

Elastic Stack 을 활용한 Data Dashboard 만들기

Week 2 - Data를 시각화해보자 II



Fast Campus

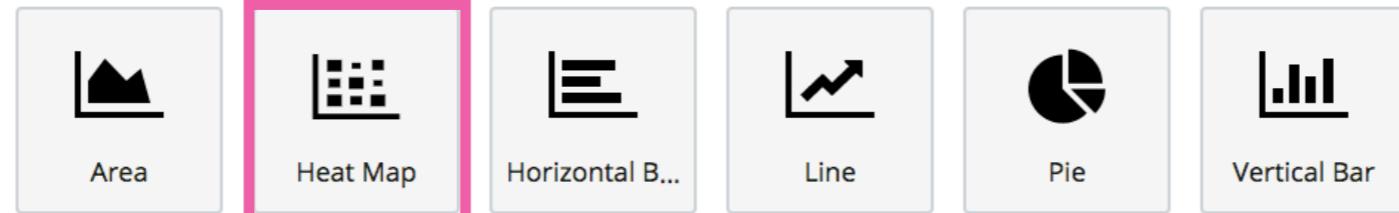
내용	페이지
Visualize 실전	
Heat Map	4
Goal/Gauge	13
Data Table	18
Area	24
Timelion	32
Aggregation	
Parent Pipeline Aggregation	52
Sibling Pipeline Aggregation	75
FAQ	98

우선 지난 시간에 못 다룬 Visualization Type를 간단히 살펴보자
(기본적인 Metric/Bucket Aggregation과 Visualization UI에 익숙해지자)

Select visualization type

Search visualization types...

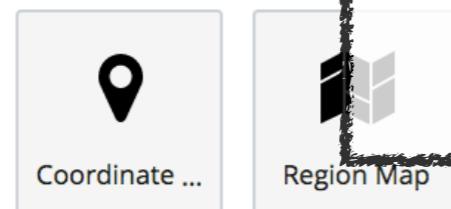
Basic Charts



Data

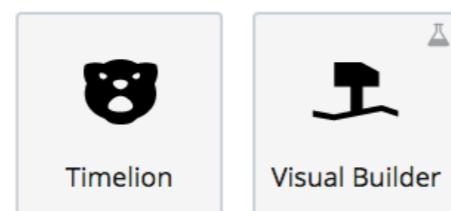


Maps



Visualize - Heat Map

Time Series

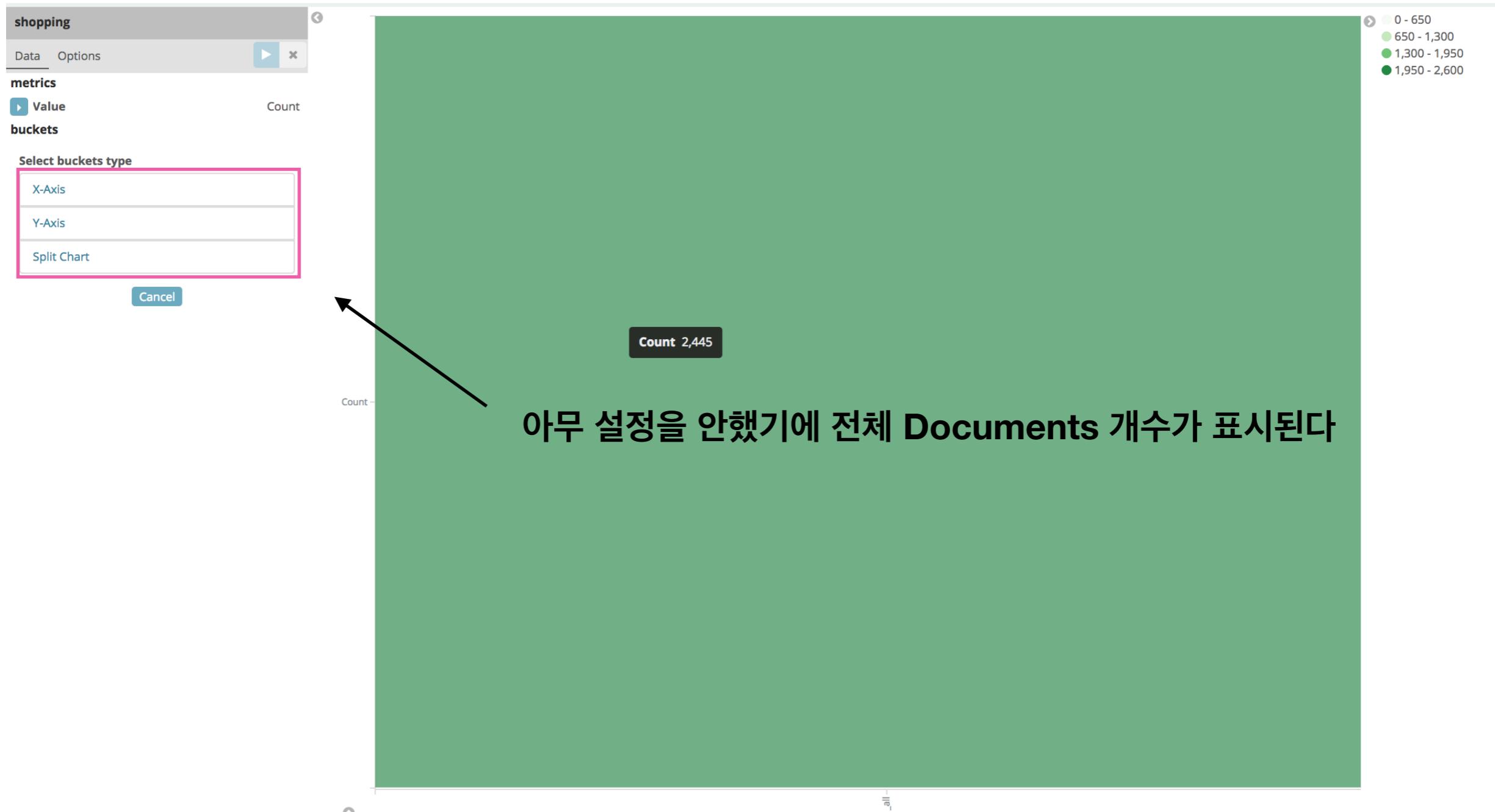


Other



Heat Map

- x축/y축 buckets에 따른 특정 metrics의 값 시각화
 - metric aggregation는 value count로 고정하고 buckets를 나누는 과정을 학습하자



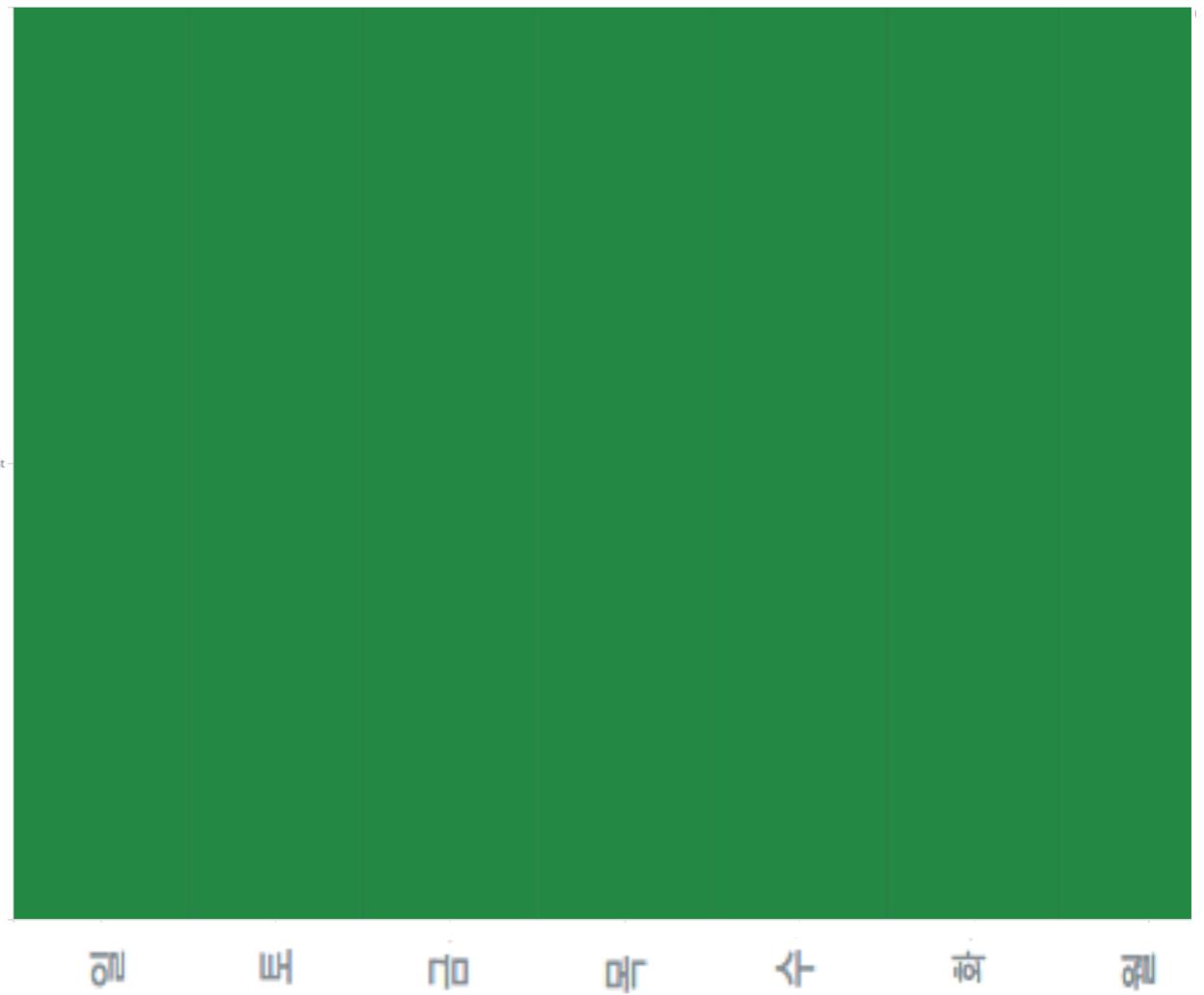
Heat Map



- shopping index 중에서 year to date 기간 동안의
- “주문시간” field를 기준으로 한
- 주별 (weekly) documents 개수

1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. 날짜를 기준으로 bucket을 생성하기 위해 ②를 눌러서 date histogram 선택
3. ③을 눌러서 ②를 적용할 대상 Field 선택
4. ④를 눌러서 ③을 나눌 간격 설정
5. ⑤를 눌러서 시각화

Heat Map



0 - 100
100 - 200
200 - 300
300 - 400

- shopping index 중에서 year to date 기간 동안의
- “주문시간_요일” field를 기준으로 한
- “주문시간_요일_sort”의 average가 큰
- 상위 7개 “주문시간_요일” 별
- documents 개수

1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. 특정 Field의 Value를 기준으로 bucket을 생성하기 위해 ②를 눌러서 Terms 선택
3. ③을 눌러서 ②를 적용할 대상 Field 선택
4. ④를 눌러서 ②~③을으로 생성한 buckets을 어떤 값을 기준으로 정렬할지 선택
5. ⑤로 시작화

이번에는 Y-Axis 또한 나누어 보자



그전에 이게 왜 필요할까?



지금까지는 **하나의** Field를 기준으로 시각화 했다



하지만 현실 세계의 문제는 그리 간단하지 않다면?

Heat Map



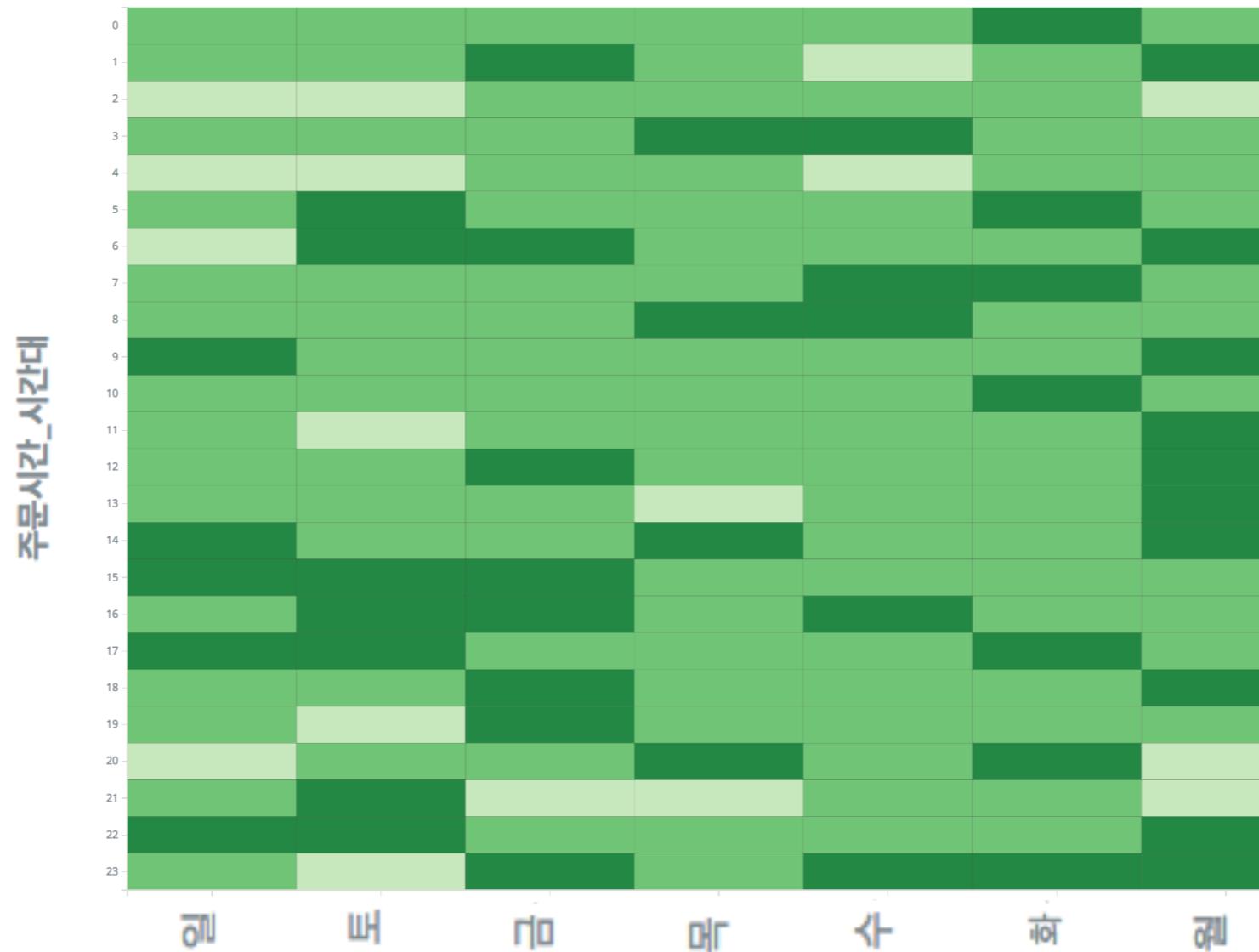
1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. X-Axis 설정은 7 page의 설정을 유지하자
3. Y-Axis에 적용할 bucket aggregation을 ③에서 설정
4. ④를 눌러 시각화

Heat Map 예제 1

metrics : 각 bucket에서 “배송소요시간” Field가 가장 컸던 10개 Field의 “배송소요시간” Field Value의 평균

X-Axis : “주문시간_요일” Field를 기준으로 “월, 화, 수, 목, 금, 토, 일”로 표시

Y-Axis : “주문시간_시간대” Field를 기준으로 1시간 단위로 표시

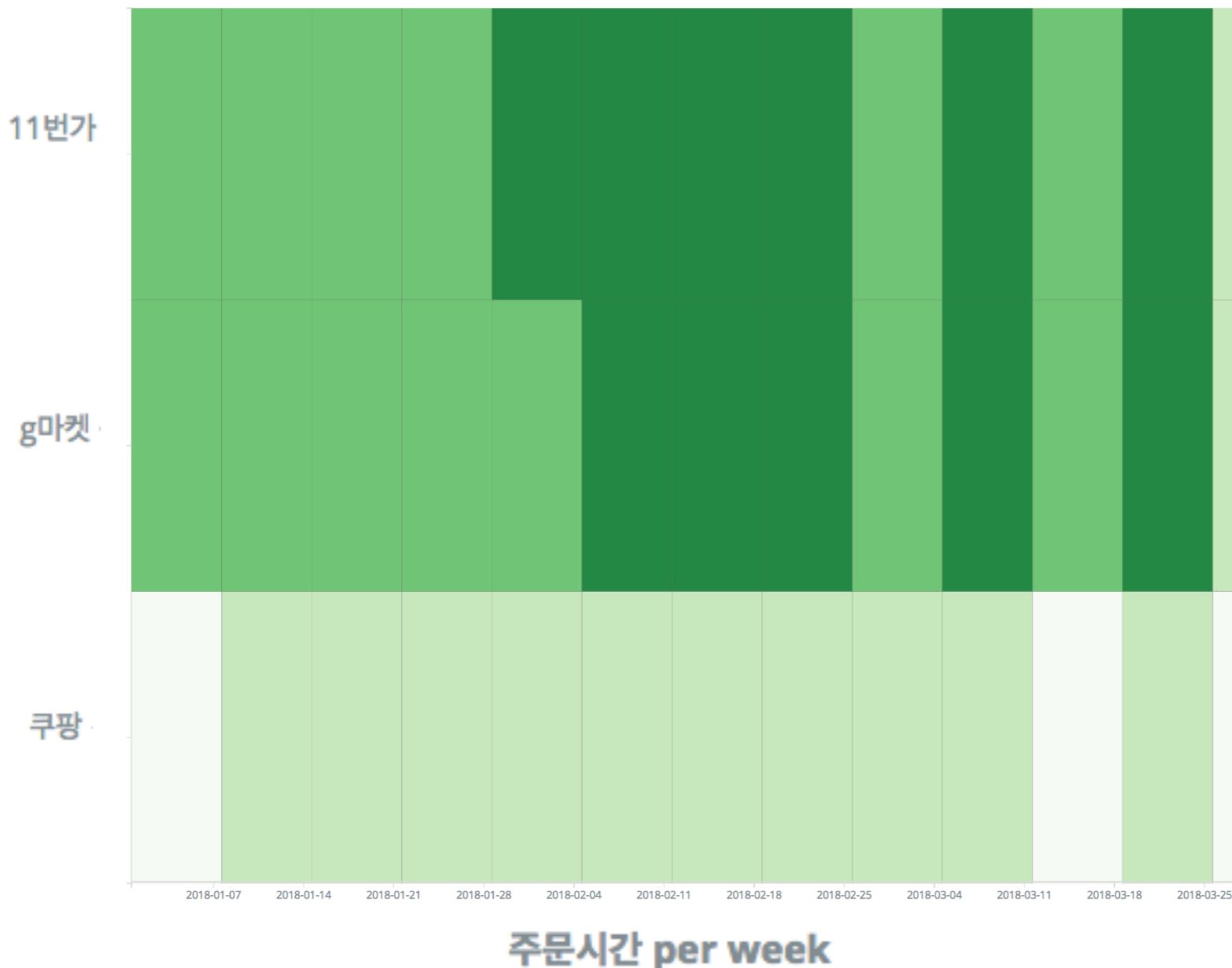


Heat Map 예제 2 ↗

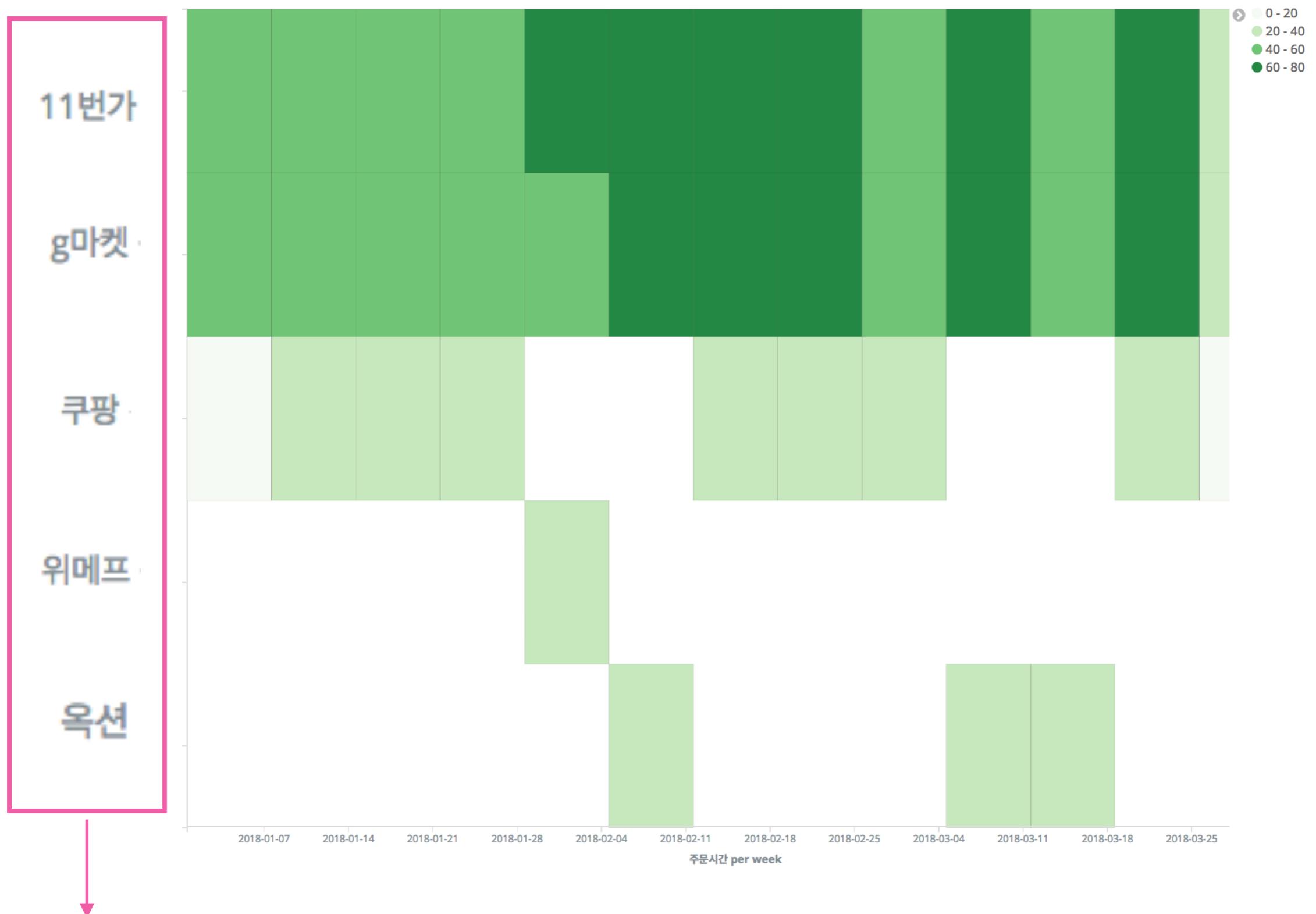
metrics : 각 bucket 별 count

X-Axis : “구매사이트” Field를 기준으로 count가 많았던 상위 3개 bucket 선정

Y-Axis : “주문시간” Field를 weekly로 나누기



Heat Map 예제 2 ↗

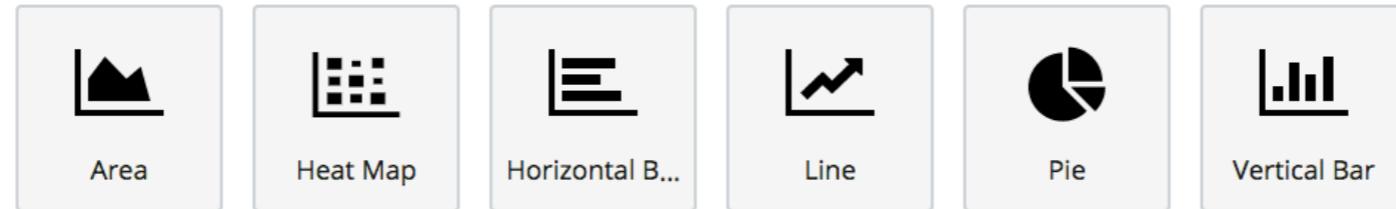


왜 5개가 나올까?

Select visualization type

Search visualization types...

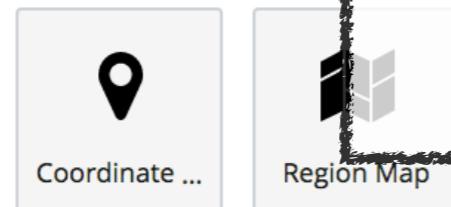
Basic Charts



Data

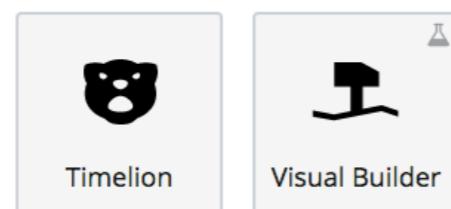


Maps

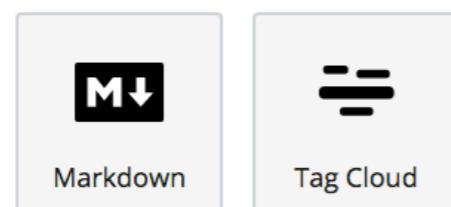


Visualize - Goal

Time Series



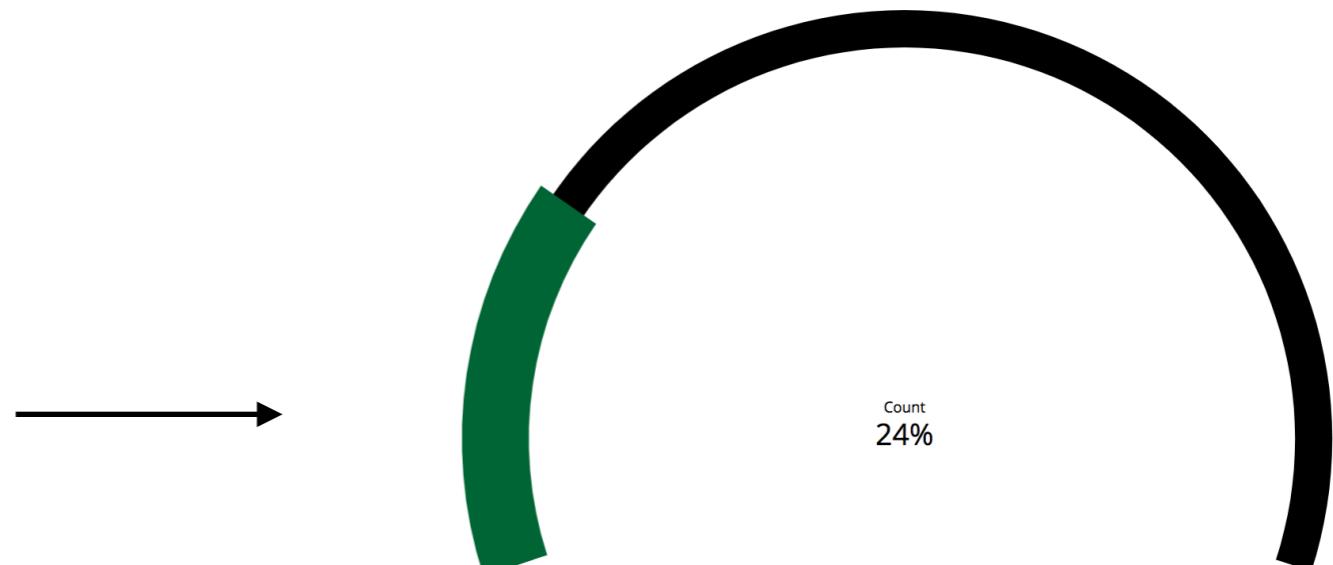
Other



Goal

- 특정 지표의 최종목표 달성을 시각화 할 때 사용
- metric aggregation는 value count로 고정하자

The screenshot shows the configuration for a 'shopping' index. In the 'metrics' section, 'Count' is selected (①). Under 'Gauge Type', 'Arc' is chosen (④). In the 'Ranges' section, 'From' is set to 0 and 'To' is set to 10000 (③). The 'Percentage Mode' checkbox is checked (②). Other settings like 'Vertical Split', 'Show Legend', 'Show Labels', and 'Sub Text' are also visible.



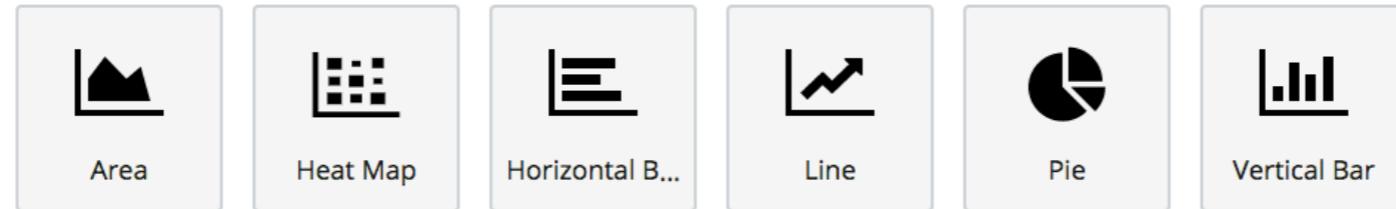
- shopping index 중에서 year to date 기간 동안의
- documents 개수의 목표를 10,000이라고 했을 때
- 현재까지의 달성을

1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. 목표 대비 현재 달성을 표시하기 위해 ②에서 Percentage Mode 선택
3. ③에서 최종 목표값 입력 ※ 일반적으로 From은 0으로 고정하고 To에 목표값을 입력
4. ④를 눌러서 시작화

Select visualization type

Search visualization types...

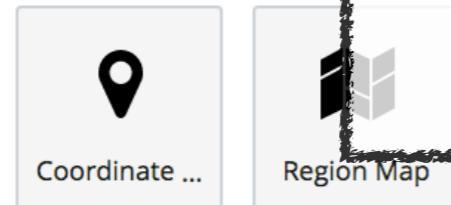
Basic Charts



Data

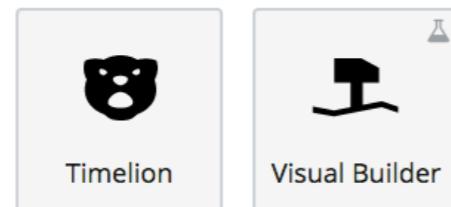


Maps

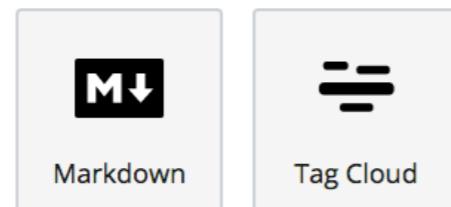


Visualize - Gauge

Time Series



Other



Gauge

- 특정 지표의 단계별 목표 달성을 시각화 할 때 사용
- 단계별 목표 구간을 설정한다는 점을 제외하고는 goal과 동일

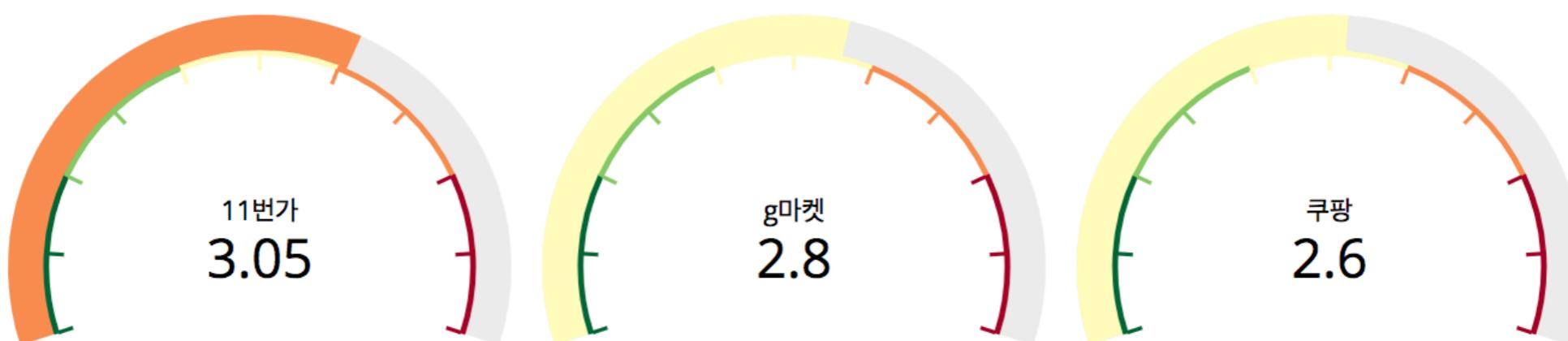


1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. 목표 대비 현재 달성을 표시하기 위해 ②에서 Percentage Mode 선택
3. ③에서 단계별 목표값 설정
4. ④를 눌러서 시작화

Gauge - 예제 1

“상품가격”의 합이 가장 컸던 3개의 “구매사이트”를 선별한 후, “구매사이트” 별로 “상품가격” 값이 가장 컸던 20개 Documents의 “판매자평점” 값의 평균을 모니터링한다고 하자. 단계별 목표를 아래와 같이 설정하고 시각화하자.

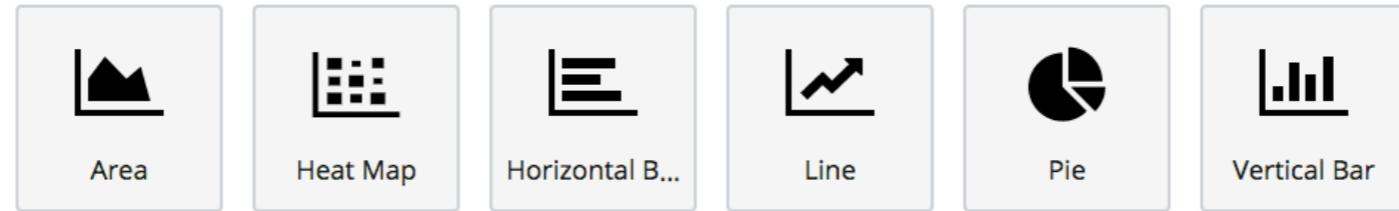
Ranges	
From	To
0	1
1	2
2	3
3	4
4	5



Select visualization type

Search visualization types...

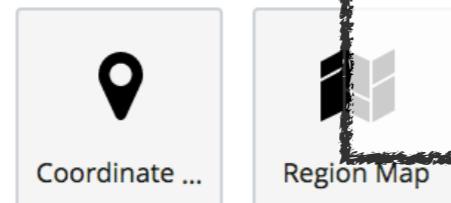
Basic Charts



Data

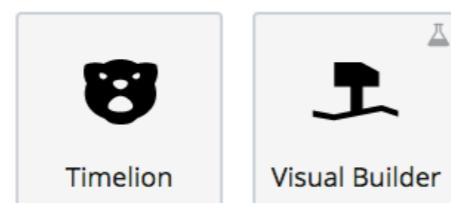


Maps



Visualize - Data Table

Time Series

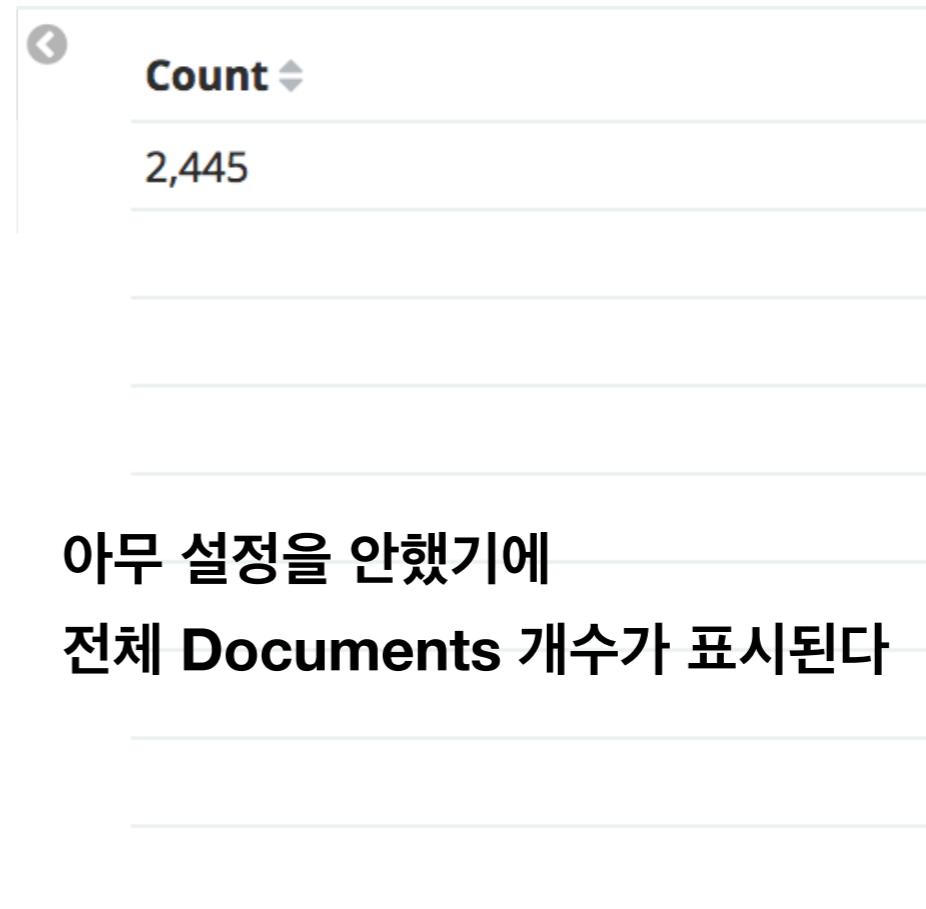
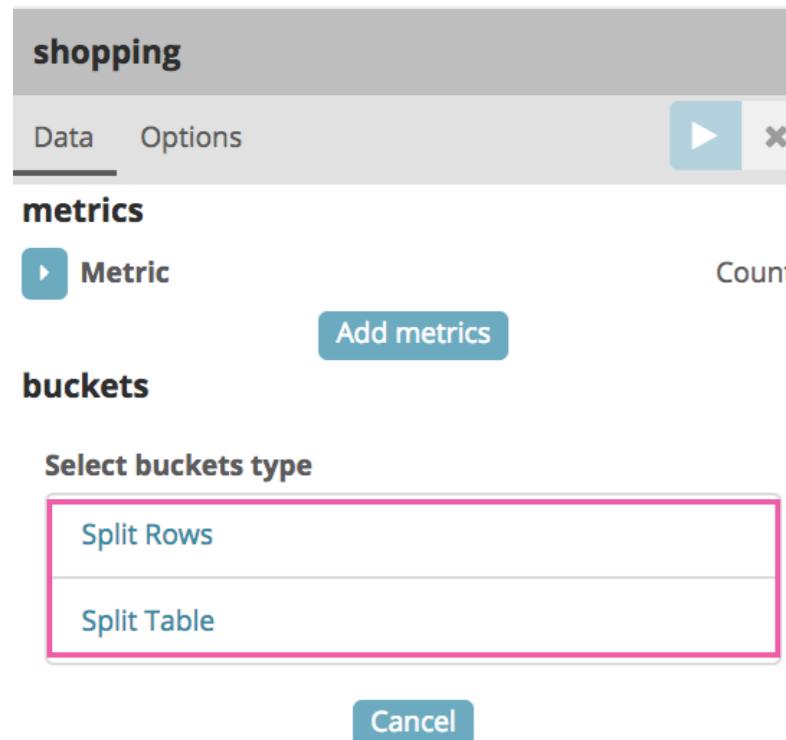


Other



Data Table

- 정형화된 형태로 데이터를 시각화하고 싶을 때 사용 (≒ excel)



Count
2,445

아무 설정을 안했기에
전체 Documents 개수가 표시된다

Export: [Raw](#) [Formatted](#)

Data Table

The screenshot shows the Elasticsearch Data Table interface. On the left, there is a search bar with the term "shopping" and a play button icon. Below it, under "metrics", is a dropdown menu set to "Count" (marked with ①). Under "buckets", there is a "Split Rows" section with a dropdown menu set to "Date Histogram" (marked with ②), a "Field" dropdown set to "주문시간" (marked with ③), and an "Interval" dropdown set to "Weekly" (marked with ④). At the bottom right of the search bar area is a circled number "5". A large arrow points from the search bar area to the right, where the results are displayed in a table.

주문시간 per week	Count
01월01일	166
01월08일	177
01월15일	161
01월22일	180
01월29일	192
02월05일	230
02월12일	233
02월19일	217
02월26일	185
03월05일	217

- shopping index 중에서 year to date 기간동안의
- “주문시간” Field를 주별로 나눈 후,
- 주별 documents 개수

1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. Date Field의 값을 기준으로 bucket을 생성하기 위해 ②를 눌러서 Date Histogram 선택
3. ③을 눌러서 ②를 적용할 대상 Field 선택
4. ④를 눌러서 ③을 나눌 간격 설정
5. ⑤를 눌러서 시작화

Data Table

The screenshot shows the Elasticsearch Data Table interface with the following components and annotations:

- Top Bar:** "shopping" index, "Data" tab selected, "Options" button.
- Metrics Section (①):** "Metric" dropdown set to "Count".
- Aggregation Section (②):** "Sum" aggregation for field "상품가격".
- Aggregation Section (③):** "Average" aggregation for field "상품가격".
- Buckets Section (④):** "Split Rows" button, bucket key "주문시간 per week", "Add sub-buckets" button.
- Play Button (⑤):** A large play button with a circled "5" above it.

주문시간 per week	Count	Sum of 상품가격	Average 상품가격
01월01일	166	2,890,000	17,409.6
01월08일	177	2,823,000	15,949.2
01월15일	161	2,721,000	16,900.6
01월22일	180	3,144,000	17,466.7
01월29일	192	3,176,000	16,541.7
02월05일	230	3,732,000	16,226.1
02월12일	233	4,211,000	18,073
02월19일	217	3,575,000	16,474.7
02월26일	185	3,276,000	17,708.1
03월05일	217	3,655,000	16,843.3

- shopping index 중에서 year to date 기간동안의
- “주문시간” Field를 주별로 나눈 후,
- 주별 documents 개수 // “상품가격”의 합 // “상품가격” 평균

1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. ②에서 metric aggregation 추가 설정
3. ③에서 metric aggregation 추가 설정
4. ④는 앞 설정 유지
5. ⑤를 눌러서 시작화

Data Table

The screenshot shows the Elasticsearch Data Table interface. At the top, there's a header with 'shopping' and a play button labeled '⑤'. Below it, the 'metrics' section has 'Metric' selected under 'Aggregation'. A dropdown labeled '① Percentiles' is open, showing '상품가격' (product price) under 'Field'. Another dropdown labeled '③ Percents' shows '50' and '99'. In the 'buckets' section, '④ Terms' is selected, with '연령대' (age group) under 'Field'. Under 'Order By', 'Term' is selected. The 'Order' dropdown shows 'Descending' and a 'Size' input field with '5'. The entire interface is highlighted with pink boxes and numbers.

A screenshot of a data table showing the results of the search. The columns are '연령대' (Age Group), '50th percentile of 상품가격' (50th percentile of product price), and '99th percentile of 상품가격' (99th percentile of product price). The data is as follows:

연령대	50th percentile of 상품가격	99th percentile of 상품가격
10대	17,000	29,000
20대	16,000	29,000
30대	17,000	29,000
40대	17,000	29,000
50대 이상	17,000	29,000

- shopping index 중에서 year to date 기간동안의
- “연령대” Field를 기준으로 나눈 후,
- 각 “연령대” bucket에서 “상품가격” Field의 50 백분위수 // 99 백분위수

1. ①을 눌러서 percentiles aggregation 선택
2. ②을 눌러서 ①를 적용할 대상 Field 선택
3. 구하고자 하는 백분위수를 ③에 입력
4. ④에서 term aggregation 설정
5. ⑤를 눌러서 시작화

Data Table - 예제 1

Documents 개수가 많은 상위 5개 “구매사이트” 별로 다음을 표시해보자

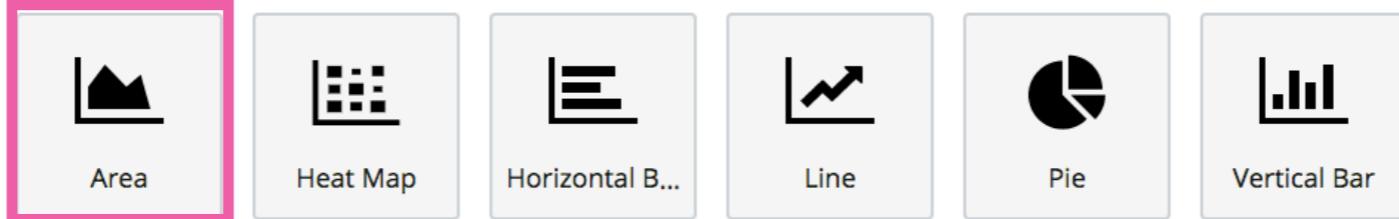
- “판매자평점” Field의 평균
- “배송소요시간” Field가 가장 컸던 10개 Documents의 “판매자평점” Field의 평균
- “배송소요시간” Field가 가장 작았던 10개 Documents의 “판매자평점” Field의 평균
- “고객연령” Field가 가장 컸던 10개 Documents의 “판매자평점” Field의 평균
- “고객연령” Field가 가장 작았던 10개 Documents의 “판매자평점” Field의 평균

사이트	평균평점 ↓	배송소요시간↑ ↓	배송소요시간↓ ↑	연령↑ ↓	연령↓ ↑
11번가	2.6	2.4	3.4	2.5	2.3
g마켓	2.7	2.6	2.7	2.5	2.4
쿠팡	2.7	2.6	2.9	3.4	2.5
옵션	2.8	2.5	2.8	2.9	2.6
위메프	2.7	3.3	2.6	2.6	2.8

Select visualization type

Search visualization types...

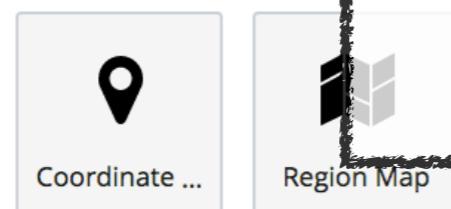
Basic Charts



Data

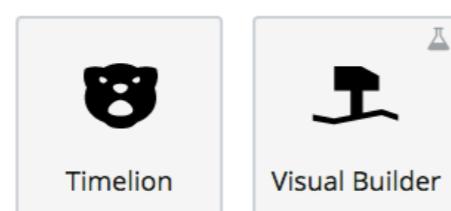


Maps

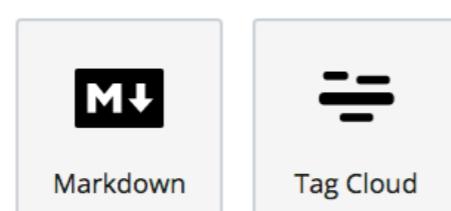


Visualize - Area

Time Series

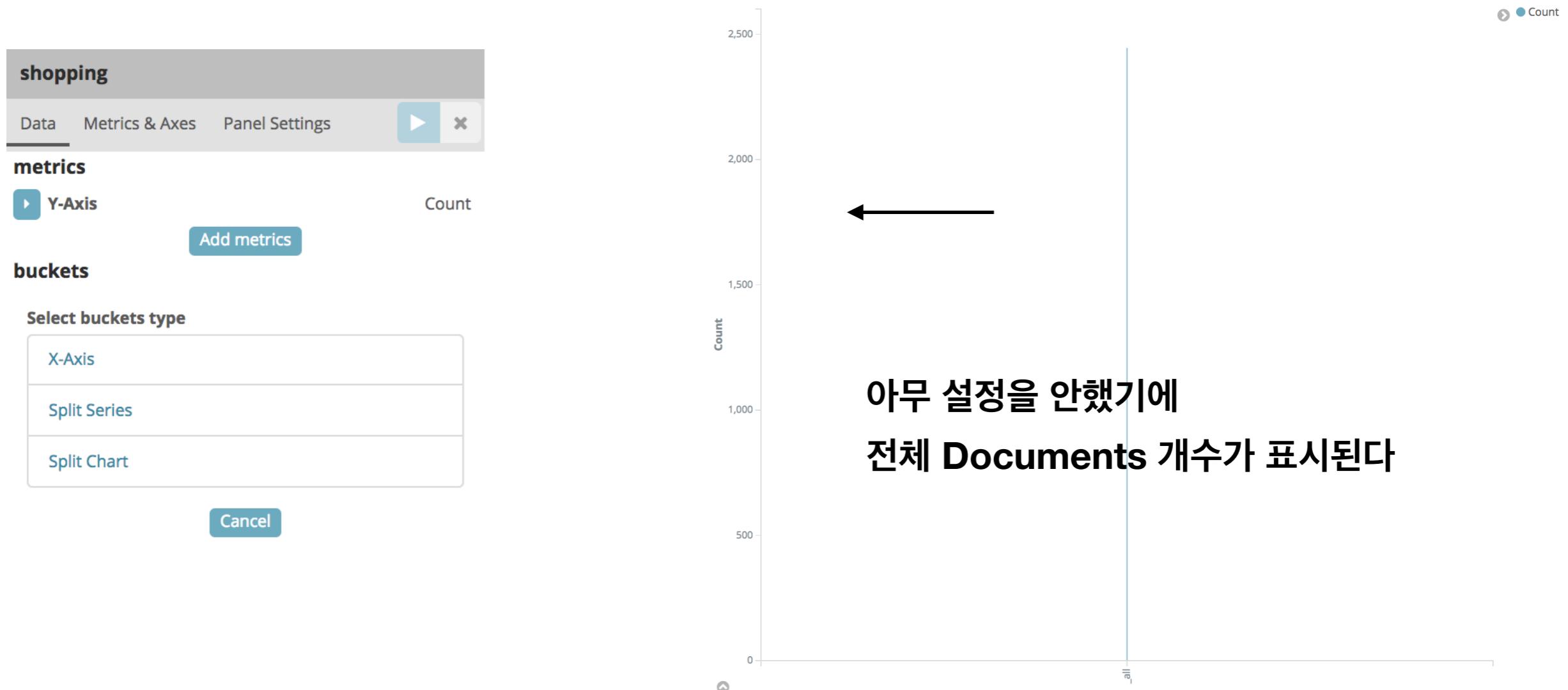


Other



Area

- 전체와 그를 구성하는 요소를 동시에 보여주어 구성요소의 중요도를 파악하는데 용이
- Area/Line/Bar Chart는 UI만 다를 뿐 큰 차이 없음



Area

shopping

Data Metrics & Axes Panel Settings  

metrics

 Y-Axis  Count

buckets

 X-Axis  

Aggregation

② Date Histogram

Field

③ 주문시간

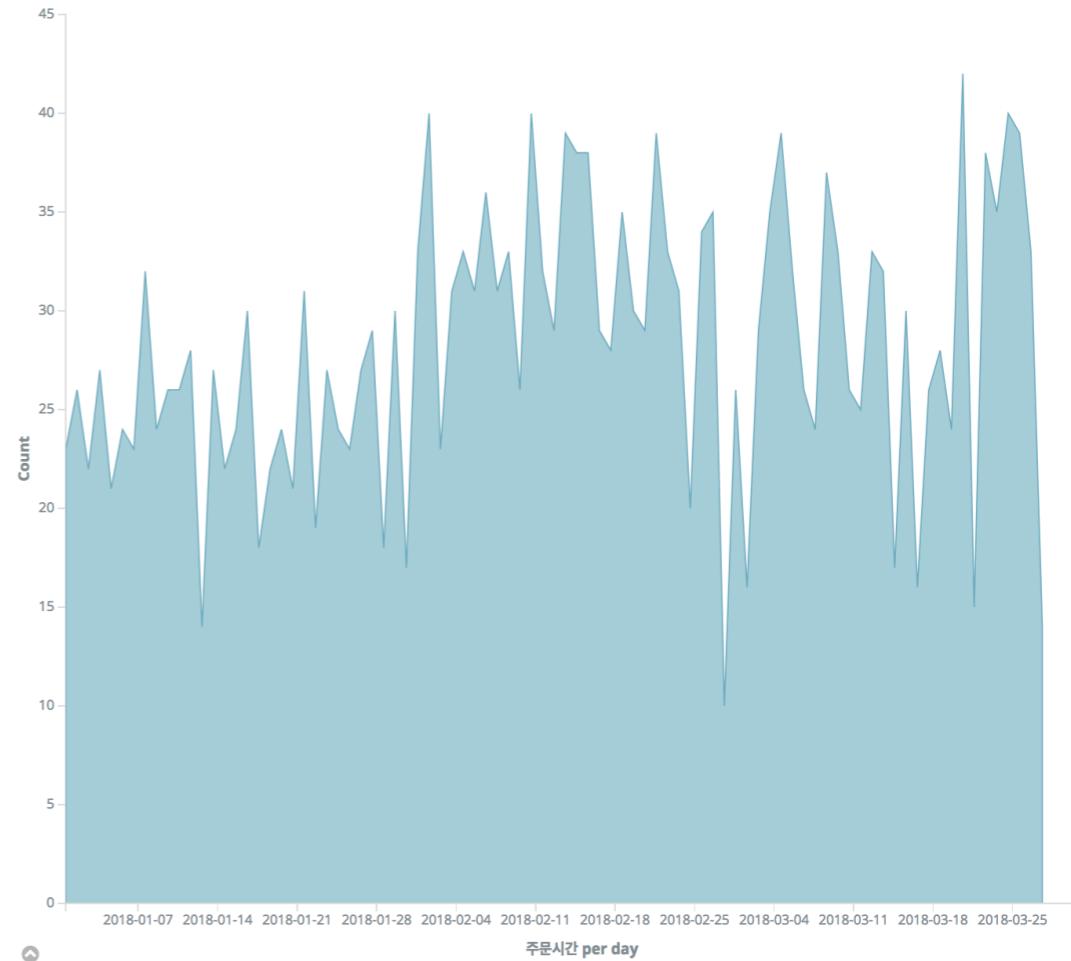
Interval

④ Daily

Custom Label

 Advanced

Add sub-buckets



- shopping index 중에서 year to date 기간동안의
- “주문시간” Field를 일별로 나눈 후,
- 일별 documents 개수 시각화

1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. Date Field의 값을 기준으로 bucket을 생성하기 위해 ②를 눌러서 Date Histogram 선택
3. ③을 눌러서 ②를 적용할 대상 Field 선택
4. ④를 눌러서 ③을 나눌 간격 설정
5. ⑤를 눌러서 시각화

Area



1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. ②는 26 page 설정 유지
3. ③을 눌러서 Series를 Split할 Term Aggregation 설정
4. ④를 눌러서 시각화

Area

The screenshot shows the Kibana interface for a search named "shopping". The search bar at the top contains the term "shopping". Below it, the "Metrics & Axes" tab is selected. The "Y-Axis" section has "Count" selected. The "X-Axis" section has "주문시간 per day" selected. The "Split Series" section has "고객성별: Descending" selected. The "Split Chart" section is expanded, showing "Rows" selected, and the "Sub Aggregation" section with "Terms" selected. The "Field" section shows "구매사이트" selected. The "Order By" section shows "metric: Count" selected, "Order" set to "Descending", and "Size" set to 2. The "Custom Label" section shows "사이트" selected. At the bottom, there is an "Advanced" button.

① Count
② 주문시간 per day
③ 고객성별: Descending
④ Split Chart
⑤ Advanced

왜 7개가 나올까??

The resulting chart displays seven stacked area series representing different customer segments over time. The segments are color-coded and stacked vertically. The x-axis represents time from January 7, 2018, to March 25, 2018. The y-axis represents the count of documents.

List of points:

- shopping index 중에서 year to date 기간동안의
- “주문시간” Field를 일별로 나눈 후,
- “고객성별” 별로
- “구매사이트” 상위 2개 별
- documents개수 ???

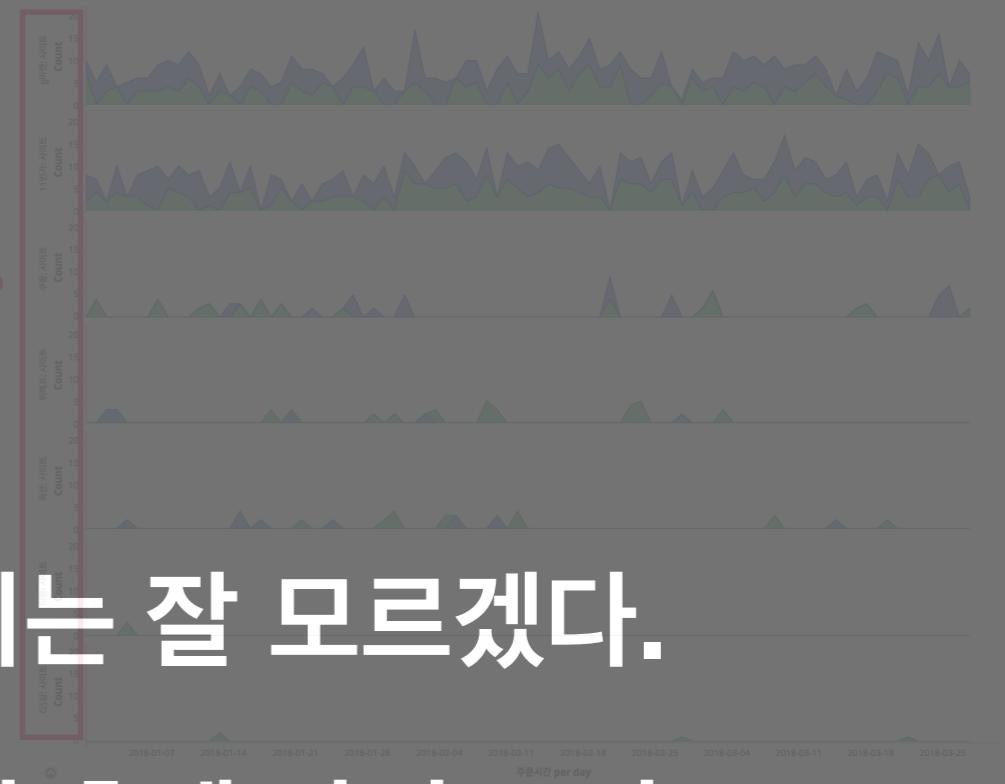
1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. ②~③는 page 26~27 설정 유지
3. ④을 눌러서 Chart를 Split할 Term Aggregation 설정
4. ⑤를 눌러서 시각화

Area

The screenshot shows the Kibana interface for the 'shopping' index. At the top, there's a search bar with placeholder text 'Search shopping' and a date range selector from '2018-01-01' to '2018-03-25'. Below the search bar is a card-based search interface. The first card has a title 'shopping' with a play button and a close button. The second card is titled 'metrics' with a Y-axis icon, showing a count of 1. The third card is titled 'buckets' with an X-axis icon, showing '주문시간 per day' with a descending order option. The fourth card is titled 'Split Series' with a split chart icon, showing '고객성별: Descending' with a descending order option. The fifth card is titled 'Sub Aggregation' with a terms icon, showing '구매사이트' with a count of 2. The bottom section has an 'Order By' dropdown set to 'metric: Count', an 'Order' dropdown set to 'Descending', a 'Size' input field with value '2', and a 'Custom Label' input field with value '사이트'. A 'Advanced' button is at the bottom right.

왜 7개가 나올까??

왜 카드가 7개가 나왔는지는 잘 모르겠다.

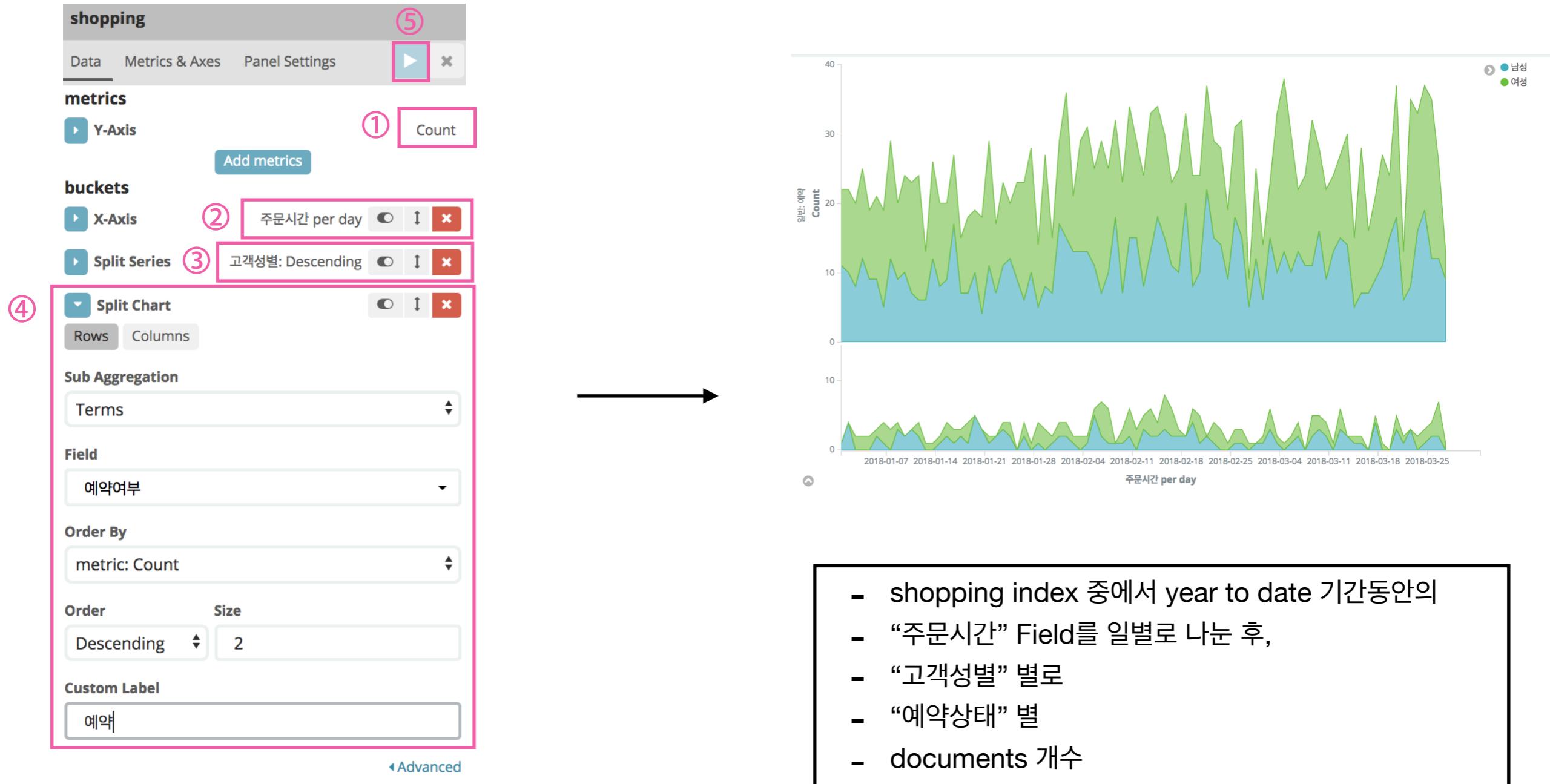


우선 설정을 변경하고 잠시 후에 다시 보자.

- shopping index 중에서 year to date 기간동안의
- “주문시간” Field를 일별로 나눈 후,
- “고객성별” 별로
- “구매사이트” 상위 2개 별
- documents개수 ???

1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. ②~③는 page 26~27 설정 유지
3. ④을 눌러서 Chart를 Split할 Term Aggregation 설정
4. ⑤를 눌러서 시각화

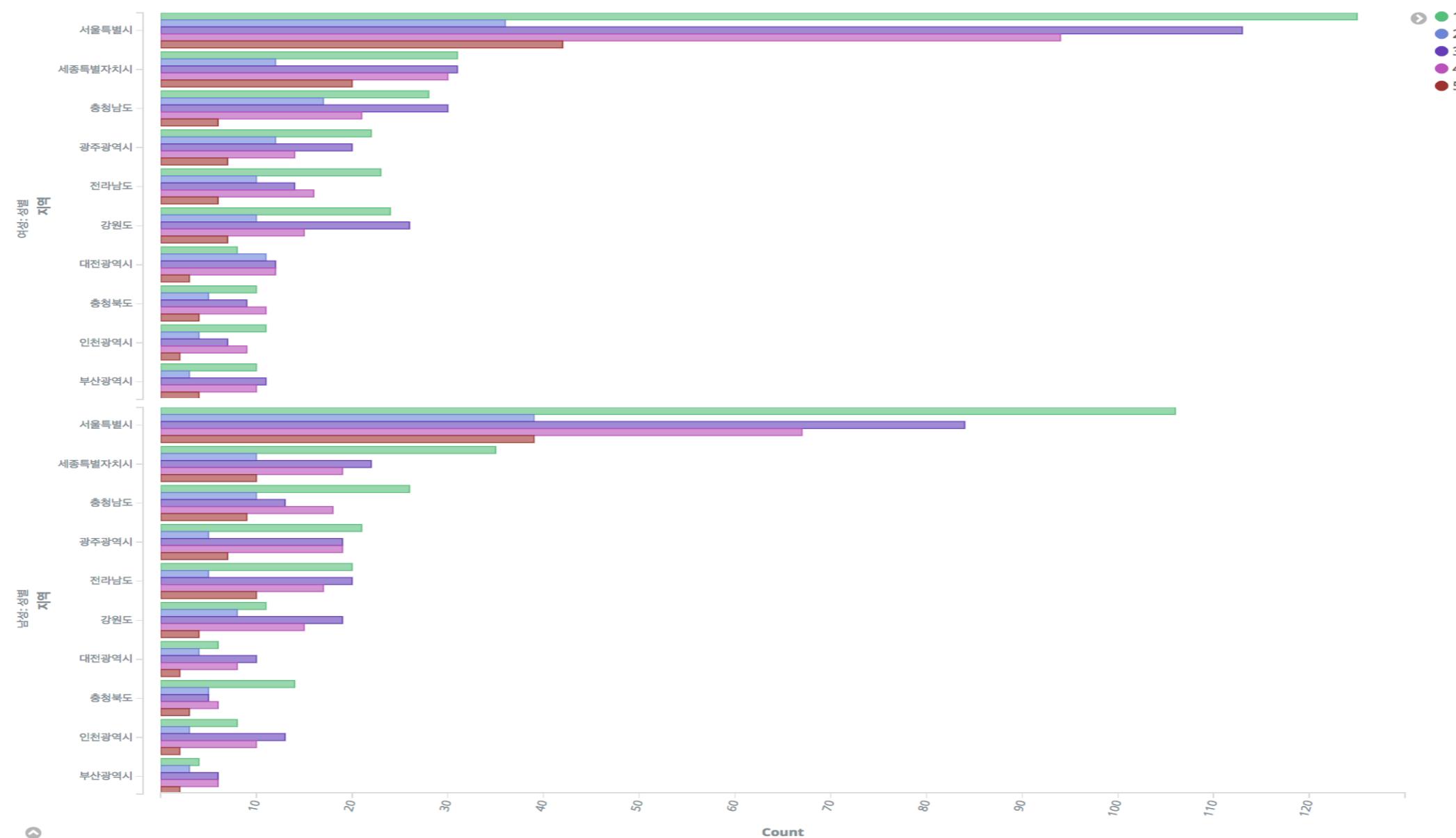
Area



1. metric은 value count로 고정하기 위해 ①은 원래 상태로 두자
2. ②~③는 page 26~27 설정 유지
3. ④을 눌러서 Chart를 Split할 Term Aggregation 설정
4. ⑤를 눌러서 시작화

Data Table - 예제 1

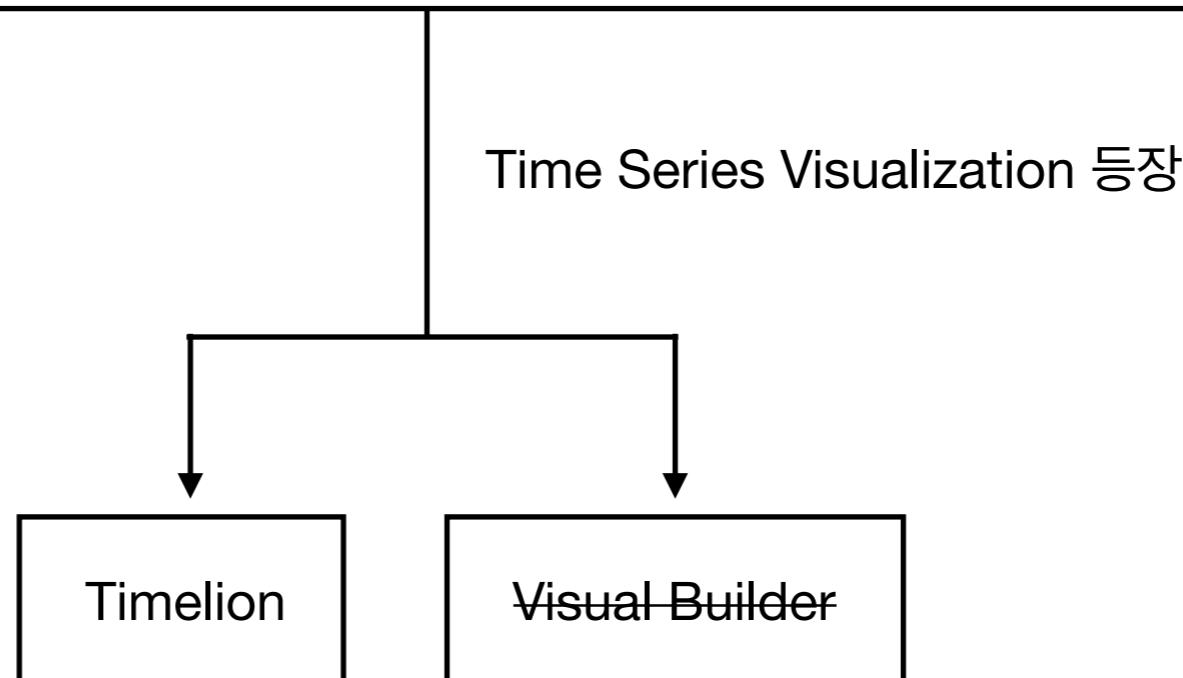
“판매자평점” Field의 평균이 높은 “고객주소_시도” 10개에 대해서 남성, 여성 각각의 Documents의 개수를 “판매자평점” 값에 따라 (Interval=1) 나누어 표시하자



Line/Bar/Area Chart로 시계열 데이터를 시각화하면서 아쉬운 점?

- 서로 다른 Index (Pattern) 의 데이터를 하나의 Visualization으로 시각화 못한다
- 특정 Field Value의 기준시점과 비교시점 값을 시각적으로 비교하기 어렵다

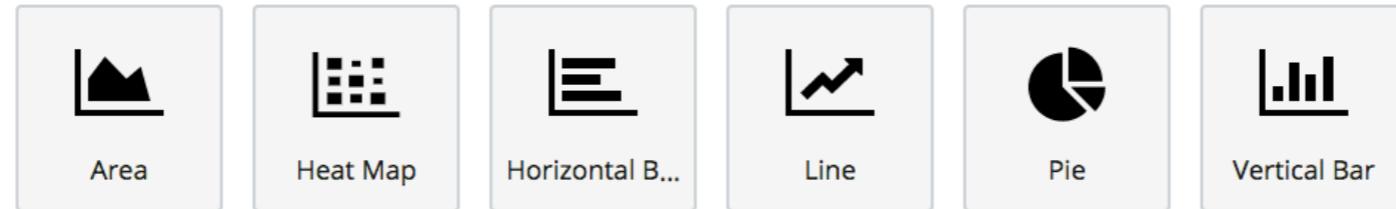
:



Select visualization type

Search visualization types...

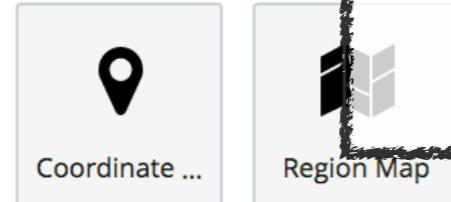
Basic Charts



Data



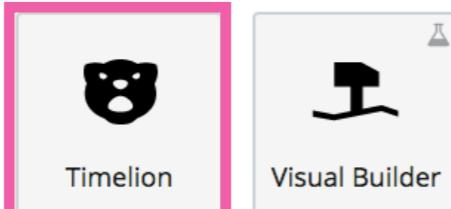
Maps



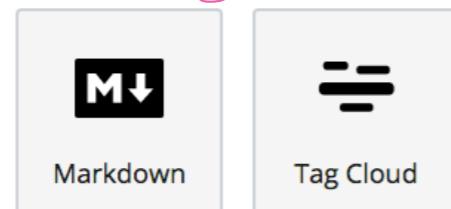
Visualize - Timelion

* 생소한 문법 주의

Time Series



Other



기본적으로 아래와 같은 함수를 제공한다

목록

- 기본함수 - `.es()`
- 조건함수 - `.es().if()`
- 수학함수 - `.es().multiply()`
- 수학함수 - `.es().divide()`
- 수학함수 - `.es().subtract()`
- 수학함수 - `.es().sum()`
- 수학함수 - `.es().abs()`
- 수학함수 - `.es().log()`
- 수학함수 - `.es().max()`
- 수학함수 - `.es().min()`
- 수학함수 - `.es().static()`
- 수학함수 - `.es().cusum()`
- 수학함수 - `.es().derivative()`
- 수학함수 - `.es().movingaverage()`
- 수학함수 - `.es().scale_interval()`
- 수학함수 - `.es().range()`
- 수학함수 - `.es().trend()`
- 스타일함수 - `.es().bars()`
- 스타일함수 - `.es().lines()`
- 스타일함수 - `.es().points()`
- 스타일함수 - `.es().label()`
- 스타일함수 - `.es().color()`
- 스타일함수 - `.es().yaxis()`
- 스타일함수 - `.es().title()`



시계열 데이터를 시각화 하는 데 있어 **기존 Visualization 과의 차이**에 집중하자

Timelion

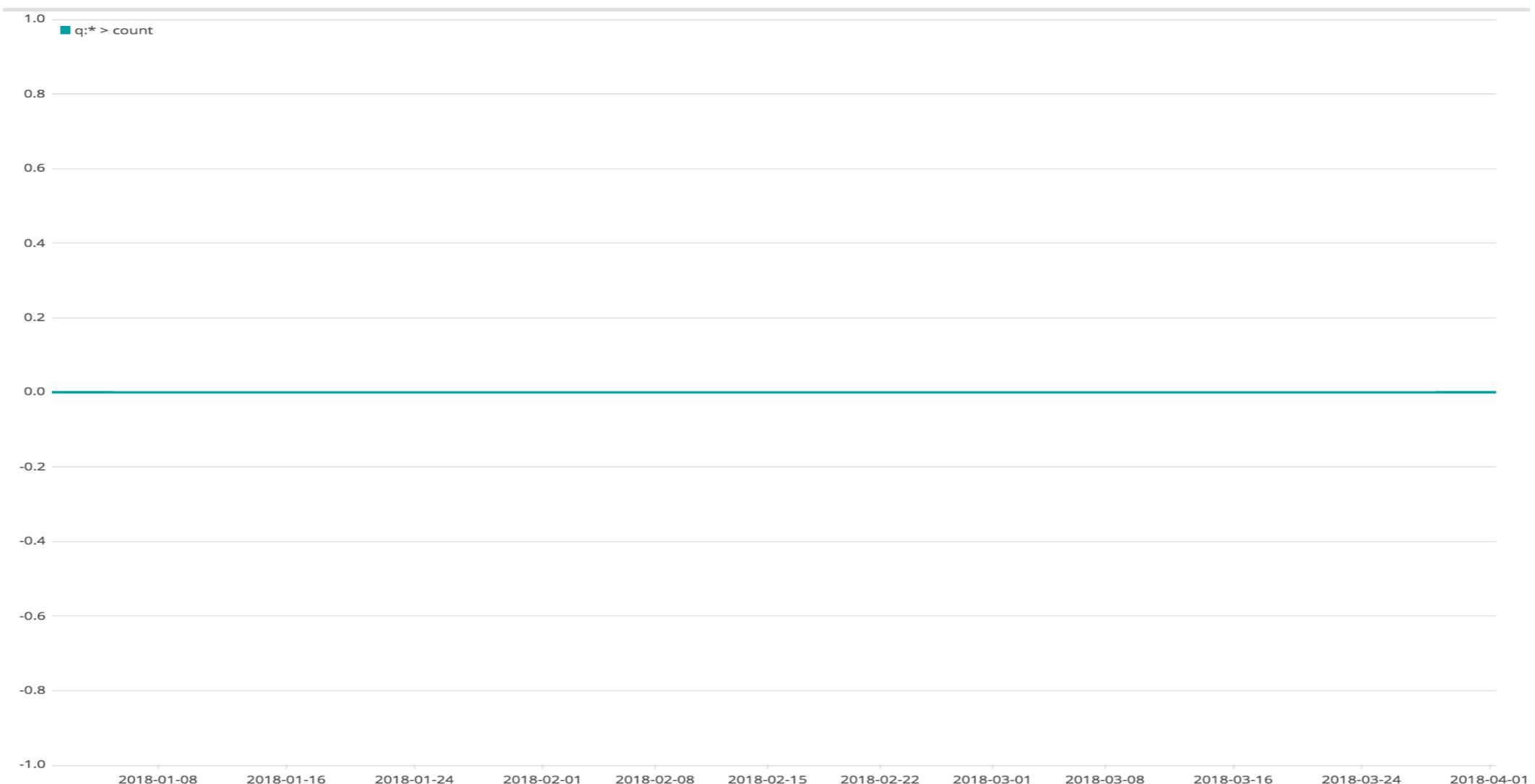


- ①을 통해 오른쪽 그래프에 표시할 데이터 간 간격 설정 (date histogram)
- ②에 Timelion Expression을 작성해서 시작화
- ③를 눌러서 시작화

Timelion - Index 설정

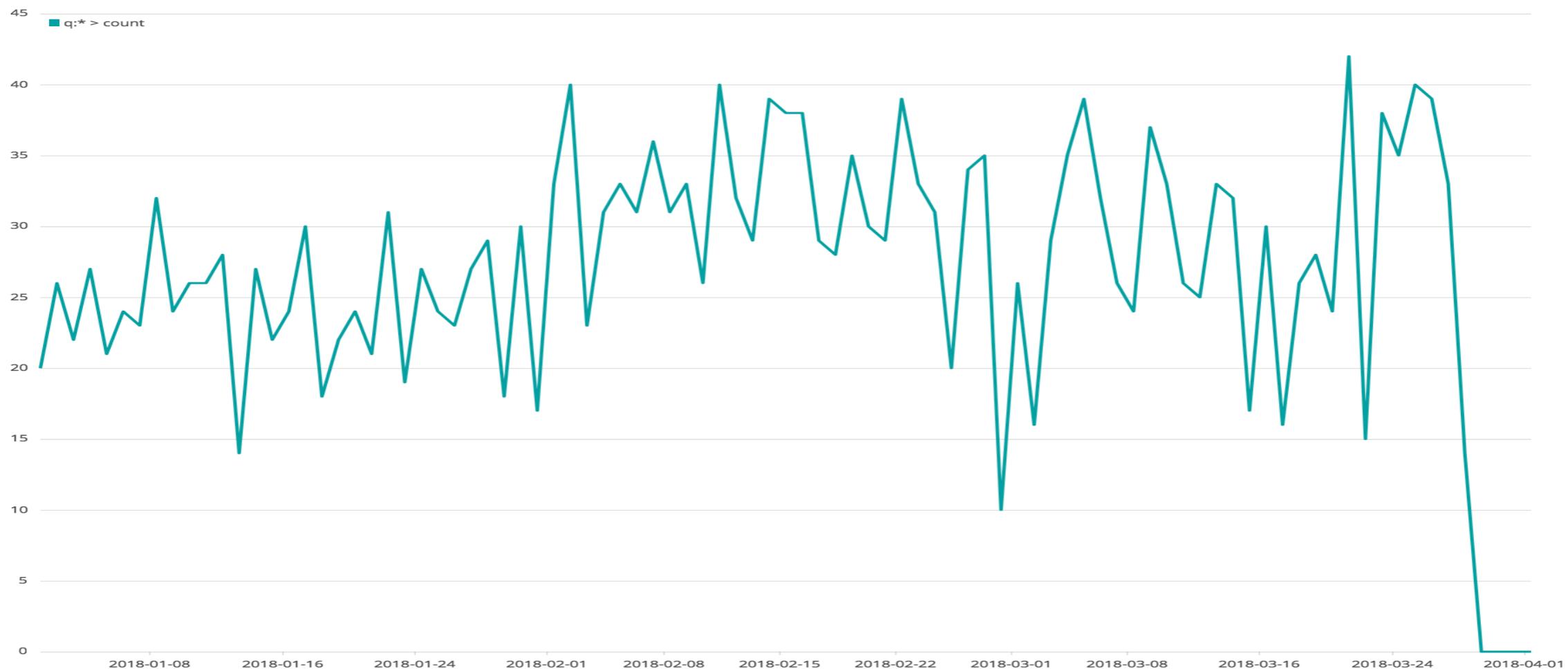
.es(index=shopping)

* index가 존재하면 Kibana에서 Index Patterns에 등록하지 않고도 사용할 수 있다



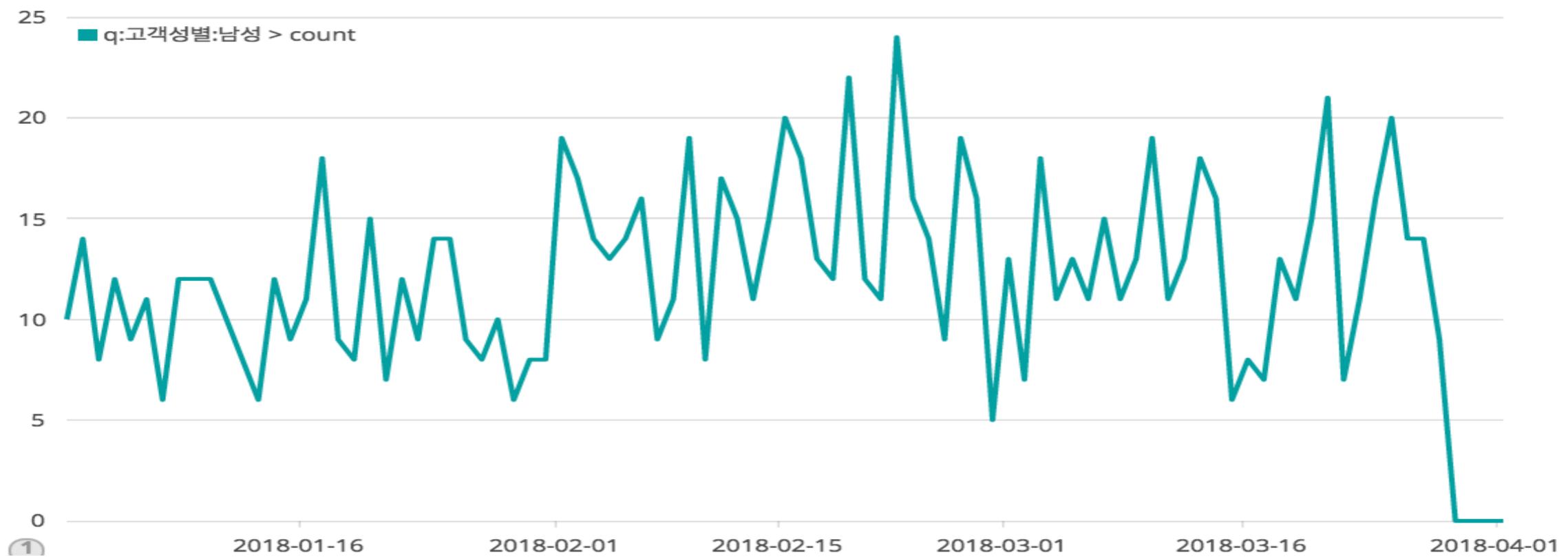
Timelion - Timefield 설정

.es(index=shopping, timefield=주문시간)



Timelion - Query 설정

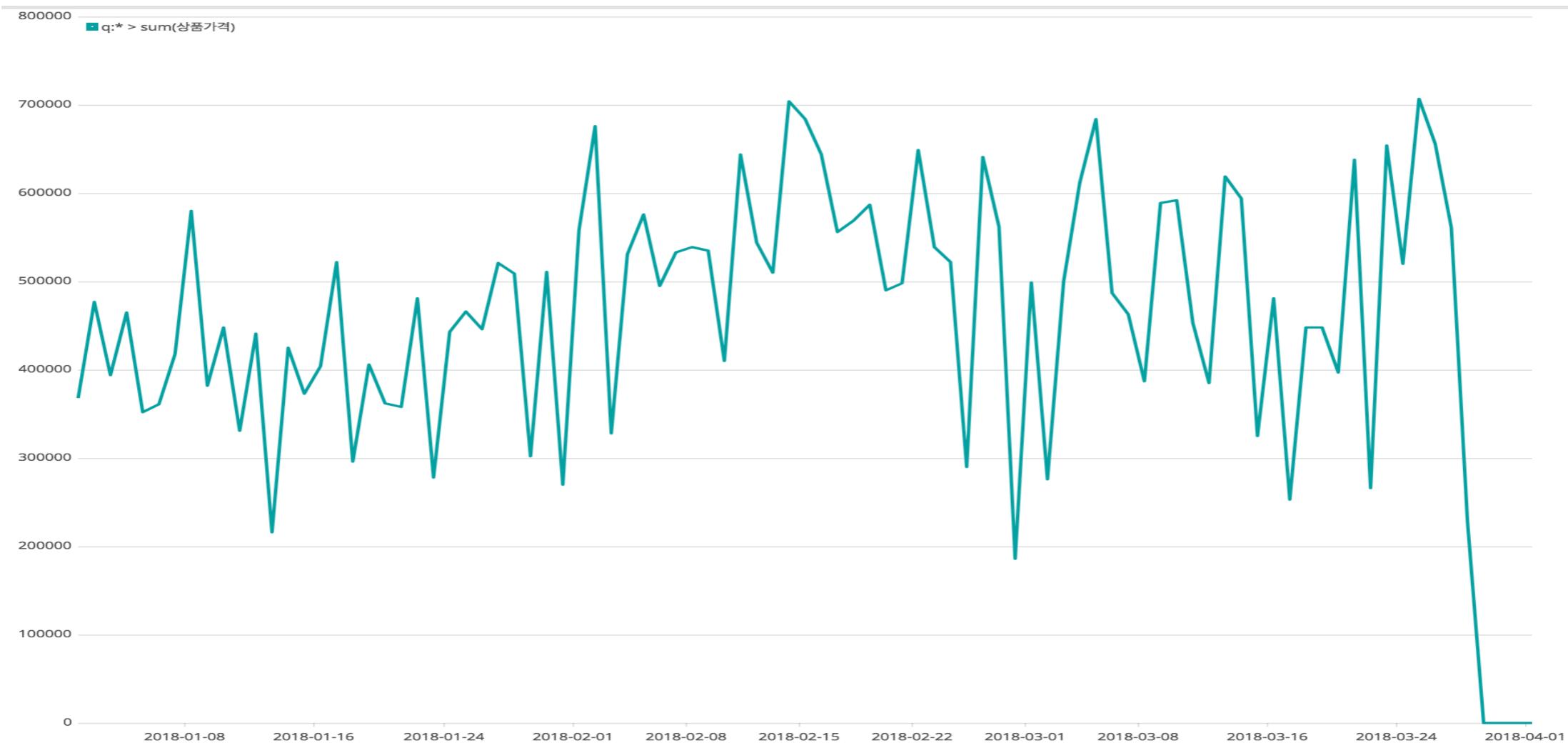
```
.es(index=shopping, timefield=주문시간, q=고객성별:남성)
```



Timelion - Metric 설정

```
.es(index=shopping, timefield=주문시간, metric=sum:상품가격)
```

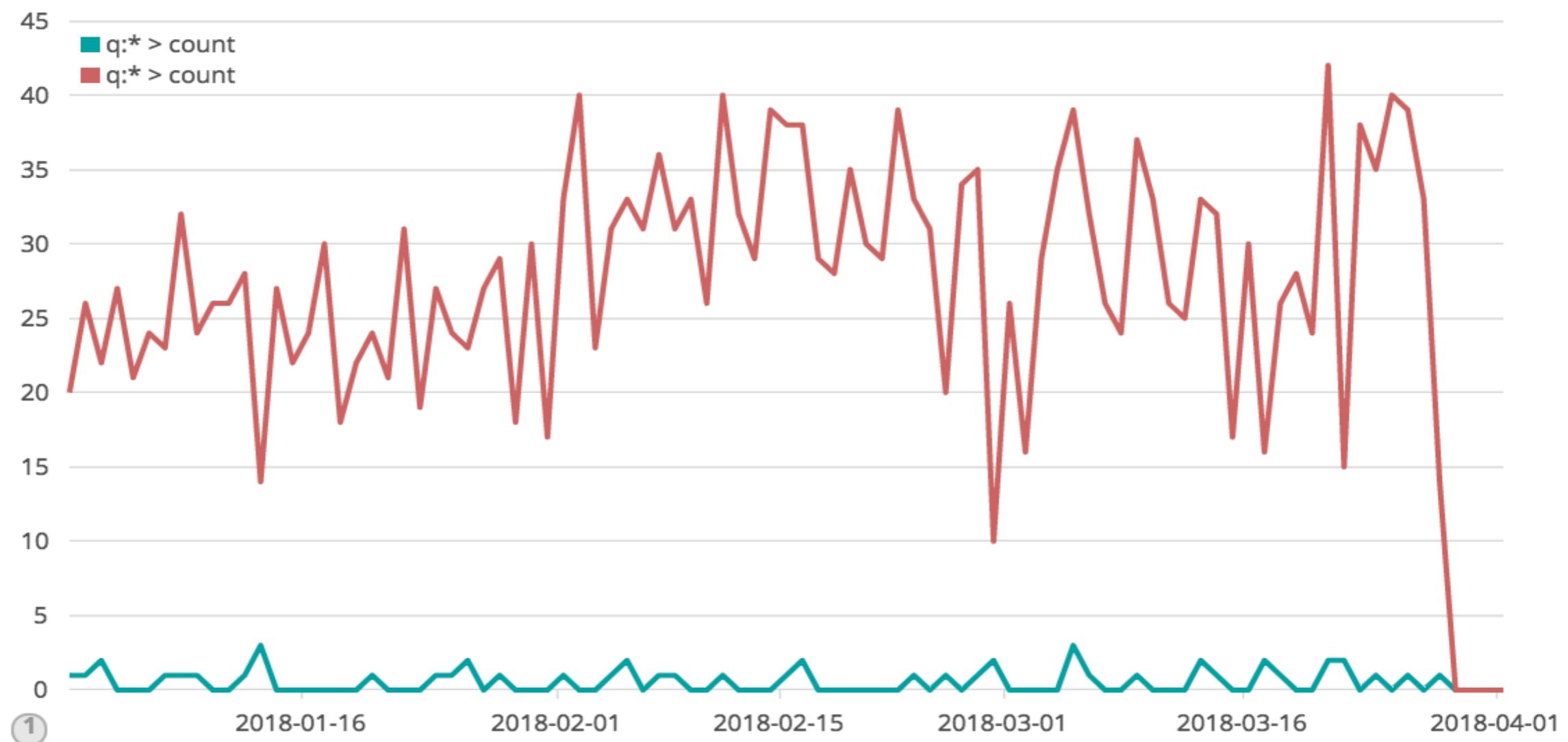
* default : count



Timelion - 서로 다른 Index

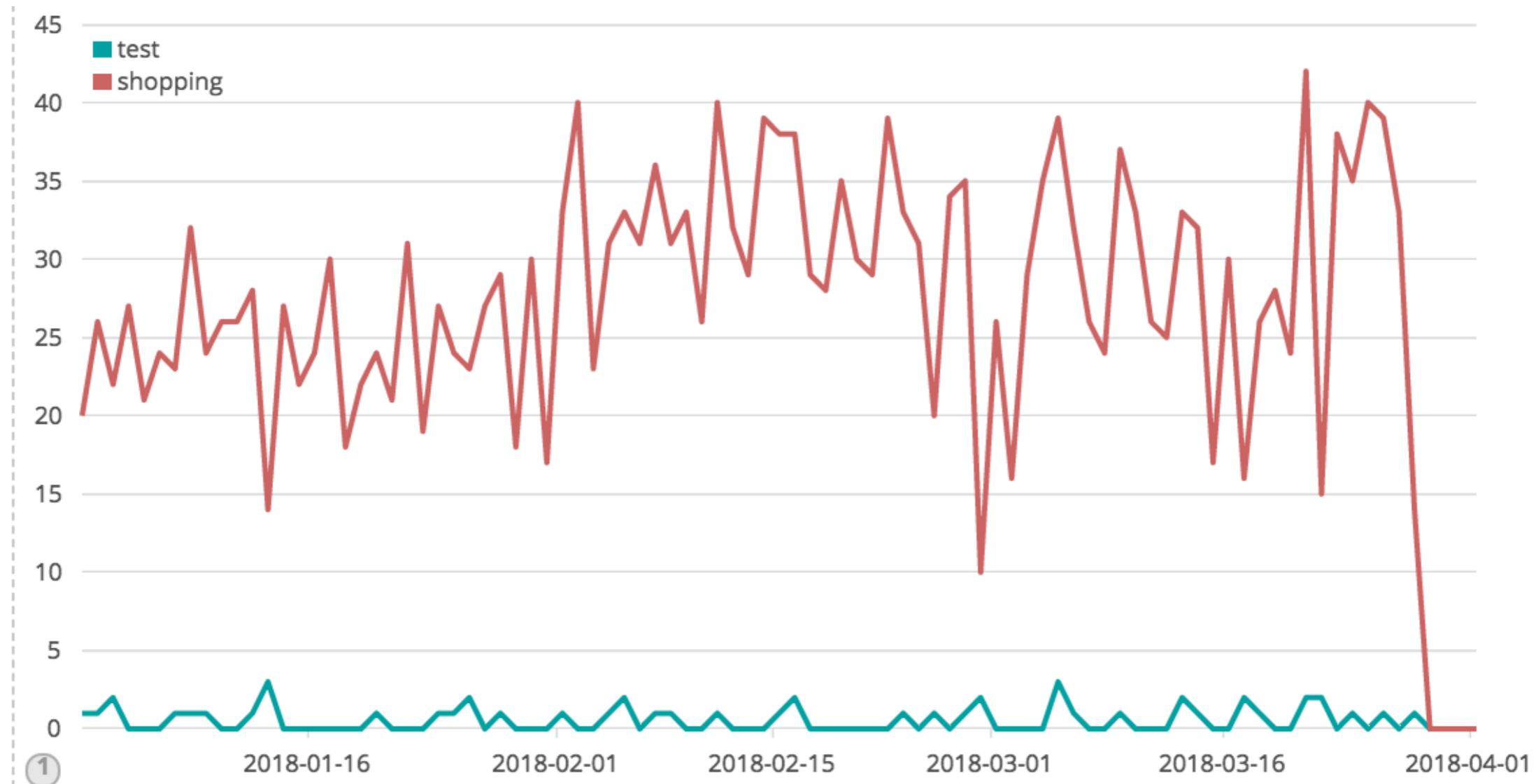
.es(index=test2_2018.03.01, timefield=주문시간), .es(index=shopping, timefield=주문시간)

“test2_2018.03.01”과 “shopping”과 같이 전혀 연관 없어 보이는 index도 통합해서 시각화



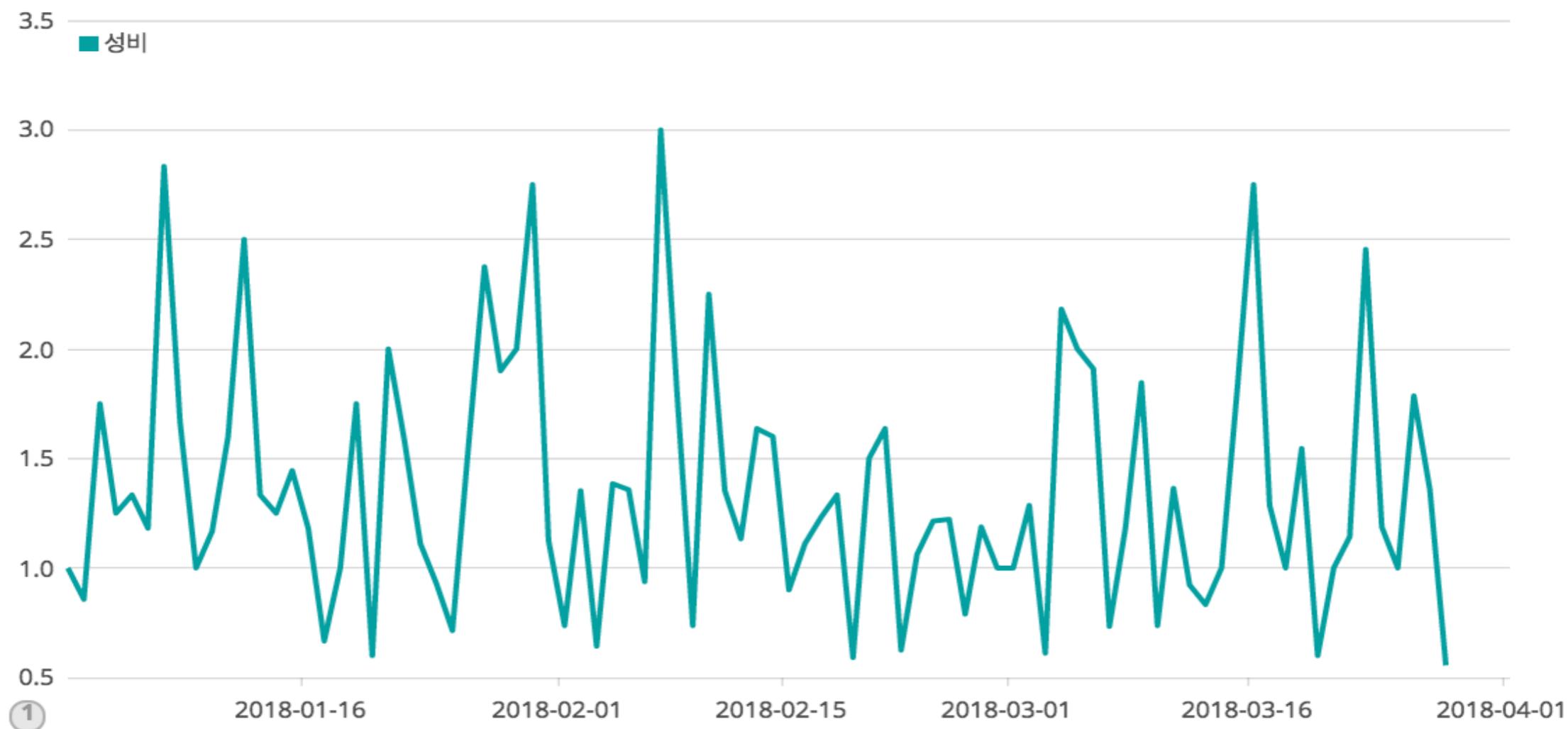
Timelion - Label

```
.es(index=test2_2018.03.01, timefield=주문시간).label(test), .es(index=shopping, timefield=주문시간).label(shopping)
```



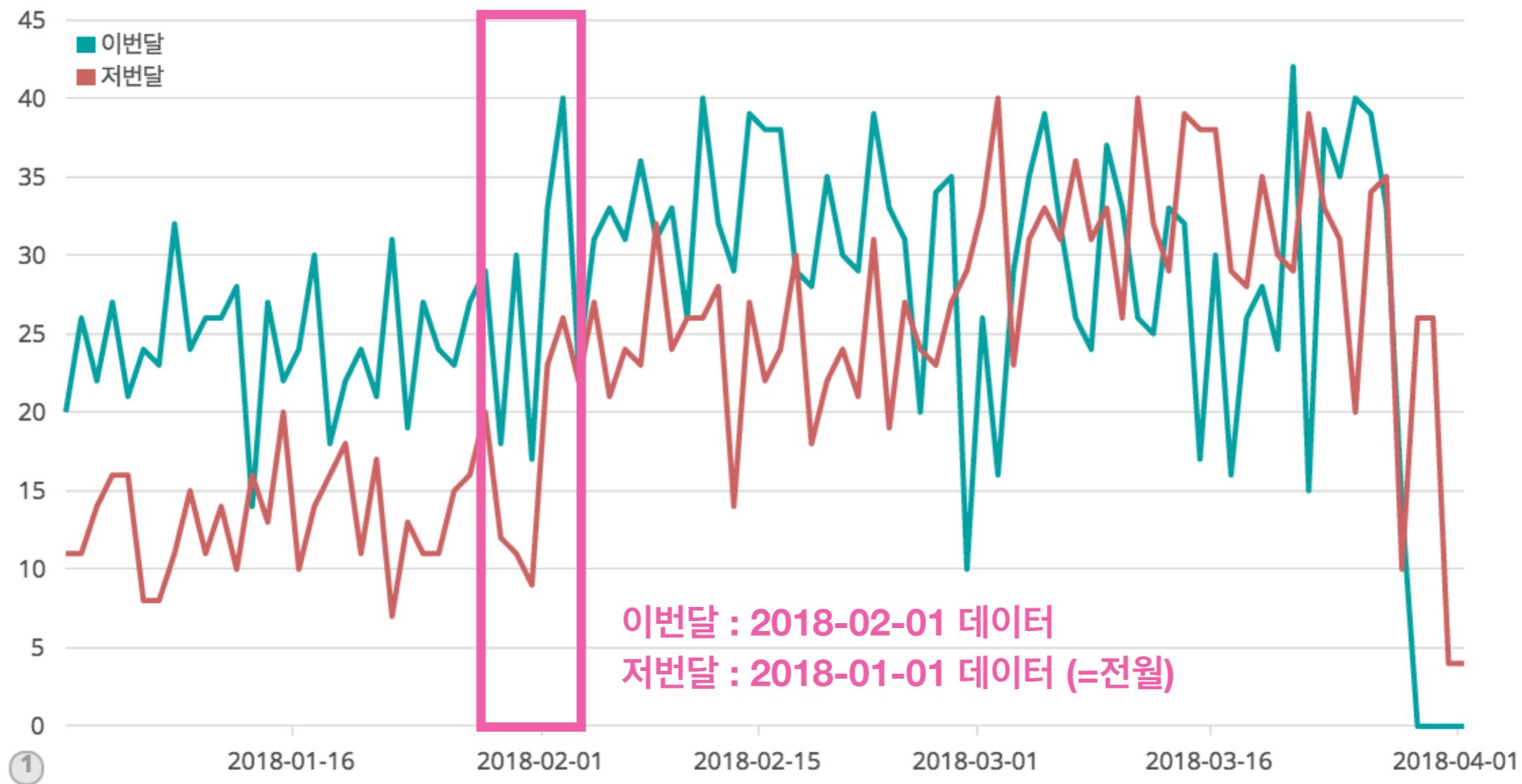
Timelion - divide

```
.es(index=shopping, timefield=주문시간, q=고객성별:여성).divide(.es(index=shopping, timefield=주문시간, q=고객성별:남성)).label(성비)
```



Timelion - Offset 설정

```
.es(index=shopping, timefield=주문시간).label(이번달), .es(index=shopping, timefield=주문시간, offset=-1M).label(저번달)
```



이 정도가 Timelion의 기본문법이며, 나머지는 를 참고해서 한 번씩 입력해보자. (수업 후에)

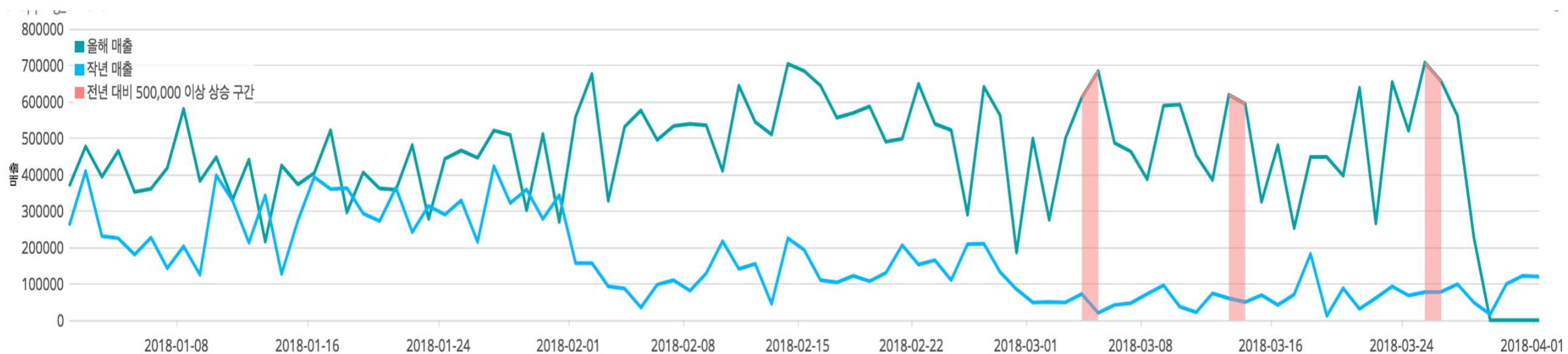
그리고, 다음 페이지에 나오는 몇 가지 예시를 보면서 Timelion을 마무리 하자.

이 때, Timelion 기능을 자신의 문제에 어떻게 응용/적용할 수 있을 지 생각해보자.

Timelion - 예시 1

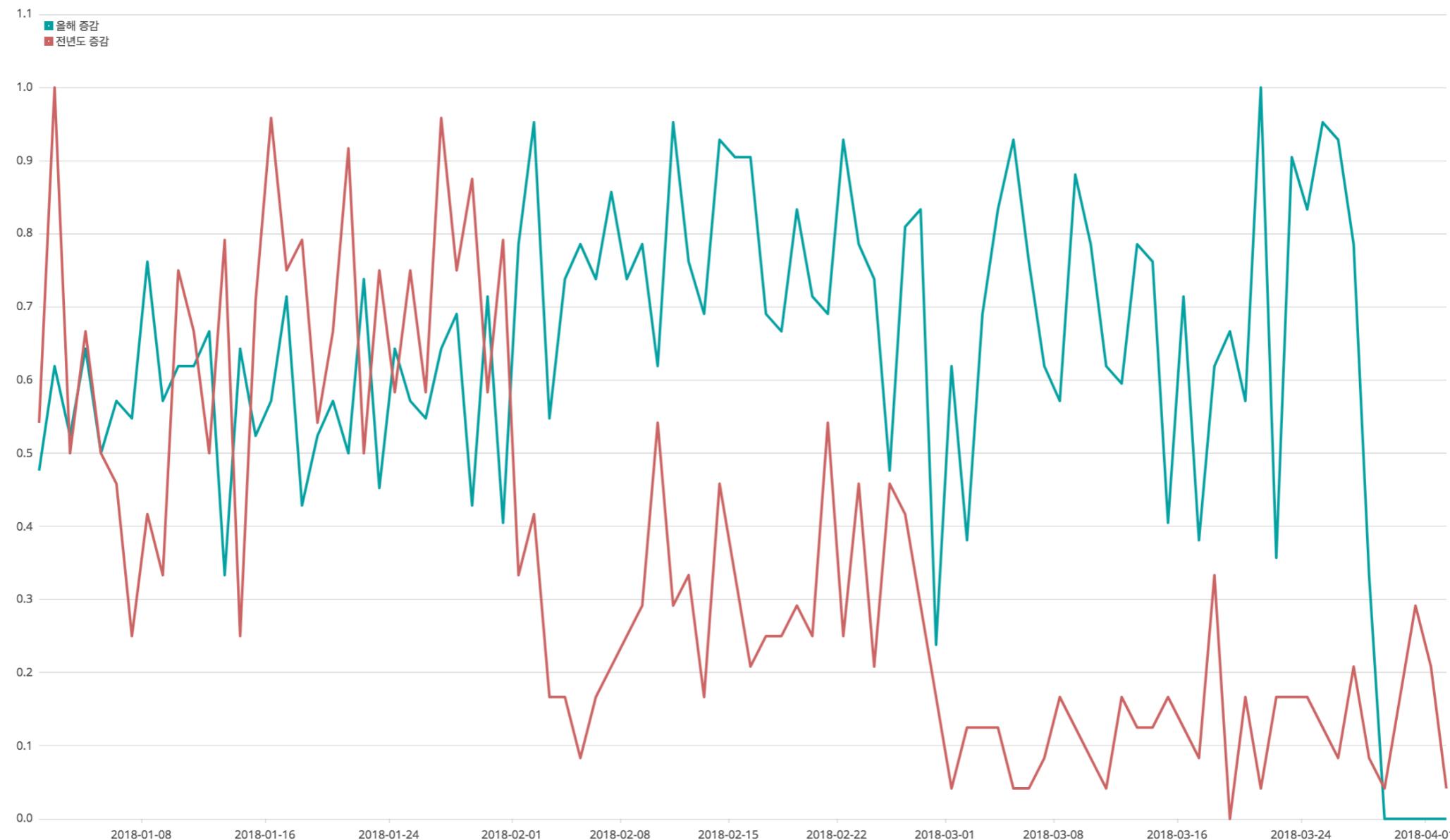


“올해 매출”과 “작년 매출”을 함께 표시하며 “올해 매출”이 “작년 매출” 보다 500,000 이상 큰 구간 탐지



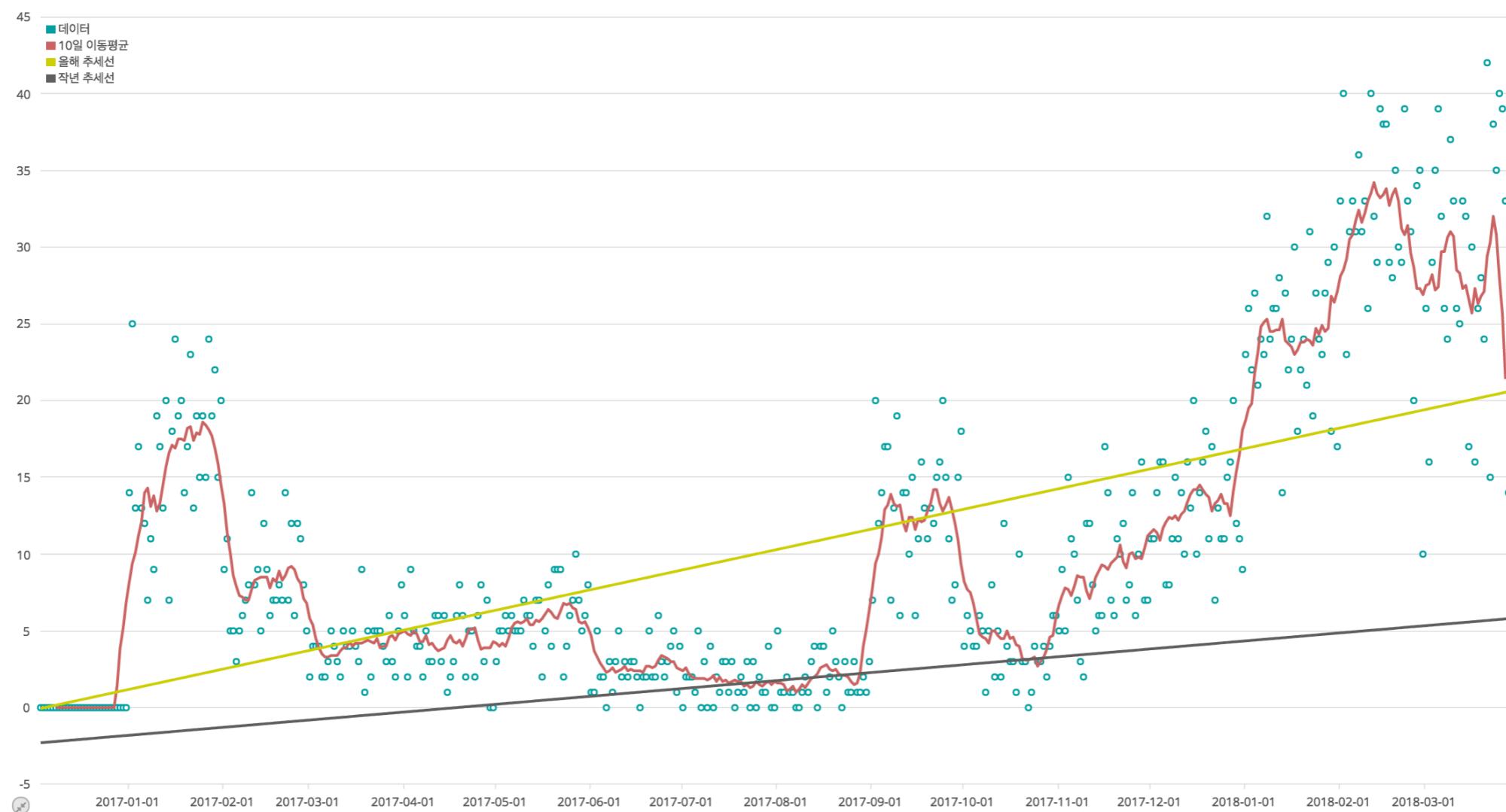
Timelion - 예시 2 ✎ 🎉

같은 기간 동안 전년과 올해 수치의 증감 추이 (0~1 사이로 정규화) 비교 분석



Timelion - 예시 3 ✎ 🤴

올해의 이동평균 데이터 및 추세선과 함께 같은 기간 동안 전년도의 추세선 시각화



Aggregation - Pipeline

Kibana에서 사용할 수 있는 Aggregation

= ~~Bucket + Metrics + Pipeline~~

(... 지난 주에 배웠고)

Pipeline Aggregation

Pipeline Aggregations

① 특징

Pipeline aggregations work on the outputs produced from other aggregations rather than from document sets, adding information to the output tree. There are many different types of pipeline aggregation, each computing different information from other aggregations, but these types can be broken down into two families:

Parent

② 종류

A family of pipeline aggregations that is provided with the output of its parent aggregation and is able to compute new buckets or new aggregations to add to existing buckets.

Sibling

Pipeline aggregations that are provided with the output of a sibling aggregation and are able to compute a new aggregation which will be at the same level as the sibling aggregation.

Aggregation - Parent Pipeline

종류	상세
Derivative	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, 연속한 Bucket 간의 차이를 구함
Cumulative Sum	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Bucket 값들의 누적합을 구함
Moving Average	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, {n개} Bucket 간의 평균을 구함
Serial Diff	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, {n번째 이전} Bucket 과의 차이를 구함

↓

(Date Histogram 또는 Histogram 만 지원)

다음과 같은 데이터가 있다고 하자

{"data" : 0}

{"data" : 1}

{"data" : 2}

{"data" : 3}

{"data" : 4}

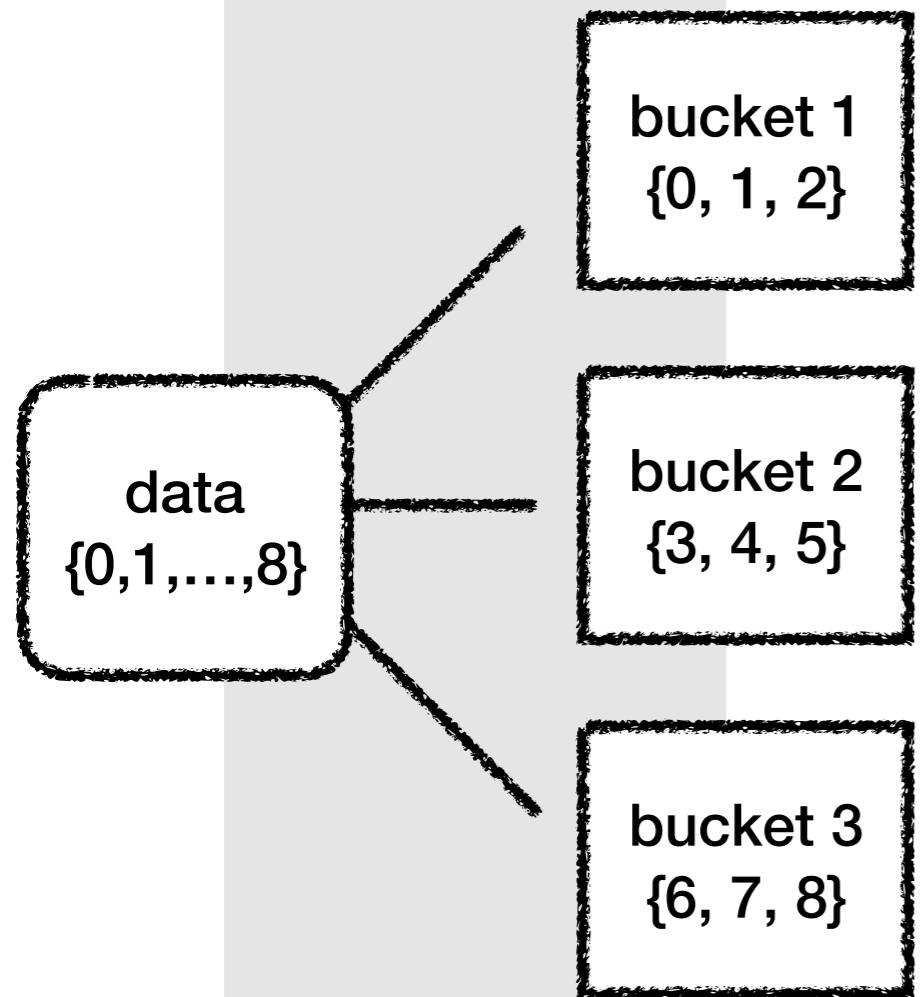
{"data" : 5}

{"data" : 6}

{"data" : 7}

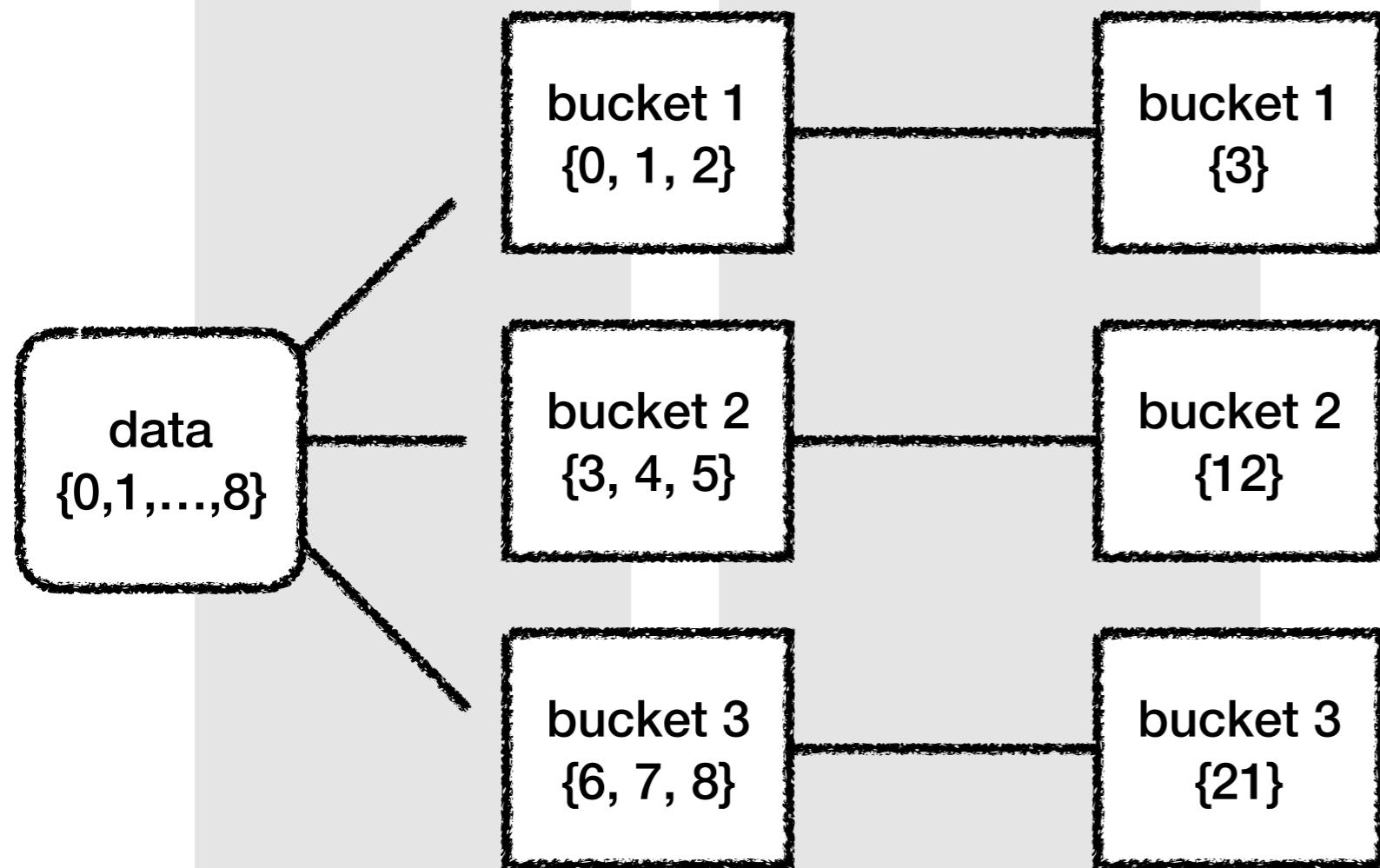
{"data" : 8}

Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

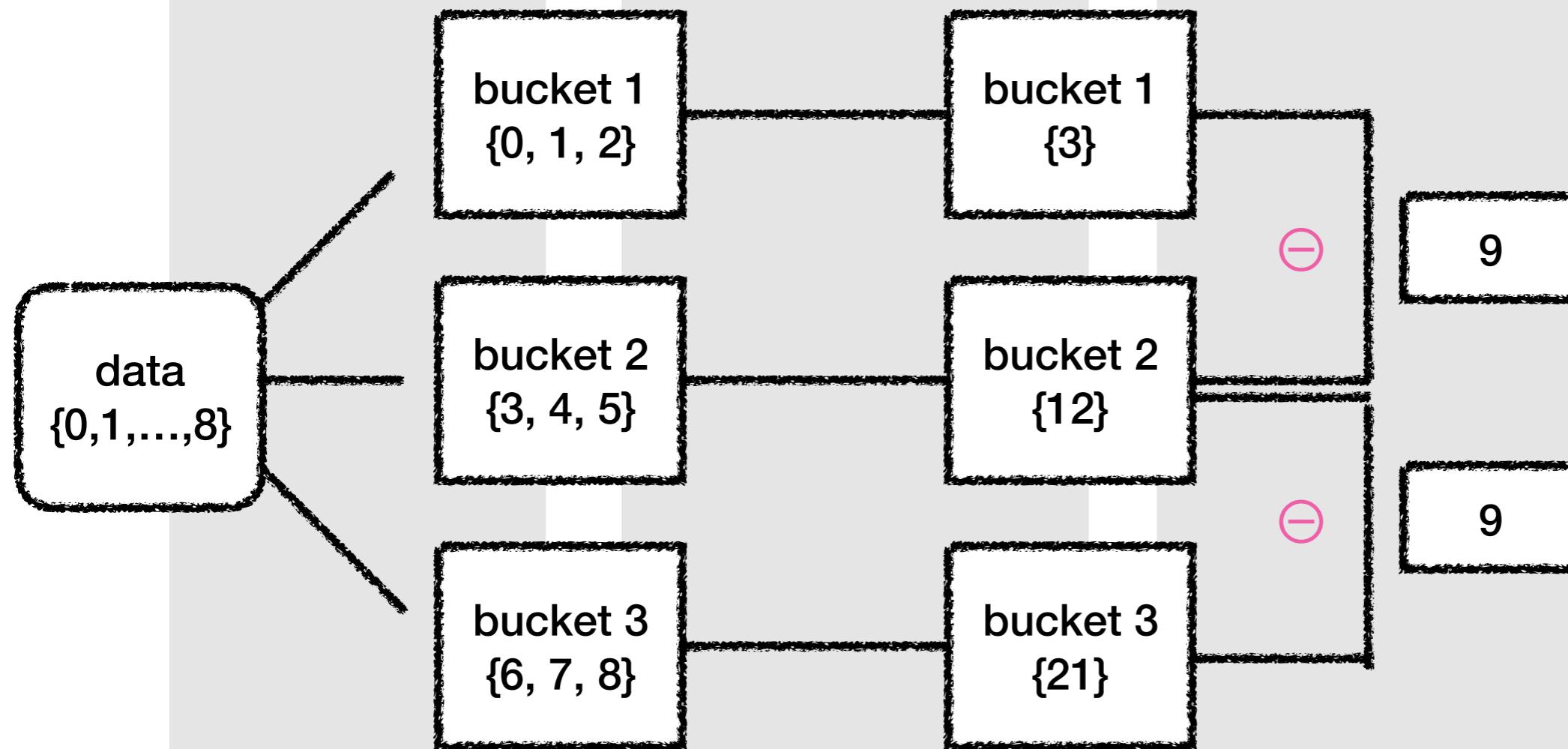
Metric Aggregation (Sum)



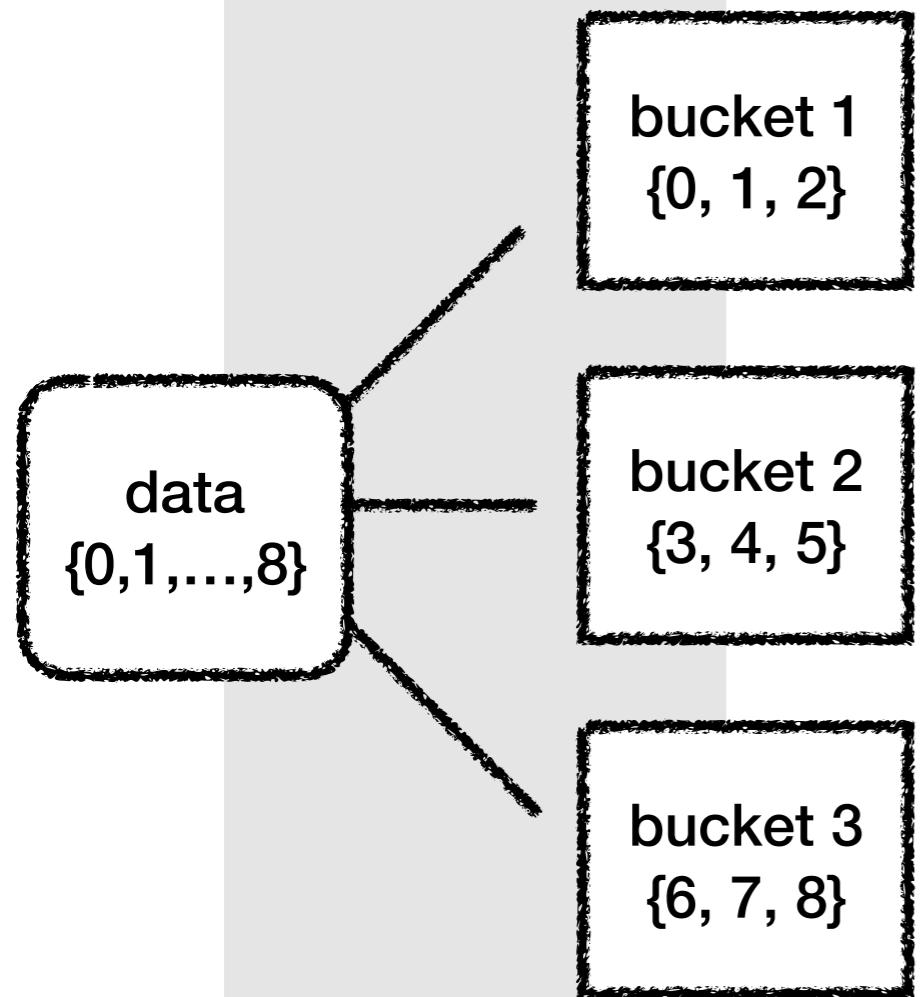
Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Parent Pipeline Aggregation (Derivative)

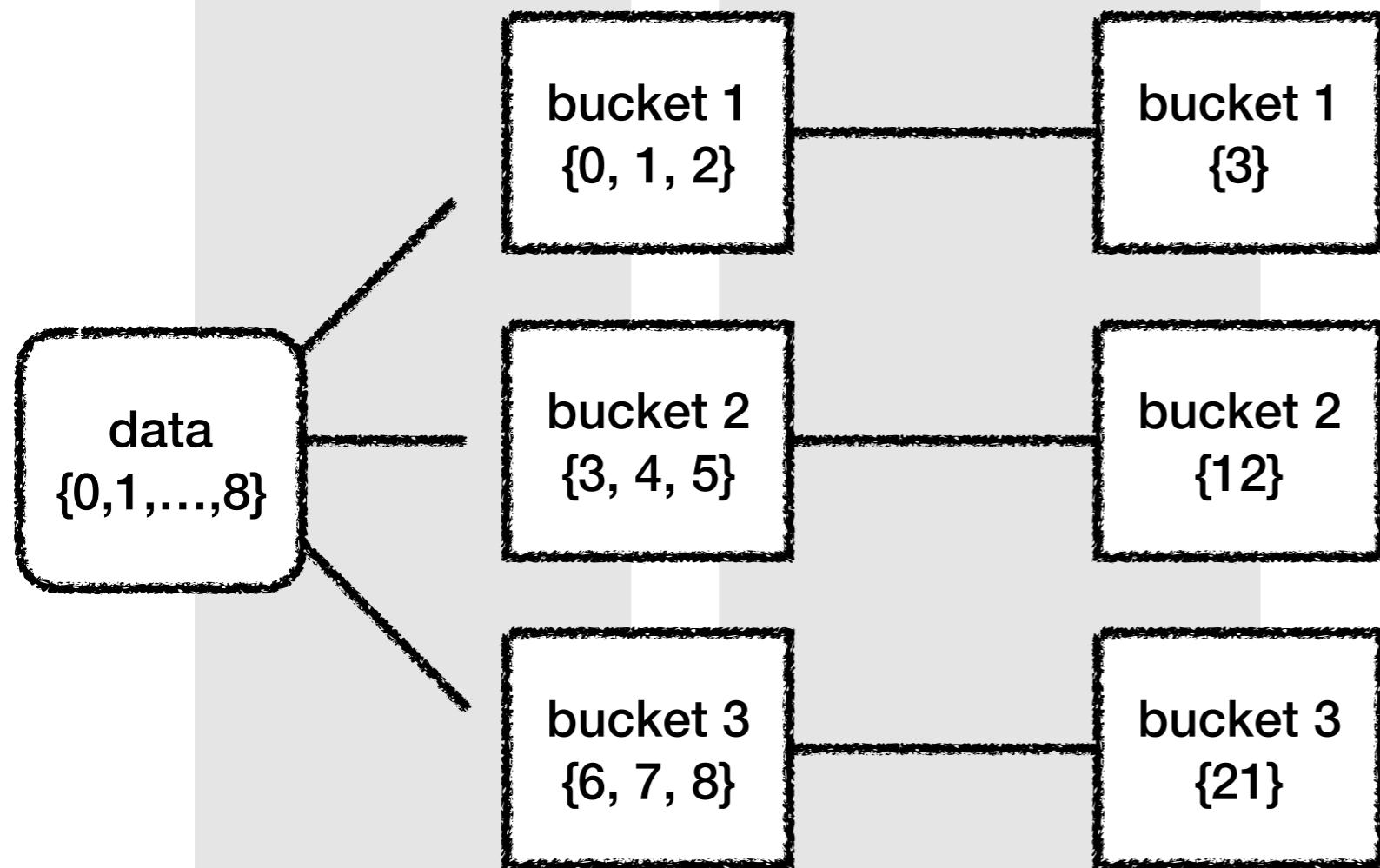


Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

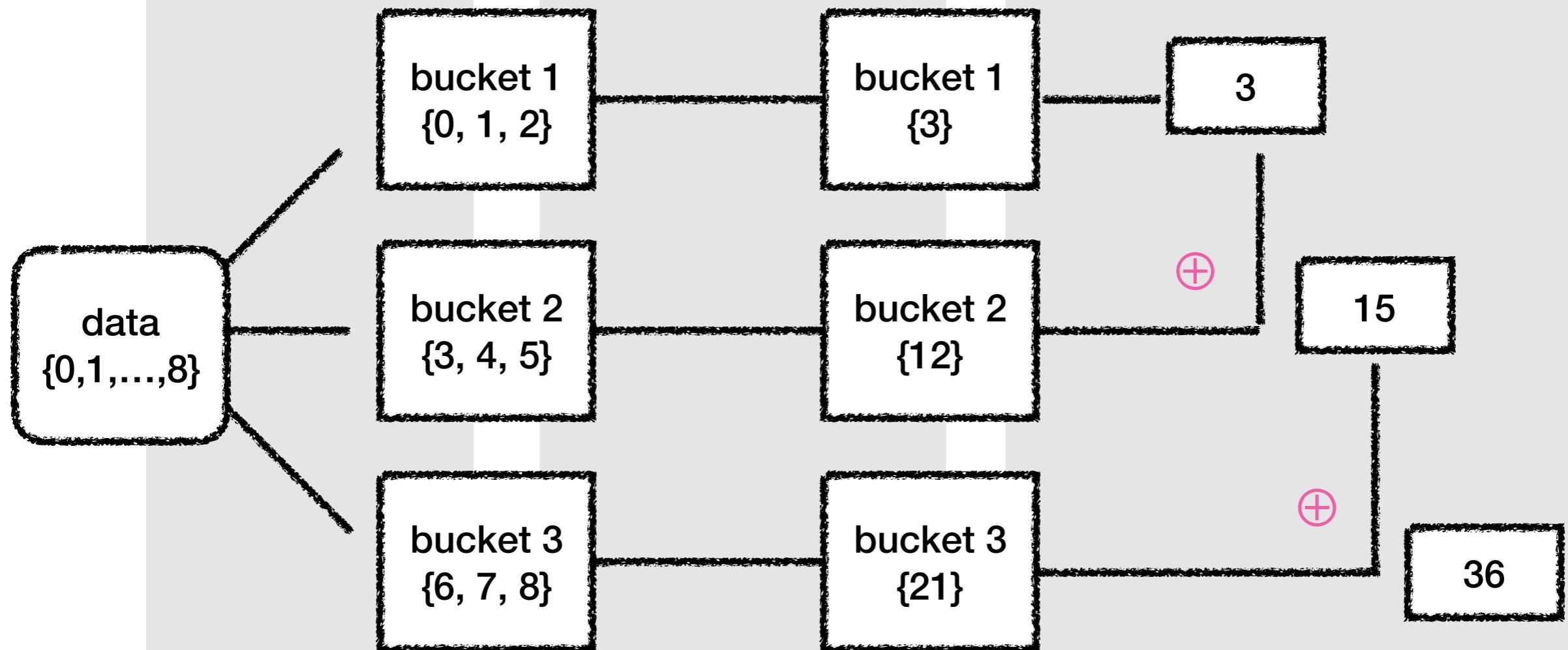
Metric Aggregation (Sum)



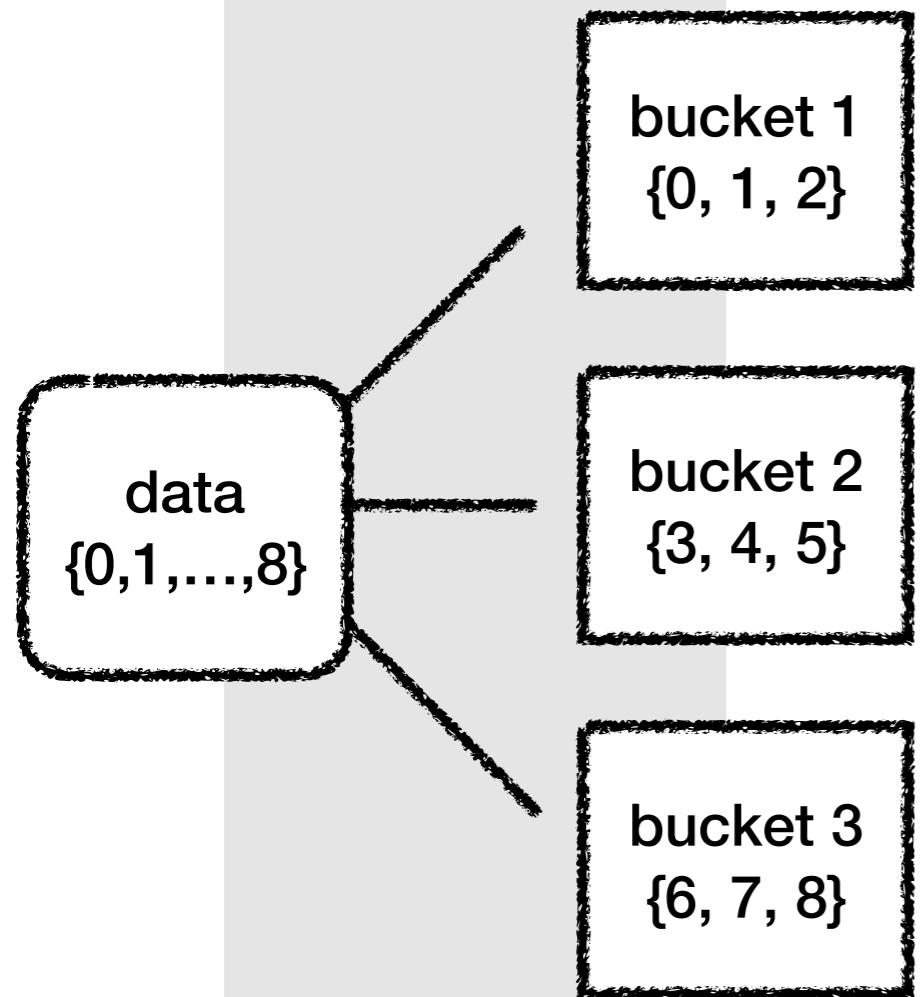
Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Parent Pipeline Aggregation (Cumulative Sum)

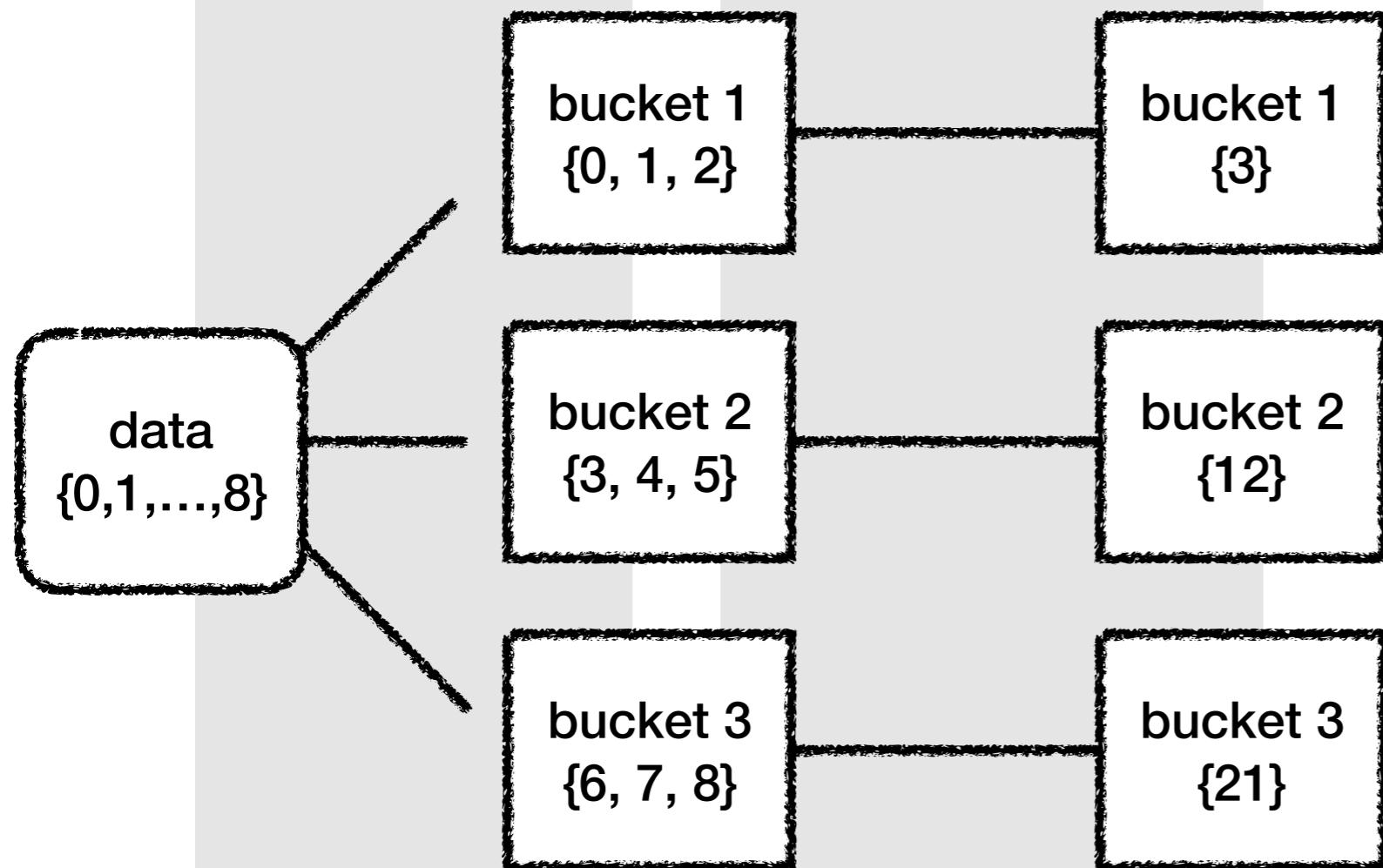


Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

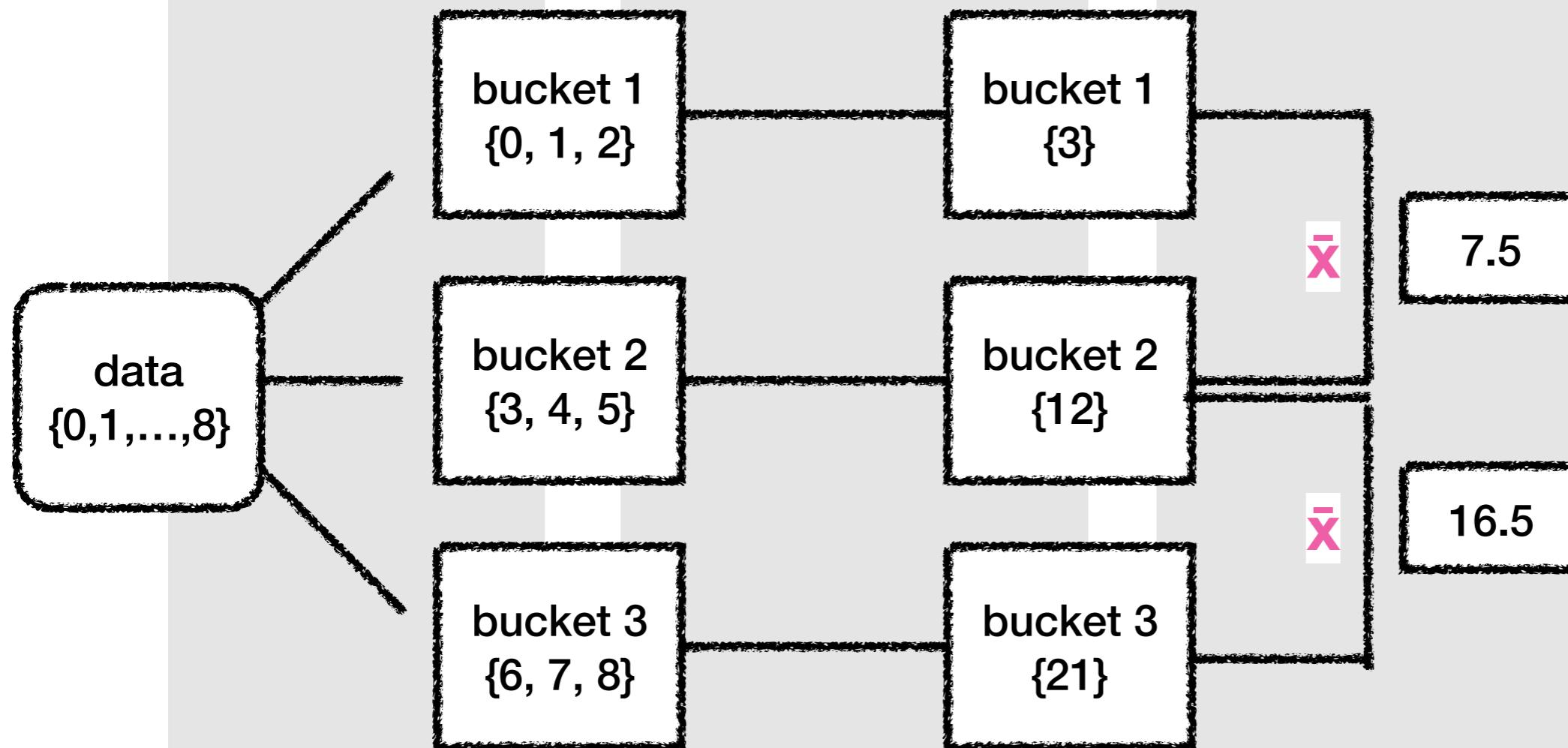
Metric Aggregation (Sum)



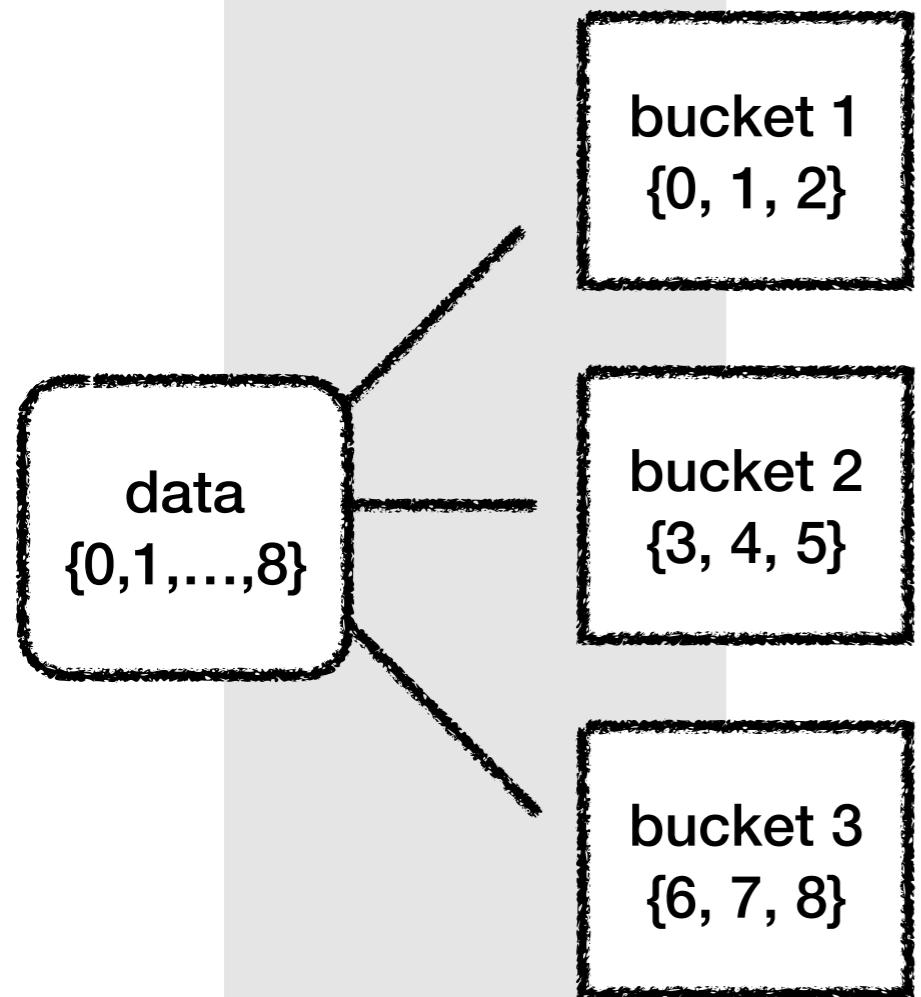
Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Parent Pipeline Aggregation (Moving Average, window=2)

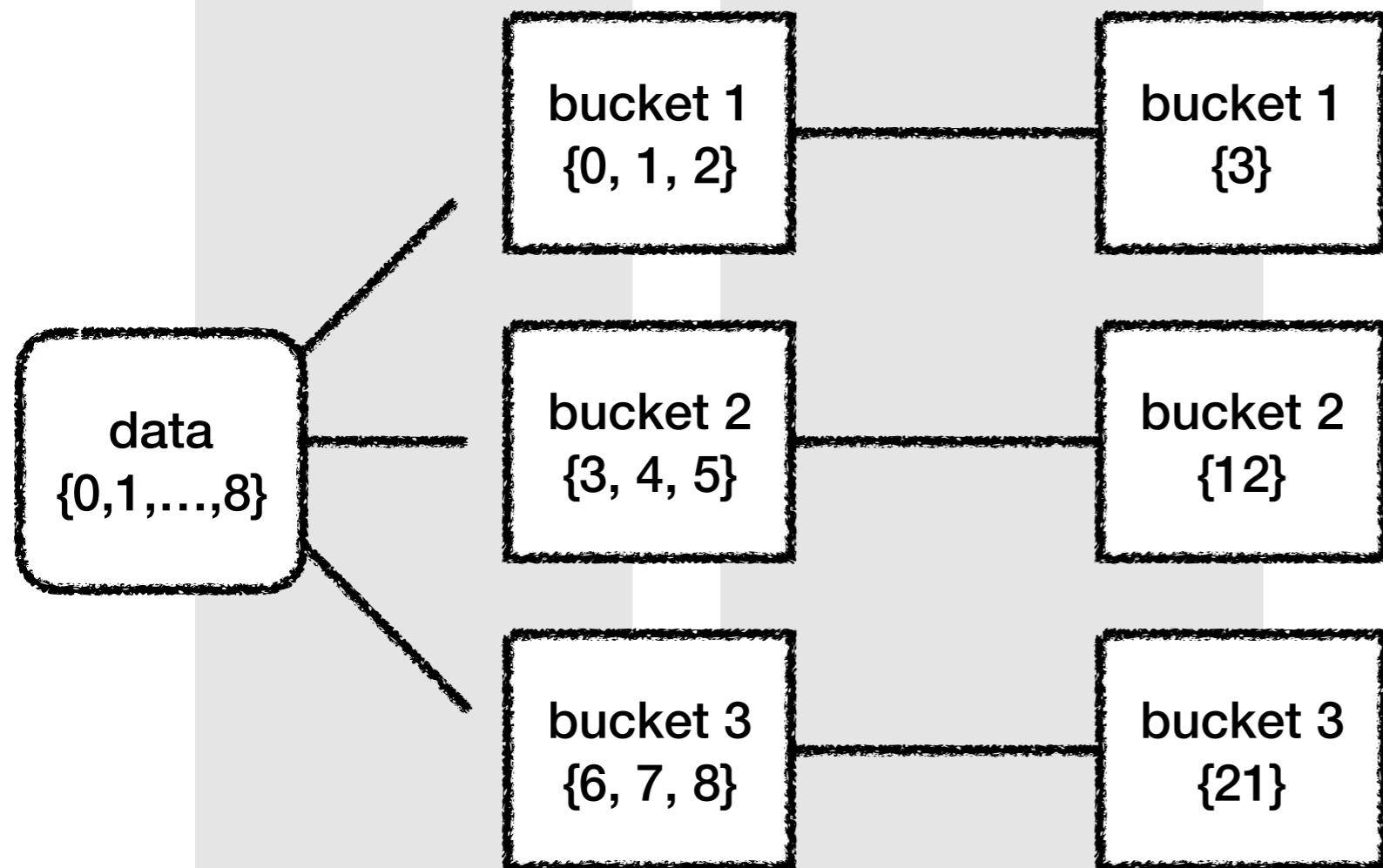


Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

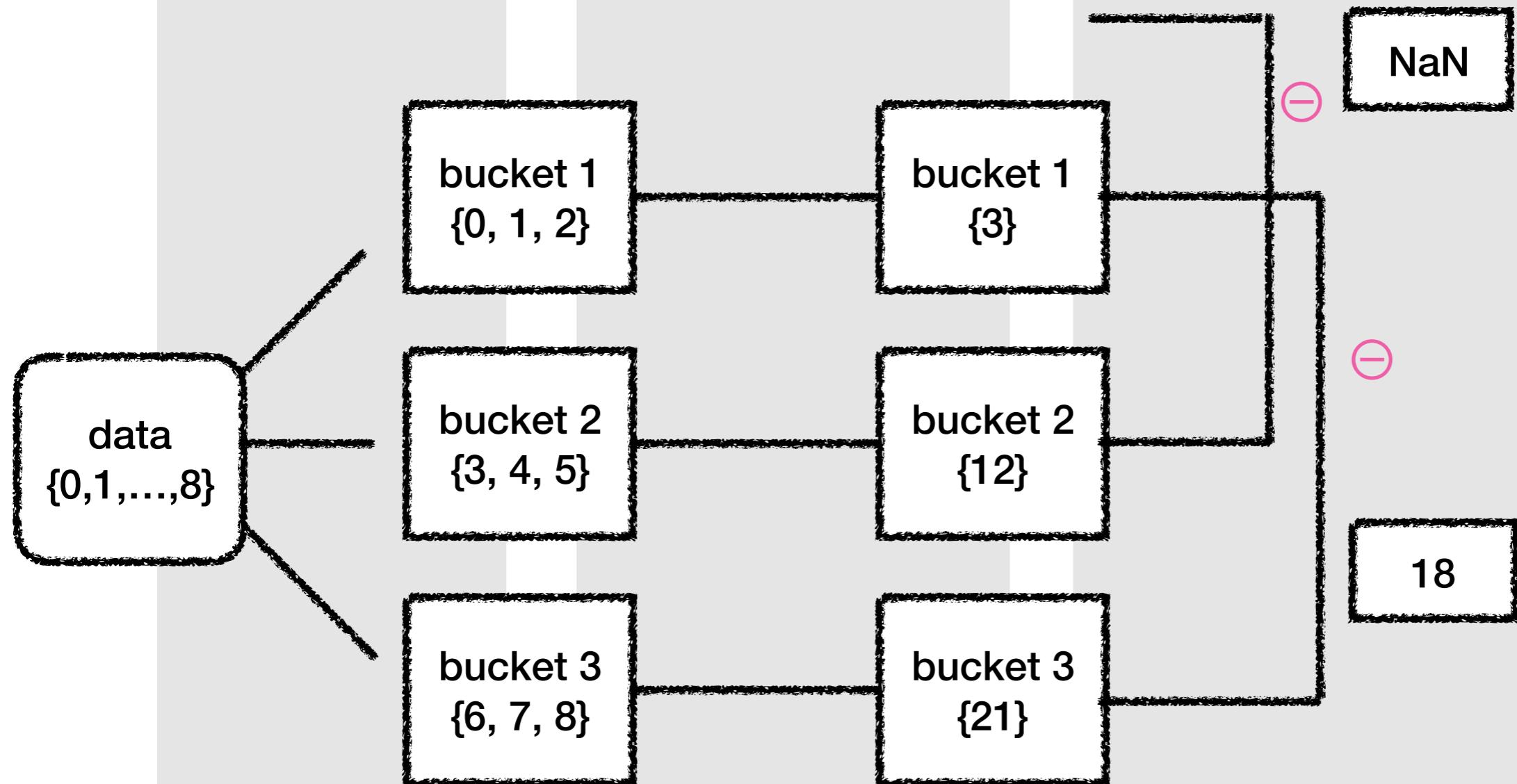
Metric Aggregation (Sum)



Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Parent Pipeline Aggregation (Serial Diff, lag=2)

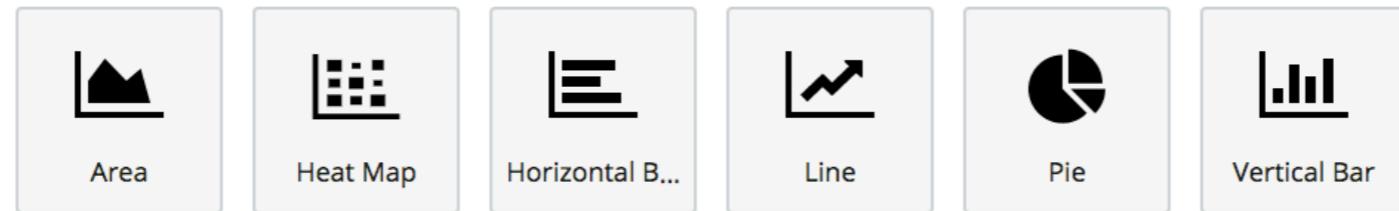


실제 Kibana에서 **Parent** Pipeline Aggregation을 어떻게 사용하는지 보자

Select visualization type

Search visualization types...

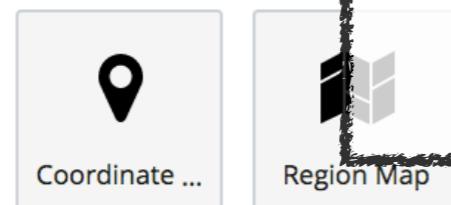
Basic Charts



Data

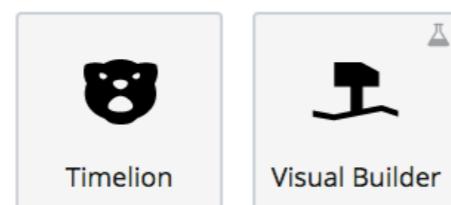


Maps

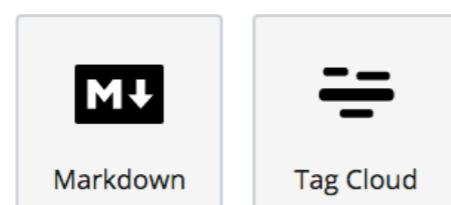


Visualize - Data Table

Time Series



Other



Derivative

shopping

Data Options

metrics

② Metric **Sum of 상품가격**

Metric

Aggregation

③ Derivative

Metric

④ metric: Sum of 상품가격

Custom Label

① Split Rows 주문시간 per day

Add metrics

Advanced

Add sub-buckets

주문시간 per day	Sum of 상품가격	Derivative of Sum of 상품가격
01월01일	423,000	-
01월02일	477,000	54,000
01월03일	394,000	-83,000
01월04일	465,000	71,000
01월05일	352,000	-113,000
01월06일	361,000	9,000
01월07일	418,000	57,000
01월08일	580,000	162,000
01월09일	382,000	-198,000
01월10일	448,000	66,000

- shopping index 중에서 year to date 기간 동안의
- “주문시간” field를 기준으로, 일 별 (daily)
- “상품가격” field의 합과
- 전날 대비 (=이전 bucket) “상품가격 field의 합”의 증감

1. Date Histogram을 이용해서 bucket을 ①에서 생성
2. “상품가격” Field의 합을 구하는 metric aggregation을 ②에서 생성
3. 사용할 Parent Pipeline Aggregation을 ③에서 선택
4. ③을 직접적으로 적용할 Metric Aggregation을 ④에서 선택
5. ⑤를 눌러서 시작화

Cumulative Sum

The screenshot shows the Elasticsearch Query DSL interface with the following configuration:

- shopping** (⑤)
- Data Options**
- metrics**
- Metric**: **Sum of 상품가격** (②)
- Metric**
- Aggregation**
- Cumulative Sum** (③)
- Metric**: **metric: Sum of 상품가격** (④)
- Custom Label**
- buckets**
- Split Rows** (①)
- Add metrics**
- Advanced**
- Add sub-buckets**

주문시간 per day	Sum of 상품가격	Cumulative Sum of Sum of 상품가격
01월01일	423,000	423,000
01월02일	477,000	900,000
01월03일	394,000	1,294,000
01월04일	465,000	1,759,000
01월05일	352,000	2,111,000
01월06일	361,000	2,472,000
01월07일	418,000	2,890,000
01월08일	580,000	3,470,000
01월09일	382,000	3,852,000
01월10일	448,000	4,300,000

- shopping index 중에서 year to date 기간 동안의
- “주문시간” field를 기준으로 일 별 (daily)
- “상품가격” field의 합과
- “상품가격” field의 누적합 (=이전 모든 bucket)

1. Date Histogram을 이용해서 bucket을 ①에서 생성
2. “상품가격” Field의 합을 구하는 metric aggregation을 ②에서 생성
3. 사용할 Parent Pipeline Aggregation을 ③에서 선택
4. ③을 직접적으로 적용할 Metric Aggregation을 ④에서 선택
5. ⑤를 눌러서 시작화

Moving Average

The screenshot shows the Elasticsearch Pipeline configuration interface for a pipeline named "shopping".

Metrics:

- ② Metric: Sum of 상품가격 (highlighted with a pink border)
- ③ Metric: Moving Avg (highlighted with a pink border)
- ④ metric: Sum of 상품가격 (highlighted with a pink border)

Custom Label:

buckets:

- ① Split Rows (highlighted with a pink border)

Advanced Options:

- Add metrics (button)
- Add sub-buckets (button)

Resulting Data Table:

주문시간 per day	Sum of 상품가격	Moving Avg of Sum of 상품가격
01월01일	423,000	-
01월02일	477,000	423,000
01월03일	394,000	450,000
01월04일	465,000	431,333.3
01월05일	352,000	439,750
01월06일	361,000	422,200
01월07일	418,000	409,800
01월08일	580,000	398,000
01월09일	382,000	435,200
01월10일	448,000	418,600

Notes:

- shopping index 중에서 year to date 기간 동안의
- “주문시간” field를 기준으로 일 별 (daily)
- “상품가격” field의 합과
- “상품가격” field의 이동평균

* default window size : 5

1. Date Histogram을 이용해서 bucket을 ①에서 생성
2. “상품가격” Field의 합을 구하는 metric aggregation을 ②에서 생성
3. 사용할 Parent Pipeline Aggregation을 ③에서 선택
4. ③을 직접적으로 적용할 Metric Aggregation을 ④에서 선택
5. ⑤를 눌러서 시작화

Serial Diff

The screenshot shows the Elasticsearch Serial Diff interface with the following configuration steps highlighted:

- ① Split Rows**: A button to split rows.
- ② Metric**: A dropdown menu for metrics, with "Sum of 상품가격" selected.
- ③ Serial Diff**: A dropdown menu for aggregation type, with "Serial Diff" selected.
- ④ metric: Sum of 상품가격**: A dropdown menu for the metric used in the Serial Diff aggregation.
- ⑤**: A play button icon to start the process.

An arrow points from the interface to the resulting table on the right:

주문시간 per day	Sum of 상품가격	Serial Diff of Sum of 상품가격
01월01일	423,000	-
01월02일	477,000	54,000
01월03일	394,000	-83,000
01월04일	465,000	71,000
01월05일	352,000	-113,000
01월06일	361,000	9,000
01월07일	418,000	57,000
01월08일	580,000	162,000
01월09일	382,000	-198,000
01월10일	448,000	66,000

Advanced and **Add metrics** buttons are also visible in the interface.

- shopping index 중에서 year to date 기간 동안의
 - “주문시간” field를 기준으로 일 별 (daily)
 - “상품가격” field의 합과
 - “상품가격 field의 합”의 증감
- * default lag size : 1

1. Date Histogram을 이용해서 bucket을 ①에서 생성
2. “상품가격” Field의 합을 구하는 metric aggregation을 ②에서 생성
3. 사용할 Parent Pipeline Aggregation을 ③에서 선택
4. ③을 직접적으로 적용할 Metric Aggregation을 ④에서 선택
5. ⑤를 눌러서 시작화

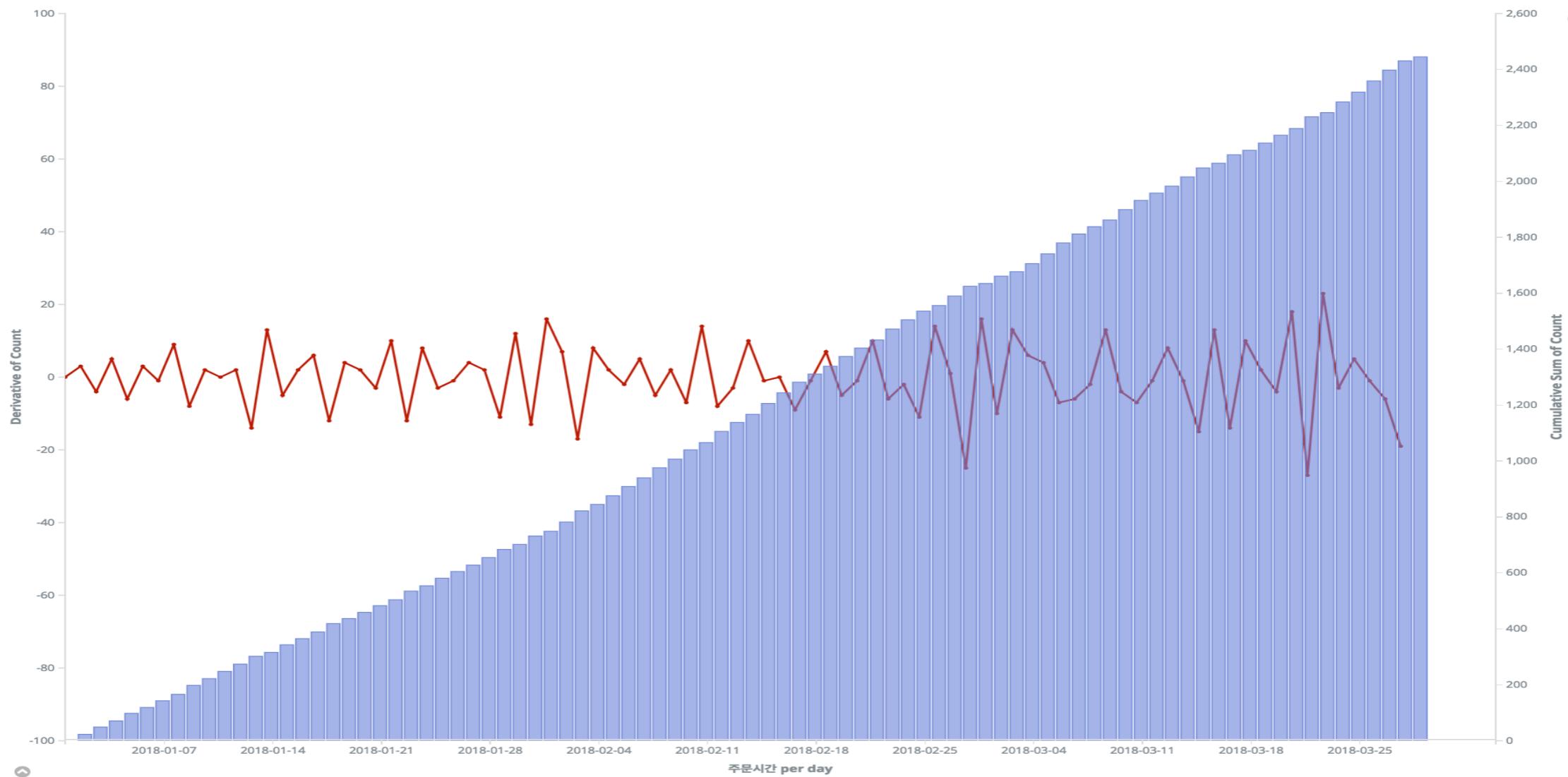
Parent Pipeline - 예제 1

“주문시간” Field를 기준으로 일 별로 Documents의 개수와 그 이동평균을 시각화해보자



Parent Pipeline - 예제 2 ✎

“주문시간” Field를 기준으로 일 별로 1) documents 개수의 증감 및 2) 누적 documents 개수를 시각화



Aggregation - Sibling Pipeline

종류	상세
Average Bucket	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Average Agg 적용
Max Bucet	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Max Agg 적용
Min Bucket	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Min Agg 적용
Sum Bucket	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Sum Agg 적용

다음과 같은 데이터가 있다고 하자

{"data" : 0}

{"data" : 1}

{"data" : 2}

{"data" : 3}

{"data" : 4}

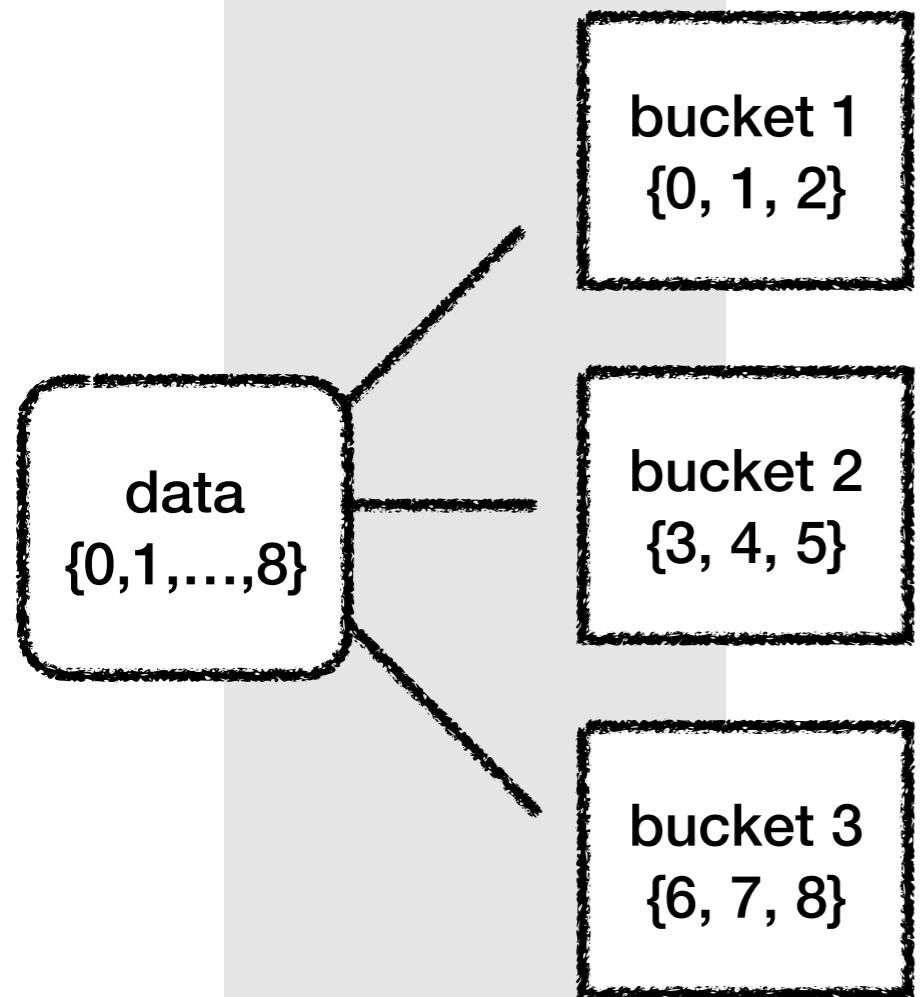
{"data" : 5}

{"data" : 6}

{"data" : 7}

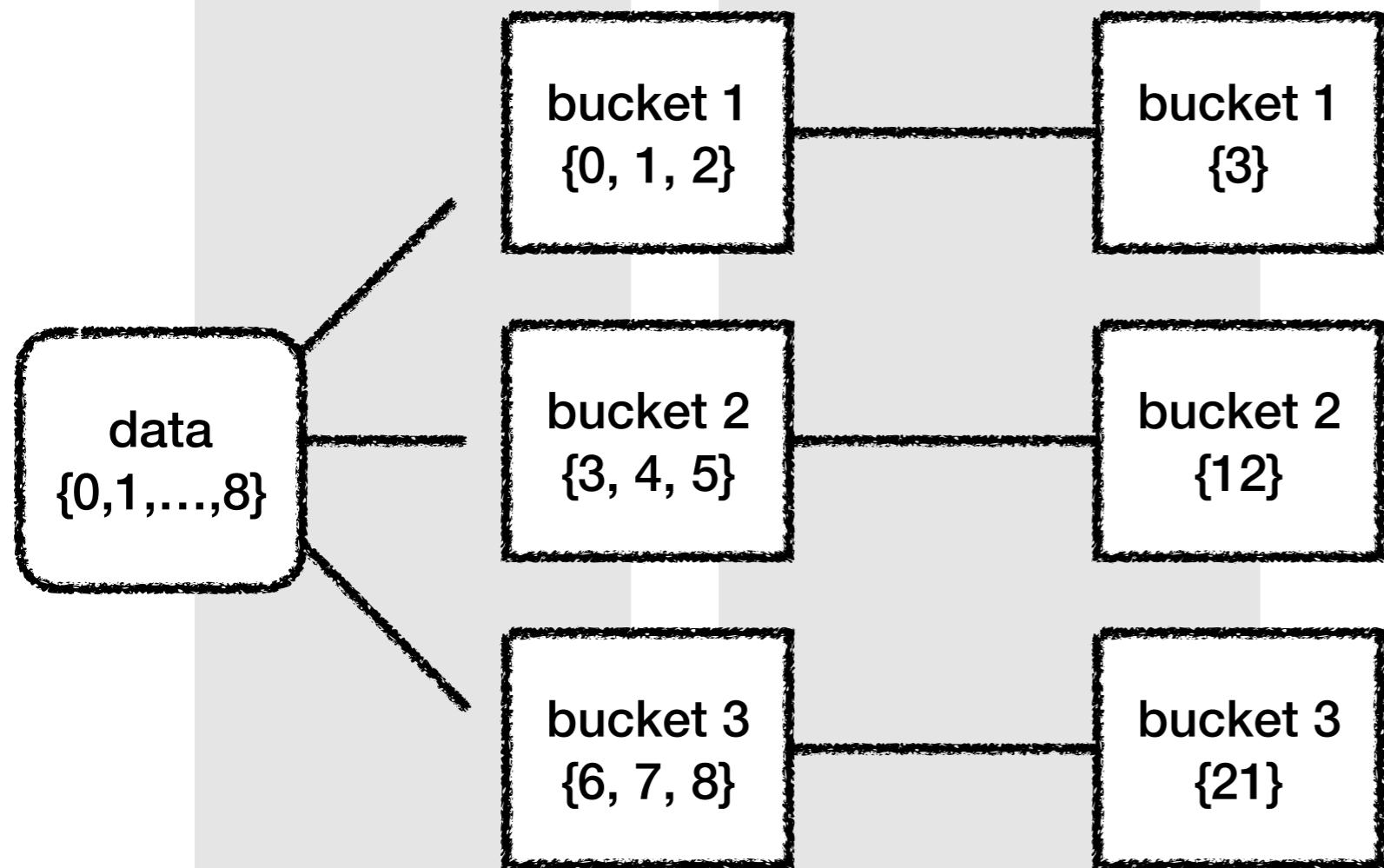
{"data" : 8}

Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

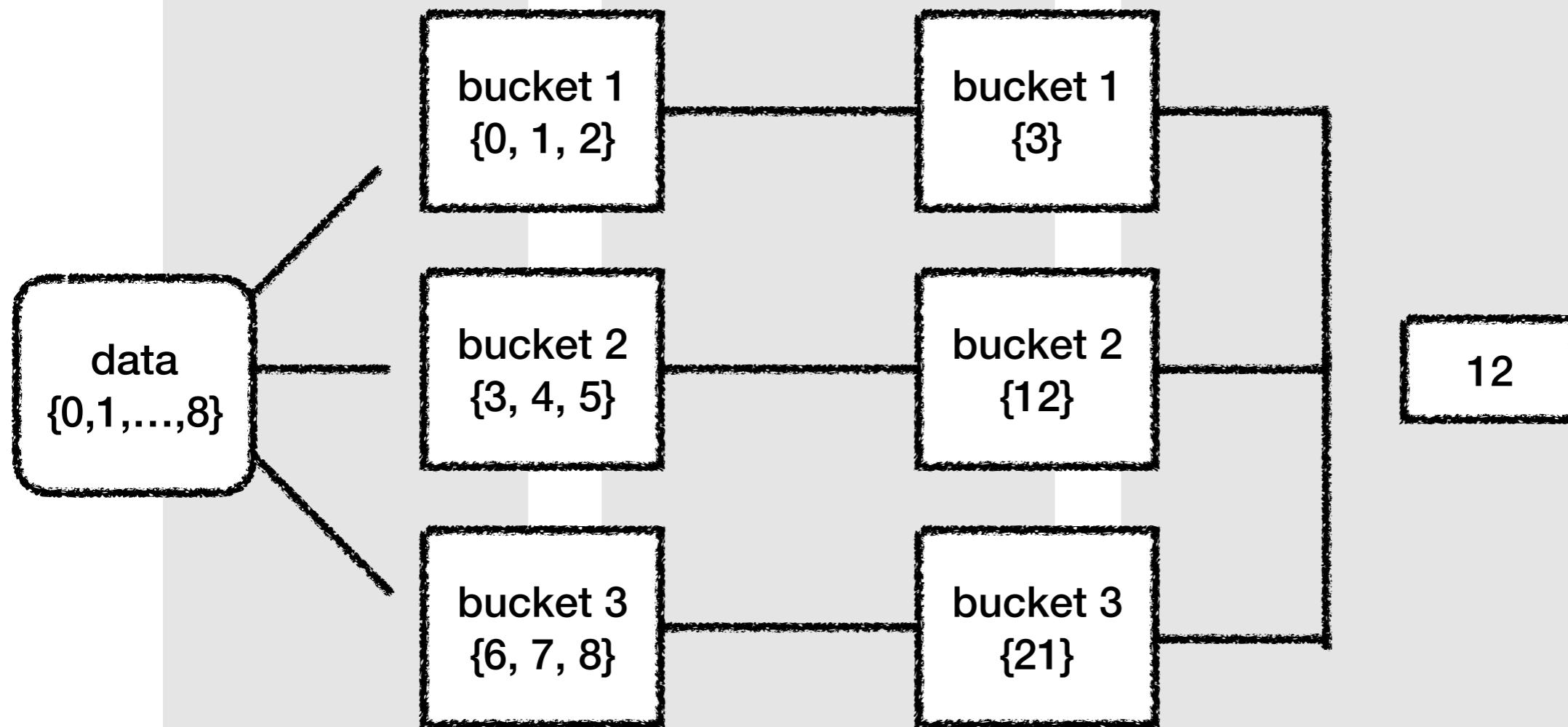
Metric Aggregation (Sum)



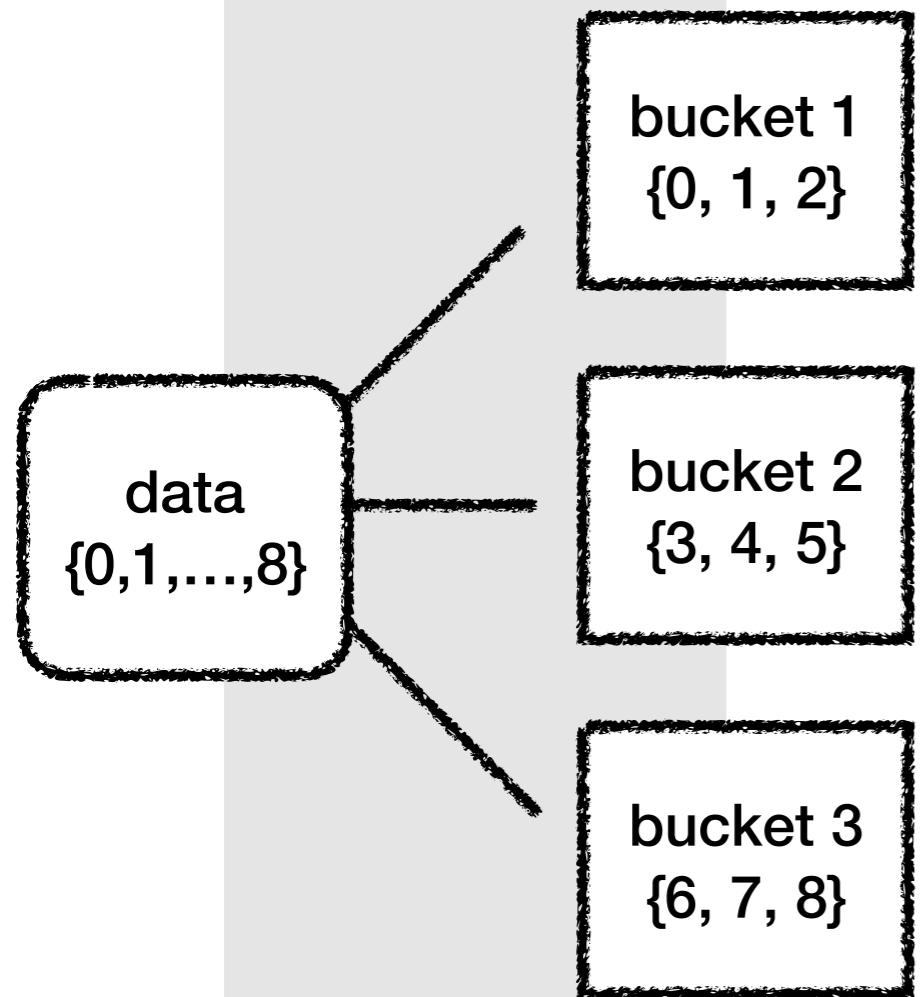
Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Sibling Pipeline Aggregation (Average)

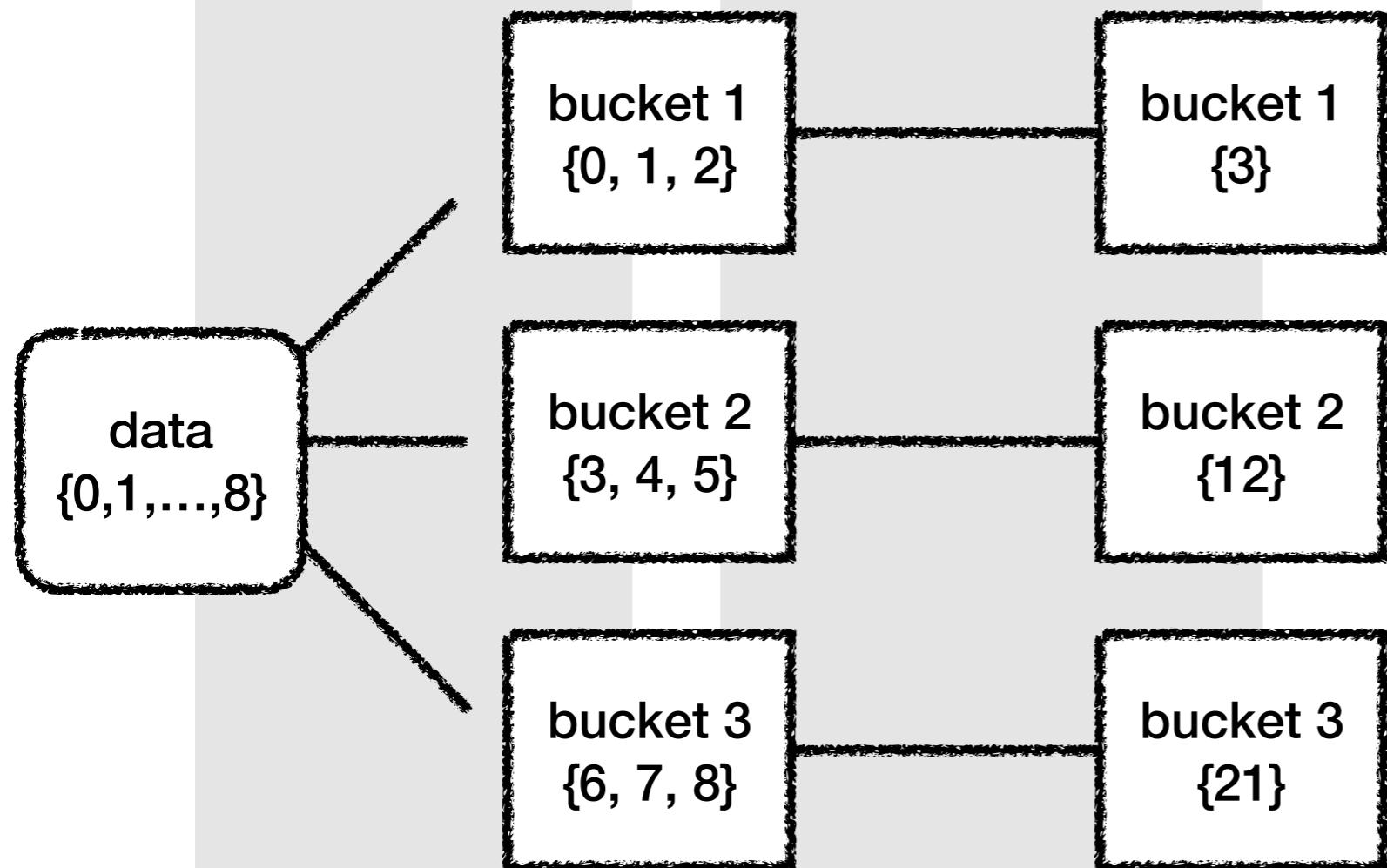


Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

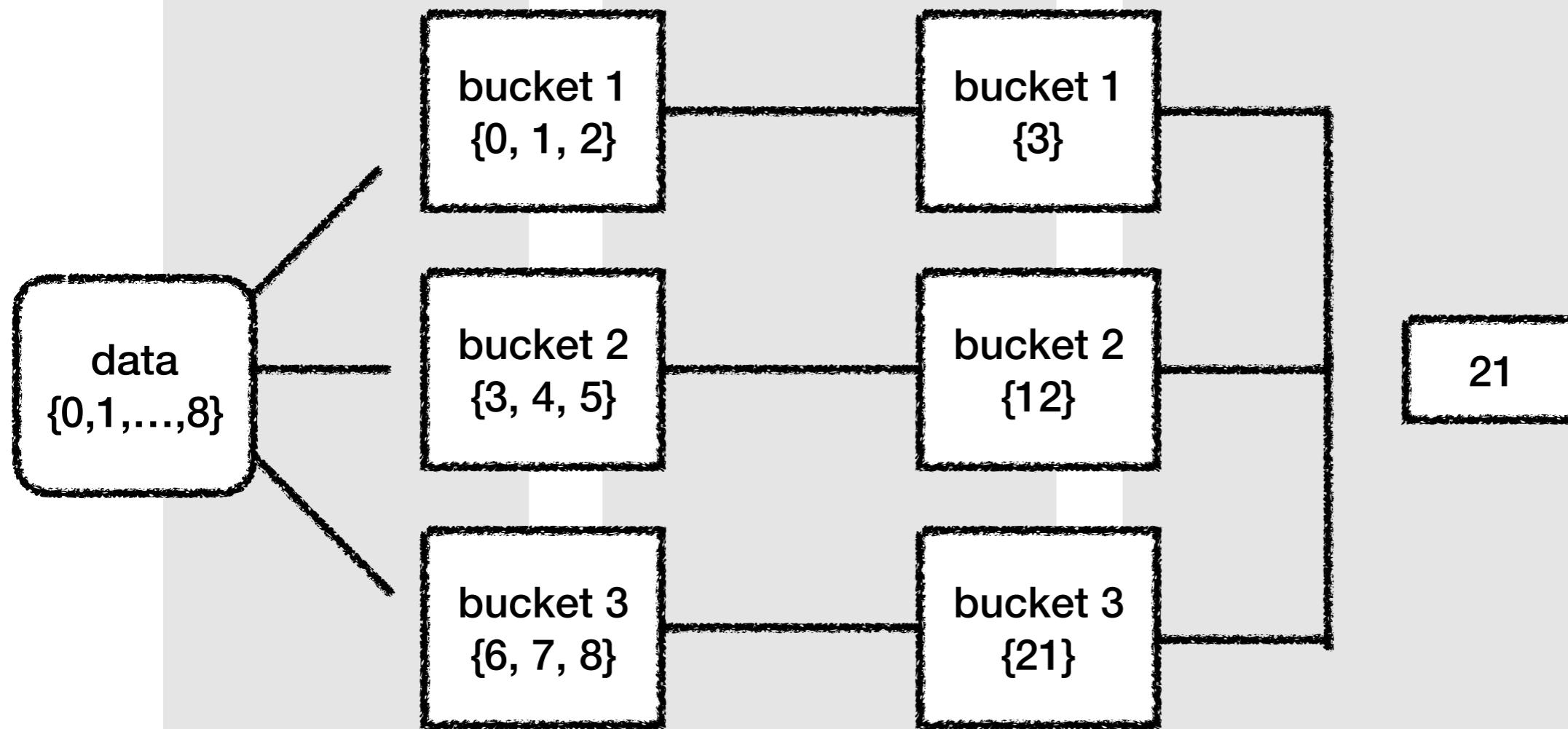
Metric Aggregation (Sum)



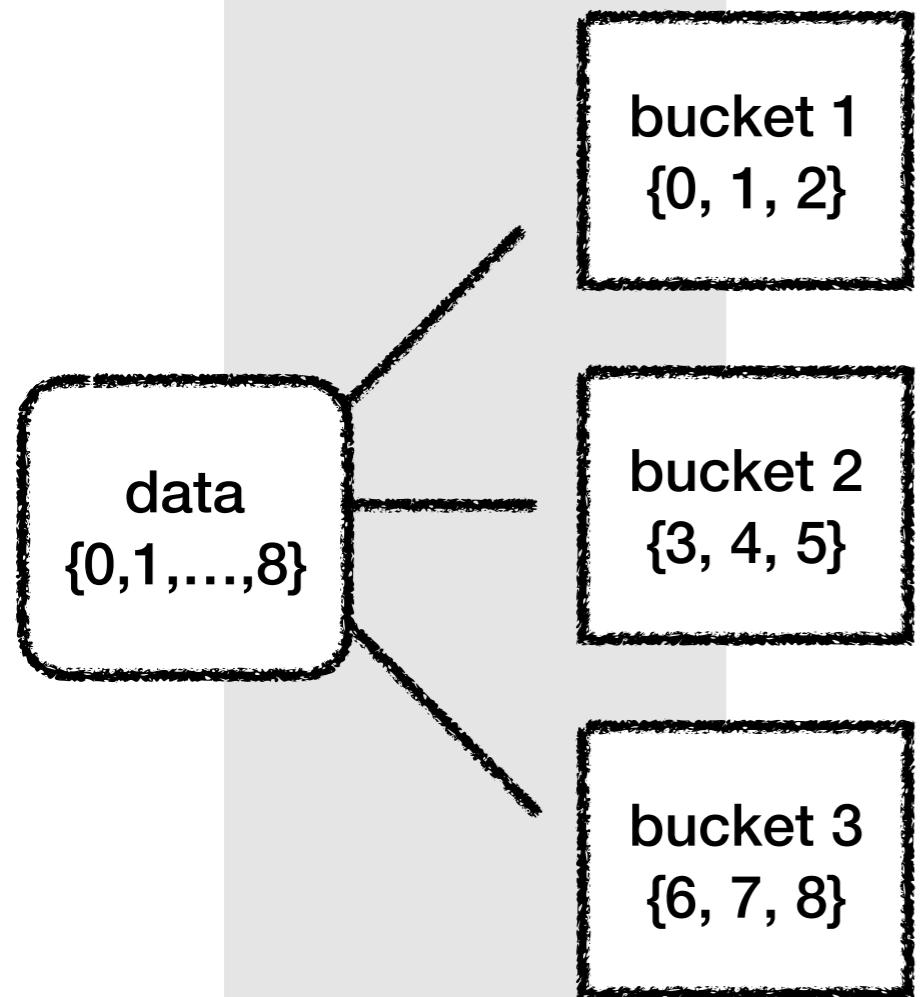
Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Sibling Pipeline Aggregation (Max)

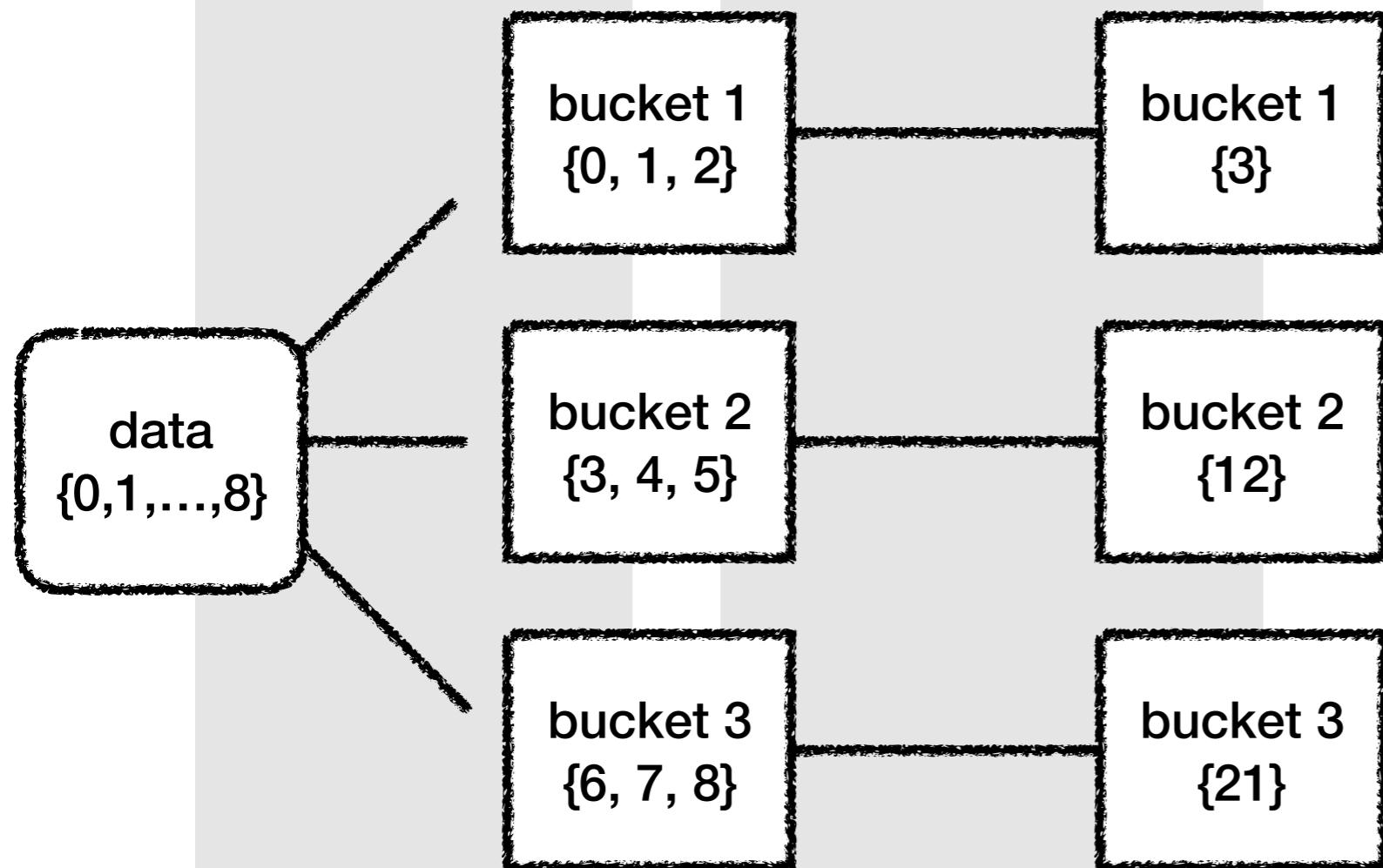


Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

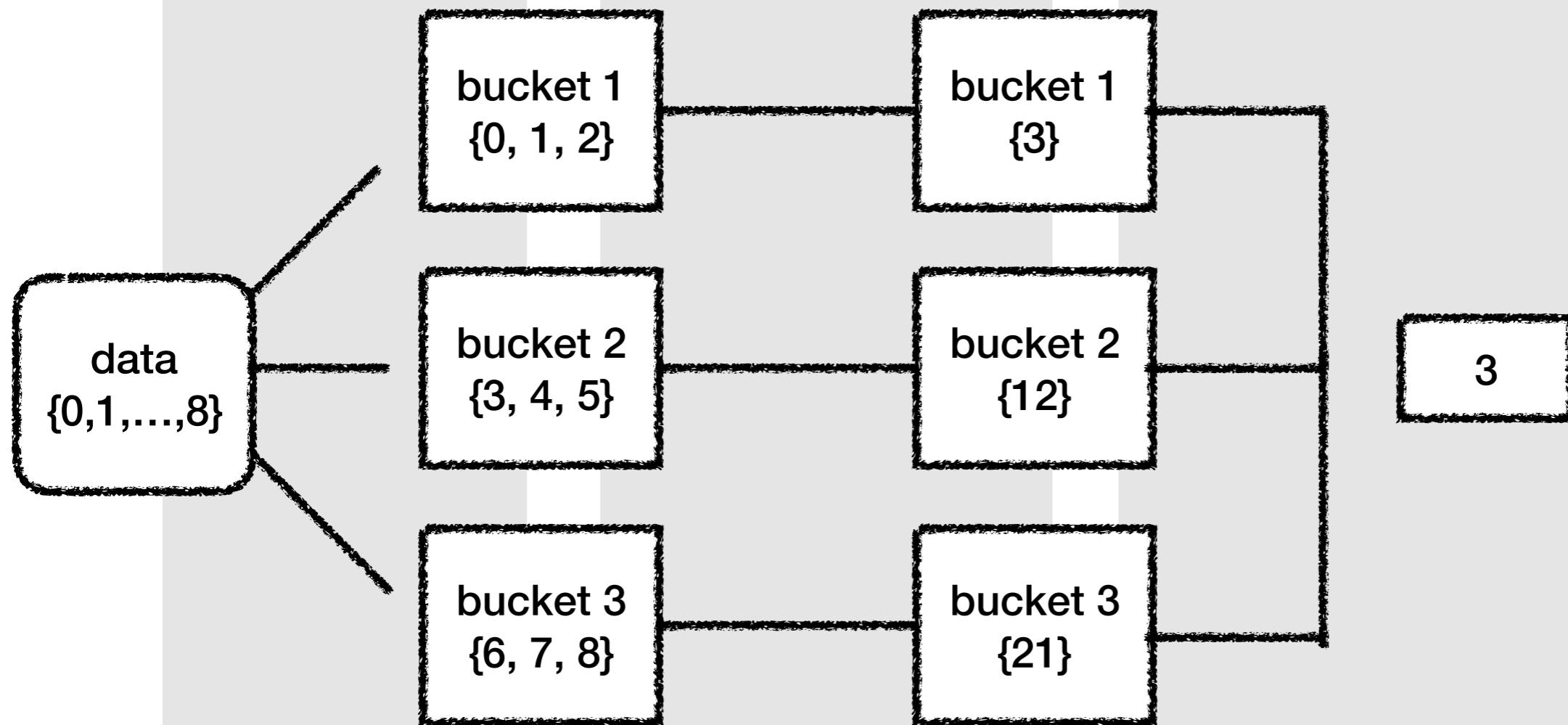
Metric Aggregation (Sum)



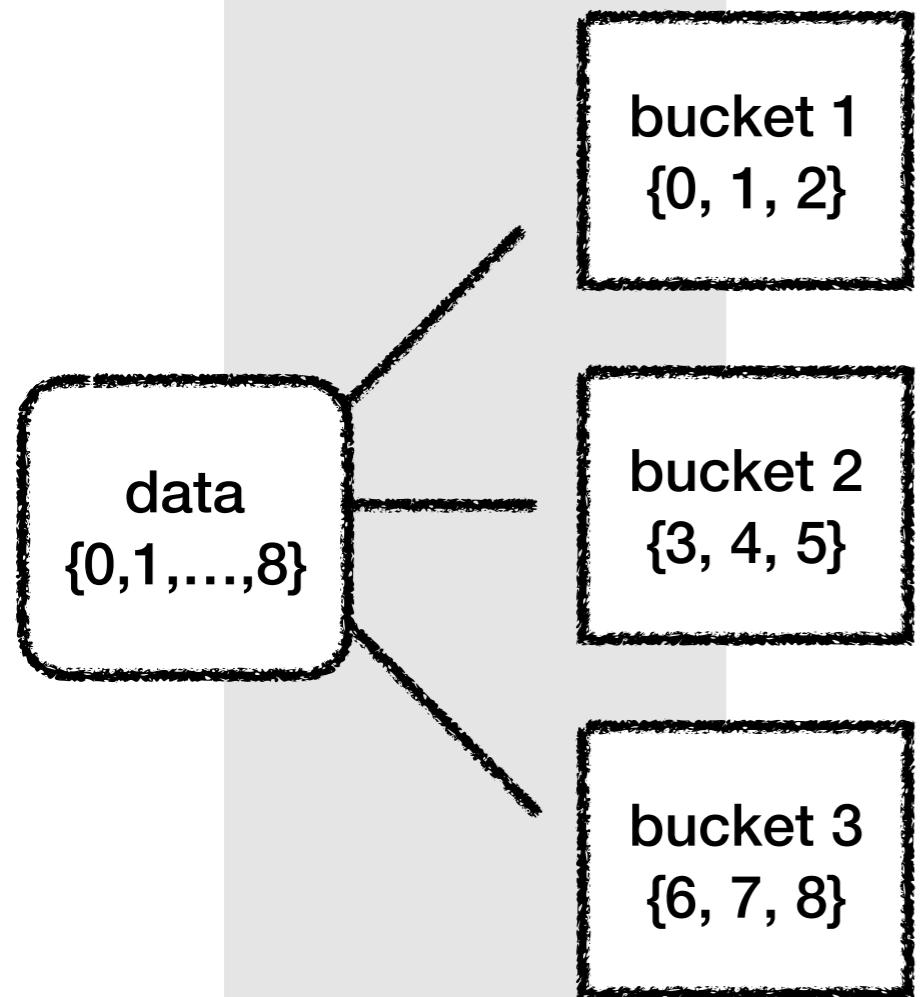
Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Sibling Pipeline Aggregation (Min)

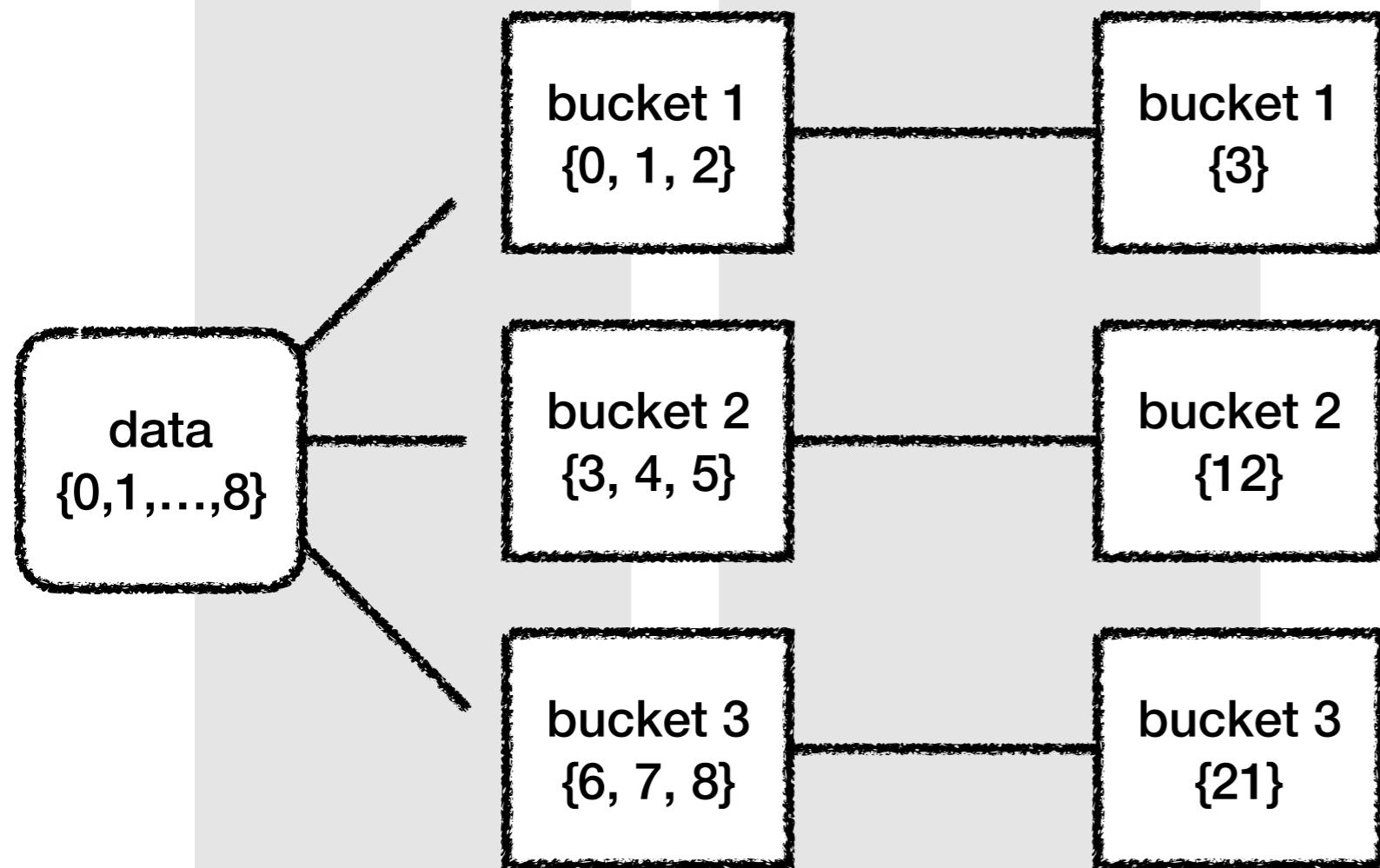


Bucket Aggregation (Histogram)



Bucket Aggregation (Histogram)

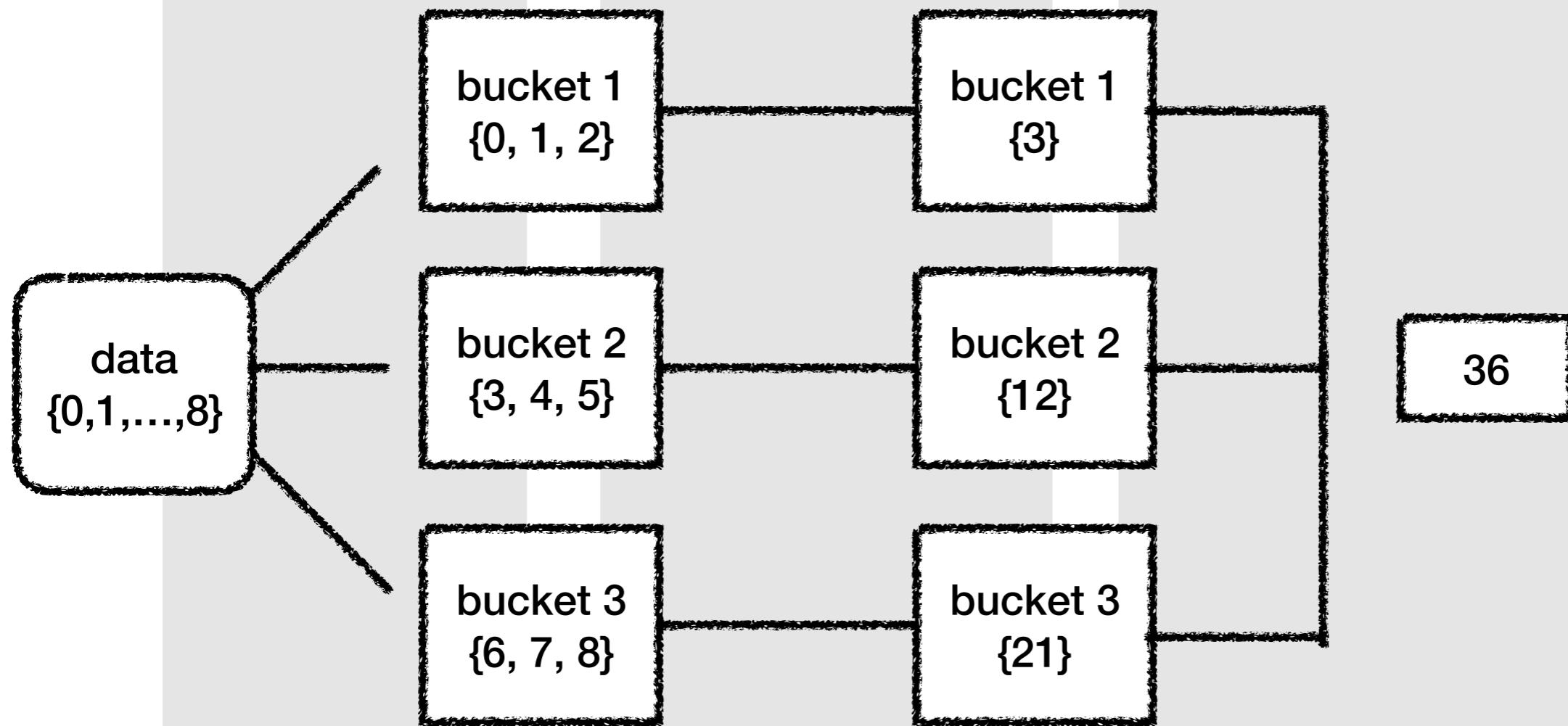
Metric Aggregation (Sum)



Bucket Aggregation (Histogram)

Metric Aggregation (Sum)

Sibling Pipeline Aggregation (Sum)

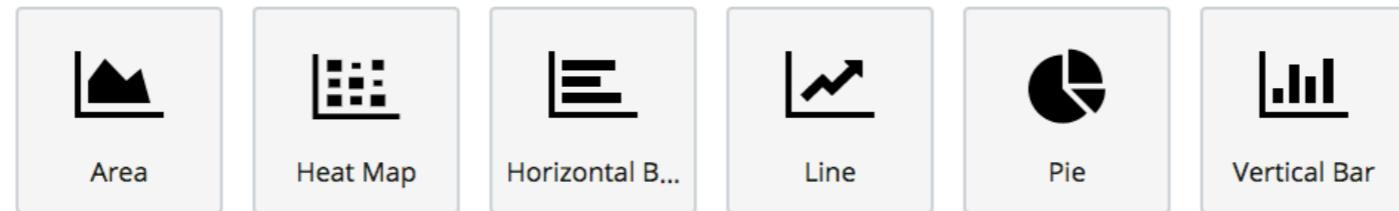


실제 Kibana에서 **Sibling** Pipeline Aggregation을 어떻게 사용하는지 보자

Select visualization type

Search visualization types...

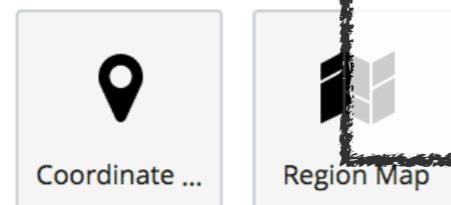
Basic Charts



Data

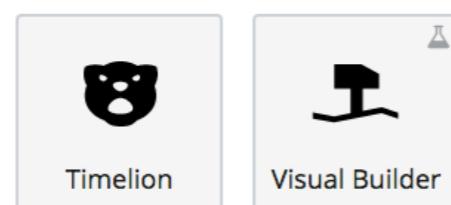


Maps

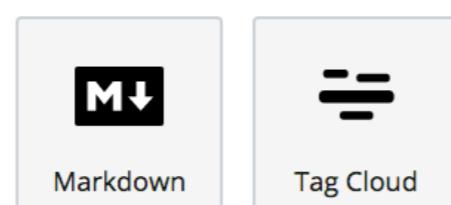


Visualize - Data Table

Time Series



Other



Average Bucket

The screenshot shows the Elasticsearch Query Builder interface for the 'shopping' index. The configuration steps are highlighted with pink circles:

- ① Buckets: A bucket named '판매자평점' is selected.
- ② Aggregation: 'Average Bucket' is selected from the dropdown.
- ③ Aggregation: The 'Field' is set to '구매사이트' and 'Order By' is set to 'Custom Metric'.
- ④ Aggregation: The 'Field' is set to '상품가격'.
- ⑤ Start button: The play button is highlighted.

An arrow points from the configuration to the results table on the right:

판매자평점	Overall Average of Sum of 상품가격
1	3,971,000
2	1,509,000
3	3,423,000
4	2,662,500
5	1,233,500

- shopping index 중에서 year to date 기간 동안의
- “판매자평점” Field Value 별로
- documents가 가장 많은 2개의 “구매사이트”를 선별하여
- “구매사이트” 별로 “상품가격” Field의 합을 구하고
- 그에 대한 평균을 취한 값

1. Histogram Aggregation을 이용해서 bucket을 ①에서 생성
2. 사용할 Sibling Pipeline Aggregation을 ②에서 선택
3. Sibling Pipeline Aggregation 내부에서 사용할 Bucket Aggregation을 ③에서 설정
4. Sibling Pipeline Aggregation 내부에서 사용할 Metric Aggregation을 ④에서 설정
5. ⑤를 눌러서 시작화

Max Bucket

The screenshot shows the Elasticsearch Query DSL interface with the following configuration:

- Bucket Aggregation:** Set to "Max Bucket" (②).
- Bucket Pipeline:** Set to "Date Histogram" (③).
 - Field:** "주문시간" (Order Time).
 - Interval:** "Daily".
- Metric Aggregation:** Set to "Sum" (④).
 - Field:** "상품가격" (Product Price).

At the bottom, there is a "buckets" section with a "Split Rows" button (①) and a "결제카드: Descending" button.

An arrow points from the interface to the results table on the right:

결제카드: Descending	Overall Max of Sum of 상품가격
하나	162,000
우리	291,000
신한	124,000

List of steps (⑤):

- shopping index 중에서 year to date 기간 동안의
- “결제카드” Field 이름의 역순으로 선별한 3개의 “결제카드” 별로
- “상품가격”의 합을 일 별로 (daily) 계산 한 후
- 가장 컸던 값을 표시

1. Term Aggregation을 이용해서 bucket을 ①에서 생성
2. 사용할 Sibling Pipeline Aggregation을 ②에서 선택
3. Sibling Pipeline Aggregation 내부에서 사용할 Bucket Aggregation을 ③에서 설정
4. Sibling Pipeline Aggregation 내부에서 사용할 Metric Aggregation을 ④에서 설정
5. ⑤를 눌러서 시작화

Min Bucket

The screenshot shows the Elasticsearch Query DSL interface with several steps highlighted by pink circles:

- ① Filters: A box containing three filters: 서울특별시, 쿠팡, and 국민.
- ② Aggregation: A dropdown menu set to "Min Bucket".
- ③ Bucket: A section for Date Histogram settings, including Field (주문시간), Interval (Daily), and Aggregation (Sum).
- ④ Metric: A section for Metric settings, including Field (상품가격) and Aggregation (Sum).
- ⑤ Play button: A play button at the top right.

An arrow points from the interface to the results table:

filters	Overall Min of Sum of 상품가격
국민	36,000
서울특별시	44,000
쿠팡	11,000

- shopping index 중에서 year to date 기간 동안의
- “서울특별시” // “쿠팡” // “국민”에 해당하는 Bucket 별로
- “상품가격”의 합을 일 별로 (daily) 계산 한 후
- 가장 작았던 값을 표시

1. Filters Aggregation을 이용해서 bucket을 ①에서 생성
2. 사용할 Sibling Pipeline Aggregation을 ②에서 선택
3. Sibling Pipeline Aggregation 내부에서 사용할 Bucket Aggregation을 ③에서 설정
4. Sibling Pipeline Aggregation 내부에서 사용할 Metric Aggregation을 ④에서 설정
5. ⑤를 눌러서 시작화

Sum Bucket

The screenshot shows the Elasticsearch Query Builder interface with the following steps highlighted:

- ① In the "Bucket" section, "Sum Bucket" is selected under "Aggregation".
- ② In the "Bucket" section, the "Field" is set to "구매사이트" and the "Order By" is set to "Custom Metric".
- ③ In the "Metric" section, the "Aggregation" is set to "Count".
- ④ A play button icon at the top right is highlighted.

An arrow points from the interface to the results section, which displays:

Overall Sum of Count ◆

1,498

- shopping index 중에서 year to date 기간 동안의 documents가 가장 많은 2개의 “구매사이트”를 선별하여
- 각각의 documents 개수를 구하고
- 최종적으로 합한 결과

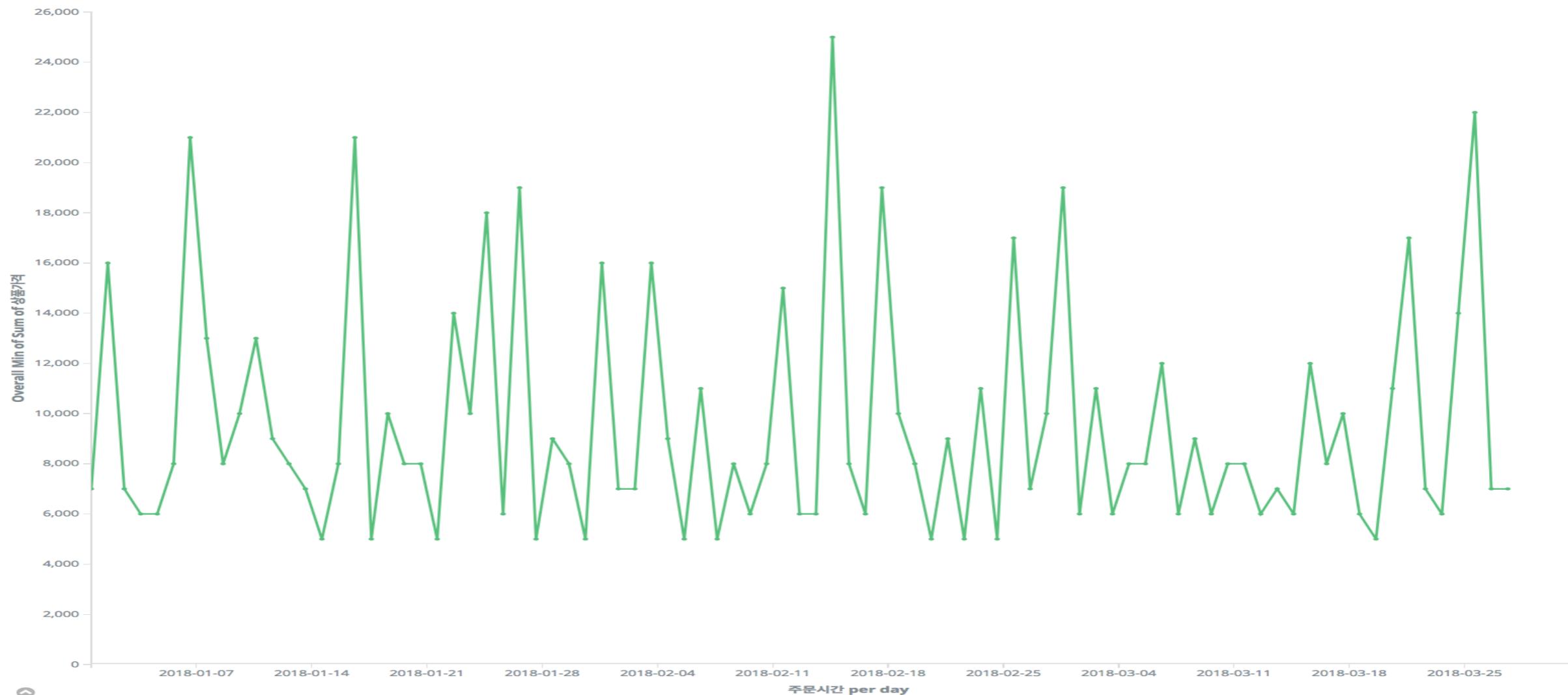
= documents 개수 기준 상위 2개 “구매사이트”의 documents 개수

1. 사용할 Sibling Pipeline Aggregation을 ①에서 선택
2. Sibling Pipeline Aggregation 내부에서 사용할 Bucket Aggregation을 ②에서 설정
3. Sibling Pipeline Aggregation 내부에서 사용할 Metric Aggregation을 ③에서 설정
4. ④를 눌러서 시작화

Parent Pipeline - 예제 1 ✎

X 축 : “주문시간” Field를 기준으로 일 별로(daily)

Y 축 : 가나다 순으로 상위 5개 “상품분류” Field 별 “상품가격” Field의 합을 구하고, 그 중에서 가장 작은 값을 표시하자



Parent Pipeline - 예제 2 ✎

X 축 : “주문시간” Field를 기준으로 일 별로

Y 축 :

- documents 개수와

- documents 개수 기준 상위 3개 “구매사이트” Field의 documents 개수를 합한 값을 표시하자



Aggregation 사용 시 헷갈리는 사항 몇 가지만 정리하고 넘어가자

Q. Bucket Aggregation은 한 번만 할 수 있나?

A. Bucket Aggregation 할 Field가 남아있을 때까지 여러 번 가능하다

bucket aggregation 1회

The screenshot shows the Kibana interface with a single bucket aggregation setup. The left sidebar shows the dataset 'shopping' selected. Under 'metrics', 'Metric' is chosen with 'Count' as the metric. Under 'buckets', 'Split Rows' is selected, and 'Date Histogram' is chosen as the type. The 'Field' dropdown is set to '주문시간' and the 'Interval' is set to 'Monthly'. The resulting table on the right has two columns: '주문시간 per month' and 'Count'. The data rows are: 01월01일 (Count: 749), 02월01일 (Count: 886), and 03월01일 (Count: 810). A black arrow points from the 'Count' column header to the value 749.

주문시간 per month	Count
01월01일	749
02월01일	886
03월01일	810

bucket aggregation 2회

The screenshot shows the Kibana interface with a nested bucket aggregation setup. The left sidebar shows the dataset 'shopping' selected. Under 'metrics', 'Metric' is chosen with 'Count' as the metric. Under 'buckets', 'Split Rows' is selected, and '주문시간 per month' is chosen as the field. Another 'Split Rows' step is added, and 'Sub Aggregation' is selected. 'Terms' is chosen as the type. The 'Field' dropdown is set to '고객성별'. The 'Order By' dropdown is set to 'metric: Count' with 'Descending' order and size 2. The resulting table on the right has three columns: '주문시간 per month', '고객성별: Descending', and 'Count'. The data rows are: 01월01일 (여성: 430), 01월01일 (남성: 319), 02월01일 (여성: 477), 02월01일 (남성: 409), 03월01일 (여성: 449), and 03월01일 (남성: 361). The 'Count' column for the first row (01월01일 여성) is highlighted with a black box.

주문시간 per month	고객성별: Descending	Count
01월01일	여성	430
01월01일	남성	319
02월01일	여성	477
02월01일	남성	409
03월01일	여성	449
03월01일	남성	361

이 내용을 활용해서 Average Bucket Aggregation의 결과 (page 91)를 추적해보자

판매자평점 ◆	Overall Average of Sum of 상품가격 ◆
1	3,971,000
2	1,509,000
3	3,423,000
4	2,662,500
5	1,233,500

이 값은 어떻게 나온 것일까?

Average Bucket Aggregation 계산 과정에 사용되는 Bucket 정보 표시

The image shows the Elasticsearch Dev Tools interface with two main sections: a configuration panel on the left and a results table on the right.

Configuration Panel (Left):

- ① metrics:** Contains a Metric named "Overall Average of Sum of 상품가격" and a button "Add metrics".
- buckets:** Contains a "Split Rows" button and a "판매자평점" field.
- ② Sub Aggregation:** Contains a "Split Rows" button and a "Field" dropdown set to "구매사이트".
- Order By:** Set to "Custom Metric".
- Aggregation:** Set to "Count".
- Order:** Set to "Descending" with a size of 2.

Results Table (Right):

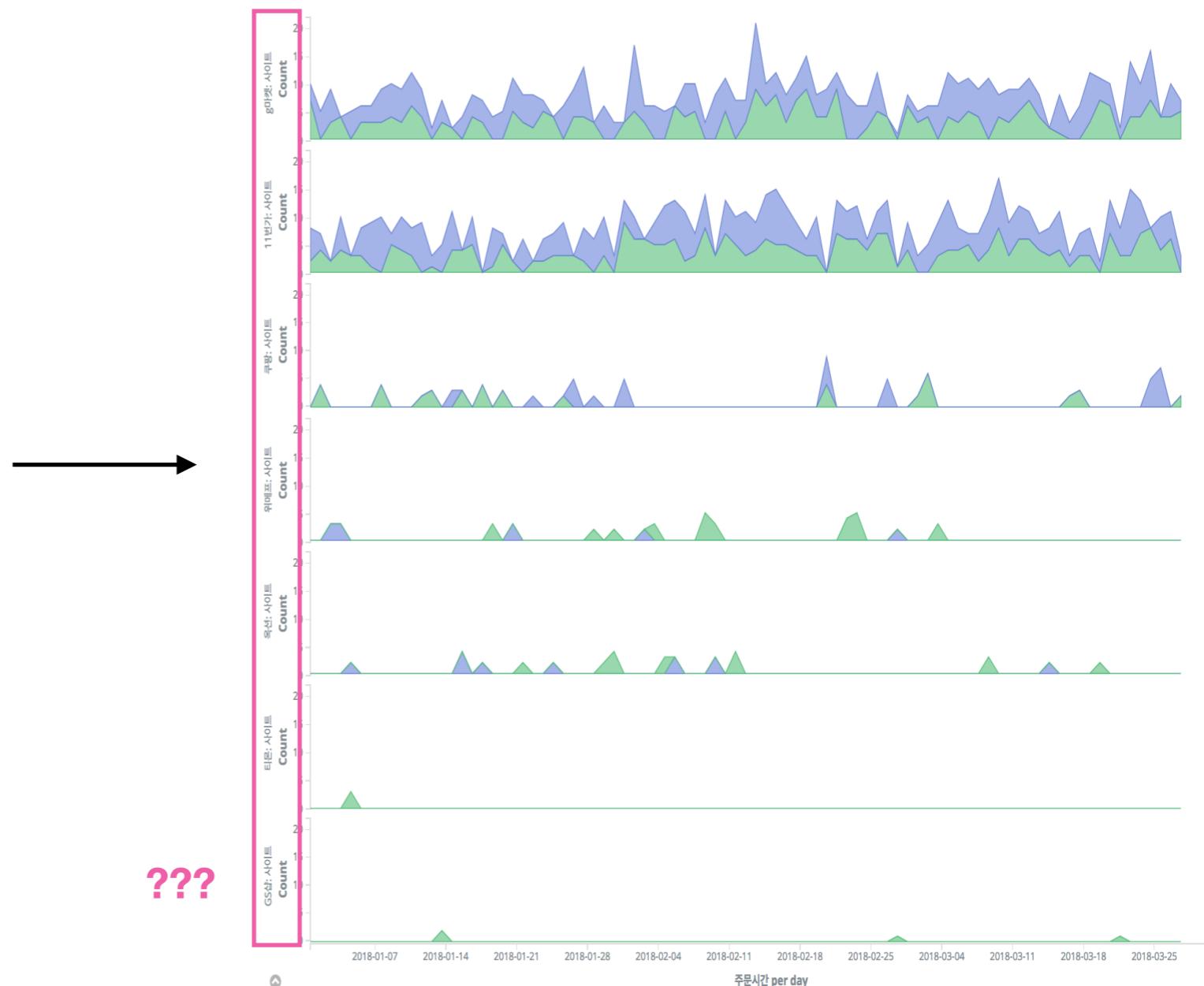
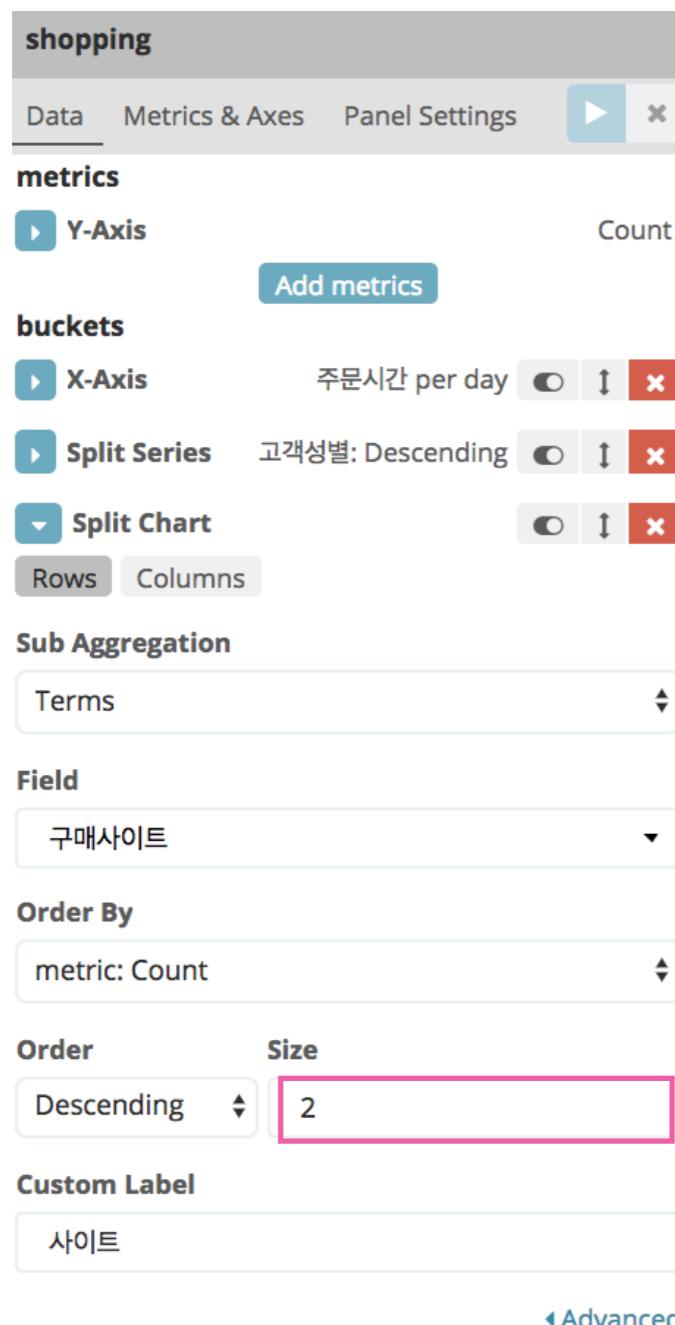
판매자평점	구매사이트: Descending	Overall Average of Sum of 상품가격
1	11번가	4,025,000 => 3,971,000
1	g마켓	3,917,000
2	11번가	1,523,000 => 1,509,000
2	g마켓	1,495,000
3	11번가	3,471,000 => 3,423,000
3	g마켓	3,375,000
4	11번가	2,638,000 => 2,662,500
4	g마켓	2,687,000
5	g마켓	1,319,000 => 1,233,500
5	11번가	1,148,000

- ①은 page 91 설정 유지
- ①의 계산 과정에 이용되는 Bucket 정보를 명시적으로 표시하기 위해 ②를 설정

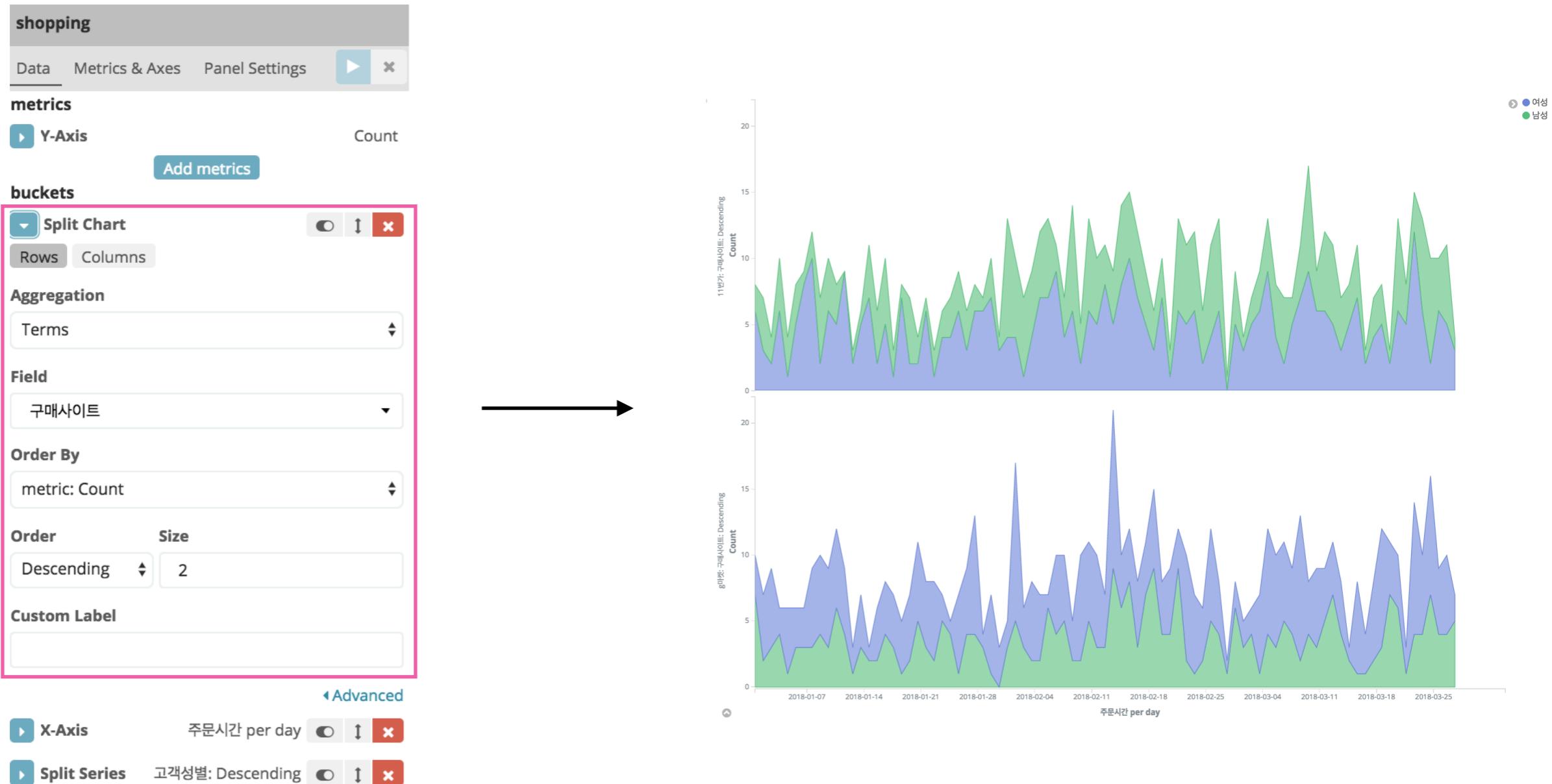
Q. 여러 Bucket Aggregation을 사용할 때 순서가 결과에 영향을 주는지?

A. 영향을 준다. page 12 및 page 28의 결과가 그러한 예이다.

“구매사이트”를 2개로 설정했음에도 총 7개의 “구매사이트”가 표시됐다



“Split Chart”를 buckets 최상위로 이동하니 원하는대로 표시됐다



기준 : “주문시간” 일 별로 “고객성별” 별로 documents가 많은 2개의 “구매사이트” 별로...

주문시간 per day	고객성별: Descending	구매사이트: Descending	Count
01월01일	남성	g마켓	7
01월01일	남성	11번가	2
01월01일	여성	11번가	6
01월01일	여성	g마켓	3
01월02일	남성	11번가	4
01월02일	남성	쿠팡	4
01월02일	여성	g마켓	5
01월02일	여성	11번가	3
01월03일	여성	g마켓	6
01월03일	여성	위메프	3

“주문시간”, “고객성별”에 종속되어 “구매사이트”가 선정되므로 전체로 봤을 경우 선정된 “구매사이트”가 2개보다 많을 수 있다.

변경 : documents가 가장 많은 2개의 “구매사이트” 별 “주문시간” 기준 일 별로 “고객성별” 별로 ...

주문시간 per day	고객성별: Descending	구매사이트: Descending	Count
11번가	01월01일	여성	6
11번가	01월01일	남성	2
11번가	01월02일	남성	4
11번가	01월02일	여성	3
11번가	01월03일	남성	2
11번가	01월03일	여성	2
11번가	01월04일	여성	6
11번가	01월04일	남성	4
11번가	01월05일	남성	3
11번가	01월05일	여성	1

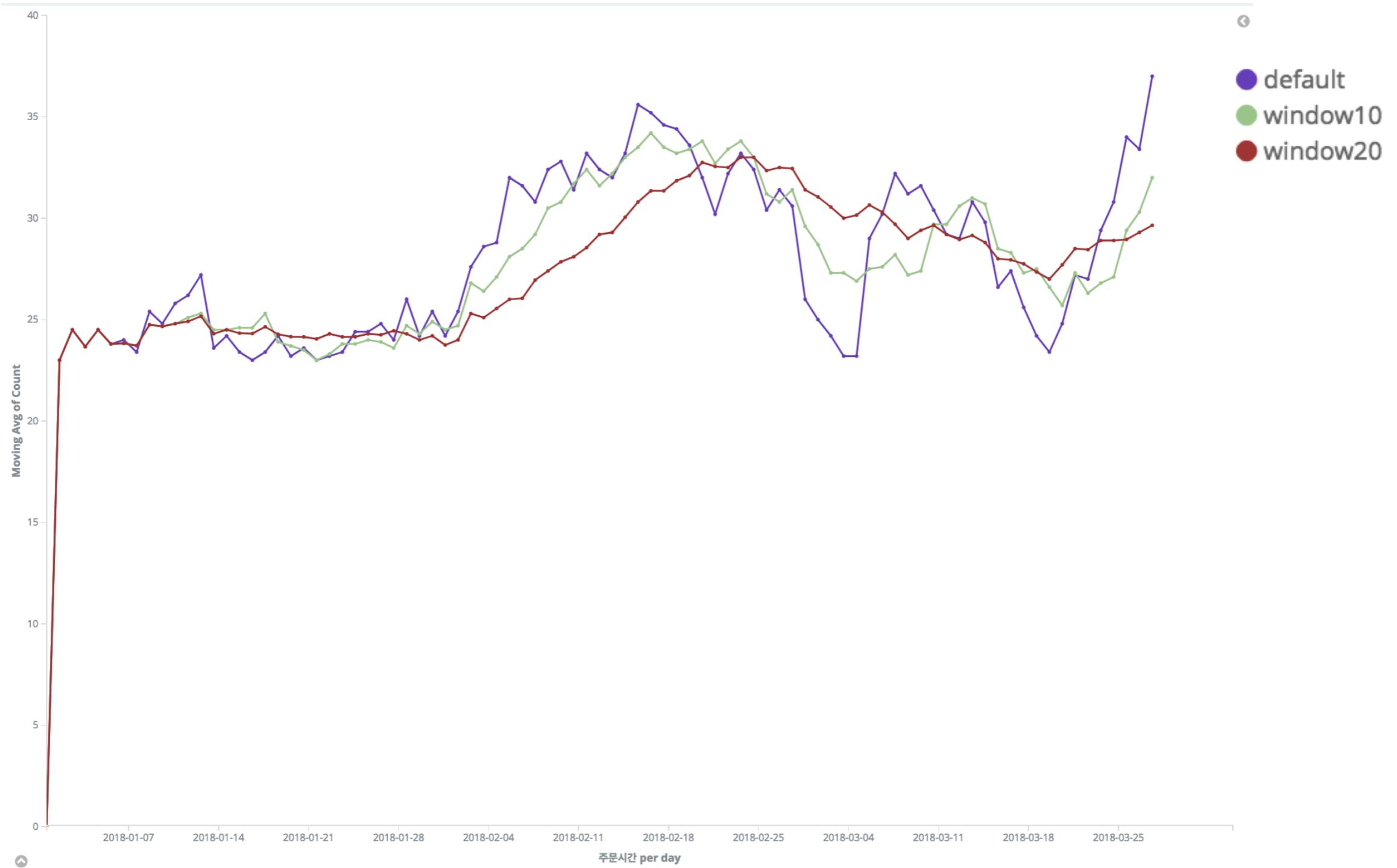
가장 먼저 “구매사이트”로 Bucket을 나누니 “구매사이트”가 원하는 대로 2개 이상으로 쪼개지지 않는다.

심화

Q. Aggregation 이용 시 customize 할 수 없나?

A. Aggregation에 사용되는 parameter 정보는 변경 가능하다.

예를 들어 moving average를 구할 때, window size를 변경할 수 있다.



How? JSON Input에 parameter 값 설정 (UI 형태로 제공되기를...)

shopping

Data Metrics & Axes Panel Settings ▶ ×

Moving Avg

Metric

Custom Metric

Aggregation

Count ◀ Advanced

Custom Label

default ◀ Advanced

▼ Y-Axis ○ ↔ ×

Aggregation

Moving Avg

Metric

Custom Metric

Aggregation

Count ◀ Advanced

Custom Label

window10

▼ Advanced

JSON Input ⓘ

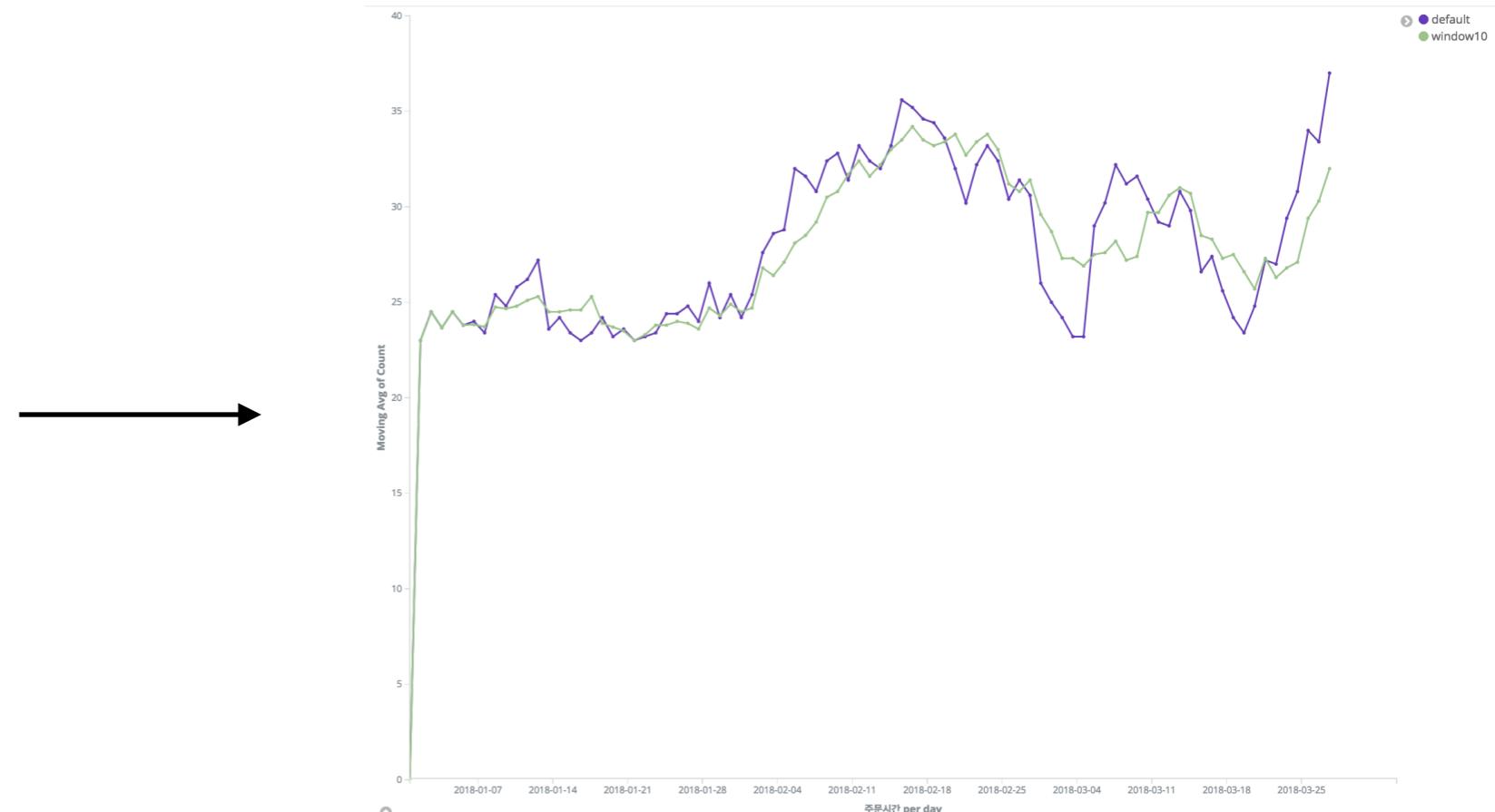
```
{"window": 10}
```

Add metrics

buckets

▶ X-Axis 주문시간 per day ○ ×

Add sub-buckets



변경하려는 parameter 이름과 값을 넣어준다

Aggregation 마다 어떤 parameter를 제공하는지 외워야 되는지?

1. 사용하려는 Aggregation 검색 
2. Parameter List에서 Parameter 확인
3. Kibana Visualize 화면 내의 적용하려는 Aggregation 내에서 Advanced 클릭
4. JSON Input에서 변경 {"parameter name" : "value"} 형태

JSON Input 예제 1

JSON Input을 이용해서 왼쪽 Visualize를 오른쪽과 같이 해보자.

힌트 : Terms Aggregation 의 “min_doc_count” parameter 사용

shopping

Data Options ▶ x

metrics

Metric Count Add metrics

buckets

Split Rows ✖

Aggregation

Terms

Field 구매사이트

Order By metric: Count

Order Descending Size 10

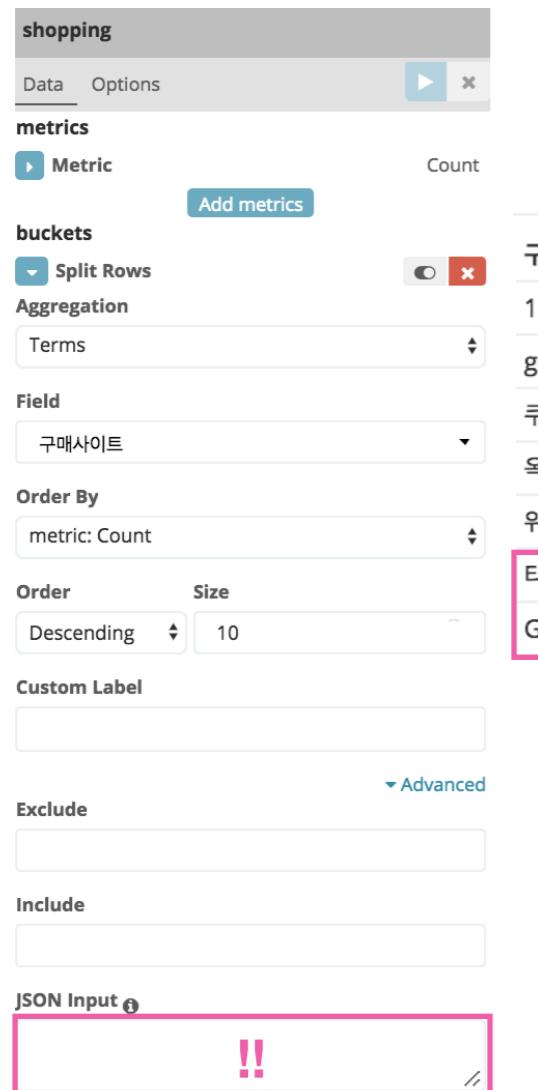
Custom Label

Advanced

Exclude

Include

JSON Input !



구매사이트: Descending		Count
11번가		756
g마켓		742
쿠팡		292
옥션		241
위메프		238
티몬		111
GS샵		65

제거



구매사이트: Descending		Count
11번가		756
g마켓		742
쿠팡		292
옥션		241
위메프		238

질문 및 Feedback은

gshock94@gmail.com로 주세요