1. 简述 text 数据类型和 keyword 数据类型的使用场景。

答：text 数据类型用于全文检索，如电子邮件正文或产品说明。text 字段不用于排序，并且很少用于聚合。

keyword 数据类型用于结构化数据的检索，如电子邮件地址，主机名，状态码，邮政编码或标签。keyword 字段通常用来过滤、排序和聚合，并且只能通过确切值来查询。

2. 运行一个 copy_to 的例子。

答：copy_to 可以将多个字段的值复制到一个组的字段，然后可以将其作为单个字段进行查询。例子：

```
DELETE my_index
PUT my_index
{
    "mappings": {
        "_doc": {
            "properties": {
                "first_name": {
                    "type": "text",
                    "copy_to": "full_name"
                },
                "last_name": {
                    "type": "text",
                    "copy_to": ["full_name", "nickname"]
                },
                "full_name": {
                    "type": "text"
                },
                "nickname": {
                    "type": "text"
                }
            }
        }
    }
}
PUT my_index/_doc/1
{
    "first_name": "John",
    "last_name": "Smith"
}
```

```
GET my_index/_search
{
    "query": {
        "match": {
            "full_name":
            {
                "query": "john smith",
                "operator": "and"
            }
        }
    }
}
```
返回：
```
{
    "took": 4,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 0.5753642,
        "hits": [
            {
                "_index": "my_index",
                "_type": "_doc",
                "_id": "1",
                "_score": 0.5753642,
                "_source": {
                    "first_name": "John",
                    "last_name": "Smith"
                }
            }
        ]
    }
}
GET my_index/_search
{
    "query": {
        "match": {
            "nickname": "smith"
```

```
                }
            }
        }
    返回：
    {
        "took": 26,
        "timed_out": false,
        "_shards": {
            "total": 5,
            "successful": 5,
            "skipped": 0,
            "failed": 0
        },
        "hits": {
            "total": 1,
            "max_score": 0.2876821,
            "hits": [
                {
                    "_index": "my_index",
                    "_type": "_doc",
                    "_id": "1",
                    "_score": 0.2876821,
                    "_source": {
                        "first_name": "John",
                        "last_name": "Smith"
                    }
                }
            ]
        }
    }
```

3. 运行一个动态模板的例子， 截图

答：

下面运行的例子为： 匹配字符串字段并且字段名以 long_开头的所有字符串字段（并排除以 _text 结尾的字符串字段）然后将它们映射为 long 类型的字段：

代码为：
```
DELETE my_index

PUT my_index
{
    "mappings": {
```

```
        "_doc": {
          "dynamic_templates": [
            {
              "longs_as_strings": {
                "match_mapping_type": "string",
                "match":      "long_*",
                "unmatch": "*_text",
                "mapping": {
                  "type": "long"
                }
              }
            }
          ]
        }
      }
    }
```

```
PUT my_index/_doc/1
{
  "long_num": "5",
  "long_text": "foo"
}
```

```
GET my_index/_mapping
```

运行截图如下：