# 深入ElasticSearch分布式搜索引擎

## 前言

同学们，前两天的内容已经学会了ES的基本用法和搜索相关功能！今天继续深入ES的搜索功能，学习更复杂的用法：

今天内容主要四部分：

### 1：数据聚合

数据聚合可以帮助我们对**海量数据做统计、分析**，如果结合Kibana，还可以形成可视化的图形报表，让用户一目了然看到数据的结果，更方便地去做数据分析。

### 2：自动补全

其实就是在用户输入时，根据用户输入的部分关键字信息去自动补全和提示用户真正想要搜索的内容，提高用户体验，自动补全可以认为是搜索的必备功能。

### 3：数据同步

分析MySQL和ES数据双写一致性问题，以及对应的解决方案。

### 4：集群

了解ES集群结构，以及集群中不同ES节点的角色和职责，最后搭建企业级高可用集群。

## 数据聚合（用于数据的统计、分析）

了解包聚合的种类，看一看聚合到底有那些；然后再分别利用DSL和RestAPI实现聚合功能。

### 聚合（aggregations）的分类

聚合（aggregations）可以实现对文档数据的统计、分析、运算。这么一聊你就会觉得聚合不陌生了，可以类比与Mysql中的聚合，即在MySQL也有聚合功能，在MySQL里有一种东西叫聚合函数，比如我们可以在MySQL中可以用聚合函数求平均值、最大值、最小值、求和等等，当然了，MySQL中的聚合函数常常要结合group by分组一起使用。同样ES也具备MySQL类似的聚合功能，并且还做了更丰富、更完善的其他的聚合功能。ES中常见的聚合有三大类，分别是：桶聚合、度量聚合、管道聚合！

**记住了，聚合的字段一定是不分词的，字段数据类型可以是keyword，可以是数值、可以是日期、可以是布尔，但一定不会是text。这是做聚合时对字段数据类型的限制！**

ES中聚合常见的有三类：

#### 桶（bucket）聚合：

用来对文档做分组（类型MySQL中group by），为什么叫桶？可以用垃圾桶来理解，垃圾桶都会贴上标识：干回湿害，代表不同的垃圾，如果说你现在提着垃圾过来了，你的垃圾类型不同就要进到不同的垃圾桶里去，就是垃圾桶对垃圾做分类。所以桶就是起一个分类、分组的作用！因此bucket聚合就是对数据做分组！

常见的桶聚合有下面两种：

**TermAggregation**：按照文档字段值做分组，字段值一样放在一个桶里，这个和MySQL中的group by效果是一致的。从这你可以看出，我做桶聚合是可以针对字符串类型字段对文档做分组，但是你叫term，是根据字段值，或者叫根据词条值，所以这个文档字段将来绝对不能分词，如果你将来分词了，一个字段你分成有好几个词条，将来怎么给你这个文档分组，没法分！

**Date Histogram**：按照日期阶梯分组，例如一周为一组，或者一月 为一组、季度、年等，这样时间在同一个时间段内的数据放在一个桶里去。这个功能如果你要用MySQL实现就复杂多了，在ES中只需要一个Date Histogram。

当然桶聚合不止上面两个，还有很多，只是上面两个最常用！

#### 度量（metric）聚合：

度量顾名思义就是做运算，用以计算一些值，比如：最大值、最小值、平均值等，注意了：做度量聚合，肯定针对的是数值字段，不是字符串字段。

Avg：求平均值

Max：求最大值

Min：求最小值

Stats：同时求max、min、avg、sum等。

#### 管道（pipeline）聚合：

这个聚合跟前面两个聚合不太一样，管道聚合不对文档当中的字段做聚合，而是对其他聚合的结果做聚合。比方说现在先对酒店数据按照品牌做了分组（桶聚合）后，然后想算一下不同品牌酒店的价格平均值（度量聚合），算完平均值后，又要对平均值做排序或者在平均值里求最高或最低的，这个时候其实就是对度量聚合的结果再次聚合了，那么就要用到管道聚合。即：

1：相对酒店文档按照品牌字段分组（桶聚合）；

2：然后计算不同品牌酒店均价（度量聚合）

3：再按均价由高到低排序（管道聚合）

管道聚合用得相对少。主要还是桶聚合和度量聚合！

#### 总结：

##### 什么是聚合？

聚合就是对索引库文档数据做统计、分析和运算。

##### 聚合常见的种类有？

Bucket桶聚合：对文档数据做分组，并统计每组数量

Metric度量聚合：对文档数据做计算，例如avg

Pipeline管道聚合：基于其他聚合结果再做聚合

##### 参与聚合的字段类型必须是？

一定不能是text类型，可以是Keyword、数值、日期、布尔

## DSL实现聚合

了解了聚合的分类之后，下面我们一起来动手实践，看看如何用DSL来实现聚合！

### Bucket桶聚合

需求：我们要统计hotel索引库中，所有酒店数据中的酒店品牌有几种？

这个需求在MySQL中你肯定会做啊：group by 品牌！你就能查出所有的酒店品牌了。

对应到ES中，这种对数据分组的需求就是用Bucket聚合来实现的。此时可以根据酒店品牌的名称来对酒店数据做桶聚合！那具体是桶聚合中的哪一种呢？是TermAggregation还是Date Histogram？显然是TermAggregation！因为你根据酒店的名称，其实就是根据字段的值做分组！而TermAggregation恰好就是根据字段值做分组，跟数据库中的group by是一样的。

#### DSL基本语法

**GET /hotel/\_search**

**{**

**"size": 0, // 设置size为0，那么结果中不包含文档，只包含聚合结果**

**"aggs": { // 定义聚合，聚合的简写，在里面可以定义多个聚合**

**"brandAgg": { // 给聚合起个名字**

**"terms": { // 聚合的类型，按照品牌值聚合，所以选择桶聚合中的term**

**"field": "brand", // 参与聚合的字段**

**"size": 20 // 希望获取的聚合结果数量，不指定默认是10**

**}**

**}**

**}**

**}**

size值为0，就是说显示的文档数是0，这样结果中就不包含文档数据了，只包含聚合结果，因为我的目的是看聚合嘛！

aggs就是定义聚合了，是聚合aggregations的缩写，也就是说aggs里可以定义好多个聚合的，实例中只定义了一个聚合。

brandAgg就是聚合名称，聚合名称是自定义的，实例中的这个brandAgg就是品牌的聚合的意思。

termsj就是聚合的类型，这里选择的是TermAggregation。

field就是参与聚合的字段。

size指定聚合结果的数据，不指定默认就是10，也就是说你根据品牌对酒店数据做聚合，聚合统计出来如果酒店品牌有一两百种，但是size默认是10，所以结果只显示前十个！

所以一个聚合最重要的就是四个东西：

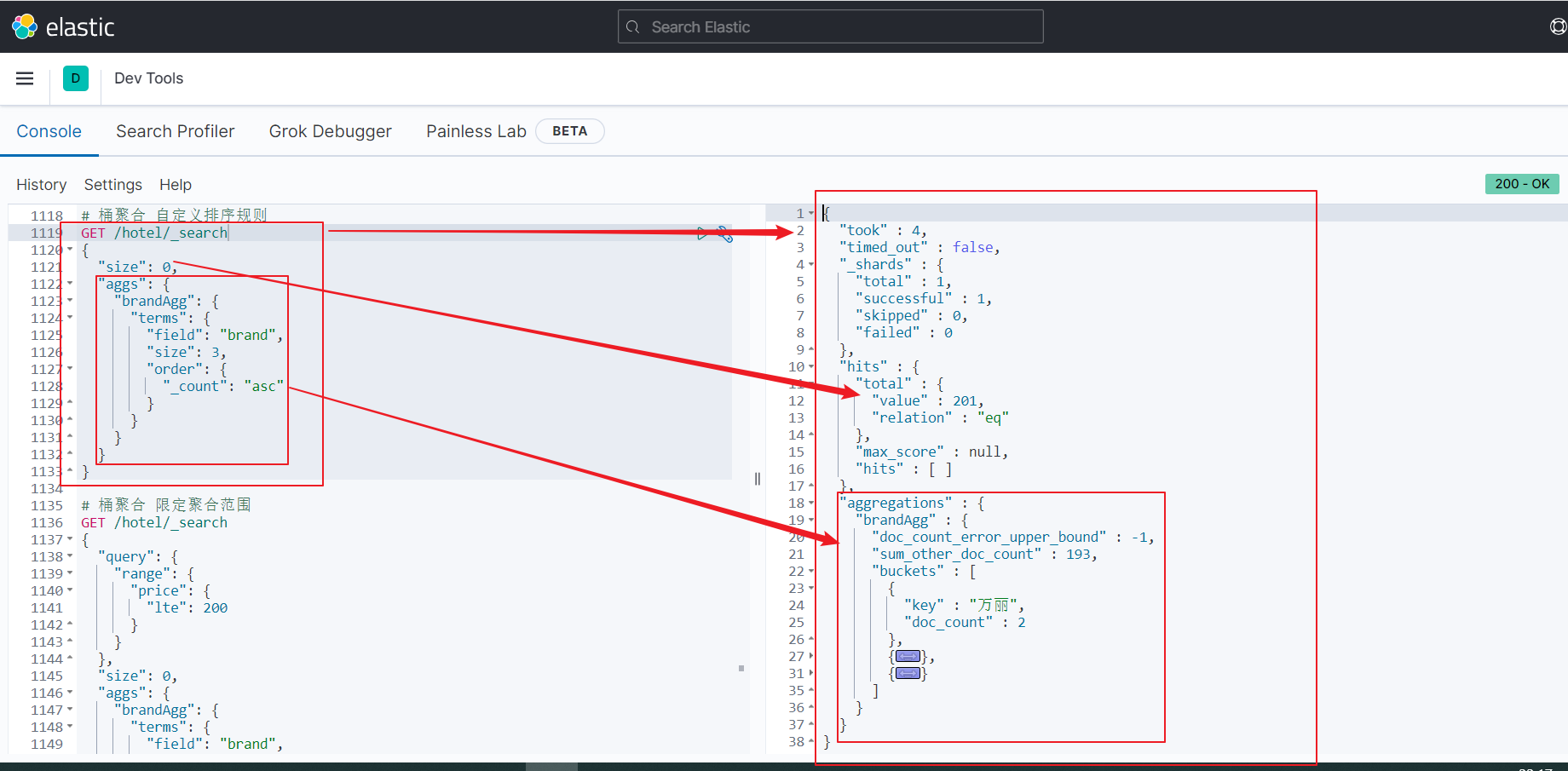
1：你给聚合取什么名称？

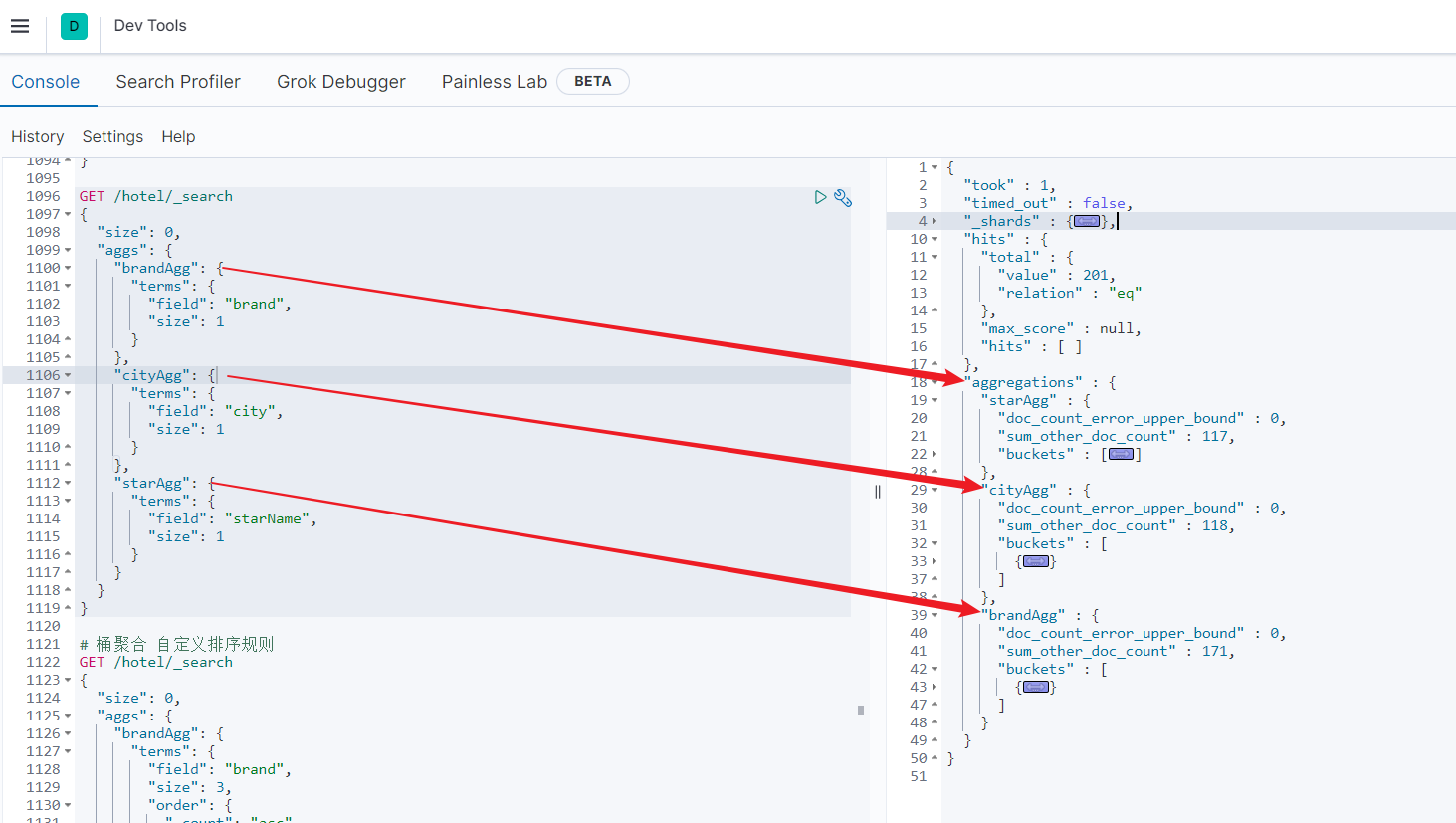
1：聚合类型是什么？

2：要对什么字段做聚合？

3：希望获取的聚合结果数是多少？

#### 基本语法演示：





size为0，所以hits种的hits数组是空的；

DSL中的aggs对应结果中的aggregations；

DSL中的brandAgg对应结果中的brandAgg，所以将来你在DSL中的aggs中定义多个不同的属性，aggregations中就会有多个不同的聚合结果；

聚合结果brandAgg内有个buckets属性，聚合是桶聚合，所以结果就是返回多个桶，这里就是品牌的桶！所以buckets是个数组。

每个桶有两个属性，key就是你聚合字段的值，doc\_count就是这个桶有多少文档数量；

聚合结果默认的排序规则是桶中文档数量越多的排在前面；

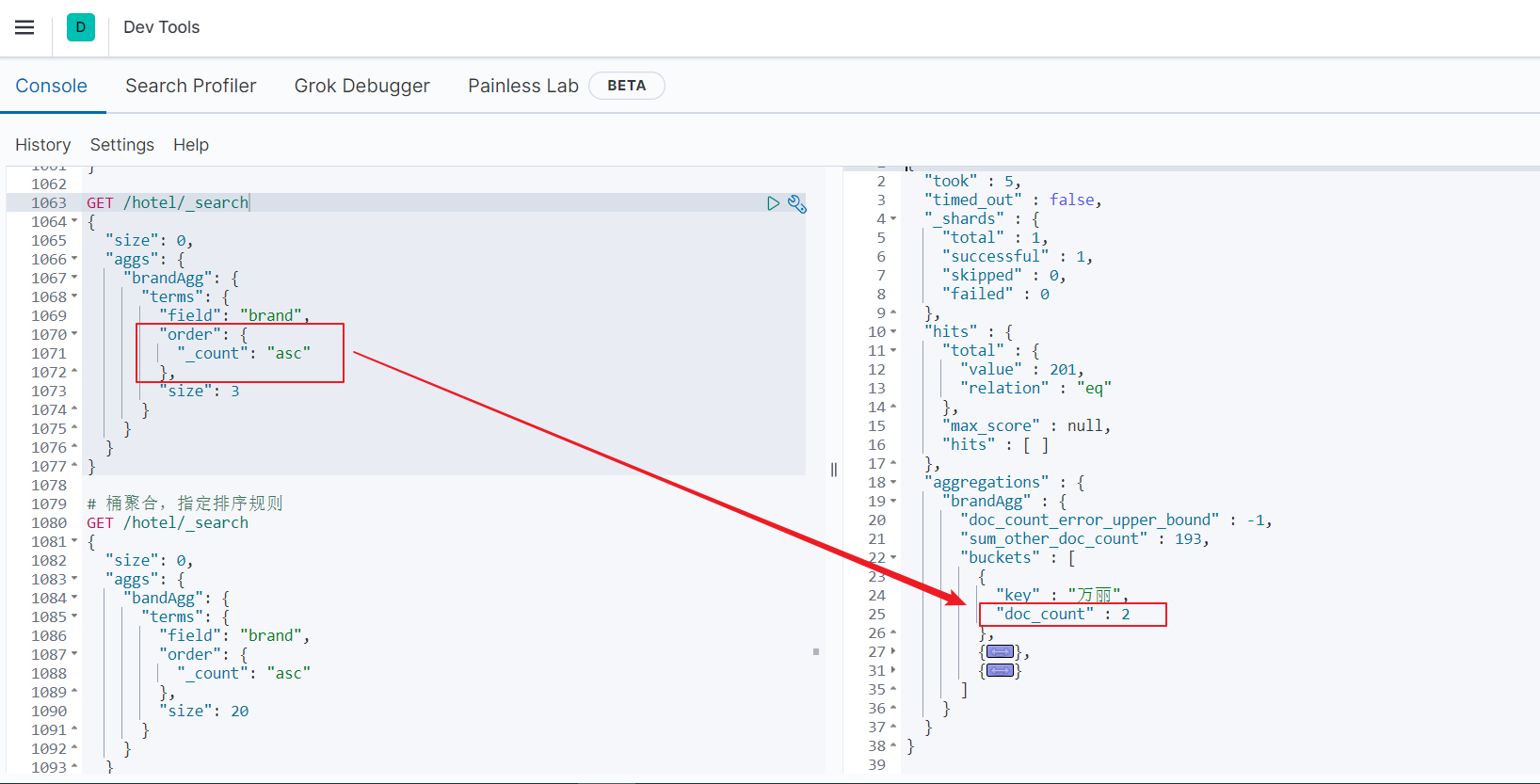
aggs中size为3，所以根据品牌聚合后结果返回了3个！

#### 指定排序规则

默认情况下，bucket聚合会统计bucket内的文档数量，记为\_count，并且按照\_count降序排序，我们可以修改结果排序方式，只需要在聚合中添加order属性，并且指定排序的字段\_count和排序方式，如下：



你看你看，可以自定义排序规则呢：一盘情况下就按默认的降序就可以了，都不会去改！

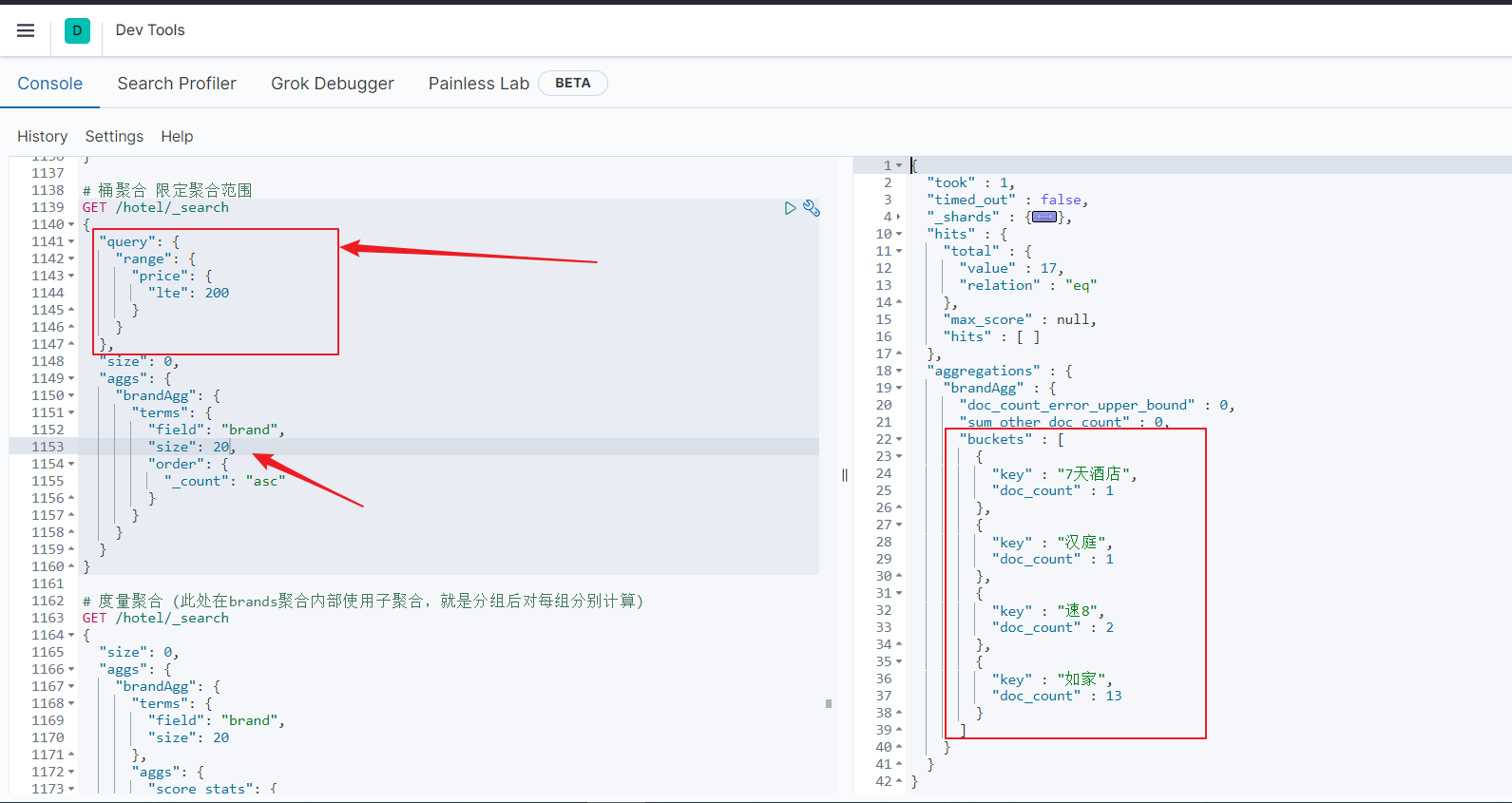


#### 限定聚合范围

以上我们做的聚合搜索其实是对整个索引库的数据做的聚合搜索，如果说索引库数据是海量的，比如上亿条，那么这个聚合对内存的消耗是巨大的，因此我们可以去限定这个聚合搜索的范围：默认情况下，bucket聚合是对索引库的所有文档做聚合，我们可以限定要聚合的文档范围，只要加query条件即可：



其实就是利用query，先对索引库做筛选，然后再对筛选出来的搜索结果做聚合操作，和数据库中MySQL中的select \* from 表名 where xxx group by xxx的where一样！



可以看到，使用query限定了聚合范围后，因为要聚合的数据少了，就算size的值改成了20，聚合结果中每个桶的文档数量加起来也只有17个，说明使用query只筛选出了17条文档！

#### 总结

aggs代表聚合，与query同级，此时query的作用是？

限定聚合的文档范围

聚合必须的三要素？

1：聚合名称

2：聚合类型

3：聚合字段

聚合可以配置的属性有？

1：size：指定聚合结果的数量，默认是10；

2：order：指定聚合结果的排序方式；

3：field：指定聚合字段

### metric度量聚合

前边学习了bucket桶聚合的DSL语法，现在来学习metric度量聚合的DSL语法。

#### DSL基本语法

需求：例如我们要获取每个酒店品牌的用户评分的min、max、avg等值。

分析：用户评分是酒店的score字段，是数值类型，求最小值、最大值、平均值，其实就是一种度量，即metric聚合，而且同时要求多个，那么可以用metric度量聚合中的stats。

需要注意的是：“获取每个品牌的用户评分的min、max、avg等值”，而不是整个索引库中所有酒店的评分的最小值、最大值、平均值，如果说直接做度量聚合的话，那么是对整个索引库最大值，最小值计算了，显然与需求不符合。我们要在品牌内去做限定，因此，必须要和bucket聚合结合使用，语法如下：



可以看到是做了聚合的嵌套，即现在外层做了按品牌桶聚合，也就是先对酒店品牌做桶聚合，然后在品牌桶里面对桶里的文档去计算评分的最大值、最小值、平均分。你会发现除了上节讲的brandAgg内部锁了一个属性：aggs，这个aggs就是对brandAgg的聚合结果基础上又做了一次聚合！这就是聚合的嵌套！aggs也是聚合三要素：聚合名称、聚合类型、聚合字段！

#### 基本语法演示

此处在brands聚合内部使用子聚合，就是分组后对每组分别计算

**GET /hotel/\_search**

**{**

**"size": 0,**

**"aggs": {**

**"brandAgg": {**

**"terms": {**

**"field": "brand",**

**"size": 20**

**},**

**"aggs": { // 是brandAgg聚合的子聚合，也就是分组后对每组分别计算**

**"score\_stats": { // 聚合名称**

**"stats": { // 聚合类型，这里stats可以计算min、max、avg等**

**"field": "score" // 聚合字段，这里是score**

**}**

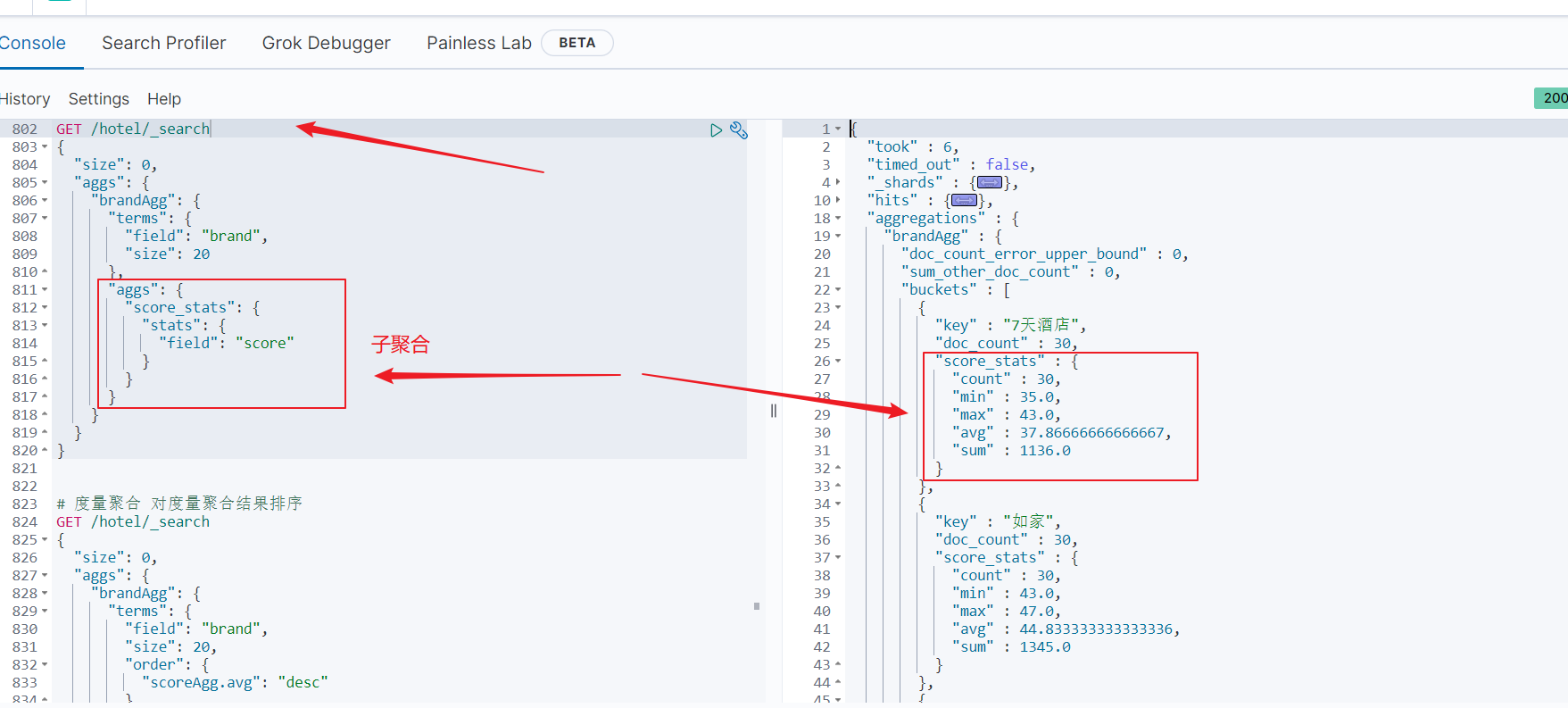
**}**

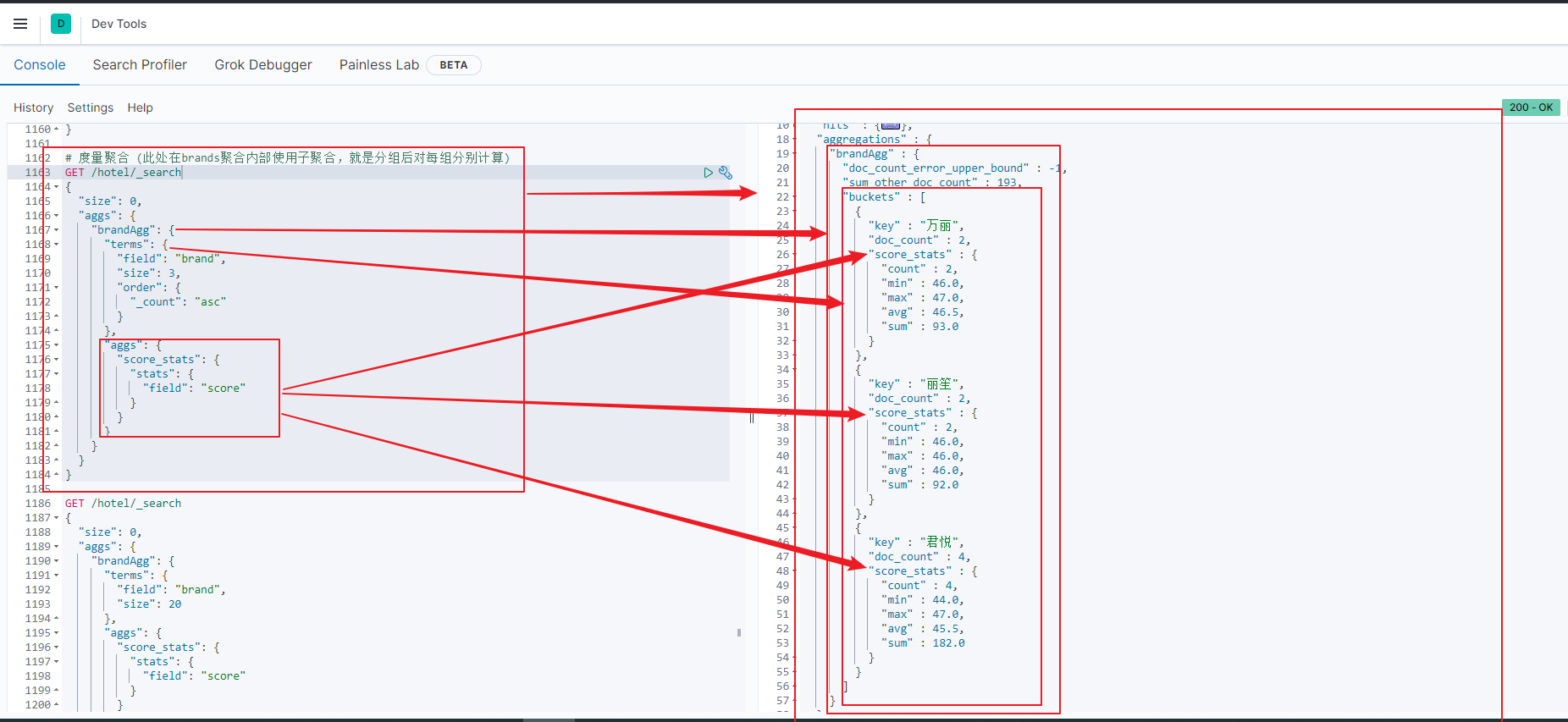
**}**

**}**

**}**

**}**





#### 对度量聚合结果排序

以上的的聚合我们是按照桶中文档数量\_count从少到高升序排序的，那我们能不能对brandAgg聚合结果按照评分的平均值做排序的，即品牌酒店评分平均分最高的排在前面。是可以的，聚合里定义order！

注意了，排序是对桶里的数据进行排序哦，所以应该在terms这个桶聚合中定义order！而且排序是按照scoreAgg中的avg做排序，那要怎么写呢：scoreAgg.avg

如下DSL语句解读：先对酒店文档按照品牌做桶排序分组，然后计算每个品牌酒店的平均评分、最高评分、最小评分，最后再按照平均分对酒店品牌做排序！

**GET /hotel/\_search**

**{**

**"size": 0,**

**"aggs": {**

**"brandAgg": {**

**"terms": {**

**"field": "brand",**

**"size": 20,**

**"order": {**

**"scoreAgg.avg": "desc"**

**}**

**},**

**"aggs": {**

**"scoreAgg": {**

**"stats": {**

**"field": "score"**

**}**

**}**

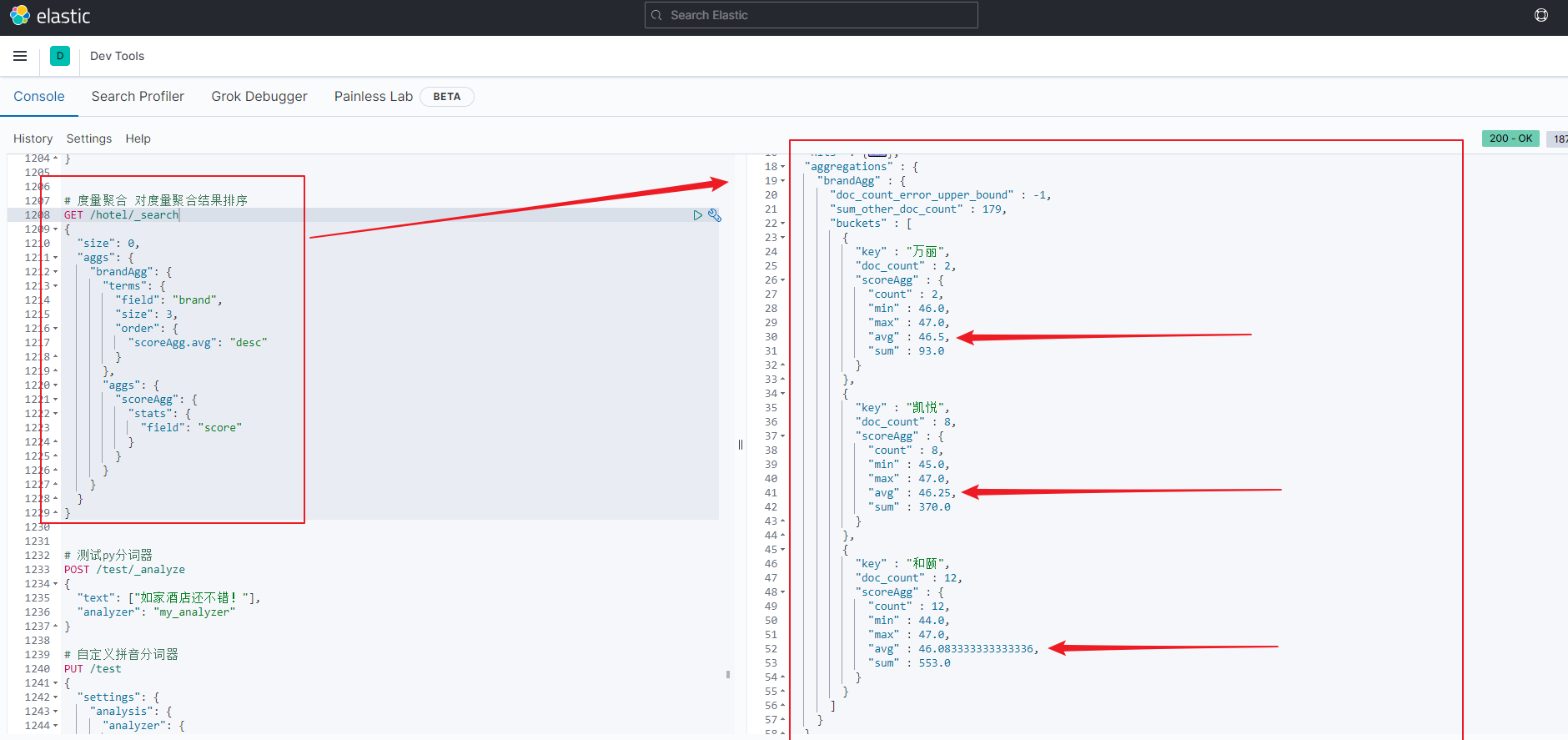
**}**

**}**

**}**

**}**

DevTools控制台检查聚合结果，发现确实按照评分平均分排序了！



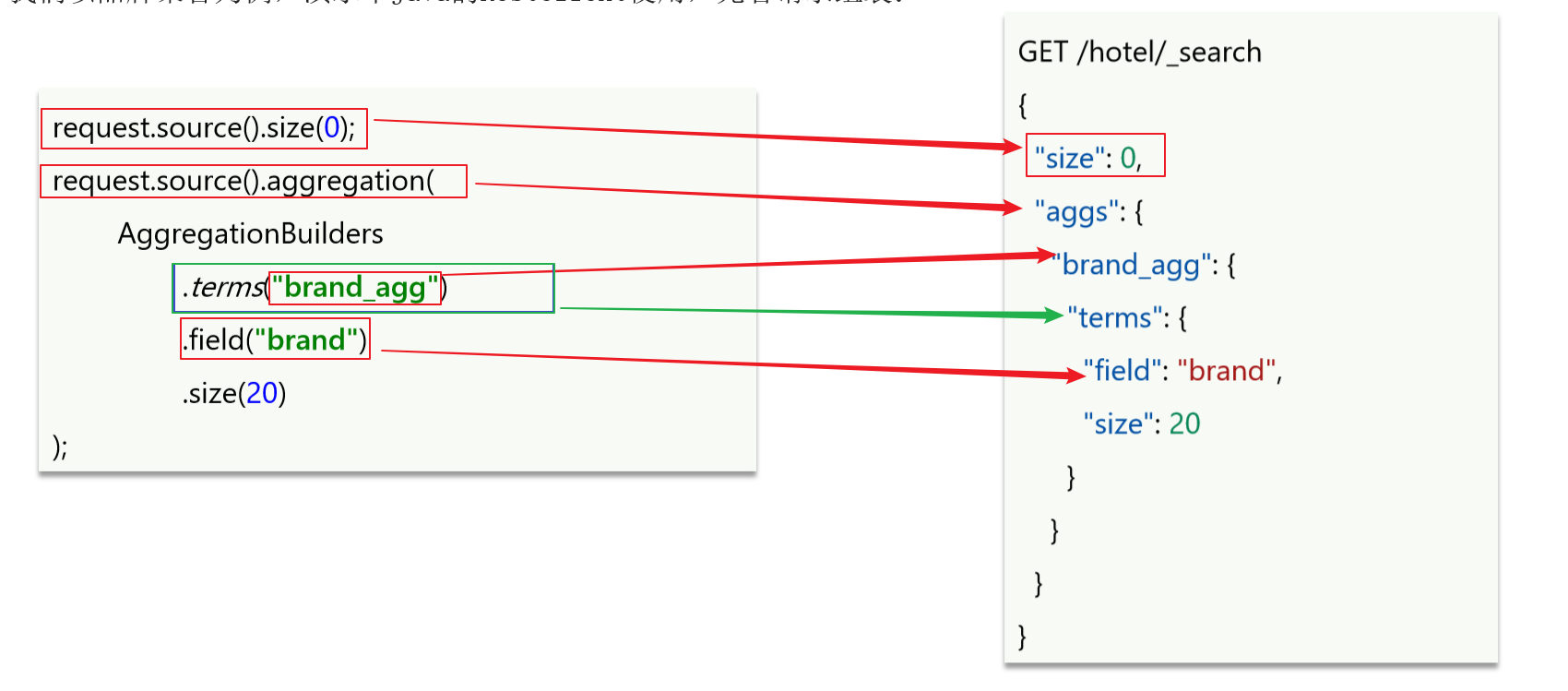
## RestAPI实现聚合

上面我们已经通过DSL的方式了解的聚合的基本玩法，现在我们必须用JavaRestClient实现一下聚合的功能！与以前所学的JavaRestClient实现搜索功能类似，我们的聚合也分成了请求条件的组装以及响应结果解析两部分！

### 请求条件组装

#### DSL通用语法

先看请求条件组装，学习方式依然能通过与DSL对比来学习

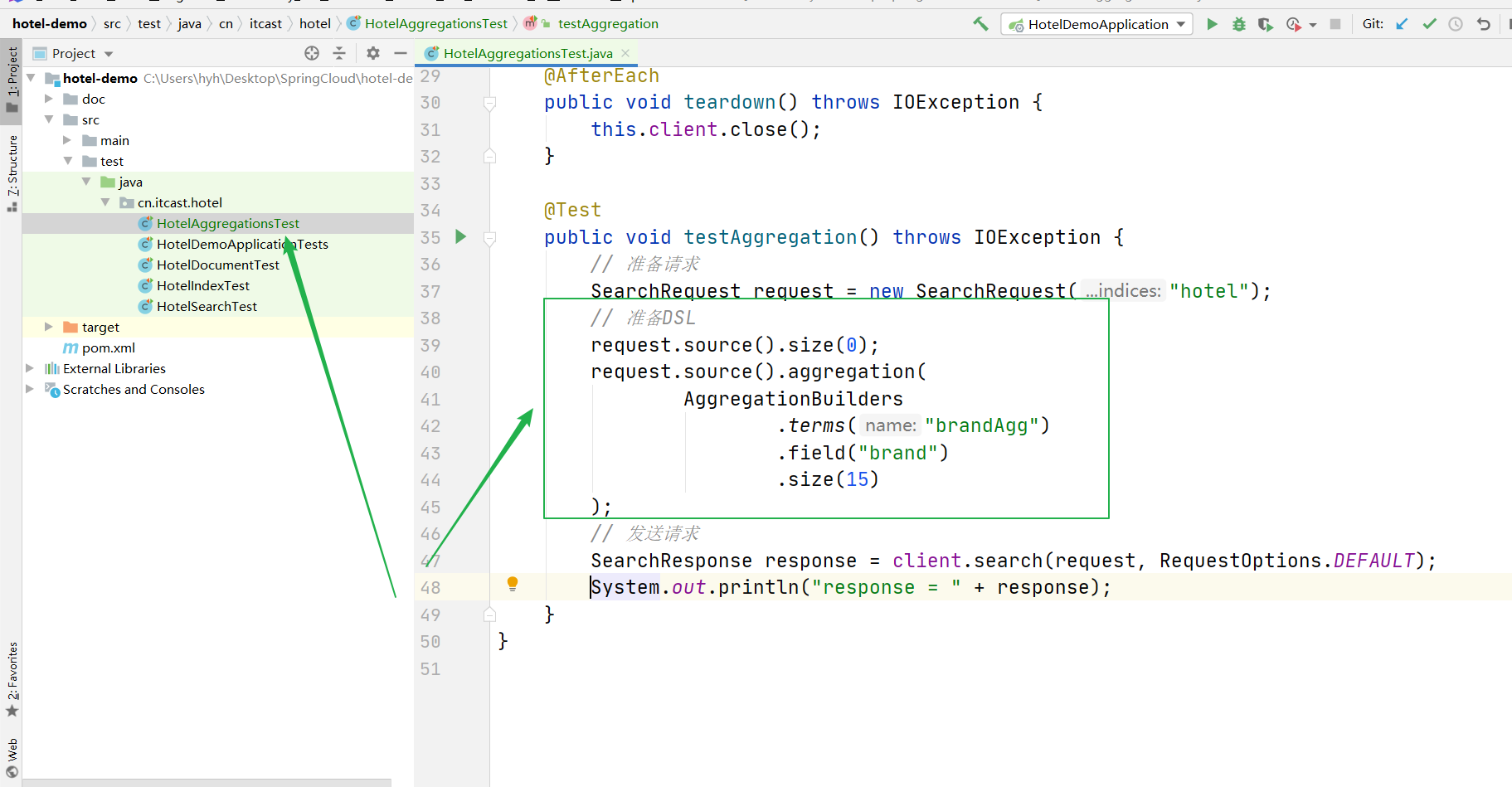


如上图，可以看到左边的DSL语法，可以看到请求方式、请求路径、JSON格式的DSL语法！

Java代码中request.source()代表的就是最外层的DSL，request.source()里面你可以指定查询条件、分页条件、排序条件、高亮条件、包括聚合条件，所以request.source.aggregation()就是组装聚合请求条件的！

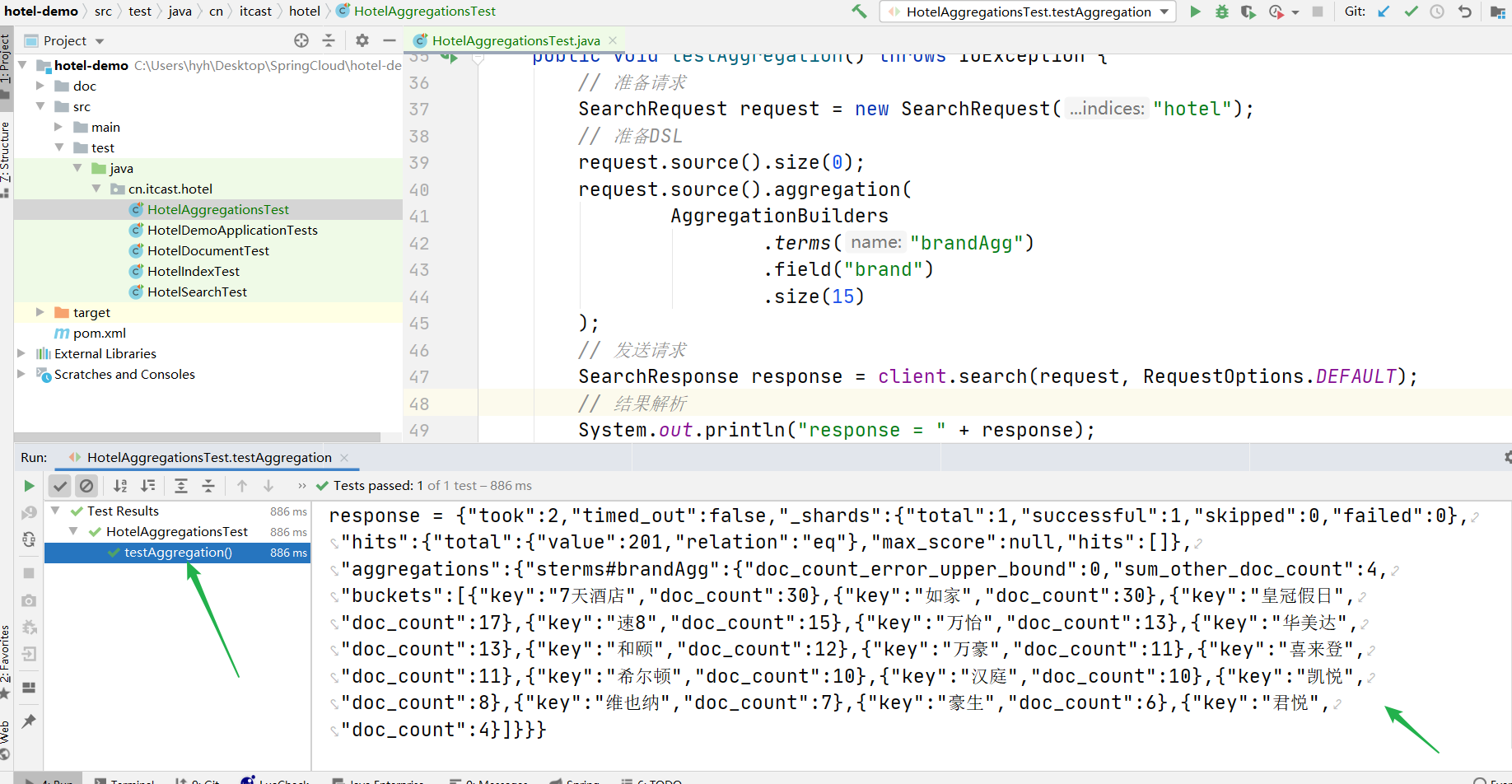
上图中看可以看到Java代码实现聚合和DSL语句实现聚合的一一对应关系，没必要再啰嗦了！

#### 代码实现

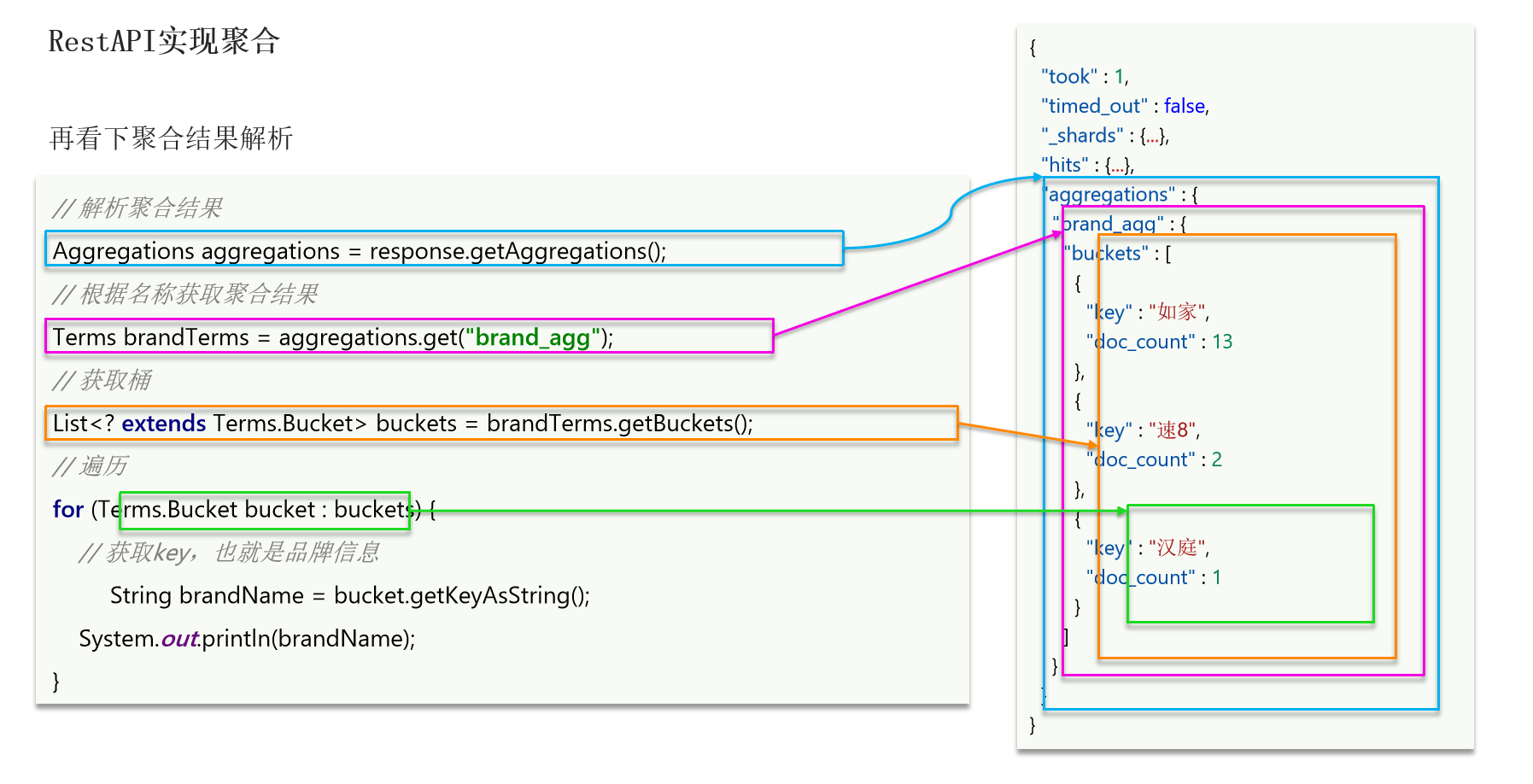


#### 运行测试方法，控制台查看聚合结果：

可以看到打印出来的结果是JSON风格的，所以解析要做的事就算是再response对象中解析JSON风格的结果，取出aggregations聚合结果里面的每一个值！



### 聚合结果解析



解析代码参考下面黑色部分：

**/\*\***

**\* 聚合**

**\***

**\* @throws IOException**

**\*/**

**@Test**

**public void testAggregation() throws IOException {**

**// 1：准备request**

**SearchRequest searchRequest = new SearchRequest("hotel");**

**// 2：准备DSL**

**// 2.1：设置size**

**searchRequest.source().size(0);**

**// 2.2：聚合**

**searchRequest.source().aggregation(**

**AggregationBuilders**

**.terms("brandAgg") // 定义聚合类型和名称**

**.field("brand") // 字段**

**.size(20)**

**);**

**// 3：发送请求**

**SearchResponse searchResponse = client.search(searchRequest, RequestOptions.DEFAULT);**

**// 4：解析结果**

**// 4.1：获取聚合结果**

**Aggregations aggregations = searchResponse.getAggregations();**

**// 4.2：根据名称获取具体聚合结果**

**Terms brandTerms = aggregations.get("brandAgg");**

**// 4.3：获取桶**

**List<? extends Terms.Bucket> buckets = brandTerms.getBuckets();**

**// 4.4：便历**

**for (Terms.Bucket bucket : buckets) {**

**System.out.println("bucket.getKeyAsString() = " + bucket.getKeyAsString());**

**System.out.println("bucket.getDocCount() = " + bucket.getDocCount());**

**}**

**}**