** selects a finite portion of a stream

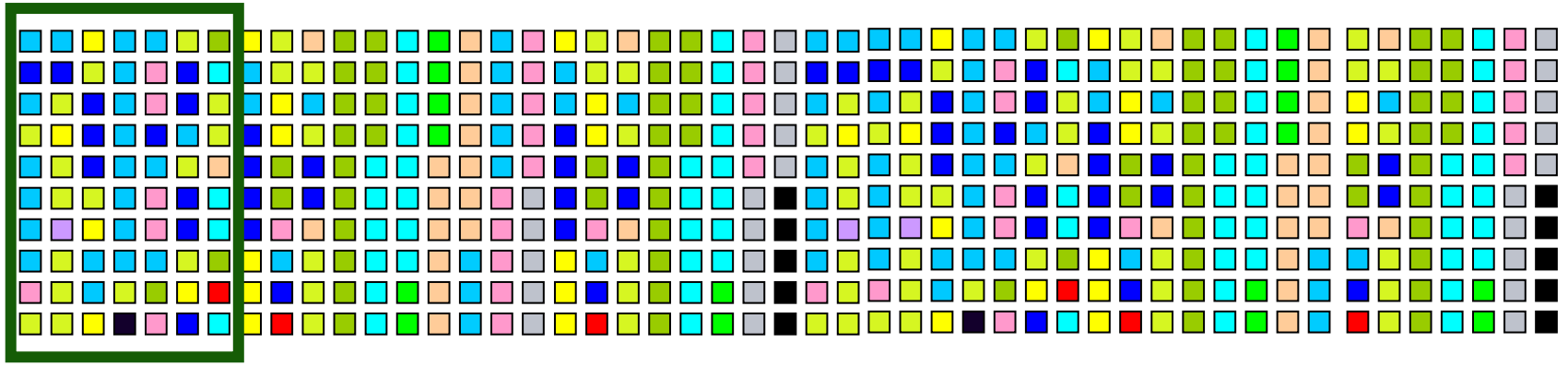
We have seen the time range operator

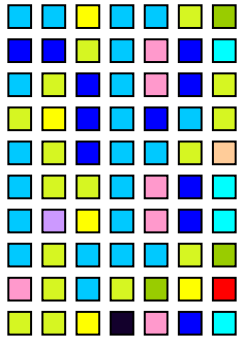
Flux includes a set of window operators that **tumble** across the results of the time range operator

- e.g. aggregateWindow(every: 60m, fn: mean)

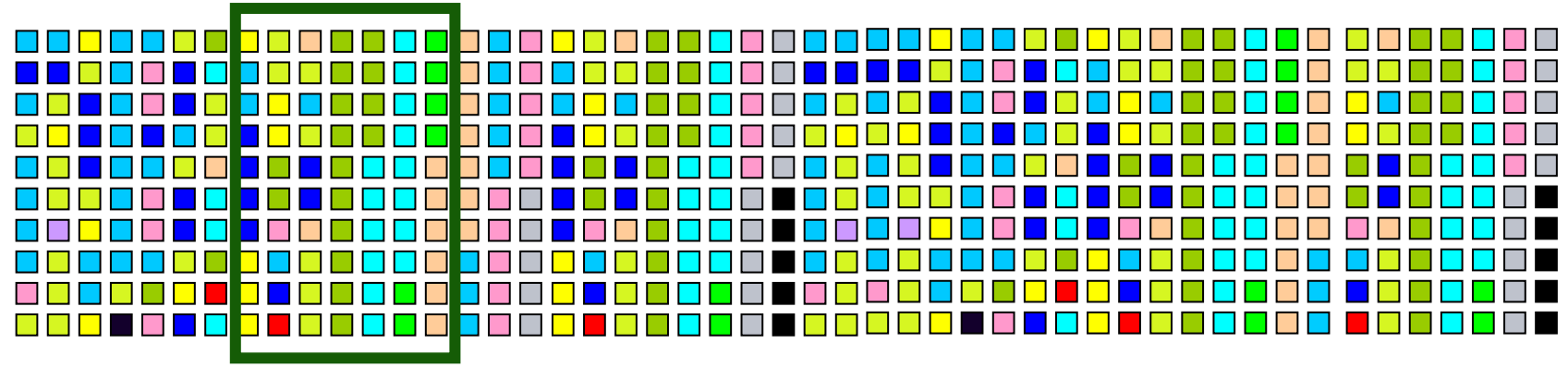
Tumble?

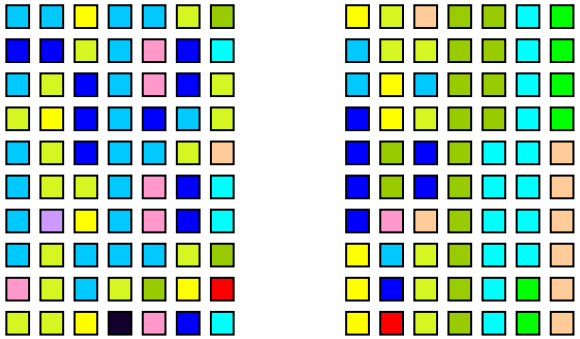
Tumbling windows





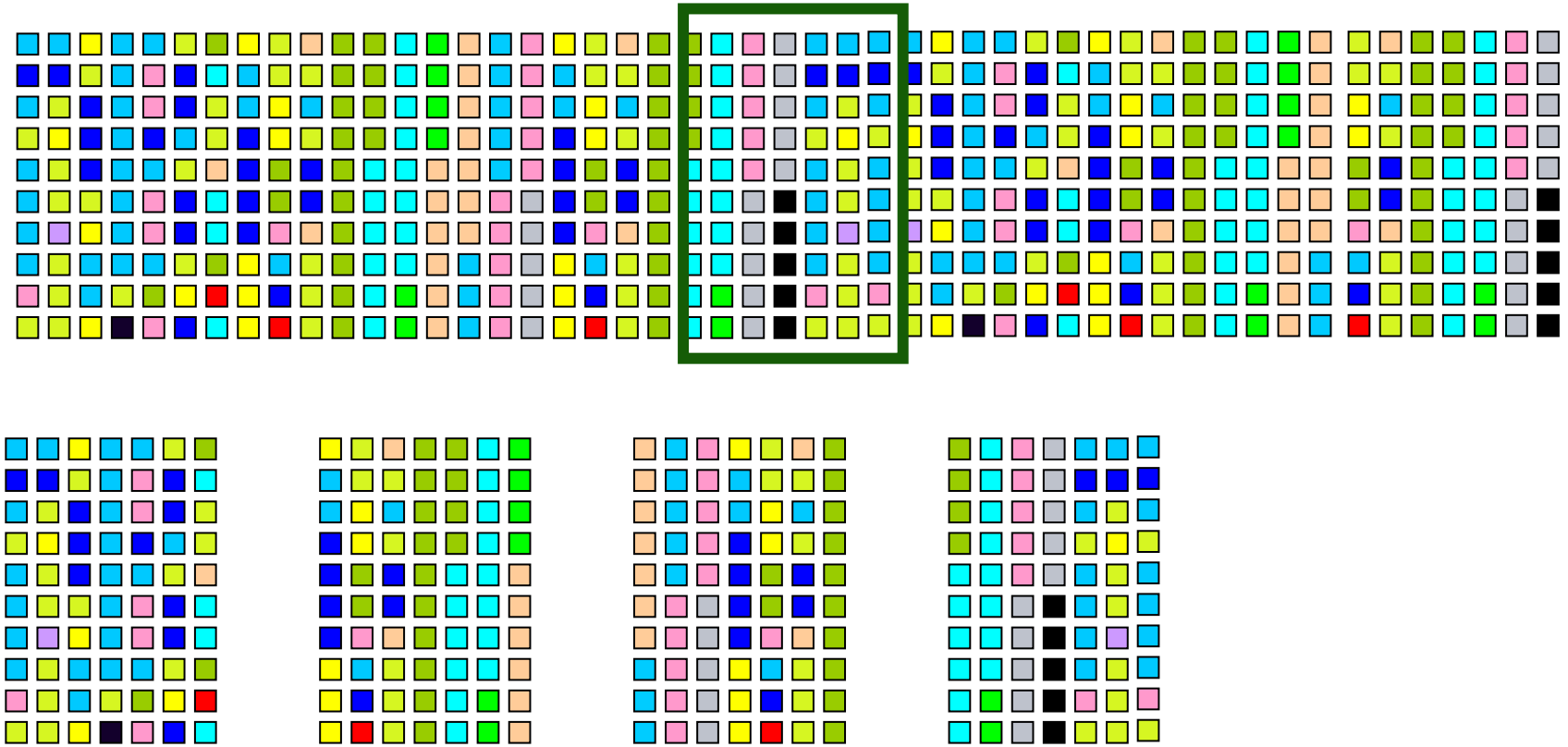
Tumbling windows



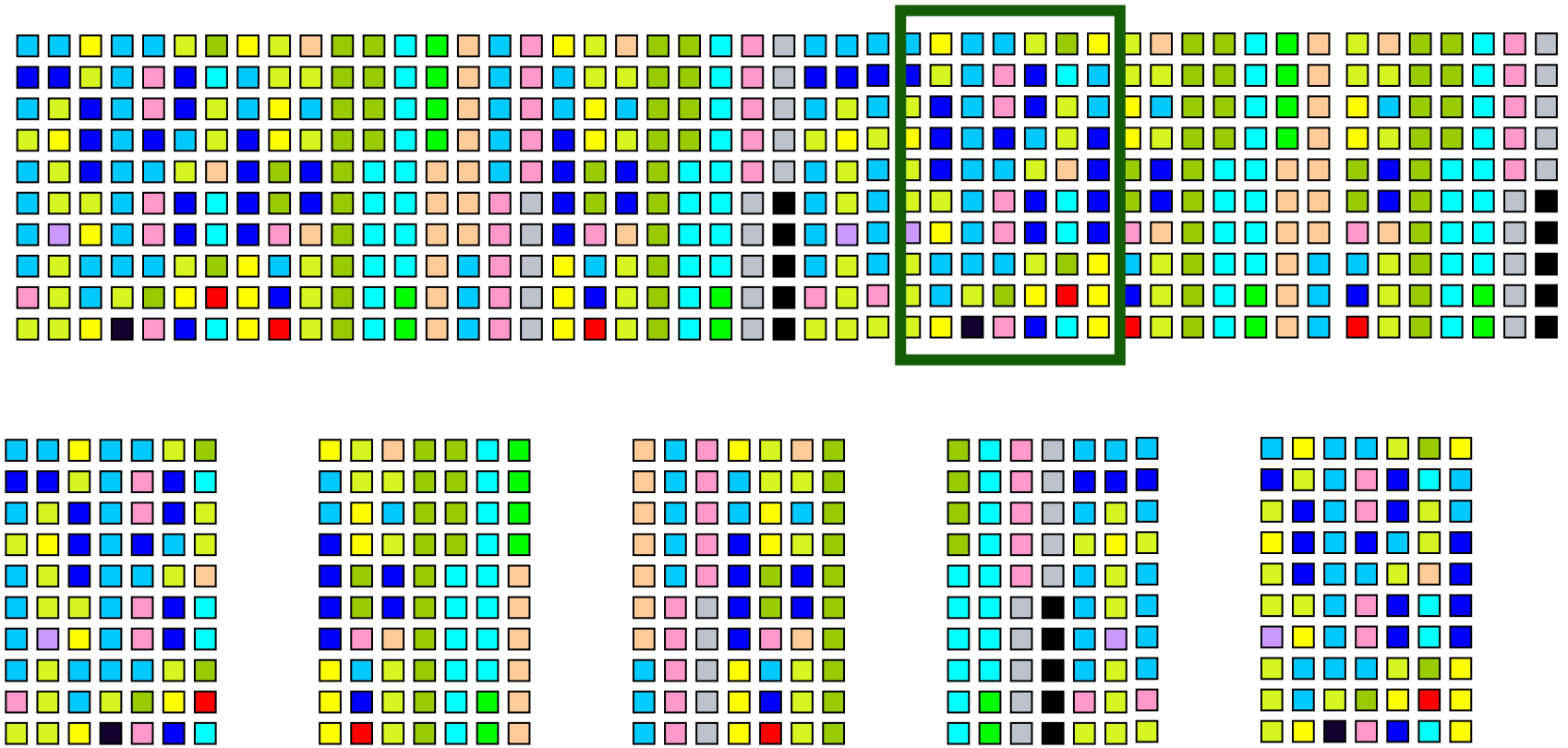


Tumbling windows

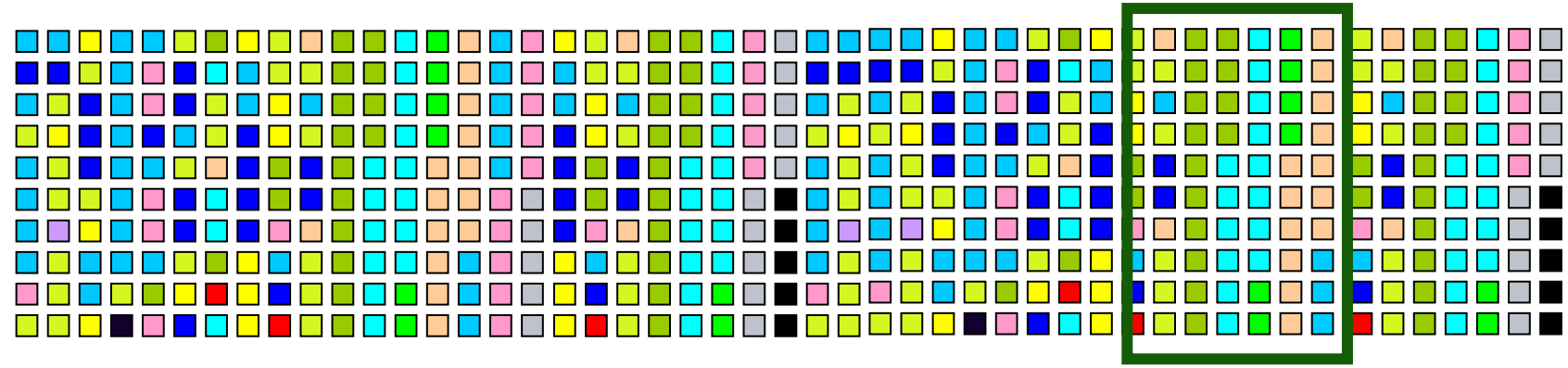
Tumbling windows

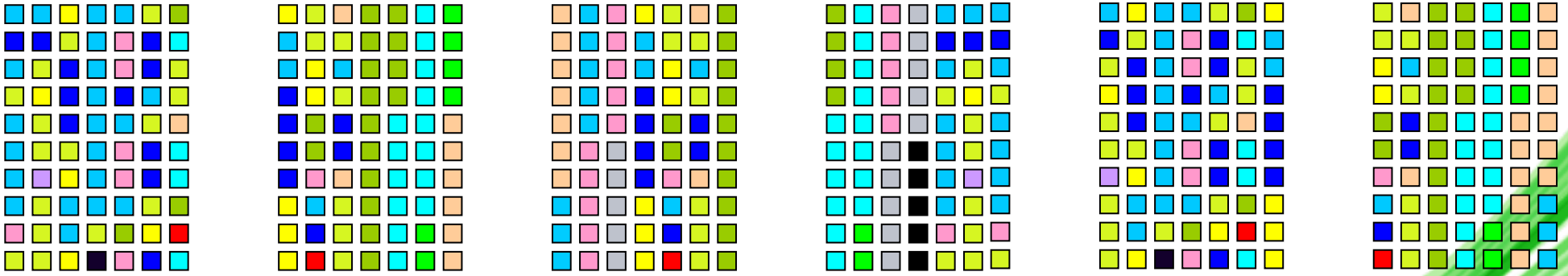


Tumbling windows



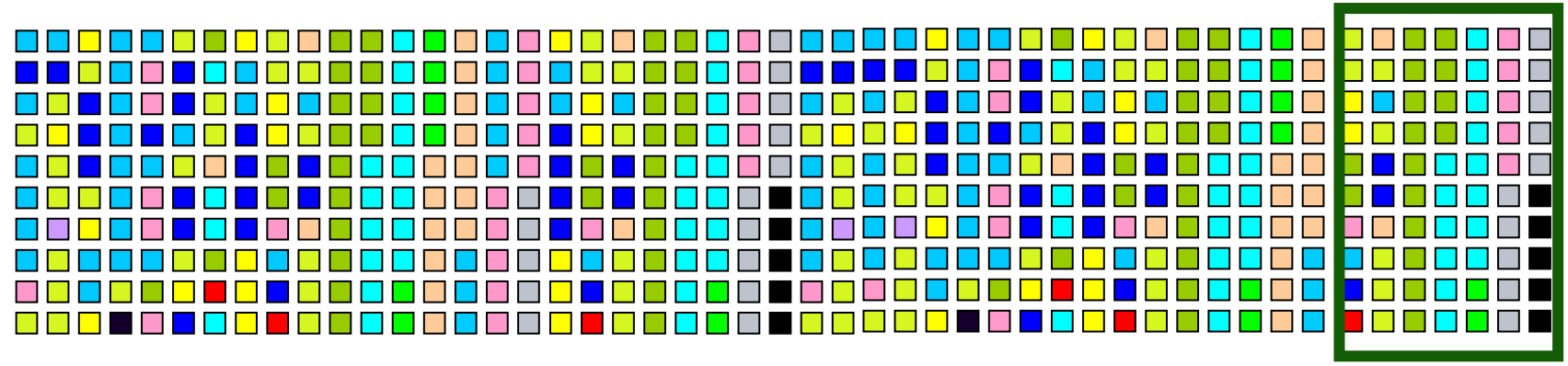
Tumbling windows

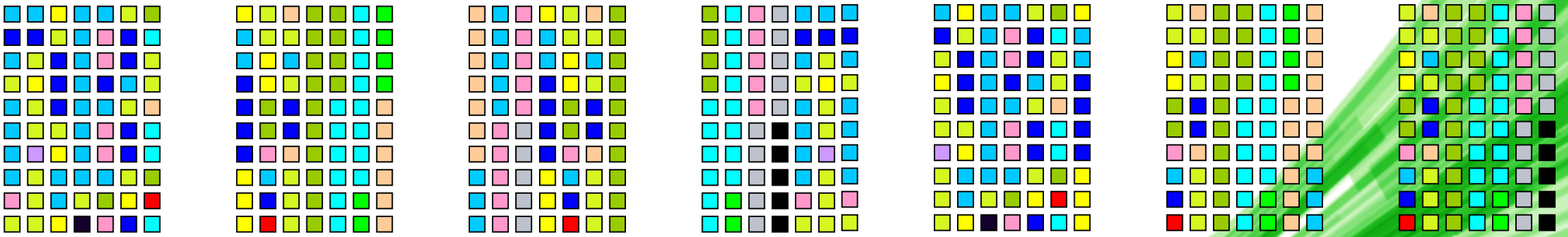




[6]

Tumbling windows





[1] [2] [3] [4] [5] [6] **[7]**

Recall the Query Process

For Each Table:

1. Convert incoming rows into 0 or more outgoing rows
2. Map outgoing rows to tables

_time	host	_field	_value
10:45	serverA	system	1000
11:00	serverA	system	2000
11:15	serverA	system	1000
11:30	serverA	system	2000
11:45	serverA	system	3000

GroupKey[host=serverA,_field=system]

_time	host	_field	_value
10:35	serverA	idle	100
11:15	serverA	idle	200
11:50	serverA	idle	400

GroupKey[host=serverA,_field=idle]

```
|>
aggregateWindow(
    every: 60m,
    fn: mean)
```

_time	host	_field	_value
10:00	serverA	system	1000

GroupKey[host=serverA,_field=system]

_time	host	_field	_value
10:00	serverA	idle	100

GroupKey[host=serverA,_field=idle]

_time	host	_field	_value
10:45	serverA	system	1000
11:00	serverA	system	2000
11:15	serverA	system	1000
11:30	serverA	system	2000
11:45	serverA	system	3000

GroupKey[host=serverA,_field=system]

_time	host	_field	_value
10:35	serverA	idle	100
11:15	serverA	idle	200
11:50	serverA	idle	400

GroupKey[host=serverA,_field=idle]

```
|>
aggregateWindow(
    every: 60m,
    fn: mean)
```

_time	host	_field	_value
10:00	serverA	system	1500
11:00	serverA	system	2000

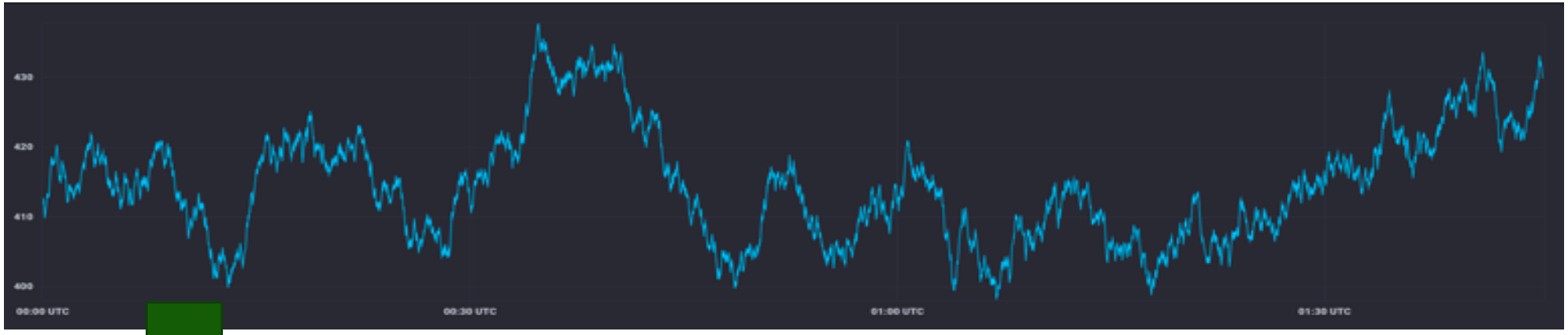
GroupKey[host=serverA,_field=system]

_time	host	_field	_value
10:00	serverA	idle	100
11:00	serverA	idle	300

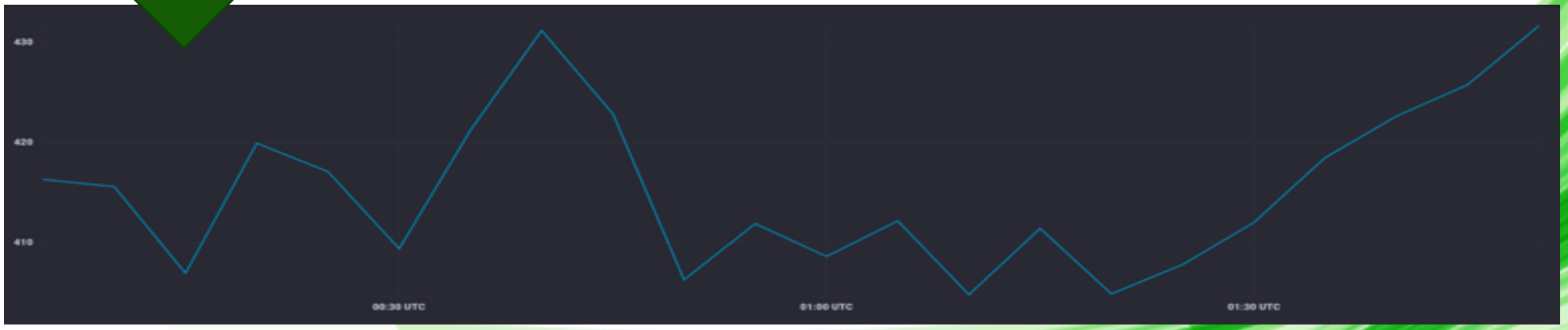
GroupKey[host=serverA,_field=idle]

Typical use case: downsampling data

Typical use case: downsampling data (cont.)



aggregateWindow(every: 5m, fn: mean)

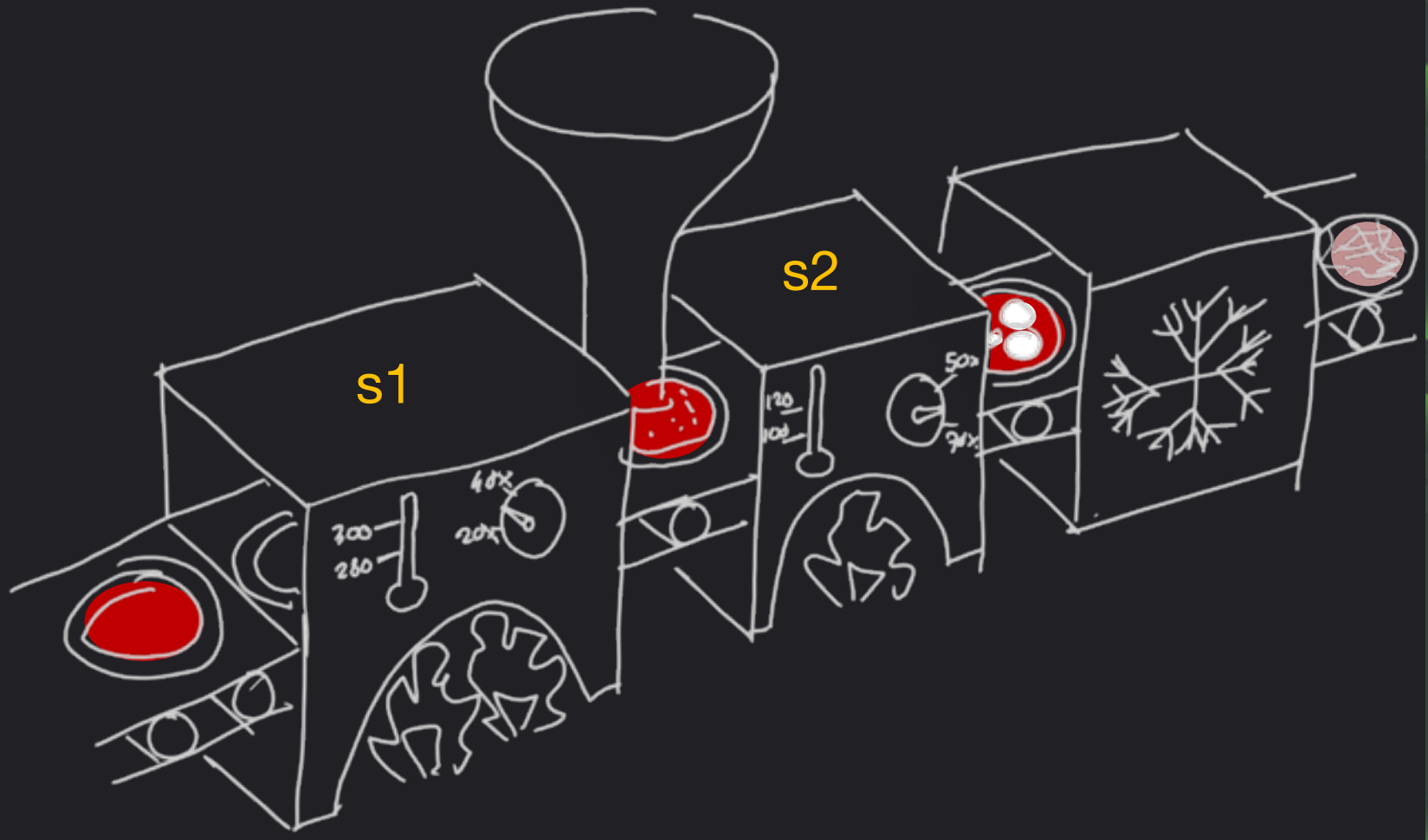


Let’s get dirty!

Continuous Linear Pizza Oven

Learning goals:

- aggregateWindow



Task

Extract the moving average of the temperatures observed in the cooking base area over a window of 2 minutes

Extract the moving average of the temperatures observed by S2 over a window of 3 minutes

Take home message

- The aggregateWindow() function eases the **downsampling** task
- **The createEmpty flag changes the behaviour of the aggregateWindow() function.**
  - **If createEmpty = false the aggregateWindow() function does not consider the empty windows in the aggregation**
  - **If createEmpty = true** (default) the aggregateWindow() function considers **all the windows**


influx/days