InfoSearch
Answer (6)

Abbreviations:
ES: Elastic Search

Steps for the Solution Walk through:

Step1 :
First we have to create an index. We name the index as *'netflix'*. As in in part C of (a) we have to access data using n-gram (n=2; i.e. (8%4)+2; group_no = 8) for our group, we have decided to create our mapping as follows :

```
PUT /netflix
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_analyzer": {
          "type": "custom",
          "tokenizer": "whitespace",
          "filter": [
            "lowercase",
            "2_grams"
          ]
        }
      },
      "filter": {
        "2_grams": {
          "type": "ngram",
          "min_gram": 2,
          "max_gram": 2
        }
      }
    }
  },

  "mappings": {
    "properties": {
```

```
      "director": {
        "type": "text",
        "analyzer": "my_analyzer"
      }
    }
  }
}
```

Step2 :

We have to insert the data using "*_bulk*" api for ES.
We wrote a python script to prepare the data (i.e. netflix.json). Which can be found in the submission folder as 'convert.py', which takes *'netflix.json*' as input and produces *'elastic_netflix.json'* as output.

```
import json

with open('./netflix.json') as f:
    data = json.load(f)



for key in data:
    add_str = { "index" : { "_id": key} }

    with open('elastic_netflix.json', 'a') as json_file:
        json.dump(add_str, json_file)
        json_file.write("\n")
        json.dump(data[key], json_file)
        json_file.write("\n")
```

Step3 :

We inserted data to bulk api using CURL.

```
$ curl -H "Content-Type: application/json" -XPOST
"localhost:9200/netflix/_bulk?pretty&refresh" --data-binary
"@elastic_netflix.json"
```

Step4 :

Now for making REST api we have used Node.js.
We have used the “elasticsearch” module for connecting to elastic search and “express” module for HTTP request and response.

The complete Node.js module can be found with the submission folder.
Dependencies of the node module can be installed by using package.json file.

Different endpoint for different questions:
All the requests to API are **HTTP GET** requests only.
We used an url encoding method for sending GET request with multiple values.

A.

(a) Endpoint for Adults :

Query(sample) : localhost:3000/InfoSearch/for_adult/to

End Point : /InfoSearch/for_adult/
Query : to

(b) Child Proof Endpoint :

Query(sample) : localhost:3000/InfoSearch/for_child/to

End Point : /InfoSearch/for_child/
Query : to

B.

(a) Pagination for movies :

Query(sample) : localhost:3000/InfoSearch/page_movies/?ps=10&pn=1

End Point: /InfoSearch/page_movies/

ps is page size which is equals 10
pn is page number which is equals 1

(b) Pagination for tvshows :

Query(sample) : localhost:3000/InfoSearch/page_tvshows/?ps=10&pn=1

End Point: /InfoSearch/page_tvshows/

ps is page size which is equals 10
pn is page number which is equals 1

C.

(a) Exact match Endpoint :

We have extracted all the 2-grams from the given name. For each 2-gram we have “and” the term search.

Query(sample): localhost:3000/InfoSearch/exact_match/?name=Jorge Michel Grau

End Point: /InfoSearch/exact_match/
Query name: Jorge Michel Grau

(b) Prefix Match Endpoint :

Query(sample):
localhost:3000/InfoSearch/prefix_match/after

Endpoint: /InfoSearch/prefix_match/
Query phrage: after

After will be searched for 1st term in “description” field.

(c) General Match Endpoint :

Query(sample):
localhost:3000/InfoSearch/genre_match/?q_and=drama&q1_or=comedy&q2_or=horror

End Point: /InfoSearch/genre_match/

q1_or: 1st category to be or with 2nd one
q2_or: 2nd category to be or with 1st one
q_and: This category to be and with both of them