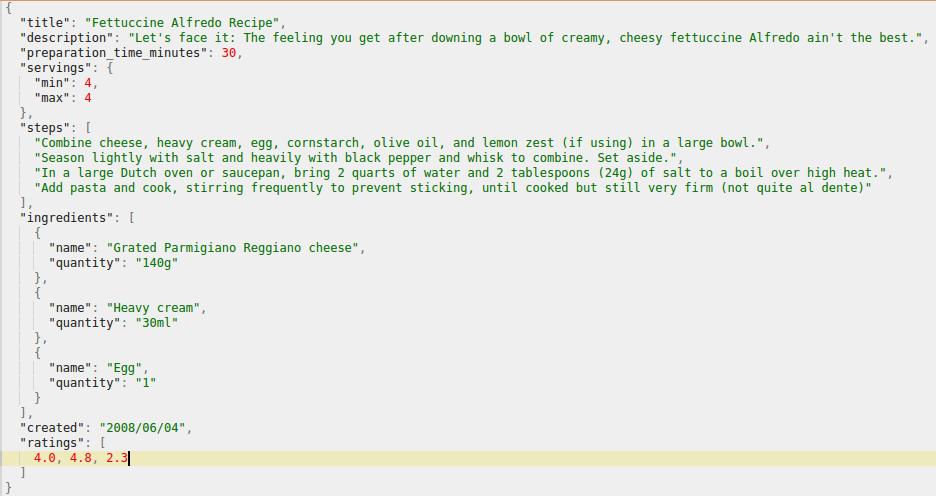
# MAJOR ASSESSMENT

## Related to recipe.json

1) Look at the sample json data in the picture below.



Picture I: Sample JSON Data

Create an **index with any name** that you prefer (example: recipes) along with **mapping**. When creating mapping:

* decide on what are the fields required and their respective types that should be available in the mapping of the index by looking at the sample json data in the picture above (Picture I: Sample JSON Data).

PUT recipes

{

"mappings": {

"\_doc": {

"dynamic": false,

"properties": {

"title": {

"type": "text"

},

"description": {

"type": "text"

},

"preparation\_time\_minutes": {

"type": "integer"

},

"servings": {

"type": "nested",

"properties": {

"min": {

"type": "integer"

},

"max": {

"type": "integer"

}

}

},

"steps":

{

"type":"text"

},

"ingredients": {

"type": "nested",

"properties": {

"name": {

"type": "text"

},

"quantity": {

"type": "text"

}

}

},

"created": {

"type": "date",

"format": "MM-dd-yyyy || yyyy/mm/dd"

},

"ratings": {

"type": "float"

}

}

}

}

}

* create **custom analyzer** (you can provide it any name, example: new\_analyzer) that can handle english stop-words and synonyms available in a file created inside the config directory of elasticsearch (example, synonyms\_recipes.txt). In order to add synonyms, you can download the file in [**this link**](https://drive.google.com/file/d/1PLc4lG2hAU7kXlRhhM0Jvt0GkpCXMZJo/view?usp=sharing)**,** open it in notepad or any other text editor to see what kind of synonyms can be created and add the synonyms that you have thought of in synonyms\_recipes.txt file following proper format. The custom analyzer should be provided to **description** and **steps** fields.

PUT recipes\_index

{

"settings": {

"index": {

"analysis": {

"analyzer": {

"recipes\_analyzer": {

"type": "custom",

"tokenizer": "standard",

"filter": [

"lowercase",

"my\_stop",

"my\_synonyms"

]

}

},

"filter": {

"my\_stop": {

"type": "stop",

"stopwords": "\_english\_"

},

"my\_synonyms": {

"type": "synonym",

"synonyms": [

"fast=>speedy, quick, swift, rapid",

"easy=>simple, plain",

"large=> big, great, huge",

"serve=> dish out, give out, distribute, set out, plate up, spoon out, ladle out"

]

}

}

}

}

},

"mappings": {

"\_doc": {

"properties": {

"desc": {

"type": "text",

"analyzer": "recipes\_analyzer"

}

}

}

}

}

POST recipes\_index/\_analyze

{"analyzer": "recipes\_analyzer",

"text":"The service in the newly opened resturant is fast, it is easy to reach there and the food is serve in a large bowl."}

PUT recipes\_index/\_doc/1

{"desc":"The service in the newly opened resturant is fast, it is easy to reach there and the food is served in a large bowl."}

GET recipes\_index/\_doc/\_search

2) **Bulk insert** the data downloaded from [**this link**](https://drive.google.com/file/d/1PLc4lG2hAU7kXlRhhM0Jvt0GkpCXMZJo/view?usp=sharing) into the newly created index. The data has different recipes as documents.

curl -H "Content-Type: application/json" -XPOST "[http://localhost:9200/recipes/\_doc/\_bulk?pretty](http://localhost:9200/accounts/_doc/_bulk?pretty)" --data-binary @recipe.json

3) For documents having preparation\_time\_minutes of less than of equal to 15, the difference between min and max servings should be at least 1 and at most 3 (i.e. the difference between min and max servings should be between 1-3, inclusive). Write update by query using script to make sure that this case prevails.

4) Search for all the documents in the index. View the nested field **ingredients** of the documents. You can see that in some documents, some ingredients are missing quantity field. First, find the count of such documents. Then, update such documents in which at least one of the ingredients is missing quantity field by adding the quantity field with value ‘Per Choice’.

**Hint**: ingredients is a nested field (array of JSON objects), so you need to perform nested query along with update by query and script to do this question.

**Side Note**: First perform nested query to see how many such documents exist. After executing nested query, you can see that each document/hit that is returned has at least one JSON object that match the nested query condition in the array of JSON objects of the nested field. Only then, try performing update by query. Following this approach helps you to verify your work properly.

5) Delete unrated documents, that is, documents that have empty array in ratings field.

POST recipes/\_delete\_by\_query

{

"query":{

"terms":{

"ratings":[]

}

}

}

6) Find all the recipes that use Egg as one of the ingredients. Display only title, ratings, steps, number of steps. Note that number of steps is a derived field. The documents should be ordered by average rating.

GET recipes/\_search

{

"\_source": [

"title",

"ratings",

"steps"

],

"query": {

"nested": {

"path": "ingredients",

"query": {

"match": {

"ingredients.name": "egg"

}

}

}

},

"script\_fields": {

"number\_of\_steps": {

"script": {

"lang": "painless",

"source": "params['\_source']['steps'].size()"

}

}

},

"sort": [

{

"ratings": {

"mode": "avg"

}

}

]

}

7) Execute at least 10 different aggregations for performing analysis on the data.

* You need to prepare at least 3 different metric aggregations

GET accounts/\_doc/\_search

{"size":0,

"aggs":{

"avg\_balance":{

"avg":{

"field":"balance"

}

}

}}

GET accounts/\_doc/\_search

{"size":0,

"aggs":{

"avg\_balance":{

"avg":{

"script":"doc['balance'].value"

}

}

}}

GET accounts/\_doc/\_search

{"size":0,

"aggs":{

"avg\_balance":{

"avg":{

"field":"balance",

"missing":100

}

}

}}

* You need to prepare at least 5 different bucket aggregations

1. Date Range Aggregation

GET test/\_doc/\_search

{"size":0,

"aggs": {

"created\_range": {

"date\_range": {

"field": "created",

"format": "YYY/MM/DD",

"ranges": [

{

"to": "now-10M/M"

},

{

"from": "now-10M/M"

}

]

}

}

}

}

1. Range Aggregation

GET accounts/\_doc/\_search

{

"size": 0,

"aggs": {

"age\_range": {

"range": {

"field": "age",

"ranges": [

{

"to": 15

},

{

"from": 25,

"to": 40

}

]

}

}

}

}

GET accounts/\_doc/\_search

1. Date histogram

GET orders/\_doc/\_search

{

"size": 0,

"aggs": {

"order\_over\_month": {

"date\_histogram": {

"field": "purchased\_at",

"interval": "month",

"format":"MM-yyyy"

}

}

}

}

1. Terms Aggregation

GET orders/\_doc/\_search

{

"aggs" : {

"five\_status" : {

"terms" : {

"field" : "amount",

"size" : 10

}

}

}

}

1. Filter Aggregation

POST orders/\_search

{

"aggs" : {

"Sale\_channels" : {

"filter" : { "term": { "sales\_channel":"web" } },

"aggs" : {

"avg\_amount" : { "avg" : { "field" : "amount" } }

}

}

}

}

You need to prepare at least 2 different sub aggregations

* At least 1 sub aggregation having metric aggregation within bucket aggregation

GET orders/\_doc/\_search

{

"size":0,

"aggs":{

"sells":{

"filter":{

"term":{"sales\_channel":"phone"}

},

"aggs":{

"average\_amount":{

"avg":{

"field":"total\_amount"

}

}

}

}

}

}

* At least 1 sub aggregation having bucket aggregation within another bucket aggregation

GET orders/\_doc/\_search

{

"size": 0,

"aggs": {

"sales\_from": {

"terms": {

"field": "sales\_channel"

},

"aggs": {

"status\_orders": {

"terms": {

"field": "status"

}

}

}

}

}

}

8) Design your own search using compound bool query. The bool query should have at least 2 musts, at least 2 filters and at least 1 should. Showcase the use of synonyms, proximity (slop) and fuzziness parameters where possible in your search query.

GET accounts/\_search

{

"query": {

"bool": {

"must": [

{

"match\_phrase": {

"employer": {

"query": "Irack",

"slop": 2

}

}

},

{

"fuzzy": {

"city": {

"value": "Shaft",

"fuzziness": 2

}

}

}

],

"filter": [

{

"term": {

"lastname": "Wooten"

}

},

{

"range": {

"age": {

"lte": 45

}

}

}

],

"should": [

{

"range": {

"balance": {

"gt": 20000

}

}

}

]

}

}

}

GET accounts/\_doc/\_search

9) Practice using cut-off frequency to handle domain specific stop-words in match query and common-terms query. You can make use of **steps** field for this purpose.

GET recipes/\_search

{

## "query": {

## "match": {

## "description": {

## "query": "one fewer ingredient",

## "cutoff\_frequency": 11

## }

## }

## }

## }

## GET recipes/\_search

## {

## "query": {

## "match": {

## "description": {

## "query": "one fewer ingredient ",

## "cutoff\_frequency": 0.001

## 

## }

## }

## }

## }

## GET shakespeare/\_search

## {"size":20,

## "query": {

## "common": {

## "text\_entry": {

## "query": "I should be hanged",

## "cutoff\_frequency": 0.001,

## "low\_freq\_operator": "and"

## }

## }

## }

## }

## GET shakespeare/\_search

## {

## "query": {

## "common": {

## "text\_entry": {

## "query": "I should be hanged",

## "cutoff\_frequency": 0.005,

## "high\_freq\_operator": "or"

## }

## }

## }

## }

## Terms Lookup Mechanism and Geo Queries

10)

1. Create an index **items** with mapping having following fields:

**item\_id**: integer field

**name**: text field

**stock**: integer field

**vendor**: object with properties name (text field), contact (keyword), address(geo\_point)

PUT items

{

"mappings": {

"\_doc": {

"dynamic": false,

"properties": {

"item\_id": {

"type": "integer"

},

"name": {

"type": "text"

},

"stock": {

"type": "integer"

},

"vendor": {

"properties": {

"name": {

"type": "text"

},

"contact": {

"type": "keyword"

},

"address": {

"type": "geo\_point"

}

}

}

}

}

}}

1. Bulk insert at least 5 documents into **items** index.

POST items/\_doc/\_bulk

{"index":{"\_id":1}}

{"item\_id":"1234","name":"books","stock":567,"vendor":{"name":"Sweekriti","contact":"982900189209","address":{"lat":30.9,"lon":-50.34}}}

{"index":{"\_id":2}}

{"item\_id":"1829","name":"marker","stock":1000,"vendor":{"name":"Shreeya","contact":"9883829783","address":{"lat":20.9,"lon":-40.34}}}

{"index":{"\_id":3}}

{"item\_id":"6738","name":"copy","stock":899,"vendor":{"name":"Rachana","contact":"983892292","address":{"lat":23.9,"lon":-42.13}}}

{"index":{"\_id":4}}

{"item\_id":"5283","name":"duster","stock":721,"vendor":{"name":"Ayusha","contact":"98777732928","address":{"lat":34.49,"lon":-50.34}}}

{"index":{"\_id":5}}

{"item\_id":"1234","name":"eraser","stock":567,"vendor":{"name":"Sovit","contact":"982988937827","address":{"lat":28.31,"lon":-47.25}}}

1. Try performing geo bounding box and geo distance queries in the vendor’s address.

GET items/\_search

{"query":{

"bool": {

"filter": {

"geo\_bounding\_box": {

"vendor.address": {

"top\_left": {

"lat": 40.85,

"lon": -40

},

"bottom\_right": {

"lat": 35,

"lon": 30

}

}

}

}

}

}}

GET items/\_search

1. Create an index **category\_items** with mapping having following fields:

**category:** text field

**Items:** array of item\_ids, that means, array of integer

PUT category\_items

{

"mappings": {

"\_doc": {

"dynamic": false,

"properties": {

"category": {

"type": "text"

},

"items": {

"properties": {

"items\_ids": {

"type": "integer"

}

}

}

}

}

}

}

1. Insert two documents in category\_items: one related to cosmetic category and next related to household category. When inserting documents in category\_items, please note that each category should have at least one item\_id inserted in items index.

POST category\_items /\_doc/1

{

"category": "cosmetic",

"items": {

"items\_ids": [

541,

543

]

}

}

POST category\_items /\_doc/2

{

"category": "household",

"items": {

"items\_ids": [

789,

456

]

}

}

1. Using terms query with terms lookup mechanism, find the items from items index that belong to cosmetic category.

GET category\_items/\_search

{

"query" : {

"terms" : {

"category" : {

"index" : "category\_items",

"type" : "\_doc",

"id" : "1",

"path" : "category"

}

}

}

}

## Related to [orders-bulk.json](https://drive.google.com/file/d/1z7xKaCZF7WgCmqvyu2sX7KnsVbw6Zddi/view?usp=sharing)

11) Create filtered alias of documents of orders index fulfilling the following conditions:

* status: processed **or** completed
* sales\_channel: phone **and** app
* Having total\_amount >=100

POST /\_aliases

{

"actions": [

{

"add": {

"index": "orders",

"alias": "orders\_alias",

"filter": {

"bool": {

"must": [

{

"query\_string": {

"default\_field": "status",

"query": "(processed) OR (completed)"

}

},

{

"query\_string": {

"default\_field": "sales\_channel",

"query": "(phone) AND (app)"

}

},

{

"range": {

"total\_amount": {

"gte": 100

}

}

}

]

}

}

}

}

]

}

12) Perform the following queries in the filtered alias. Design your search conditions yourself.

* Term query

GET orders\_alias/\_search

{

"query": {

"term": {

"purchased\_at": "2016-01-31T13:06:58Z"

}

}

}

GET orders\_alias/\_search

* Range query

GET orders\_alias/\_search

{

"query": {

"range": {

"total\_amount": {

"gte": 120,

"lte": 150

}

}

}

GET orders\_alias/\_search

* Prefix query

GET orders\_alias/\_search

{

"query": {

"prefix": {

"sales\_channel": "st"

}

}

}

GET orders\_alias/\_search

* Wildcard Query

GET orders\_alias/\_search

{

"query": {

"wildcard": {

"salesman.name": "E\*e"

}

}

}

* Match

GET orders\_alias/\_search

{

"query": {

"match": {

"salesman.name": "Uri"

}

}

}

* Fuzzy Match

GET orders\_alias/\_search

{

"query": {

"match": {

"salesman.name": {

"query": "Annissa Cleever",

"fuzziness": 2

## }

## }

## }

## }

## Creativity Check

* Create an index of your own choice with mapping having fields of different types. Consider using all the types that we have learnt so far. Prefer creating analyzer and try using synonyms and stop-words filters in the analyzer.

PUT cafe

{

"settings": {

"index": {

"analysis": {

"analyzer": {

"new\_analyzer": {

"type": "custom",

"tokenizer": "standard",

"filter": [

"my\_stop",

"my\_synonyms"

]

}

},

"filter": {

"my\_stop": {

"type": "stop",

"stopwords": "\_english\_"

},

"my\_synonyms": {

"type": "synonym",

"synonyms": [

"location=>place, site",

"good=>quality, superior"

]

}

}

}

}

},

"mappings": {

"\_doc": {

"properties": {

"desc": {

"type": "text",

"analyzer": "new\_analyzer"

},

"name": {

"type": "text"

},

"address": {

"type": "keyword"

},

"contact": {

"type": "long"

},

"salary": {

"type": "double"

},

"paid": {

"type": "boolean"

},

"location": {

"type": "geo\_point"

},

"started": {

"type": "date",

"format": "MM-dd-yyyy || yyyy/mm/dd"

},

"expected\_employee": {

"type": "integer\_range"

},

"menu": {

"type": "nested",

"properties": {

"food\_name": {

"type": "text"

},

"price": {

"type": "integer"

},

"description": {

"type": "text"

}

}

},

"manager": {

"properties": {

"age": {

"type": "integer"

},

"about": {

"properties": {

"first": {

"type": "text"

},

"last": {

"type": "text"

}

}

}

}

}

}

}

}

}

* Bulk insert at least 10 documents creating a json file yourself and by using curl.

POST cafe/\_doc/\_bulk

{"index":{"\_id":1}}

{"name":"A1 Cafe","address":"Chabahil","contact":"9812345600","salary":25000,"paid":true,"location":{"lat":41.12,"lon":-71.34},"started":"12-11-2018","expected\_employee":{"gte":10,"lte":20},"menu":{"food\_name":"Chickenchilly","price":100,"description":"Very tasty food"},"manager":{"age":30,"name":{"first":"John","last":"Smith"}}}

{"index":{"\_id":2}}

{"name":"Apple Cafe","address":"Old Baneshwor","contact":"9818298392","salary":35000,"paid":false,"location":{"lat":31.12,"lon":-23.31},"started":"10-9-2018","expected\_employee":{"gte":20,"lte":50},"menu":{"food\_name":"Hot Chocolate","price":500,"description":" Hot chocolate may be topped with whipped cream or marshmallows"},"manager":{"age":25,"name":{"first":"Hary","last":"Pal"}}}

{"index":{"\_id":3}}

{"name":"G-cafe","address":"Boudhar","contact":"98892382","salary":5000,"paid":true,"location":{"lat":21.13,"lon":-68.45},"started":"10-11-2017","expected\_employee":{"gte":50,"lte":100},"menu":{"food\_name":"Chickenmomo","price":300,"description":"Very good food for Nepalese"},"manager":{"age":45,"name":{"first":"Henry","last":"Jason"}}}

{"index":{"\_id":4}}

{"name":"Mechung","address":"Tushal","contact":"98839836","salary":3000,"paid":true,"location":{"lat":29.03,"lon":-51.14},"started":"12-11-2012","expected\_employee":{"gte":20,"lte":40},"menu":{"food\_name":"Brownie","price":100,"description":"A chocolate brownie is a square, baked, chocolate dessert. Brownies come in a variety of forms and may be either fudgy or cakey, depending on their density."},"manager":{"age":51,"name":{"first":"Alexander","last":"Smith"}}}

{"index":{"\_id":5}}

{"name":"Orange Cafe","address":"Patan","contact":"98932839","salary":4500,"paid":true,"location":{"lat":21.12,"lon":-51.34},"started":"9-6-2013","expected\_employee":{"gte":20,"lte":50},"menu":{"food\_name":"Spring roll","price":100,"description":"Spring rolls are a large variety of filled, rolled appetizers. Best spring rolls in this location"},"manager":{"age":61,"name":{"first":"Riya","last":"Jackson"}}}

{"index":{"\_id":6}}

{"name":"Bakery Cafe","address":"Chucchepati","contact":"98849384","salary":1000,"paid":true,"location":{"lat":34.12,"lon":-71.51},"started":"10-5-2010","expected\_employee":{"gte":50,"lte":75},"menu":{"food\_name":"Idli","price":100,"description":"Originating from the Indian subcontinent, popular as breakfast foods in southern India and northern Sri Lanka has a good taste"},"manager":{"age":52,"name":{"first":"Jennifer","last":"Krisht"}}}

{"index":{"\_id":7}}

{"name":"Zoom Cafe","address":"Santinagar","contact":"98874837","salary":1500,"paid":true,"location":{"lat":35.63,"lon":-61.54},"started":"11-10-2015","expected\_employee":{"gte":10,"lte":20},"menu":{"food\_name":"Chicken pizza","price":450,"description":"savory dish of Italian origin, consisting of a usually round, flattened base of leavened wheat-based dough topped with tomatoes, cheese, and various other ingredients baked at a high temperature location, traditionally in a wood-fired oven."},"manager":{"age":35,"name":{"first":"David","last":"Woods"}}}

{"index":{"\_id":8}}

{"name":"Everest Cafe","address":"Bhatbhateni","contact":"989378293","salary":7000,"paid":true,"location":{"lat":21.67,"lon":-71.34},"started":"11-11-2017","expected\_employee":{"gte":32,"lte":50},"menu":{"food\_name":"Kati Roll","price":150,"description":"A kati roll is a street-food dish originating from Kolkata, India. In its original form, it is a skewer-roasted kebab wrapped in a paratha bread, although over the years many variants have evolved all of which now go under the generic name of kati roll. It is a good choice for snacks"},"manager":{"age":55,"name":{"first":"Bishnu","last":"Shrestha"}}}

{"index":{"\_id":9}}

{"name":"Hyderbadi house","address":"Tangal","contact":"989327893","salary":15000,"paid":true,"location":{"lat":52.13,"lon":-67.89},"started":"10-5-2010","expected\_employee":{"gte":50,"lte":70},"menu":{"food\_name":"Biryani","price":650,"description":"Biryani, also known as biriyani, biriani, birani or briyani, is a mixed rice dish that originates from the Muslims of the Indian subcontinent. It is a common choice for customers in this location "},"manager":{"age":67,"name":{"first":"Krishna","last":"Shyam"}}}

{"index":{"\_id":10}}

{"name":"Palpasa Cafe","address":"Bhaktapur","contact":"989327893","salary":15000,"paid":true,"location":{"lat":52.13,"lon":-67.89},"started":"10-5-2015","expected\_employee":{"gte":50,"lte":70},"menu":{"food\_name":"Panner","price":650,"description":"Panneer dish that originates from the Muslims of the Indian subcontinent. It is a common choice for customers in this location also a good choice"},"manager":{"age":57,"name":{"first":"Ammy","last":"Smith"}}}

{"index":{"\_id":11}}

{"name":"Palpasa Cafe","address":"Bhaktapur","contact":"989327893","salary":25000,"paid":true,"location":{"lat":52.13,"lon":-67.89},"started":"10-5-2015","expected\_employee":{"gte":50,"lte":70},"menu":{"food\_name":"Panner","price":650,"description":"Panneer dish that originates from the Muslims of the Indian subcontinent. It is a common choice for customers in this location also a good choice"},"manager":{"age":57,"name":{"first":"Ammy","last":"Smith"}}}

GET cafe/\_doc/\_search

Using Curl:

curl -H "Content-Type: application/json" -XPOST "http://localhost:9200/cafe/\_doc/\_bulk?pretty" --data-binary @cafe.json

* Perform as many different searches and aggregations as you would like to for analyzing various aspects of your data.

1. Term Query:

GET cafe/\_search

{

"query": {

"term": {

"name": {

"value": "Orange Cafe"

}

}

}

}

GET cafe/\_doc/\_search

1. Range Query on salary

GET cafe/\_search

{

"query": {

"range" : {

"salary": {

"gte" : 1000,

"lte" : 5000

}

}

}

}

1. Prefix Query

GET cafe/\_search

{"query":{

"prefix":{"address":"Bh"}

}}

1. Find ids 1 and 3

GET cafe/\_search

{

"query": {

"ids": {

"values": [

1,

3

]

}

}

}

1. Fuzzy Query

GET cafe/\_search

{

"query": {

"fuzzy" : {

"name" : {

"value": "Zoom",

"fuzziness": 2,

"max\_expansions":1

}

}

}

}

1. Metric Aggregation

GET cafe/\_doc/\_search

{"size":0,

"aggs":{

"avg\_salary":{

"avg":{

"field":"salary"

}

}

}}

1. Sum, min and max aggregation

GET cafe/\_doc/\_search

{

"size": 0,

"aggs": {

"sum\_of\_salary": {

"sum": {

"field": "salary"

}

},

"max\_salary": {

"max": {

"field": "salary"

}

},

"min\_salary": {

"min": {

"field": "salary"

}

}

}

}

1. Metric Aggregation within bucket aggregation

GET cafe/\_doc/\_search

{

"size":0,

"aggs":{

"sold\_from\_store":{

"filter":{

"term":{"address":"Bhaktapur"}

},

"aggs":{

"average\_salary":{

"avg":{

"field":"salary"

}

}

}

}

}

}

1. Date histogram

GET cafe/\_doc/\_search

{

"size": 0,

"aggs": {

"started\_date": {

"date\_histogram": {

"field": "started",

"interval": "year",

"format":"MM-dd-yyyy"

}

}

}

}

1. Filter Aggregation

GET cafe/\_doc/\_search

{

"query": {

"match": {

"address": {

"query": "Bhaktapur"

}

}

},

"highlight": {

"fields": {

"address": {}

}

}

}