



# Streaming Data Engineering

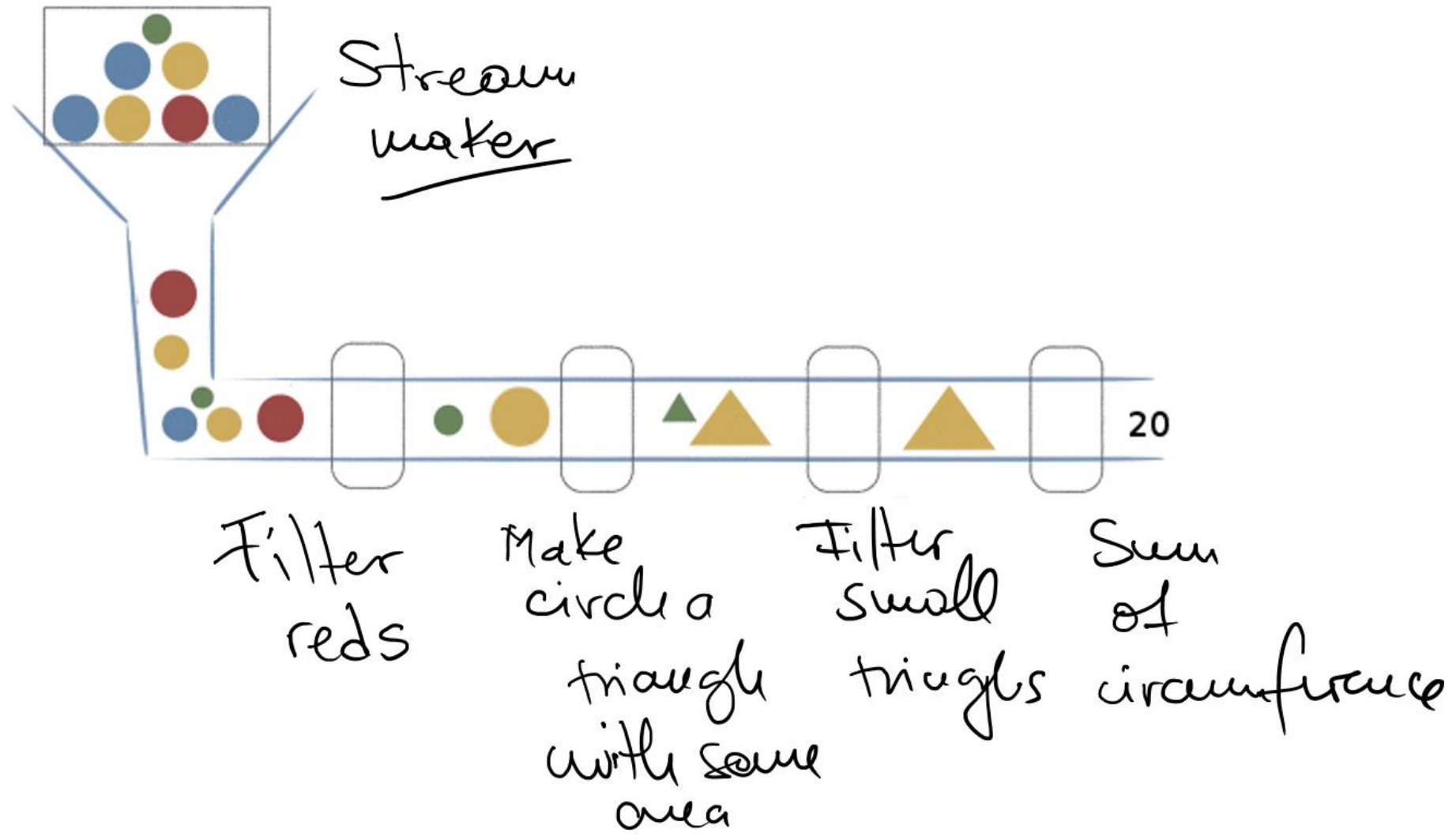
*Feature Extraction from Road Traffic Data*

# Outline

- Streaming Data Engineering
  - *Concepts*
  - *Functions*
- A concrete example: Road Traffic Monitoring and Analysis
  - *Applied concepts*
  - *Technology*
- Conclusions from practice

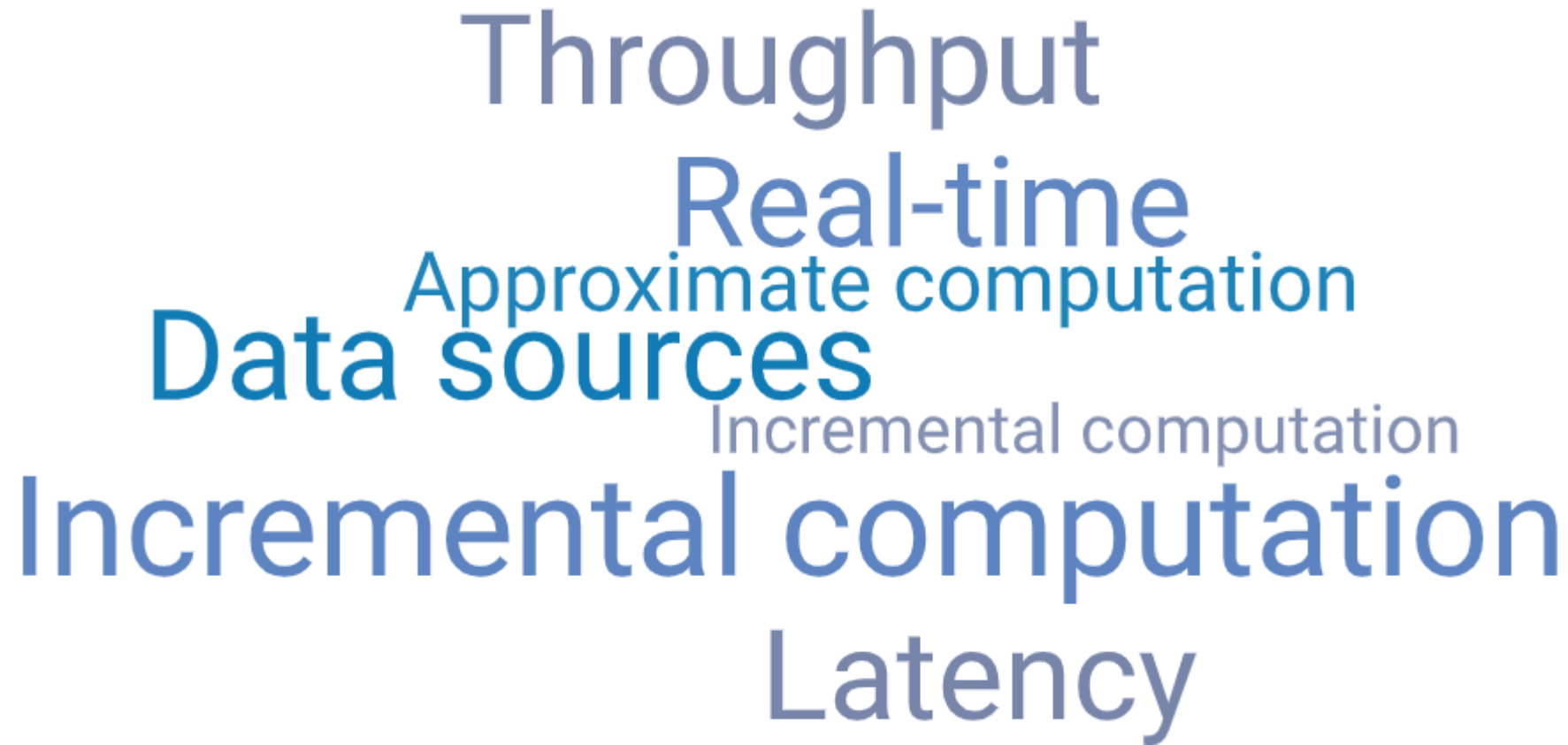
# Streaming Data Engineering

*An intuitive introduction*



# Streaming Data Engineering

*Concepts*



A word cloud of streaming data engineering concepts. The words are arranged in a roughly triangular shape, with 'Throughput' at the top, 'Real-time' and 'Approximate computation' in the middle, 'Data sources' and 'Incremental computation' below them, and 'Latency' at the bottom. The words are in various shades of blue and purple, with different font sizes and weights.

Throughput

Real-time

Approximate computation

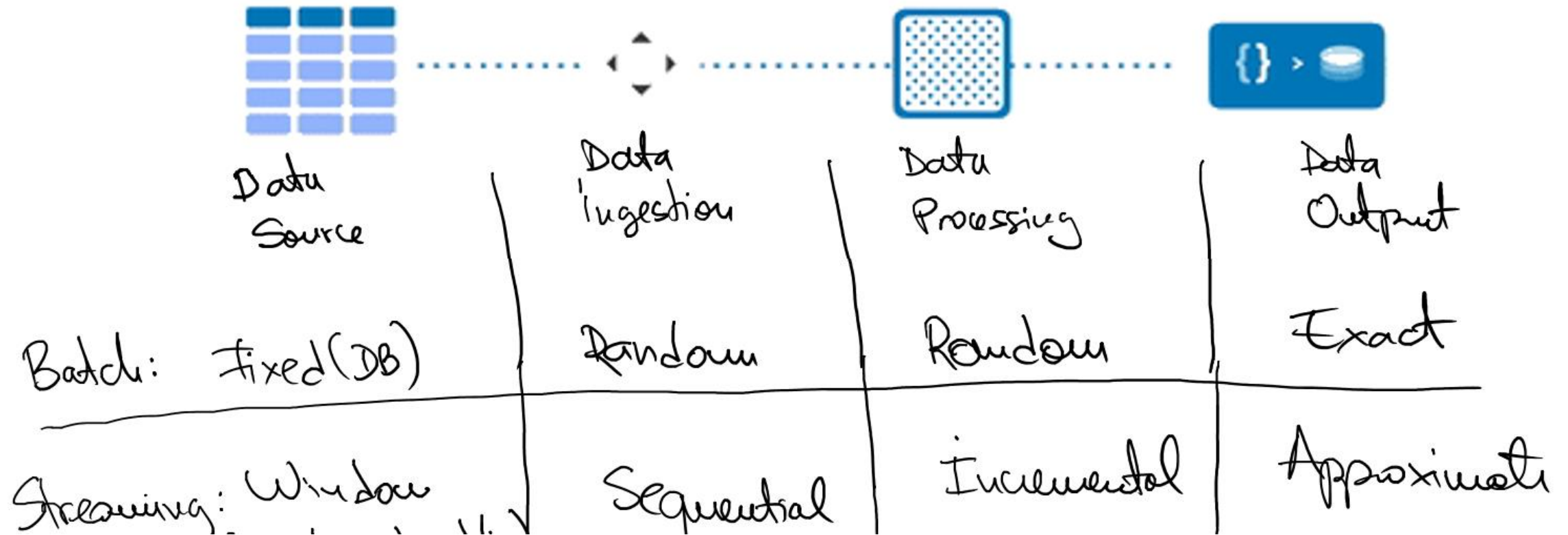
Data sources

Incremental computation

Latency

# Streaming Data Engineering

## Functions





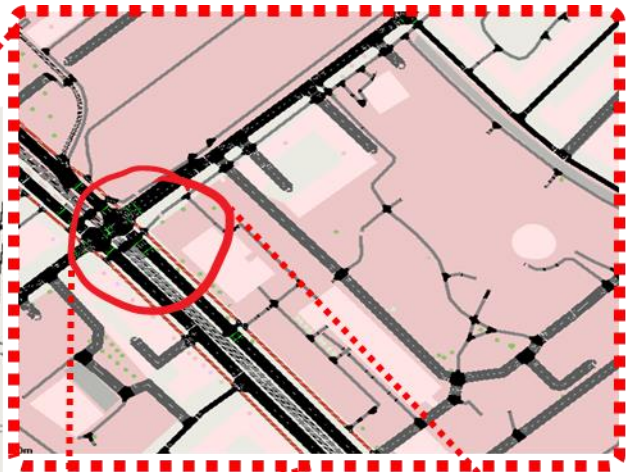
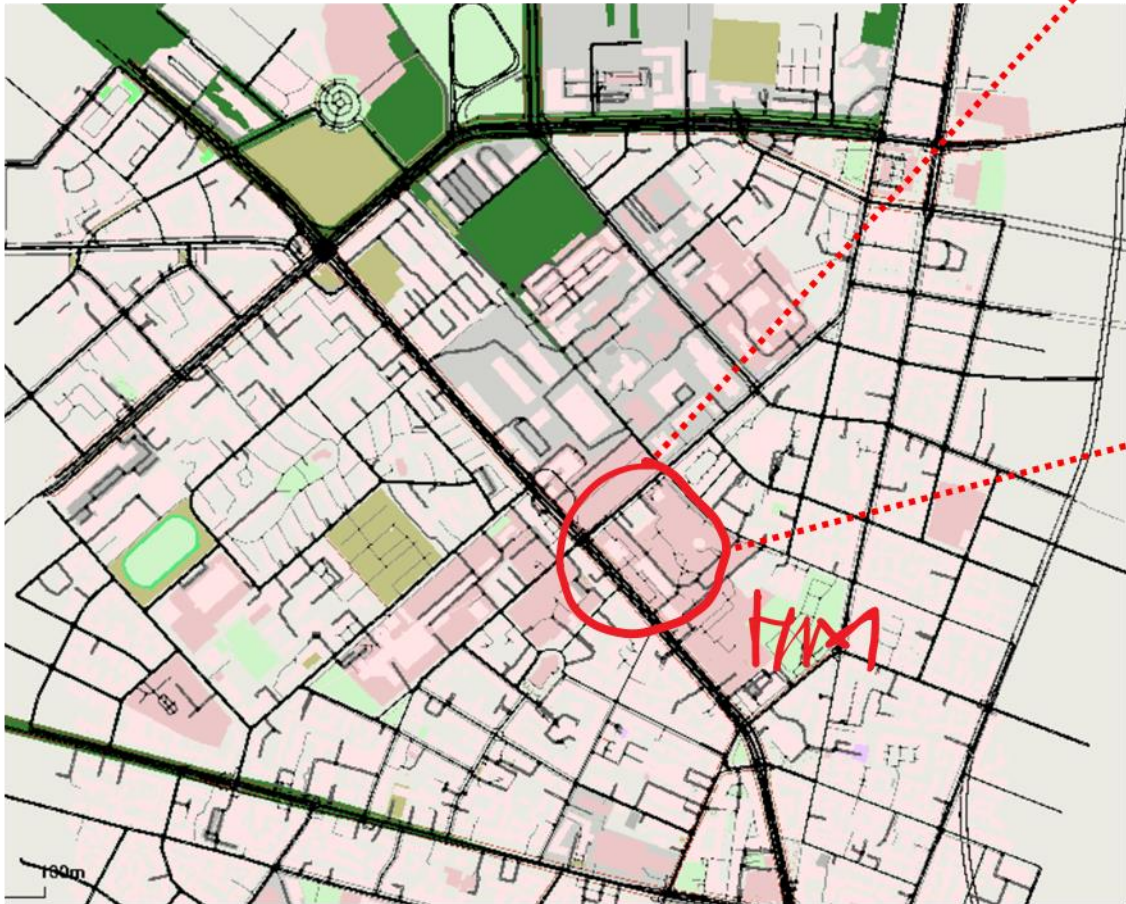
# Road Traffic Monitoring and Analysis



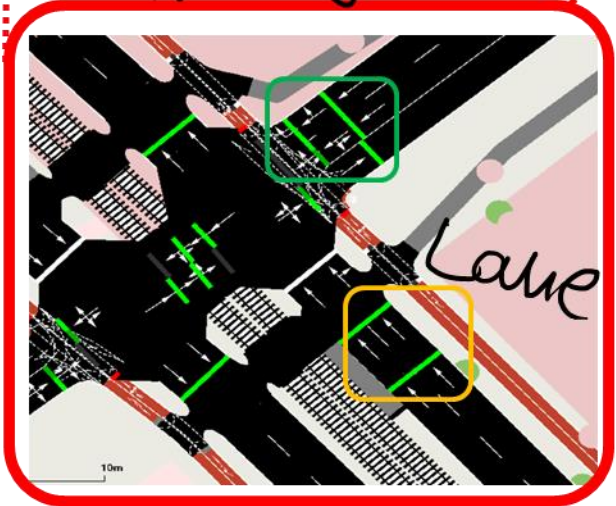


# Road Traffic Monitoring and Analysis

Demo



Traffic Lights

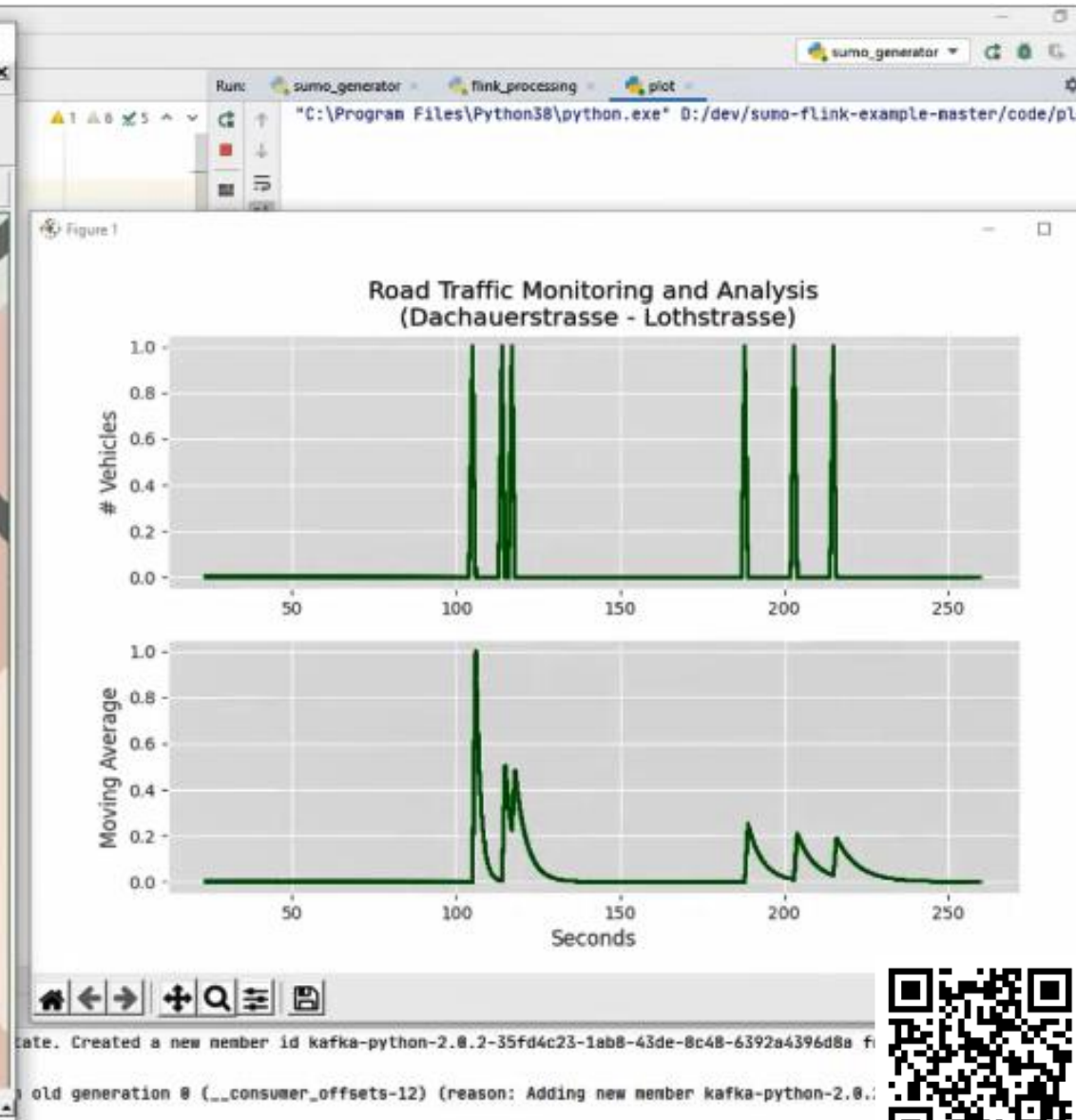
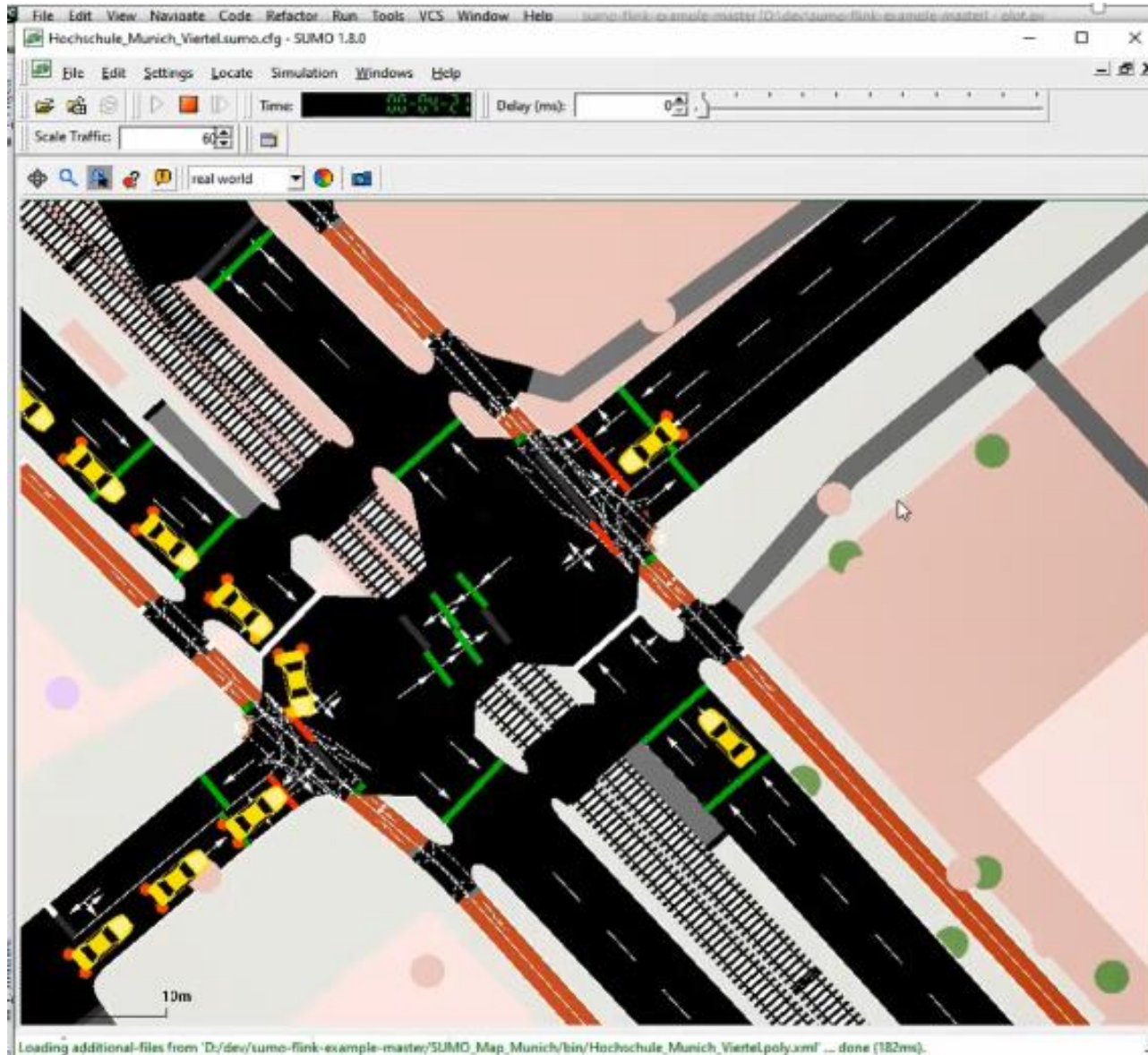


induction  
loops





# Road Traffic Monitoring and Analysis

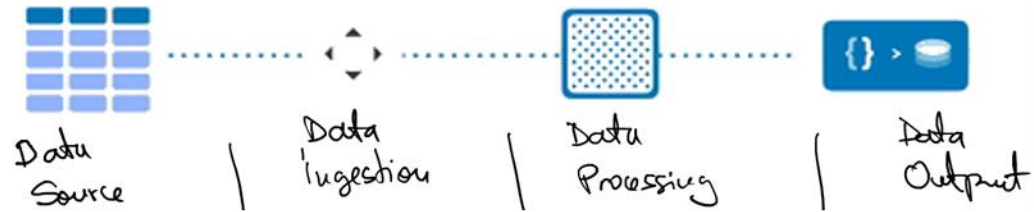




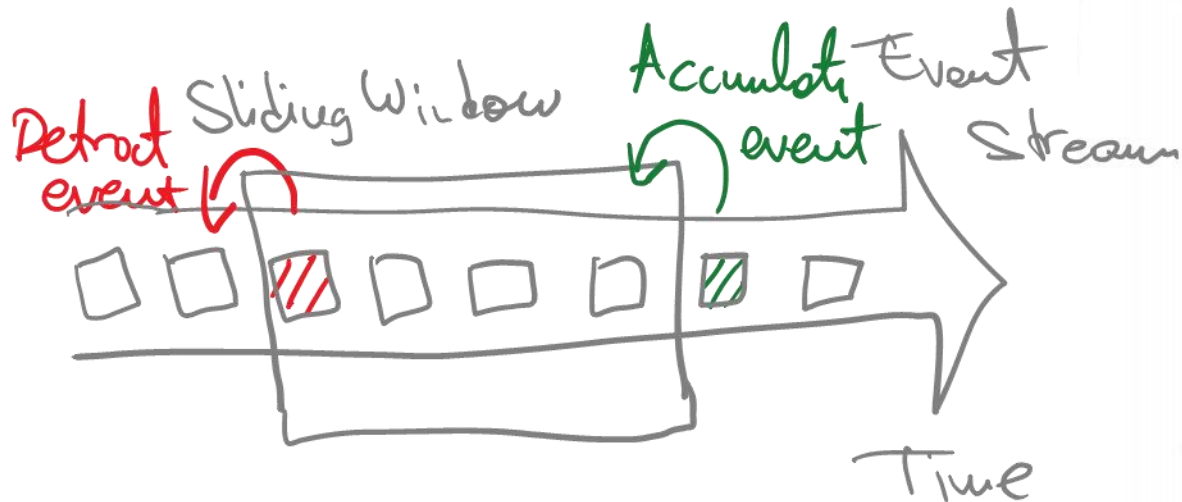
# Road Traffic Monitoring and Analysis

## Applied concepts

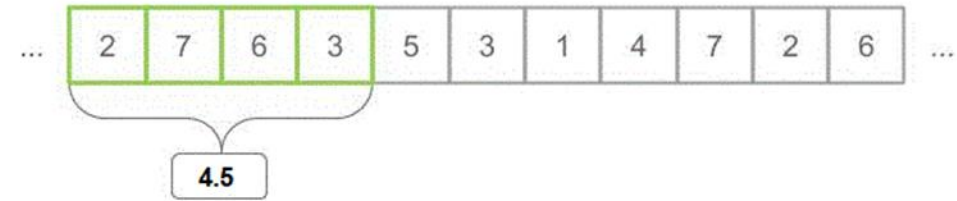
### Sliding window



### Principle



### Processing



Consider a real valued sample  $x_1, \dots, x_t, \dots, x_{t/k}$  for all  $i \in \{1, \dots, t, \dots\}$  drawn from the distribution of the random variable  $X$ .

The sample mean of the sample of size  $t$  is  $\bar{x}_t$

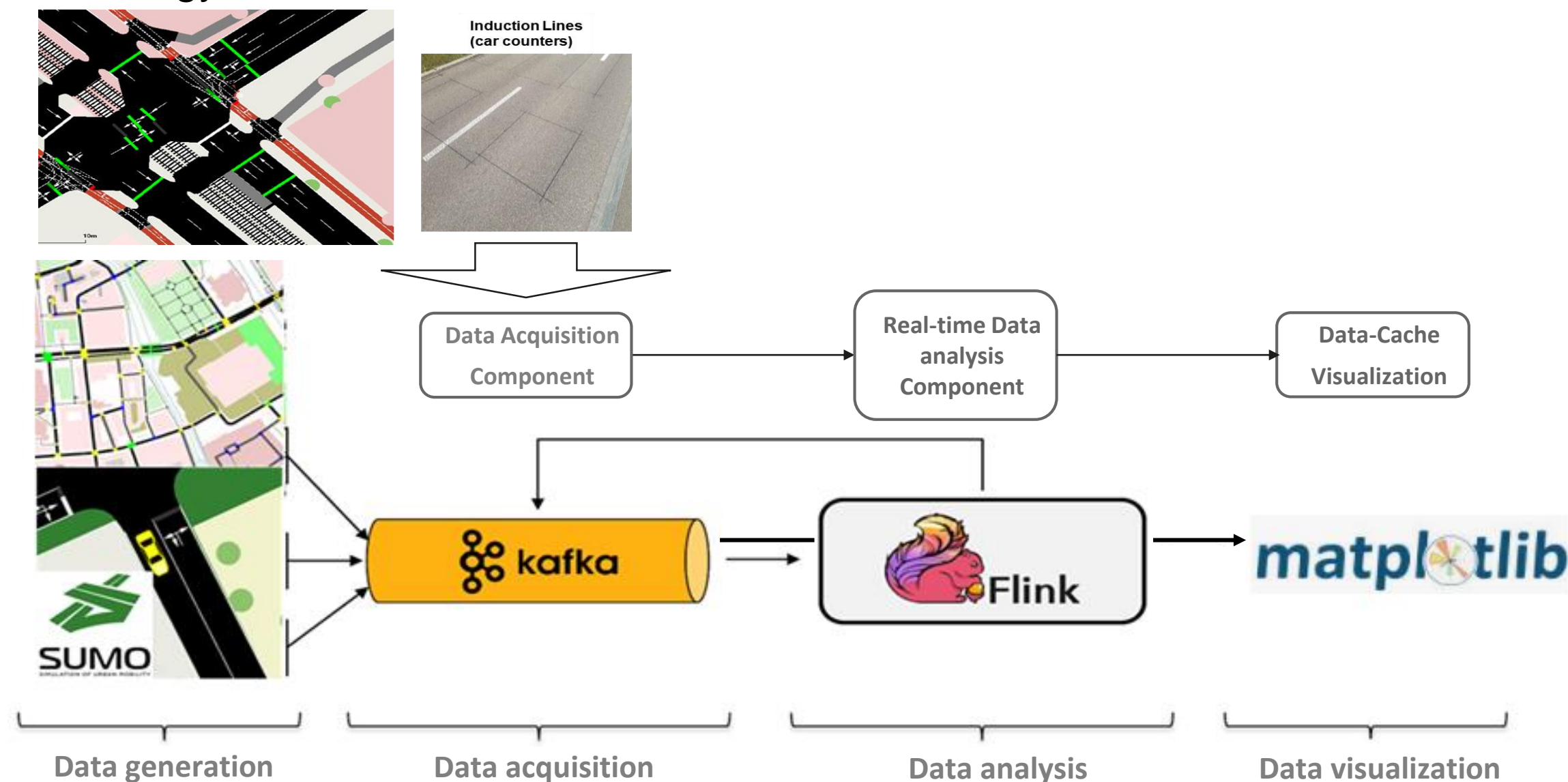
$$\bar{x}_t = \frac{1}{t} \cdot \sum_{i=1}^t x_i \quad \text{for incremental version}$$

$$\bar{x}_t = \frac{1}{t} \left( x_t + \sum_{i=1}^{t-1} x_i \right) = \frac{1}{t} \left( x_t + (t-1)\bar{x}_{t-1} \right)$$

$$\bar{x}_t = \bar{x}_{t-1} + \frac{1}{t} (x_t - \bar{x}_{t-1}) \quad \text{① mean}$$

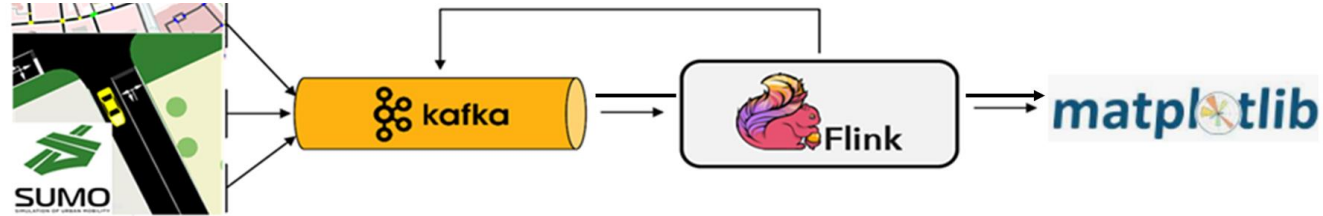
# Road Traffic Monitoring and Analysis

## Technology



# Road Traffic Monitoring and Analysis

## Technology



```
sumo-flink-example-master | code | flink_processing.py
sumo_generator.py | flink_processing.py | plot.py

1 from pyflink.datastream import StreamExecutionEnvironment
2 from pyflink.table import StreamTableEnvironment, EnvironmentSettings
3
4 # Setup the Flink
5 def flink_processing():
6     env = StreamExecutionEnvironment.get_execution_environment()
7     env.set_parallelism(1)
8     env_settings = EnvironmentSettings.Builder().use_blink_planner().build()
9     t_env = StreamTableEnvironment.create(stream_execution_environment=env,
10                                         environment_settings=env_settings)
11
12     t_env.get_config().get_configuration().set_string(
13         "pipeline.jar",
14         "file:///D:/temp/kafka_2.12-2.7.0/flink-connector-kafka_2.11-1.12.0.jar;"
15         "file:///D:/temp/kafka_2.12-2.7.0/flink-sql-connector-kafka_2.11-1.12.0.jar"
16     )
17
18     source_ddl = """
19         CREATE TABLE source_num(
20             ts TIMESTAMP(3) METADATA FROM 'timestamp',
21             step FLOAT,
22             edge_id STRING,
23             vehicle_num INT
24         ) WITH (
25             'connector' = 'kafka',
26             'topic' = 'source_num',
27             'properties.bootstrap.servers' = 'localhost:9092',
28             'properties.group.id' = 'new_group2',
29             'format' = 'json'
30         )
31     """
32
33     sink_ddl = """
34         CREATE TABLE sink_table_num(
35             ts TIMESTAMP(3) METADATA FROM 'timestamp',
36             step FLOAT,
37             edge_id STRING,
38             vehicle_num INT
39         ) WITH (
40             'connector' = 'kafka',
41             'topic' = 'sink_table_num',
42             'properties.bootstrap.servers' = 'localhost:9092',
43             'properties.group.id' = 'new_group2',
44             'format' = 'json'
45         )
46     """
47
48     t_env.execute_sql(source_ddl)
49     t_env.execute_sql(sink_ddl)
50
51     # Create a source table
52     source_num = t_env.create_table(source_ddl)
53
54     # Create a sink table
55     sink_table_num = t_env.create_table(sink_ddl)
56
57     # Write data to the sink table
58     source_num.to(sink_table_num)
```

Terminal: kafka-server --zookeeper-server --

[2021-02-02 13:40:22,334] INFO [GroupCoordinator 0]: Dynamic Member with unknown member id joins group my-group in Empty state. Created a new member id kafka-python-2.0.2-35fd4c23-1ab8-43de-8c48-6392a4396d8a for this member and add to the group. (kafka.coordinator.group.GroupCoordinator)

[2021-02-02 13:40:22,344] INFO [GroupCoordinator 0]: Specified to rebalance group my-group in state SpecifiedRebalance with old generation 0 / consumer offsets: 19 / reason: kafka-python-2.0.2-35fd4c23-1ab8-43de-8c48-6392a4396d8a





# Conlcusions from practice

Offenbach



Shenzen



## In a **real application**

(China, Shenzen, Bantian Neighborhood,  
8 intersections, 28 traffic light controlled street lanes):

- **Data collected by cameras and induction loops every second (280 Kafka messages/s).**
- **Processing steps (SELECT - FROM - WHERE – ML) sequentially.**
- **Data is processed immediately and incrementally** (e.g. counting the number of cars passing, queue length, flow, congestion index).
- Processing with **low-latency (0.03s/traffic light)** and **high throughput (28 traffic light control signals/s).**



# Streaming Data Engineering

*Feature Extraction  
from  
Road Traffic  
Data*

