Database Project (SWE3033) (Fall 2023)

Homework #8 (50pts, Due date: 11/22)

Student ID: 2020315798

Student Name: Choi Jin Woo

Instruction: The objective of this homework is to construct a Python Elasticsearch client class than can be used to interact with an Elasticsearch cluster. Implement '**ElasticsearchClient'** class in 'elasticsearch_client.py' and use the correct function for the annotation in 'main.ipynb'.

We have provided the datasets and all the Python files you need. The dataset is about movie reviews and is provided in the form of JSON file. Please write your code to get the correct result. **If you edit anything other than EDIT HERE, you may be penalized.**

Submit two files as follows:

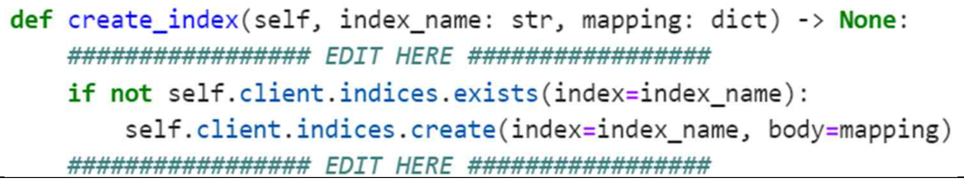
- 'DBP_Homework8_STUDENTID.zip'
  - Code:
    - main.ipynb
    - elasticsearch_client.py
  - Document: DBP_Homework8_STUDENTID.pdf

1. **[40pts]** Implement the various functions for common Elasticsearch operations at **'elasticsearch_client.py'** and report the result at **'main.ipynb'**.

   A. **[30pts]** Implement the following functions.

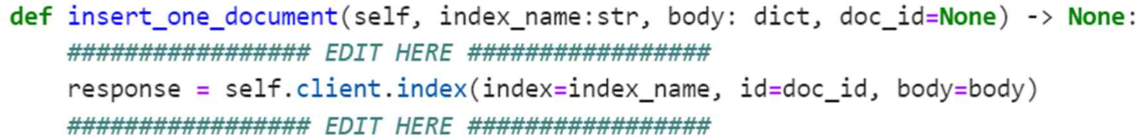
   Answer: Enter your **codes** here.

[def create_index]



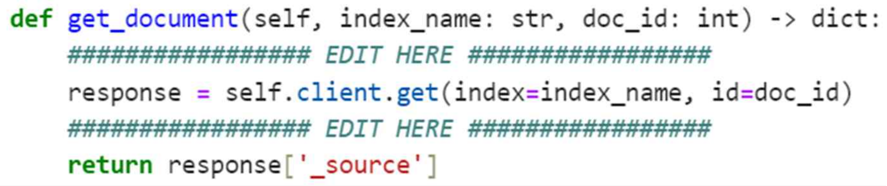
```
def create_index(self, index_name: str, mapping: dict) -> None:
    ################## EDIT HERE ##################
    if not self.client.indices.exists(index=index_name):
        self.client.indices.create(index=index_name, body=mapping)
    ################## EDIT HERE ##################
```

[def insert_one_document]



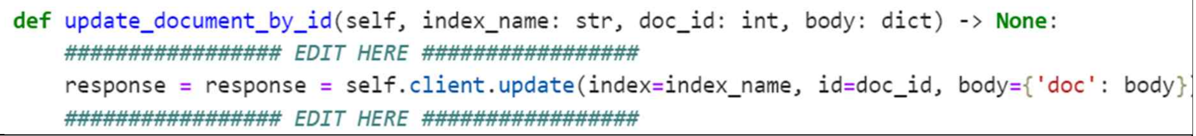
```
def insert_one_document(self, index_name:str, body: dict, doc_id=None) -> None:
    ################## EDIT HERE ##################
    response = self.client.index(index=index_name, id=doc_id, body=body)
    ################## EDIT HERE ##################
```

[def get_document]



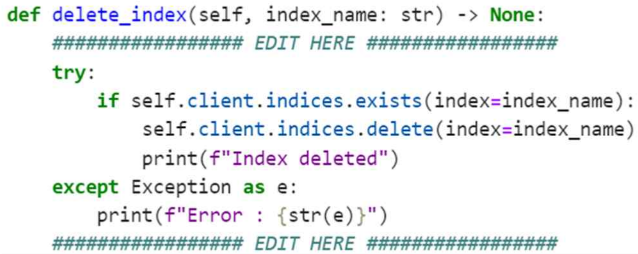
```
def get_document(self, index_name: str, doc_id: int) -> dict:
    ################# EDIT HERE #################
    response = self.client.get(index=index_name, id=doc_id)
    ################# EDIT HERE #################
    return response['_source']
```

[def update_document_by_id]



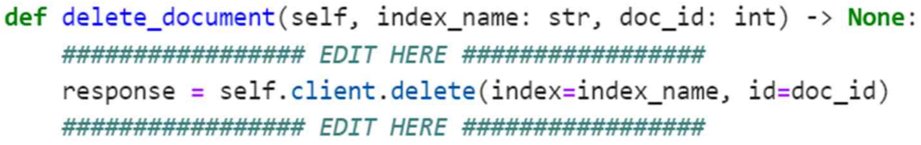
```
def update_document_by_id(self, index_name: str, doc_id: int, body: dict) -> None:
    ################# EDIT HERE #################
    response = response = self.client.update(index=index_name, id=doc_id, body={'doc': body}
    ################# EDIT HERE #################
```

[def delete_index]



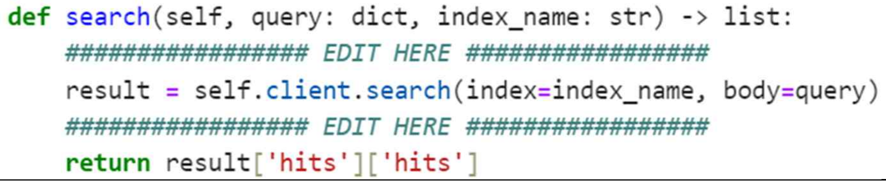
```
def delete_index(self, index_name: str) -> None:
    ################# EDIT HERE #################
    try:
        if self.client.indices.exists(index=index_name):
            self.client.indices.delete(index=index_name)
            print(f"Index deleted")
    except Exception as e:
        print(f"Error : {str(e)}")
    ################# EDIT HERE #################
```

[def delete_document]



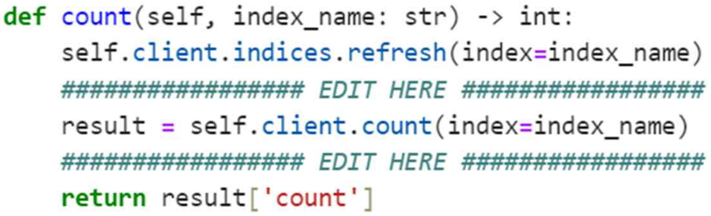
```
def delete_document(self, index_name: str, doc_id: int) -> None:
    ################# EDIT HERE #################
    response = self.client.delete(index=index_name, id=doc_id)
    ################# EDIT HERE #################
```

[def search]



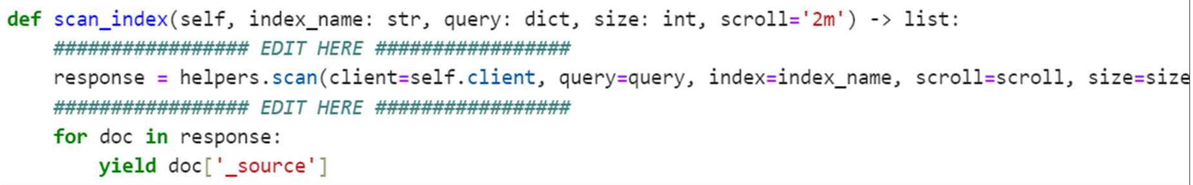
```
def search(self, query: dict, index_name: str) -> list:
    ################# EDIT HERE #################
    result = self.client.search(index=index_name, body=query)
    ################# EDIT HERE #################
    return result['hits']['hits']
```

[def count]



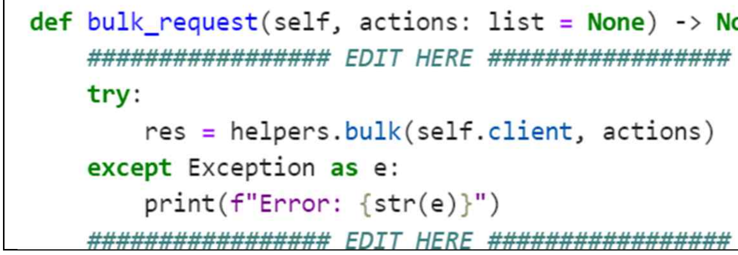
```
def count(self, index_name: str) -> int:
    self.client.indices.refresh(index=index_name)
    ################# EDIT HERE #################
    result = self.client.count(index=index_name)
    ################# EDIT HERE #################
    return result['count']
```

[def scan_index]



```
def scan_index(self, index_name: str, query: dict, size: int, scroll='2m') -> list:
    ################# EDIT HERE #################
    response = helpers.scan(client=self.client, query=query, index=index_name, scroll=scroll, size=size
    ################# EDIT HERE #################
    for doc in response:
        yield doc['_source']
```

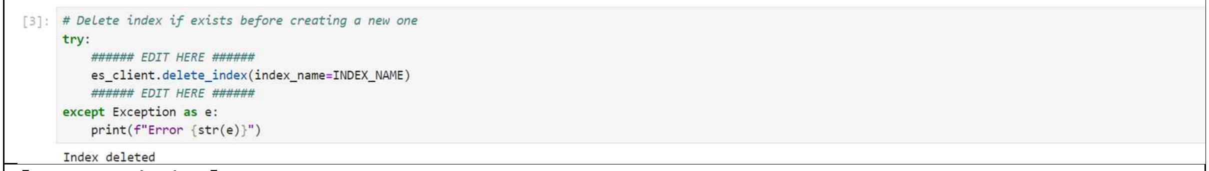
[def bulk_request]



```
def bulk_request(self, actions: list = None) -> None:
    ################# EDIT HERE #################
    try:
        res = helpers.bulk(self.client, actions)
    except Exception as e:
        print(f"Error: {str(e)}")
    ################# EDIT HERE #################
```

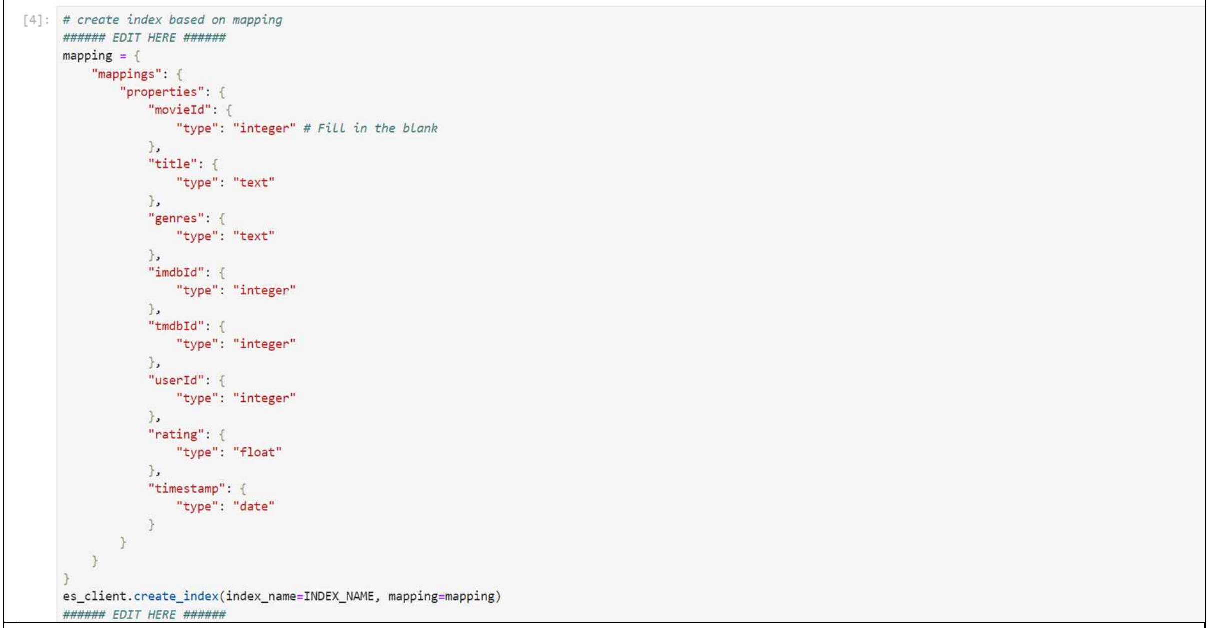
B. **[10pts]** Use the correct function to get the correct result. Enter your **codes** and **results** here. You must take a screenshot of the result. If there is no return value, you don't ne ed to report the result.

[delete index]



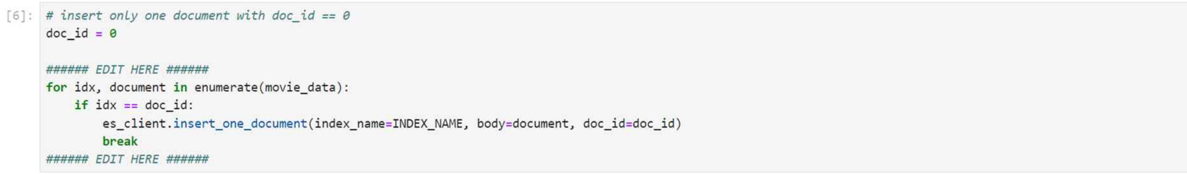
```
[3]: # Delete index if exists before creating a new one
     try:
         ###### EDIT HERE ######
         es_client.delete_index(index_name=INDEX_NAME)
         ###### EDIT HERE ######
     except Exception as e:
         print(f"Error {str(e)}")

     Index deleted
```

[create index]



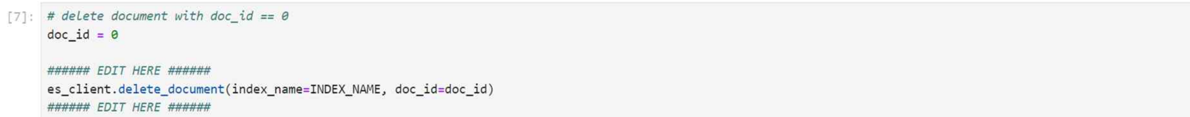
```
[4]: # create index based on mapping
     ###### EDIT HERE ######
     mapping = {
         "mappings": {
             "properties": {
                 "movieId": {
                     "type": "integer" # Fill in the blank
                 },
                 "title": {
                     "type": "text"
                 },
                 "genres": {
                     "type": "text"
                 },
                 "imdbId": {
                     "type": "integer"
                 },
                 "tmdbId": {
                     "type": "integer"
                 },
                 "userId": {
                     "type": "integer"
                 },
                 "rating": {
                     "type": "float"
                 },
                 "timestamp": {
                     "type": "date"
                 }
             }
         }
     }
     es_client.create_index(index_name=INDEX_NAME, mapping=mapping)
     ###### EDIT HERE ######
```

[insert only one document]



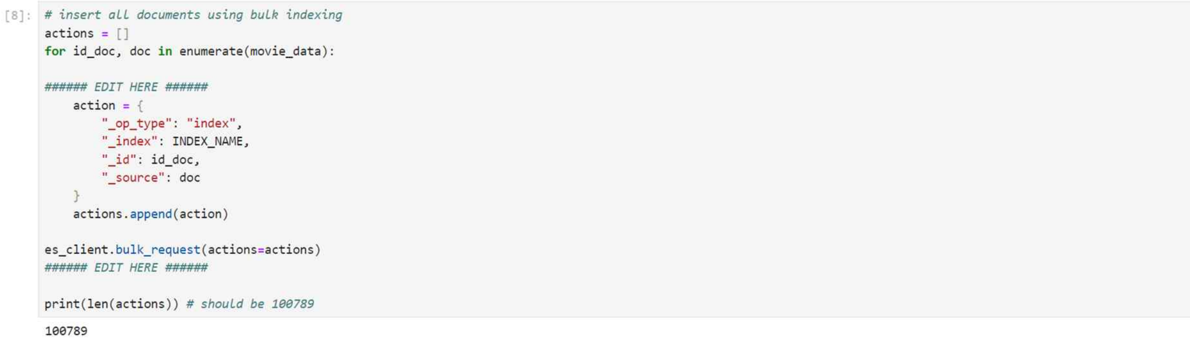
```
[6]: # insert only one document with doc_id == 0
     doc_id = 0

     ###### EDIT HERE ######
     for idx, document in enumerate(movie_data):
         if idx == doc_id:
             es_client.insert_one_document(index_name=INDEX_NAME, body=document, doc_id=doc_id)
             break
     ###### EDIT HERE ######
```

[Delete one document]



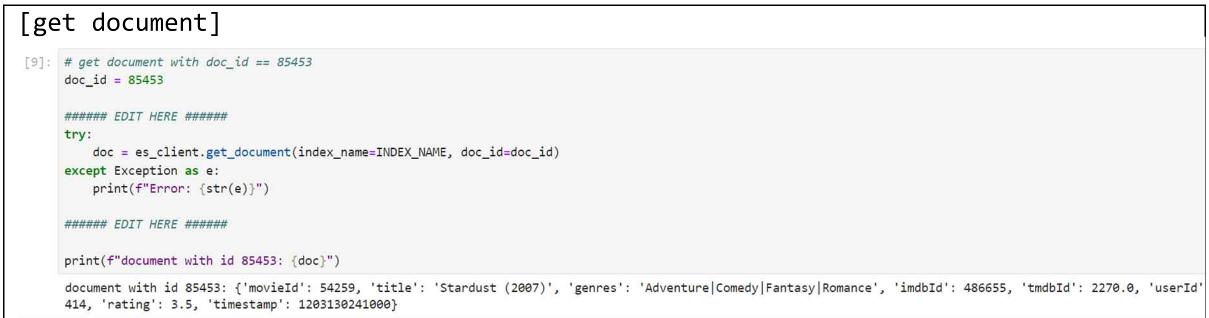
```
[7]: # delete document with doc_id == 0
     doc_id = 0

     ###### EDIT HERE ######
     es_client.delete_document(index_name=INDEX_NAME, doc_id=doc_id)
     ###### EDIT HERE ######
```

[insert all document using bulk]



```
[8]: # insert all documents using bulk indexing
     actions = []
     for id_doc, doc in enumerate(movie_data):

     ###### EDIT HERE ######
         action = {
             "_op_type": "index",
             "_index": INDEX_NAME,
             "_id": id_doc,
             "_source": doc
         }
         actions.append(action)

     es_client.bulk_request(actions=actions)
     ###### EDIT HERE ######

     print(len(actions)) # should be 100789

     100789
```

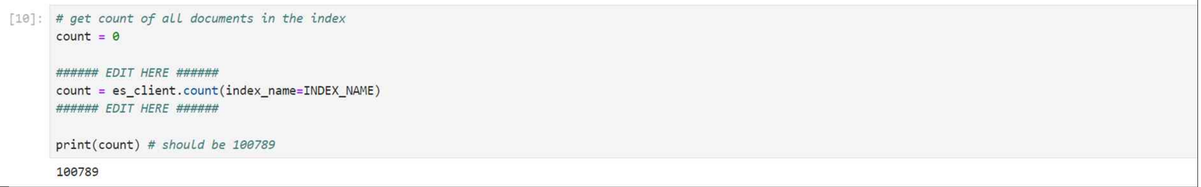
[get document]



```
[9]: # get document with doc_id == 85453
     doc_id = 85453

     ###### EDIT HERE ######
     try:
         doc = es_client.get_document(index_name=INDEX_NAME, doc_id=doc_id)
     except Exception as e:
         print(f"Error: {str(e)}")

     ###### EDIT HERE ######

     print(f"document with id 85453: {doc}")
```

```
document with id 85453: {'movieId': 54259, 'title': 'Stardust (2007)', 'genres': 'Adventure|Comedy|Fantasy|Romance', 'imdbId': 486655, 'tmdbId': 2270.0, 'userId'
414, 'rating': 3.5, 'timestamp': 1203130241000}
```

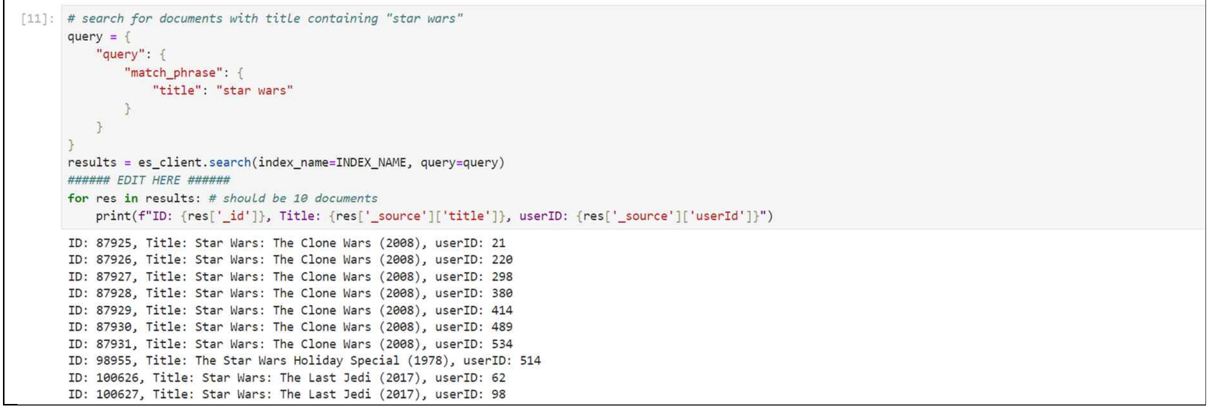
[get count of all documents in the index]



```
[10]: # get count of all documents in the index
      count = 0

      ###### EDIT HERE ######
      count = es_client.count(index_name=INDEX_NAME)
      ###### EDIT HERE ######

      print(count) # should be 100789
```

```
100789
```

[search document]



```
[11]: # search for documents with title containing "star wars"
      query = {
          "query": {
              "match_phrase": {
                  "title": "star wars"
              }
          }
      }
      results = es_client.search(index_name=INDEX_NAME, query=query)
      ###### EDIT HERE ######
      for res in results: # should be 10 documents
          print(f"ID: {res['_id']}, Title: {res['_source']['title']}, userID: {res['_source']['userId']}")
```

```
ID: 87925, Title: Star Wars: The Clone Wars (2008), userID: 21
ID: 87926, Title: Star Wars: The Clone Wars (2008), userID: 220
ID: 87927, Title: Star Wars: The Clone Wars (2008), userID: 298
ID: 87928, Title: Star Wars: The Clone Wars (2008), userID: 380
ID: 87929, Title: Star Wars: The Clone Wars (2008), userID: 414
ID: 87930, Title: Star Wars: The Clone Wars (2008), userID: 489
ID: 87931, Title: Star Wars: The Clone Wars (2008), userID: 534
ID: 98955, Title: The Star Wars Holiday Special (1978), userID: 514
ID: 100626, Title: Star Wars: The Last Jedi (2017), userID: 62
ID: 100627, Title: Star Wars: The Last Jedi (2017), userID: 98
```

2. **[10pts]** Compare the difference in execution time between basic and helper operatio ns and report the result and your findings.

Answer: Enter your **code and result** here. You must show your result, either an image or text.

A. **[5pts]** Compare the difference in execution time between performing 'bulk' and 'insert_one_document' multiple times and explain the reason.

[Execution time report]

method	w/o bulk	w/ bulk
Time consumed	1418.2433474063873 seconds	23.01433706283569 seconds

[explanation]
Using bulk API was significantly faster than inserting data without bulk. This is because of reduced overhead when inserting documents by sending multiple documents in a single request, rather than sending one by one without using bulk.

B. **[5pts]** Compare the difference in execution time between performing 'scan' and 'search' multiple times and explain the reason.

[Execution time report: search and scroll]

Query	Query 1	Query 2
count	100789	6781
Time consumed	6.6975953578949 seconds	6.07894802093506 seconds

[Execution time report: scan]

Query		Query 1	Query 2
count		87571(10pages), 100789(1000pages)	6781(10pages), 6781(1000pages)
Page size	10	17.140448570251465 seconds	1.296100616455078 seconds
	1000	0.8979315757751465 seconds	0.120013952255249 seconds

[explanation]

1. Compare between scan and search & scroll
   Scan() is faster than the other, because scan() is designed for efficient retrieval of large result sets. It is optimized for scrolling through large numbers of document efficiently. Using scan() makes Elasticsearch prefetch the next batch of result to make faster work.
2. Compare between page size in scan()
   If page size is larger, it is both faster and memory-intensive. A large page size lead to fast retrieval of documents because more documents are fetched in each batch. Also, large page size require more memory to store the retrieved document in memory before they are returned to the client.