1. **Mapping：**
   1. 概念：mapping就是ES数据字段field的type元数据，ES在创建索引的时候，dynamic mapping会自动为不同的数据指定相应mapping，mapping中包含了字段的类型、搜索方式（exact value或者full text）、分词器等。
   2. 查看mapping

GET /product/\_mappings

* 1. Dynamic mapping
     1. “Elasticsearch”：text/keyword
     2. 123456 => long ？为什么不是integer
     3. 123.123 => double
     4. true false => boolean
     5. 2020-05-20 => date

为啥price是long类型而不是integer？因为es的mapping\_type是由JSON分析器检测数据类型，而Json没有隐式类型转换（integer=>long or float=> double）,所以dynamic mapping会选择一个比较宽的数据类型。

* 1. 搜索方式：
     1. exact value 精确匹配：在倒排索引过程中，分词器会将field作为一个整体创建到索引中，
     2. full text全文检索：分词、近义词同义词、混淆词、大小写、词性、过滤、时态转换等（normaliztion）
  2. ES数据类型：
     1. **核心类型**
        1. **数字类型**：
           1. long, integer, short, byte, double, float, half\_float, scaled\_float
           2. 在满足需求的情况下，尽可能选择范围小的数据类型。
        2. **字符串：string**：
           1. **keyword**：适用于索引结构化的字段，可以用于过滤、排序、聚合。keyword类型的字段只能通过精确值（exact value）搜索到。Id应该用keyword

**text**：当一个字段是要被全文搜索的，比如Email内容、产品描述，这些字段应该使用text类型。设置text类型以后，字段内容会被分析，在生成倒排索引以前，字符串会被分析器分成一个一个**词项**。**text类型的字段不用于排序，很少用于聚合**。（解释一下为啥不会为text创建索引：**字段数据会占用**大量**堆空间，尤其是在加载高基数text字段时。字段数据一旦加载到堆中，就在该段的生命周期内保持在那里。同样，加载字段数据是一个昂贵的过程，可能导致用户遇到延迟问题。这就是默认情况下禁用字段数据的原因**）

* + - * 1. 有时，在同一字段中同时具有全文本（text）和关键字（keyword）版本会很有用：一个用于全文本搜索，另一个用于聚合和排序。
      1. **date**（时间类型）：exact value
      2. 布尔类型：boolean
      3. [binary](https://www.elastic.co/guide/en/elasticsearch/reference/current/binary.html" \o "Binary datatype)（二进制）：[binary](https://www.elastic.co/guide/en/elasticsearch/reference/current/binary.html" \o "Binary datatype)
      4. [range](https://www.elastic.co/guide/en/elasticsearch/reference/current/range.html" \o "Range datatypes)（区间类型）：integer\_range、float\_range、long\_range、double\_range、date\_range
    1. 复杂类型：
       1. Object：用于单个JSON对象
       2. Nested：用于JSON对象数组
    2. 地理位置：
       1. Geo-point：纬度/经度积分
       2. Geo-shape：用于多边形等复杂形状
    3. 特有类型：
       1. IP地址：ip 用于IPv4和IPv6地址
       2. [Completion](https://www.elastic.co/guide/en/elasticsearch/reference/current/search-suggesters.html" \l "completion-suggester" \o "Completion Suggester)：提供自动完成建议
       3. Tocken\_count：计算字符串中令牌的数量
       4. [Murmur3](https://www.elastic.co/guide/en/elasticsearch/plugins/7.7/mapper-murmur3.html" \t "https://www.elastic.co/guide/en/elasticsearch/reference/current/_top)：在索引时计算值的哈希并将其存储在索引中
       5. [Annotated-text](https://www.elastic.co/guide/en/elasticsearch/plugins/7.7/mapper-annotated-text.html" \t "https://www.elastic.co/guide/en/elasticsearch/reference/current/_top)：索引包含特殊标记的文本（通常用于标识命名实体）
       6. [Percolator](https://www.elastic.co/guide/en/elasticsearch/reference/current/percolator.html" \o "Percolator type)：接受来自query-dsl的查询
       7. Join：为同一索引内的文档定义父/子关系
       8. [Rank features](https://www.elastic.co/guide/en/elasticsearch/reference/current/rank-features.html" \o "Rank features datatype)：记录数字功能以提高查询时的点击率。
       9. [Dense vector](https://www.elastic.co/guide/en/elasticsearch/reference/current/dense-vector.html" \o "Dense vector datatype)：记录浮点值的密集向量。
       10. [Sparse vector](https://www.elastic.co/guide/en/elasticsearch/reference/current/sparse-vector.html" \o "Sparse vector datatype)：记录浮点值的稀疏向量。
       11. [Search-as-you-type](https://www.elastic.co/guide/en/elasticsearch/reference/current/search-as-you-type.html" \o "Search-as-you-type datatype)：针对查询优化的文本字段，以实现按需输入的完成
       12. [Alias](https://www.elastic.co/guide/en/elasticsearch/reference/current/alias.html" \o "Alias datatype)：为现有字段定义别名。
       13. [Flattened](https://www.elastic.co/guide/en/elasticsearch/reference/current/flattened.html" \o "Flattened datatype)：允许将整个JSON对象索引为单个字段。
       14. [Shape](https://www.elastic.co/guide/en/elasticsearch/reference/current/shape.html" \o "Shape datatype)：shape 对于任意笛卡尔几何。
       15. [Histogram](https://www.elastic.co/guide/en/elasticsearch/reference/current/histogram.html" \o "Histogram datatype)：histogram 用于百分位数聚合的预聚合数值。
       16. [Constant keyword](https://www.elastic.co/guide/en/elasticsearch/reference/current/constant-keyword.html" \o "Constant keyword datatype)：keyword当所有文档都具有相同值时的情况的 专业化。
    4. Array（数组）：在Elasticsearch中，数组不需要专用的字段数据类型。默认情况下，任何字段都可以包含零个或多个值，但是，数组中的所有值都必须具有相同的数据类型。
    5. ES 7新增：
       1. Date\_nanos：date plus 纳秒
       2. Features：
       3. Vector：as
  1. 手工创建mapping

PUT /product

{

"mappings": {

"properties": {

"field": {

"mapping\_parameter": "parameter\_value"

}

}

}

}

* 1. Mapping parameters
     1. **index**：是否对创建对当前字段创建索引，默认true，如果不创建索引，该字段不会通过索引被搜索到,但是仍然会在source元数据中展示
     2. analyzer:指定分析器（character filter、tokenizer、Token filters）。
     3. boost：对当前字段相关度的评分权重，默认1
     4. **coerce**：是否允许强制类型转换 true “1”=> 1 false “1”=< 1
     5. copy\_to：

"field": {

"type": "text",

"copy\_to": "other\_field\_name"

},

* + 1. **doc\_values**：为了提升排序和聚合效率，默认true，如果确定不需要对字段进行排序或聚合，也不需要通过脚本访问字段值，则可以禁用doc值以节省磁盘空间（不支持text和annotated\_text）
    2. dynamic：控制是否可以动态添加新字段
       1. true 新检测到的字段将添加到映射中。（默认）
       2. false 新检测到的字段将被忽略。这些字段将不会被索引，因此将无法搜索，但仍会出现在\_source返回的匹配项中。这些字段不会添加到映射中，必须显式添加新字段。
       3. strict 如果检测到新字段，则会引发异常并拒绝文档。必须将新字段显式添加到映射中
    3. **eager\_global\_ordinals：用于聚合的字段上，优化聚合性能。**
       1. Frozen indices（冻结索引）：有些索引使用率很高，会被保存在内存中，有些使用率特别低，宁愿在使用的时候重新创建，在使用完毕后丢弃数据，Frozen indices的数据命中频率小，不适用于高搜索负载，数据不会被保存在内存中，堆空间占用比普通索引少得多，Frozen indices是只读的，请求可能是秒级或者分钟级。**eager\_global\_ordinals不适用于Frozen indices**
    4. enable：是否创建倒排索引，可以对字段操作，也可以对索引操作，如果不创建索引，让然可以检索并在\_source元数据中展示，谨慎使用，该状态无法修改。

PUT my\_index{

"mappings": {

"enabled": false

}}

PUT my\_index{

"mappings": {

"properties": {

"session\_data": {

"type": "object",

"enabled": false

}

}

}}

* + 1. **fielddata**：查询时****内存****数据结构，在首次用当前字段聚合、排序或者在脚本中使用时，需要字段为fielddata数据结构，并且创建倒排索引保存到堆中
    2. **fields：**给field创建多字段，用于不同目的（全文检索或者聚合分析排序）
    3. format：格式化

"date": {

"type": "date",

"format": "yyyy-MM-dd"

}

* + 1. ignore\_above：**超过长度将被忽略**
    2. ignore\_malformed：忽略类型错误

PUT my\_index{

"mappings": {

"properties": {

"number\_one": {

"type": "integer",

"ignore\_malformed": true

},

"number\_two": {

"type": "integer"

}

}

}}

PUT my\_index/\_doc/1{

"text": "Some text value",

"number\_one": "foo" } //虽然有异常 但是不抛出

PUT my\_index/\_doc/2{

"text": "Some text value",

"number\_two": "foo" } //数据格式不对

* + 1. index\_options：控制将哪些信息添加到反向索引中以进行搜索和突出显示。仅用于text字段
    2. Index\_phrases：提升exact\_value查询速度，但是要消耗更多磁盘空间
    3. Index\_prefixes：前缀搜索
       1. min\_chars：前缀最小长度，>0，默认2（包含）
       2. max\_chars：前缀最大长度，<20，默认5（包含）

"index\_prefixes": {

"min\_chars" : 1,

"max\_chars" : 10

}

* + 1. meta：附加元数据
    2. normalizer：
    3. norms：是否禁用评分（在filter和聚合字段上应该禁用）。
    4. null\_value：为null值设置默认值

"null\_value": "NULL"

* + 1. position\_increment\_gap：
    2. proterties：除了mapping还可用于object的属性设置
    3. **search\_analyzer**：设置单独的查询时分析器：

PUT my\_index{

"settings": {

"analysis": {

"filter": {

"autocomplete\_filter": {

"type": "edge\_ngram",

"min\_gram": 1,

"max\_gram": 20

}

},

"analyzer": {

"autocomplete": {

"type": "custom",

"tokenizer": "standard",

"filter": [

"lowercase",

"autocomplete\_filter"

]

}

}

}

},

"mappings": {

"properties": {

"text": {

"type": "text",

"analyzer": "autocomplete",

"search\_analyzer": "standard"

}

}

}}

PUT my\_index/\_doc/1{

"text": "Quick Brown Fox" }

GET my\_index/\_search{

"query": {

"match": {

"text": {

"query": "Quick Br",

"operator": "and"

}

}

}}

* + 1. similarity：为字段设置相关度算法，支持BM25、claassic（TF-IDF）、boolean
    2. store：设置字段是否仅查询
    3. term\_vector：

聚合查询：

* 1. bucket和metirc：
  2. 语法:

aggs:{

code...

}

* 1. “goup by”：
     1. 以tag维度每个产品的数量，即每个标签
     2. 在的基础上增加筛选条件：统计价格大于1999的数据
  2. “avg”：
     1. 价格大于1999的每个tag产品的平均价格
  3. 分组聚合
  4. 按照千元机：1000以下 中端机：2000-3000 高端机：3000以上分组聚合，分别计算数量