Jio internship

elasticsearch

bibliography
https://coralogix.com/blog/42-elasticsearch-query-examples-hands-on-tutorial/
google.com
youtube
https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
stackoverflow

“/” does not work while entering date.
==========boolean query

Must is analogous to the boolean **AND**, **must_not** is analogous to the boolean **NOT**, and **should** is roughly equivalent to the boolean **OR**. Note that **should** isn't exactly like a boolean **OR**, but we can use it to that effect. And we'll take a look at **filter** later on.

Boolean AND and NOT are easy, so let's look at those two first.

1) If you want documents where preference_1 = Apples AND preference_2 = Bananas the bool query looks like this:

```
{ "query" : { "bool" : { "must": [{ "match": { "preference_1": "Apples" } }, { "match": { "preference_2": "Bananas" } }] } } }
```

2) If you want documents where preference_1 != Apples:

```
{ "query" : { "bool" : { "must_not": { "match": { "preference_1": "Apples" } } } } }
```

3) If you want the set of documents where preference_1 = Apples OR preference_1 = Raspberries:

```
{ "query" : { "bool" : { "should": [{ "match": { "preference_1": "Apples" } }, { "match": { "preference_1": "Raspberries" } }] } } }
```

4) these, then, can all be combined into much more complex boolean logic, because we can easily nest bool queries. So let's look at this boolean logic:

```
(preference_1 = Apples AND preference_2=Bananas) OR (preference_1 = Apples AND preference_2 = Cherries) OR preference_1 = Grapefruits
```

```
{ "query": { "bool": { "should": [{ "bool": { "must": [{ "match": { "preference_1": "Apples" } }, { "match": { "preference_2": "Bananas" } }] } }, { "bool": { "must": [{ "match": { "preference_1": "Apples" } }, { "match": { "preference_2": "Cherries" } }] } }, { "match": { "preference_1": "Grapefruit" } }] } } }
```

---

show only some fields out of many

```
{
  "_source": [
  "name",
  "street"
 ]
```

```
}
```

score

Let us include two filters in the “functions” part of the query. The first one would search for the name “dawarka” in the “street” field of the document and if found will boost the score by a weight of 2. The second clause would search for the name “new” in the field “street” and will boost by a factor of 10, for such documents. Here is the query for the same:

```
"query": {
  "function_score": {
   "query": {
    "match": {
     "street": "dawarka"
    }
   },
   "functions": [
    {
     "filter": {
      "match": {
       "name": "dawarka"
      }
     },
     "weight": 2
    },
    {
     "filter": {
      "match": {
       "name": "new"
      }
     },
     "weight": 10
    }
   ],
   "score_mode": "multiply",
   "boost": "5",
   "boost_mode": "multiply"
  }
 }
```

Hadoop installation when elasticsearch is already installed on the system

.

.

.

.

.

.

python.....

A Python port of the Apache Tika library, According to the documentation Apache tika supports text extraction from over 1500 file formats.

```
pip install tika
from tika import parser
parsed = parser.from_file('/path/to/file')
print(parsed["metadata"]) #To get the meta data of the file
print(parsed["content"]) # To get the content of the file
```