4️⃣

# [4주차] 5장 엘라스틱서치: 집계

> 집계: 데이터를 그룹핑하고 통곗값을 얻는 기능

- SQL의 group by  와 통계함수를 포함하는 개념

▼ ex

항공사별 수하물 점유율은?

비행기 티켓의 평균값은?

시간별 비행기 연착/취소 비율은?

5.1 집계의 요청-응답 형태

▼ 요청 기본

```
GET <인덱스>/_search
{
  "aggs": {
    "NAME": { <- 집계 이름
      "AGG_TYPE": { <- 집계 타입: 메트릭 or 버킷
        ....
      }
    }
  }
}
```

▼ 응답 기본

```
{ ...
"hits": {
  "total": {
  ....
},
"aggregations": {
  "my_aggs": { -> 집계 이름
    "value":  -> 실제 집계 결과
```

```
  }
}
}
```

5.2 매트릭 집계

- 최소/최대/합계/평균/중간값 같은 통계 결과
- 집계 타입에 대한 제한O (ex.텍스트 타입 필드)
- 지리 정보 필드(Geo_point) 를 위한 집계 존재

| | |
|---|---|
| avg | 필드의 평균값 계산 |
| min | 필드의 최솟값 계산 |
| max | 필드의 최댓값 계산 |
| sum | 필드의 총합 계산 |
| percentiles | 필드의 백분윗값 계산 |
| status | 필드의 min,max,sum,avg,count(도큐먼트수)를 한번에 볼 수 있다. |
| cardinality | 필드의 유니크한 값 갯수 |
| geo-centroid | 필드 내부의 위치 정보의 중심점 계산 |

▼ 평균값/중간값 구하기

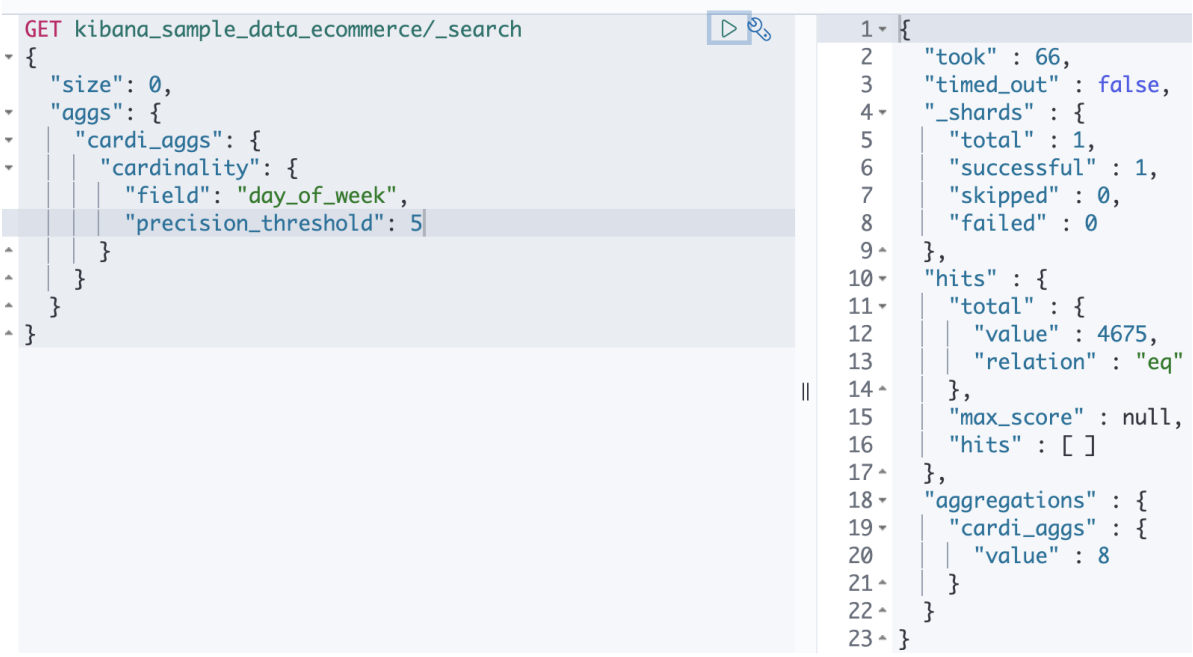
- 평균 집계를 사용하기 위해서 필드 타입은 정수나 실수 타입이어야 한다.
- “size”:0 ⇒ 집계에 사용한 도큐먼트를 결과에 포함하지 않음 ⇒ 비용절감


평균값을 구하는 집계 요청

- 백분위 집계는 필드의 특정 백분위에 속하는 데이터를 찾아준다.


백분위를 구하는 집계 요청- 25~50%에 속하는 데이터를 요청한다.

▼ 필드의 유니크한 값 개수 확인하기 = 카디널리티



```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "cardi_aggs": {
      "cardinality": {
        "field": "day_of_week",
        "precision_threshold": 5
      }
    }
  }
}
```

```
{
  "took" : 66,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 4675,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "cardi_aggs" : {
      "value" : 8
    }
  }
}
```

precision_threshold 카디널리티 실제 결과인 7 보다 낮게 설정하면 잘못된 결과를 알려준다.

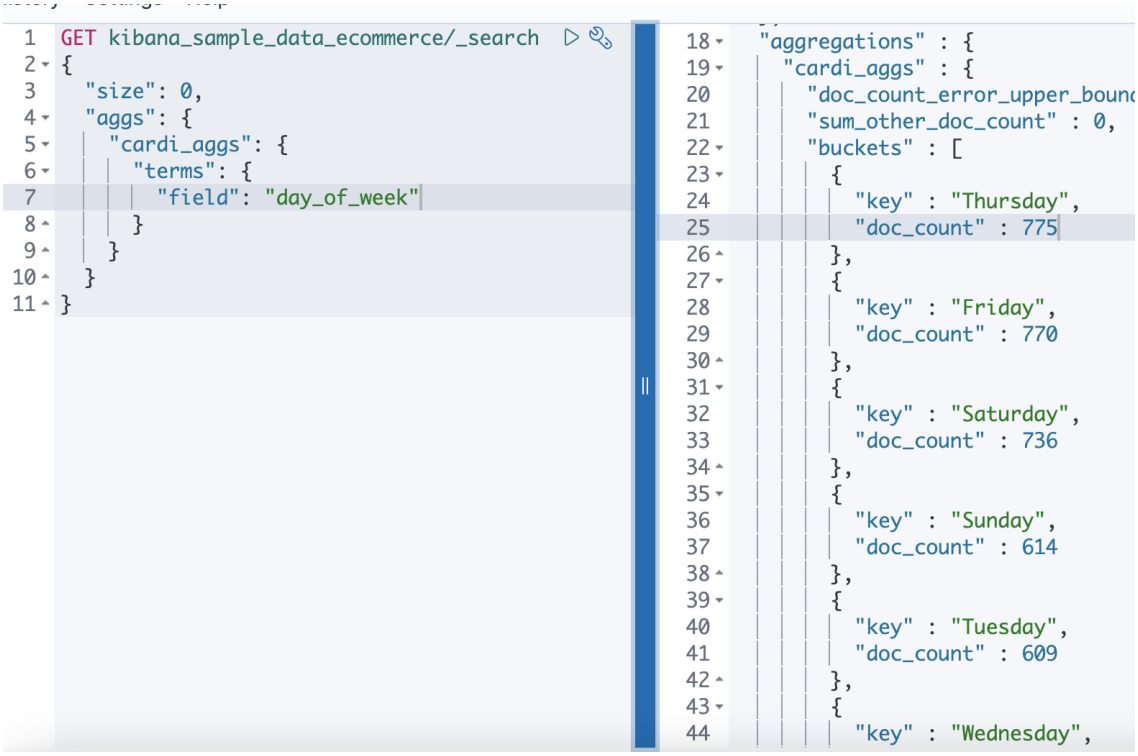
- SQL의 distinct count
- 범주형 데이터에서 유니크한 데이터를 확인하는 용도
- 하위 집계에서 사용되지 않는다.
- **percision_threshold**: 정확도 수치
  - 메모리 사용률 = percision_threshold * 8 bytes
  - 값이 크면: 정확도가 올라가고 시스템 리소스를 많이 소모
  - 값이 작으면: 정확도가 떨어지고 시스템 리소스를 덜 소모
  - 실제 정확한 결과를 모르기 때문에 값을 변경하면서 임계값을 찾는다.
  - 기본값 3000, 최대값 40000
  - 성능 향상을 위한 방안
    - cardinality 대상 필드를 hash 로 정의한다.
    - 단 cardinality 가 크거나 string 인 경우만 효과적이다.

▼ HyperLogLog++ 알고리즘 기반 동작

- 집합 내 중복되지 않은 항목의 개수를 세기 위한 알고리즘

- HyperLogLog 를 병렬처리하여 개선한 버전
- 완전히 정확한 값을 반환하진 않지만 5% 이내의 오차를 가지며, 정밀도를 직접 지정하면 오차율을 낮출 수 있다.
- 카디널리티가 낮은 (=중복을 제거한 항목 수가 적은) 집합일수록 100%에 가깝다

▼ VS 용어 집계 (=버킷 집계)

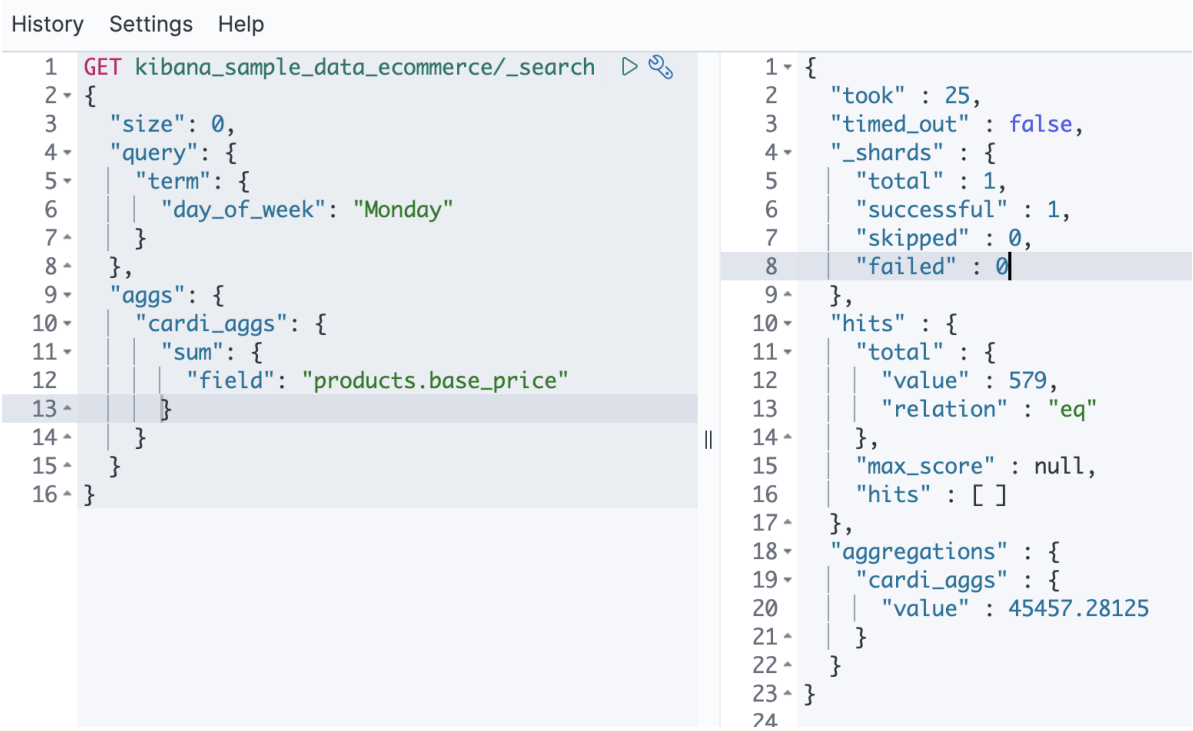


```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "cardi_aggs": {
      "terms": {
        "field": "day_of_week"
      }
    }
  }
}
```

```
"aggregations" : {
  "cardi_aggs" : {
    "doc_count_error_upper_bound
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "Thursday",
        "doc_count" : 775
      },
      {
        "key" : "Friday",
        "doc_count" : 770
      },
      {
        "key" : "Saturday",
        "doc_count" : 736
      },
      {
        "key" : "Sunday",
        "doc_count" : 614
      },
      {
        "key" : "Tuesday",
        "doc_count" : 609
      },
      {
        "key" : "Wednesday",
```

- 유니크한 필드 개수와 필드 값을 확인 할 수 있다.
  - doc_count: 각각의 유니크한 필드를 가진 도큐먼트 개수

▼ 검색 결과 내에서의 집계 = 검색 쿼리 + 집계




용어 쿼리로 Monday 도큐먼트만 골라낸다.

- 집계 전에 쿼리로 도큐먼트 범위를 제한한다.

## 5.3 버킷 집계

- 버킷: 도큐먼트가 분할되는 단위로 나뉜 각 그룹
- 특정 기준에 맞춰 도큐먼트를 그룹핑하는 역할

| | |
|---|---|
| histogram | 숫자 타입 필드를 일정 간격으로 분류한다. |
| data_histogram | 날짜/시간 타입 필드를 일정 날짜/시간 간격으로 분류한다 |
| range | 숫자 타입 필드를 사용자가 지정하는 날짜/시간 간격으로 분류 |
| date_range | 날짜/시간 타입 필드를 사용자가 지정하는 날짜/시간 간격으로 분류한다. |
| terms | 필드에 많이 나타나는 용어들을 기준으로 분류한다. |
| significant_terms | terms 버킷과 유사하지만 모든 값을 대상으로 하지 않고 인덱스 내 전체 문서 대비 현재 검색 조건에서 **통계적으로 유의미한** 값들을 기준으로 분류한다. |

| filters | 각 그룹에 포함시킬 문서의 조건을 직접 지정한다. 이때 조건은 일반적으로 검색에 사용되는 쿼리와 동일하다. |
|---|---|

▼ 히스토그램 집계

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "histogram_aggs": {
      "histogram": {
        "field": "products.base_price",
        "interval": 100
      }
    }
  }
}
```

```
  },
  "aggregations" : {
    "histogram_aggs" : {
      "buckets" : [
        {
          "key" : 0.0,
          "doc_count" : 4672
        },
        {
          "key" : 100.0,
          "doc_count" : 263
        },
        {
          "key" : 200.0,
          "doc_count" : 12
        },
        {
          "key" : 300.0,
          "doc_count" : 1
        },
        {
          "key" : 400.0,
          "doc_count" : 1
        },
        {
          "key" : 500.0,
          "doc_count" : 0
```

필드의 값을 100단위로 구분함

- key : 필드값이 0~(key+interval-1) 값 사이의 값임을 의미
- doc_count: 버킷에 속한 도큐먼트의 개수
- 각 **버킷의 범위를 동일하게** 지정할 수 밖에 없다.
  - 데이터가 특정 구간에 몰려 있거나 데이터 편차가 크면 비효율적이다.

▼ 범위 집계

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "range_aggs": {
      "range": {
        "field": "products.base_price",
        "ranges": [
          {"from": 0, "to": 50},
          {"from": 50, "to": 100},
          {"from": 100, "to": 200}
        ]
      }
    }
  }
}
```

```
  },
  "aggregations" : {
    "range_aggs" : {
      "buckets" : [
        {
          "key" : "0.0-50.0",
          "from" : 0.0,
          "to" : 50.0,
          "doc_count" : 4341
        },
        {
          "key" : "50.0-100.0",
          "from" : 50.0,
          "to" : 100.0,
          "doc_count" : 1902
        },
        {
          "key" : "100.0-200.0",
          "from" : 100.0,
          "to" : 200.0,
          "doc_count" : 263
        }
```

- 데이터 편차가 클때 사용
- 각 버킷의 범위를 사용자가 직접 설정한다.
- 범위에 따라 도큐먼트 총합이 다를 수 있다.
    - 필드가 배열로 이루어졌을 때, 구간에 속할때마다 카운트하기 때문이다.

▼ 용어 집계

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "term_aggs": {
      "terms": {
        "field": "day_of_week",
        "size": 6
      }
    }
  }
}
```

```
  "doc_count_error_upper_bound" : 0,
  "sum_other_doc_count" : 579,
  "buckets" : [
    {
      "key" : "Thursday",
      "doc_count" : 775
    },
    {
      "key" : "Friday",
      "doc_count" : 770
    },
    {
      "key" : "Saturday",
      "doc_count" : 736
    },
    {
      "key" : "Sunday",
      "doc_count" : 614
    },
    {
      "key" : "Tuesday",
      "doc_count" : 609
    },
    {
      "key" : "Wednesday",
      "doc_count" : 592
```

size:6 → 상위 6개만 보인다.

- 용어집계 안의 size 파라미터: 기본값 10, 만들 버킷 수 지정
  - size 수가 생성되는 버킷보다 작으면 size만큼만 보인다.
- doc_count_error_upper_bound: 버킷이 잠재적으로 카운트하지 못할 도큐먼트의 수
- sum_other_doc_count: 버킷에는 있지만 size 때문에 보이지 않는 도큐먼트 수
- 용어 집계가 정확하지 않은 이유
  - 분산시스템의 집계 과정에서 발생하는 잠재적인 오류 가능성 때문
  - 모든 도큐먼트를 가져와 한 번에 집계하는 것이 아니라 분산되어 있는 개별 노드단에서 먼저 집계를 하고 그 결과를 취합해 다시 집계를 하기 때문
- 용어 집계 정확성 높이기
  - 용어 집계를 요청할 때 `show_term_doc_count_error` 파라미터를 추가한다. → 버킷마다 doc_count_errror_uppper_bound값을 확인할 수 있다.
  - 샤드 크기 조정 파라미터 shard_size : 용어 집계 과정에서 개별 샤드에서 집계를 위해 처리하는 개수
    - 샤드 크기가 클수록 정확도가 올라간다. (=리소스 사용량이 올라가 성능이 떨어질 수 있다.)

- 보편적인 샤드 크기 = size(버킷수) * 1.5 + 10

- 이상 값이 나오면 샤드 크기 파라미터를 늘려본다.

## 5.4 집계의 조합

▼ 버킷 집계 + 매트릭 집계 ⇒ 그룹별 통계 계산

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,

  "aggs": {
    "term_aggs": {
      "terms": {
        "field": "day_of_week",
        "size": 5
      },
      "aggs": {
        "avg_aggs": {
          "avg": {
            "field": "products.base_price"
          }
        }
      }
    }
  }
}
```

```
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 1171,
    "buckets" : [
      {
        "key" : "Thursday",
        "doc_count" : 775,
        "avg_aggs" : {
          "value" : 34.68040897713688
        }
      },
      {
        "key" : "Friday",
        "doc_count" : 770,
        "avg_aggs" : {
          "value" : 34.665464386512184
        }
      },
      {
        "key" : "Saturday",
        "doc_count" : 736,
        "avg_aggs" : {
          "value" : 34.35796178343949
        }
```

용어 집계로 상위 5개의 버킷을 만들고 각각의 버킷 내부에서 서치한 필드의 평균값을 구한다.

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "term_aggs": {
      "terms": {
        "field": "day_of_week",
        "size": 5
      },
      "aggs": {
        "avg_aggs": {
          "avg": {
            "field": "products.base_price"
          }
        },
        "sum_aggs": {
          "sum": {
            "field": "products.base_price"
          }
        }
      }
    }
  }
}
```

```
      "buckets" : [
        {
          "key" : "Thursday",
          "doc_count" : 775,
          "avg_aggs" : {
            "value" : 34.68040897713688
          },
          "sum_aggs" : {
            "value" : 58020.32421875
          }
        },
        {
          "key" : "Friday",
          "doc_count" : 770,
          "avg_aggs" : {
            "value" : 34.665464386512184
          },
          "sum_aggs" : {
            "value" : 58341.9765625
          }
        },
        {
          "key" : "Saturday",
          "doc_count" : 736,
          "avg_aggs" : {
            "value" : 34.35796178343949
          },
```

버킷 집계 후 다수의 매트릭 집계 요청

- 버킷 집계 → 매트릭 집계(들) : 버킷별 통계 집계

▼ 서브 버킷 집계 = 버킷안의 버킷 집계 (트리)

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "histogram_aggs": {
      "histogram": {
        "field": "products.base_price",
        "interval": 100
      },
      "aggs": {
        "terms_aggs": {
          "terms": {
            "field": "day_of_week",
            "size": 2
          }
        }
      }
    }
  }
}
```

```
  "aggregations" : {
    "histogram_aggs" : {
      "buckets" : [
        {
          "key" : 0.0,
          "doc_count" : 4672,
          "terms_aggs" : {
            "doc_count_error_upper_bound" : 0,
            "sum_other_doc_count" : 3128,
            "buckets" : [
              {
                "key" : "Thursday",
                "doc_count" : 775
              },
              {
                "key" : "Friday",
                "doc_count" : 769
              }
            ]
          }
        },
        {
          "key" : 100.0,
          "doc_count" : 263,
          "terms_aggs" : {
            "doc_count_error_upper_bound" : 0,
            "sum_other_doc_count" : 176,
```

히스토그램 버킷 집계로 필드를 100단위로 구분한 후, 버킷 내부에서 유니크한 값 기준으로 다시 버킷을 나눈다.

- 서브 버킷은 2단계를 초과해서 만들지 않는 편이 좋다.
    - 버킷의 수가 n^(서브버킷수) 만큼 늘어남 → 성능 저하 → 클러스터에 부하를 준다

## 5.5 파이프라인 집계

- 이전 집계 결과를 입력받아 다시 집계하는 방식
- 반드시 버킷 경로(`bucket_path`)를 입력해야 한다.
    - 버킷 경로에서 '>' 는 하위 집계 경로를 나타낸다.

▼ 부모 집계

- 기존 집계 내부에서 작성한다
- 부모 집계 파이프라인 종류

| | |
|---|---|
| derivative | 기존 집계의 미분을 구한다 |
| cumulative_sum | 기존 집계의 누적합을 구한다. |

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "histogram_aggs": {
      "histogram": {
        "field": "products.base_price",
        "interval": 100
      },
      "aggs": {
        "sum_aggs": {
          "sum": {
            "field": "taxful_total_price"
          }
        },
        "cum_cum": {
          "cumulative_sum": {
            "buckets_path": "sum_aggs"
          }
        }
      }
    }
  }
}
```

```
      "doc_count" : 263,
      "sum_aggs" : {
        "value" : 44002.0
      },
      "cum_cum" : {
        "value" : 392126.12890625
      }
    },
    {
      "key" : 200.0,
      "doc_count" : 12,
      "sum_aggs" : {
        "value" : 3163.0
      },
      "cum_cum" : {
        "value" : 395289.12890625
      }
    },
    {
      "key" : 300.0,
      "doc_count" : 1,
      "sum_aggs" : {
        "value" : 2250.0
      },
      "cum_cum" : {
```

누적합을 구하는 부모 집계, 히스토그램 버킷집계 + 합계 집계

- 단독으로 사용할 수 없고 반드시 먼저 다른 집계가 있어야 부모 집계가 그 결과를 사용한다.

- 결과값도 기존 집계 내부에서 나타난다.

▼ 형제 집계

- 기존 집계 외부에서 작성한다
- 형제 집계 파이프라인 종류

| | |
|---|---|
| min_bucket | 기존 집계 중 최솟값을 구한다 |
| max_bucket | 기존 집계 중 최댓값을 구한다 |
| avg_bucket | 기존 집계 중 평균값을 구한다. |
| sum_bucket | 기존 집계 중 총합을 구한다. |
| stat_bucket | 기존 집계의 min, max, sum, count, avg를 구한다. |
| percetile_bucket | 기존 집계의 백분윗값을 구한다. |
| moving_avg | 기존 집계의 이동 평균을 구한다. 단 기존 집계는 순차적인 데이터 구조여야 한다. |

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "term_aggs": {
      "terms": {
        "field": "day_of_week",
        "size": 2
      },
      "aggs": {
        "sum_aggs": {
          "sum": {
            "field": "products.base_price"
          }
        }
      }
    },
    "sum_total_price" :{
      "sum_bucket": {
        "buckets_path": "term_aggs>sum_aggs"
      }
    }
  }
}
```

```
  "aggregations" : {
    "term_aggs" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 3130,
      "buckets" : [
        {
          "key" : "Thursday",
          "doc_count" : 775,
          "sum_aggs" : {
            "value" : 58020.32421875
          }
        },
        {
          "key" : "Friday",
          "doc_count" : 770,
          "sum_aggs" : {
            "value" : 58341.9765625
          }
        }
      ]
    },
    "sum_total_price" : {
      "value" : 116362.30078125
    }
  }
```

형제 집계로 기존 버킷별 합을 구한 집계를 다시 합친다.

- 버킷 경로가 term_aggs → sum_aggs인 합계를 구한다.
- 기존 집계 외부에서 결과를 보여준다.

- 집계 순서를 어떤 식으로 하느냐에 따라 결과가 다르다.
- 집계 결과를 페이징 처럼 받으려면? → bucket_sort: {”from”: ?, “size”: ?}