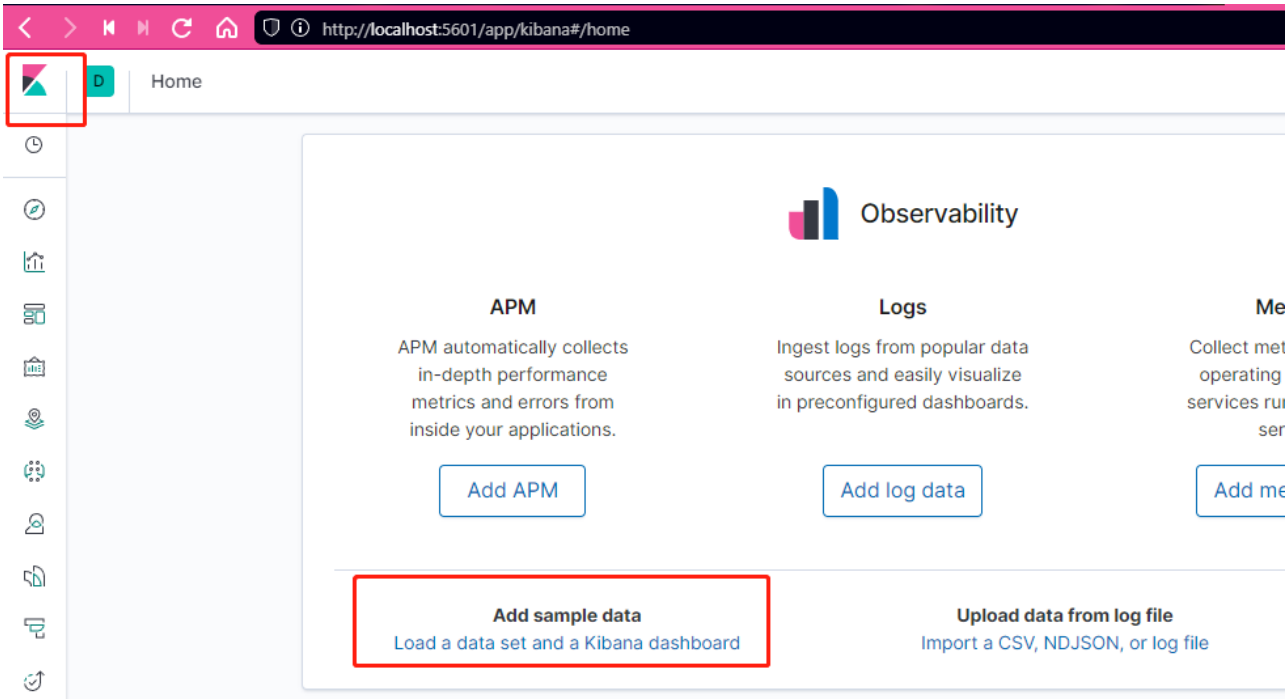
# es应用笔记2-sql查询

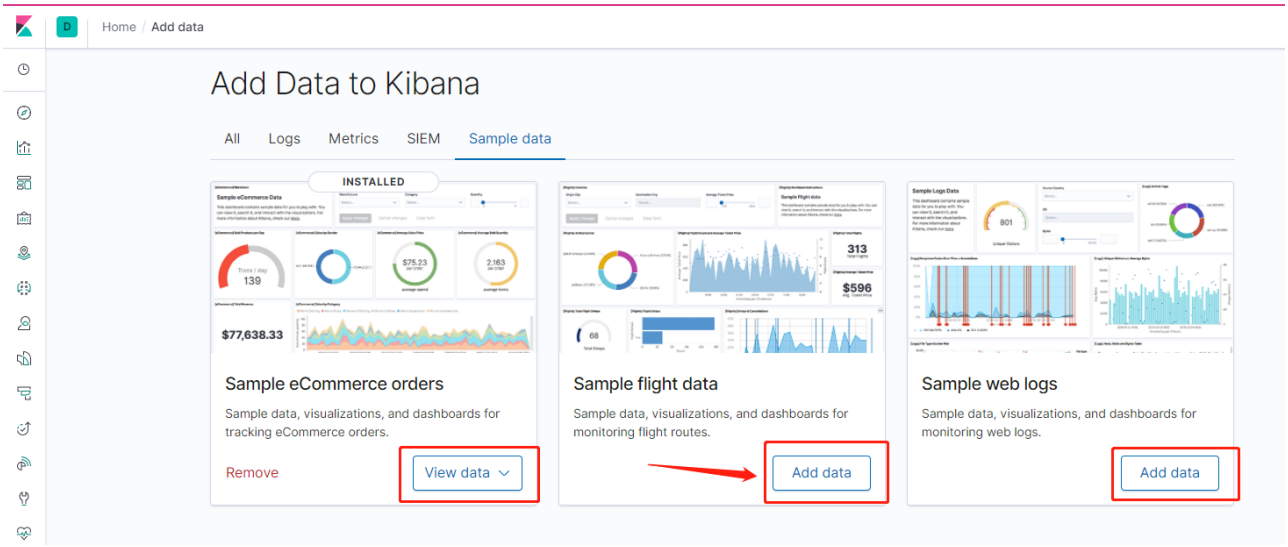
> *es*作为一个搜索索引，在分析场景中，作为明细查询的场景会比*kylin*、*impala*、*hive*等更加合适。
>
> *es*在*6.3*版本开始支持*sql*查询，且其*sql*基础语法与大数据端的语法较兼容，函数库略有不同。
>
> 对于多数据源的接入，通过*jdbc*接入*es*改造成本较低，但是*xpack-sql-jdbc*这个客户端的包是收费的，但是其服务端仍提供了*rest api* 供查询。

## 界面查询

### kibana中添加简单数据

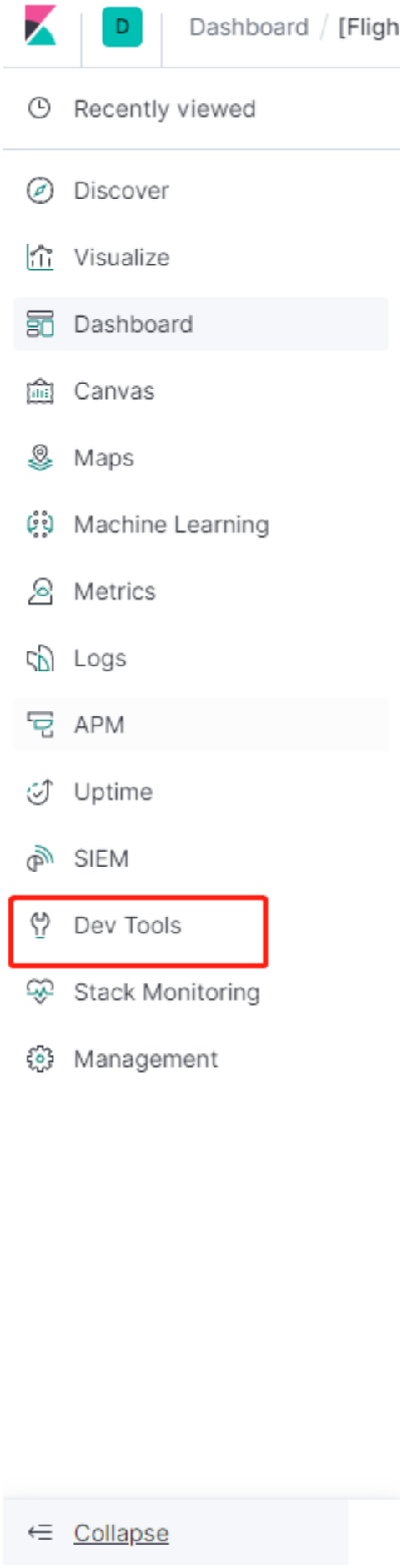



选择想要的一个栗子



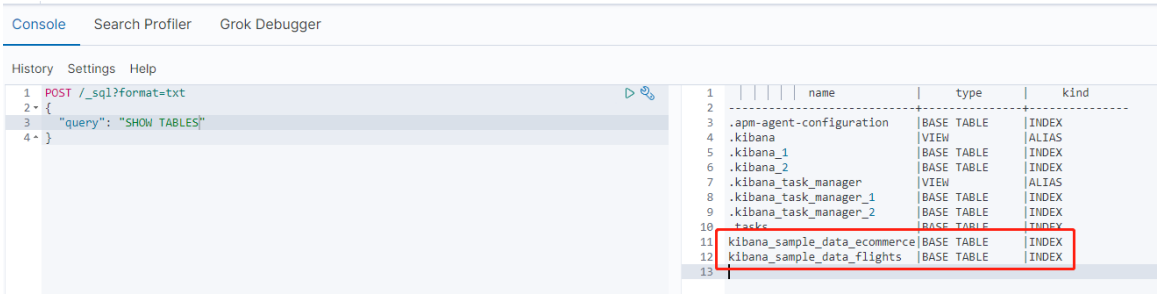

# 开发者工具查询

- 进入开发者工具界面



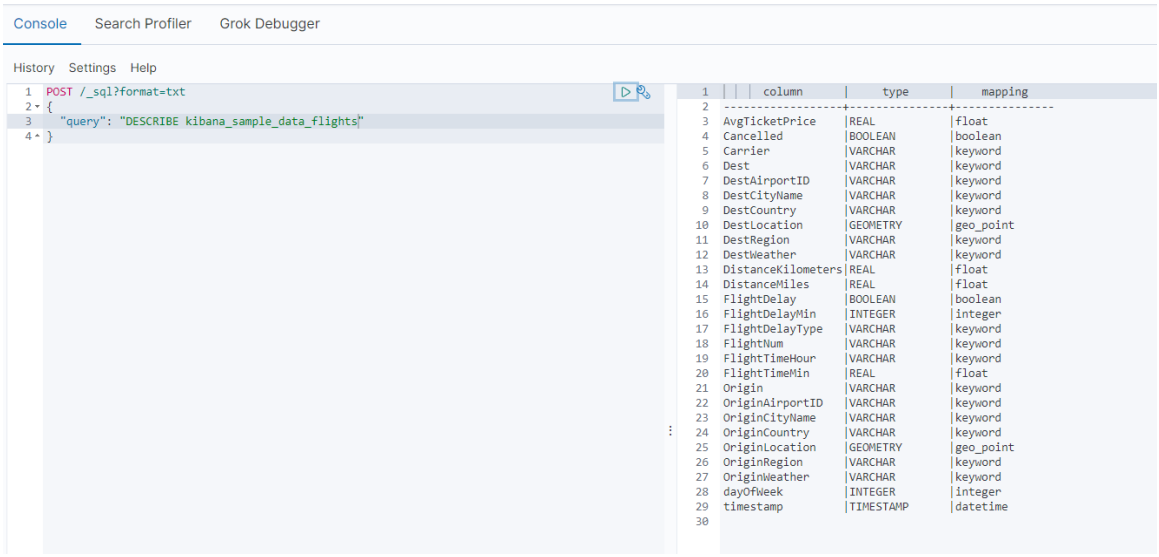

- 查看有什么表

  使用 SHOW TABLES查询



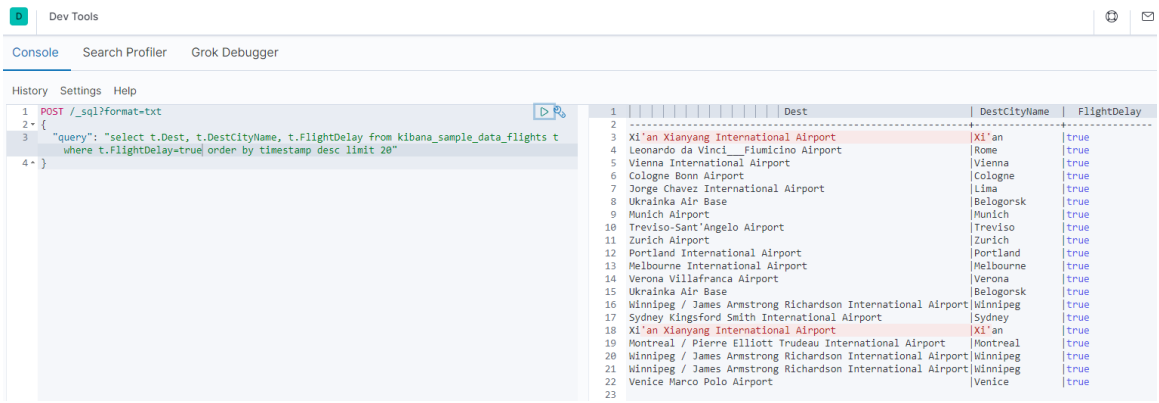

- 查看表有什么列

  使用 DESCRIBE [TABLENAME]




- SQL查询记录

  查询一下延误的航班




# REST API

REST API 才是其他程序可以通过SQL查询ES的关键。

## kibana rest api

通过浏览器F12可以获取到查询kibana的api接口，不过我们并不关心它的API：

```
curl 'http://localhost:5601/api/console/proxy?
path=%2F_sql%3Fformat%3Dtxt&method=POST' \
  -H 'Connection: keep-alive' \
  -H 'sec-ch-ua: "Chromium";v="98", " Not A;Brand";v="99"' \
  -H 'Accept: text/plain, */*; q=0.01' \
  -H 'Content-Type: application/json' \
  -H 'sec-ch-ua-mobile: ?0' \
  -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.136
Safari/537.36' \
  -H 'kbn-version: 7.6.2' \
  -H 'sec-ch-ua-platform: "Windows"' \
  -H 'Origin: http://localhost:5601' \
  -H 'Sec-Fetch-Site: same-origin' \
  -H 'Sec-Fetch-Mode: cors' \
  -H 'Sec-Fetch-Dest: empty' \
  -H 'Referer: http://localhost:5601/app/kibana' \
  -H 'Accept-Language: zh-CN,zh;q=0.9,zh-Hans;q=0.8,en;q=0.7' \
  --data-raw $'{\r\n  "query": "select t.Dest from
kibana_sample_data_flights t limit 20"\r\n}\n' \
  --compressed
```

## es rest api

其实kibana的开发者工具以及告诉我们ES的查询API为`POST /_sql?format=txt`，那么稍作改造直接发给ES：

```
curl 'http://localhost:9200/_sql?format=txt' \
  -H 'Connection: keep-alive' \
  -H 'Accept: text/plain, */*; q=0.01' \
  -H 'Content-Type: application/json' \
  -d $'{\r\n  "query": "select t.Dest from
kibana_sample_data_flights t limit 20"\r\n}\n' \
  --compressed
```

其结果如下：

```
sh-4.2# curl 'http://localhost:9200/_sql?format=txt' \
>   -H 'Connection: keep-alive' \
>   -H 'Accept: text/plain, */*; q=0.01' \
>   -H 'Content-Type: application/json' \
>   -d $'{\r\n  "query": "select t.Dest from
kibana_sample_data_flights t limit 1"\r\n}\n' \
>   --compressed
                    Dest
-------------------------------------------
Sydney Kingsford Smith International Airport
```

对于应用程序，我们选择接收JSON，那么`format=json`即可，结果如下：

```
sh-4.2# curl 'http://localhost:9200/_sql?format=json' \
>   -H 'Connection: keep-alive' \
>   -H 'Accept: text/plain, */*; q=0.01' \
>   -H 'Content-Type: application/json' \
>   -d $'{\r\n  "query": "select t.Dest from
kibana_sample_data_flights t limit 1"\r\n}\n' \
>   --compressed
{"columns":[{"name":"Dest","type":"keyword"}],"rows":[["Sydney
Kingsford Smith International Airport"]]}sh-4.2#
```

## 主要参数介绍

### format

格式化返回结果，摘抄自官网：

| FORMAT | ACCEPT HTTP HEADER | DESCRIPTION |
| --- | --- | --- |
| **Human Readable** | | |
| `csv` | `text/csv` | Comma-separated values |
| `json` | `application/json` | JSON (JavaScript Object Notation) human-readable format |
| `tsv` | `text/tab-separated-values` | Tab-separated values |
| `txt` | `text/plain` | CLI-like representation |

| FORMAT | ACCEPT HTTP HEADER | DESCRIPTION |
| --- | --- | --- |
| `yaml` | `application/yaml` | YAML (YAML Ain't Markup Language) human-readable format |
| **Binary Formats** | | |
| `cbor` | `application/cbor` | Concise Binary Object Representation |
| `smile` | `application/smile` | Smile binary data format similar to CBOR |

## 分页

如果在查询时，使用了DSL的`fetch_size`如：

```
POST /_sql?format=json
{
    "query": "SELECT * FROM library ORDER BY page_count DESC",
    "fetch_size": 5
}
```

其返回中就会有游标：

```
{
    "columns": [

    ],
    "rows": [

    ],
    "cursor":
"sDXF1ZXJ5QW5kRmVOY2gBAAAAAAAAAAEWWWdrRlVfSS1TbDYtcW9lc1FJNmlYdw==:
BAFmBmF1dGhvcgFmBG5hbWUBZgpwYWdlX2NvdW50AWYMcmVsZWFzZV9kYXRl+v///w8
="
}
```

可以通过发送游标进行下一页查询，同时，游标还必须手动进行关闭。

```
POST /_sql/close
{
    "cursor":
"sDXF1ZXJ5QW5kRmV0Y2gBAAAAAAAAAAEWYUpOYklQMHhRUEtld3RsNnFtYU1hQQ==:
BAFmBGRhdGUBZgVsaWtlcwFzB21lc3NhZ2UBZgR1c2Vy9f///w8="
}
```

## columnar

是否返回列信息

默认为true，查询返回列信息。

```
POST /_sql?format=json
{
    "query": "SELECT * FROM library ORDER BY page_count DESC",
    "fetch_size": 5,
    "columnar": true
}
```

结果：

```
{
    "columns": [
        {"name": "author", "type": "text"},
        {"name": "name", "type": "text"},
        {"name": "page_count", "type": "short"},
        {"name": "release_date", "type": "datetime"}
    ],
    "values": [

    ],
    "cursor":
"sDXF1ZXJ5QW5kRmV0Y2gBAAAAAAAAAAEWWWdrRlVfSS1TbDYtcW9lc1FJNmlYdw==:
BAFmBmF1dGhvcgFmBG5hbWUBZgpwYWdlX2NvdW50AWYMcmVsZWFzZV9kYXRl+v///w8
="
}
```

官方推荐在分页查询第一次查询时返回列信息，后续查询不再返回列信息的方式。

### 其他rest参数

官网链接：https://www.elastic.co/guide/en/elasticsearch/reference/7.6/sql-rest-fields.html

fetch_size、filter、request_timeout、page_timeout也是会用到的参数。

## SQL转DSL

可以通过/_sql/translate进行转换

```
POST /_sql/translate
{
    "query": "SELECT * FROM library ORDER BY page_count DESC",
    "fetch_size": 10
}
```

## SQL语法、命令

https://www.elastic.co/guide/en/elasticsearch/reference/7.6/sql-spec.html

### 函数

https://www.elastic.co/guide/en/elasticsearch/reference/7.6/sql-functions.html

### 限制

> *https://www.elastic.co/guide/en/elasticsearch/reference/7.6/sql-limitations.html*

SQL查询并非ES查询主流，有许多限制需要注意，这里仅将常见的列出来。

1. 查询返回结果不能过大，会抛出异常`ParsingExpection`
2. where和 order by时，scalar函数不能在嵌套字段上使用
3. 两个不同的结构的嵌套字段不能同时使用
4. 嵌套字段不能分页
5. keyword 属性需要常态化
6. arrary类型不能搜索，可以配置`field.multi.value.leniency`争取宽大处理

7. 聚合的排序不支持，将其放在客户端实现，且不允许超过512行
8. 聚合函数中必须是直接属性，而不能是scalar函数加工的属性
9. 嵌套子查询的实力只有小学生级别，超出这个范围就不支持了：`SELECT X FROM (SELECT ...) WHERE [simple_condition]`
10. 不能再having 中使用FIRST/LAST
11. TIME类型的属性不可以在GROUP BY / HISTOGRAM中使用
12. PIVOT中只能接收一个聚合函数