## ASSIGNMENT III

**The assignment is related to the content taught in** [**WEEK II**](https://docs.google.com/document/d/10lLTy0O-IZoMlVUJ76HvgKXC8xg6Es48tqqMlBw0nrc/edit?usp=sharing) **and** [**WEEK III**](https://docs.google.com/document/d/1ItYMpdjcebQsZjJoTBTeIidzYIdkKgNw00mFDBemyd4/edit?usp=sharing)**. The concept of different types of queries taught in the class will be refreshed if you cover this assignment.**

*NOTE:*

*For doing these questions, you require executing curl command for bulk insertion of test-data.json into products index if you have not done it already. Also, you require executing curl command for bulk insertion of poetry.json into shakespeare index if you have not already.*

1. Please write a query matching products that didn’t sell very well, being products where the “sold” field has a value of less than 10 (sold < 10).

GET product/\_search

{

"query": {

"range": {

"sold": {

"lte": 10

}

}

1. Write a query that matches products that sold okay, meaning less than 30 and greater than or equal to 10 (sold < 30 && sold >= 10).

GET product/\_search

{

"query": {

"range": {

"sold": {

"gte": 10,

"lte": 30

}

}

}

}

1. Write a query that matches documents containing the term “Meat” within the “tags” field.

GET product/\_search

{

"query":{

"match":{

"tags":"meat"

}

}

}

1. Write a query matching documents containing one of the terms "Tomato" and "Paste" within the "name" field.

GET /product/\_search{

"query":{

"match": {

"name":{

"query": "Tomato Paste",

"lenient": "true",

"operator": "and"

}

}

}

}

1. Write a query that matches products with a "name" field including “pasta”, “paste”, or similar. The query should be dynamic and not use the "terms" query clause.

[Hint: match by prefix/ wildcard]

GET /product/\_search

{

"query":{

"prefix": {

"name": “paste”

}

}

}

1. Check and mention how many documents have non-empty tags field in product index.

GET /product/\_search

{

"query": {

"bool": {

"must": {

"exists":{

"field":"tags"

}

}

}

}

}

1. Find which documents of shakespeare index contain the following terms:

men, holy, fight, prince, blessed, conquest, war, knife

The number of above terms matched in a document must be at least half of number of words in the document. Refer to **terms\_set** query taught in Week II.

GET /shakespeare/\_search

{

"query":{

"terms\_set":{

"text\_entry":{

"terms":

["men", "holy", "fight", "prince", "blessed", "conquest", "war", "knife"],

"minimum\_should\_match\_script":{

"source":"params.num\_terms/2"

}

}

}

}

}

1. Look for document having the phrase **some night-tripping fairy.**

GET /shakespeare/\_search

{

"query":{

"match\_phrase": {

"text\_entry": {

“query": “some night-tripping fairy"

}

}

}

}

1. Find documents containing the words that are fuzzily similar to “sape of likehood”. Try using a fuzziness of 2. Also, make sure that all three words are available in the document that is returned.

GET /shakespeare/\_search

{

"query": {

"match": {

"text\_entry": {

"query": "sape of likehood",

"fuzziness": 2,

"operator":"and"

}

}

}

}

1. Find all documents that had **Henry IV** in fields **play\_name** and **speaker**.

GET /shakespeare/\_search

{

"query": {

"multi\_match": {

"query": "Henry IV",

"fields": ["play\_name","speaker"]

, "operator": "and"

}

}

}

1. Practice using cut-off frequency to handle domain specific stop-words in match query and common-terms query. You can consider shakespeare index’s text-entry field for this purpose.

GET /shakespeare/\_search

{

"query": {

"match": {

"text\_entry": {

"query": " lady in the light",

"cutoff\_frequency": 0.002

}

}

}

}

1. Find documents from **product** index that match the following criteria. Fit in all these criteria in a single query.

* That is currently active, has in\_stock of at least 10 and has either wine or meat or both in tags.
* Filter the documents having price of at least 150.
* May or may not have 300 or more items sold.

1. Refer to **Nested** query in [**Week III Notes**](https://docs.google.com/document/d/1ItYMpdjcebQsZjJoTBTeIidzYIdkKgNw00mFDBemyd4/edit?usp=sharing) to do the following question:
2. Create an index **college** having following fields:

* batch (integer type): example values, 2017, 2018
* students (nested type, i.e. array of inner objects): each inner object can have three properties **id**, **name** and **age**.

1. Insert a document with certain id (example, 1), your batch (example, 2017), and an array of 3 students in index **college**.
2. Use nested query to find parent documents having any student with age greater than 10. Along with parent documents, the inner hits should also be shown.

PUT college3

{

"mappings": {

"\_doc": {

"properties": {

"batch": {

"type": "integer"

},

"students": {

"type": "nested",

"properties": {

"roll": {

"type": "integer"

},

"name":{

"type":"text"

},

"age":{

"type":"byte"

}

}

}

}

}

}

}

#Inserting document

PUT college3/\_doc/1

{

"batch": 2019,

"students": [

{

"roll": 501,

"name": "Rachana Banjade",

"age": 21

},

{

"roll": 502,

"name": "Sabina Shrestha",

"age": 21

},

{

"roll": 503,

"name": "Alisha Pathak",

"age": 21

}

]

}

#query

GET college3/\_search

{

"query": {

"nested": {

"path": "students",

"ignore\_unmapped":true,

"inner\_hits": {},

"query": {

"range": {

"students.age": {

"gt": 10

}

}

}

}

}

}

1. Create a **filtered alias** of documents of **products** index having **is\_active: true**.

POST /\_aliases

{

"actions" : [

{

"add" : {

"indices":["product"],

"alias" : "active\_products",

"filter" : {

"term" : {

"is\_active": "true"

}

}

}

}

]

}