
# Progetto Batch processing

SABD 2018-2019

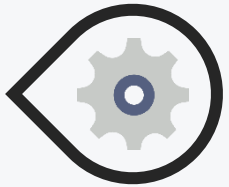
Montesano,Perrone,Pusceddu

1 Architettura

Data Ingestion 2

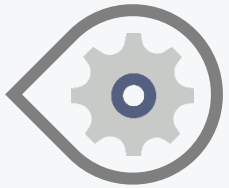
3 Query1 Core & SQL

Query2 Core & SQL 4

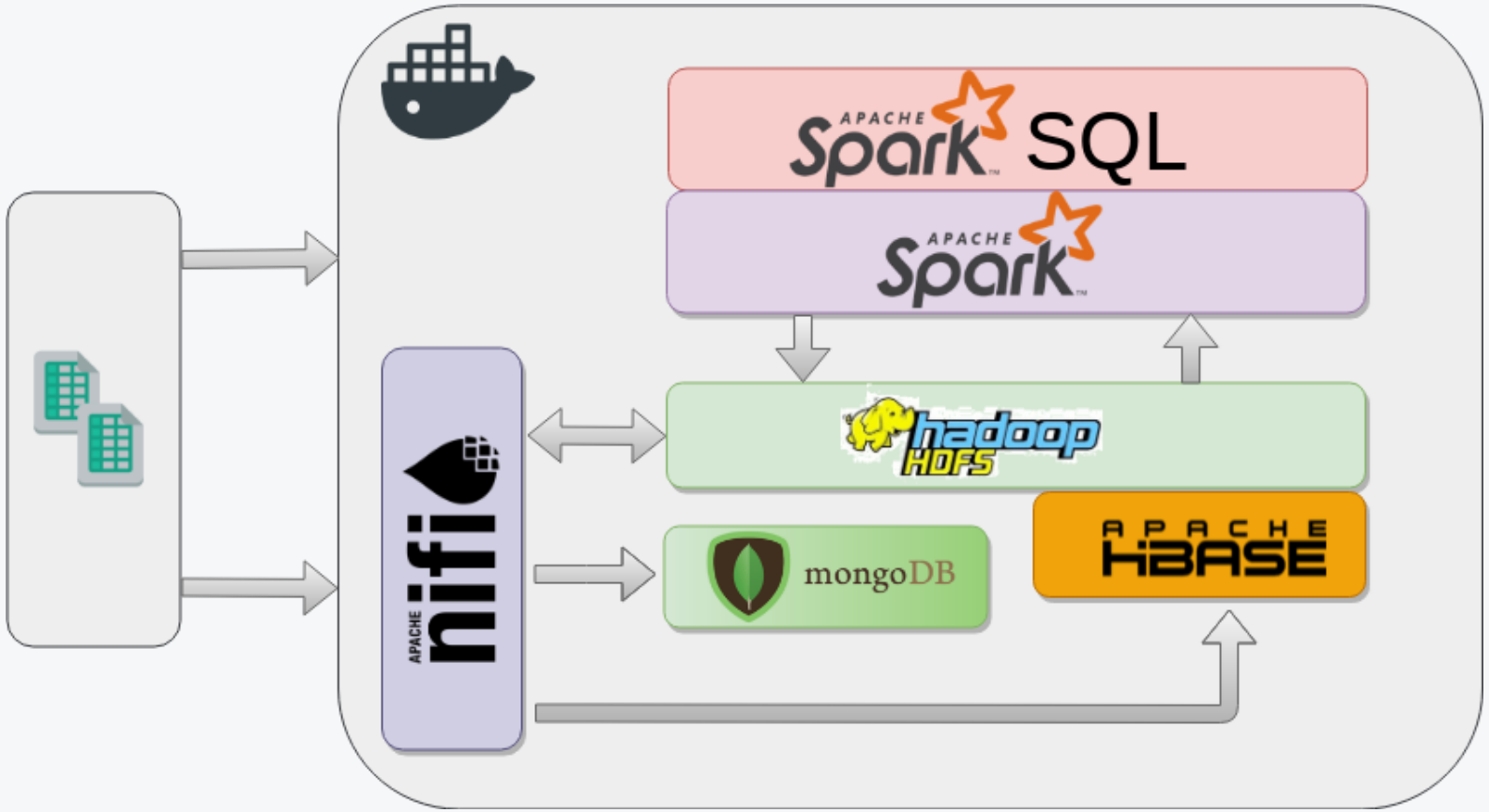


5 Query3 Core & SQL

Discussione dei risultati 6



# Architettura




- Apache Spark esegue con un master e due worker con cluster manager locale
- Ogni worker ha 1Gb di memoria
- Linguaggio utilizzato: Java
- Hdfs esegue con un master e 3 worker

# Local Deployment

- Il deploy in locale è costruito utilizzando le Docker Image
- Vengono avviati i container di : Hdfs, Nifi, Hbase, MongoDB ,Spark
- I container lavorano in modalità cluster
- I container vengono eseguiti sotto la stessa rete denominata: net

# Data Ingestion

- L’ingestion dei dati è stato realizzato con il framework Apache Nifi
- Effettua l’iniezione dei dati nel file system distribuito HDFS in due diversi formati : csv e parquet
- Trasferisce i risultati delle query verso due database non relazionali Apache Hbase e Mongodb

# Scrittura risultati

I risultati vengono salvati su due database in formato Json. Sono stati usati sia Hbase e MongoDb a scopo didattico.

- Per MongoDB è stata creata una collection diversa per ogni query per ogni modalità di spark: Core ed SQL

- Per Hbase sono state create 3 tabelle, una per query
- Le tabelle per la query 1 e query 3 sono state divise in due famiglie di colonne: Core ed Sql
- La tabella della query 2 è stata divisa in 6 famiglia di colonne,3 per i file di Spark Core e 3 per i file di Spark SQL

# QUERY

1. Per ogni anno del dataset individuare le città che hanno almeno 15 giorni al mese di tempo sereno nei mesi di marzo, aprile e maggio.

2. Individuare, per ogni nazione, la media, la deviazione standard, il minimo, il massimo della temperatura, della pressione e dell'umidità registrata in ogni mese di ogni anno.

3. Individuare, per ogni nazione, le 3 città che hanno registrato nel 2017 la massima differenza di temperature medie nella fascia oraria locale 12:00-15:00 nei mesi di giugno, luglio, agosto e settembre rispetto ai mesi di gennaio, febbraio, marzo e aprile. Confrontare la posizione delle città nella classifica dell'anno precedente (2016).

# Pre-Processing

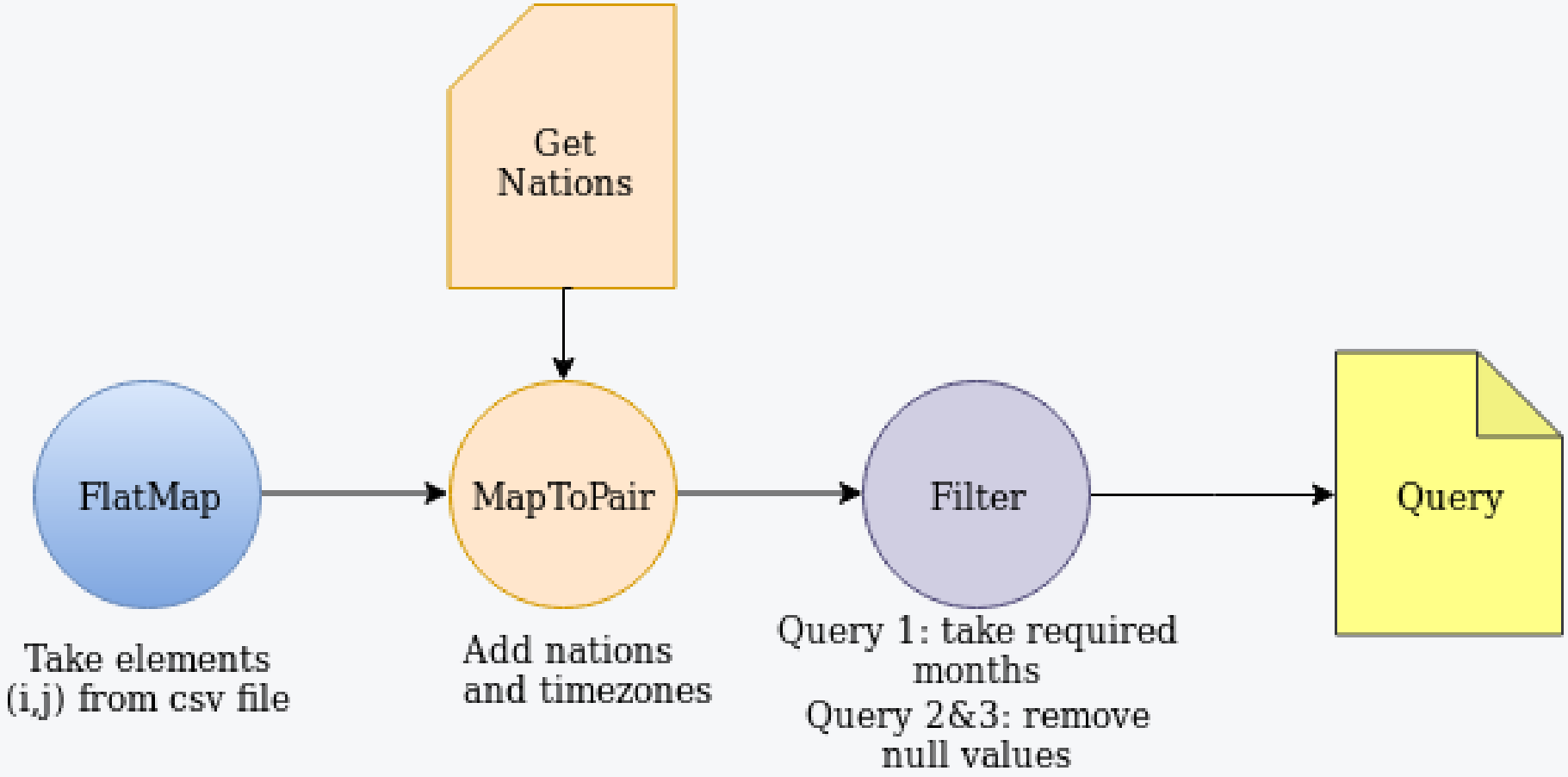
Step by step

1. Fase di pre processamento comune a tutte le query per ripulire i dati spuri

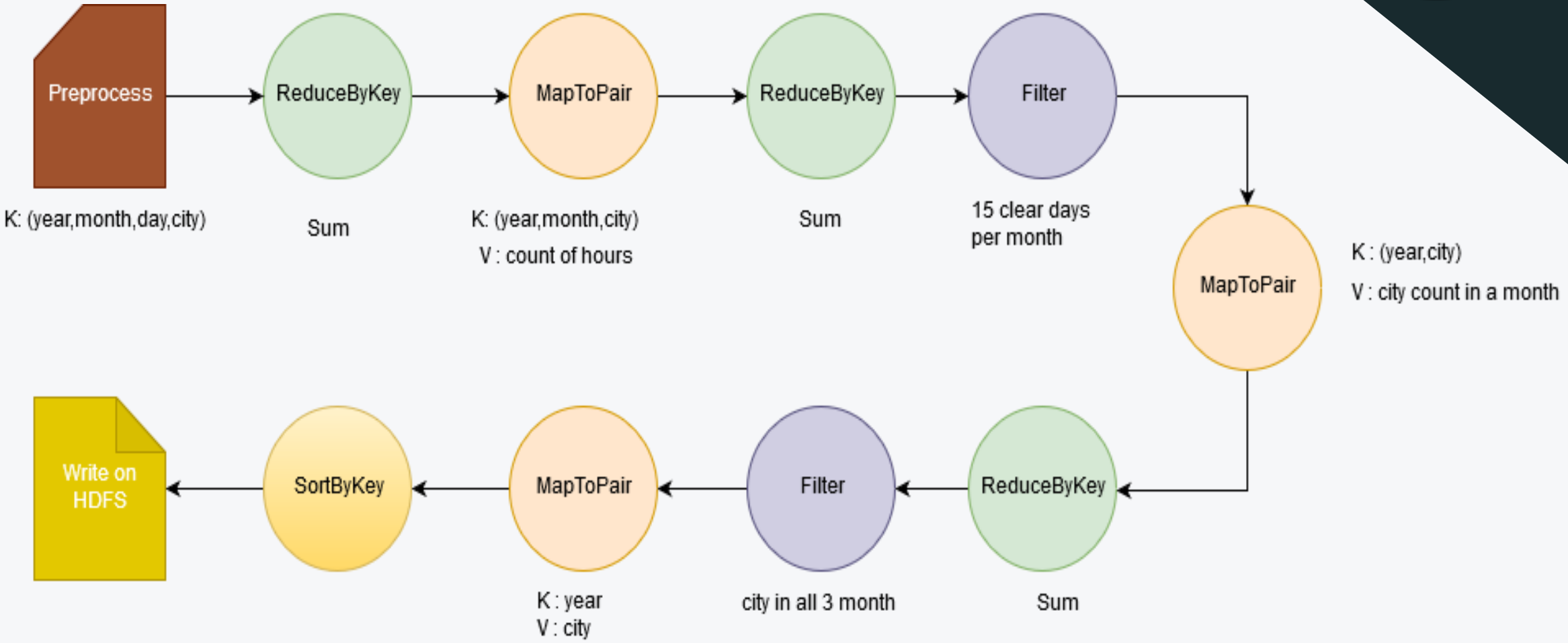
2. Chiamate REST al servizio Geonames per ottenere le nazioni tramite le coordinate e i fusi orari

3. Pre processamento dedicato alla specifica query. Per la query 1 si filtrano i mesi di interesse, per le query 2 e 3 si filtrano i valori non conformi

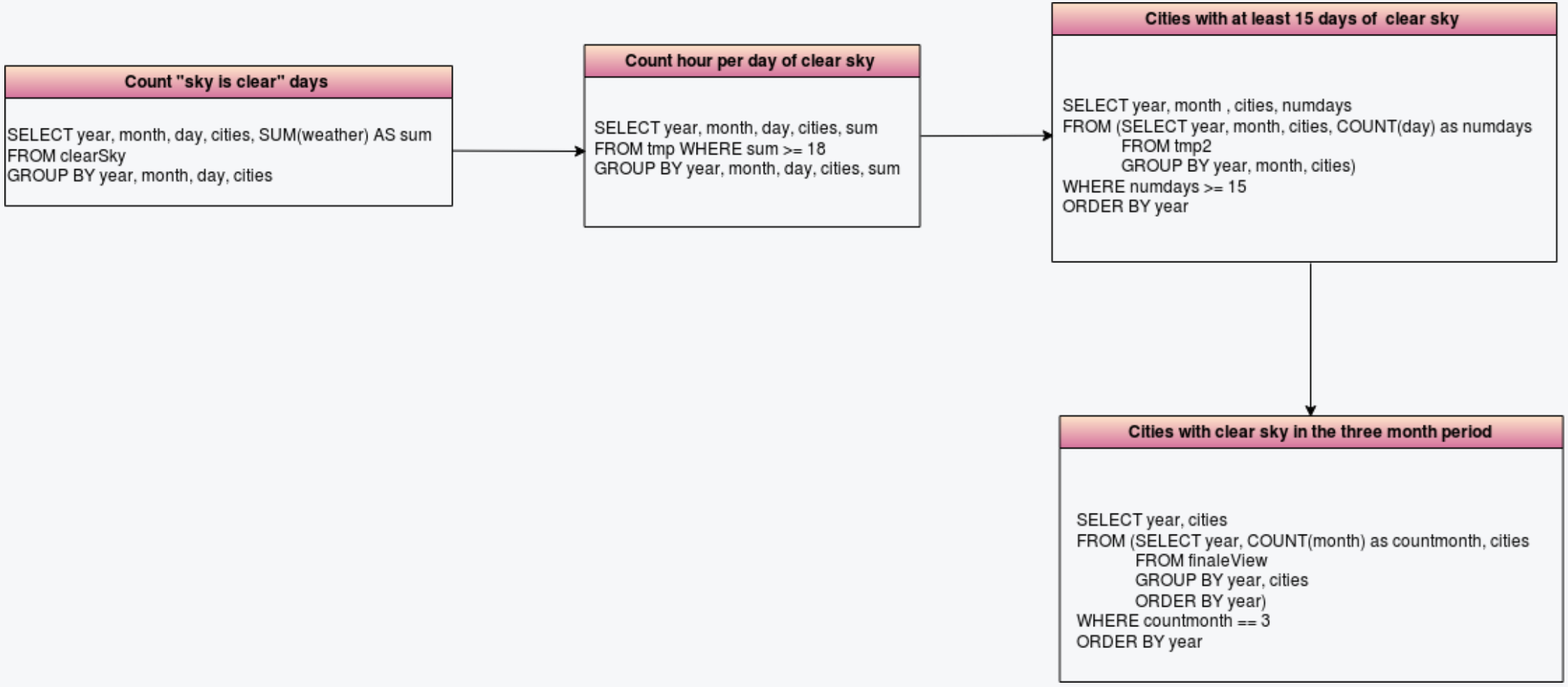
# Pre-Processing



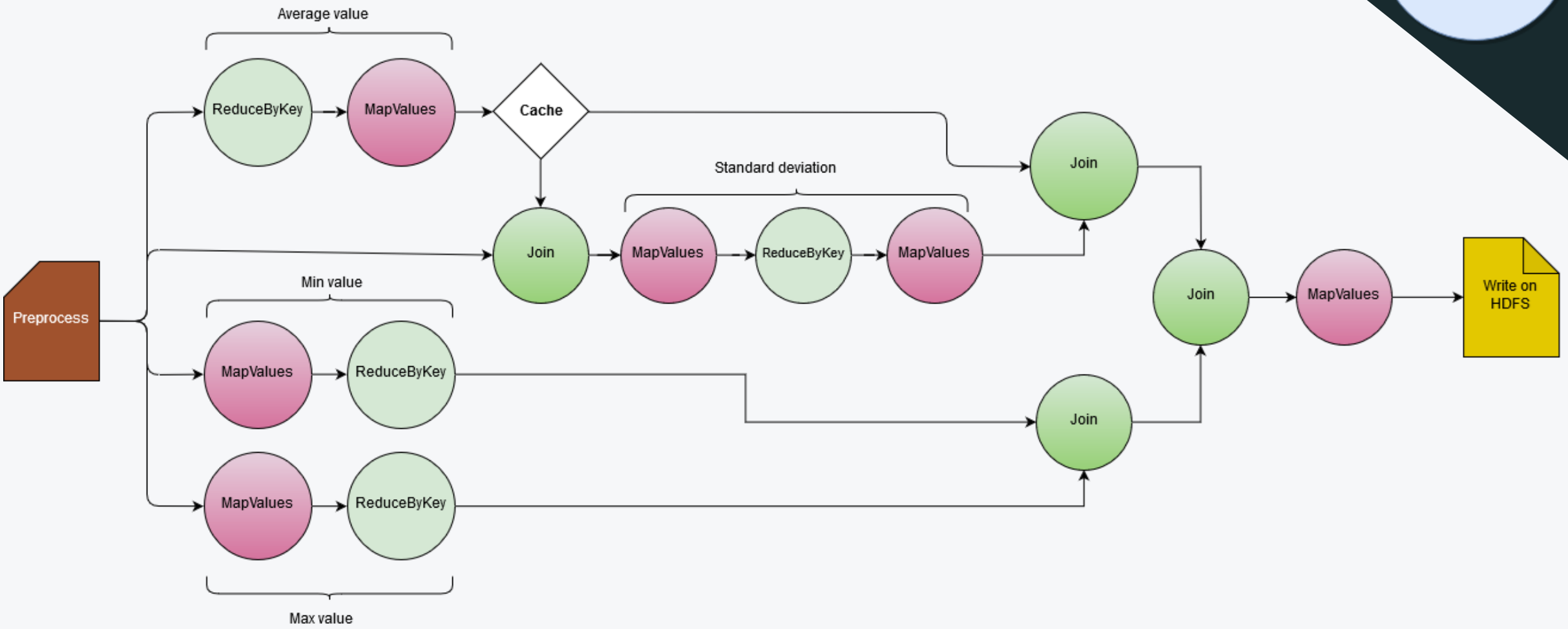
# Query 1


Preprocess
K: (year,month,day,city)
ReduceByKey
Sum
MapToPair
K: (year,month,city)
V : count of hours
ReduceByKey
Sum
Filter
15 clear days per month
MapToPair
K : (year,city)
V : city count in a month
ReduceByKey
Sum
Filter
city in all 3 month
MapToPair
K : year
V : city
SortByKey
Write on HDFS

# Query 1 SQL


Count "sky is clear" days
SELECT year, month, day, cities, SUM(weather) AS sum
FROM clearSky
GROUP BY year, month, day, cities
Count hour per day of clear sky
SELECT year, month, day, cities, sum
FROM tmp WHERE sum >= 18
GROUP BY year, month, day, cities, sum
Cities with at least 15 days of clear sky
SELECT year, month , cities, numdays
FROM (SELECT year, month, cities, COUNT(day) as numdays
FROM tmp2
GROUP BY year, month, cities)
WHERE numdays >= 15
ORDER BY year
Cities with clear sky in the three month period
SELECT year, cities
FROM (SELECT year, COUNT(month) as countmonth, cities
FROM finaleView
GROUP BY year, cities
ORDER BY year)
WHERE countmonth == 3
ORDER BY year

# Query 2

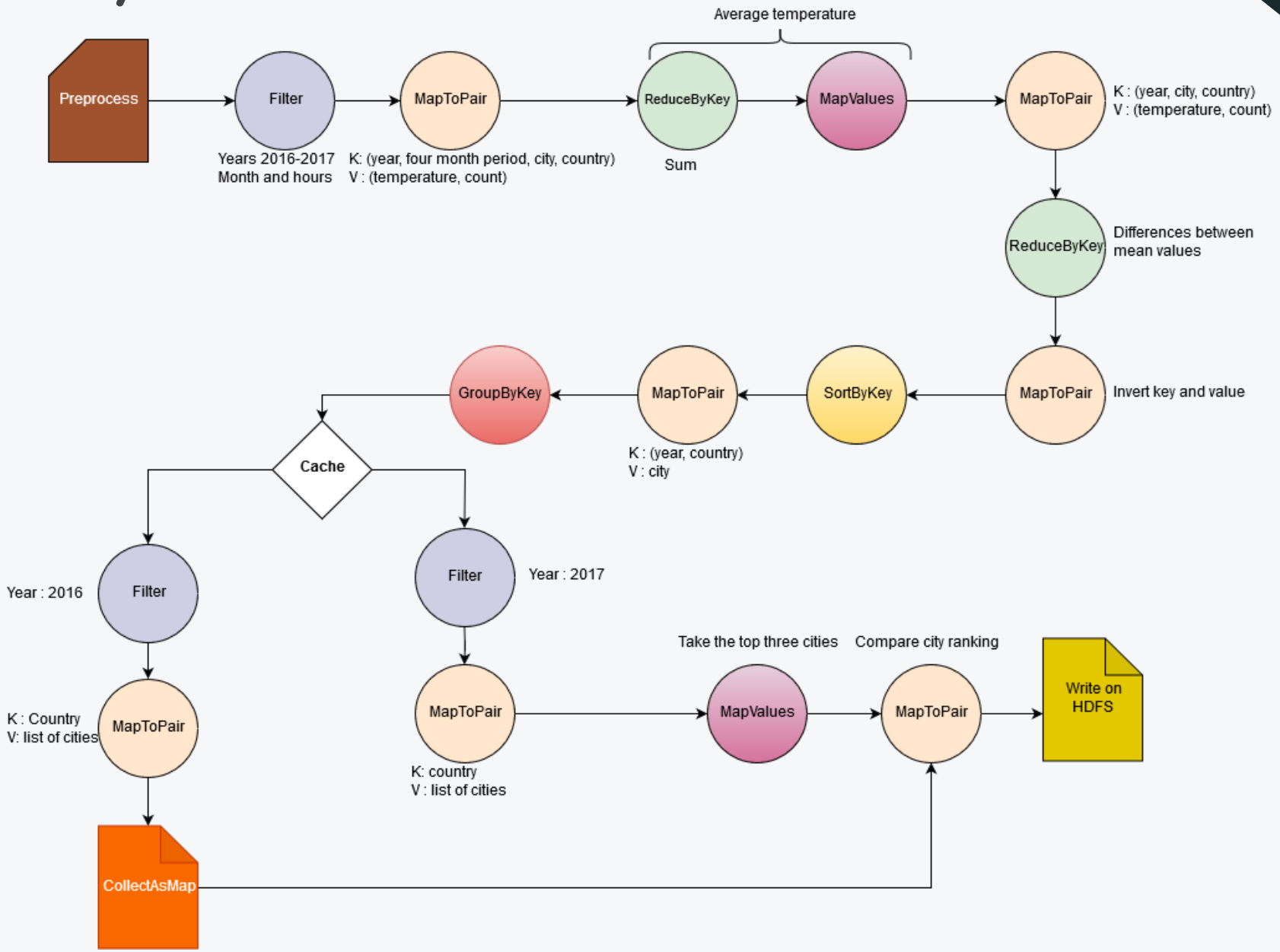

Average value
ReduceByKey
MapValues
Cache
Join
Standard deviation
Join
MapValues
ReduceByKey
MapValues
Preprocess
Min value
MapValues
ReduceByKey
Join
Join
MapValues
Write on HDFS
MapValues
ReduceByKey
Max value

# Query 2 SQL

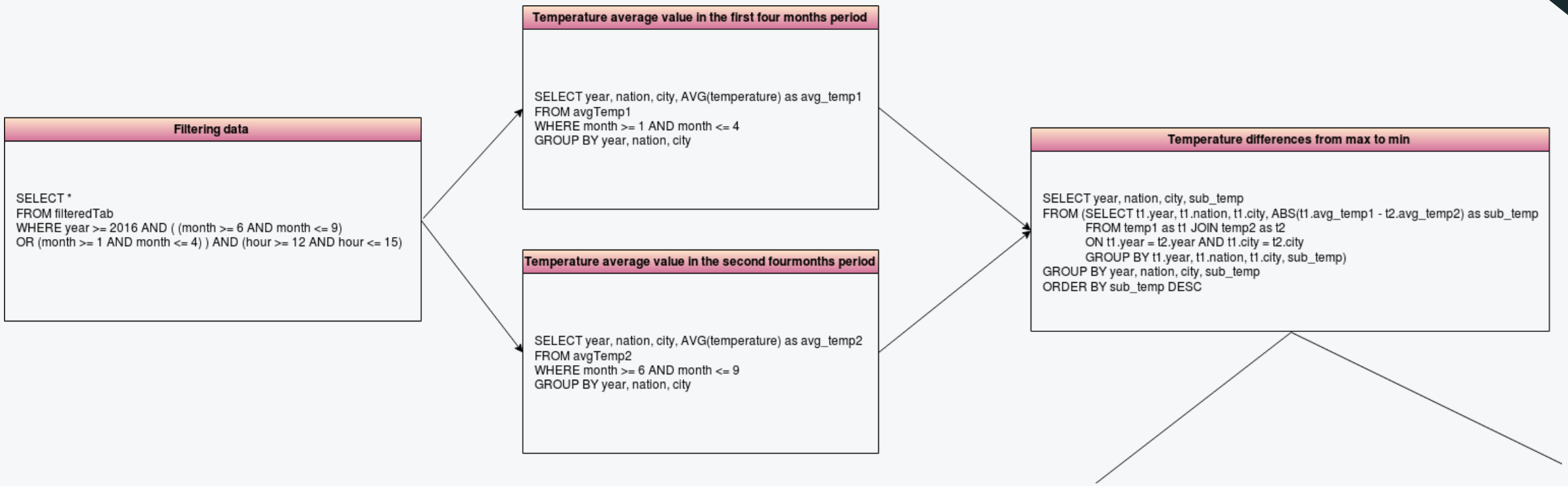
**Statistics on humidity, temperature and pressure values**

```
SELECT country,year, month, MEAN(value) AS mean ,MIN(value) as min, MAX(value) as max, STDDEV_SAMP(value) as stddev
FROM statistics
GROUP BY country,year,month
```

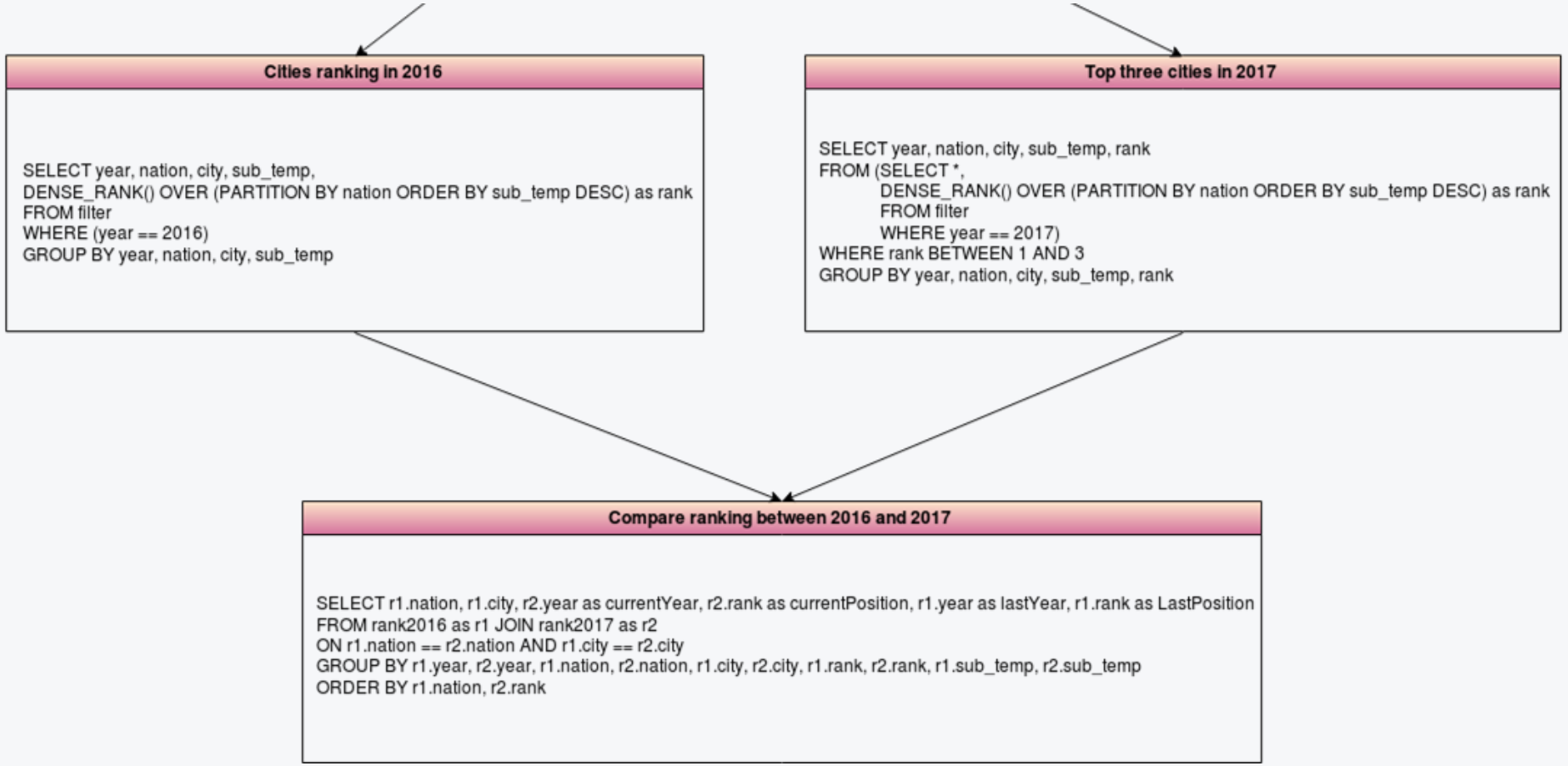
# Query 3


Average temperature
Preprocess
Filter
Years 2016-2017
Month and hours
MapToPair
K: (year, four month period, city, country)
V : (temperature, count)
ReduceByKey
Sum
MapValues
MapToPair
K : (year, city, country)
V : (temperature, count)
ReduceByKey
Differences between mean values
MapToPair
Invert key and value
SortByKey
MapToPair
K : (year, country)
V : city
GroupByKey
Cache
Year : 2016
Filter
K : Country
V: list of cities
MapToPair
CollectAsMap
Filter
Year : 2017
MapToPair
K: country
V : list of cities
Take the top three cities
MapValues
Compare city ranking
MapToPair
Write on HDFS

# Query 3 SQL - (1)


Filtering data
SELECT *
FROM filteredTab
WHERE year >= 2016 AND ( (month >= 6 AND month <= 9)
OR (month >= 1 AND month <= 4) ) AND (hour >= 12 AND hour <= 15)
Temperature average value in the first four months period
SELECT year, nation, city, AVG(temperature) as avg_temp1
FROM avgTemp1
WHERE month >= 1 AND month <= 4
GROUP BY year, nation, city
Temperature average value in the second fourmonths period
SELECT year, nation, city, AVG(temperature) as avg_temp2
FROM avgTemp2
WHERE month >= 6 AND month <= 9
GROUP BY year, nation, city
Temperature differences from max to min
SELECT year, nation, city, sub_temp
FROM (SELECT t1.year, t1.nation, t1.city, ABS(t1.avg_temp1 - t2.avg_temp2) as sub_temp
FROM temp1 as t1 JOIN temp2 as t2
ON t1.year = t2.year AND t1.city = t2.city
GROUP BY t1.year, t1.nation, t1.city, sub_temp)
GROUP BY year, nation, city, sub_temp
ORDER BY sub_temp DESC

# Query 3 SQL - (2)


Cities ranking in 2016
SELECT year, nation, city, sub_temp,
DENSE_RANK() OVER (PARTITION BY nation ORDER BY sub_temp DESC) as rank
FROM filter
WHERE (year == 2016)
GROUP BY year, nation, city, sub_temp
Top three cities in 2017
SELECT year, nation, city, sub_temp, rank
FROM (SELECT *,
DENSE_RANK() OVER (PARTITION BY nation ORDER BY sub_temp DESC) as rank
FROM filter
WHERE year == 2017)
WHERE rank BETWEEN 1 AND 3
GROUP BY year, nation, city, sub_temp, rank
Compare ranking between 2016 and 2017
SELECT r1.nation, r1.city, r2.year as currentYear, r2.rank as currentPosition, r1.year as lastYear, r1.rank as LastPosition
FROM rank2016 as r1 JOIN rank2017 as r2
ON r1.nation == r2.nation AND r1.city == r2.city
GROUP BY r1.year, r2.year, r1.nation, r2.nation, r1.city, r2.city, r1.rank, r2.rank, r1.sub_temp, r2.sub_temp
ORDER BY r1.nation, r2.rank

# Risultati query

Query 1

| year | cities |
|---|---|
| 2013 | Las Vegas |
| 2014 | Las Vegas |
| 2015 | Phoenix |
| 2015 | Las Vegas |
| 2016 | Eilat |
| 2016 | Phoenix |
| 2016 | Las Vegas |
| 2017 | Eilat |

Query 2

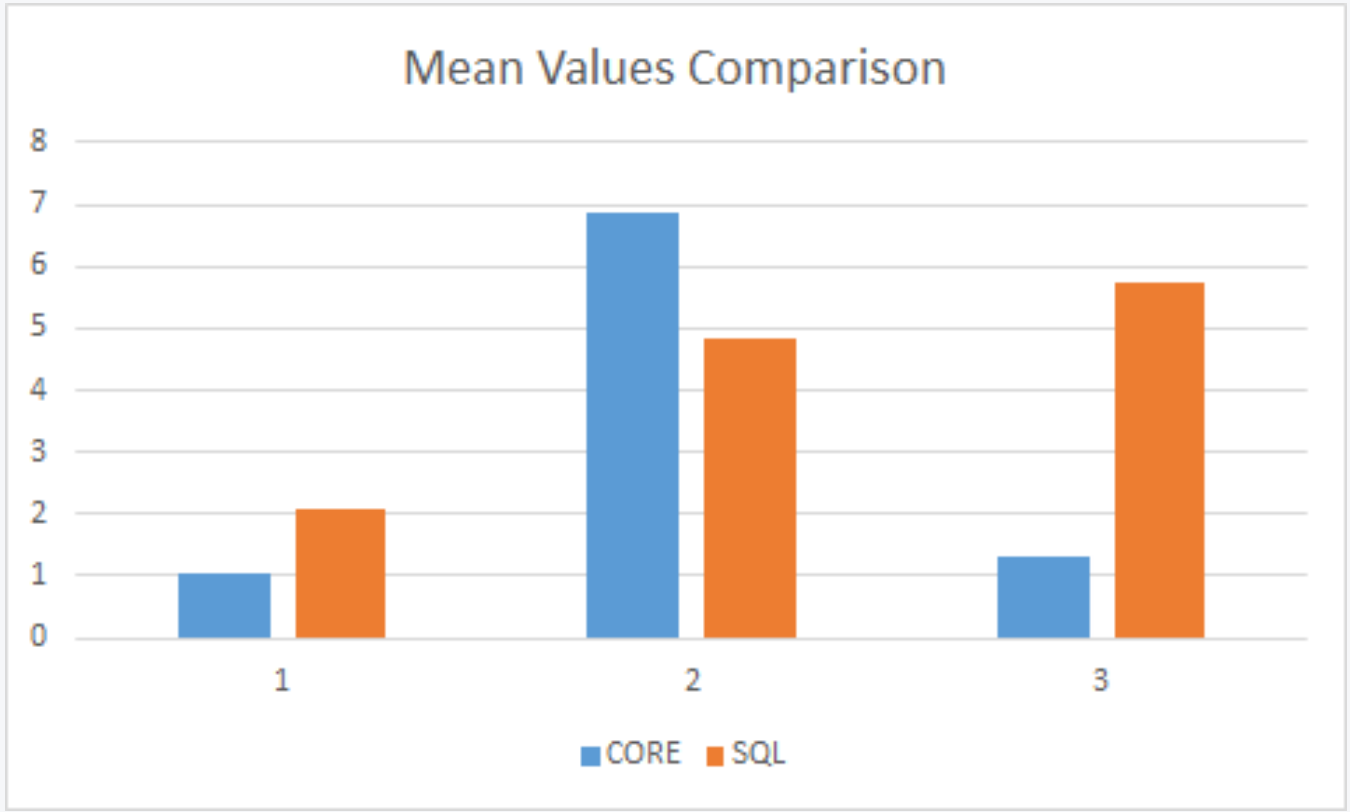
| country | year | month | mean | min | max | std_dev |
|---|---|---|---|---|---|---|
| US | 2017 | 4 | 289.0873198900438 | 268.85 | 309.43 | 6.920265998513651 |
| IL | 2013 | 11 | 290.0220528935104 | 280.368 | 305.67 | 5.526896959507662 |
| IL | 2015 | 4 | 290.0203029840284 | 276.984 | 307.246 | 4.467095681613358 |
| US | 2015 | 12 | 281.9348435066526 | 256.130423071 | 305.15 | 7.7474421142896 |
| IL | 2013 | 8 | 299.6482675851259 | 286.209 | 315.15 | 5.098120892435437 |
| US | 2014 | 9 | 293.56699400423815 | 271.170333333 | 311.889 | 5.999285077143429 |
| US | 2017 | 8 | 297.0385777080903 | 283.04 | 315.7 | 5.651514519969669 |
| US | 2014 | 2 | 277.7136012790829 | 247.13 | 302.91 | 10.59106048707934 |

Query 3

| country | city | currentYear | currentPosition | lastYear | LastPosition |
|---|---|---|---|---|---|
| IL | Beersheba | 2017 | 1 | 2016 | 1 |
| IL | Eilat | 2017 | 2 | 2016 | 5 |
| IL | Haifa | 2017 | 3 | 2016 | 2 |
| US | Minneapolis | 2017 | 1 | 2016 | 2 |
| US | Chicago | 2017 | 2 | 2016 | 3 |
| US | Detroit | 2017 | 3 | 2016 | 1 |

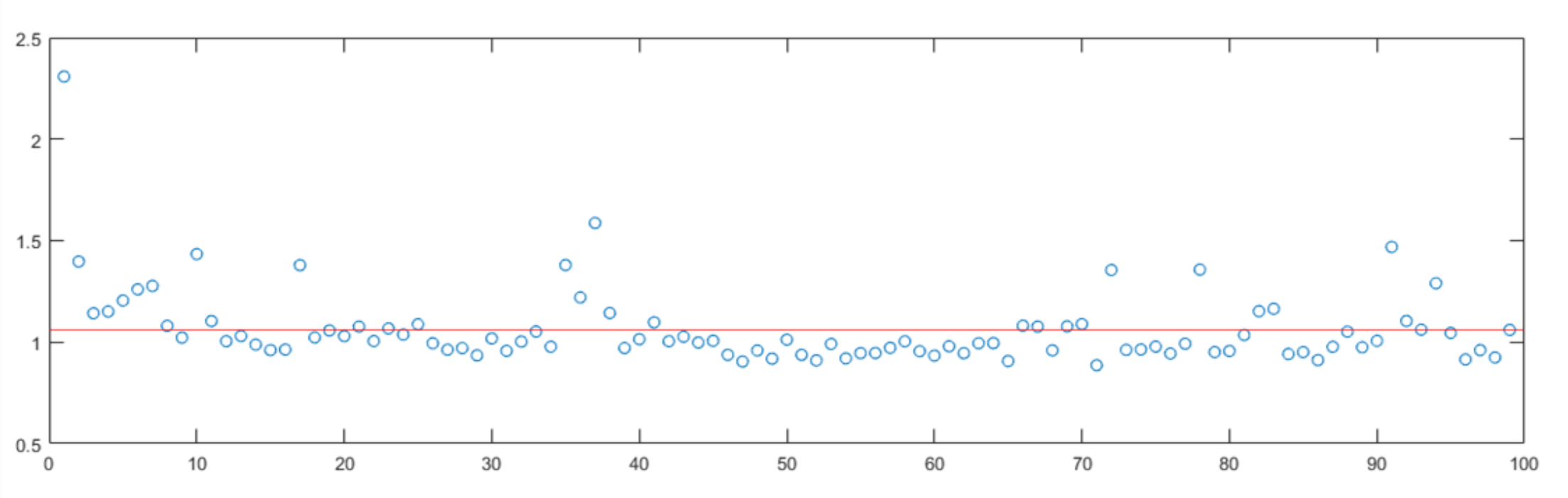
# Media e varianza - Core vs SQL

Hardware: Intel-Core I5 9600K 6-core, SSD con interfaccia PCIe su slot M.2 e  protocollo NVMe.

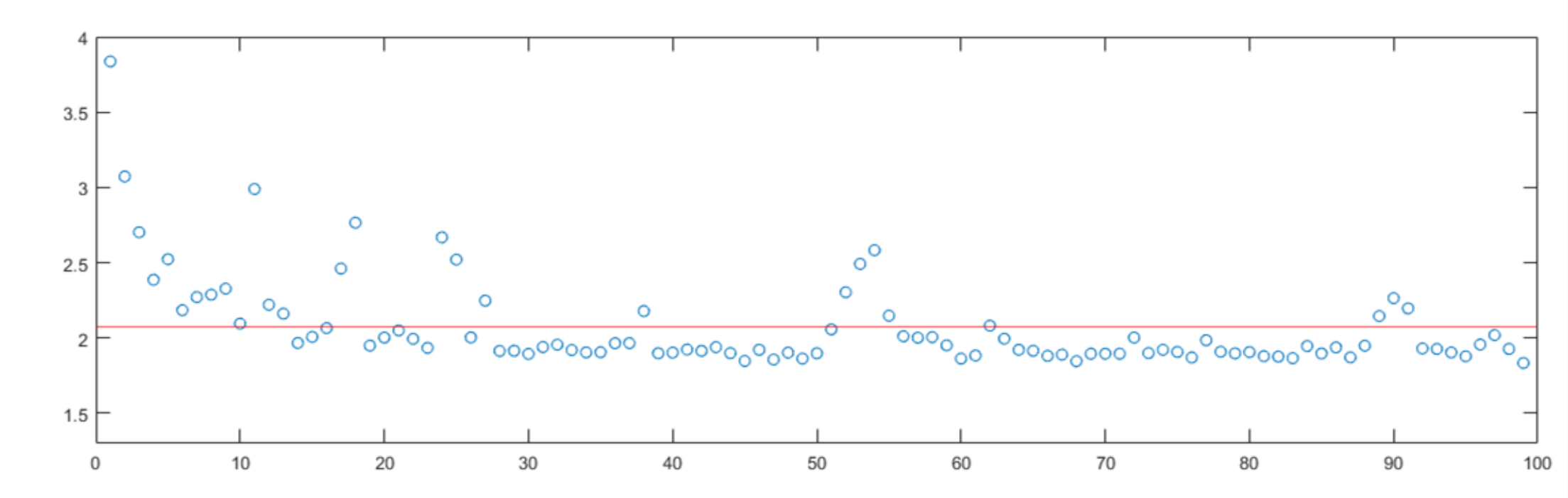



| Varianza | 1 | 2 | 3 |
|---|---|---|---|
| Core | 0.0353 | 0.471 | 0.0132 |
| SQL | 0.0951 | 0.2872 | 0.3044 |

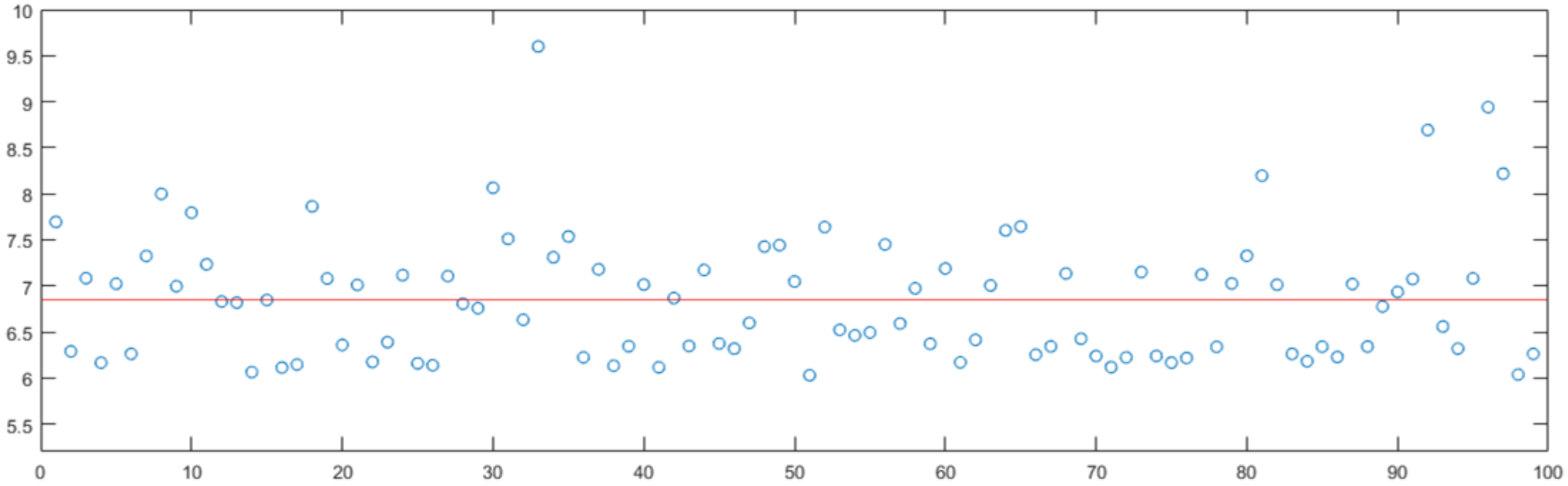
# Query1 Core & SQL



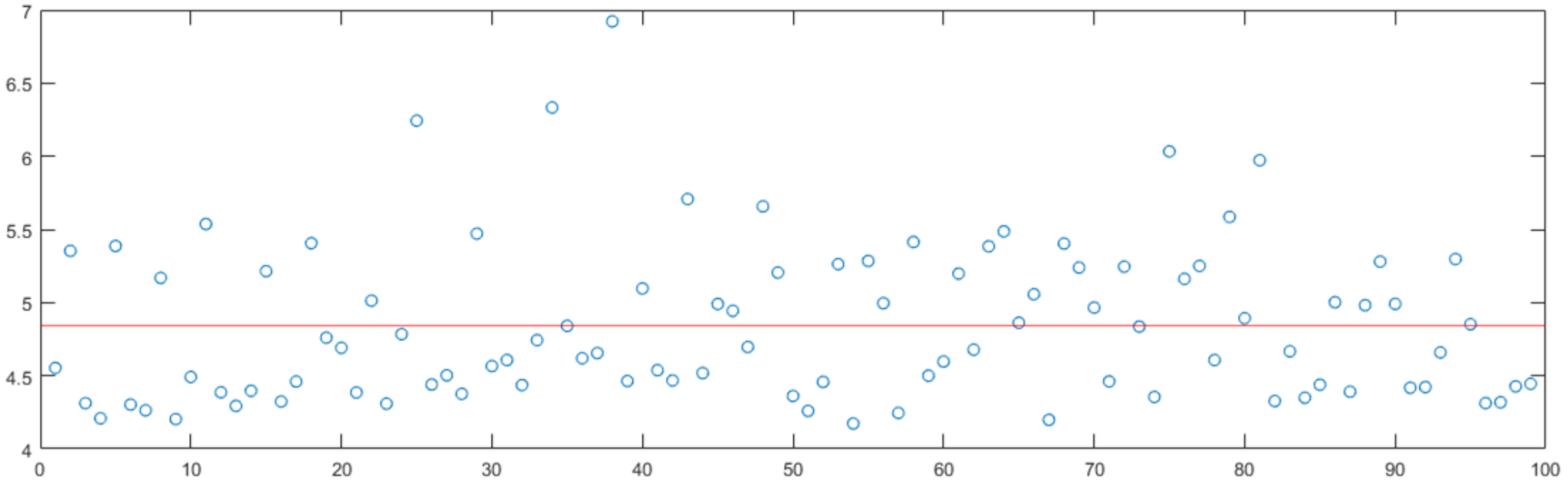
# Query1 Core & SQL



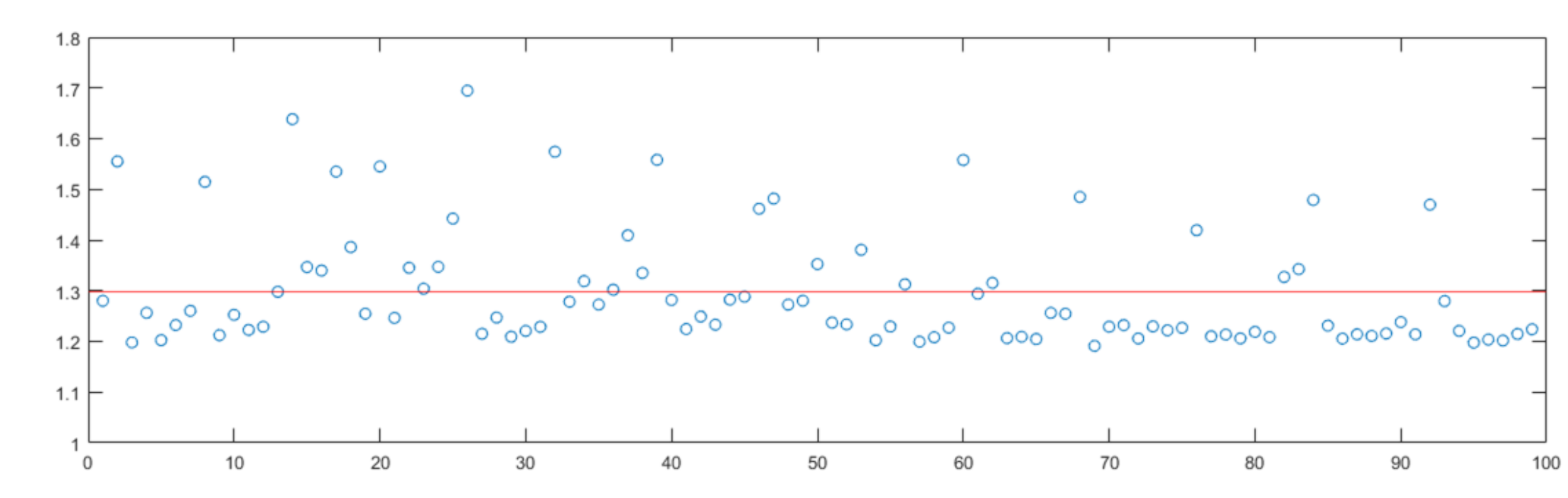
# Query2 Core & SQL



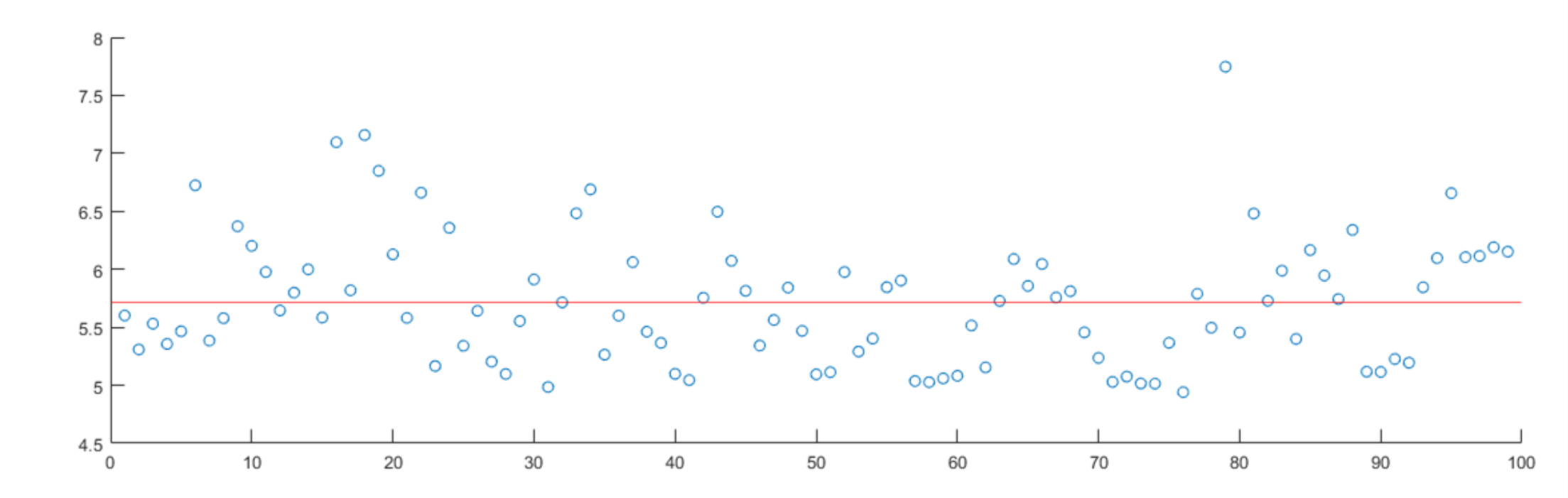
# Query2 Core & SQL



# Query3 Core & SQL



# Query3 Core & SQL

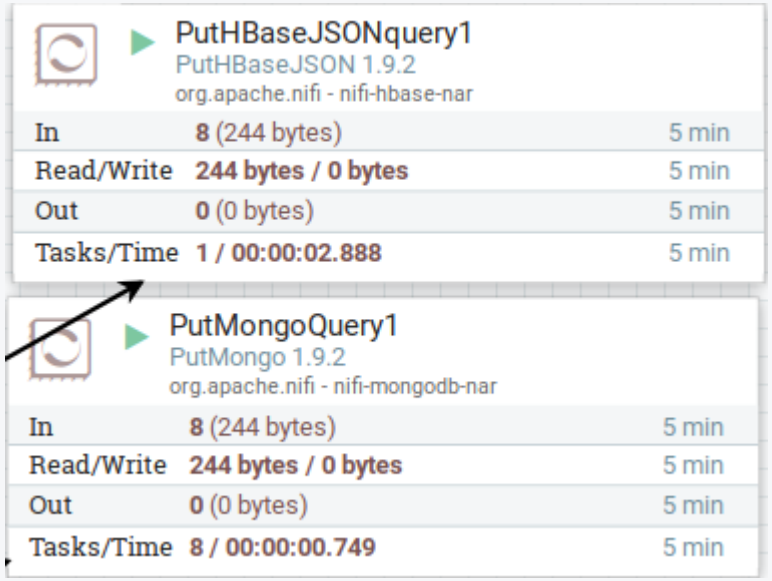


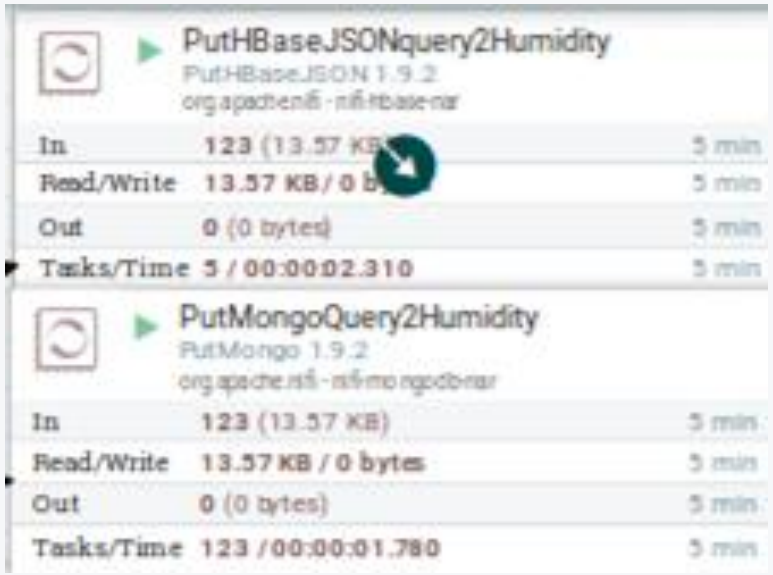
# Tempi Ni-Fi

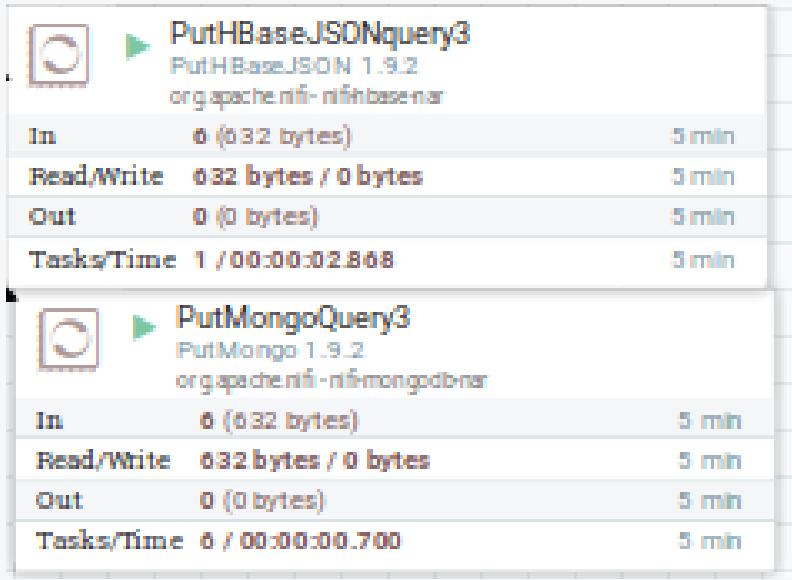
Lettura da locale fino a scrittura su HDFS ~31 s

| | | | | | | |
|---|---|---|---|---|---|---|
| | 05/29/2019 08:20:17.736 UTC | CREATE | de84144d-8515-4e99-b5a4-5... | 0 bytes | ListFile | ListFile |
| | 05/29/2019 08:20:48.886 UTC | DROP | de84144d-8515-4e99-b5a4-5... | 19.09 MB | PutParquet | PutParquet |

## Scrittura su DB per Query Core



# Considerazioni su Parquet

Utilizzando Parquet le dimensioni dei file cambiano nel modo seguente:

| | | | |
|---|---|---|---|
| city_attributes.csv | 1.01KB | city_attributes.parquet | 1.89KB |
| humidity.csv | 7.97MB | humidity.parquet | 1.39MB |
| pressure.csv | 10.68MB | pressure.csv | 1.14MB |
| temperature.csv | 12.08MB | temperature.parquet | 5.88MB |
| weather_description.csv | 19.09MB | weather_description.parquet | 806.65KB |

I tempi in lettura sono dimezzati, ma siccome erano già dell'ordine dei 300ms per brevità si è deciso di non riportarli.

# Grazie per l’attenzione

Progetto Batch processing– SABD 2018-2019

Montesano,Perrone,Pusceddu