Elastic Stack을 활용한 Data Dashboard 만들기

Week 1 - Data를 시각화해보자 kibana

Fast Campus

목차

강의 소개

강의 소개 - 목표 및 방향

강의가 끝나면	data가 주어지면 dashboard를 **구축**하고 needs에 맞게 **운영**할 수 있다.
단,	• ~~모든 기능 100% 마스터는 하지 않을 거고~~ • dashboard 구축 및 운영을 위한 전반적인 내용 학습과 • 문제가 생길 시 troubleshoot 하는 방법을 중심으로 배운다.
그러므로	• 검색엔진으로서 Elasticsearch • Elasticsearch Architecture **는 다루지 않을 것이다.** • (고급) query 및 query 최적화

강의 소개 - 운영 방식

FAQ	자주 물어보는 질문 정리 ♕
WIKI	Elastic Stack 간단한 사용법 정리 ♕
Questions	• 수업 중 : Slido ♕ • 수업 외 : [패스트캠퍼스] Elastic Stack을 활용한 Data Dashboard 만들기 CAMP ♕
Online Sources	• Elastic Stack and Product Documentation ♕ • Discuss the Elastic Stack ♕ • Facebook Elasticsearch Korea Group ♕ • Stack Overflow ♕

Kibana Visualization

	학습	실습
Elastic Stack workflow	✓	
(Elastic Stack workflow 중) Kibana workflow	✓	✓
(Kibana workflow 중) Visualize workflow	✓	✓
Kibana Dashboard 제작		✓
(Kibana Visualize를 위한) Elasticsearch Aggregation	✓	✓

Kibana Search & Filter

	학습	실습
Lucene Query Syntax	✓	✓
Search & Filter	✓	✓
Scripted Field	✓	✓
Kibana Visualization에서 Aggregation Parameter 변경		✓
(Kibana 상에서) Data Format 변경		✓

강의 소개 - 주별 커리큘럼 (3주)

Elasticsearch API

	학습	실습
Elasticsearch Index API	✓	✓
Elasticsearch Document API	✓	✓
Elasticsearch Search API	✓	✓
Elasticsearch Index Mapping	✓	✓
Elastic Stack 설치 (AWS EC2)		✓

Logstash Configuration

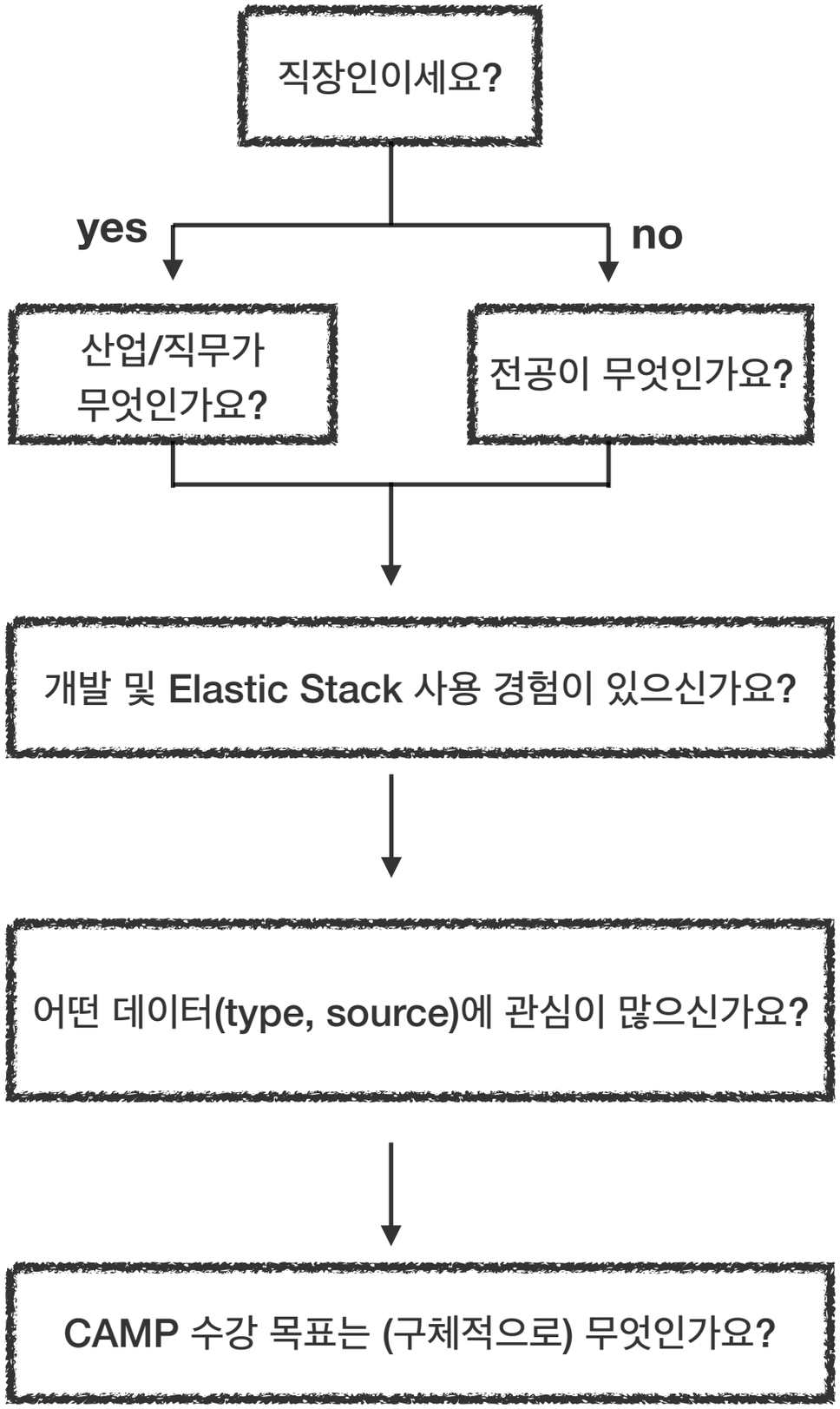
	학습	실습
(Elastic Stack workflow 중) Logstash workflow	✓	
Input Plugin	✓	✓
Filter Plugin	✓	✓
Output Plugin	✓	✓

강의 소개 - 주별 커리큘럼 (5주)

Elastic Stack을 활용한 Dashboard 구축 A-Z

	학습	실습
AWS EC2에 Elastic Stack 설치		✓
Elasticsearch Index 생성 및 Mapping 설정		✓
Logstash 통한 데이터 전처리 및 전송		✓
Kibana에 Elasticsearch Index 등록		✓
Kibana Discover를 통해 데이터 탐색		✓
Kibana Visualize를 통해 데이터 시각화		✓
Kibana Dashboard 제작		✓
Search와 Filter를 이용해 원하는 데이터 검색		✓

수강생 소개


직장인이세요?
yes
no
산업/직무가
무엇인가요?
전공이 무엇인가요?
개발 및 Elastic Stack 사용 경험이 있으신가요?
어떤 데이터(type, source)에 관심이 많으신가요?
CAMP 수강 목표는 (구체적으로) 무엇인가요?

오늘의 목표

항목	구체적 내용	난이도
Elastic Stack workflow	각 stack이 어떤 역할을 하고 어떤 흐름으로 이어지는지 이해하기	★☆☆☆☆
- Kibana workflow	Kibana 시각화를 위한 큰 틀 이해하기	★☆☆☆☆
- Index Pattern	시각화하고자 하는 Index(또는 Indices) 등록하기	★☆☆☆☆
- Discover	Discover를 이용해서 데이터 탐색하기	★★☆☆☆
- Visualize workflow	Visualize를 위한 전반적 흐름 이해하기	★☆☆☆☆
- Chart	특정 상황에서 어떤 Chart 사용해야 될 지 이해하기	★★★☆☆
- Aggregation	특정 상황에서 어떤 Aggregation 사용해야 될 지 이해하기	★★★★★
- Dashboard	Visualization을 적절히 배치해서 Dashboard 만들기	★☆☆☆☆

용어 정리

용어정리

RDBMS	***Elasticsearch***	Excel
Database	***Index***	Excel File
~~Table~~	~~***Type***~~	~~Sheet~~
Row	***Document***	Row
Column	***Field***	Column
Schema	***Mapping***	

- 위의 비교는 어디까지나 이해를 돕기 위할 목적일 뿐 정확히 일치하지는 않는다
- 6.0.0 이후에는 Index 1개에 Type 1개가 되어 사실상 폐지 👑
- 최소한 Index, Document, Field, Mapping 은 제대로 알고 넘어가자!

용어정리 (Index)

RDBMS

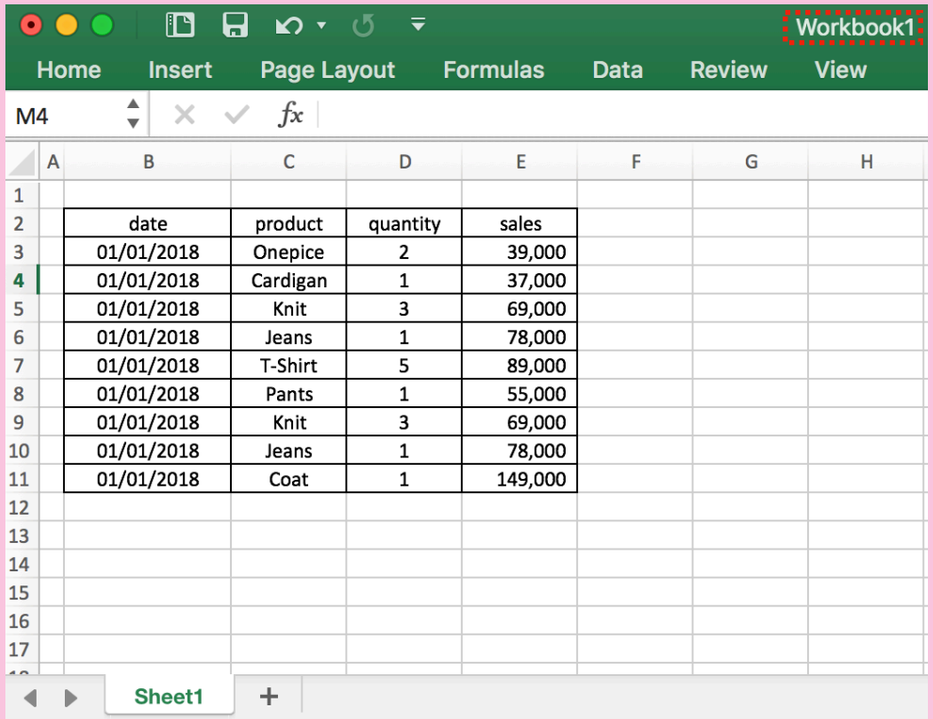
```
mysql> use Workbook1
Database changed
mysql> select * from Sheet1;
+------------+----------+----------+--------+
| date       | product  | quantity | sales  |
+------------+----------+----------+--------+
| 2018-01-01 | Onepiece |        2 |  39000 |
| 2018-01-01 | Cardigan |        1 |  37000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | T-Shirt  |        1 |  89000 |
| 2018-01-01 | Pants    |        1 |  55000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | Coat     |        1 | 149000 |
+------------+----------+----------+--------+
```

Elasticsearch


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 9,
    "max_score": 1,
    "hits": [
      {
        "_index": "workbook1",
        "_type": "sheet1",
        "_id": "5",
        "_score": 1,
        "_source": {
          "date": "2018-01-01",
          "product": "T-Shirt",
          "quantity": 5,
          "sales": 89000
        }
      },
```


⋮

Excel



	A	B	C	D	E	F	G	H
1								
2		date	product	quantity	sales			
3		01/01/2018	Onepice	2	39,000			
4		01/01/2018	Cardigan	1	37,000			
5		01/01/2018	Knit	3	69,000			
6		01/01/2018	Jeans	1	78,000			
7		01/01/2018	T-Shirt	5	89,000			
8		01/01/2018	Pants	1	55,000			
9		01/01/2018	Knit	3	69,000			
10		01/01/2018	Jeans	1	78,000			
11		01/01/2018	Coat	1	149,000			
12								
13								
14								
15								
16								
17								

용어정리 (Type)

RDBMS

```
mysql> use Workbook1
Database changed
mysql> select * from Sheet1;
+------------+---------+----------+--------+
| date       | product | quantity | sales  |
+------------+---------+----------+--------+
| 2018-01-01 | Onepiece |        2 |  39000 |
| 2018-01-01 | Cardigan |        1 |  37000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | T-Shirt  |        1 |  89000 |
| 2018-01-01 | Pants    |        1 |  55000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | Coat     |        1 | 149000 |
+------------+---------+----------+--------+
```

Elasticsearch


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 9,
    "max_score": 1,
    "hits": [
      {
        "_index": "workbook1",
        "_type": "sheet1",
        "_id": "5",
        "_score": 1,
        "_source": {
          "date": "2018-01-01",
          "product": "T-Shirt",
          "quantity": 5,
          "sales": 89000
        }
      },
```


⋮

Excel

Workbook1

Home | Insert | Page Layout | Formulas | Data | Review | View

M4

	A	B	C	D	E	F	G	H
1								
2		date	product	quantity	sales			
3		01/01/2018	Onepice	2	39,000			
4		01/01/2018	Cardigan	1	37,000			
5		01/01/2018	Knit	3	69,000			
6		01/01/2018	Jeans	1	78,000			
7		01/01/2018	T-Shirt	5	89,000			
8		01/01/2018	Pants	1	55,000			
9		01/01/2018	Knit	3	69,000			
10		01/01/2018	Jeans	1	78,000			
11		01/01/2018	Coat	1	149,000			
12								
13								
14								
15								
16								
17								

Sheet1

용어정리 (Document)

RDBMS

```
mysql> use Workbook1
Database changed
mysql> select * from Sheet1;
+------------+----------+----------+--------+
| date       | product  | quantity | sales  |
+------------+----------+----------+--------+
| 2018-01-01 | Onepiece |        2 |  39000 |
| 2018-01-01 | Cardigan |        1 |  37000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | T-Shirt  |        1 |  89000 |
| 2018-01-01 | Pants    |        1 |  55000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | Coat     |        1 | 149000 |
+------------+----------+----------+--------+
```

Elasticsearch


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 9,
    "max_score": 1,
    "hits": [
      {
        "_index": "workbook1",
        "_type": "sheet1",
        "_id": "5",
        "_score": 1,
        "_source": {
          "date": "2018-01-01",
          "product": "T-Shirt",
          "quantity": 5,
          "sales": 89000
        }
      },
```


⋮

Excel

Workbook1

Home | Insert | Page Layout | Formulas | Data | Review | View

M4

	A	B	C	D	E	F	G	H
1								
2		date	product	quantity	sales			
3		01/01/2018	Onepice	2	39,000			
4		01/01/2018	Cardigan	1	37,000			
5		01/01/2018	Knit	3	69,000			
6		01/01/2018	Jeans	1	78,000			
7		01/01/2018	T-Shirt	5	89,000			
8		01/01/2018	Pants	1	55,000			
9		01/01/2018	Knit	3	69,000			
10		01/01/2018	Jeans	1	78,000			
11		01/01/2018	Coat	1	149,000			
12								
13								
14								
15								
16								
17								

Sheet1 +

용어정리 (Field)

RDBMS

```
mysql> use Workbook1
Database changed
mysql> select * from Sheet1;
+------------+----------+----------+--------+
| date       | product  | quantity | sales  |
+------------+----------+----------+--------+
| 2018-01-01 | Onepiece |        2 |  39000 |
| 2018-01-01 | Cardigan |        1 |  37000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | T-Shirt  |        1 |  89000 |
| 2018-01-01 | Pants    |        1 |  55000 |
| 2018-01-01 | Knit     |        3 |  69000 |
| 2018-01-01 | Jeans    |        1 |  78000 |
| 2018-01-01 | Coat     |        1 | 149000 |
+------------+----------+----------+--------+
```

Elasticsearch


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 9,
    "max_score": 1,
    "hits": [
      {
        "_index": "workbook1",
        "_type": "sheet1",
        "_id": "5",
        "_score": 1,
        "_source": {
          "date": "2018-01-01",
          "product": "T-Shirt",
          "quantity": 5,
          "sales": 89000
        }
      },
```


⋮

Excel

Workbook1

Home | Insert | Page Layout | Formulas | Data | Review | View

M4

	A	B	C	D	E	F	G	H
1								
2		date	product	quantity	sales			
3		01/01/2018	Onepice	2	39,000			
4		01/01/2018	Cardigan	1	37,000			
5		01/01/2018	Knit	3	69,000			
6		01/01/2018	Jeans	1	78,000			
7		01/01/2018	T-Shirt	5	89,000			
8		01/01/2018	Pants	1	55,000			
9		01/01/2018	Knit	3	69,000			
10		01/01/2018	Jeans	1	78,000			
11		01/01/2018	Coat	1	149,000			
12								
13								
14								
15								
16								
17								

Sheet1

용어정리 (Mapping)

RDBMS

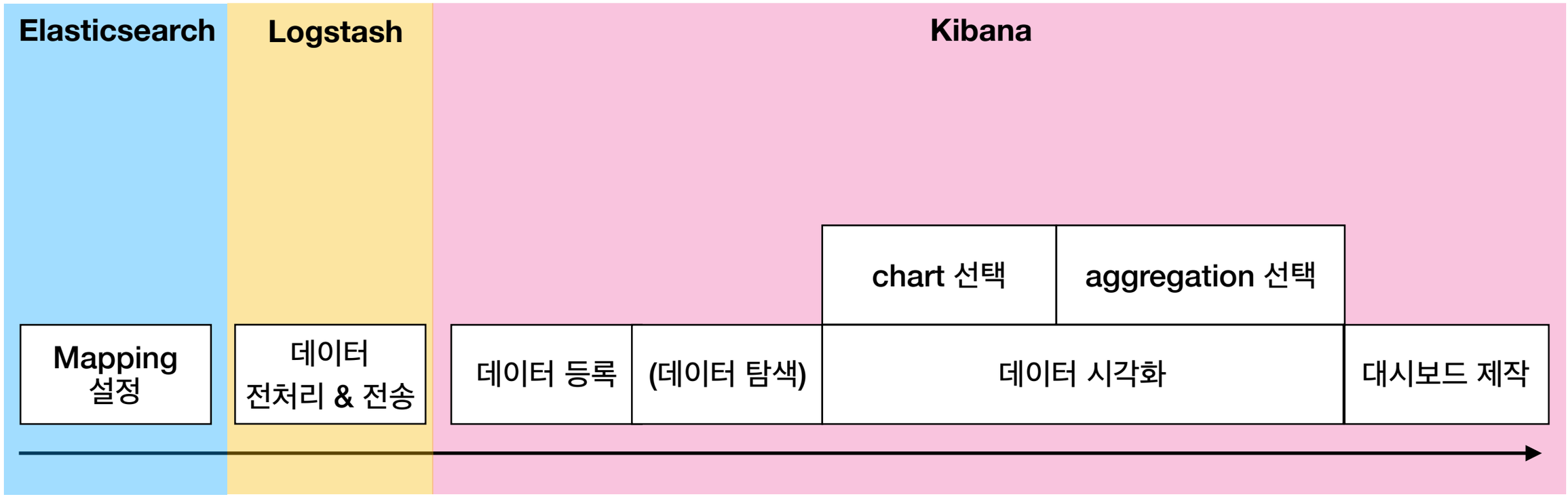
```
mysql> CREATE TABLE Sheet1 (
    -> date DATE,
    -> product VARCHAR(32),
    -> quantity INT(100),
    -> sales INT(100)
    -> );
```

Elasticsearch

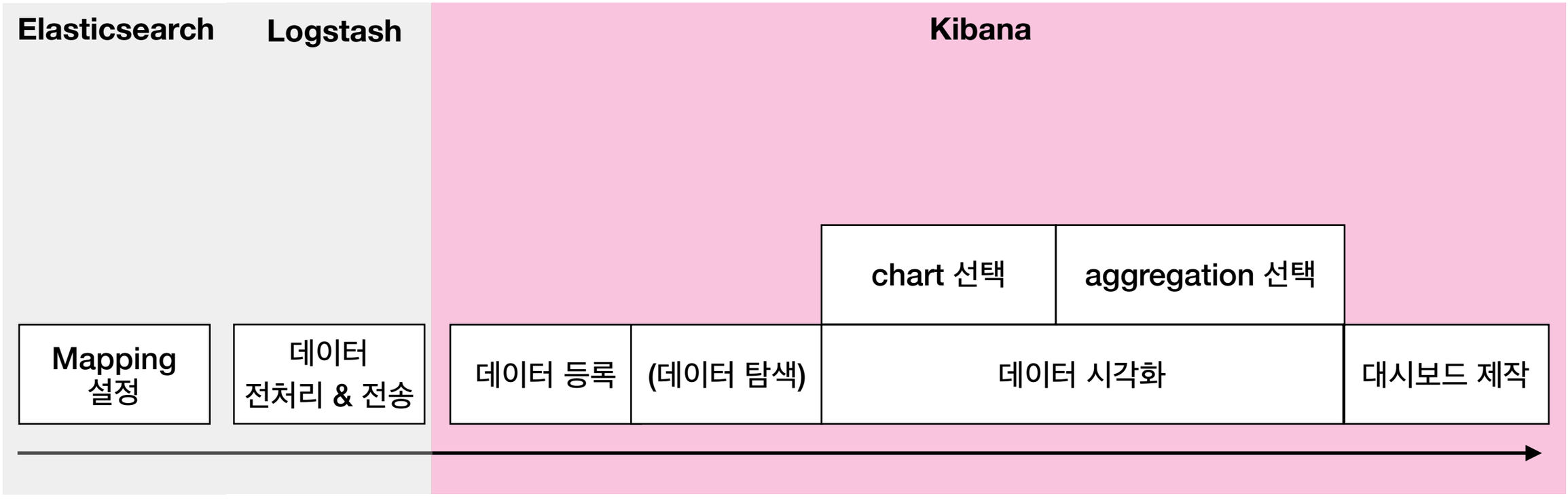
```
PUT workbook1
{
  "mappings": {
    "sheet1": {
      "properties": {
        "date": {
          "type": "date"
        },
        "product": {
          "type": "keyword"
        },
        "quantity": {
          "type": "integer"
        },
        "sales": {
          "type": "integer"
        }
      }
    }
  }
}
```

Elastic Stack Workflow

Elastic Stack Workflow



Elastic Stack Workflow



Kibana workflow

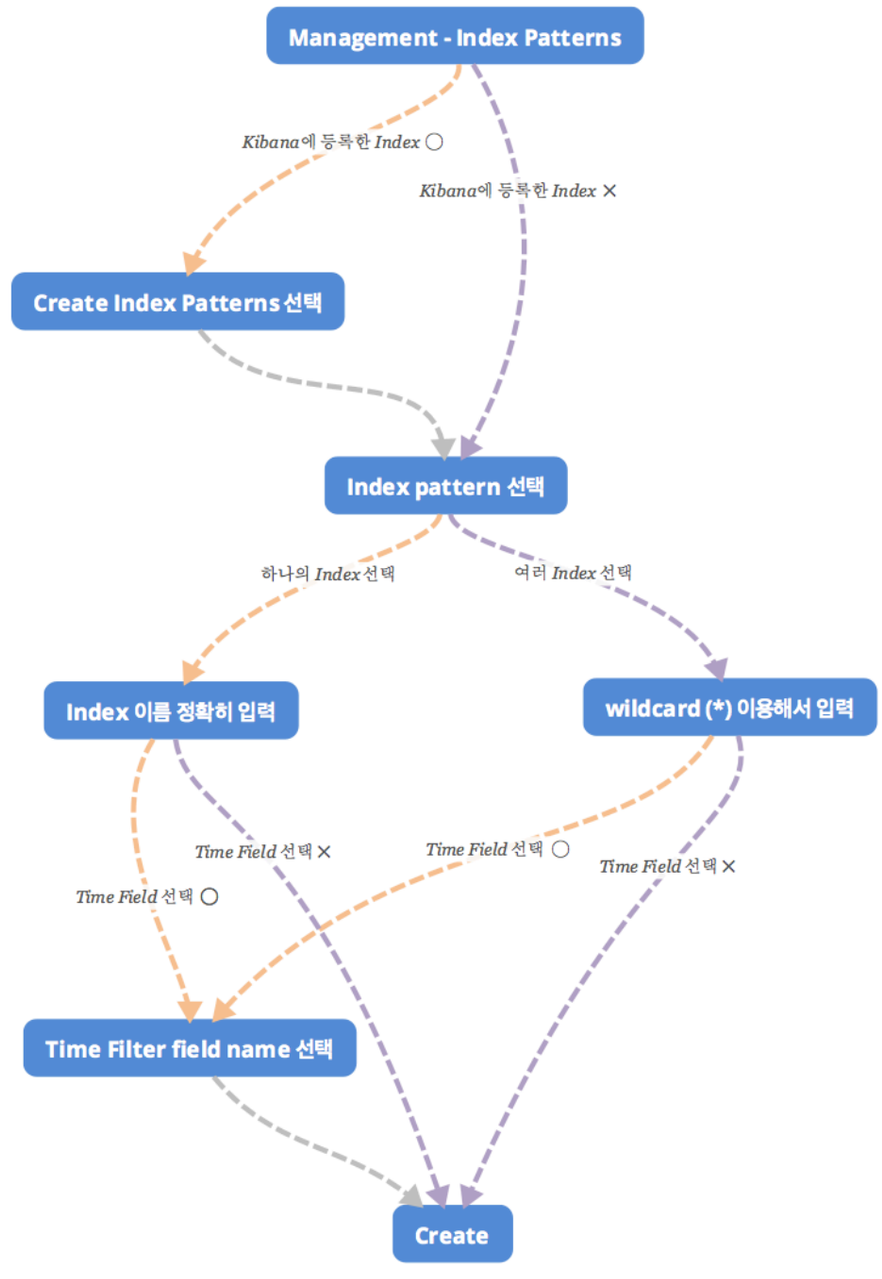
Kibana workflow

작업	어디에서?	상세
데이터 등록	Kibana Management	Elasticsearch에 저장된 데이터를 Kibana에서 사용할 수 있도록 등록
데이터 탐색	Kibana Discover	Kibana로 시각화하기 전에 데이터 탐색 (Optional)
데이터 시각화	Kibana Visualize	목적에 부합하는 Chart와 Aggregation 활용해서 Visualization 제작
대시보드 제작	Kibana Dashboard	생성한 Visualization을 적절히 배치하여 Dashboard 생성

Index 등록 👑

Kibana - Index 등록

Elasticsearch 데이터를 Kibana에서 Visualize 하기 위한 과정 (Timelion 예외)



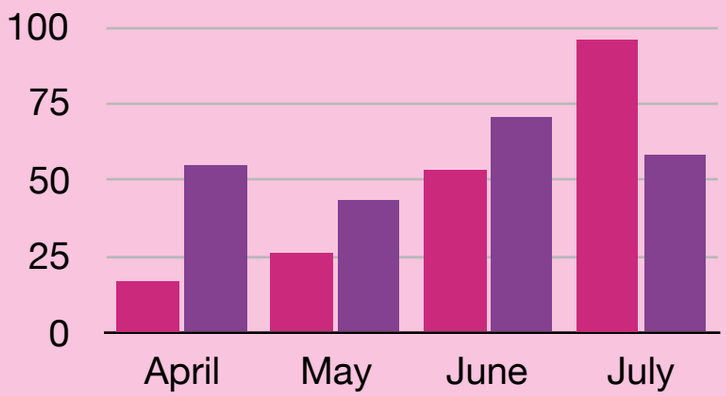
Kibana - Index 등록

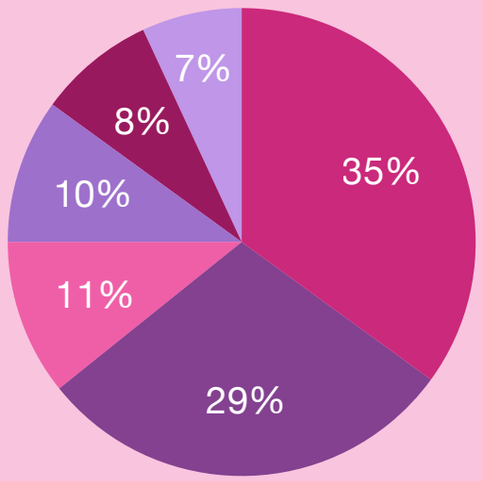
번호	실습	등록할 *index*	사용할 time field
1		*scenario1_{id}*	
2		*scenario2_{id}_1, senario2_{id}_2*	주문시간
3	✍	*scenario3_{id}*	
4	✍	*scenario4_{id}_1, scenario4_{id}_2*	주문시간
5	✍	*log_{id}_2018.01.01, log_{id}_2018.01.02*	주문시간

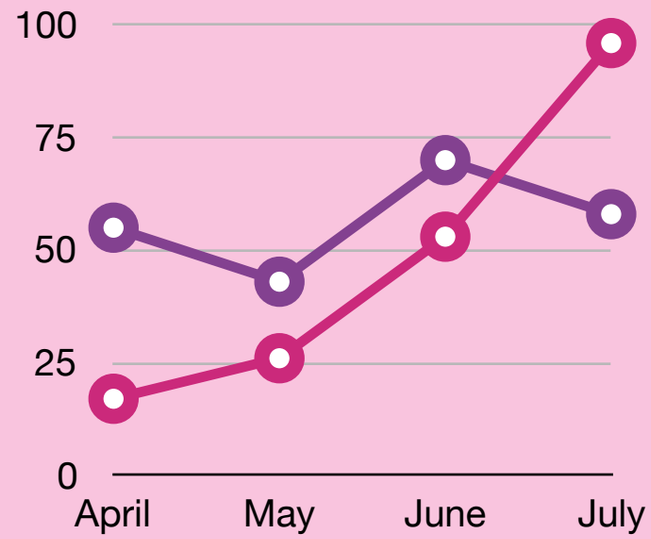
Q. Index 등록시 Wildcard는 왜 필요한지?

A. 데이터 저장은 **분산**, 검색 및 시각화는 **통합**해서 하기 위해, 즉 관리의 편의성!

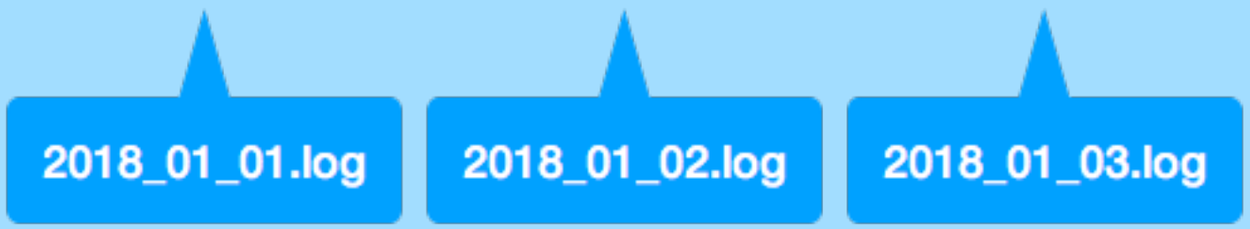
Kibana







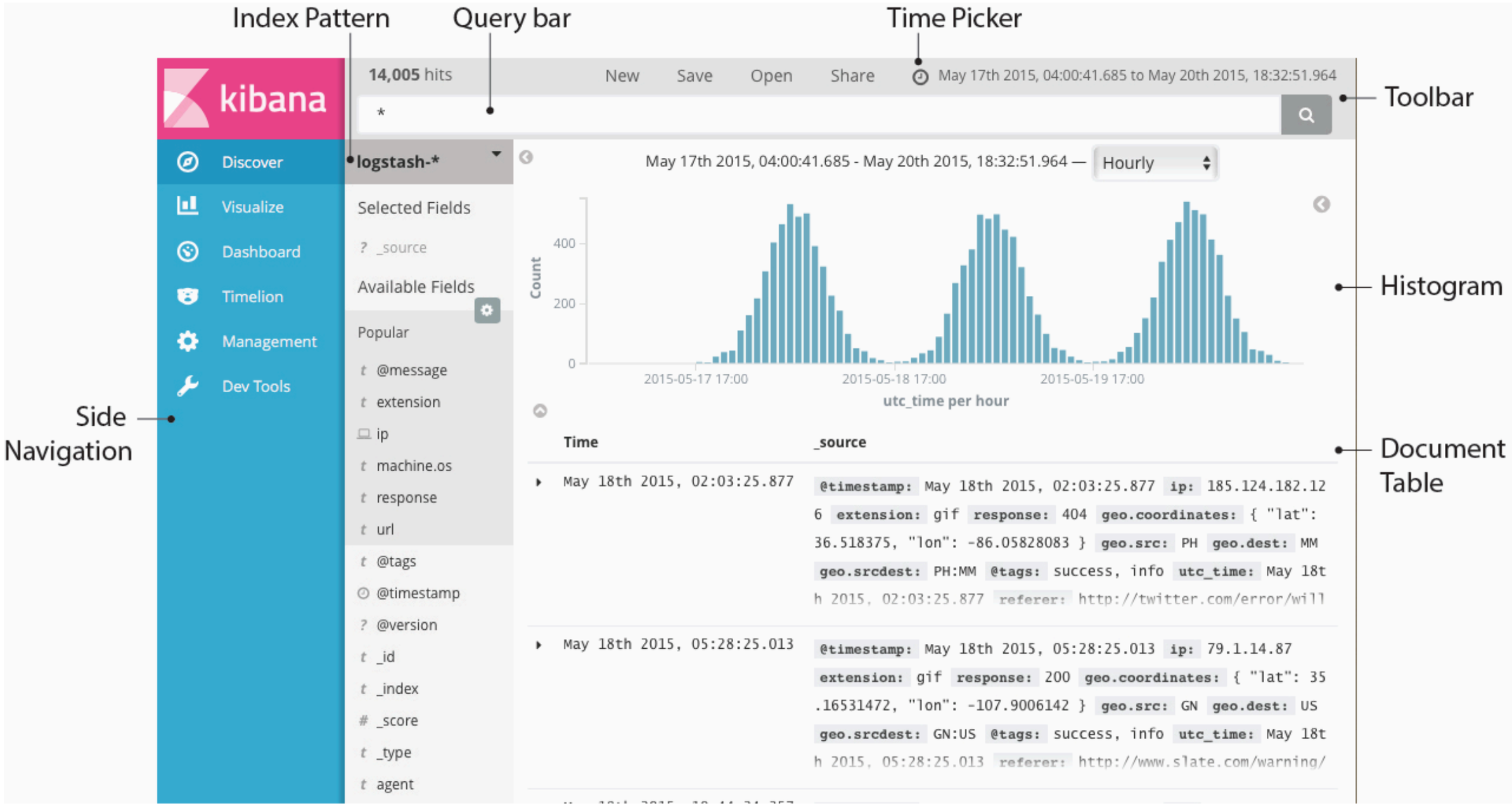
Elasticsearch



...

Kibana - Discover ♕

Kibana - Discover ♕



Kibana - Discover

데이터를 시각화 하기 전에 데이터를 탐색하는 과정

주요 기능	세부 기능
~~데이터 검색~~	~~(복잡한) 조건을 만족하는 데이터 선별적 탐색 가능~~
~~데이터 검색 저장~~	~~검색한 결고를 저장하여 Visaluze에서 사용~~
~~데이터 필터링~~	~~(간단한) 특정 조건을 만족하는 데이터 선별적 탐색 가능~~
데이터 조회	• 특정 Document를 Table/JSON 형태 조회 • Histogram 데이터를 csv 출력 • 특정 Field의 정보만 조회 • 특정 Field 값을 기준으로 정렬 • Histogram 특정 구간 내의 데이터 조회 • Histogram Bin 간격 설정
데이터 통계	• (선택한 Time Range 내의) Documents 개수 확인 • 특정 Field Value의 분포 확인 (주로 Categorical Data, 상위 500개) • 특정 Field에 non-null Value가 아닌 Documents 수 확인

Kibana - Discover ✍

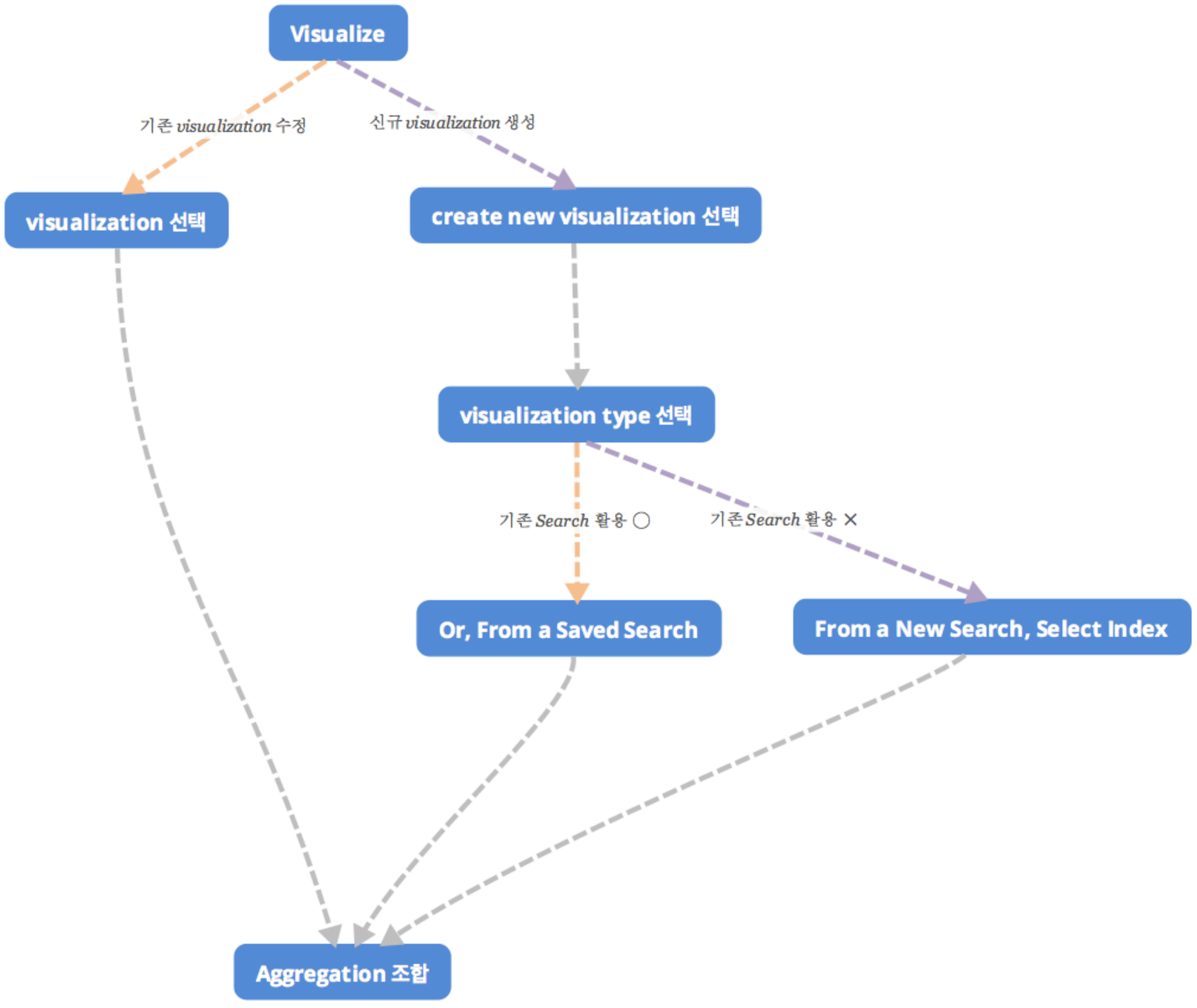
실습에 앞서 가장 중요한 2가지 설정을 하자

- *Time Picker : Last 1 year*
- *Index Pattern : shopping*

번호	항목	체크
1	Document 정보 Table/JSON 형태로 보기	
2	Histogram 데이터 csv 출력	
3	고객나이, 고객주소_시도, 상품분류 Field 정보만 보기	
4	(3번 설정을 유지한 상태로) 고객나이가 적은 순으로 정렬	
5	Histogram에서 간격을 Monthly로 설정	
6	(5번 설정을 유지한 상태로) 2017년 5월1일 ~ 2017년 8월31일 데이터만 조회	
7	(6번 설정을 유지한 상태에서) Documents는 총 몇 개 인가?	2549
8	(6번 설정을 유지한 상태에서) 상품분류 Field의 Value 분포는 어떤가?	
9	(6번 설정을 유지한 상태에서) 배송메모 Field가 존재하는 Documents는 총 몇 개 인가?	2549


Kibana Visualize workflow

Kibana Visualize workflow



만들려고 봤더니..

- metrics
- buckets
- x-axis, y-axis
- split series, split chart, split group,
- aggregation,
- field, order by, order, size,
- custom label

⋮

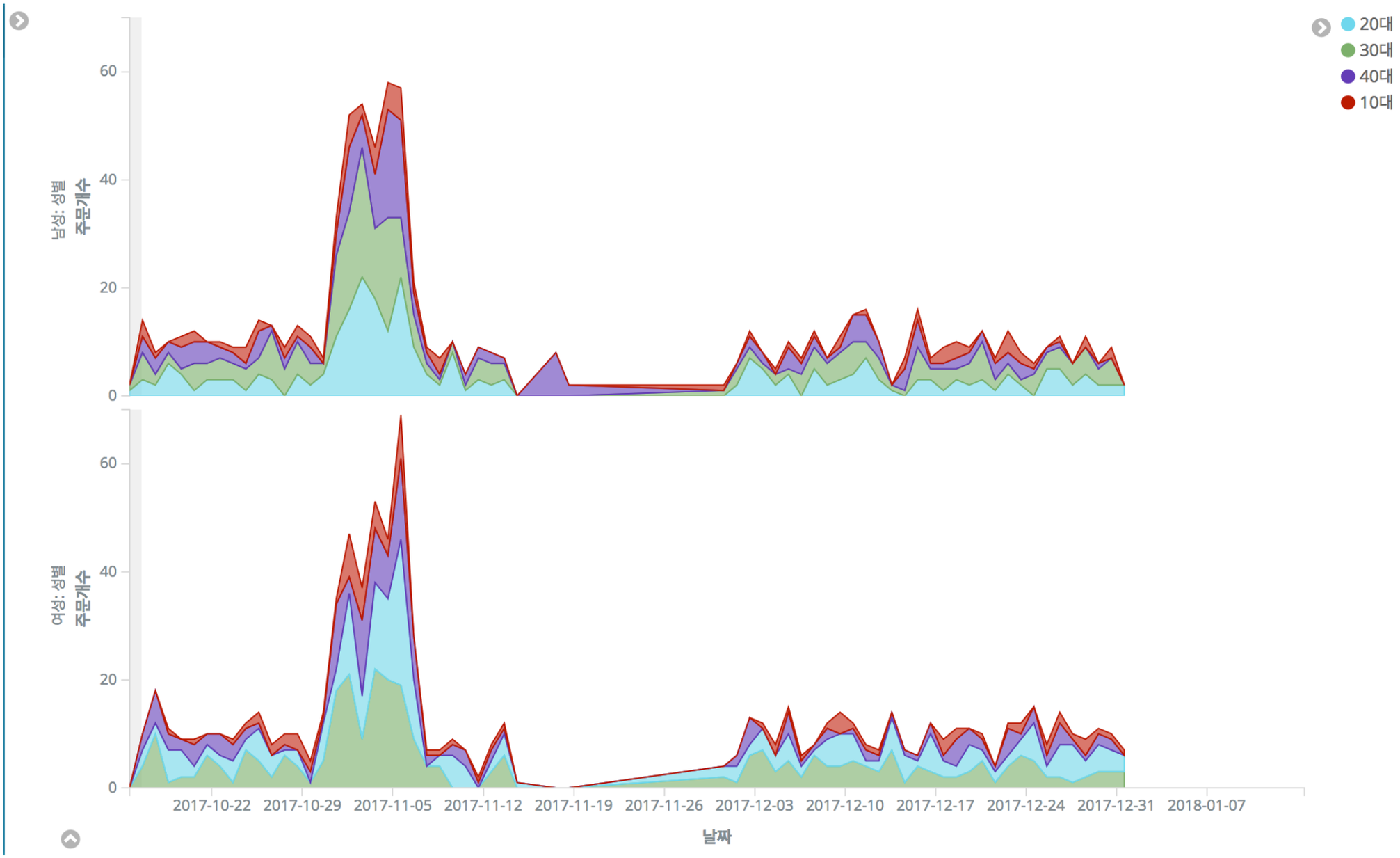
단계별로 어떻게 적용되는지 직접 하면서 익혀보자

기본 Visualization Plugin 👑

Visualize - Area

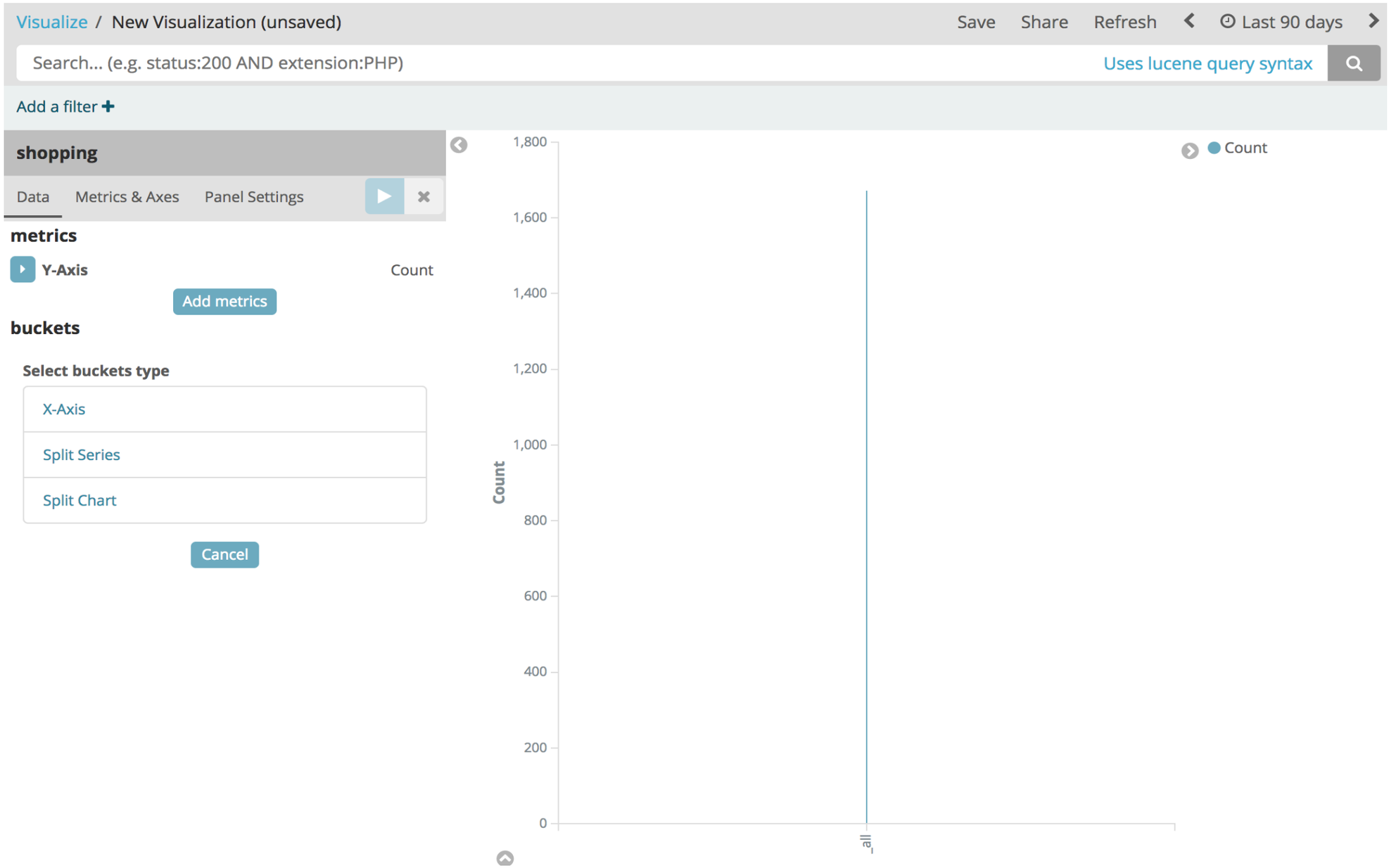
Visualize - Area

이런 Area Chart를 만들어보자



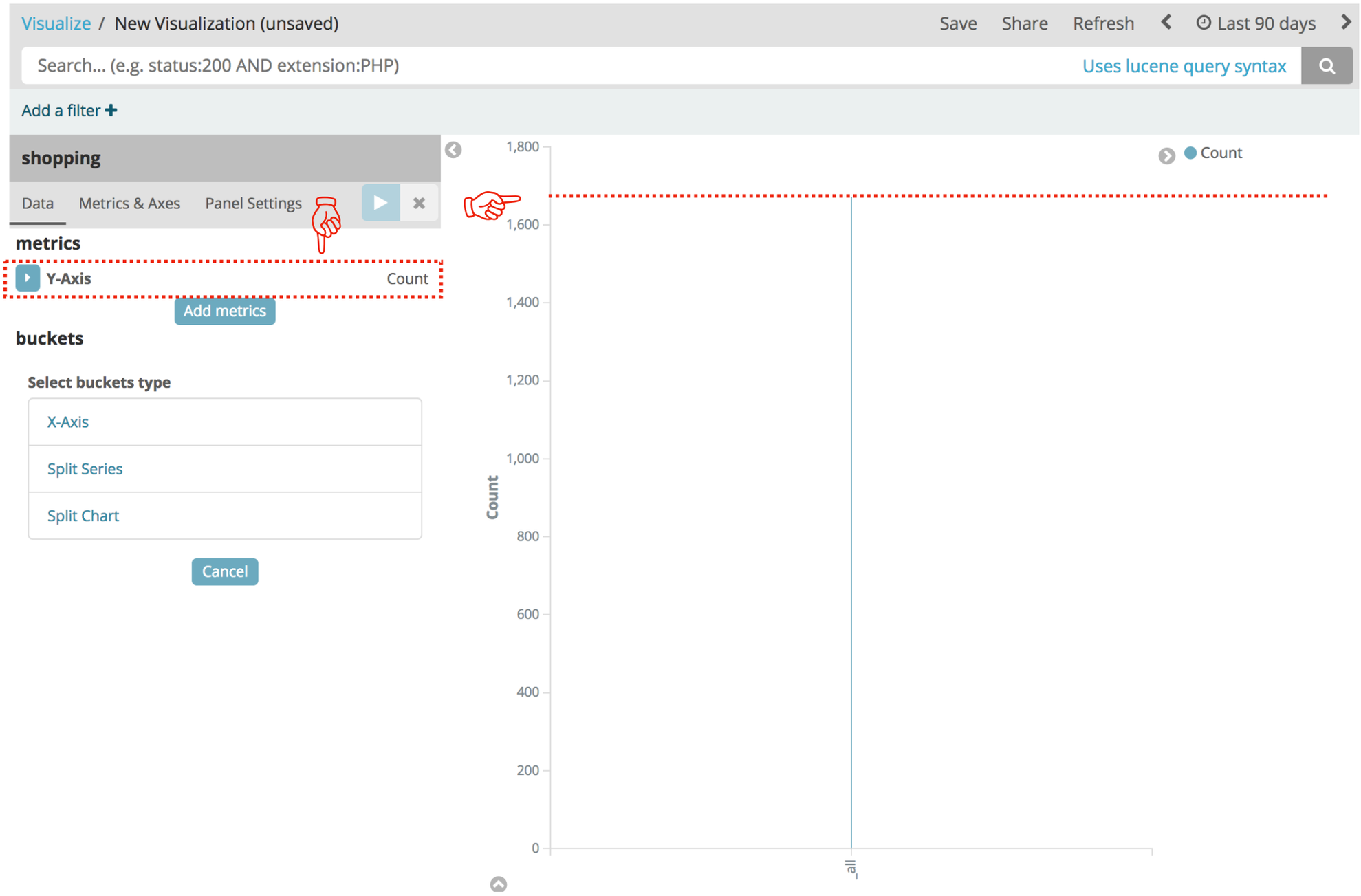
Visualize - Area

Area Chart를 선택하면 처음에 이런 화면이 나온다



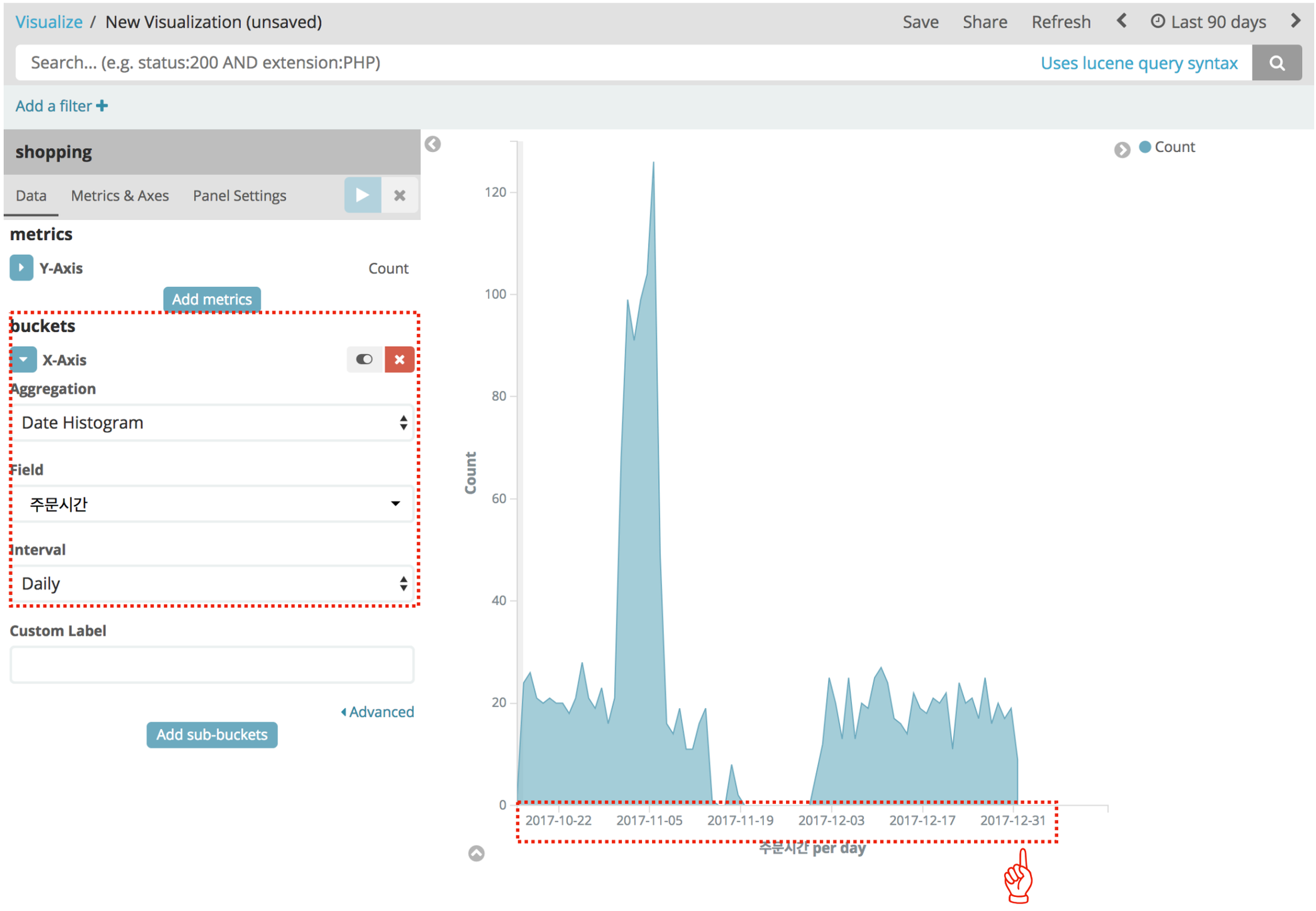
Visualize - Area

metrics를 ~~(aggregation은 잘 모르겠지만)~~ 통해 숫자가 생성된다



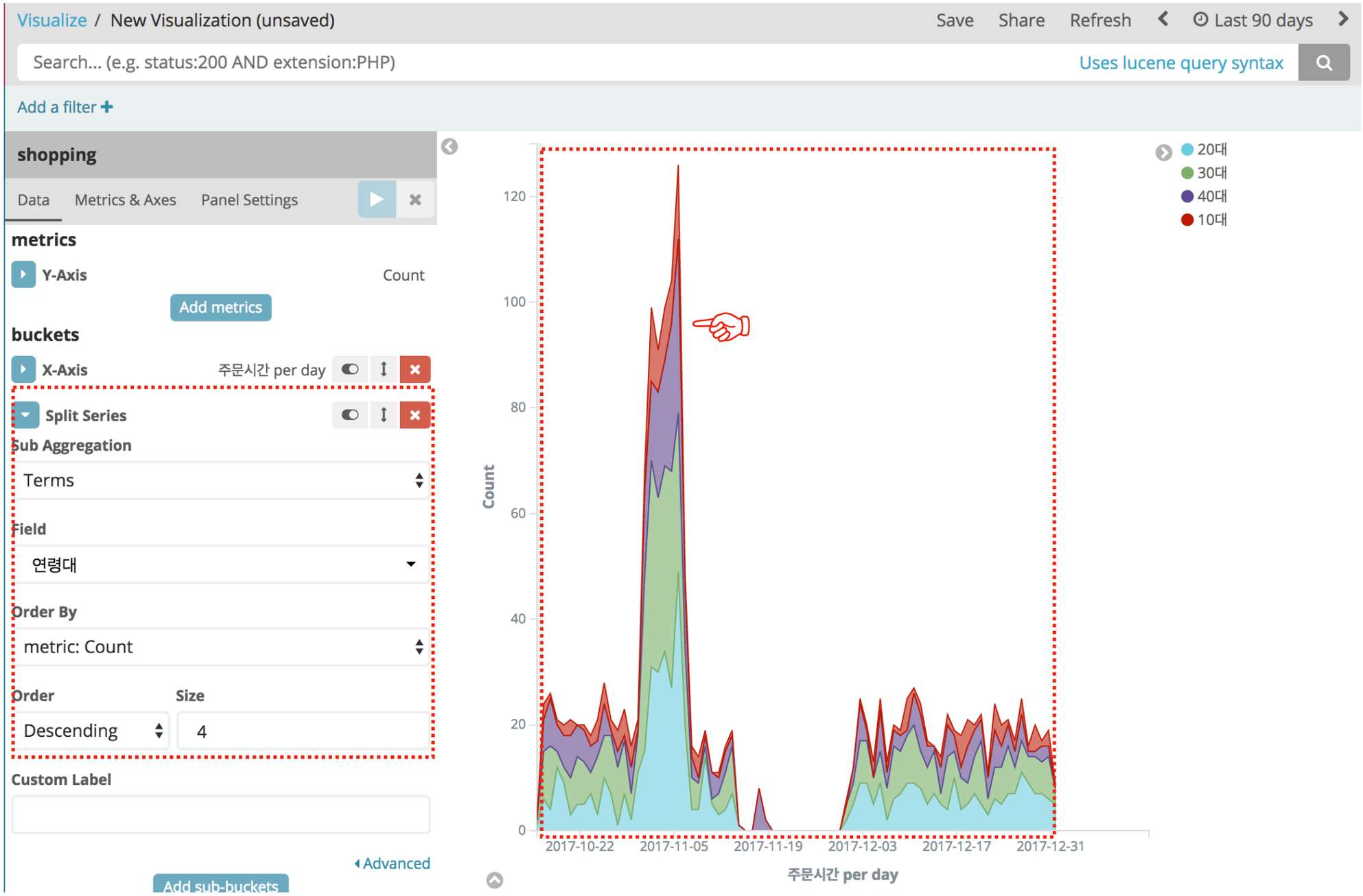
Visualize - Area

buckets (X-Axis)를 ~~(aggregation은 잘 모르겠지만)~~ 통해 x축 그룹이 생성된다



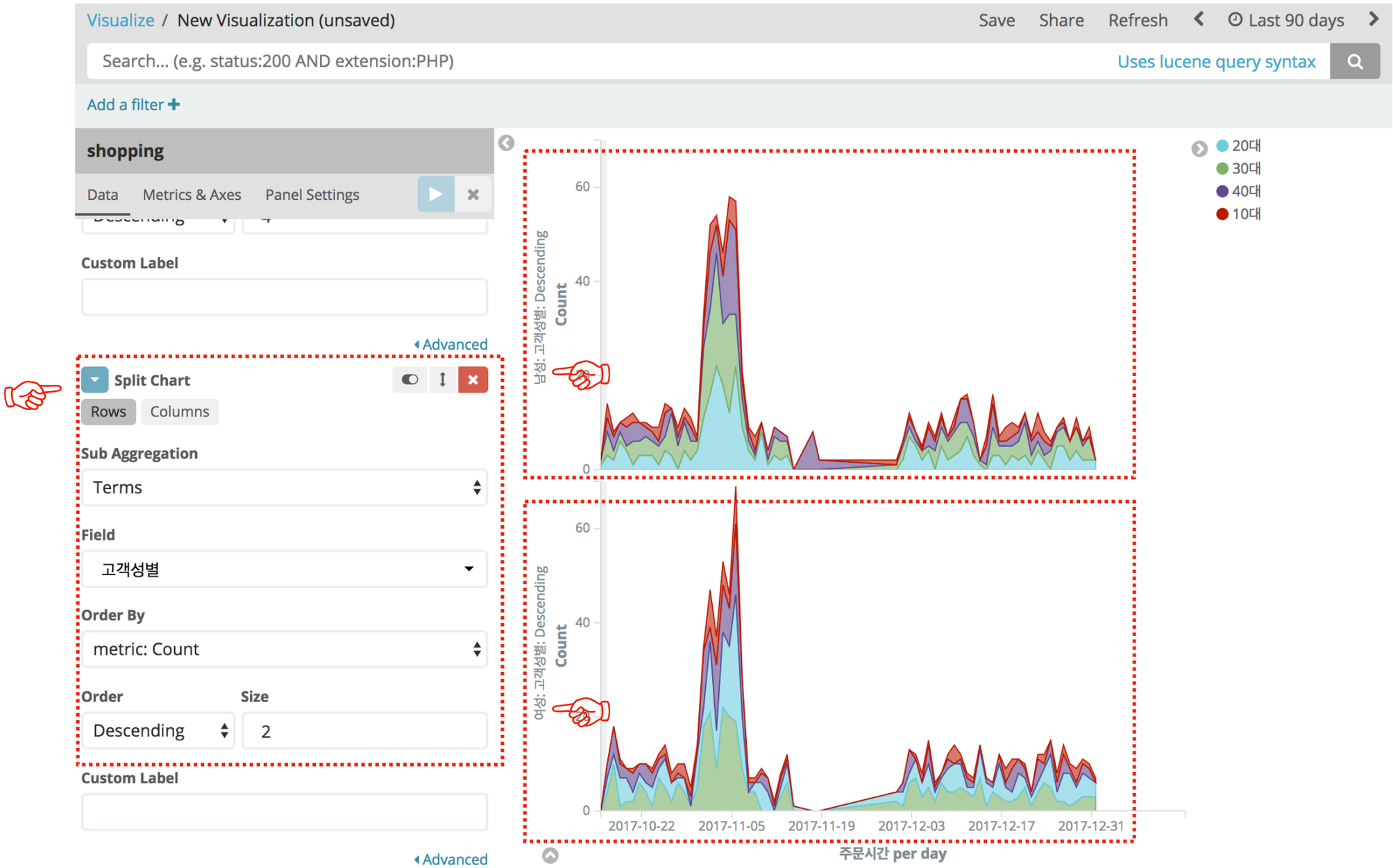
Visualize - Area

buckets (split series)를 ~~(aggregation은 잘 모르겠지만)~~ 통해 동일한 y축 그룹이 생성된다



Visualize - Area

split chart를 ~~(aggregation은 잘 모르겠지만)~~ 통해 특정 기준에 따라 차트가 같은 포맷으로 나뉜다.

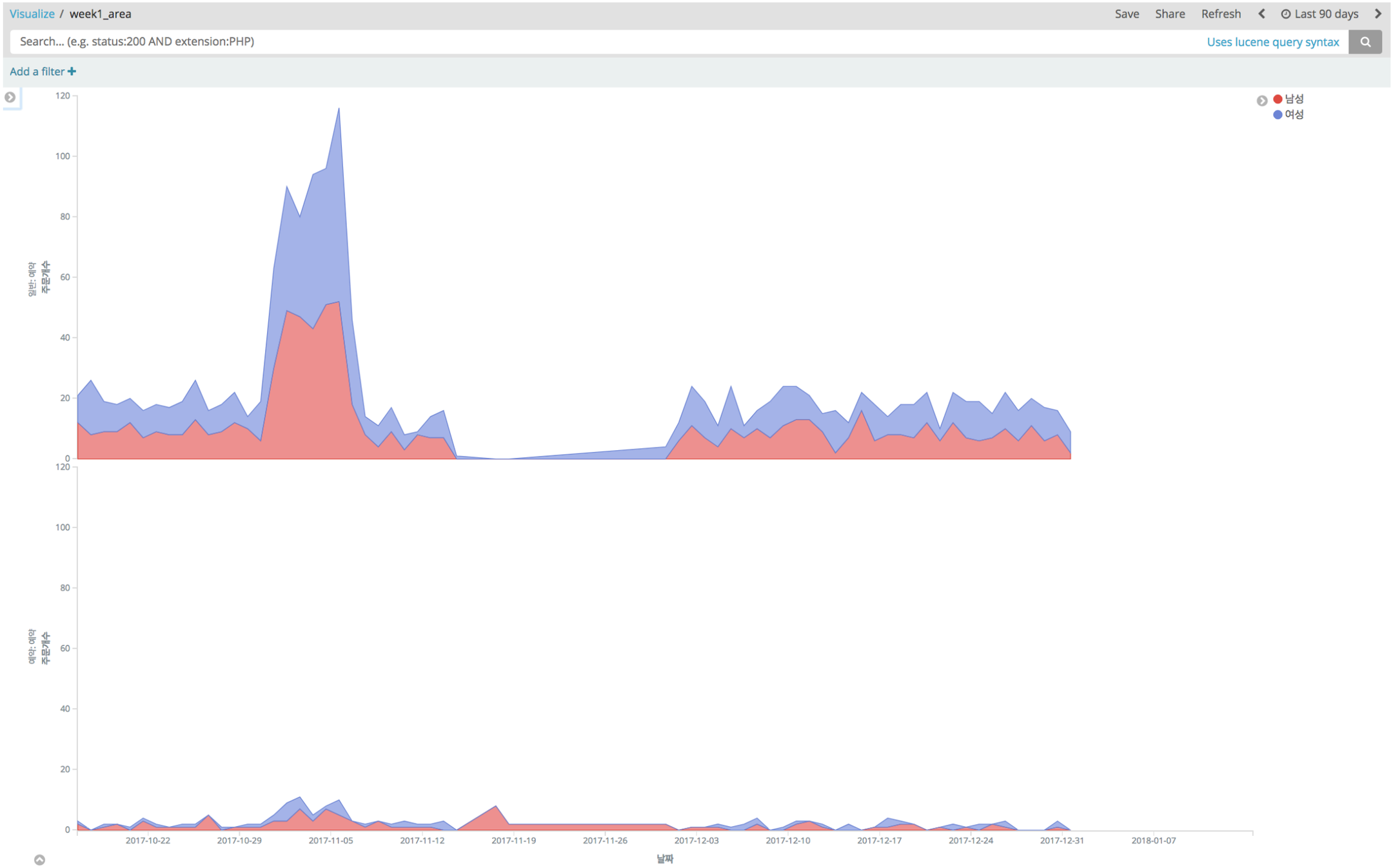


Visualize - Area

custom label로 label을 변경할 수 있다

Visualize - Area

다음과 같은 Area Chart를 만들고 저장하자 (힌트는 다음 페이지) ✍



Visualize - Area

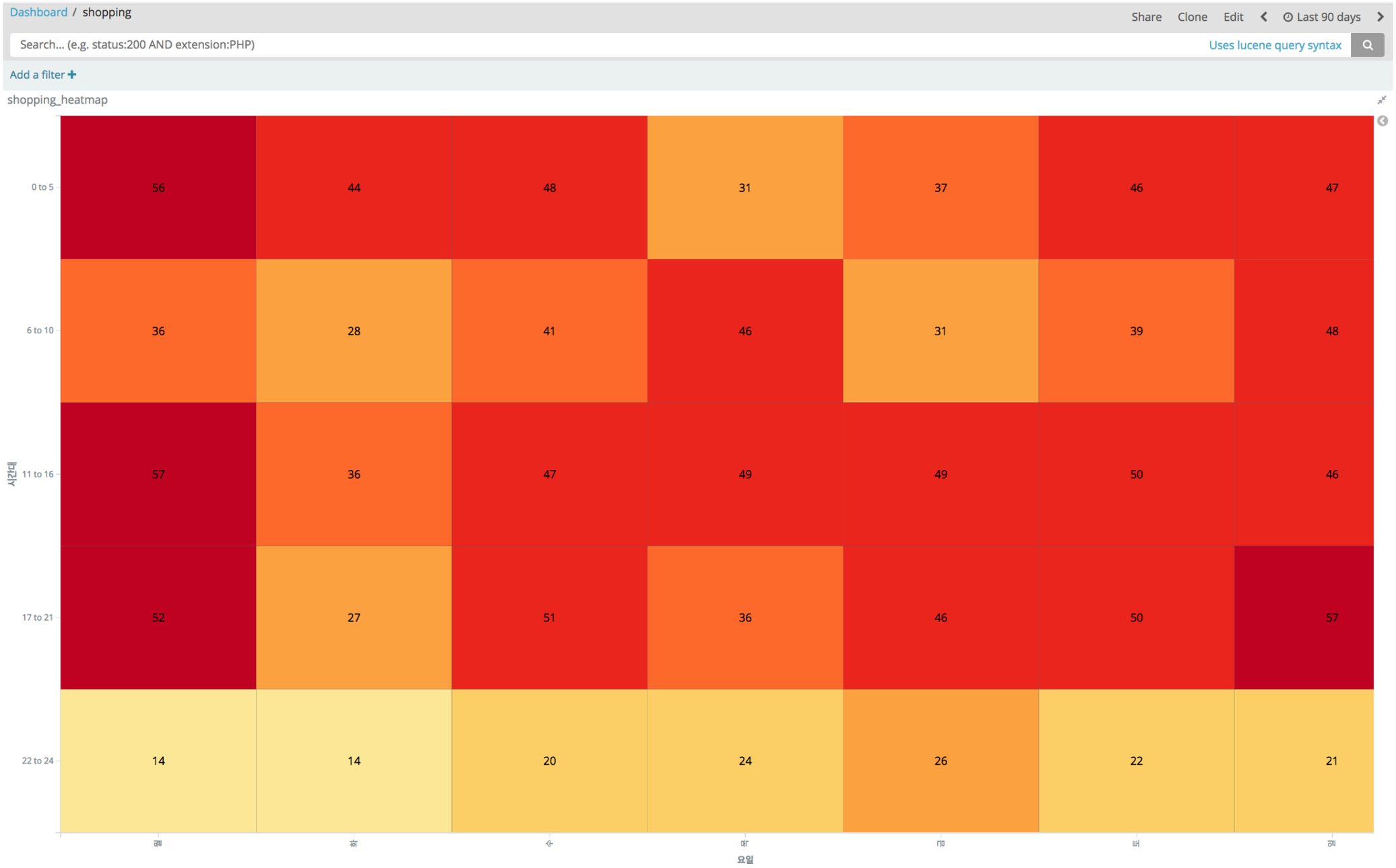
다음 조건을 만족하는 Area Chart를 만들어보자 ✍

- Index : shopping
- Time Range : Last 90 days
- metrics : Count
- buckets
  - X-Axis : “주문시간” Field를 기준으로 Daily하게 Split
  - Split Series : “고객성별” Field를 기준으로 Split
  - Split Chart ：위의 조건을 만족하는 Area Chart 생성 후 “예약여부” Field를 기준으로 Rows Split

Visualize - Heat Map

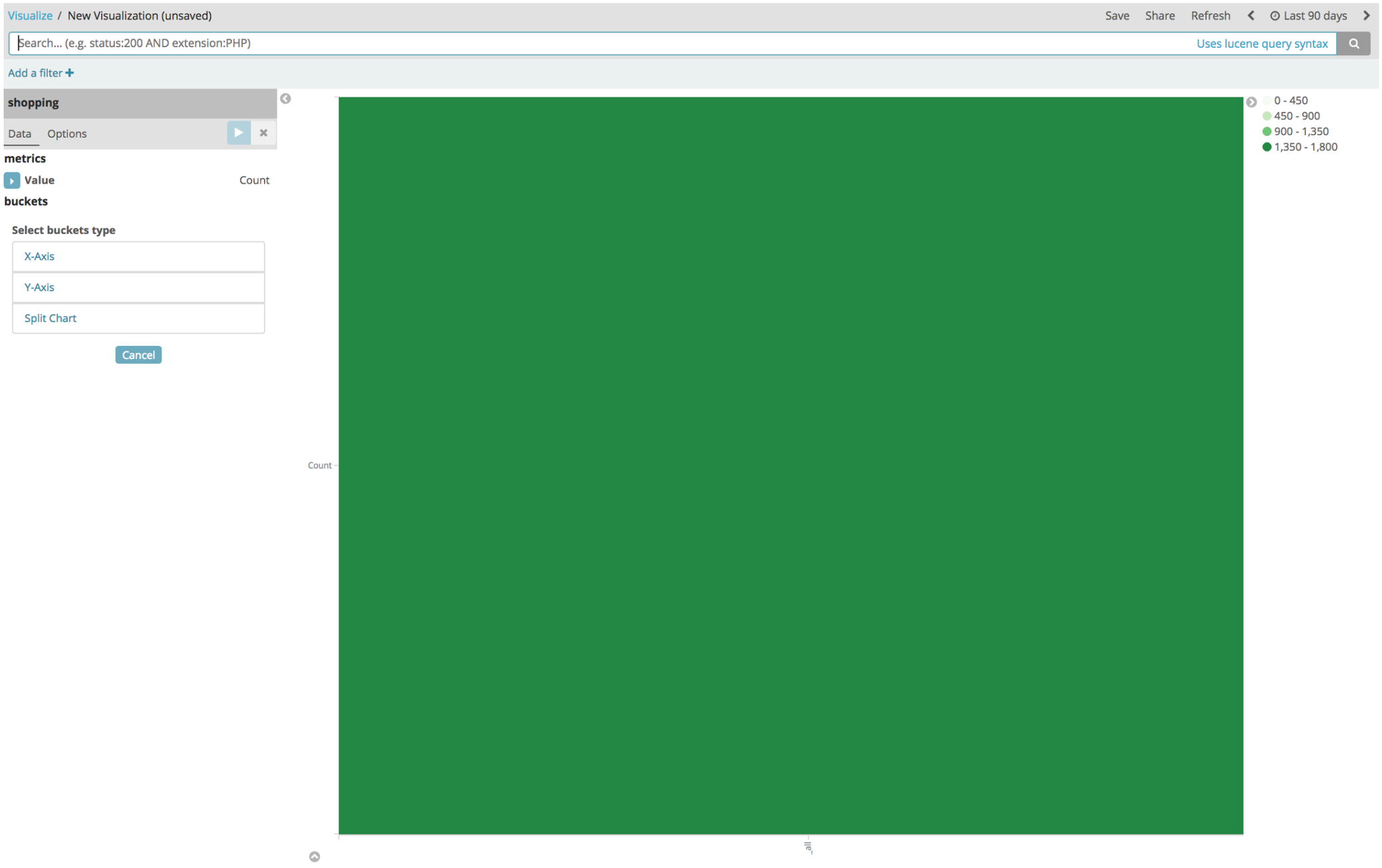
Visualize - Heat Map

이런 Heat Map을 만들어보자



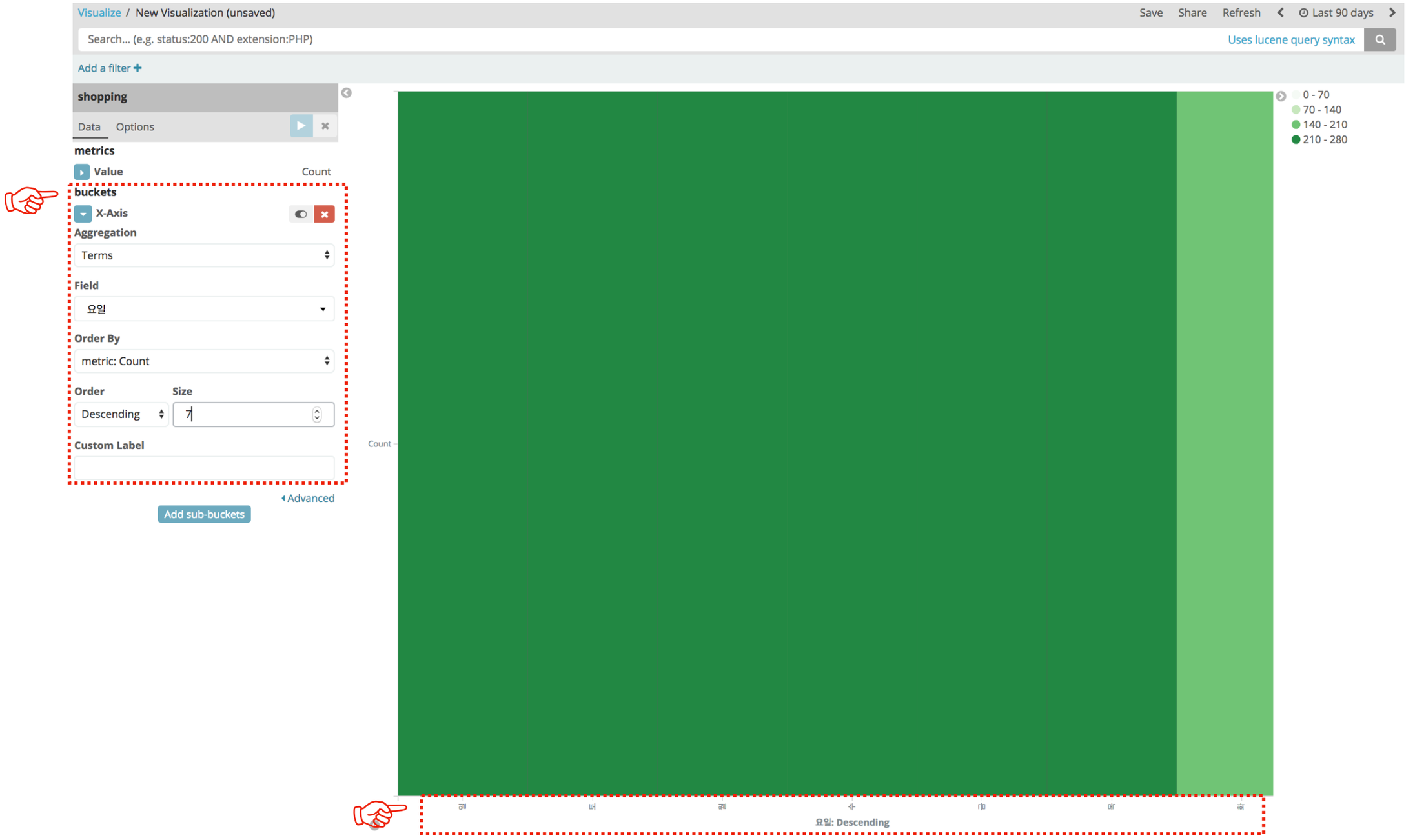
Visualize - Heat Map

Heat Map을 선택하면 처음에 이런 화면이 나온다



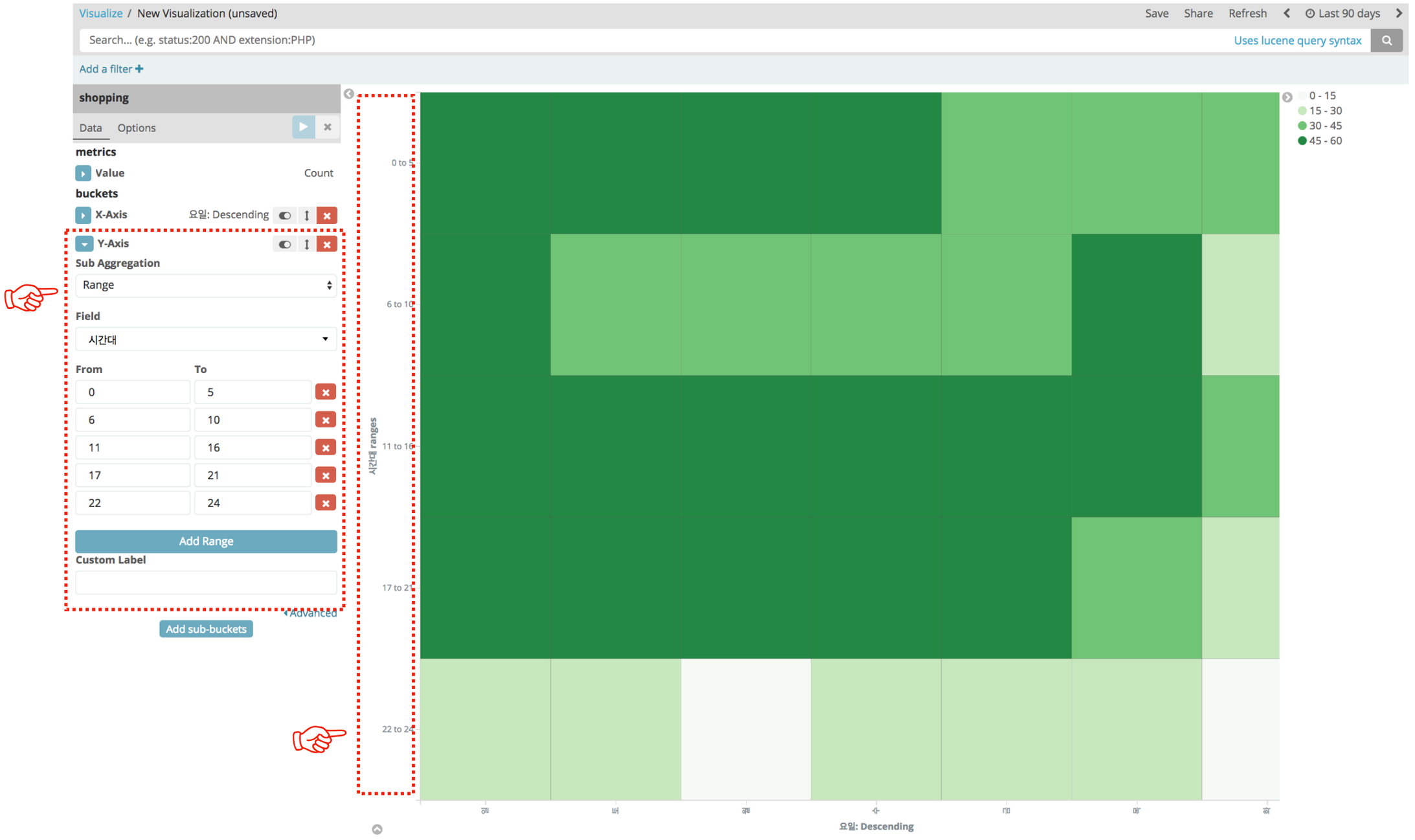
Visualize - Heat Map

buckets (X-Axis)를 통해 ~~(aggregation은 잘 모르겠지만)~~ x축이 요일별로 나뉜다



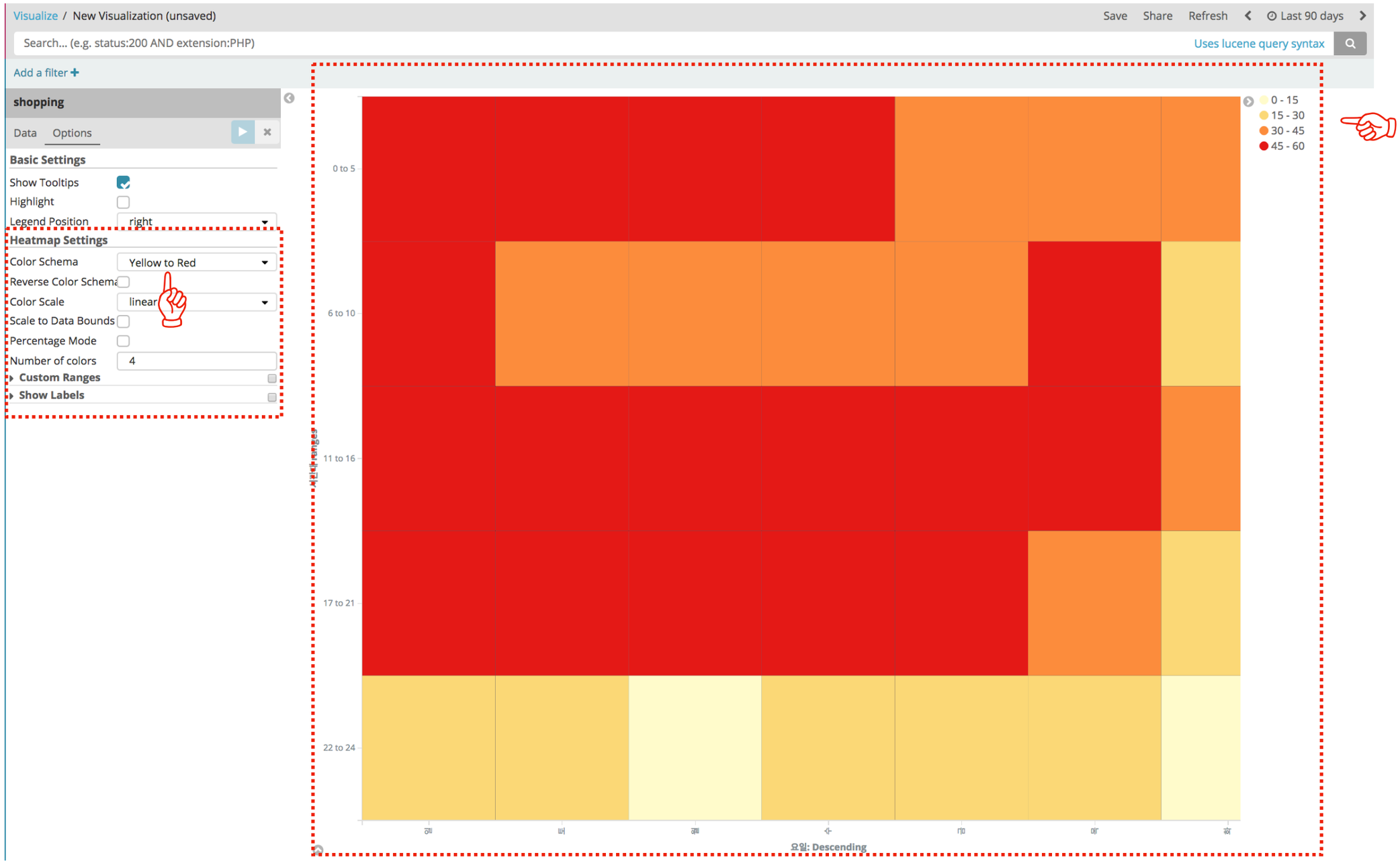
Visualize - Heat Map

buckets (Y-Axis)를 통해 ~~(aggregation은 잘 모르겠지만)~~ y축이 시간대별로 나뉜다



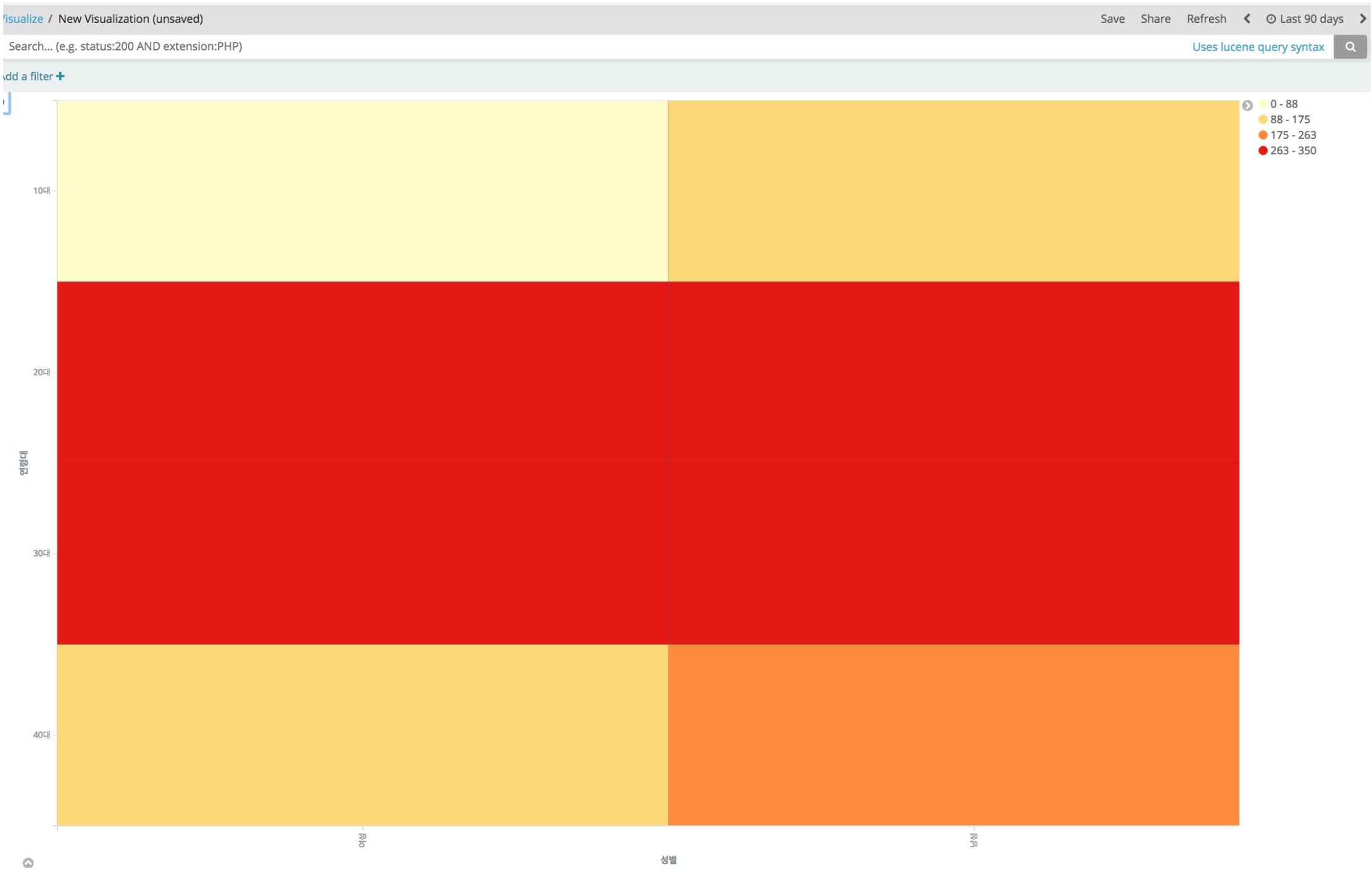
Visualize - Heat Map

Options (Heatmap Settings - Color Schema)를 통해 색상을 변경한다



Visualize - Heat Map

다음과 같은 Heat Map을 만들고 저장하자 (힌트는 다음 페이지) ✍



Visualize - Heat Map

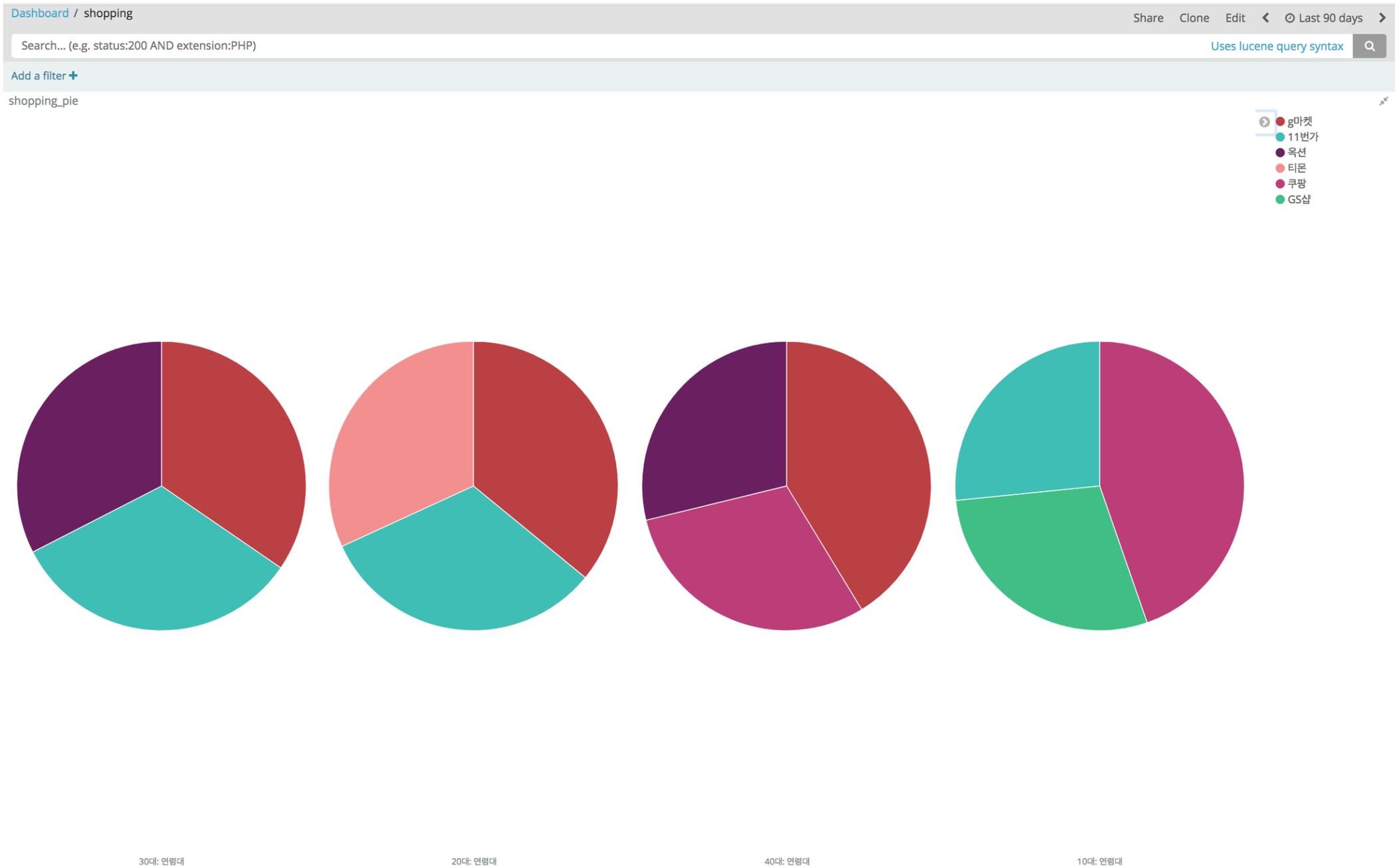
다음 조건을 만족하는 Heat Map을 만들어보자 ✍

- Index : shopping
- Time Range : Last 90 days
- metrics : Count
- buckets
  - X-Axis : “고객성별” Field를 기준으로 Split
  - Y-Axis : “연령대” Field를 기준으로 Split

Visualize - Pie Chart

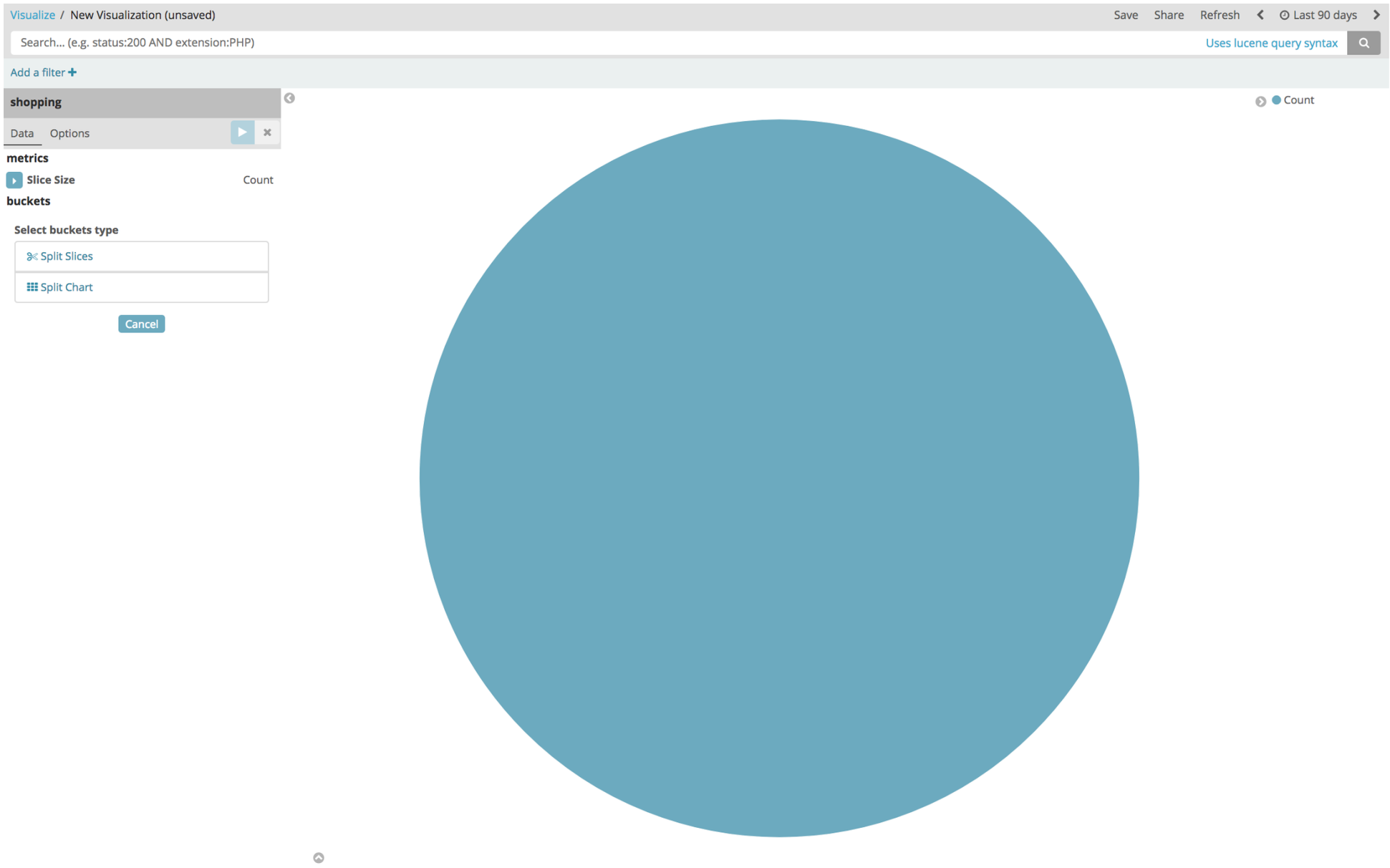
Visualize - Pie Chart

이런 Pie Chart를 만들어보자



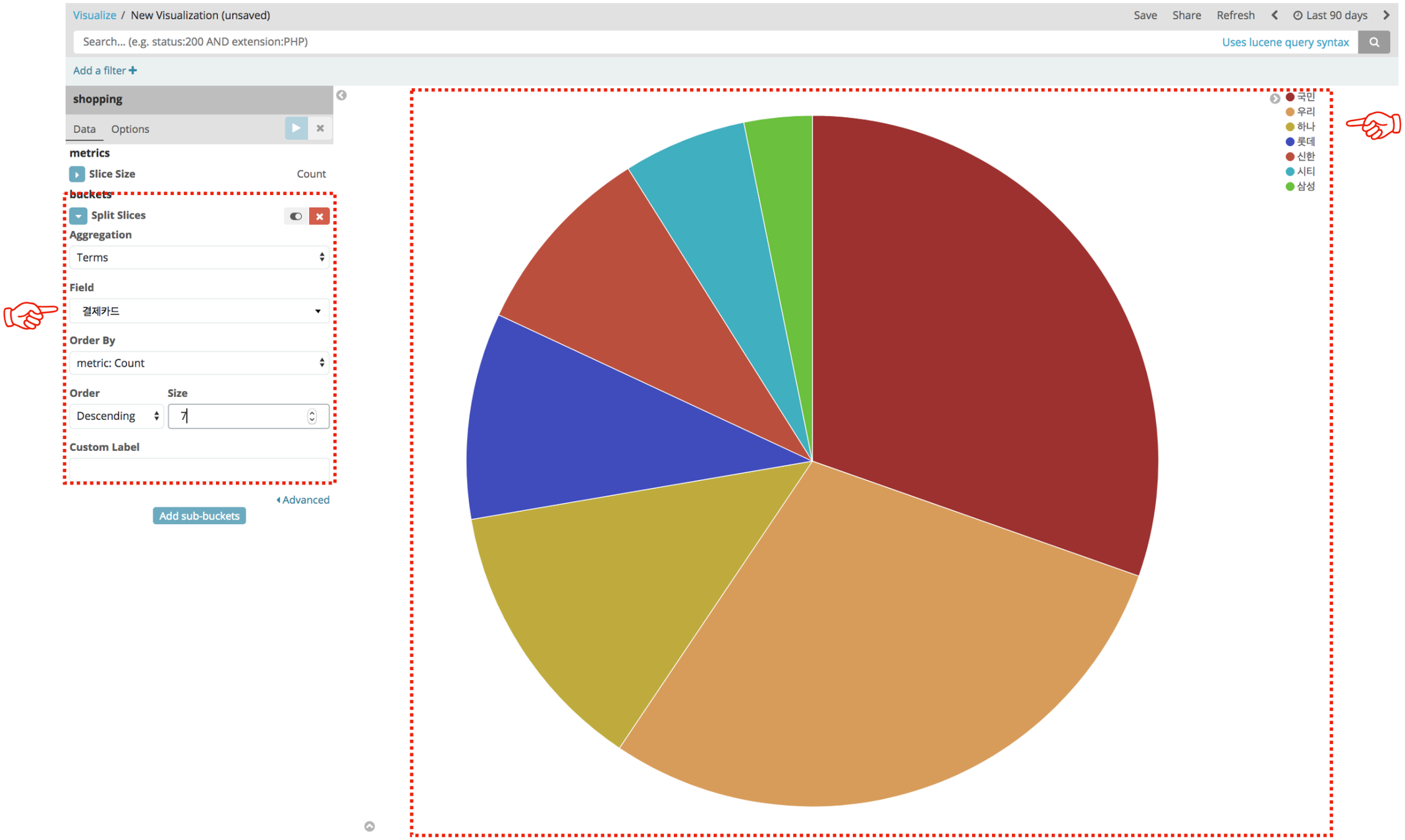
Visualize - Pie Chart

Pie Chart를 선택하면 처음에 이런 화면이 나온다



Visualize - Pie Chart

buckets (Split Charts)를 ~~(aggregation은 잘 모르겠지만)~~ 통해 여러 개의 Pie가 생긴다



Visualize - Pie Chart

Pie Chart 주의할 점

- Split Chart를 사용하는 경우, Split Chart를 Split Slices보다 앞에 둬야 한다
- 그리고 Split Chart와 Split Slices에 적당한 값을 넣은 후에 **한 번에** 실행해야 한다.

Pie chart response converter:

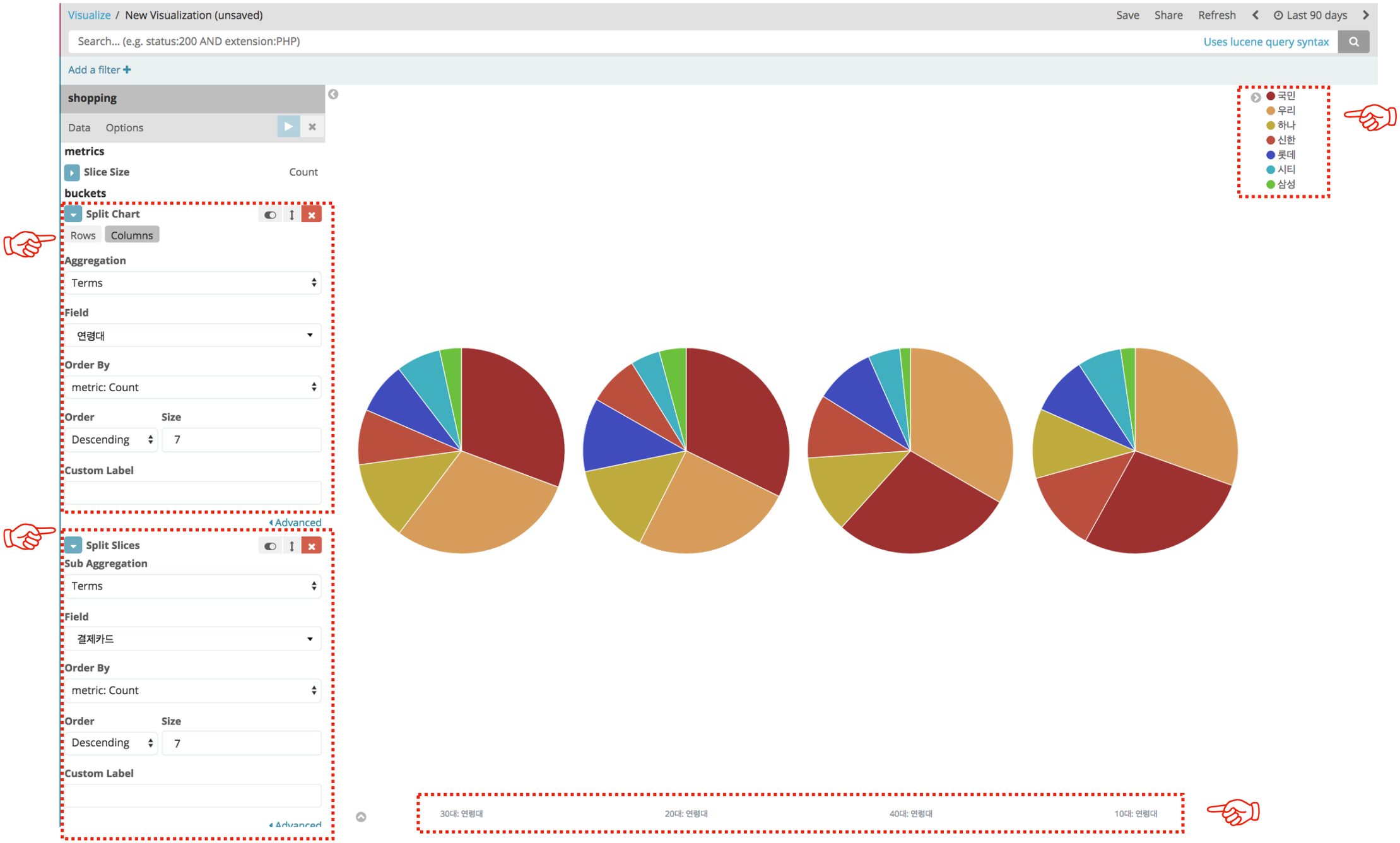
Splitting charts without splitting slices is not supported.

Pretending that we are just splitting slices.

Visualize - Pie Chart

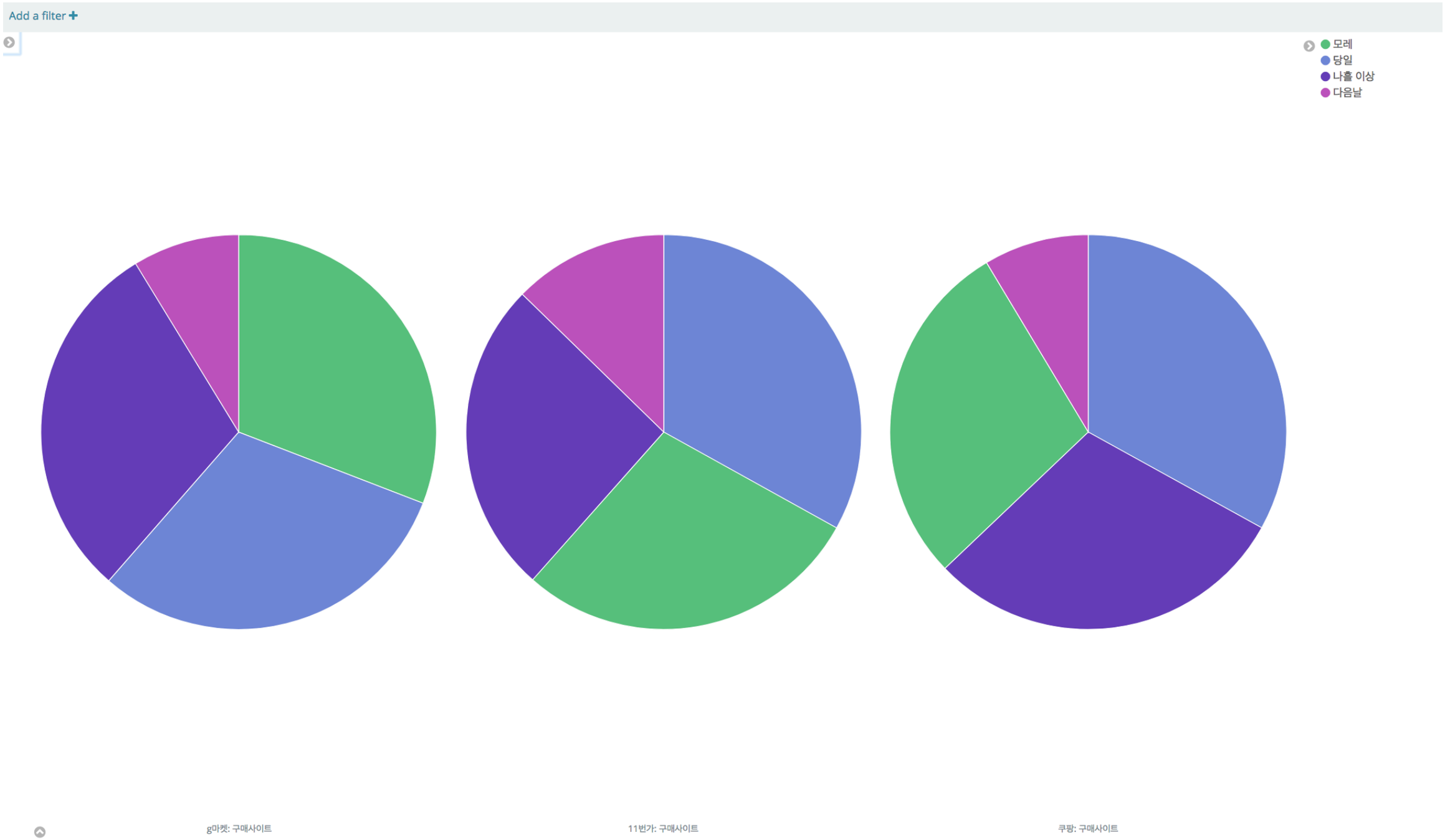
buckets (Split Chart)를 통해 Pie를 여러개 만들고,

buckets (Split Slices)를 통해 각 Pie 내부를 여러개로 나눈다



Visualize - Pie Chart

다음과 같은 Pie Chart를 만들고 저장하자 (힌트는 다음 페이지) ✍



Visualize - Pie Chart

다음 조건을 만족하는 Pie Chart를 만들어보자 ✍

- Index : shopping
- Time Range : Last 90 days
- metrics : Count
- buckets
  - Split Chart ：“구매사이트” Field를 기준으로 Pie를 Split
  - Split Series : “배송소요일수” Field를 기준으로 Split

Elasticsearch - Aggregation (Metric)

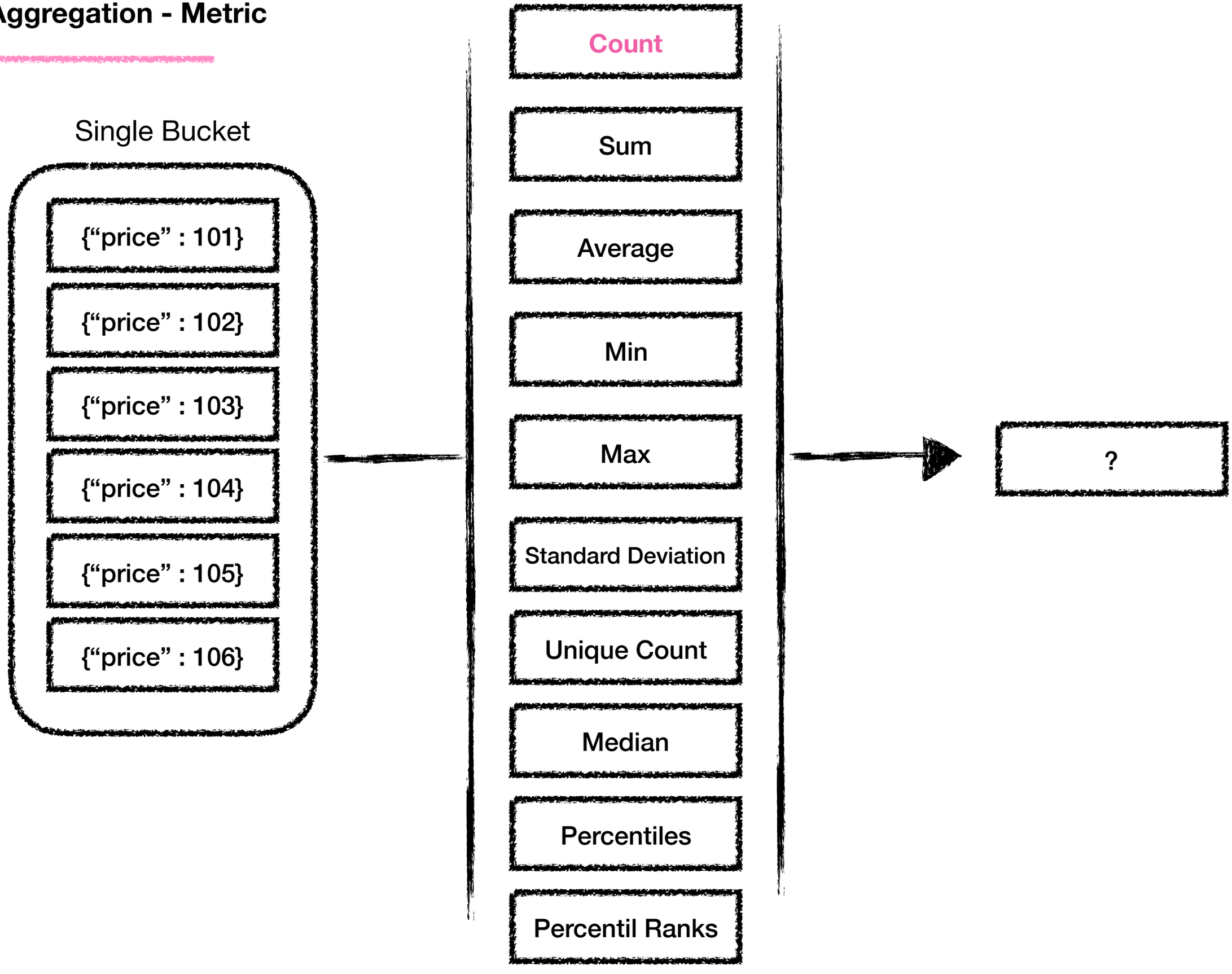
Aggregation - Metric

Metric Aggregation은 각 Bucket 내 특정 Field 값에 연산을 수행하는 Aggregation이다

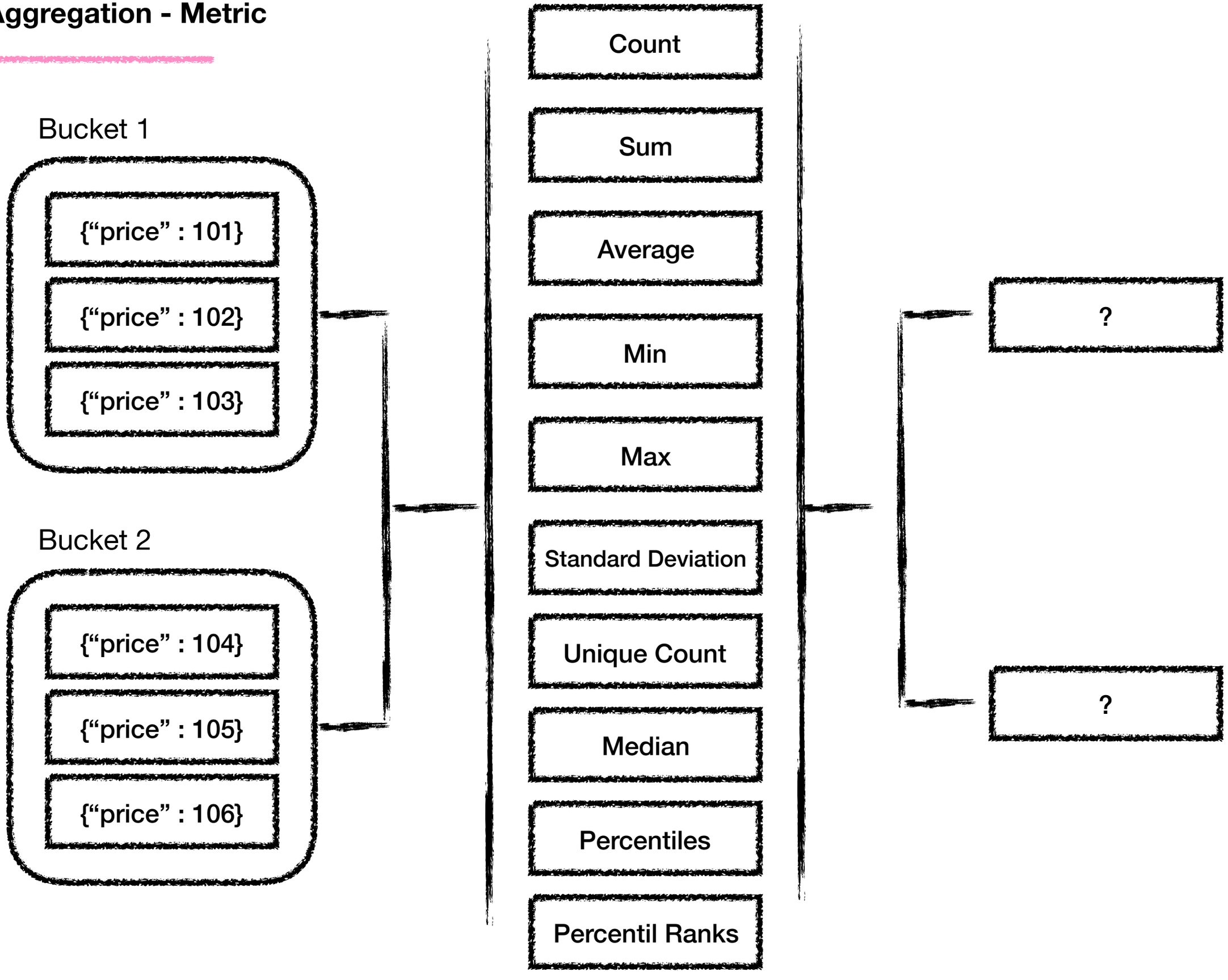
종류	적용 가능 Type	상세
Avg ♕	Number	(Bucket 내) Document의 특정 Field의 평균 계산
Sum ♕	Number	(Bucket 내) Document의 특정 Field의 합 계산
Min/Max ♕	Number	(Bucket 내) Document의 특정 Field의 최소/최대 계산
Extended Stats ♕	Number	(Bucket 내) Document의 특정 Field의 기초 통계값 계산
Cardinality ♕	Number	(Bucket 내) Document의 특정 Field의 고유한 개수 계산
Percentiles ♕	Number	(Bucket 내) Document의 특정 Field의 백분위수 계산
Percentiles Ranks ♕	Number	(Bucket 내) Document의 특정 Field의 백분위 계산
Top Hits ♕	All	(Bucket 내) 특정 조건을 만족하는 Documents의 특정 Field의 Agg 반환
Value Count ♕	All	(Bucket 내) Document의 개수 계산, Kibana 상에서 default metric aggreagtion

- Number Field : Concat, Sum, Min, Max, Count
- 기타 Field : Concat

Aggregation - Metric



Aggregation - Metric



Aggregation - Metric

{“번호” : 1, “날짜” : “10-01”, “역” : 강남 }

{“번호” : 2, “날짜” : “10-02”, “역” : 신사 }

{“번호” : 3, “날짜” : “10-03”, “역” : 역삼 }

{“번호” : 4, “날짜” : “10-10”, “역” : 송내 }

{“번호” : 5, “날짜” : “10-05”, “역” : 선릉 }

{“번호” : 6, “날짜” : “10-06”, “역” : 언주 }

{“번호” : 7, “날짜” : “10-07”, “역” : 잠원 }

{“번호” : 8, “날짜” : “10-08”, “역” : 시청 }

Top Hits Aggregation

날짜가	빠른	데이터 3개의	번호	합을 구하세요
번호가	작은	데이터 2개의	역명	모두 나열하세요
역명이	빠른	데이터 2개의	번호	평균을 구하세요

Elasticsearch - Aggregation (Bucket)

Aggregation - Bucket

Bucket Aggregation은 데이터를 **공통 속성을 가진 그룹으로** 묶는 Aggregation이다

종류	적용 가능 Type	상세
Date ♕	Date	날짜/시간을 일정하게 지정하여 구간 내의 Documents로 Bucket 형성
Date Range ♕	Date	날짜/시간을 임의로 지정하여 구간 내의 Documents로 Bucket 형성
Histogram ♕	Number	범위를 일정하게 지정하여 구간 내의 Documents로 Bucket 형성
Range ♕	Number	범위를 임의로 지정하여 구간 내의 Documents로 Bucket 형성
Terms ♕	Date, IP, Number, String	특정 Field 값을 기준으로 Bucket 생성 (카테고리 데이터에 유용)
Significant Terms ♕	String	Background 대비 Foreground에서 특별한 값으로 Bucket 생성
Filters ♕	Date, IP, Number, String	직접 조건을 입력하여 Bucket 생성 (조건 개수만큼 Bucket 생성)
Geo Hash ♕	Geo Point	특정 지점 (Centroid) 근처에 있는 값들을 모아서 Bucket 생성
IPv4 Range ♕	IP	IP 주소의 범위로 Bucket 생성

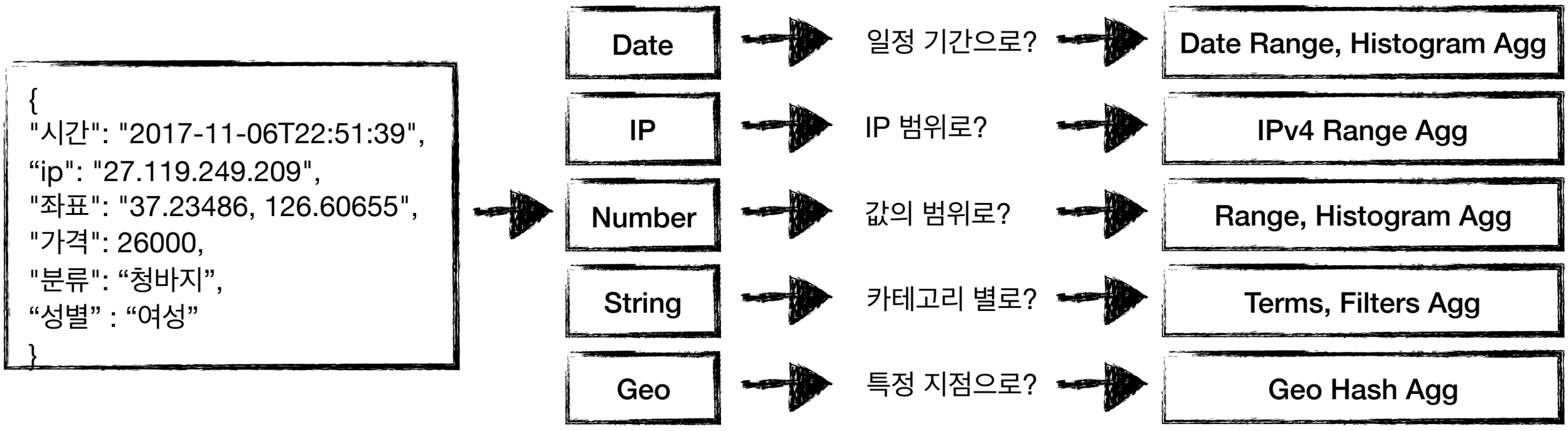
Aggregation - Bucket

- 아래와 같은 데이터가 있다고 하자
- 어떤 기준으로 Bucket을 생성할 수 있을까?

{
"시간": "2017-11-06T22:51:39",
“ip": "27.119.249.209",
"좌표": "37.23486, 126.60655",
"가격": 26000,
"분류": “청바지”,
“성별” : “여성”
},
{
"시간": "2018-01-06T13:51:39",
“ip": "27.119.249.209",
"좌표": "37.2299, 126.59903",
"가격": 35000,
"분류": “니트”,
“성별” : “남성”
},

⋮

Aggregation - Bucket



Elasticsearch - Aggregation (Parent Pipeline)

Aggregation - Parent Pipeline 👑

Parent Pipeline Aggregation은

- Date Bucket Aggregation 후,
- 각 Bucket 내 Metric Aggregation을 적용하고
- 그 **Bucket 값들 간에**

적용하는 Aggregation이다

종류	설명
Derivative 👑	Date Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, 연속한 Bucket 값들의 차이를 구함
Cumulative Sum 👑	Date Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, Bucket 값들의 누적합을 구함
Moving Average 👑	Date Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, 연속한 {n개} Bucket 값들의 평균을 구함
Serial Diff 👑	Date Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, {n번째 이전} Bucket 과의 차이를 구함

Aggregation - Parent Pipeline

다음과 같은 데이터가 있다고 하자

bucket 1

```
{
"시간": "2018-01-14",
“데이터" : 1
}
```

```
{
"시간": "2018-01-14",
“데이터" : 2
}
```

```
{
"시간": "2018-01-14",
“데이터" : 3
}
```

bucket 2

```
{
"시간": “2018-01-15",
“데이터" : 4
}
```

```
{
"시간": "2018-01-15",
“데이터" : 5
}
```

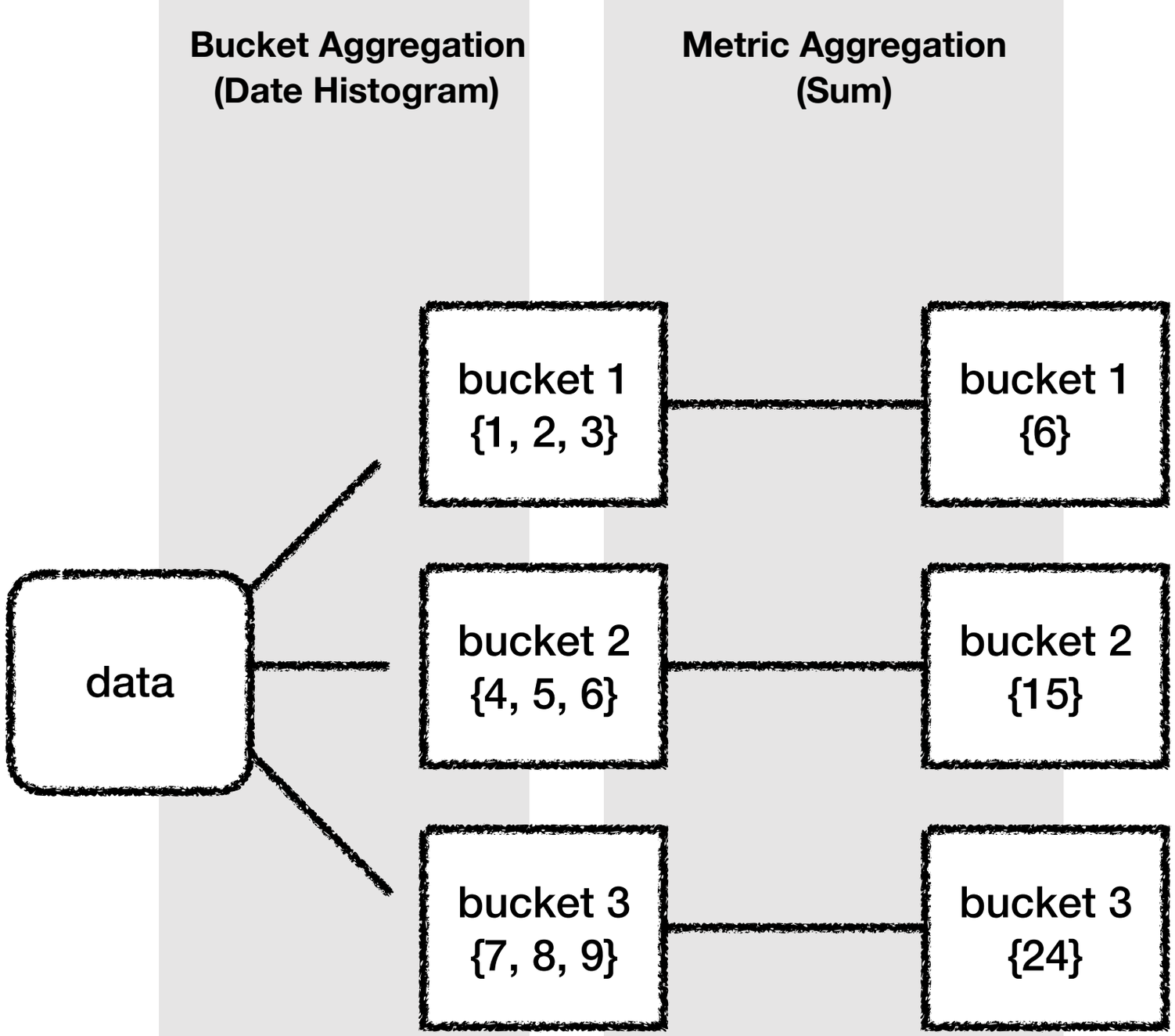
```
{
"시간": “2018-01-15",
“데이터" : 6
}
```

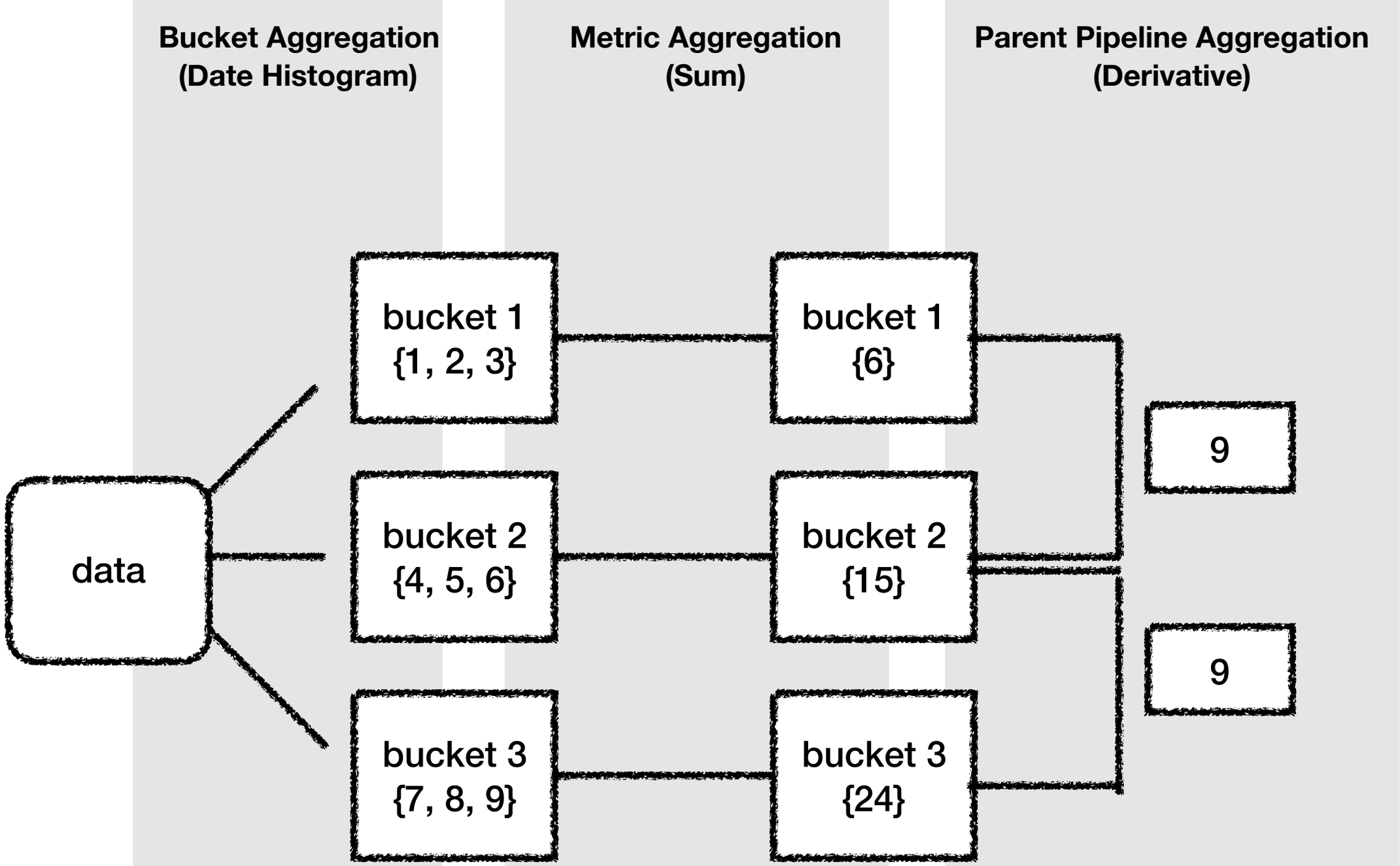
bucket 3

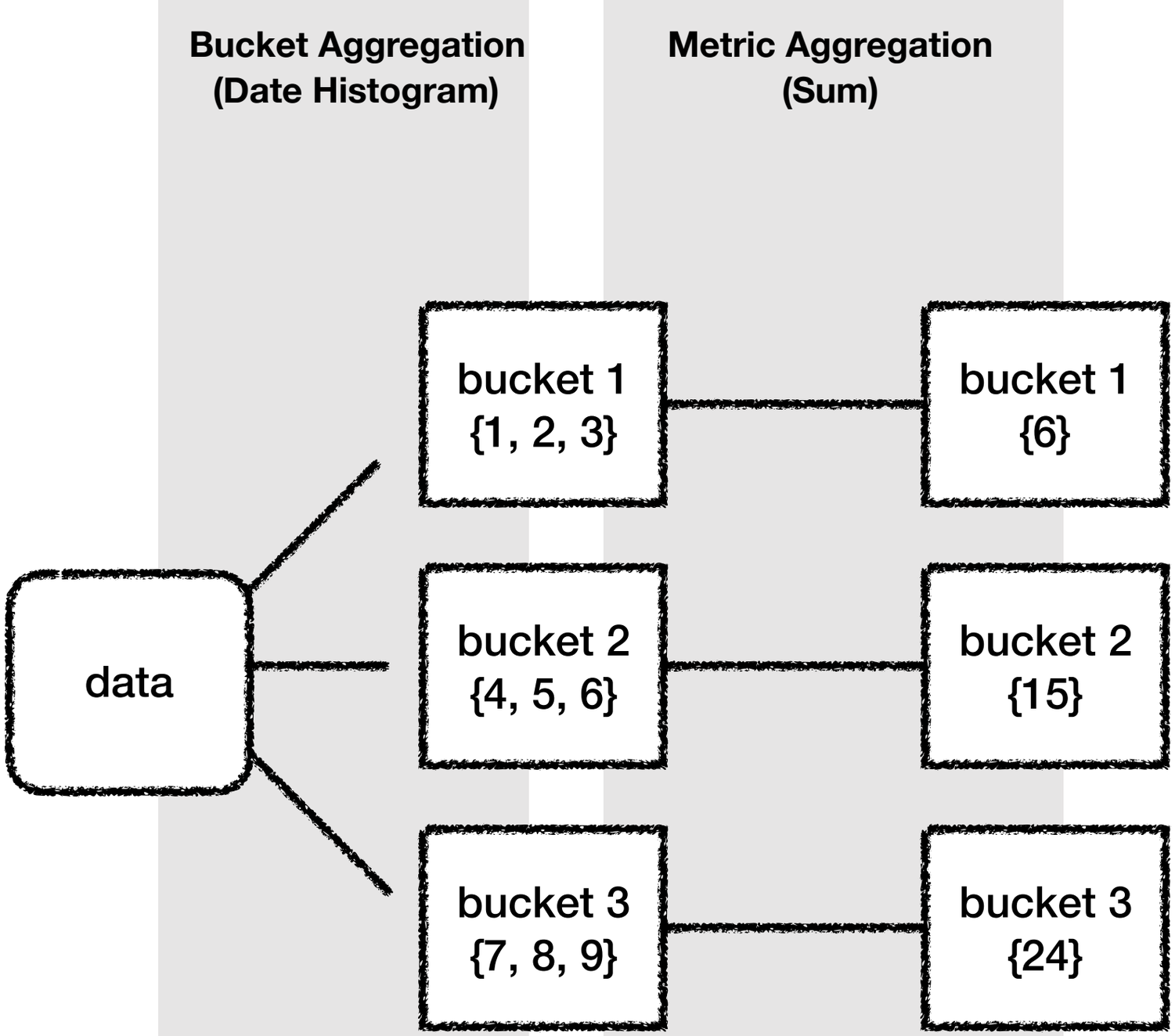
```
{
"시간": "2018-01-16",
“데이터" : 7
}
```

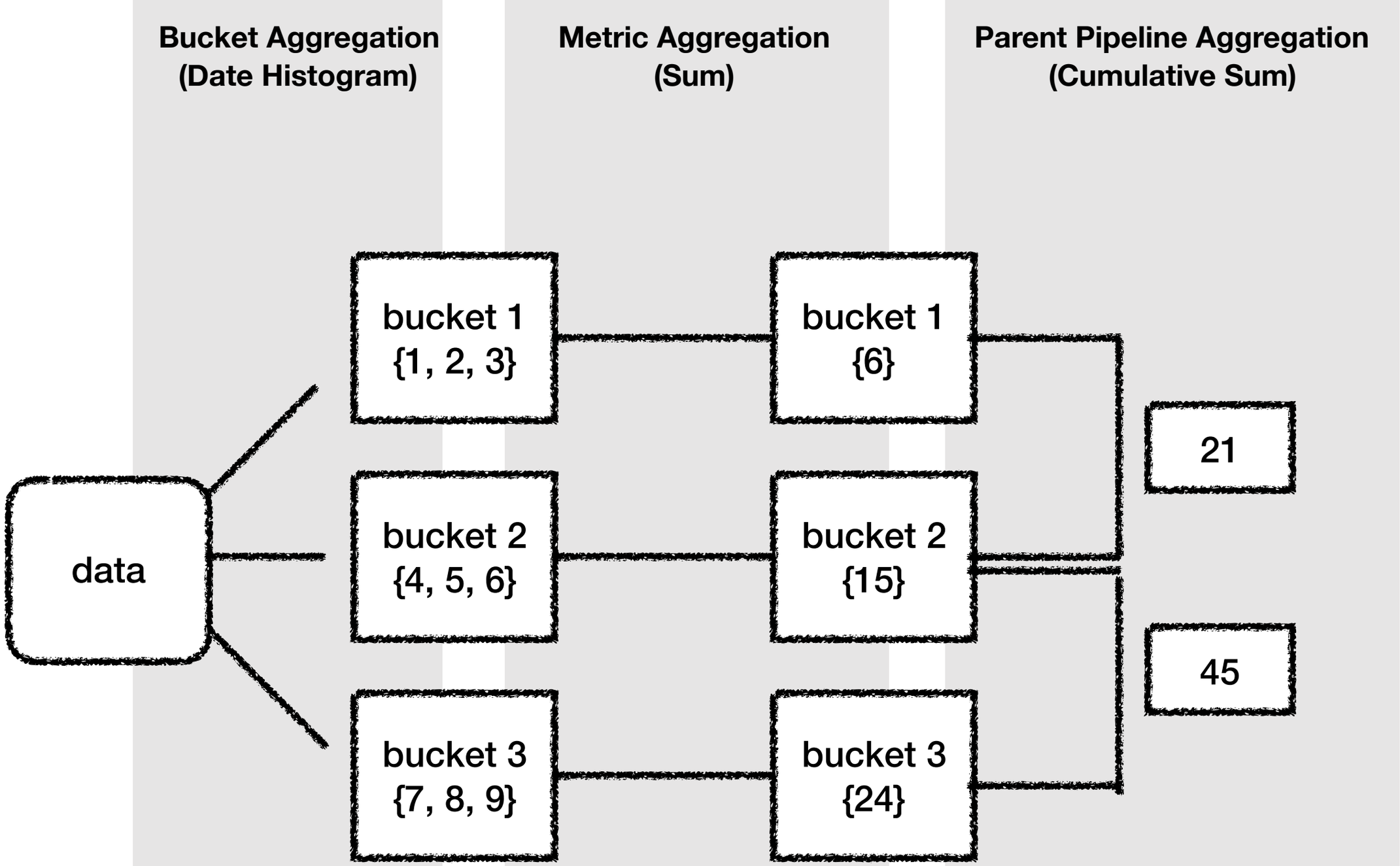
```
{
"시간": “2018-01-16",
“데이터" : 8
}
```

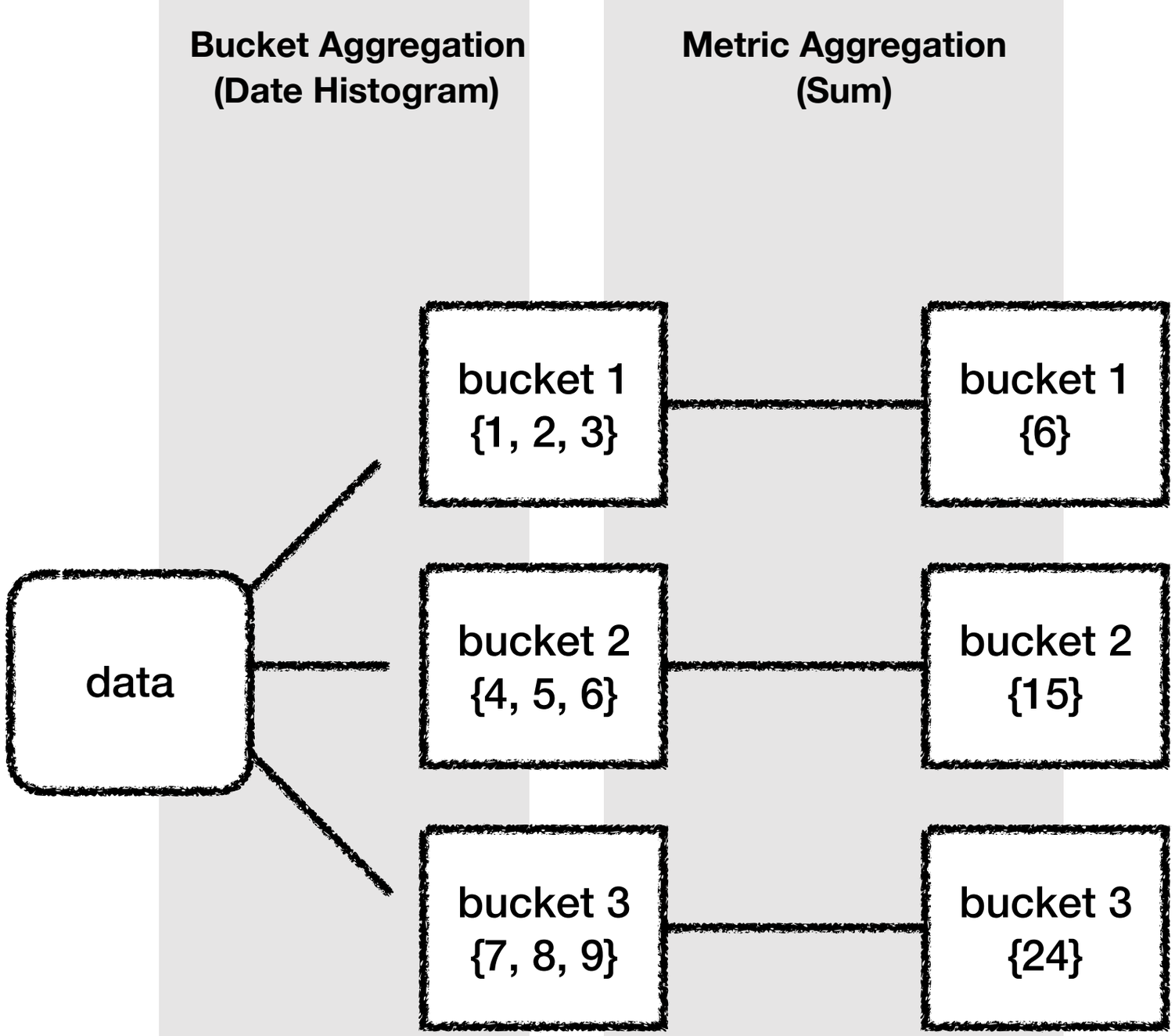
```
{
"시간": “2018-01-16",
“데이터" : 9
}
```

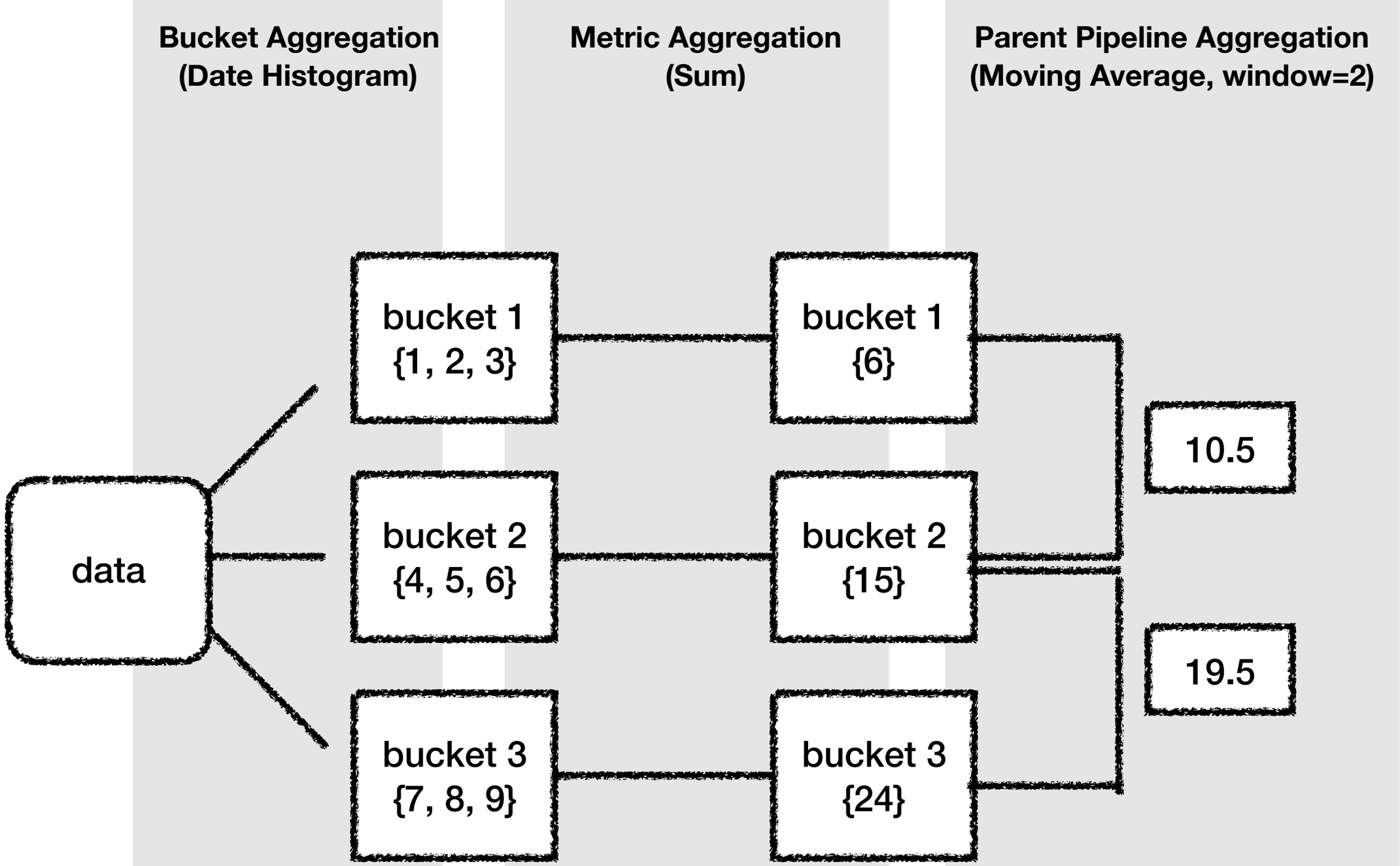

Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}

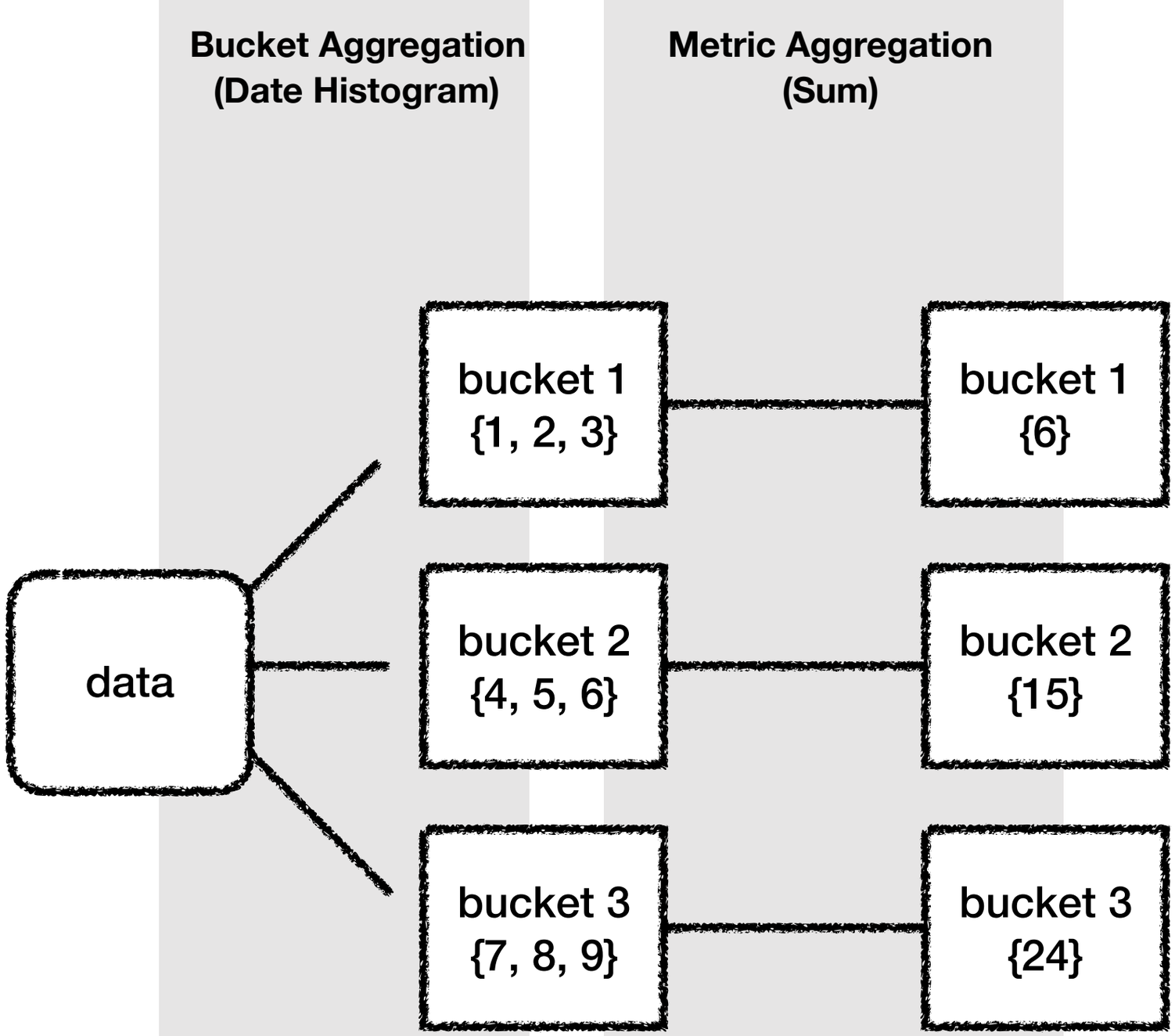

Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Derivative)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
9
9

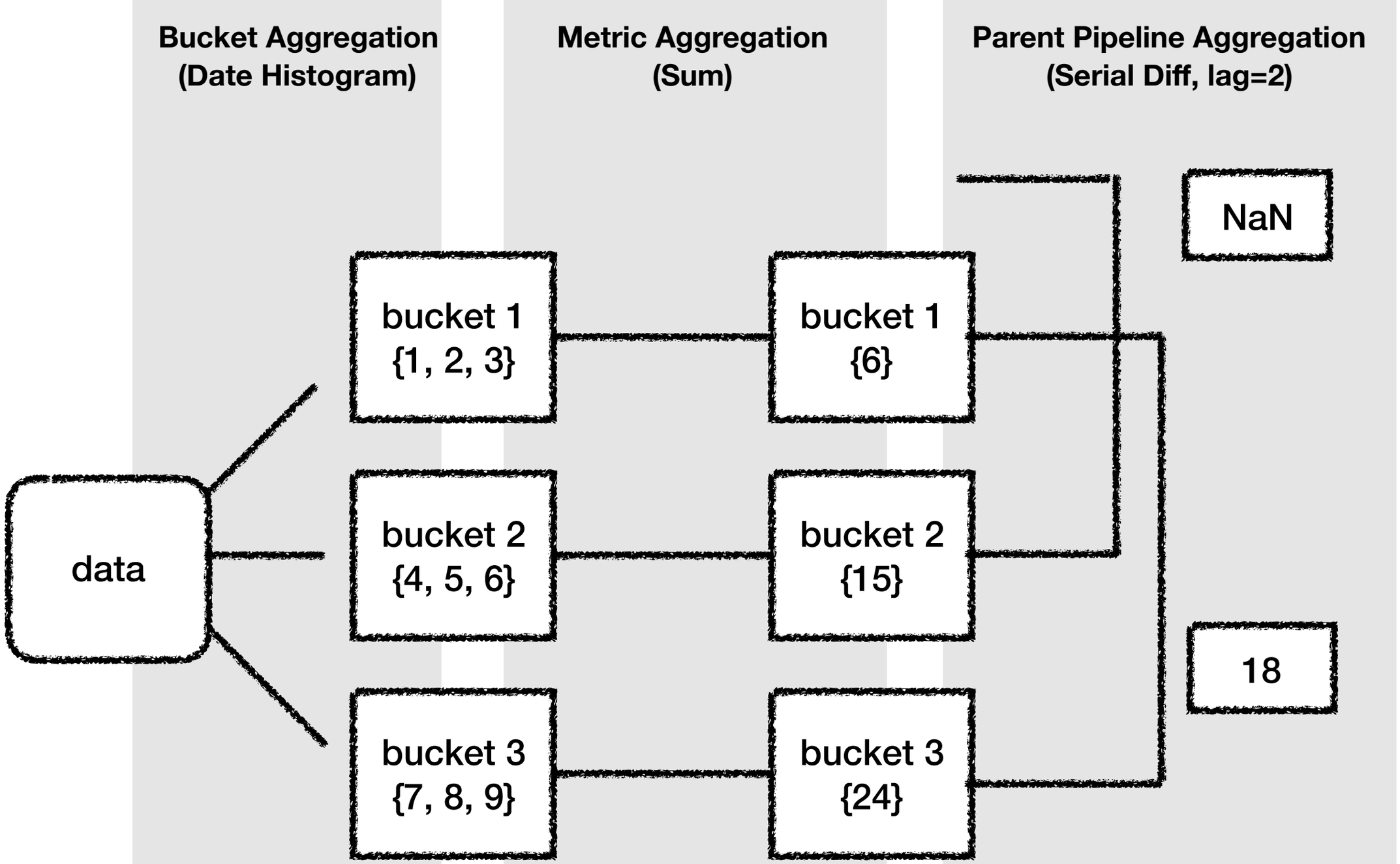

Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Cumulative Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
21
45


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Moving Average, window=2)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
10.5
19.5


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Serial Diff, lag=2)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
NaN
18

Elasticsearch - Aggregation (Sibling Pipeline)

Aggregation - Sibling Pipeline 👑

Sibling Pipeline Aggregation은

- Date Bucket Aggregation 후,
- 각 Bucket 내 Metric Aggregation을 적용하고
- **그 결과에 공통적으로**

적용하는 Aggregation이다

종류	설명
Min Bucket 👑	Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, Min Aggregation 적용
Max Bucket 👑	Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, Max Aggregation 적용
Sum Bucket 👑	Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, Sum Aggregation 적용
Average Bucket 👑	Bucket Agg 후, Bucket 내 Metric Agg 하고 난 후, Average Aggregation 적용

Aggregation - Sibling Pipeline

다음과 같은 데이터가 있다고 하자

bucket 1

```
{
"시간": "2018-01-14",
“데이터" : 1
}
```

```
{
"시간": "2018-01-14",
“데이터" : 2
}
```

```
{
"시간": "2018-01-14",
“데이터" : 3
}
```

bucket 2

```
{
"시간": “2018-01-15",
“데이터" : 4
}
```

```
{
"시간": "2018-01-15",
“데이터" : 5
}
```

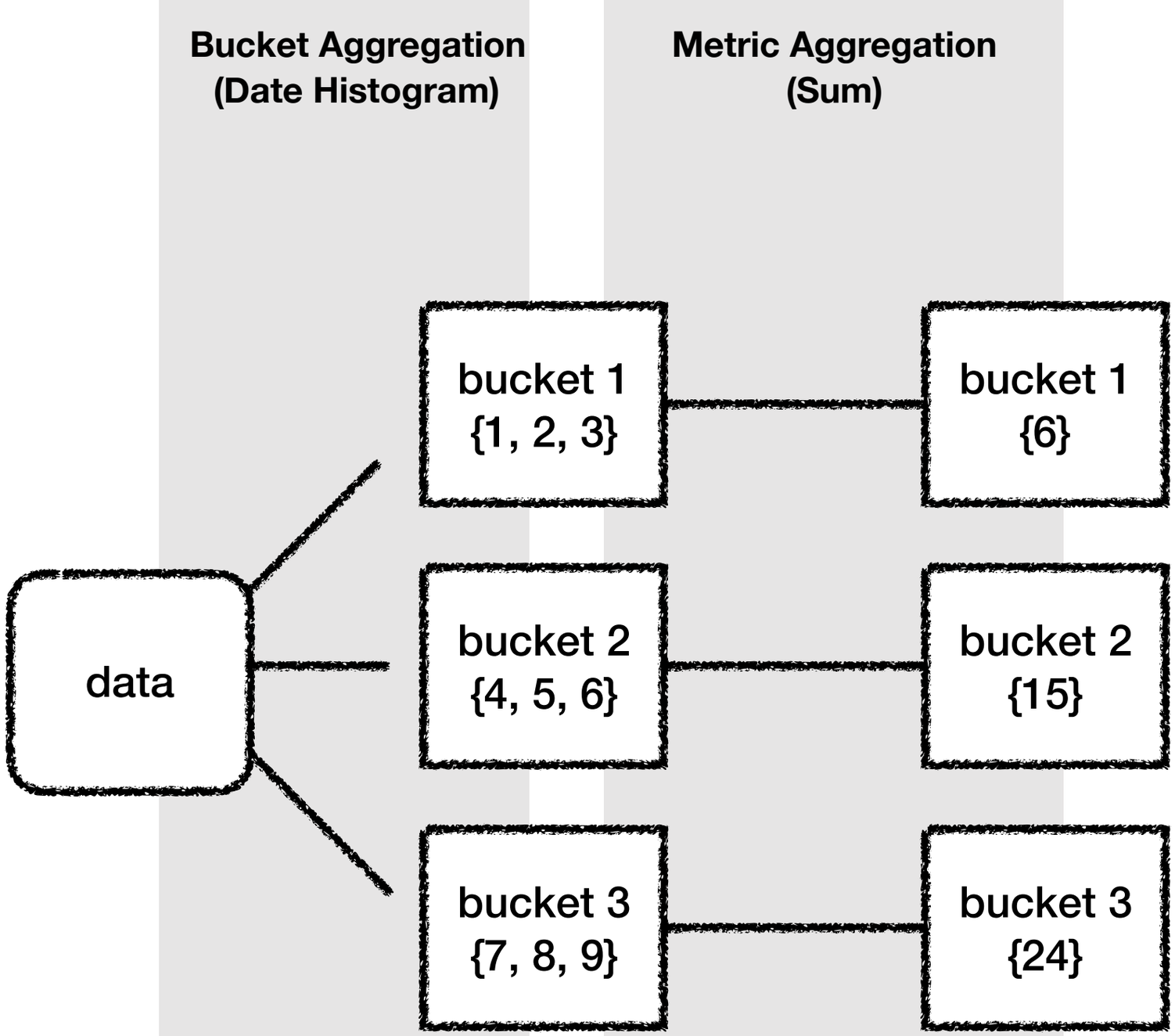
```
{
"시간": “2018-01-15",
“데이터" : 6
}
```

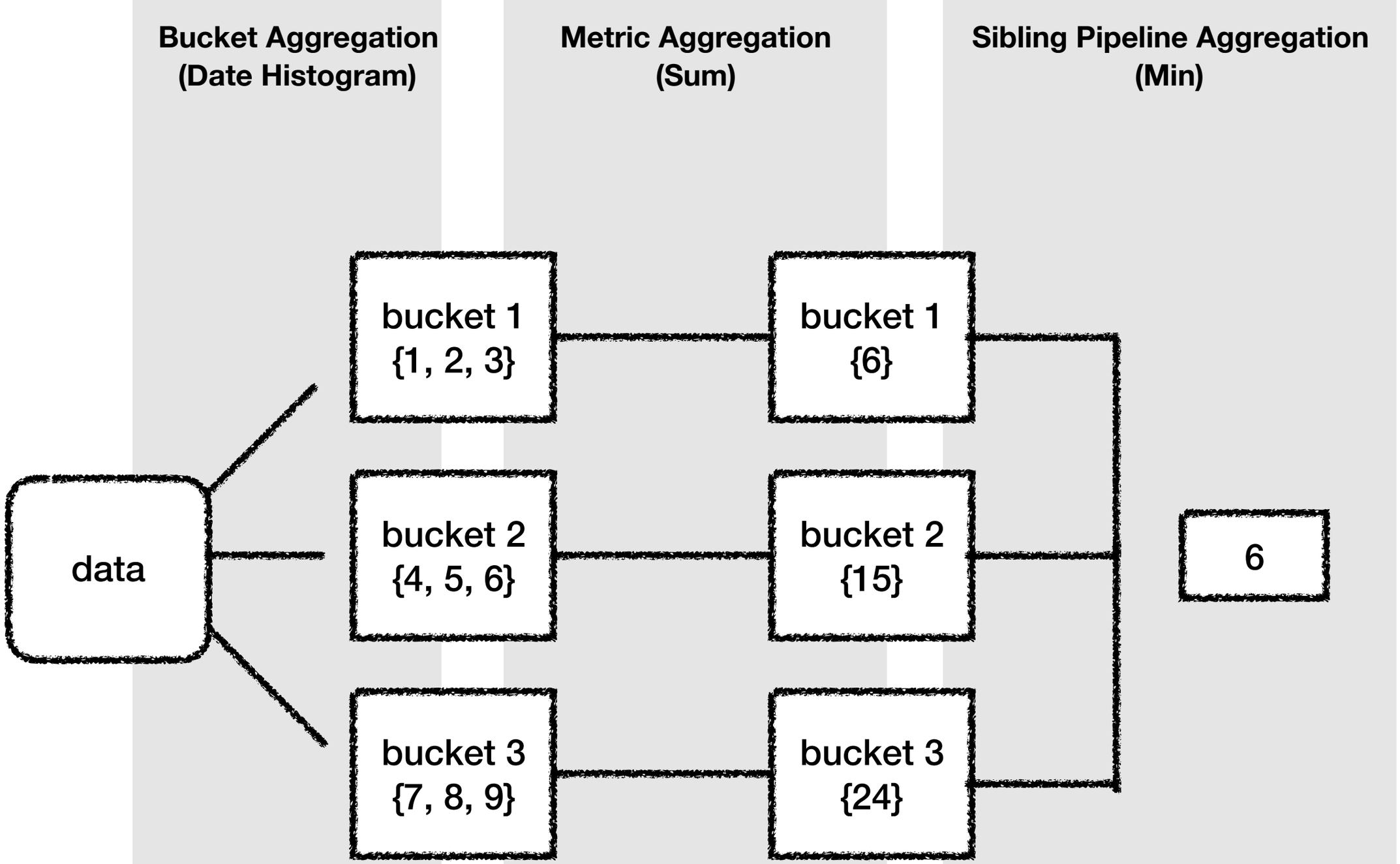
bucket 3

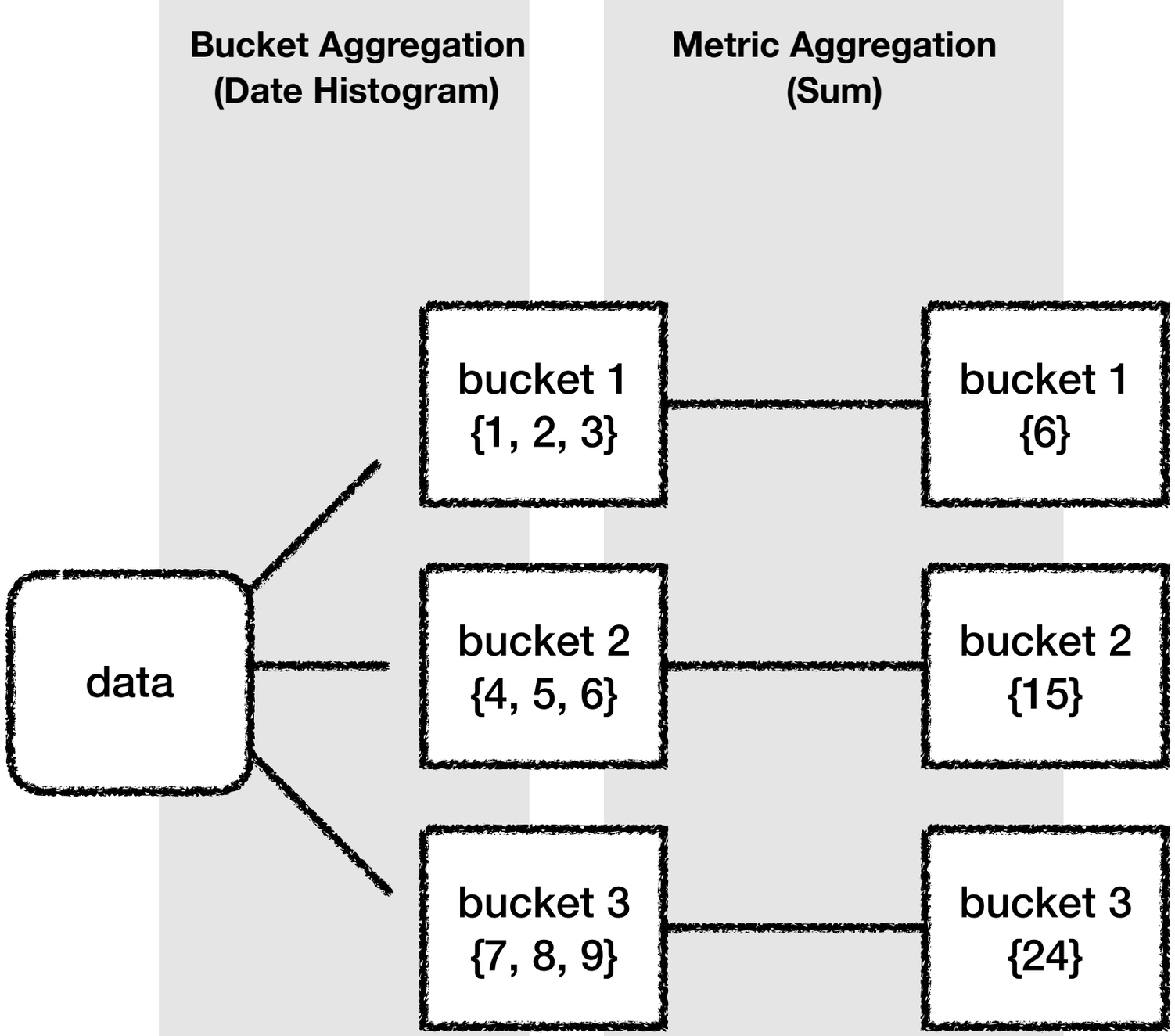
```
{
"시간": "2018-01-16",
“데이터" : 7
}
```

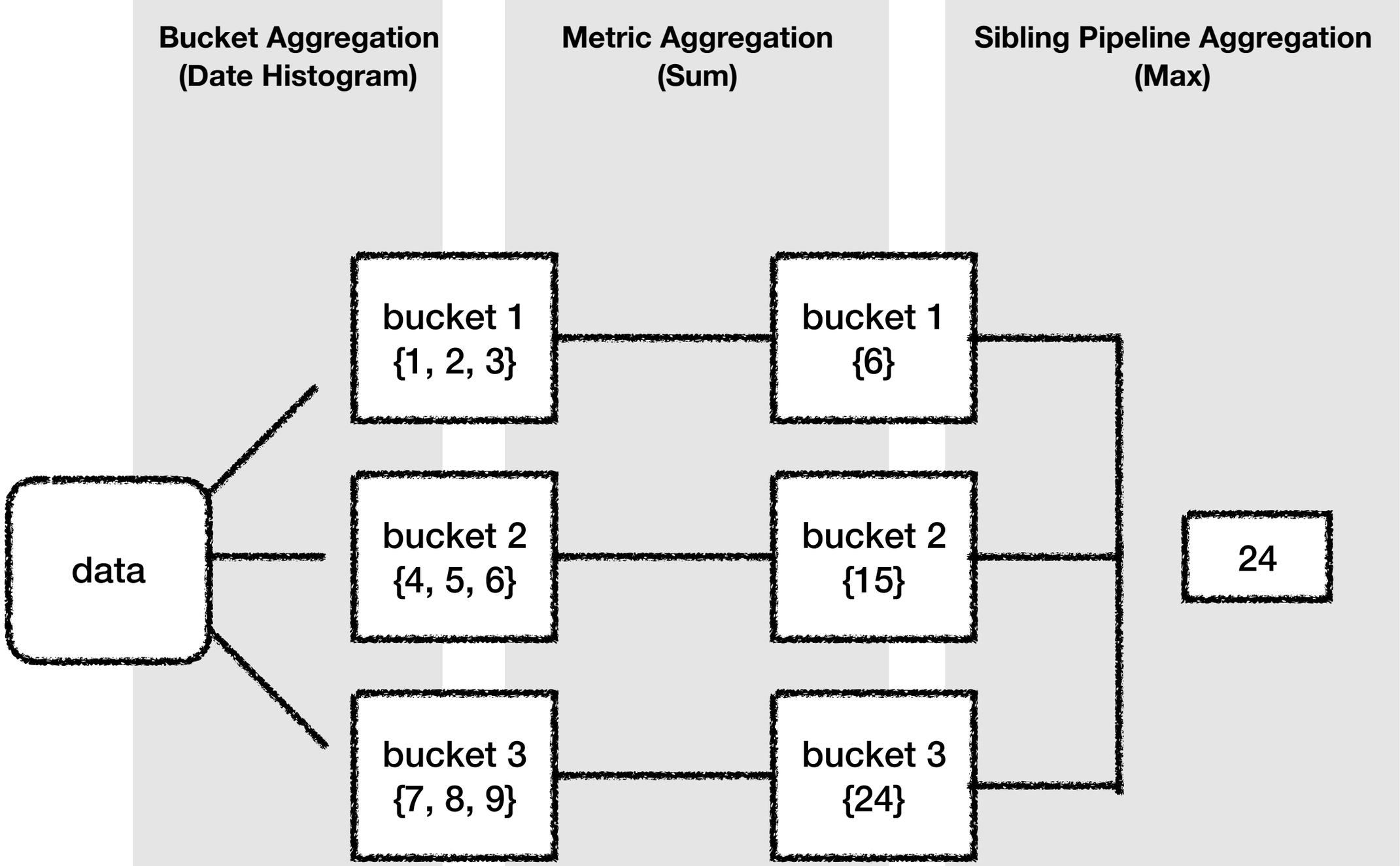
```
{
"시간": “2018-01-16",
“데이터" : 8
}
```

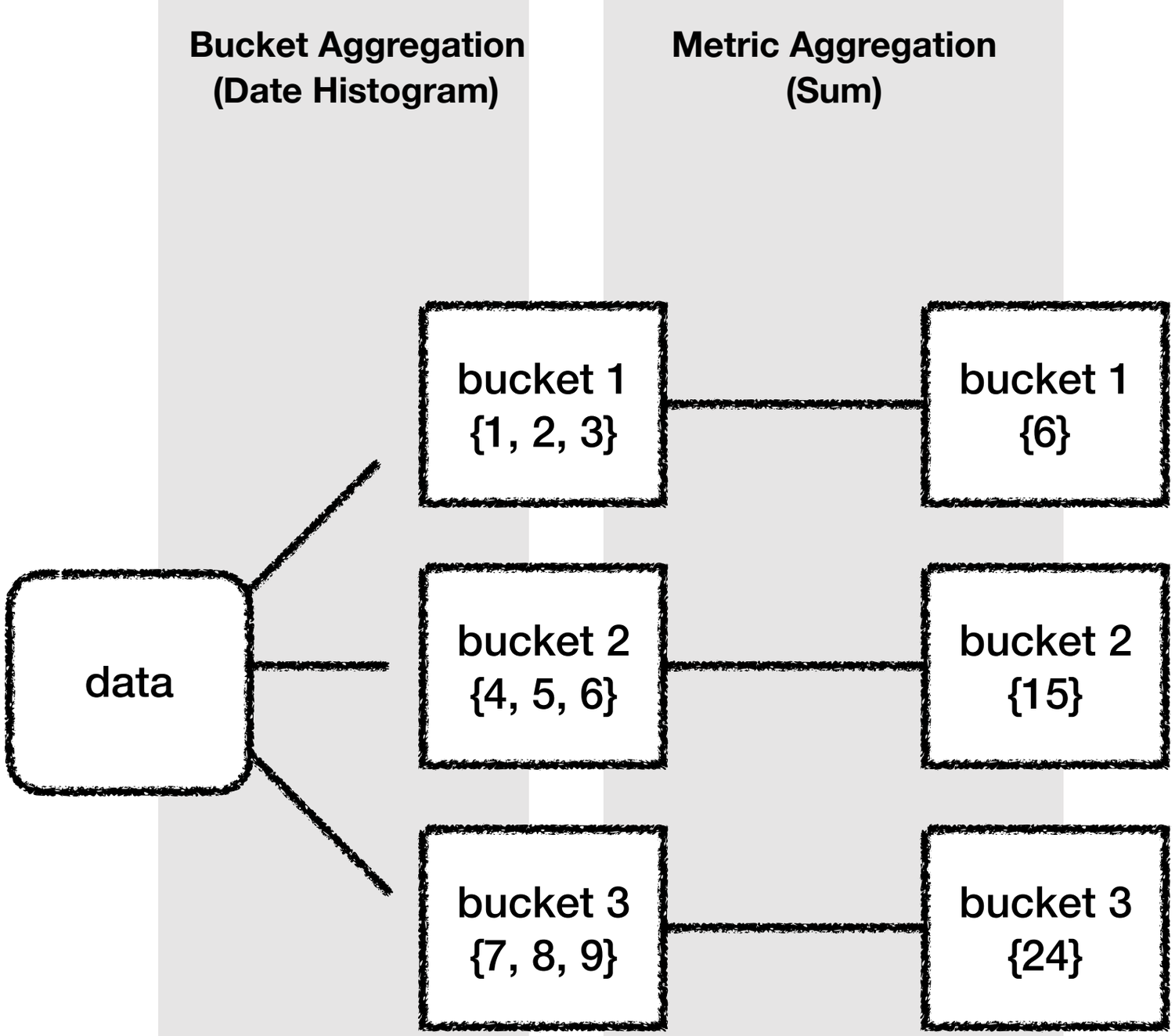
```
{
"시간": “2018-01-16",
“데이터" : 9
}
```

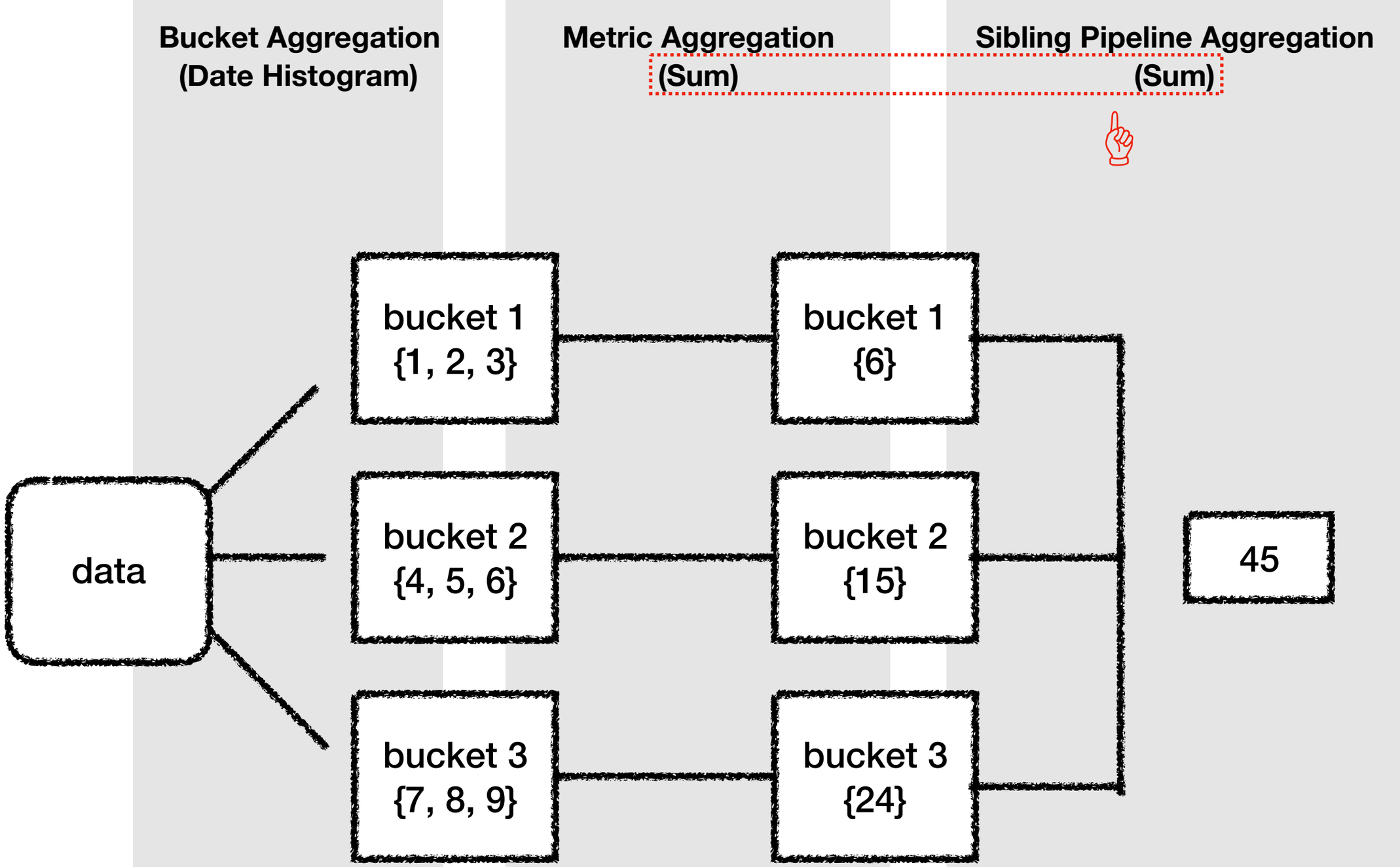

Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}

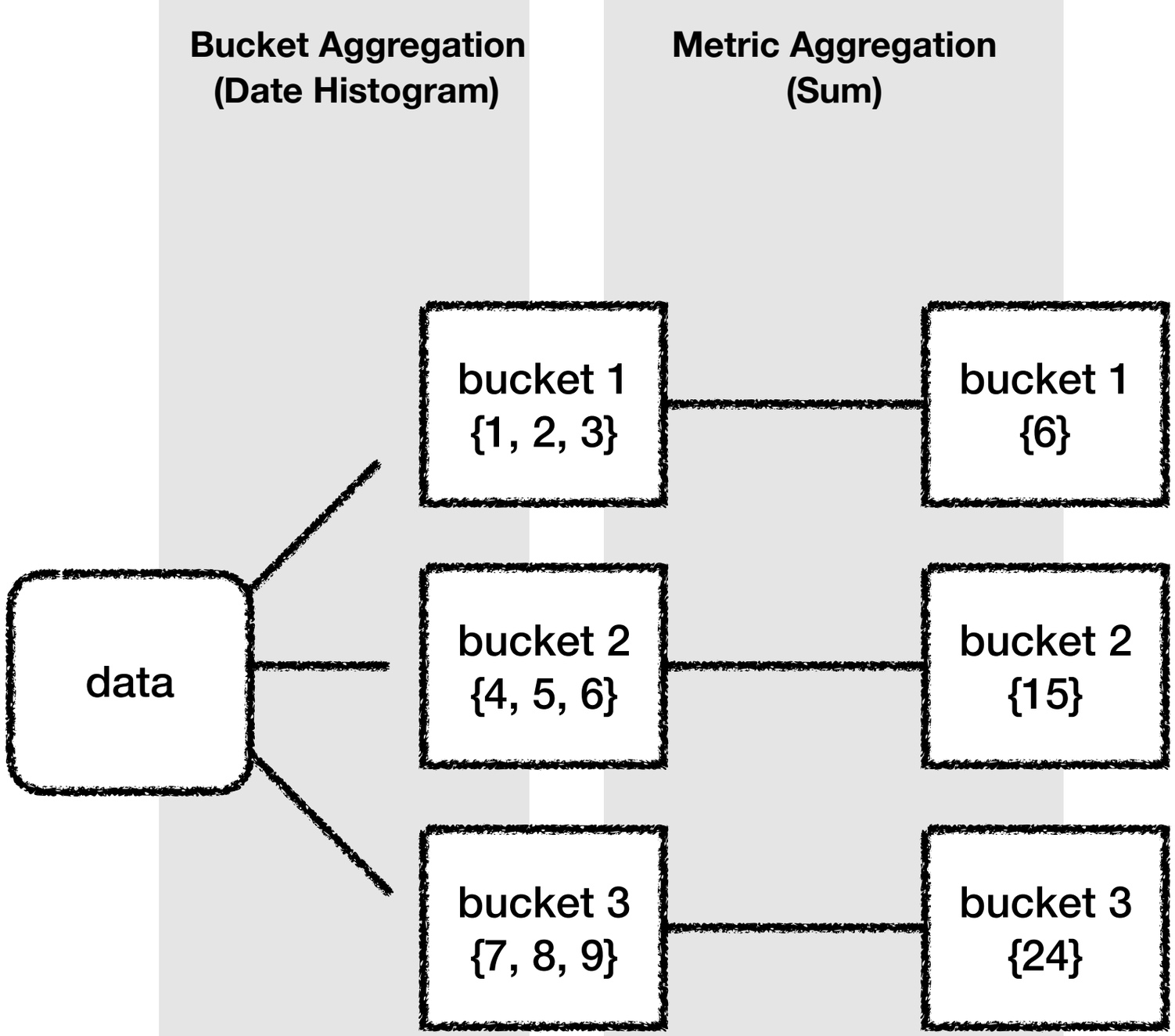

Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Min)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
6

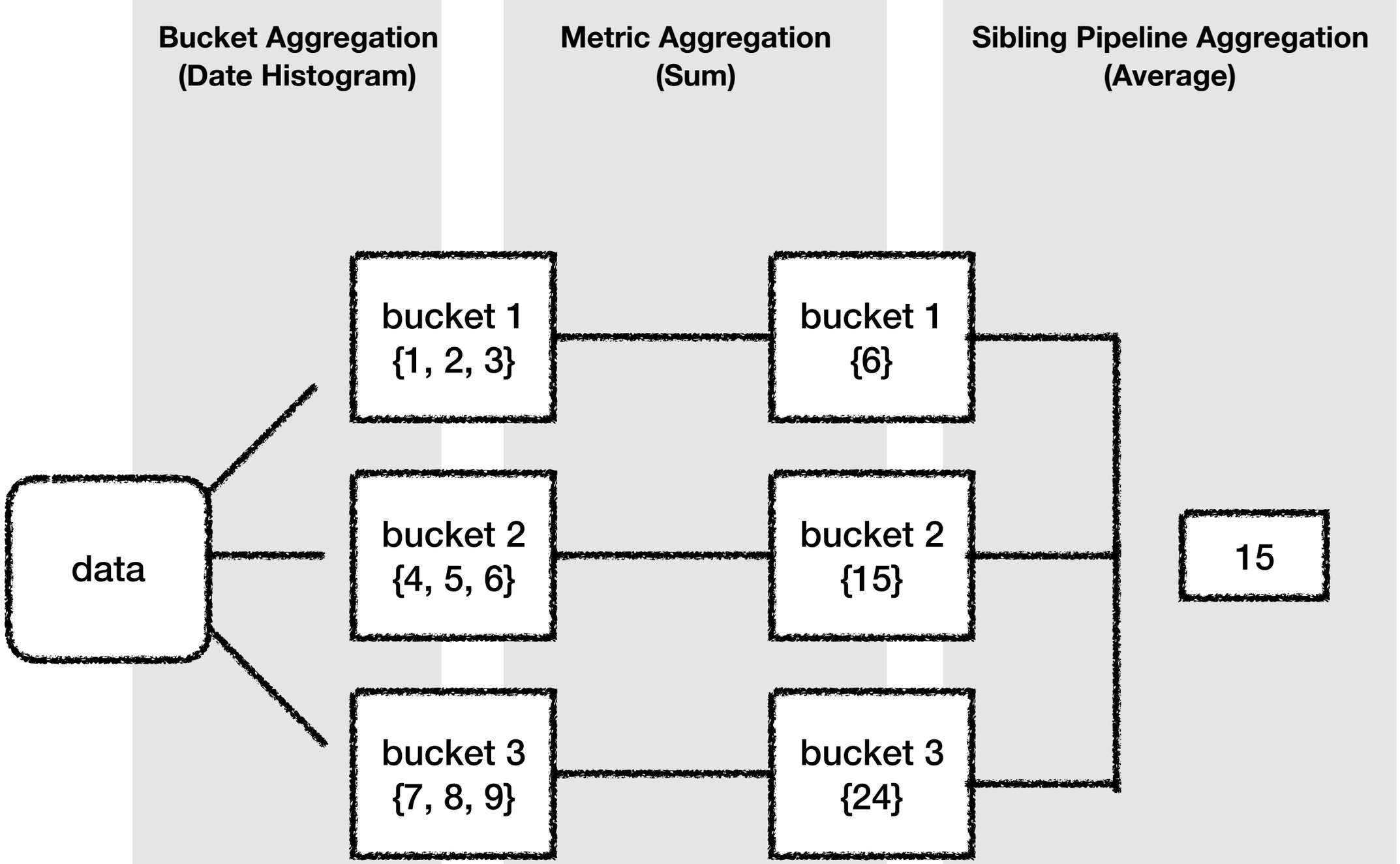

Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Max)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
24


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
45


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Average)
data
bucket 1
{1, 2, 3}
bucket 2
{4, 5, 6}
bucket 3
{7, 8, 9}
bucket 1
{6}
bucket 2
{15}
bucket 3
{24}
15

Visualize - Metric

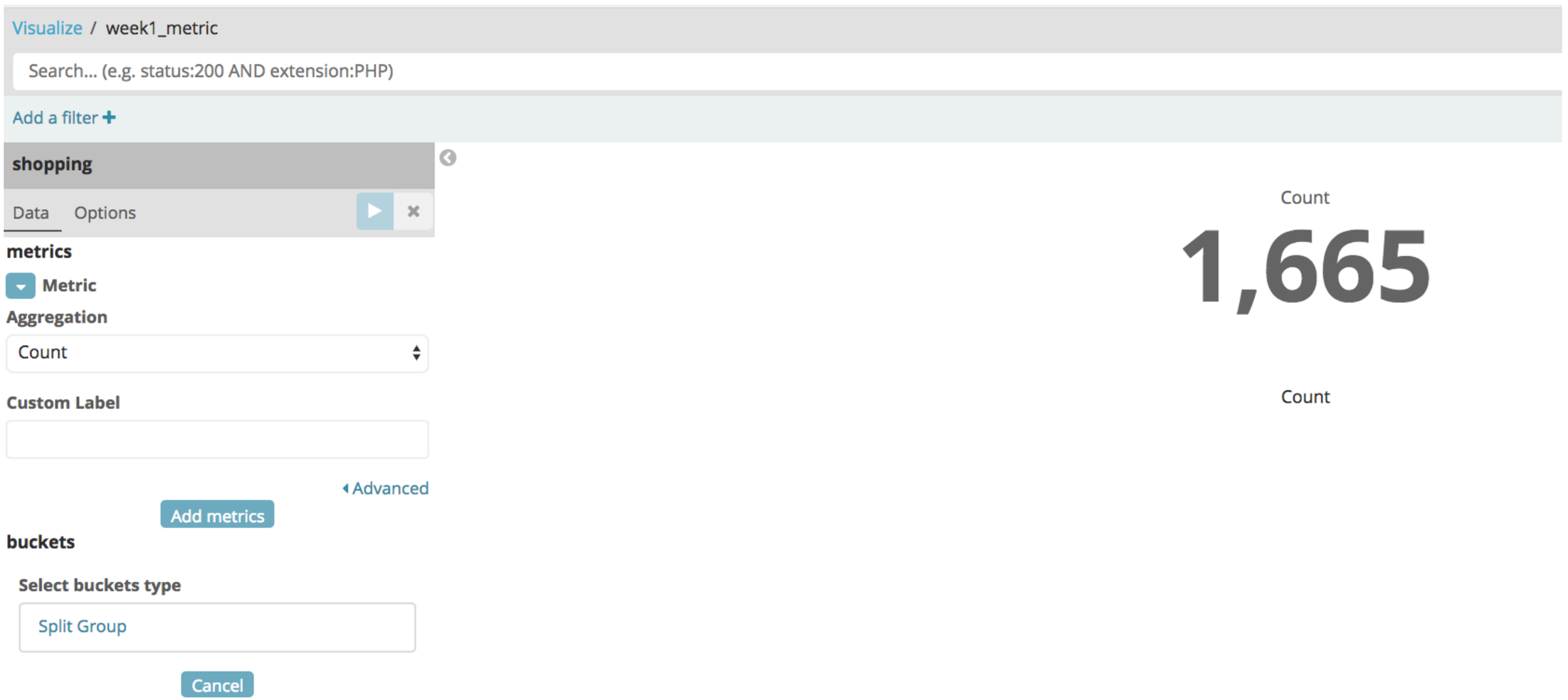
Visualize - Metric

아래 조건을 만족하는 Metric을 만들어보자 ✍

- shopping Index에서
- 최근 90일 동안
- “상품개수” 의 합이 가장 많은 구매사이트 3개의
- “상품개수”의 합을 각각 표시

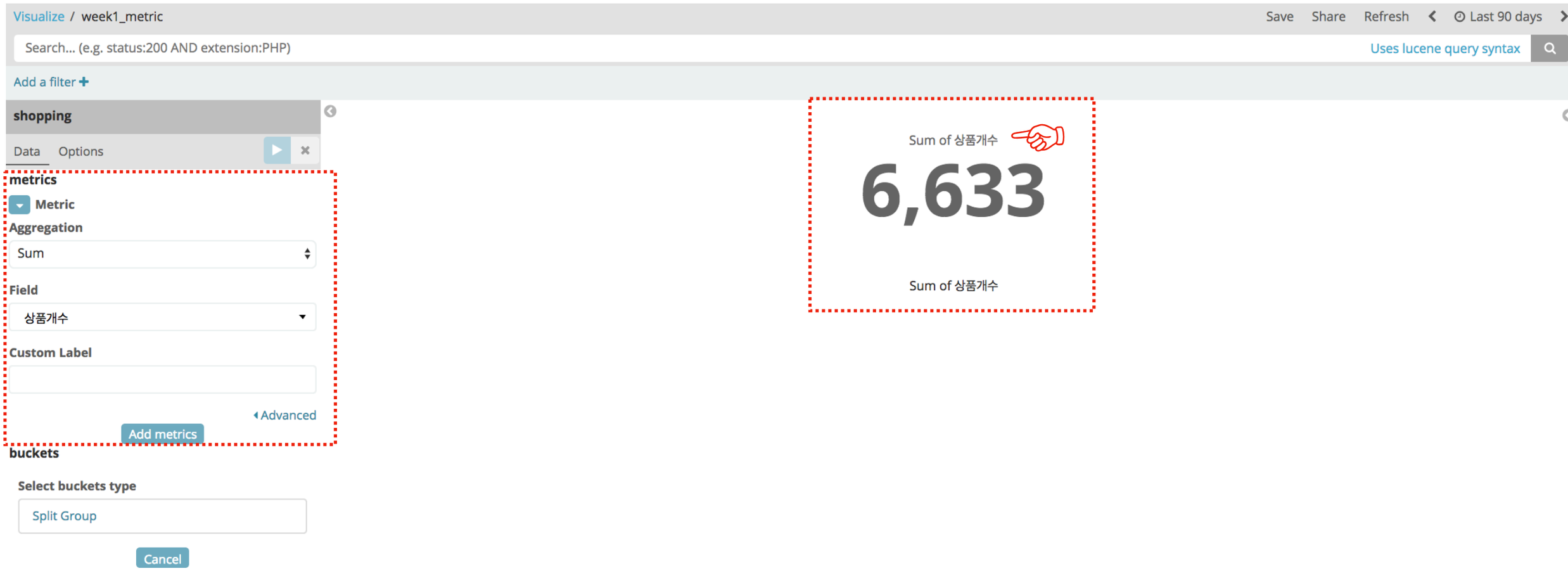
Visualize - Metric

Metric를 선택하면 처음에 이런 화면이 나온다

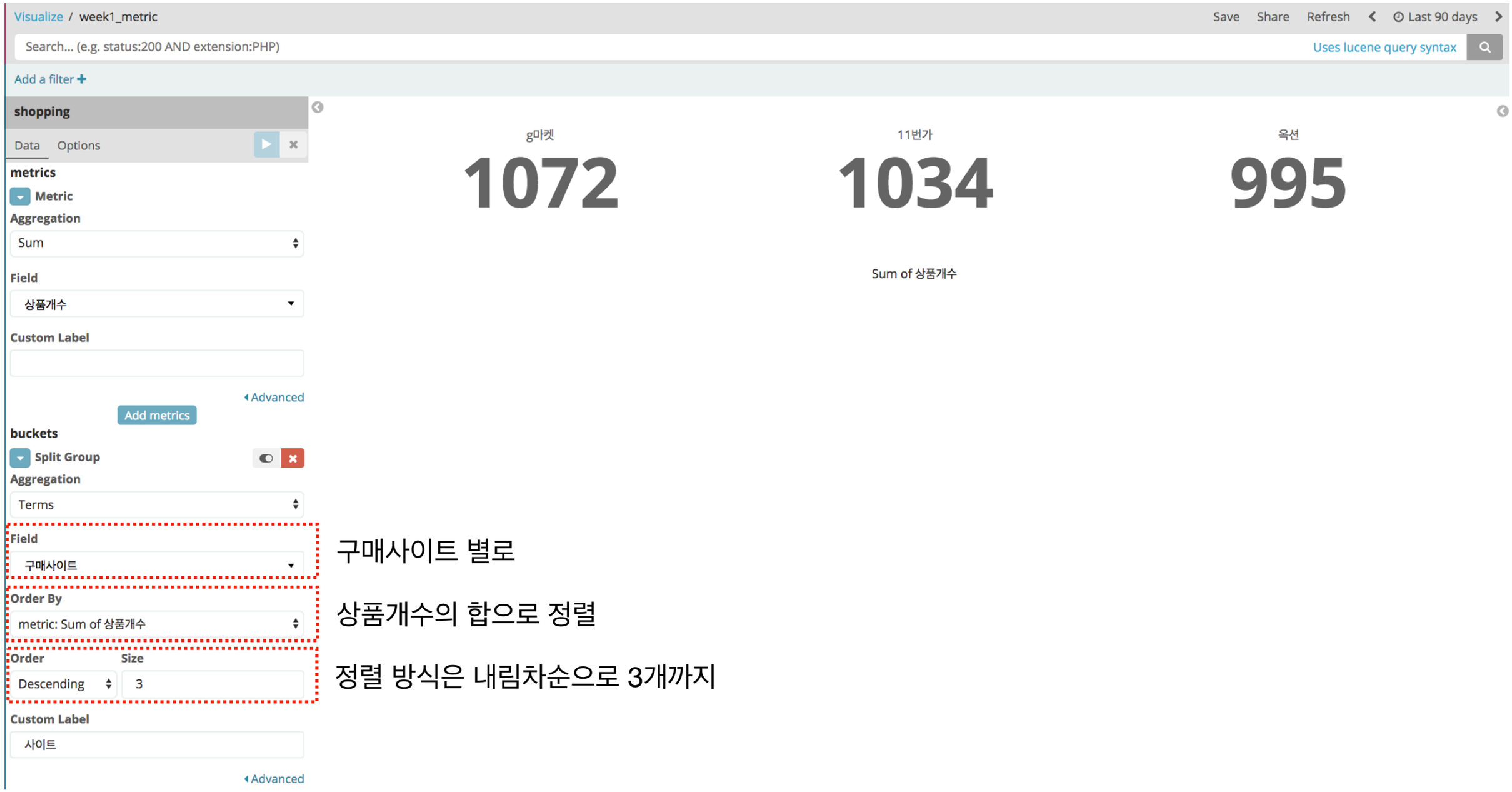


Visualize - Metric

metric aggregation에서 Sum aggregation을 선택하면 상품개수의 합이 표시된다



Visualize - Metric



구매사이트 별로

상품개수의 합으로 정렬

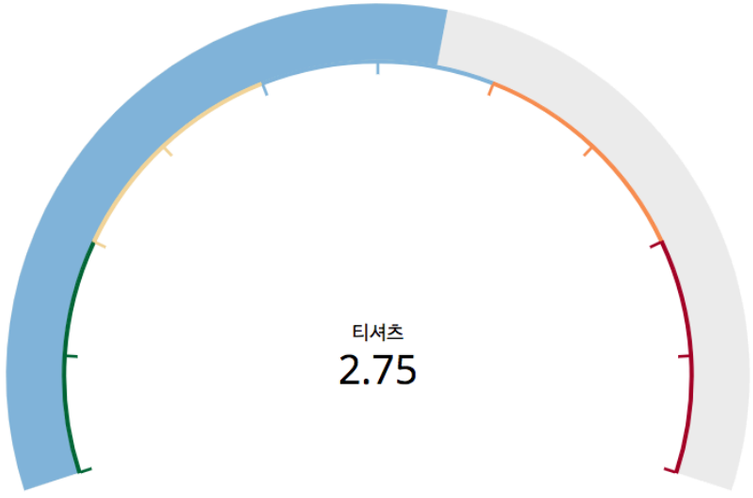
정렬 방식은 내림차순으로 3개까지

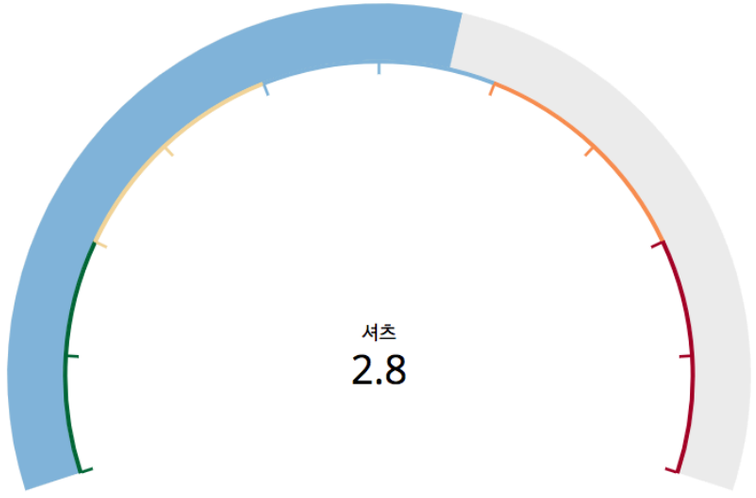
Visualize - Gauge

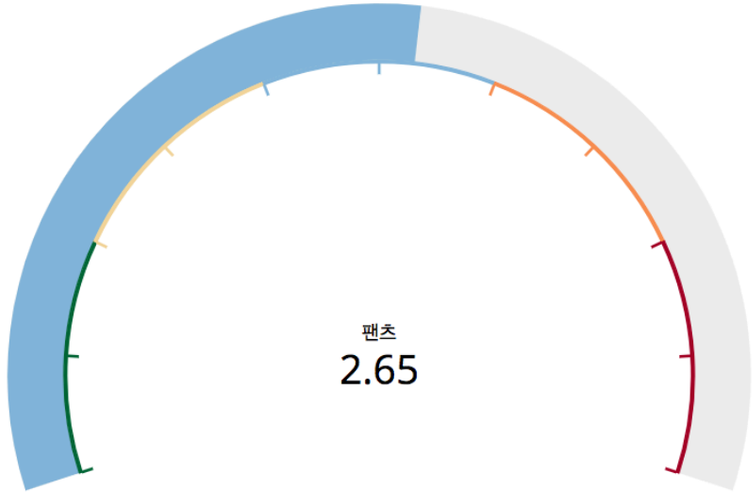
Visualize - Gauge

아래와 같은 Gauge를 만들어보자 ✍

- shopping Index에서
- 최근 90일 동안
- “상품개수” 의 합이 가장 많은 “상품분류” 3개 각각에 대해서
- “상품가격”의 큰 20개 건의 “판매자평점”(5점 만점)의 평균을
- 최종 목표 5, 구간별 목표 0~1, 1~2, 3~4, 4~5를 세우고 모니터링 하자

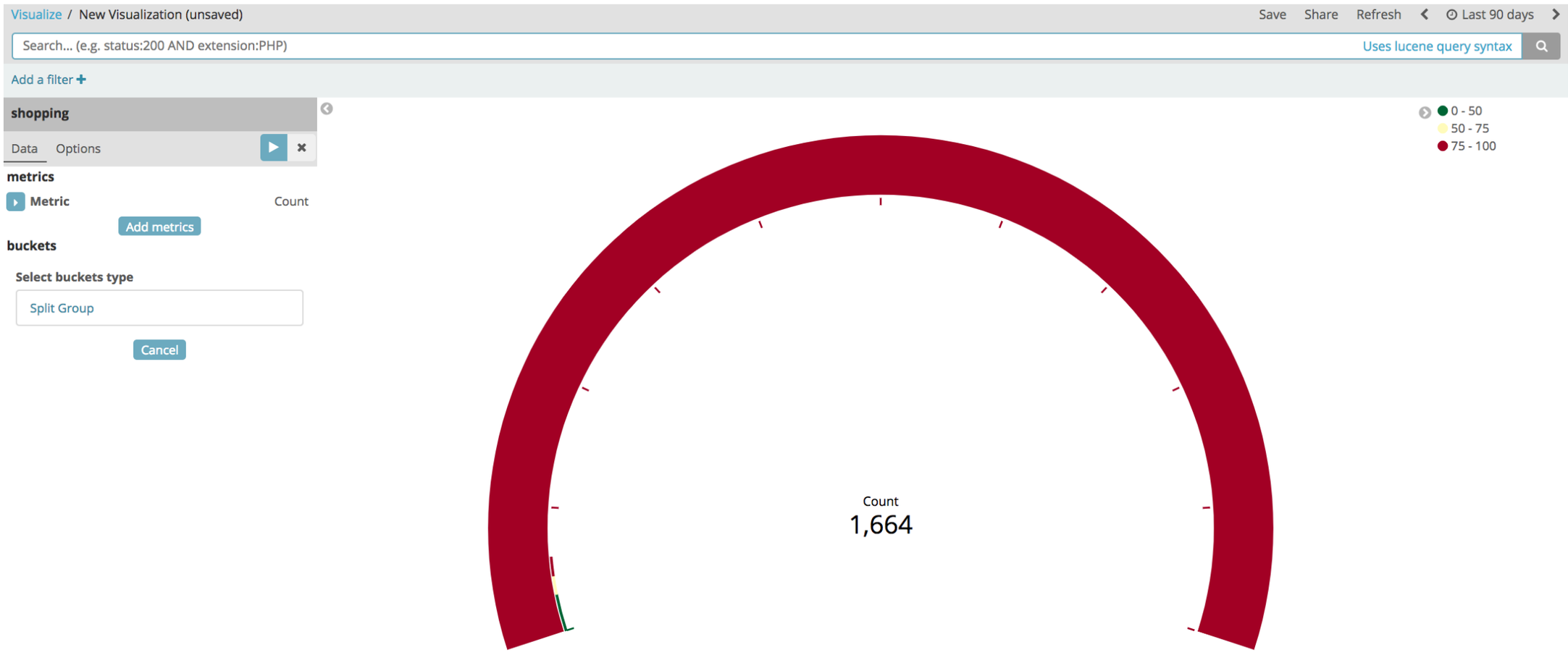






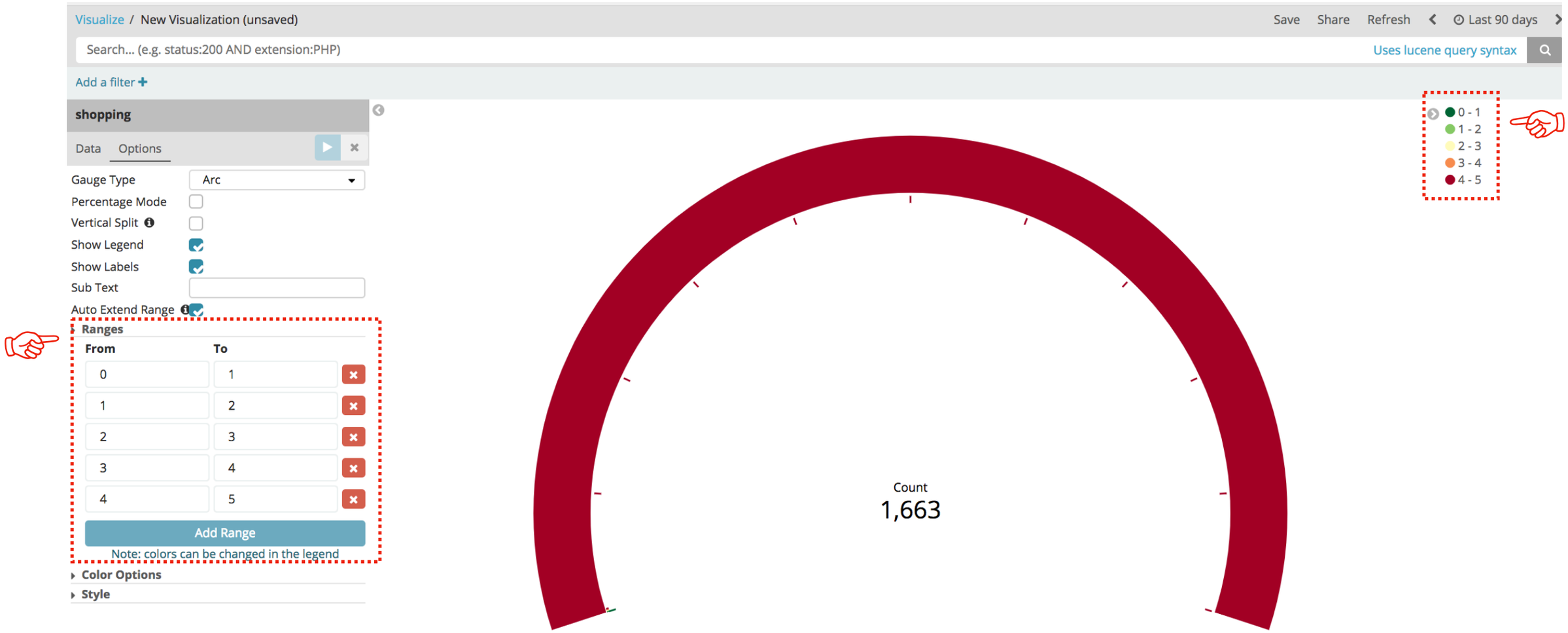
Visualize - Gauge

Gauge를 선택하면 처음에 이런 화면이 나온다



Visualize - Gauge

Options에서 목표 설정부터 하자

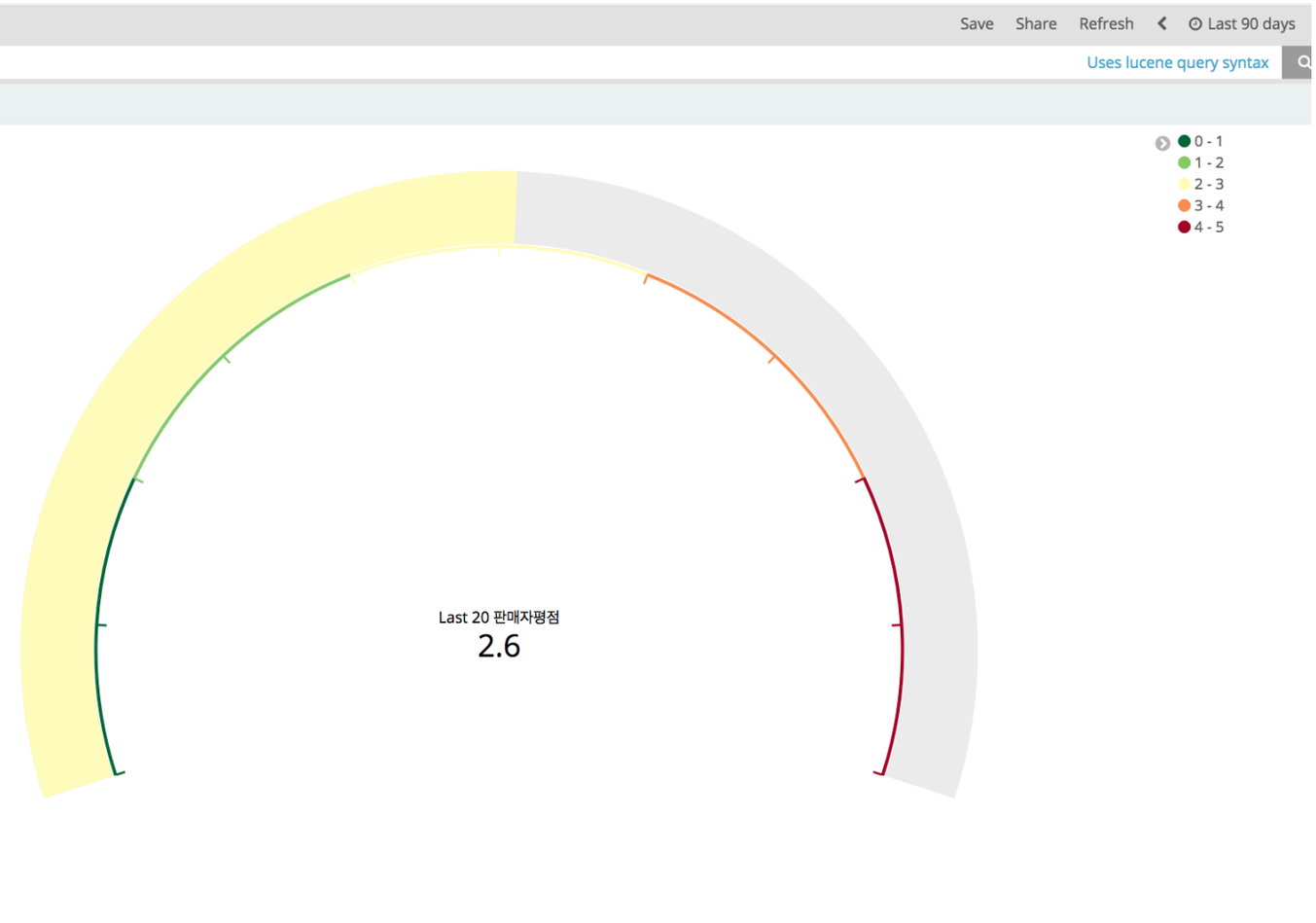


Visualize - Gauge

3) 판매자평점의 평균을 표시

2) 상위 20개의

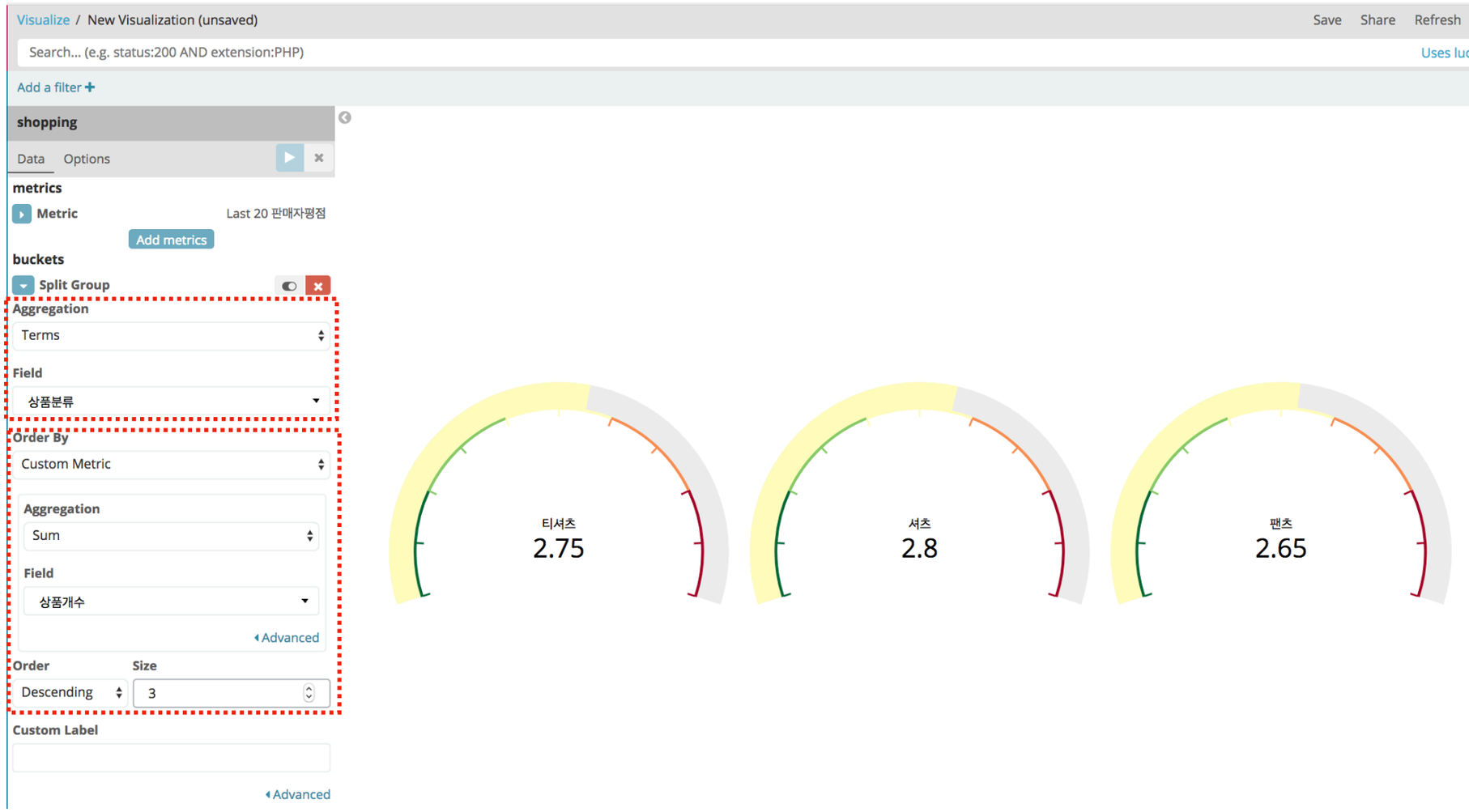
1) 상품가격이 높은 순으로 정렬해서



Visualize - Gauge

2) 상품분류를 선택

1) 상품개수의 합이 가장 큰 3개의

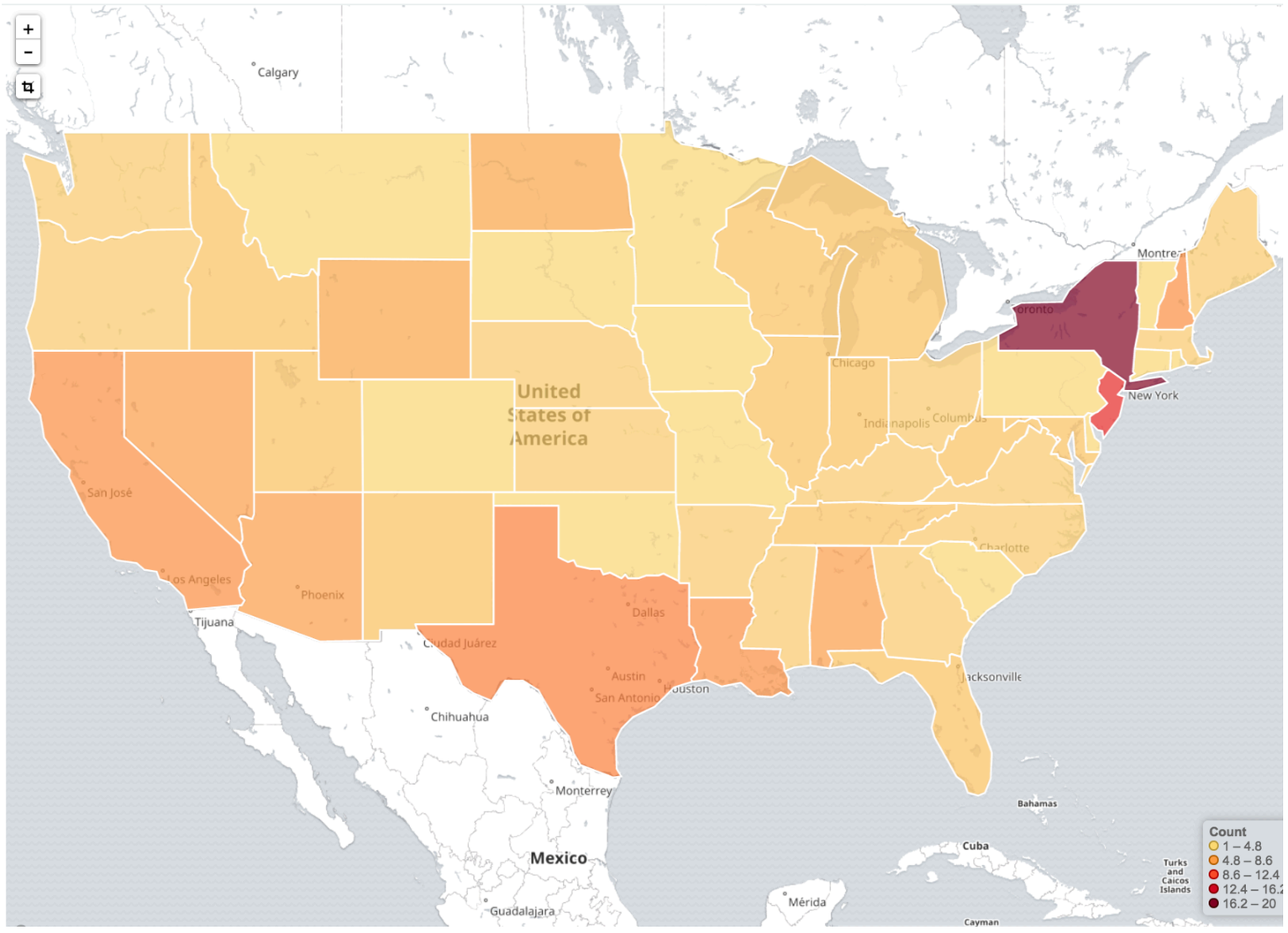


Visualize - Region Map ♕

Visualize - Region Map

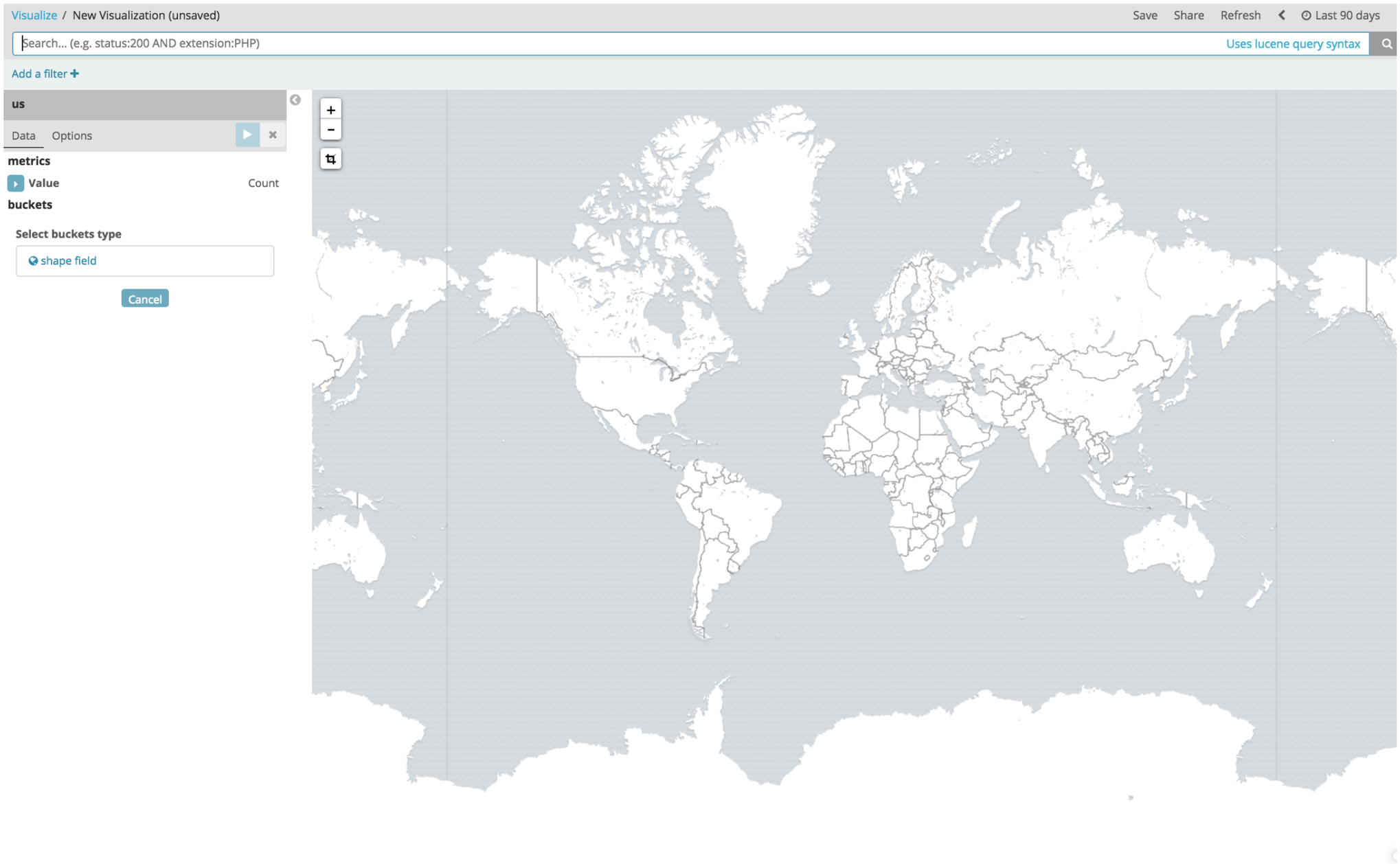
아래와 같은 Region Map를 만들어보자 ✍

- us Index에서
- 최근 90일 동안
- 주(state) 별로
- Documents Count를 표시해보자



Visualize - Region Map

Region Map를 선택하면 처음에 이런 화면이 나온다

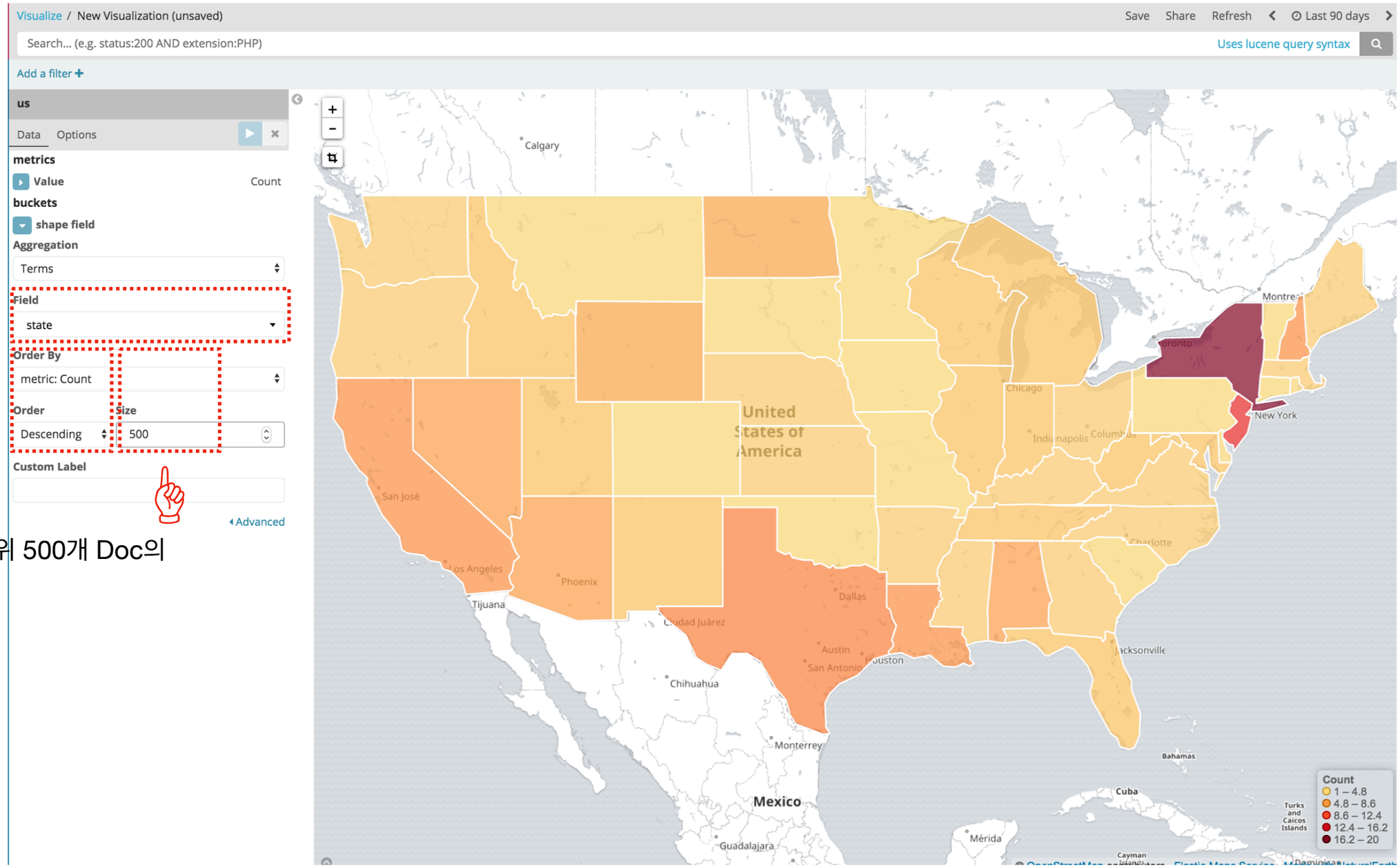


Visualize - Region Map

3) state에 대해서

1) Doc이 많은 순으로 정렬하고

2) 상위 500개 Doc의



Visualize - Data Table

Visualize - Data Table

아래와 같은 Data Table를 만들어보자 ✍

- shopping Index에서
- 최근 90일 동안
- “주문시간” Field 기준으로 일별로
- 매출 // 누적 매출 // 매출 증감

Visualize / week1_data_table

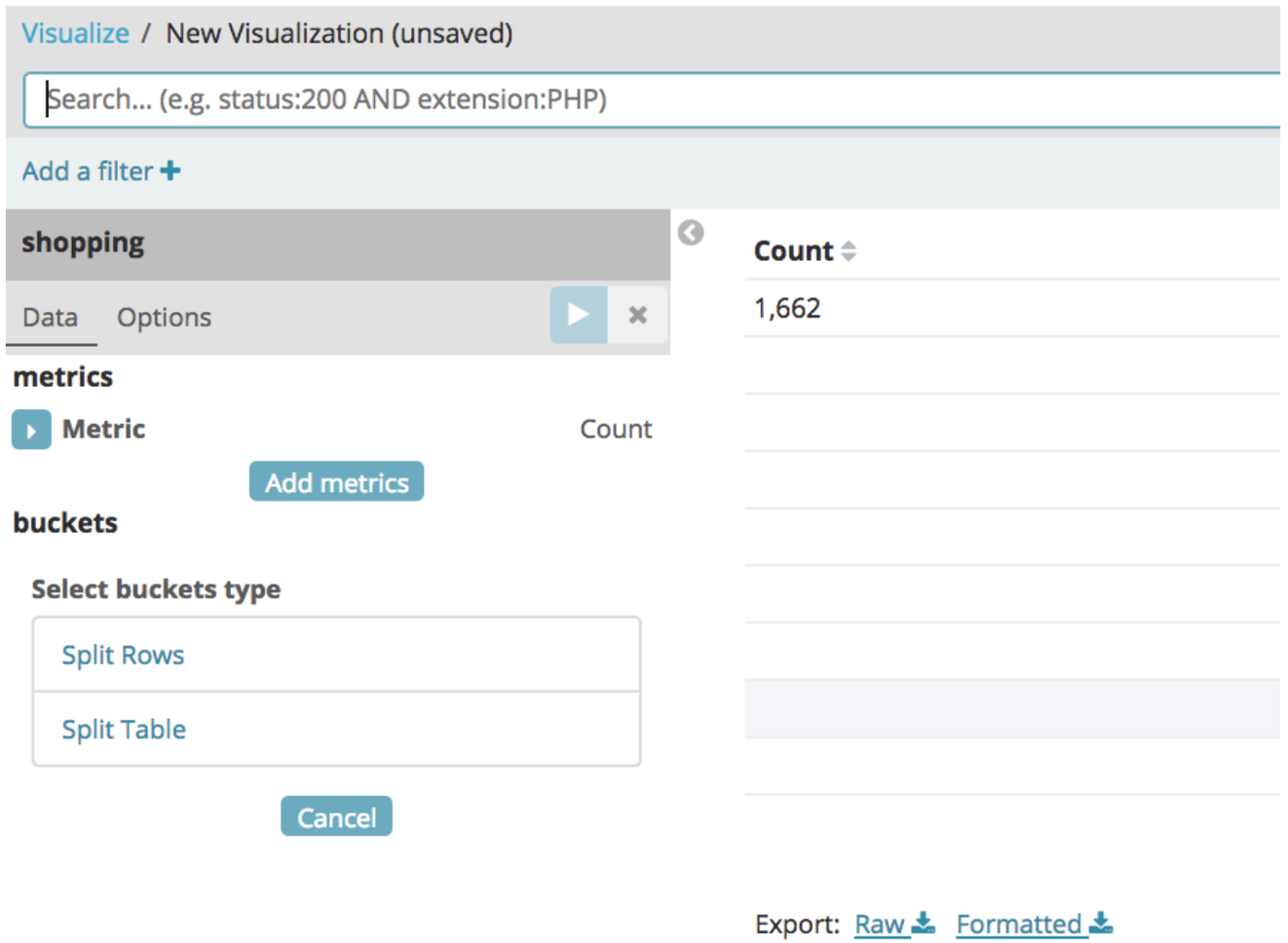
Search... (e.g. status:200 AND extension:PHP)

Add a filter +

날짜	매출	누적 매출	전일 대비 매출 증감
10월17일	304,000	304,000	-
10월18일	465,000	769,000	161,000
10월19일	347,000	1,116,000	-118,000
10월20일	320,000	1,436,000	-27,000
10월21일	380,000	1,816,000	60,000
10월22일	314,000	2,130,000	-66,000
10월23일	365,000	2,495,000	51,000
10월24일	303,000	2,798,000	-62,000
10월25일	361,000	3,159,000	58,000
10월26일	467,000	3,626,000	106,000

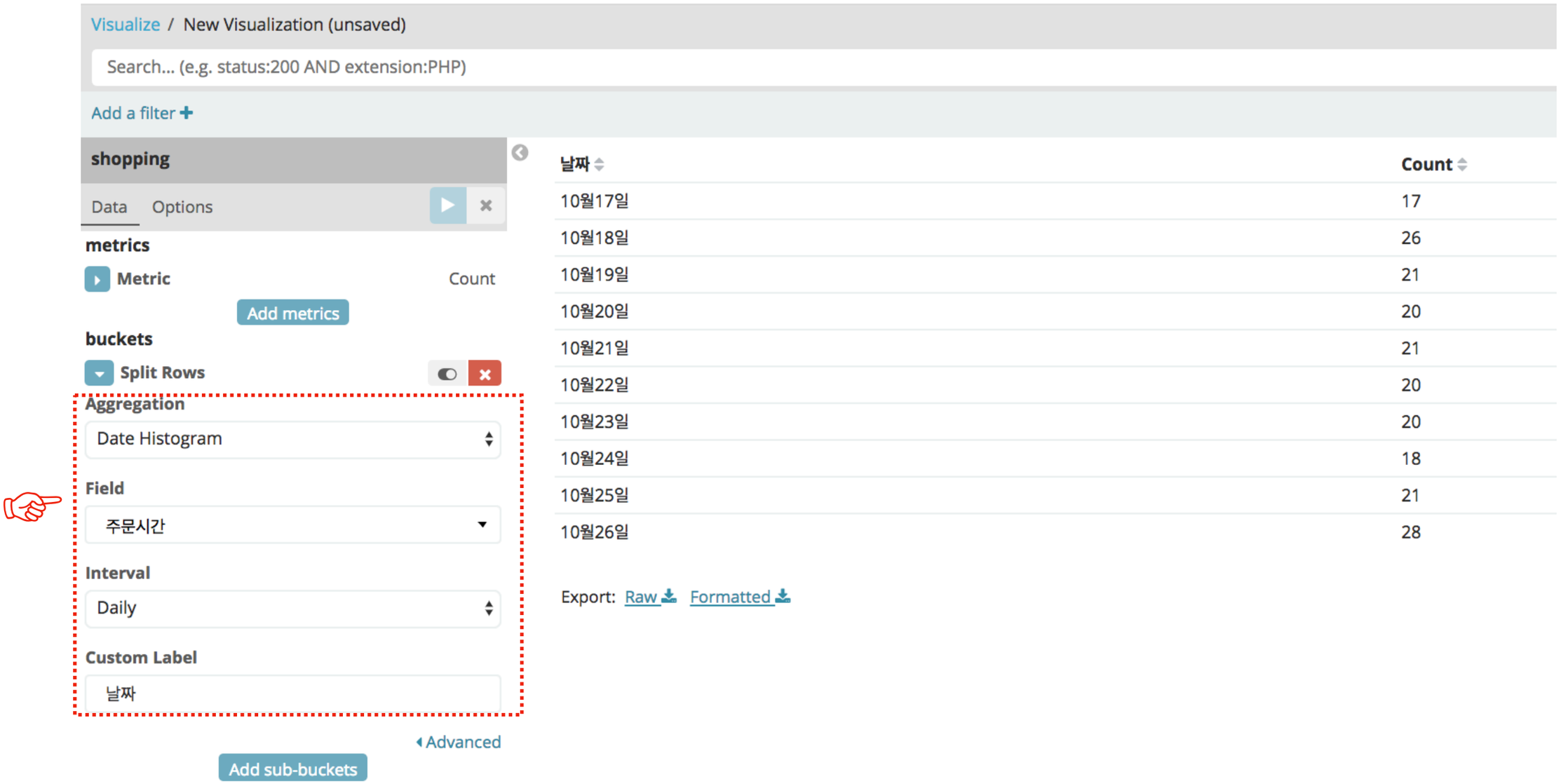
Visualize - Data Table

Data Table을 선택하면 처음에 이런 화면이 나온다



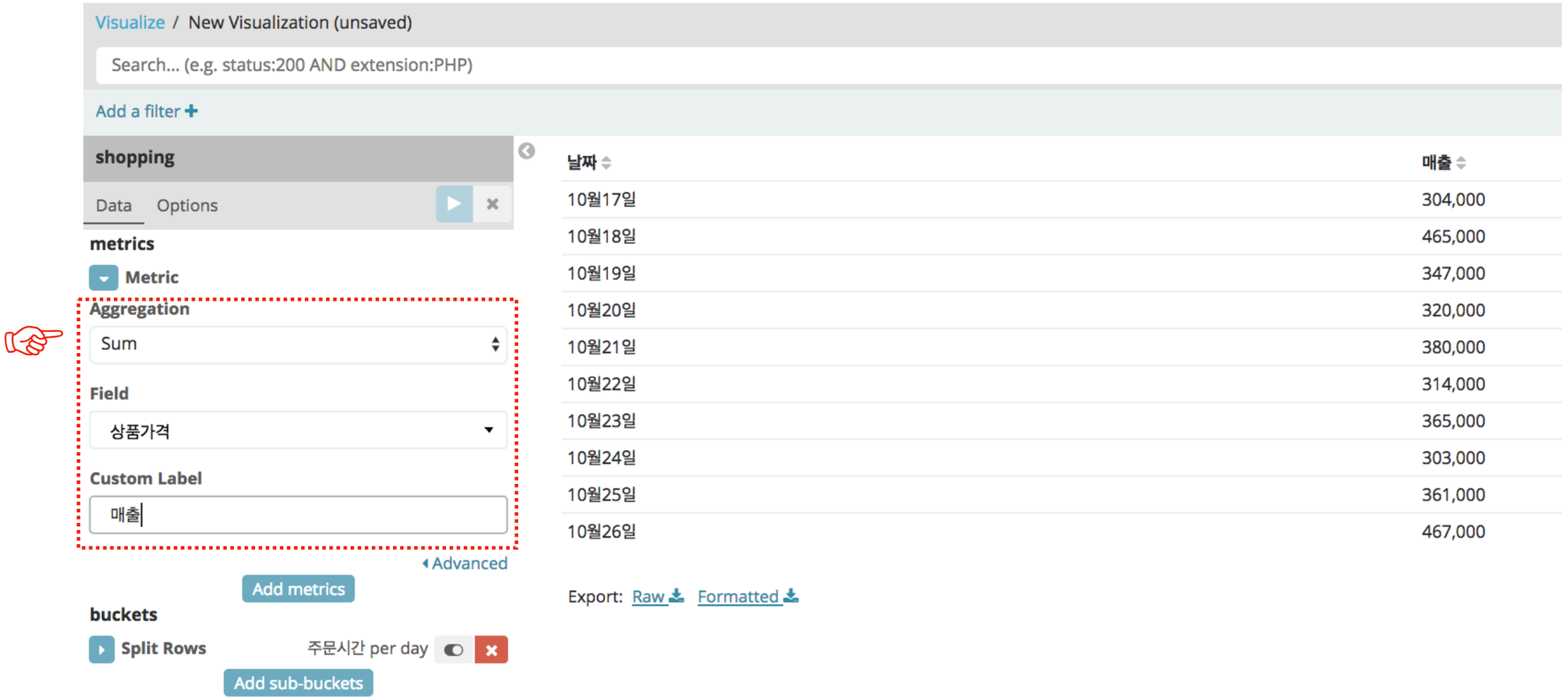
Visualize - Data Table

buckets (Split Rows)를 통해 일별 row가 생성된다



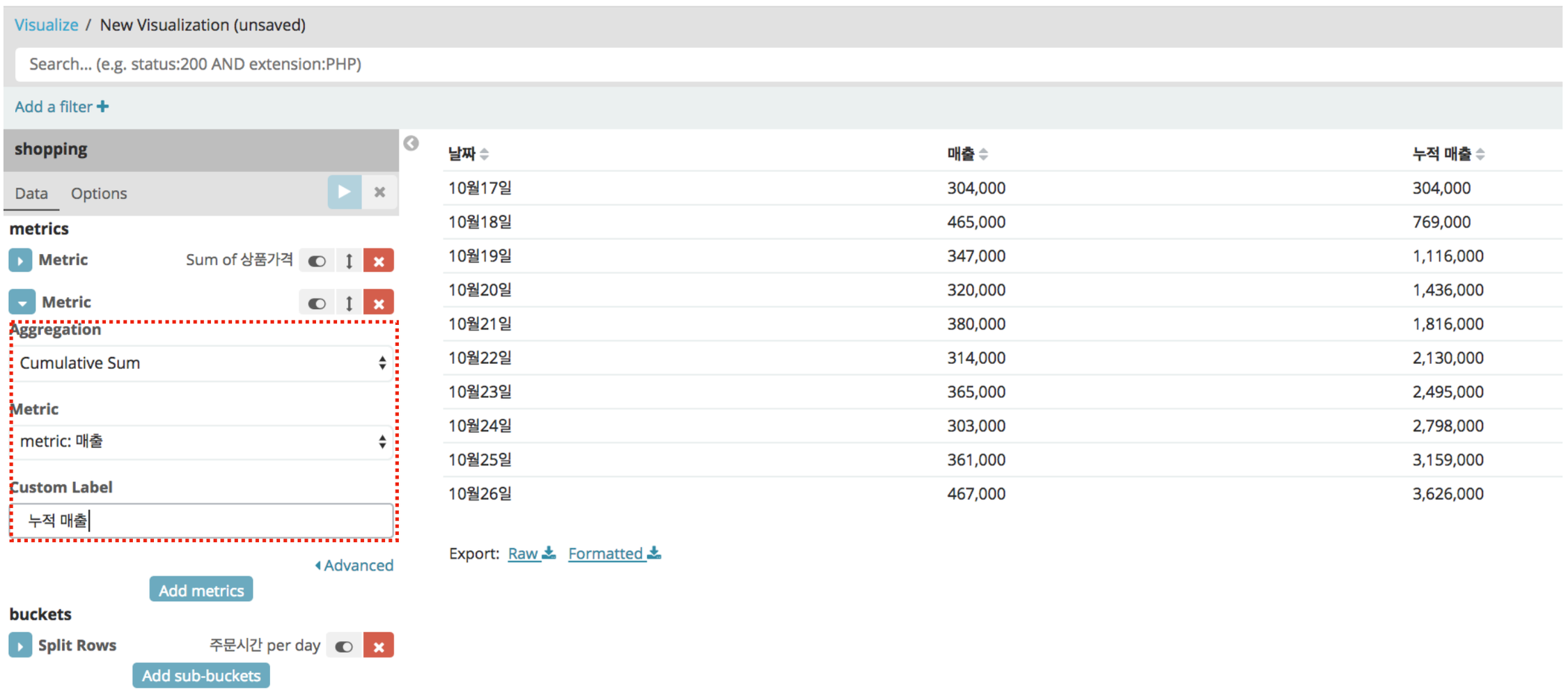
Visualize - Data Table

sum aggregation을 통해 일별 매출을 구한다



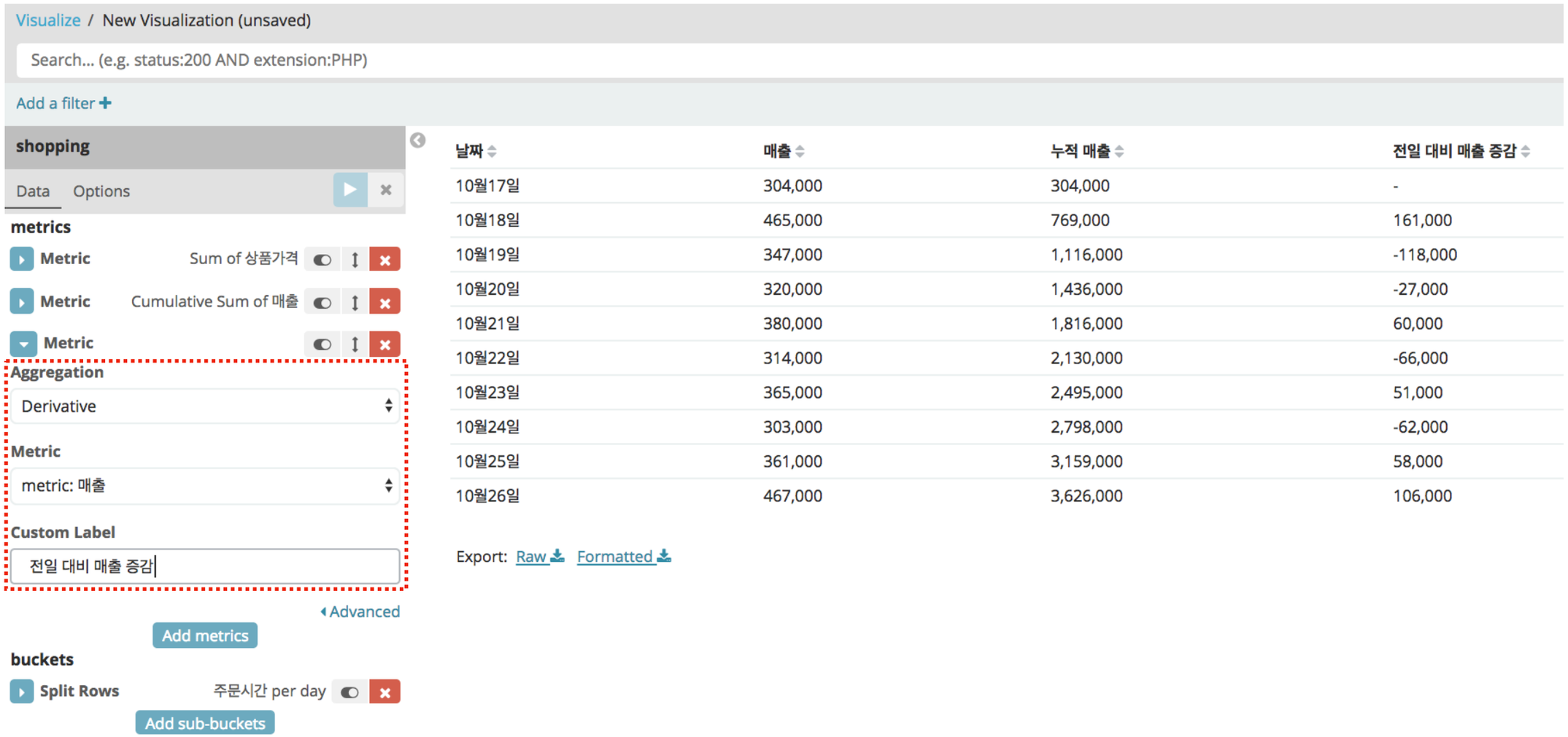
Visualize - Data Table

parent pipeline aggregation (cumulative sum)을 통해 일별 누적 매출을 구한다



Visualize - Data Table

parent pipeline aggregation (derivative)을 통해 일별 누적 매출을 구한다



Timelion ♕ ♕

Timelion은 대표적으로 다음과 같은 Function들을 제공한다

- 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()`

Kibana - Timelion

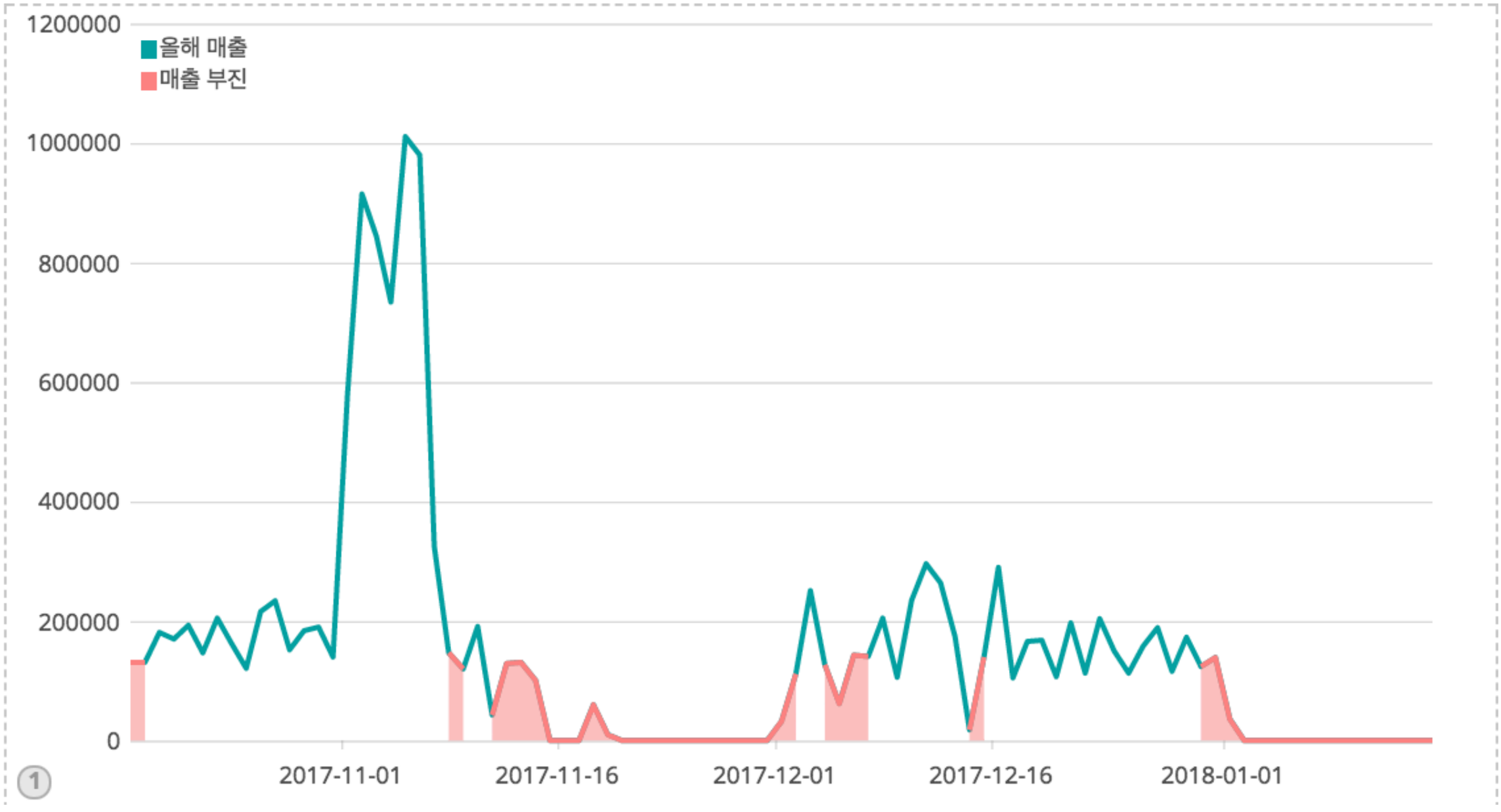
Q. 음? 기존 Line/Bar/Area 등과 뭐가 다르지?

A. ~~(적당히 사용해본 결과)~~ 다음 두 가지가 가장 매력적

- (Index Pattern 등록시 wildcard로 묶지 않은) **서로 다른 Index**의 데이터도 한 번에 시각화 가능
  - 예를 들어 *hi* index와 bye index처럼 완전히 상관없어 보이는 index도 가능
- **서로 다른 시간대**의 데이터를 같은 시간대로 표현 가능
  - 예를 들어 올해 10월 매출과 전년동월 매출이 궁금할 때 한 번에 시각화 가능

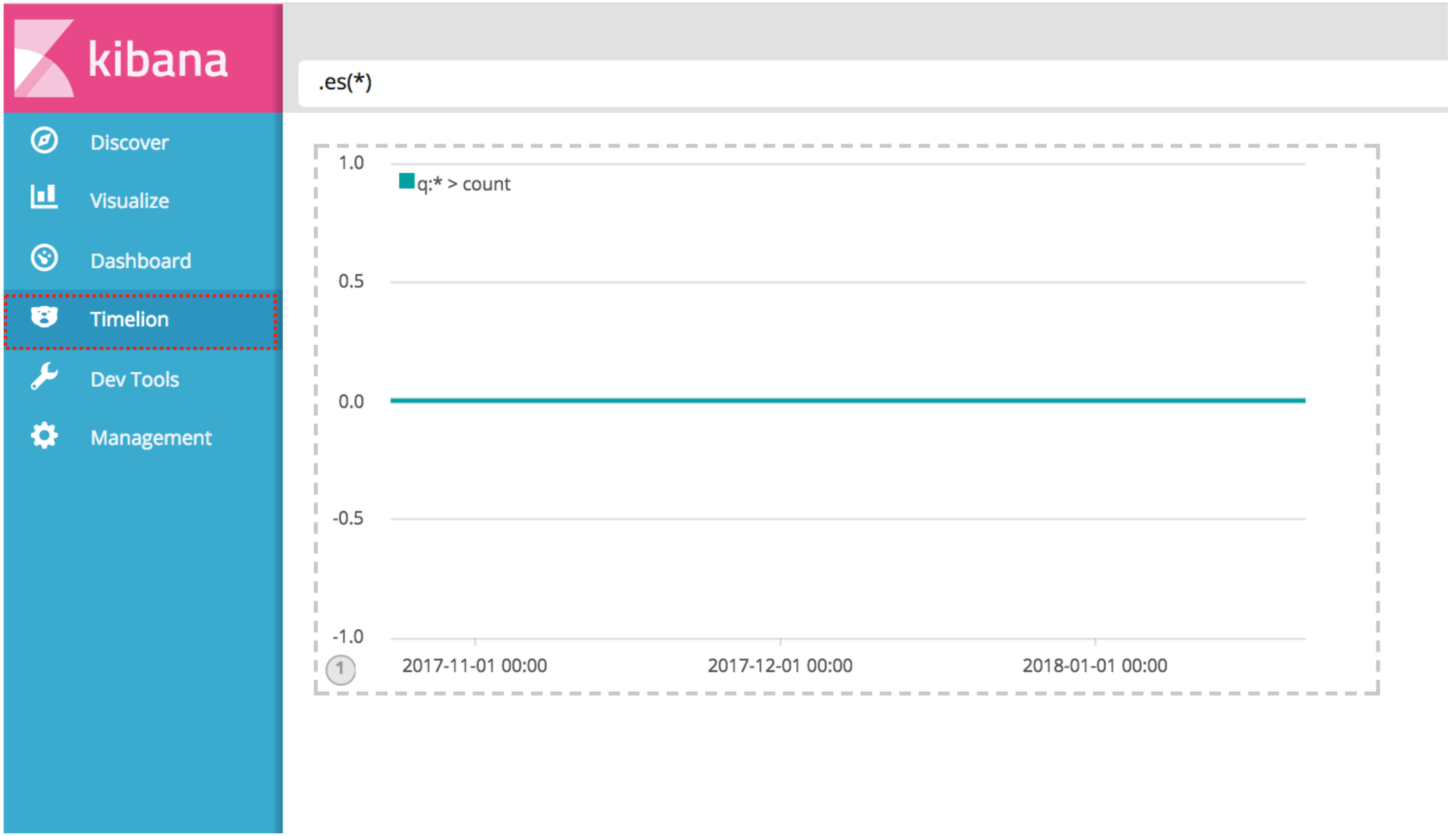
Kibana - Timelion

아래와 같은 Timelion을 만들어보자 ✍



Kibana - Timelion

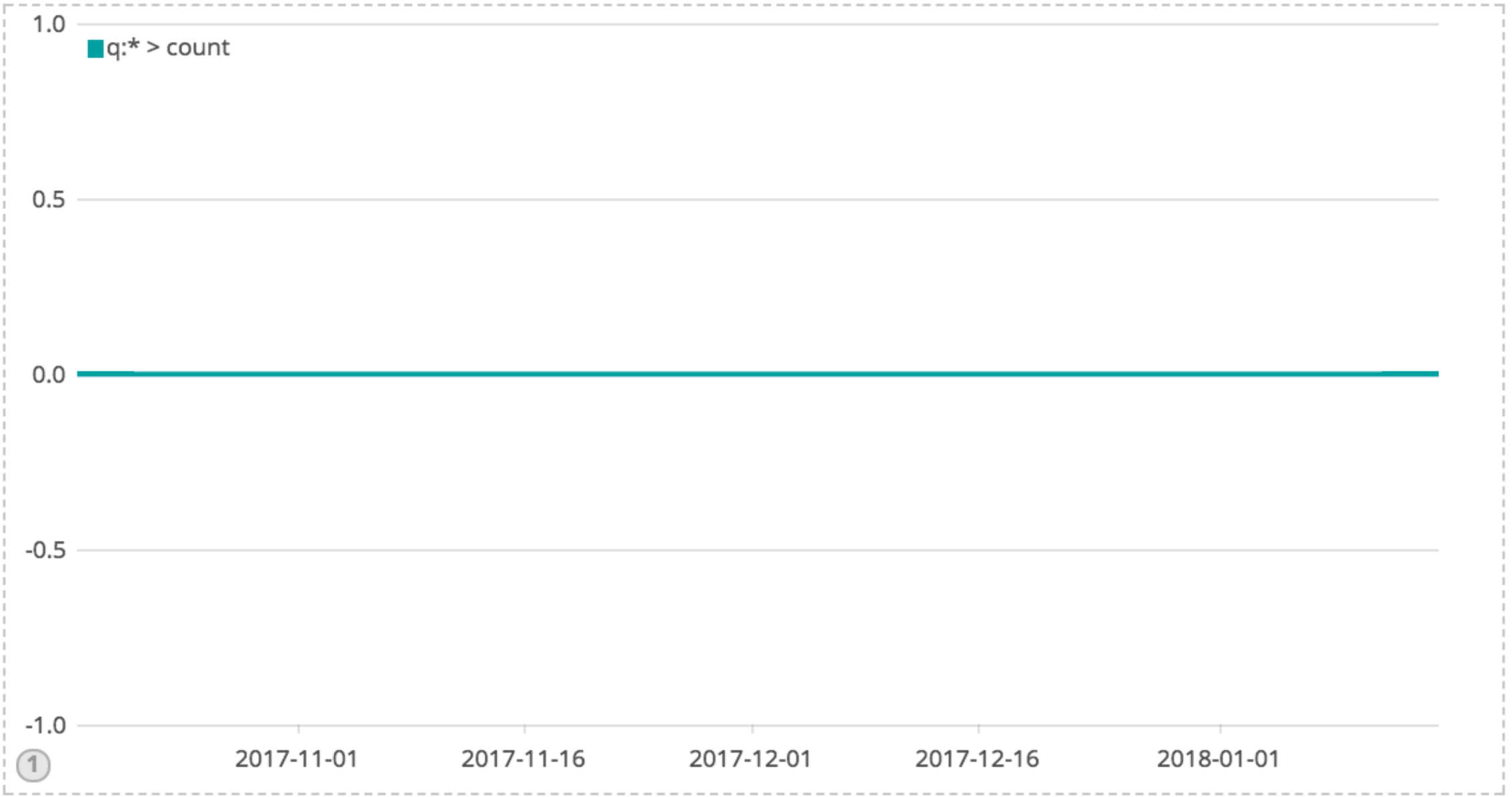
Kibana에 접속해서 Timelion 화면으로 가보자 !



Kibana - Timelion

Timelion를 선택하면 처음에 이런 화면이 나온다 👑

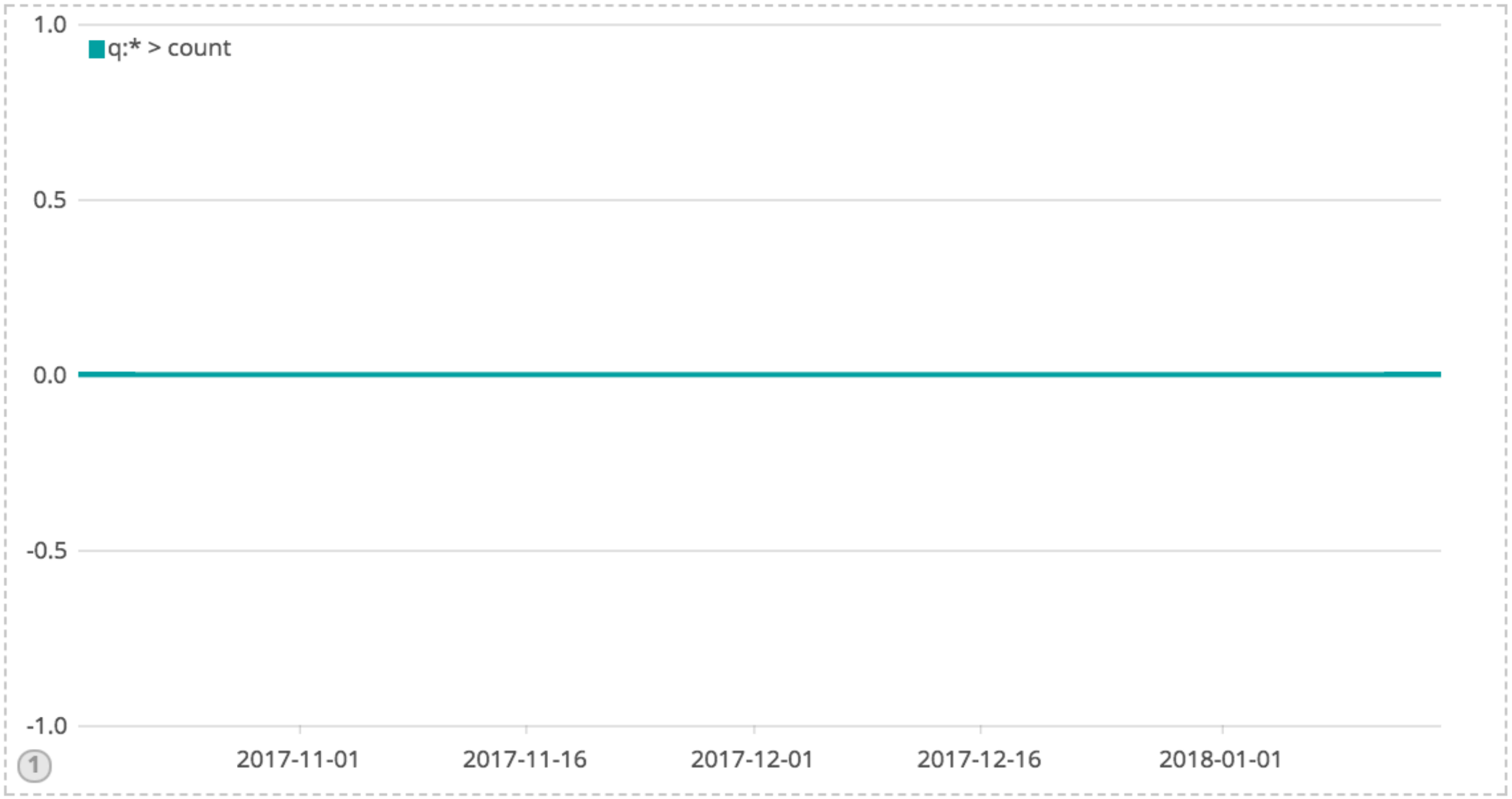
.es(*)



Kibana - Timelion

Index를 설정해주자 👑

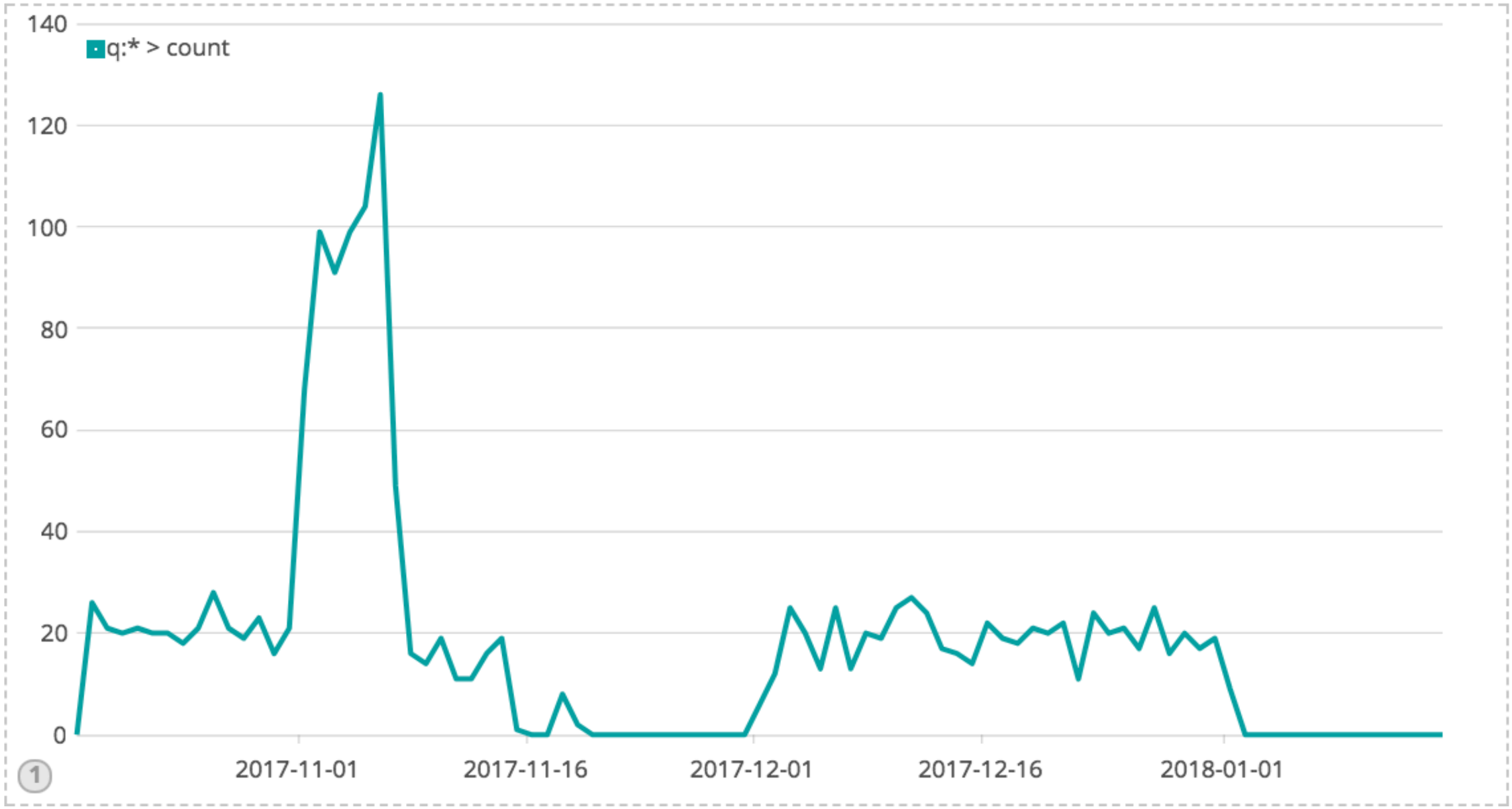
.es(index=shopping)



Kibana - Timelion

Timefield도 추가하자 👑

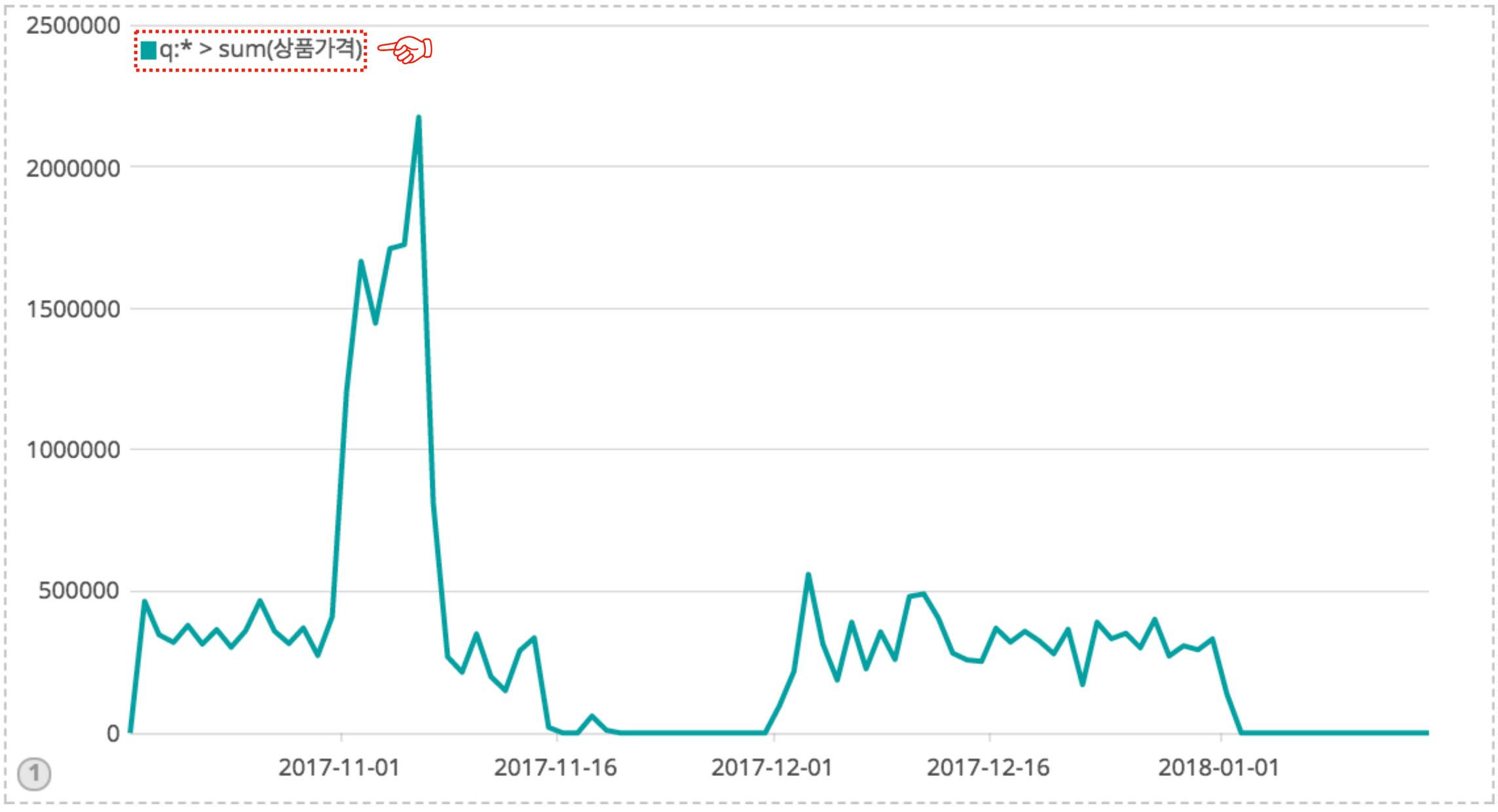
```
.es(
  index=shopping,
  timefield=주문시간
)
```



Kibana - Timelion

metric도 추가해보자 👑

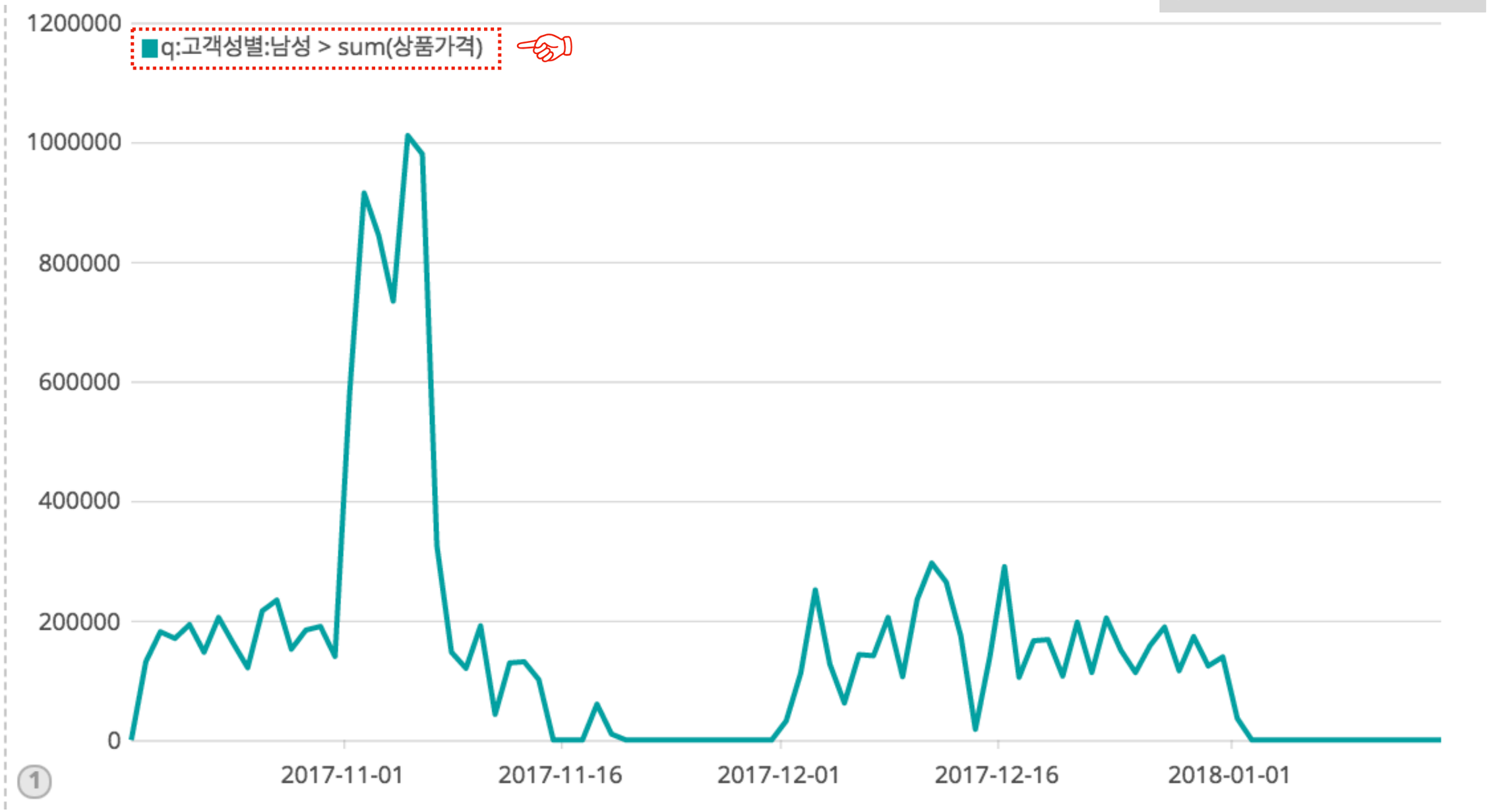
```
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격
)
```



Kibana - Timelion

query도 추가해보자 👑

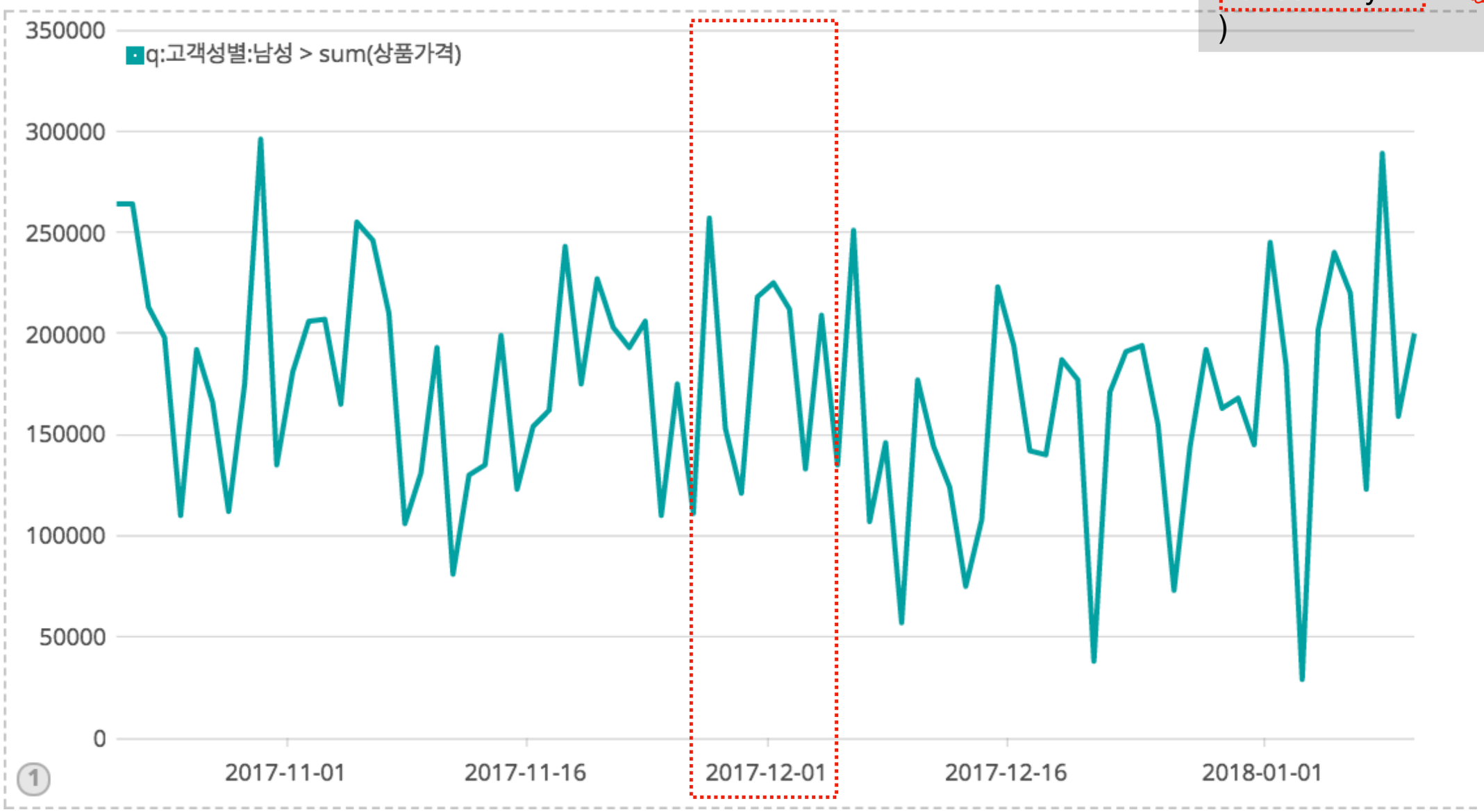
```
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성
)
```



Kibana - Timelion

offset도 추가해보자 👑

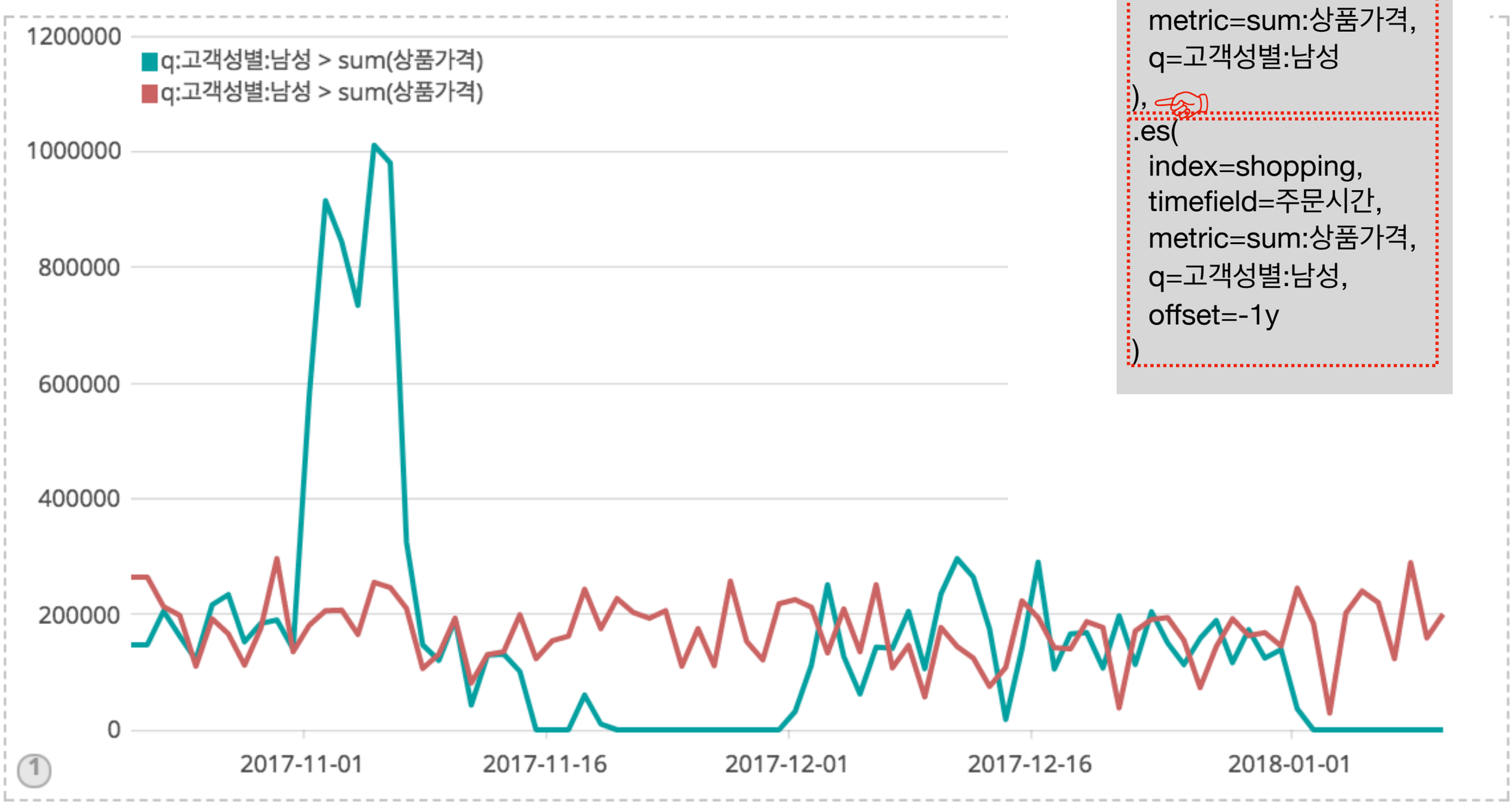
```
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성,
  offset=-1y
)
```



2016년 12월 1일 데이터 표시

Kibana - Timelion

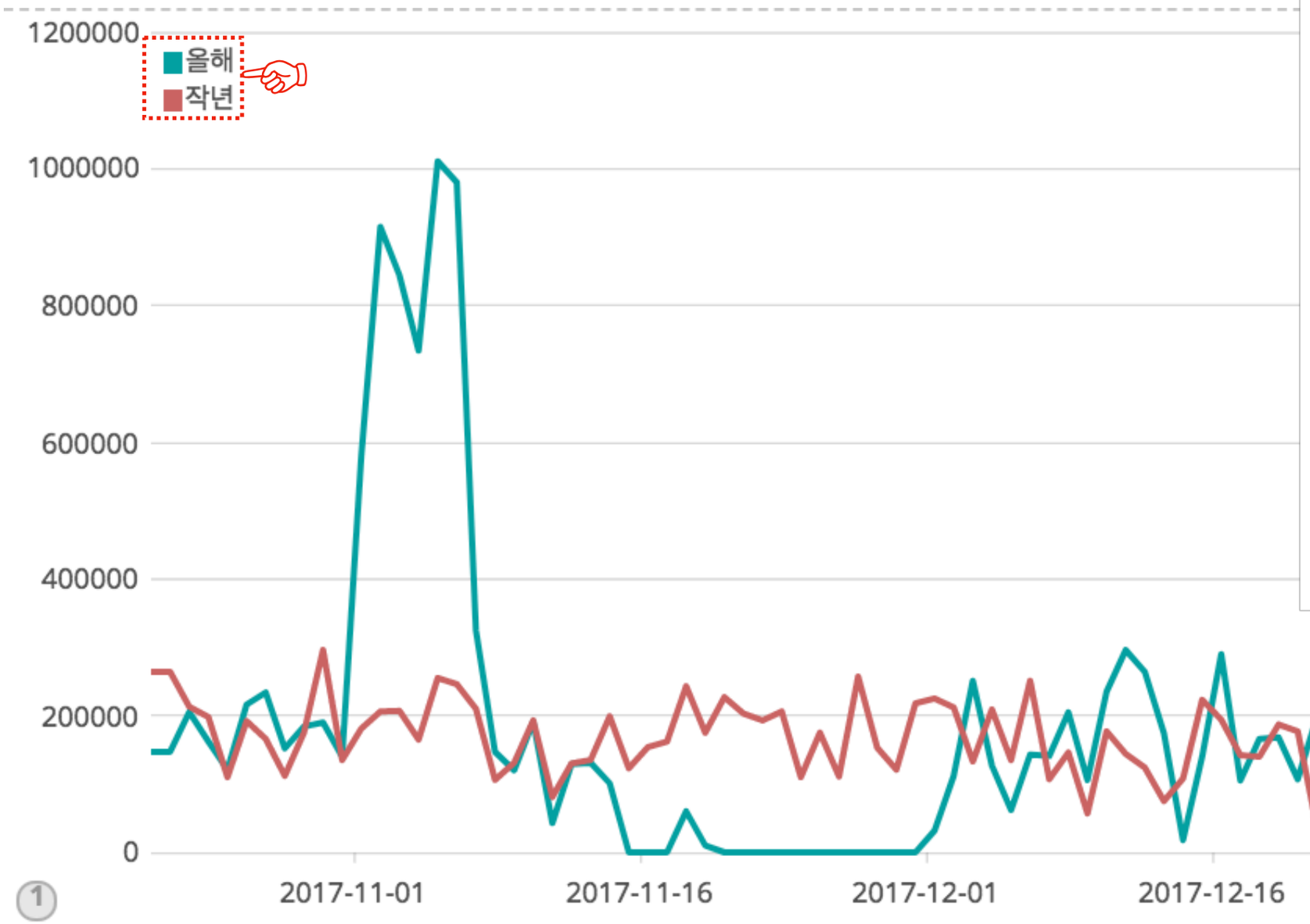
두 그래프를 같이 표시하자 ♕




```
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성
),
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성,
  offset=-1y
)
```

Kibana - Timelion

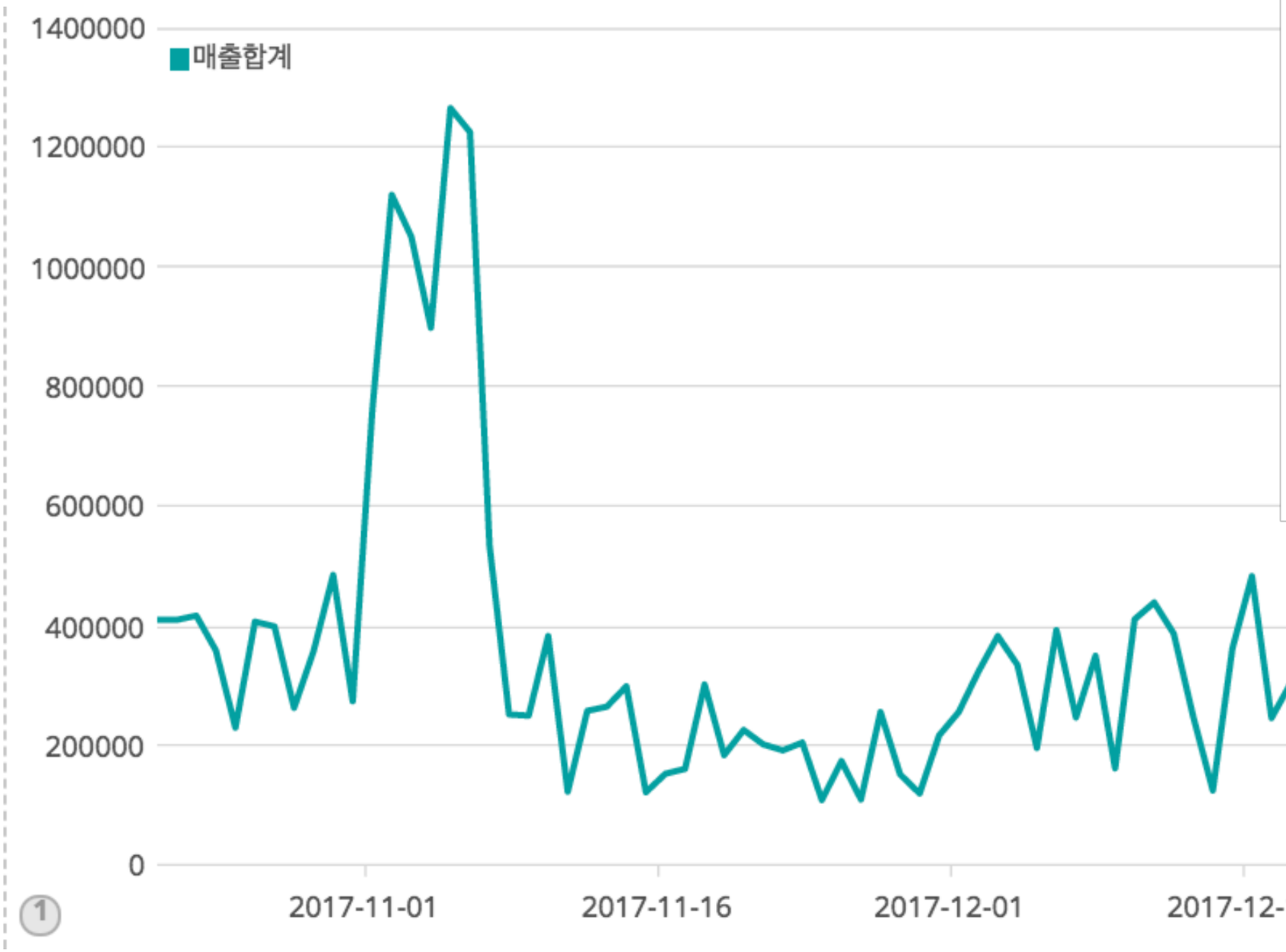
labeling을 하자 👑




```
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성
).label(올해),
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성,
  offset=-1y,
).label(작년)
```

Kibana - Timelion

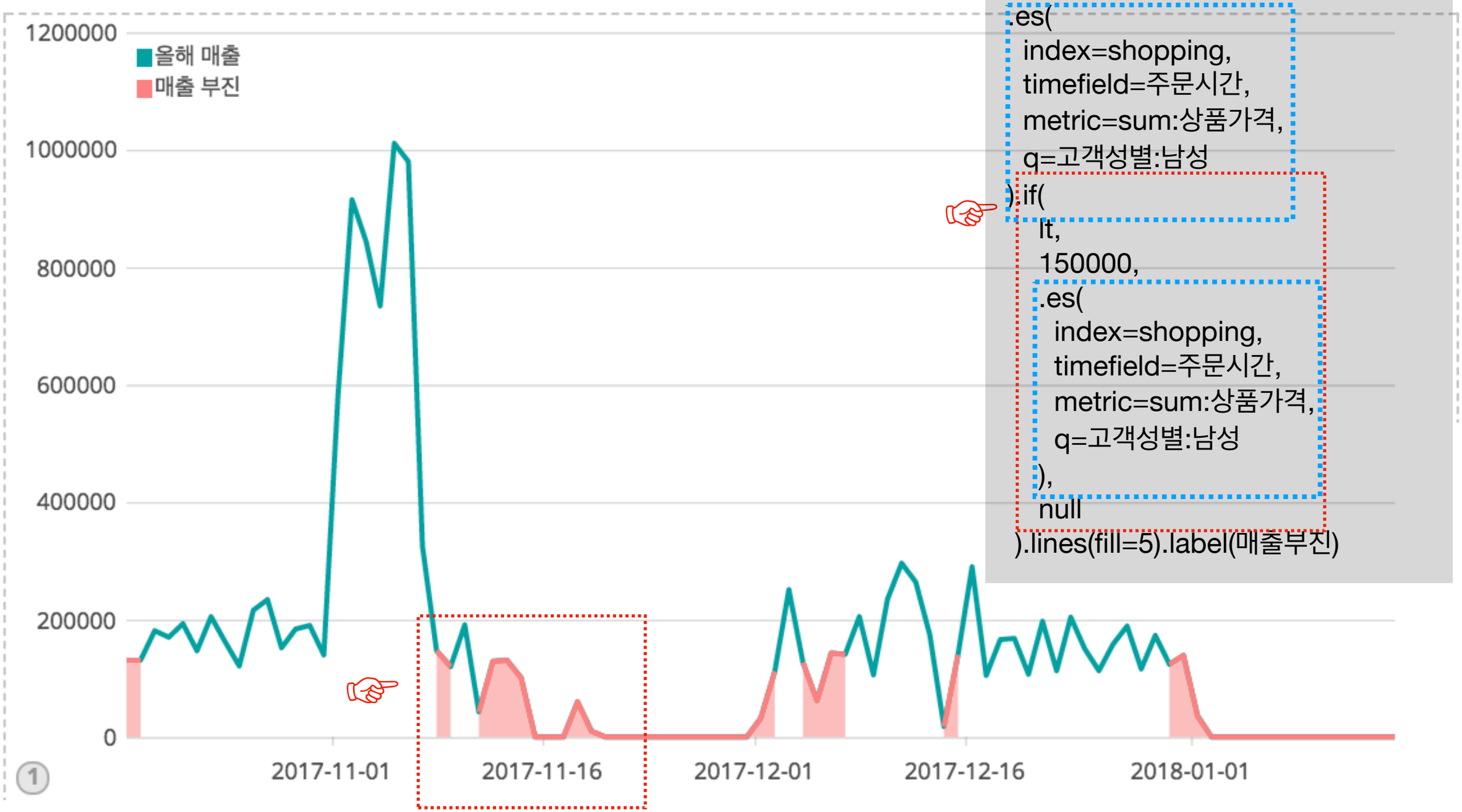
작년과 올해 매출 합을 구하자 👑




```
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성
).sum(
  .es(
    index=shopping,
    timefield=주문시간,
    metric=sum:상품가격,
    q=고객성별:남성,
    offset=-1y
  )
).label(매출합계)
```

Kibana - Timelion

올해 매출이 15만보다 작은 날을 표시해보자 👑

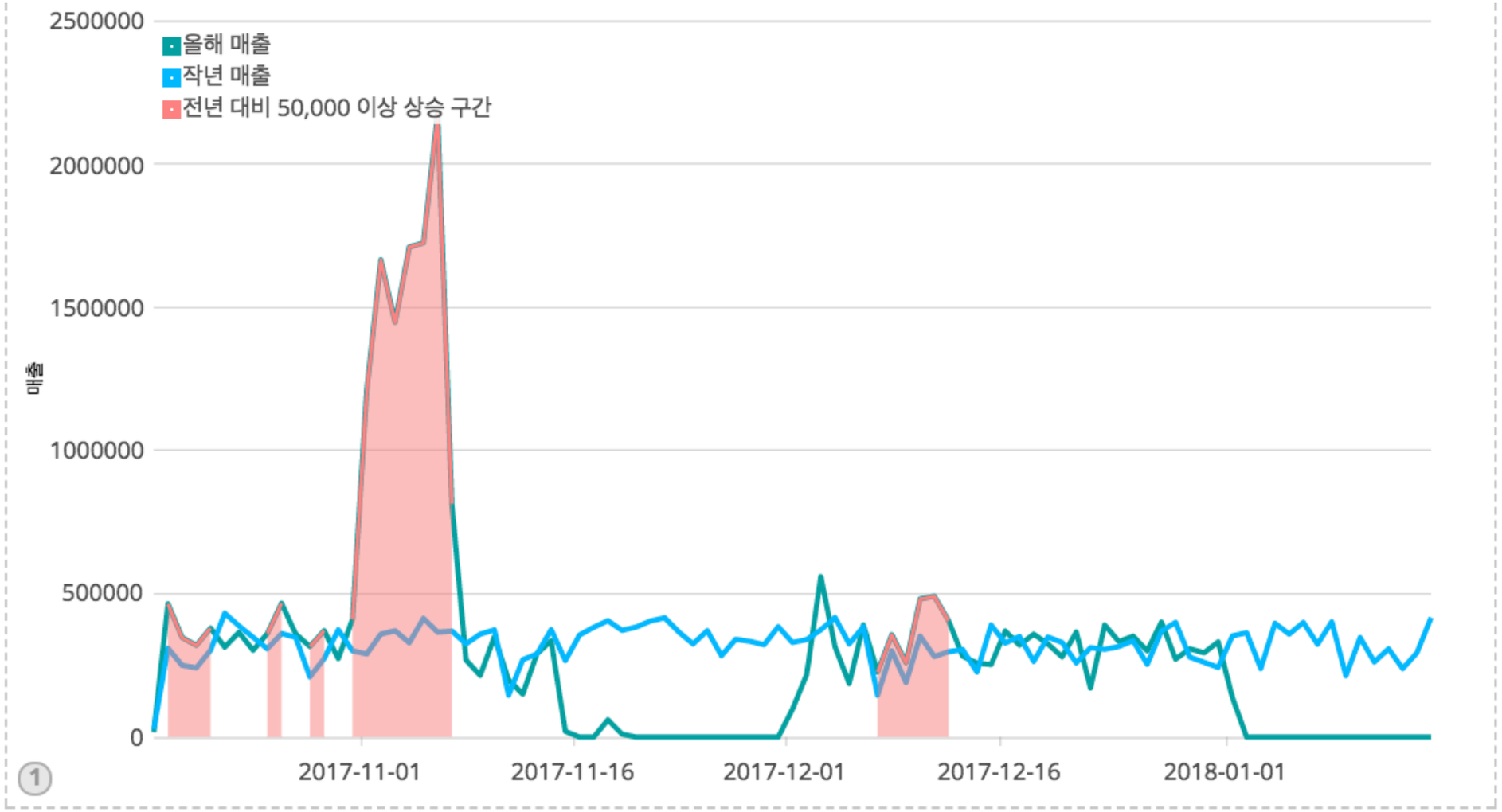



```
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성
).label(올해매출),
.es(
  index=shopping,
  timefield=주문시간,
  metric=sum:상품가격,
  q=고객성별:남성
).if(
  lt,
  150000,
  .es(
    index=shopping,
    timefield=주문시간,
    metric=sum:상품가격,
    q=고객성별:남성
  ),
  null
).lines(fill=5).label(매출부진)
```

Kibana - Timelion

아래와 같은 조건을 만족하는 Timelion를 만들어보자 ✍ 👑

- index : shopping
- time field : 주문시간
- 올해 매출 : 2017.10.17 ~ 2018.01.15 “상품가격”의 합
- 작년 매출 : 2016.10.17 ~ 2017.01.15 “상품가격”의 합
- 전년 대비 50,000 이상 상승 구간 : 올해 매출이 작년 매출 대비 50,000 이상인 날짜



Kibana - Dashboard

Kibana - Dashboard 👑

오늘 수업 때 활용한 **Dashboard** 👑

- Auto Refresh
- Interactive Dashboard
- Elasticsearch Aggregation으로 빠른 데이터 조회 및 시각화
- 간단한 쿼리로 검색 가능
- 쉬운 공유

오늘의 목표 - 확인

Kahoot ! 👑