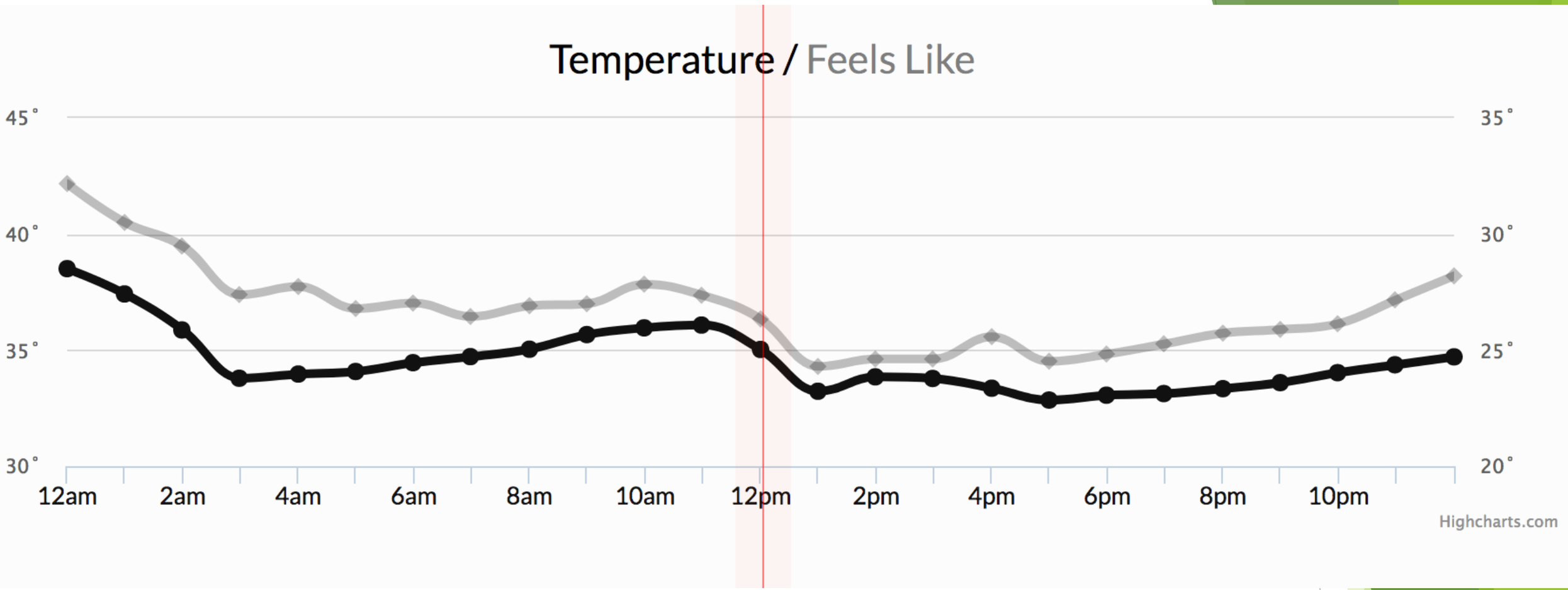
Influx**DB**
Base de datos para Series de Tiempo



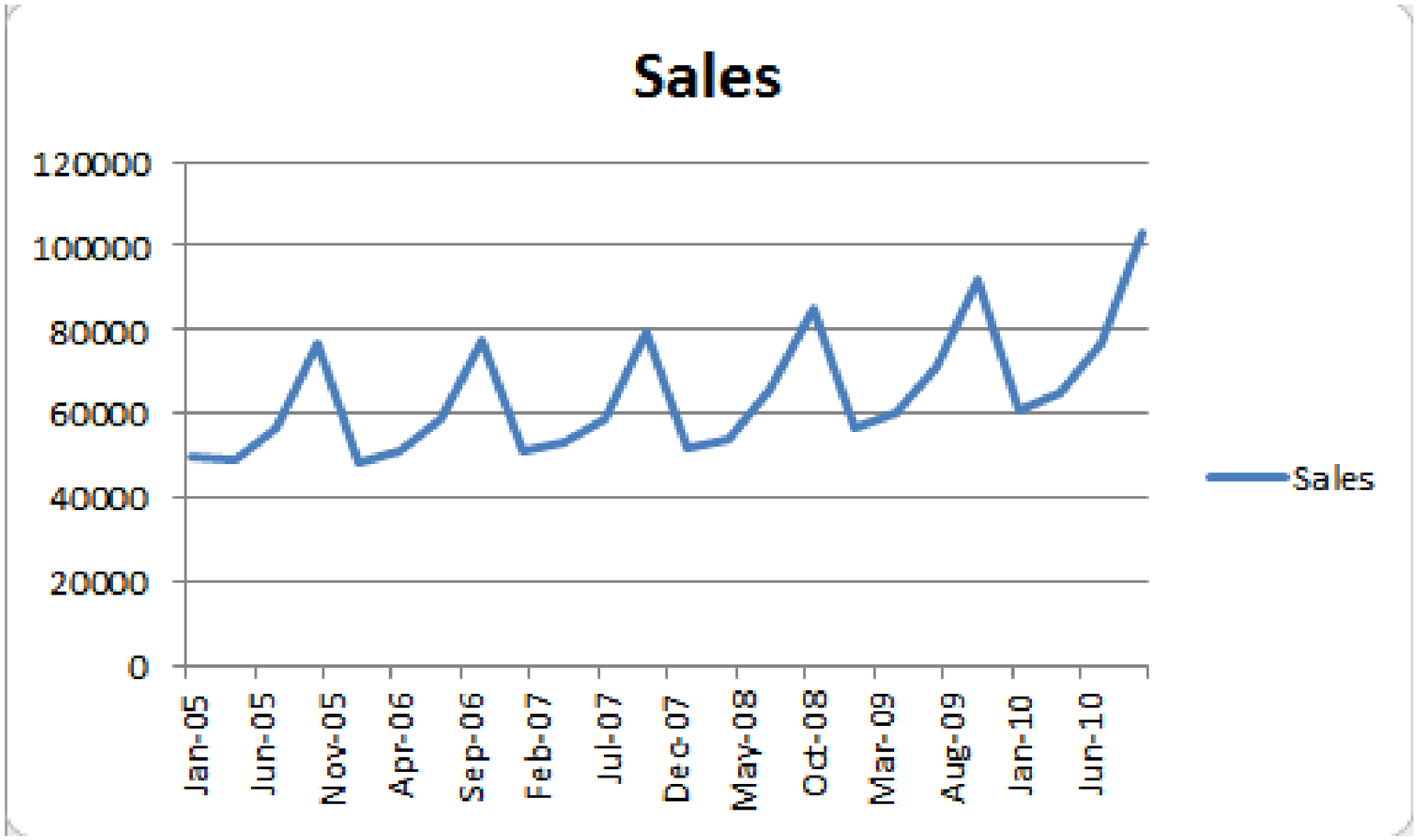
¿Qué es una serie de tiempos?

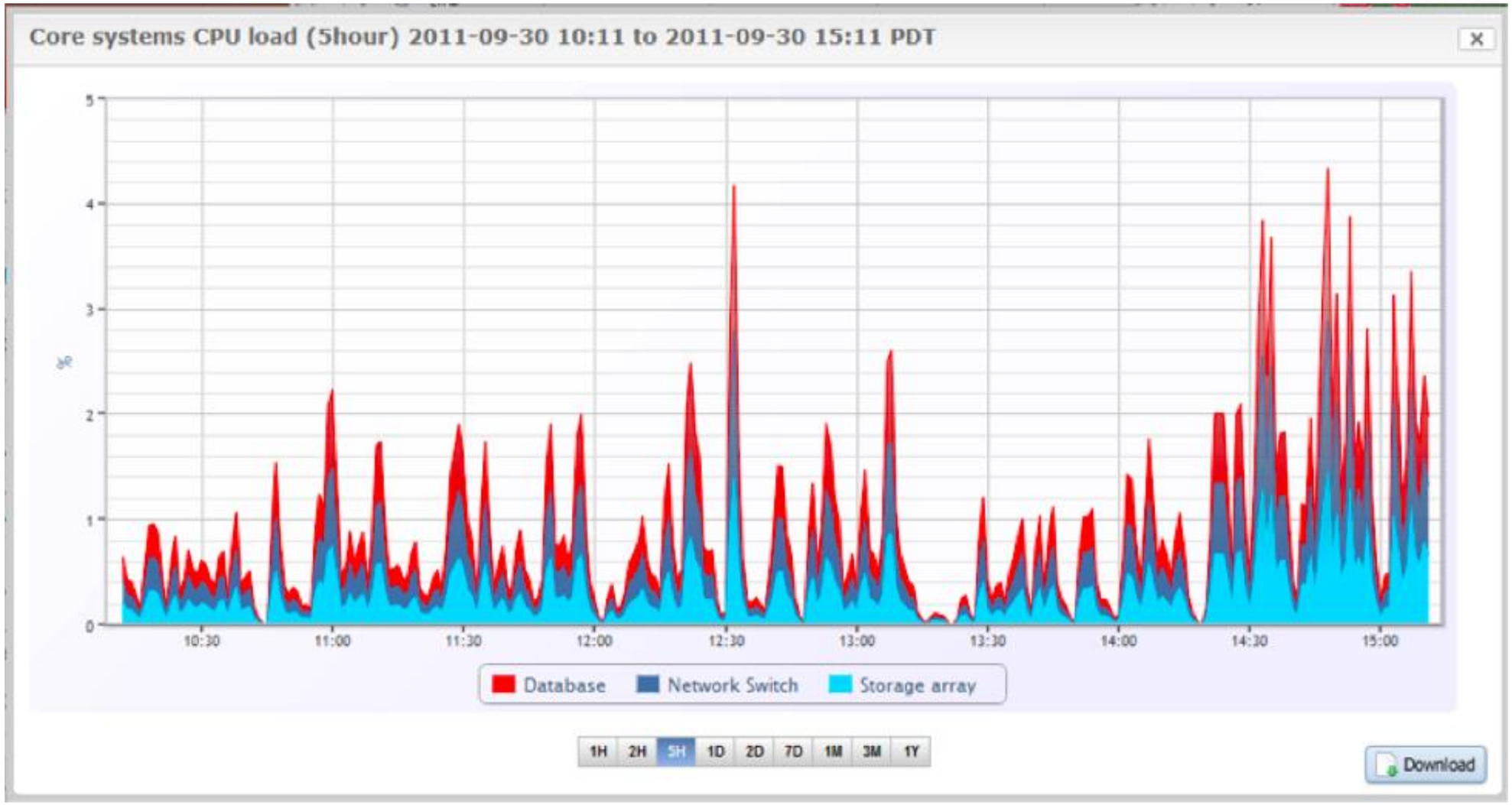
¿Qué es una serie de tiempos?

Una serie de tiempos es una secuencia de puntos de datos provenientes de la misma fuente durante un intervalo de tiempo.

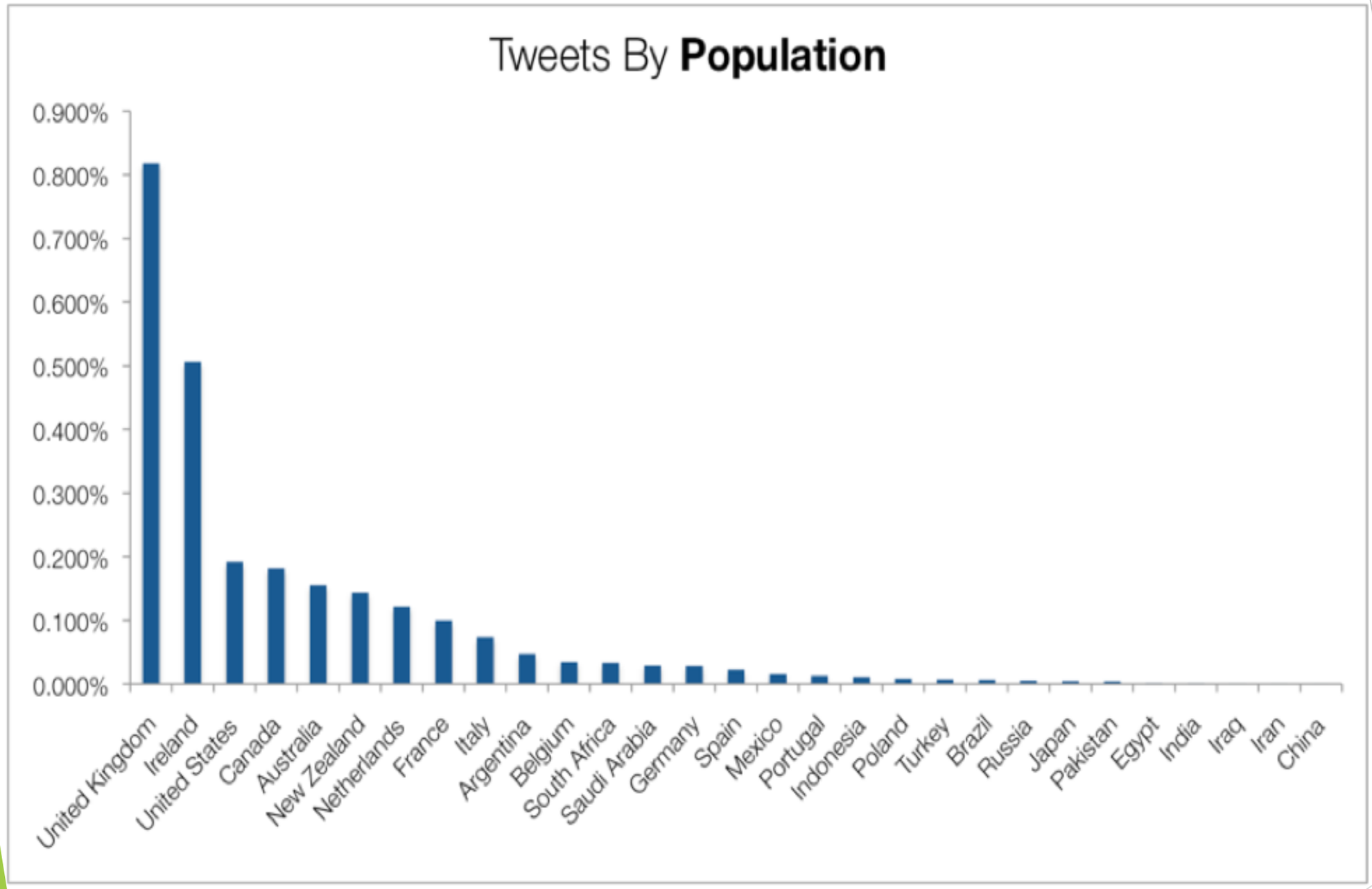
Si graficamos una serie de tiempos, uno de los ejes SIEMPRE va a ser el tiempo (normalmente un timestamp)

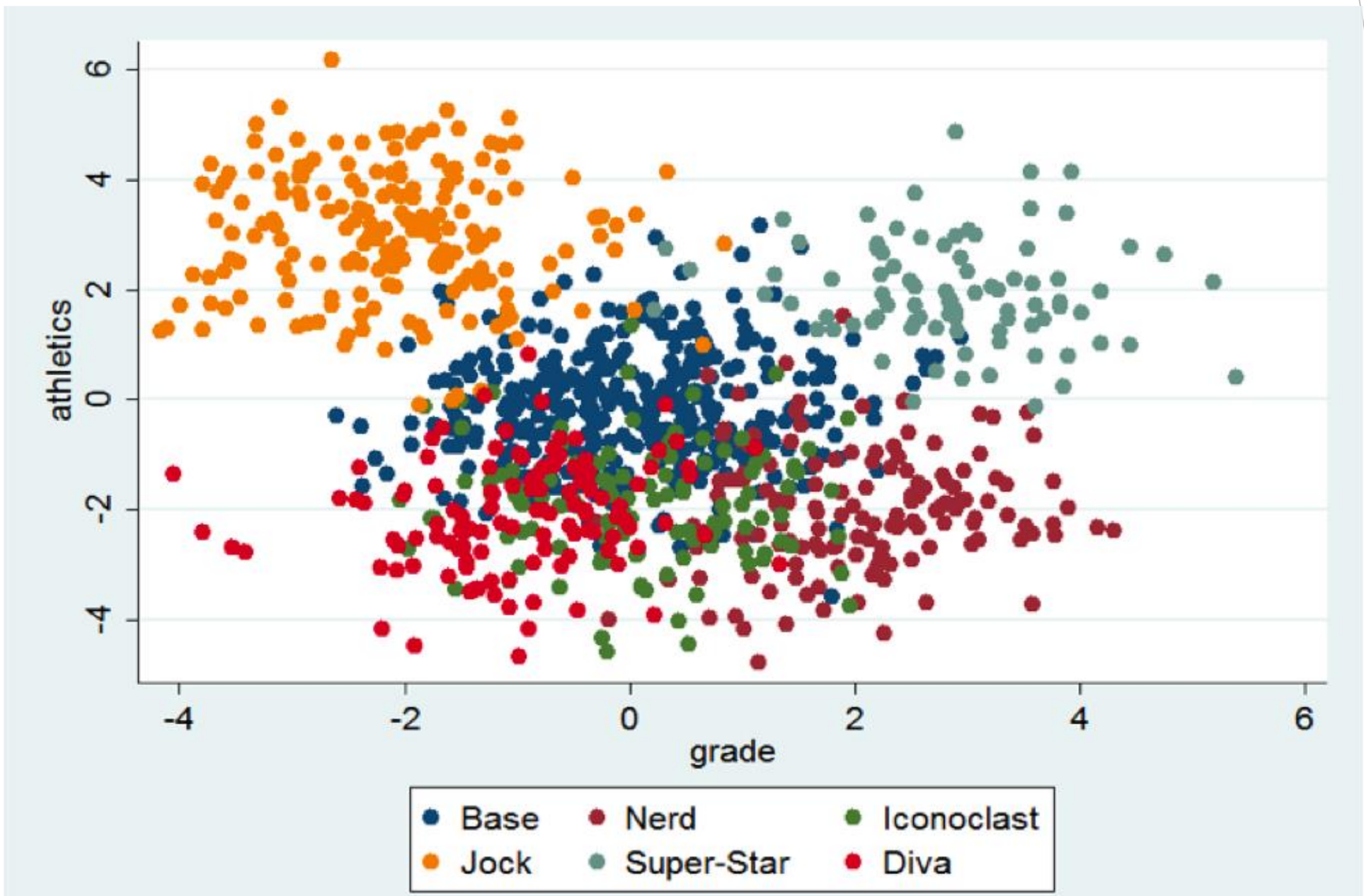
Ejemplos de datos en una serie de tiempos

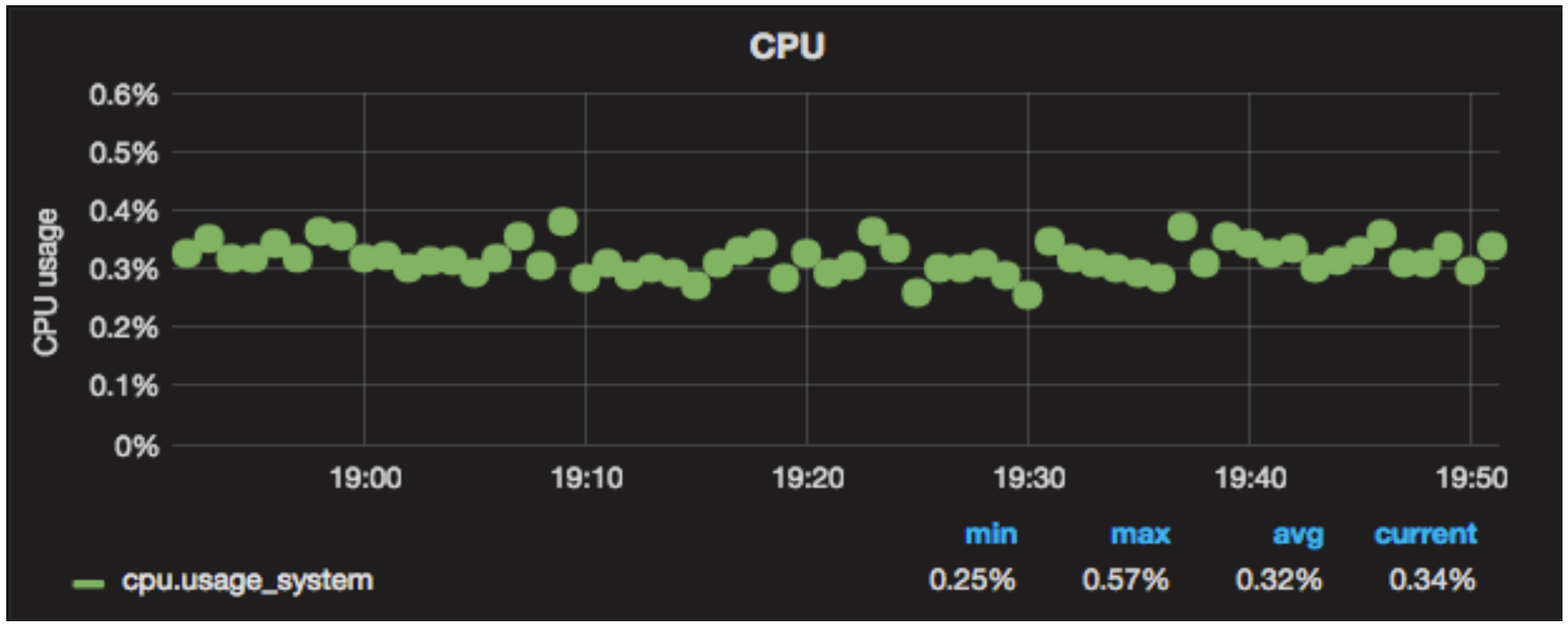

Sales
120000
100000
80000
60000
40000
20000
0
Jan-05
Jun-05
Nov-05
Apr-06
Sep-06
Feb-07
Jul-07
Dec-07
May-08
Oct-08
Mar-09
Aug-09
Jan-10
Jun-10
Sales


Core systems CPU load (5hour) 2011-09-30 10:11 to 2011-09-30 15:11 PDT
5
4
3
2
1
0
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
14:30
15:00
Database
Network Switch
Storage array
1H
2H
5H
1D
2D
7D
1M
3M
1Y
Download

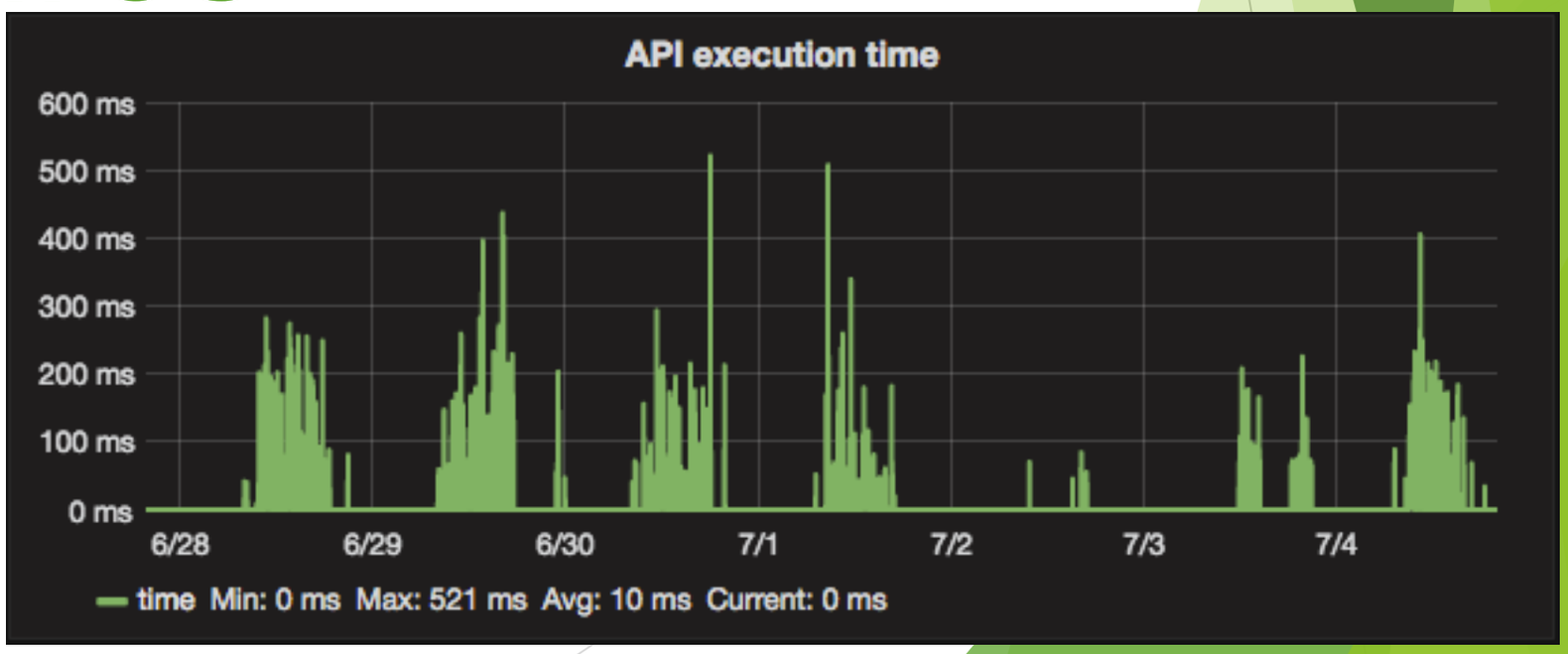
¿Qué NO es una serie de tiempos?

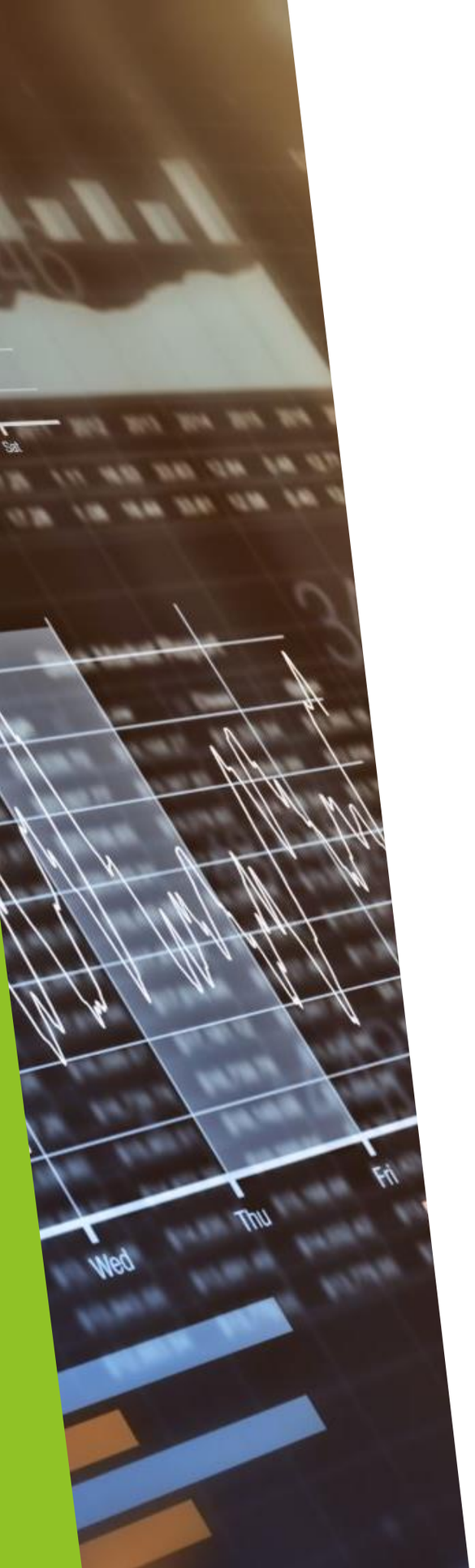

Tweets By **Population**
0.900%
0.800%
0.700%
0.600%
0.500%
0.400%
0.300%
0.200%
0.100%
0.000%
United Kingdom
Ireland
United States
Canada
Australia
New Zealand
Netherlands
France
Italy
Argentina
Belgium
South Africa
Saudi Arabia
Germany
Spain
Mexico
Portugal
Indonesia
Poland
Turkey
Brazil
Russia
Japan
Pakistan
Egypt
India
Iraq
Iran
China


athletics
6
4
2
0
-2
-4
-4
-2
0
2
4
6
grade
Base
Jock
Nerd
Super-Star
Iconoclast
Diva



Series de tiempo Regulares vs Irregulares





Aplicaciones de Series de Tiempos

- Internet de las cosas (datos de sensores)
- Alertas
- Monitorización
- Analíticas en tiempo real

InfluxDB es la **I** del stack TICK

- Telegraf – colector de datos de tiempo
- InfluxDB – base de datos de series de tiempo
- Chronograf – Visualización de datos de series de tiempo
- Kapacitor – Procesado y alertas de datos de series de tiempo

InfluxDB features

Lenguaje de consultas similar a SQL

Sin esquema

Sensible a mayúsculas

Tipos de datos: string, float64, int64, booleano

Measurement (medición)

Una medición (o Punto) es un único registro (fila) en un almacenamiento InfluxDB

Cada medición se comprende de un Tiempo (como clave primaria), tags (etiquetas - columnas indexadas) y fields (campos – columnas no indexadas)

Insertar

Nombre del measurement (“tabla”)

```
INSERT table_name,tag1=value1,tag2=value2 temp=30.5,value=1.5
```

Insertar

Nombre del measurement (“tabla”)

```
INSERT table_name,tag1=value1,tag2=value2 temp=30.5,value=1.5
```

La coma separa la medición de las etiquetas

La coma separa entre etiquetas y entre campos

Insertar

Nombre del measurement (“tabla”)

```
INSERT table_name,tag1=value1,tag2=value2 temp=30.5,value=1.5
```

El espacio separa etiquetas de campos

Insertar

Nombre del measurement (“tabla”)

Tags

value1 value2

tags

```
INSERT table_name,tag1=value1,tag2=value2 temp=30.5,value=1.5
```

Inserting

Nombre del measurement (“tabla”)

Fields

temp	value
30.5	1.5

```
INSERT table_name,tag1=value1,tag2=value2 temp=30.5,value=1.5
```

fields

Insertar

Nombre del measurement (“tabla”)

Fields

temp	value
30.5	1.5

tags

```
INSERT table_name,tag1=value1,tag2=value2 temp=30.5,value=1.5
```

La coma separa la medición de las etiquetas fields
La coma separa entre etiquetas y entre campos
El espacio separa etiquetas de campos

Consultas

- Lista las db:

```
> SHOW DATABASES
```

- Selecciona db:

```
> USE workshop
```

- Lista measurements ("tablas”)

```
> SHOW MEASUREMENTS
```

- Selección completa simple

```
> SELECT * FROM measurement_name
```

Consultas (2)

- Select con límite:

```
> SELECT * FROM measure LIMIT 10
```

- Select con un offset:

```
> SELECT * FROM measure OFFSET 10
```

- Select con una condición where:

```
> SELECT * FROM measure WHERE tag1 = 'value1'
```

- Select con una condición order_by:

```
> SELECT * FROM measure ORDER BY cpu DESC
```

Consultas (3)

- Operadores:

```
= igual que
<>, != no igual que
> mayor que
< menor que
=~ matchea con una (REGEX)
!~ no matchea con una (REGEX)
```

Agregadores - COUNT()

Devuelve el número de valores no nulos

```
> SELECT count(<field>) FROM measure
```

```
> SELECT count(cpu) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Agregadores - MEAN()

Devuelve la media de un único campo (calculado a partir de los valores no nulos)

```
> SELECT mean(<field>) FROM measure

> SELECT mean(cpu) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Agregadores - MEDIAN()

Devuelve el valor mediano de los valores ordenados en un único campo (es similar a PERCENTILE(field,50).

```
> SELECT median(<field>) FROM measure

> SELECT median(cpu) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Agregadores - SPREAD()

Devuelve la diferencia entre el valor mínimo y el máximo de un campo.

```
> SELECT spread(<field>) FROM measure

> SELECT spread(cpu) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Agregadores - SUM()

Devuelve el sumatorio de todos los valores de un campo.

```
> SELECT sum(<field>) FROM measure

> SELECT sum(cpu) FROM cpu_temp
  WHERE time  >  '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Selectores - BOTTOM(N)

Devuelve los N valores más pequeños de un campo

```
> SELECT bottom(<field>, <N>) FROM measure

> SELECT bottom(cpu, 5) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Selectores - FIRST()

Devuelve el valor más antiguo de un campo.

```
> SELECT first(<field>) FROM measure

> SELECT first(cpu) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Selectores - LAST()

Devuelve el valor más reciente de un campo

```
> SELECT last(<field>) FROM measure
```

```
> SELECT last(cpu) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Selectores - MAX()

Devuelve el mayor valor de un campo.

```
> SELECT max(<field>) FROM measure
```

```
> SELECT max(cpu) FROM cpu_temp
  WHERE time  >  '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Selectores - MIN()

Devuelve el menor valor de un campo.

```
> SELECT min(<field>) FROM measure
```

```
> SELECT min(cpu) FROM cpu_temp
  WHERE time  >  '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Selectores - PERCENTILE(N)

Devuelve el valor del percentil-N para los valores ordenados de un único campo

```
> SELECT percentile(<field>, <N>) FROM measure

> SELECT percentile(cpu, 95) FROM
  cpu_temp WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Selectores - TOP(N)

Devuelve los mayores N valores de un campo

```
> SELECT top(<field>, <N>) FROM measure
```

```
> SELECT top(cpu, 5) FROM cpu_temp
  WHERE time > '2016-07-04'
  AND time < '2016-07-05'
  GROUP BY time(1h)
```

Cláusulas GROUP BY

InfluxDB soporta la cláusula GROUP BY con valores de etiquetas, intervalos de tiempos, combinación de ambos y fill()

Downsampling

InfluxDB puede manejar cientos de miles de puntos por segundo.
Trabajar con un volumen tan alto de datos durante un largo periodo de tiempo puede provocar problemas de almacenamiento.
Una solución natural es “downsamplear” los datos; mantenemos los datos en bruto de alta precisión sólo durante un tiempo limitado y almacenamos los datos resumidos y de baja precisión durante más tiempo (o permanentemente)

Retención de Datos

Las políticas de retención son las directivas de la estructura de datos de InfluxDB que describe durante cuanto tiempo InfluxDB debe mantener los datos y cuantas copias de esos datos se almacenan en el cluster (si existe). Una base de datos puede tener múltiples RP

DEMO TIME