

案例中所用到的数据

```
1 {
2   "took": 45,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 3,
12    "max_score": 1,
13    "hits": [
14      {
15        "_index": "ecommerce",
16        "_type": "product",
17        "_id": "2",
18        "_score": 1,
19        "_source": {
20          "name": "xiaomi 6",
21          "desc": "xiaomi6 jiushi kuai!!",
22          "price": 2399,
23          "tag": [
24            "xiaomi",
25            "shouji"
26          ]
27        }
28      },
29      {
30        "_index": "ecommerce",
31        "_type": "product",
32        "_id": "1",
33        "_score": 1,
34        "_source": {
35          "name": "xiaomi 7",
36          "desc": "xiaomi shouji jiushi kuai! quanmianping",
37          "price": 2699,
38          "tag": [
39            "xiaomi",
40            "shouji",
41            "quanmianping"
42          ]

```

```

43     }
44 },
45 {
46     "_index": "ecommerce",
47     "_type": "product",
48     "_id": "3",
49     "_score": 1,
50     "_source": {
51         "name": "huawei mate 10",
52         "desc": "huawei shouji niandu niandu shangwu shouji qijian",
53         "price": 4399,
54         "tag": [
55             "huawei",
56             "shouji"
57         ]
58     }
59 }
60 ]
61 }
62 }

```

实战案例分析：

案例需求一： 计算每个 tag 下， 商品数量

```

1  GET /ecommerce/product/_search
2  {
3      "size": 0,           // 展示 doc 的数量 ， 如果只是做徐鹤分析， 设置 size: 0， 可以不
                           // 展示数据。
4      "aggs": {           //开始聚合
5          "group_by_tag": { //聚合的名称（可以自己随便取）
6              "terms": {    // 聚合函数的名字， 这里是 terms 分类聚合函数
7                  "field": "tag" // 按照 tag field 进行分类
8              }
9          }
10     }
11 }
12

```

返回结果：

```
1  {
2    "took": 307,
3    "timed_out": false,
4    "_shards": {
5      "total": 5,
6      "successful": 5,
7      "skipped": 0,
8      "failed": 0
9    },
10   "hits": {
11     "total": 3,
12     "max_score": 1,
13     "hits": [
14       {
15         "_index": "ecommerce",
16         "_type": "product",
17         "_id": "2",
18         "_score": 1,
19         "_source": {
20           "name": "xiaomi 6",
21           "desc": "xiaomi6 jiushi kuai!!",
22           "price": 2399,
23           "tag": [
24             "xiaomi",
25             "shouji"
26           ]
27         }
28       },
29       {
30         "_index": "ecommerce",
31         "_type": "product",
32         "_id": "1",
33         "_score": 1,
34         "_source": {
35           "name": "xiaomi 7",
36           "desc": "xiaomi shouji jiushi kuai! quanmianping",
37           "price": 2699,
38           "tag": [
39             "xiaomi",
40             "shouji",
41             "quanmianping"
42           ]
43         }
44       },
45       {
```

```

46     "_index": "ecommerce",
47     "_type": "product",
48     "_id": "3",
49     "_score": 1,
50     "_source": {
51         "name": "huawei mate 10",
52         "desc": "huawei shouji niandu niandu shangwu shouji qijian",
53         "price": 4399,
54         "tag": [
55             "huawei",
56             "shouji"
57         ]
58     }
59 }
60 ]
61 },
62 "aggregations": {
63     "group_by_tag": {
64         "doc_count_error_upper_bound": 0,
65         "sum_other_doc_count": 0,
66         "buckets": [          // 分类结果 buckets 结果桶
67             {
68                 "key": "shouji",
69                 "doc_count": 3
70             },
71             {
72                 "key": "xiaomi",
73                 "doc_count": 2
74             },
75             {
76                 "key": "huawei",
77                 "doc_count": 1
78             },
79             {
80                 "key": "quanmianping",
81                 "doc_count": 1
82             }
83         ]
84     }
85 }
86 }

```

案例需求二：对名称中包含“xiaomi”商品，计算每个 tag 下的商品的数量

```
1 GET /ecommerce/product/_search
2 {
3   "size": 0,
4   "query": {           // 在聚合分析之前，先进行条件查询
5     "match": {
6       "name": "xiaomi"
7     }
8   },
9   "aggs": {
10    "group_by_tags": {
11      "terms": {
12        "field": "tag"
13      }
14    }
15  }
16 }
```

返回结果：

```
1 {
2   "took": 293,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 2,
12    "max_score": 0,
13    "hits": []
14  },
15  "aggregations": {
16    "group_by_tags": {
17      "doc_count_error_upper_bound": 0,
18      "sum_other_doc_count": 0,
19      "buckets": [
20        {
21          "key": "shouji",
22          "doc_count": 2
23        },
24        {
25          "key": "xiaomi",
26          "doc_count": 2
27        }
28      ]
29    }
30  }
```

```

27     },
28     {
29         "key": "quanmianping",
30         "doc_count": 1
31     }
32 ]
33 }
34 }
35 }

```

案例需求三：先分组，再算每组的平均值，计算每个 tag 下的商品的平均价格

```

1 GET /ecommerce/product/_search
2 {
3     "size": 0,
4     "aggs": {
5         "group_by_tag": {
6             "terms": {
7                 "field": "tag"
8             },
9             "aggs": { //在前面进行了 按照tag 分组的聚合分析之后， 在 terms 并列层级添加
                        第二个聚合分析，avg，在每组中进行 avg 聚合分析
10                "price_avg": {
11                    "avg": {
12                        "field": "price"
13                    }
14                }
15            }
16        }
17    }
18 }

```

返回结果：

```

1 {
2     "took": 86,
3     "timed_out": false,
4     "_shards": {
5         "total": 5,
6         "successful": 5,

```

```

7     "skipped": 0,
8     "failed": 0
9 },
10 "hits": {
11     "total": 3,
12     "max_score": 0,
13     "hits": []
14 },
15 "aggregations": {
16     "group_by_tag": {
17         "doc_count_error_upper_bound": 0,
18         "sum_other_doc_count": 0,
19         "buckets": [
20             {
21                 "key": "shouji",           //第一个tag “shouji” 平均值 3165.666
22                 "doc_count": 3,
23                 "price_avg": {
24                     "value": 3165.6666666666665
25                 }
26             },
27             {
28                 "key": "xiaomi",           //第二个tag “xiaomi” 平均值 2549
29                 "doc_count": 2,
30                 "price_avg": {
31                     "value": 2549
32                 }
33             },
34             {
35                 "key": "huawei",           //第三个tag “huawei” 平均值 4399
36                 "doc_count": 1,
37                 "price_avg": {
38                     "value": 4399
39                 }
40             },
41             {
42                 "key": "quanmianping",     //第三个 tag “quanmianping” 平均值 2699
43                 "doc_count": 1,
44                 "price_avg": {
45                     "value": 2699
46                 }
47             }
48         ]
49     }
50 }
51 }

```

案例需求四：计算每个 tag 下的商品平均价格，并按照降序排列

```
1 GET /ecommerce/product/_search
2 {
3   "size": 0,
4   "aggs": {
5     "group_by_tag": {
6       "terms": {
7         "field": "tag",
8         "order": {                                //进行排序， 按照下面聚合分析的结果，这里是按照
下面 avg_price 求价格平均数的 结果降序
9         "avg_price": "desc"
10      }
11    },
12    "aggs": {
13      "avg_price": {
14        "avg": {
15          "field": "price"
16        }
17      }
18    }
19  }
20 }
21 }
22
```

返回结果：

```
1 {
2   "took": 22,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 3,
12    "max_score": 0,
13    "hits": []
14  },

```



```

15 "aggregations": {
16   "group_by_tag": {
17     "doc_count_error_upper_bound": 0,
18     "sum_other_doc_count": 0,
19     "buckets": [
20       {
21         "key": "huawei",
22         "doc_count": 1,
23         "avg_price": {
24           "value": 4399
25         }
26       },
27       {
28         "key": "shouji",
29         "doc_count": 3,
30         "avg_price": {
31           "value": 3165.6666666666665
32         }
33       },
34       {
35         "key": "quanmianping",
36         "doc_count": 1,
37         "avg_price": {
38           "value": 2699
39         }
40       },
41       {
42         "key": "xiaomi",
43         "doc_count": 2,
44         "avg_price": {
45           "value": 2549
46         }
47       }
48     ]
49   }
50 }
51 }

```

案例需求五：先按照价格范围区间进行分组，然后在每组内再按照 tag 进行分组，最后再计算每组的平均价格

```

1 GET /ecommerce/product/_search
2 {

```

```

3  "size": 0,
4  "aggs": {
5    "range_by_price": {
6      "range": {                                // 按照 price 进行范围分组
7        "field": "price",
8        "ranges": [
9          {
10           "from": 1000,
11           "to": 2000
12         },
13         {
14           "from": 2000,
15           "to": 3000
16         },
17         {
18           "from": 3000,
19           "to": 4000
20         },
21         {
22           "from": 4000,
23           "to": 5000
24         }
25       ]
26     },
27     "aggs": {                                // 按照 tag 进行分组
28       "group_by_tag": {
29         "terms": {
30           "field": "tag"
31         },
32         "aggs": {                            // 在计算 price 的平均值
33           "price_avg": {
34             "avg": {
35               "field": "price"
36             }
37           }
38         }
39       }
40     }
41   }
42 }
43 }

```

结果:

```

1 {
2   "took": 25,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 3,
12    "max_score": 0,
13    "hits": []
14  },
15  "aggregations": {
16    "range_by_price": {
17      "buckets": [                                // 按照范围分组桶
18        {
19          "key": "1000.0-2000.0",
20          "from": 1000,
21          "to": 2000,
22          "doc_count": 0,
23          "group_by_tag": {
24            "doc_count_error_upper_bound": 0,
25            "sum_other_doc_count": 0,
26            "buckets": []
27          }
28        },
29        {
30          "key": "2000.0-3000.0",
31          "from": 2000,
32          "to": 3000,
33          "doc_count": 2,
34          "group_by_tag": {
35            "doc_count_error_upper_bound": 0,
36            "sum_other_doc_count": 0,
37            "buckets": [                            //每个分组中按照 tag 分组桶
38              {
39                "key": "shouji",
40                "doc_count": 2,
41                "price_avg": {
42                  "value": 2549                      //计算平均值
43                }
44              },
45              {

```

```
46         "key": "xiaomi",
47         "doc_count": 2,
48         "price_avg": {
49             "value": 2549
50         }
51     },
52     {
53         "key": "quanmianping",
54         "doc_count": 1,
55         "price_avg": {
56             "value": 2699
57         }
58     }
59 ]
60 }
61 },
62 {
63     "key": "3000.0-4000.0",
64     "from": 3000,
65     "to": 4000,
66     "doc_count": 0,
67     "group_by_tag": {
68         "doc_count_error_upper_bound": 0,
69         "sum_other_doc_count": 0,
70         "buckets": []
71     }
72 },
73 {
74     "key": "4000.0-5000.0",
75     "from": 4000,
76     "to": 5000,
77     "doc_count": 1,
78     "group_by_tag": {
79         "doc_count_error_upper_bound": 0,
80         "sum_other_doc_count": 0,
81         "buckets": [
82             {
83                 "key": "huawei",
84                 "doc_count": 1,
85                 "price_avg": {
86                     "value": 4399
87                 }
88             },
89             {
90                 "key": "shouji",
91                 "doc_count": 1,
```

```
92         "price_avg": {
93             "value": 4399
94         }
95     }
96 ]
97 }
98 }
99 ]
100 }
101 }
102 }
```

注：如果第一次使用聚合函数 是要设置正排索引 所以要将 文本的 field 的 fielddata 属性设置为 true

```
1 PUT /ecommerce/_mapping/product
2 {
3     "properties":{
4         "tag":{
5             "type": "text",
6             "fielddata": "true"
7         }
8     }
9 }
10
```