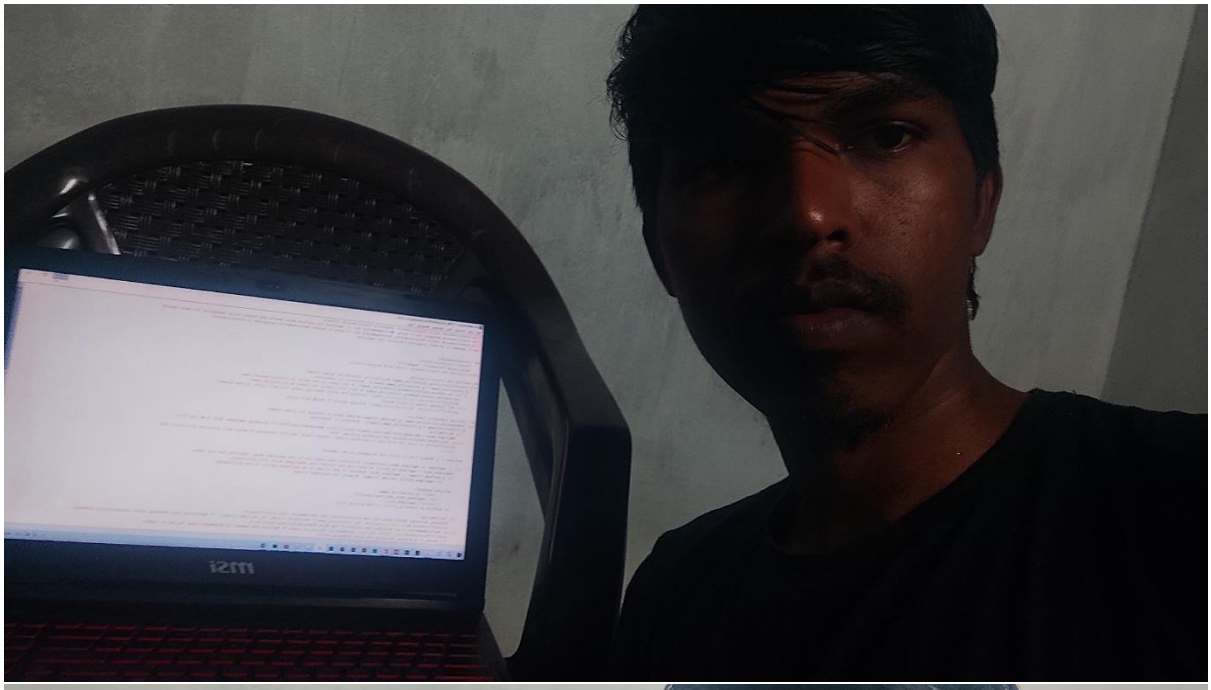
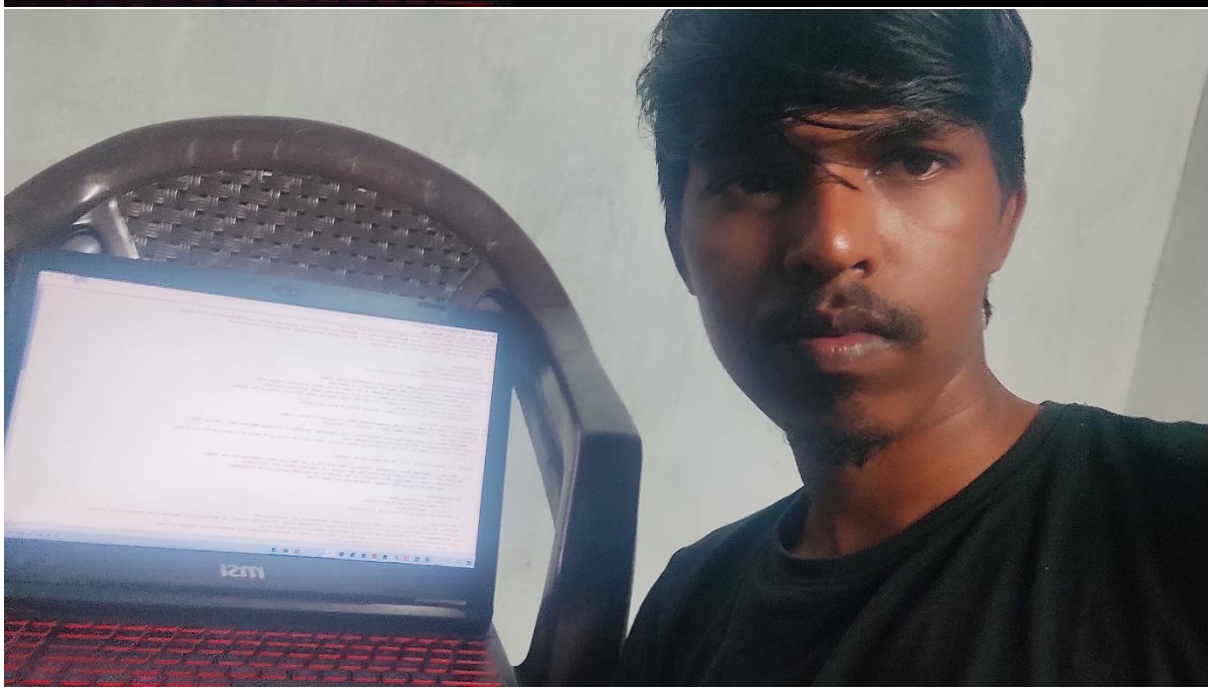
Program Round – 2

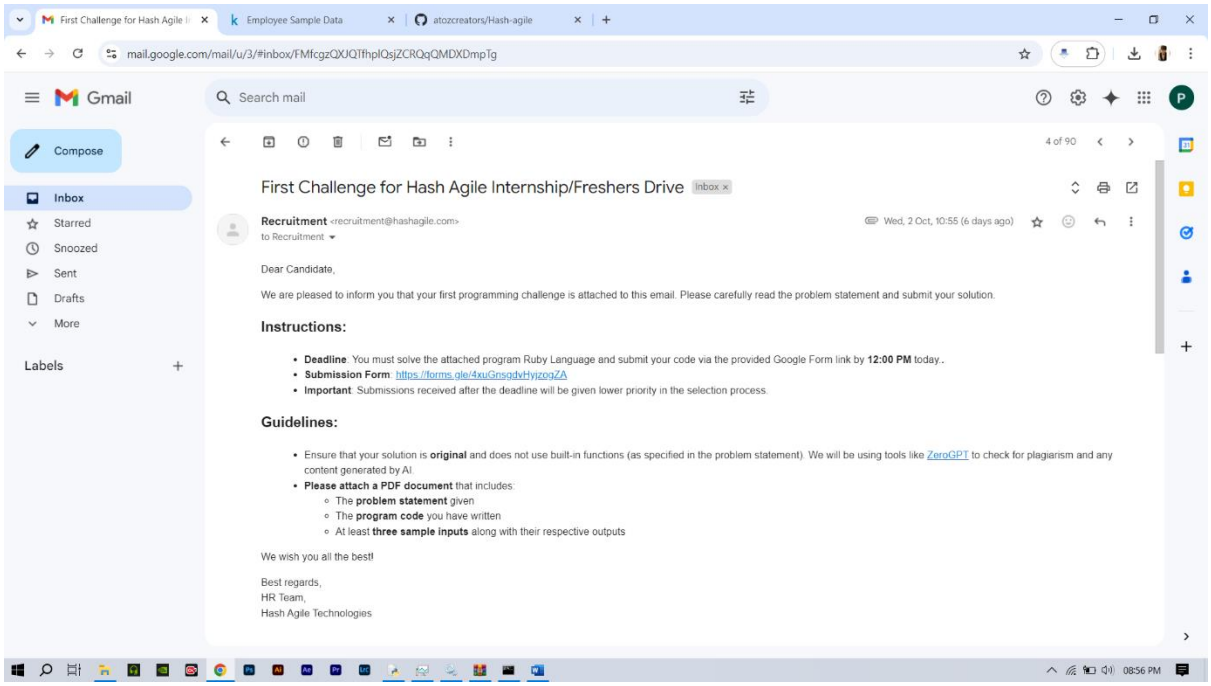
Name: PRAVEENKUMAR P

1. **Selfie Pic:**





2. 1st Task Email Screenshot:



First Challenge for Hash Agile Internship/Freshers Drive

Recruitment <recruitment@hashagile.com>
to Recruitment

Wed, 2 Oct, 10:55 (6 days ago)

Dear Candidate,

We are pleased to inform you that your first programming challenge is attached to this email. Please carefully read the problem statement and submit your solution.

Instructions:

- **Deadline**: You must solve the attached program Ruby Language and submit your code via the provided Google Form link by **12:00 PM** today..
- **Submission Form**: https://forms.gle/4xuGnsqdvHyjzcqZA
- **Important**: Submissions received after the deadline will be given lower priority in the selection process.

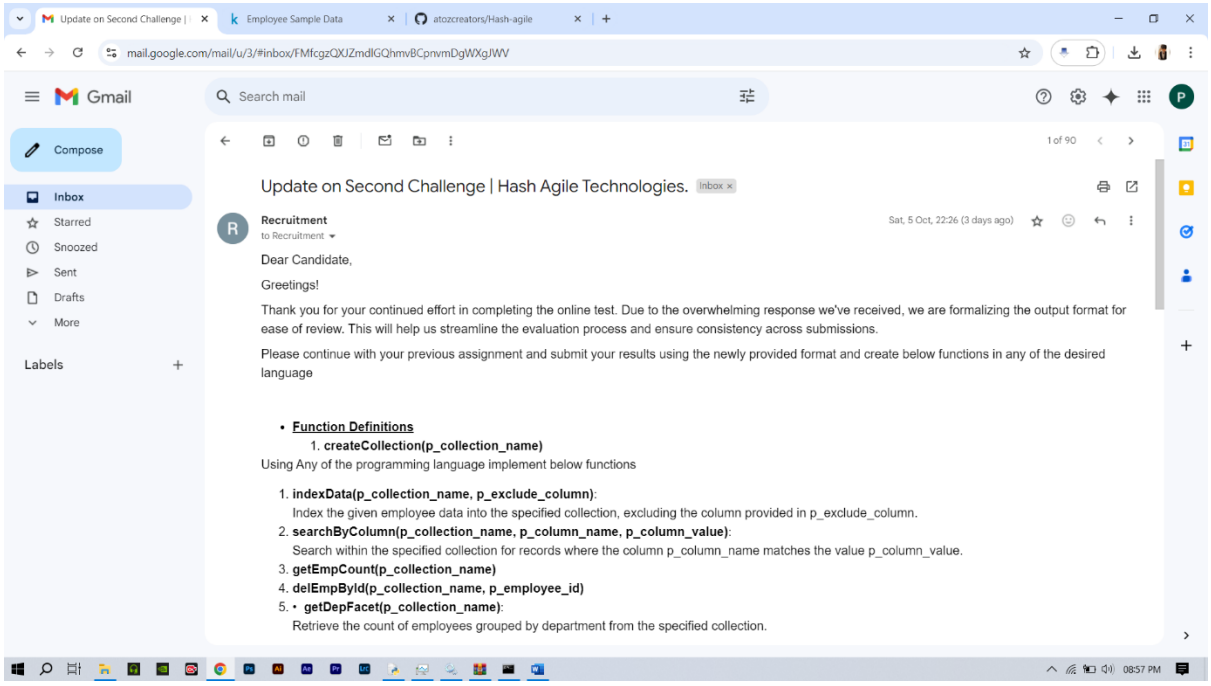
Guidelines:

- Ensure that your solution is **original** and does not use built-in functions (as specified in the problem statement). We will be using tools like ZeroGPT to check for plagiarism and any content generated by AI.
- **Please attach a PDF document** that includes:
  - The **problem statement** given
  - The **program code** you have written
  - At least **three sample inputs** along with their respective outputs

We wish you all the best!

Best regards,
HR Team,
Hash Agile Technologies

3. 2nd Task Email:



Update on Second Challenge | Hash Agile Technologies.

Recruitment
to Recruitment

Sat, 5 Oct, 22:26 (3 days ago)

Dear Candidate,

Greetings!

Thank you for your continued effort in completing the online test. Due to the overwhelming response we've received, we are formalizing the output format for ease of review. This will help us streamline the evaluation process and ensure consistency across submissions.

Please continue with your previous assignment and submit your results using the newly provided format and create below functions in any of the desired language

- **Function Definitions**
  1. **createCollection(p_collection_name)**

Using Any of the programming language implement below functions

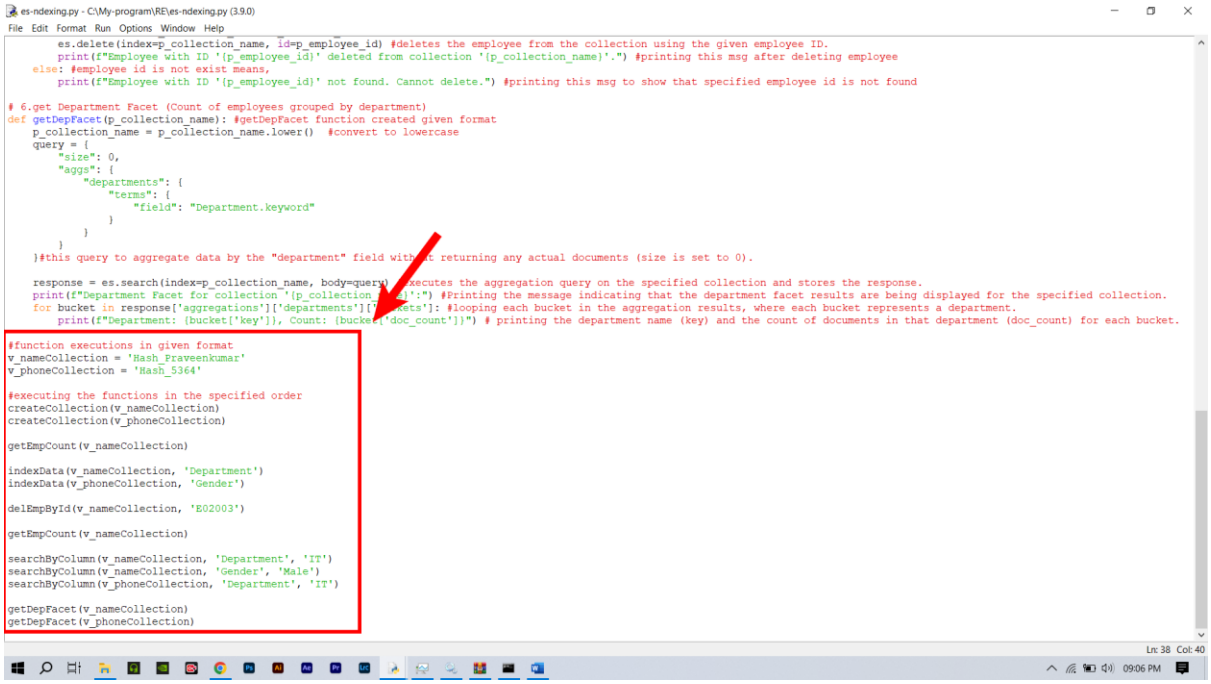
1. **indexData(p_collection_name, p_exclude_column)**:
   Index the given employee data into the specified collection, excluding the column provided in p_exclude_column.
2. **searchByColumn(p_collection_name, p_column_name, p_column_value)**:
   Search within the specified collection for records where the column p_column_name matches the value p_column_value.
3. **getEmpCount(p_collection_name)**
4. **delEmpById(p_collection_name, p_employee_id)**
5. • **getDepFacet(p_collection_name)**:
   Retrieve the count of employees grouped by department from the specified collection.

4. **Github URL For Round 1: https://github.com/atozcreators/Hash-agile/blob/main/Problem-Answer-Round-1.pdf**

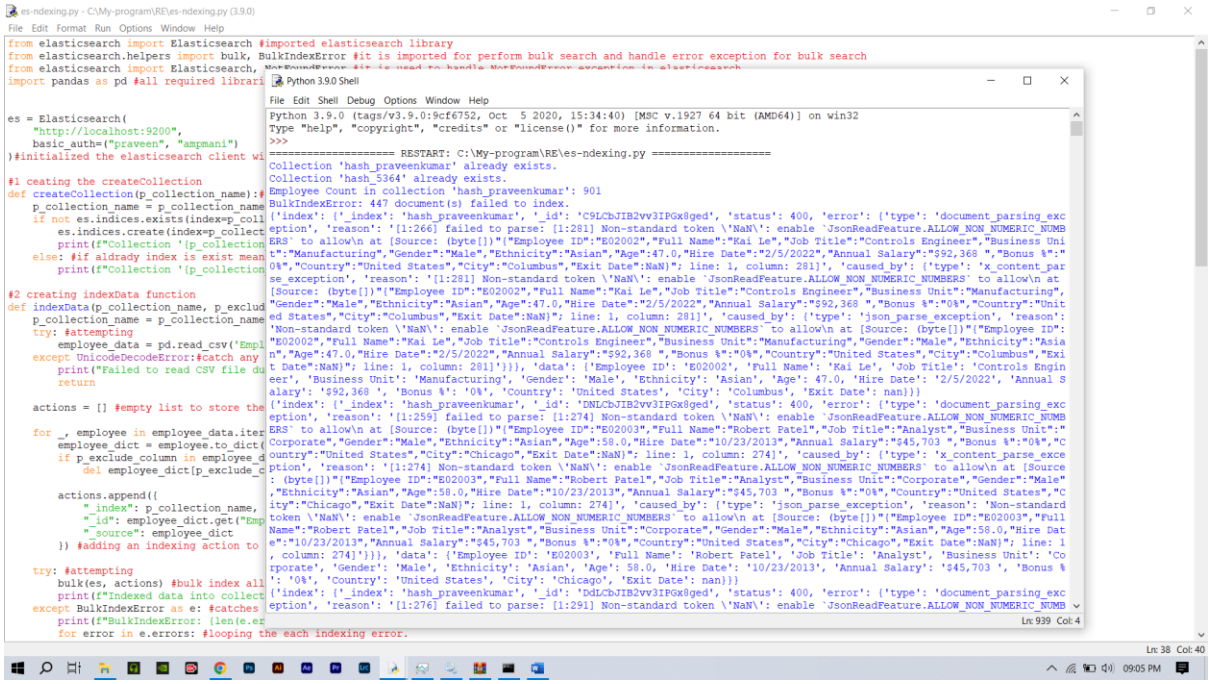
5. **Github URL For Assesment: https://github.com/atozcreators/Hash-agile**

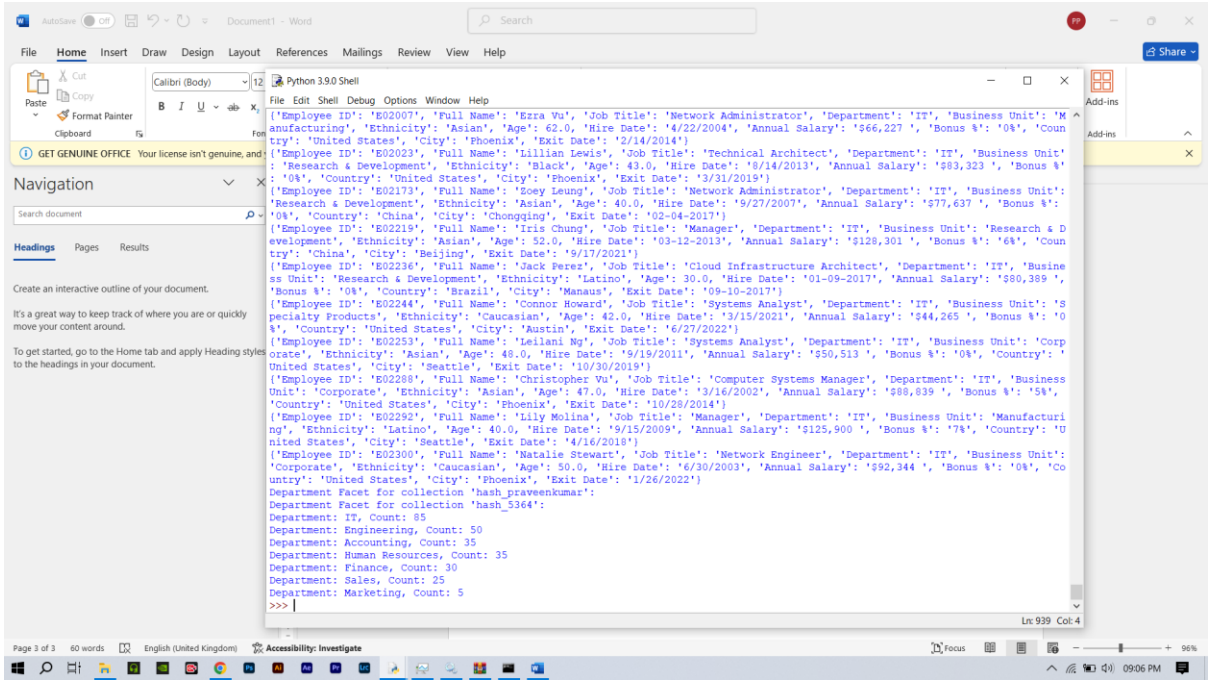
6. **Function Execution Results:**

Employee csv file is have huge data so, I only attach 1st and last screenshot only. I provide my entire code(with commands) after the screenshot.



I call all the functions at single time of code execution





Here is my full code: (Python):

from elasticsearch import Elasticsearch #imported elasticsearch library

from elasticsearch.helpers import bulk, BulkIndexError #it is imported for perform bulk search and handle error exception for bulk search

from elasticsearch import Elasticsearch, NotFoundError #it is used to handle NotFoundError exception in elasticsearch

```
import pandas as pd #all required libraries are imported



es = Elasticsearch(
    "http://localhost:9200",
    basic_auth=("praveen", "ampmani")
)#initialized the elasticsearch client with authentication


#1 ceating the createCollection
def createCollection(p_collection_name):#collection created for given format
    p_collection_name = p_collection_name.lower()  #convert to lowercase
    if not es.indices.exists(index=p_collection_name): # the index is not exist in elastic search then
        es.indices.create(index=p_collection_name) # this will create the index (p_collection_name)
        print(f"Collection '{p_collection_name}' created.")# after creating index this will print in the screen
    else: #if aldrady index is exist means,
        print(f"Collection '{p_collection_name}' already exists.") #this will print


#2 creating indexData function
def indexData(p_collection_name, p_exclude_column):#index data is created for given format
    p_collection_name = p_collection_name.lower()  #convert to lowercase
    try: #attempting
        employee_data = pd.read_csv('Employee Sample Data 1.csv', encoding='ISO-8859-1') #loading employee data from csv file
    except UnicodeDecodeError:#catch any encoding problem, then
        print("Failed to read CSV file due to encoding issues. Please check the file encoding.") #this will printing this error msg
        return
```

```
    actions = [] #empty list to store the documents to be indexed


    for _, employee in employee_data.iterrows(): #iterates over each row of the employee data, ignoring the row index.
        employee_dict = employee.to_dict() #converting the current row (employee data) into dictionary.
        if p_exclude_column in employee_dict: #checks if the column to be excluded exists in the dictionary.
            del employee_dict[p_exclude_column]  #remove the excluded column


        actions.append({
            "_index": p_collection_name,
            "_id": employee_dict.get("EmployeeID"),
            "_source": employee_dict
        }) #adding an indexing action to the actions list


    try: #attempting
        bulk(es, actions) #bulk index all the collected actions (documents) into elasticsearch.
        print(f"Indexed data into collection '{p_collection_name}' excluding column '{p_exclude_column}'.") #printing this message after successfully indexed
    except BulkIndexError as e: #catches any errors if the bulk indexing process fails as e.
        print(f"BulkIndexError: {len(e.errors)} document(s) failed to index.") #prints the number of documents that failed to index.
        for error in e.errors: #looping the each indexing error.
            print(error)  #print specific details about each error


def verifyEmpExists(p_collection_name, p_employee_id): #this function verifying the employee is exist
    try: #attempting
        es.get(index=p_collection_name, id=p_employee_id) #retriving the employee document using employee id
        print(f"Employee with ID '{p_employee_id}' exists in collection '{p_collection_name}'.") #if employee id exist then this msg will print
```

```
        return True #returning true(employye is exist)
    except NotFoundError: #catch the error then,
        print(f"Employee with ID '{p_employee_id}' does not exist in collection '{p_collection_name}'.") #printing this mesg
        return False #returning false(employee not exist)
```

#3 search by column function

```
def searchByColumn(p_collection_name, p_column_name, p_column_value):#search by column function is created given format
    p_collection_name = p_collection_name.lower()  #convert to lowercase
    query = {
        "query": {
            "match": {
                p_column_name: p_column_value
            }
        }
    } #creates a search query that matches documents where the specified column equals the given value.
    response = es.search(index=p_collection_name, body=query) #executes the search query specified collection and stores it in response variable.
    print(f"Search Results for {p_column_name} = '{p_column_value}':")#printing search is results.
    for hit in response['hits']['hits']: #looping each document in the search results.
        print(hit["_source"]) #printing the source data of each document found in the search results.
```

#4 get employee counf function

```
def getEmpCount(p_collection_name): #getEmpCount function is created given format
    p_collection_name = p_collection_name.lower()  #convert to lowercase
    count = es.count(index=p_collection_name)['count'] #retrieves the total count of documents in the specified collection and stores it in the count variable.
```

```
    print(f"Employee Count in collection '{p_collection_name}': {count}") #printing
employee count in the collection


#5 deleting employee using employee ID

def delEmpById(p_collection_name, p_employee_id):#delEmpById function is created given
format

    p_collection_name = p_collection_name.lower() #convert to lowercase

    if verifyEmpExists(p_collection_name, p_employee_id): #checking the employee ID exists
in the specified collection by calling the verifyEmpExists function.the employee exists

        es.delete(index=p_collection_name, id=p_employee_id) #deletes the employee from the
collection using the given employee ID.

        print(f"Employee with ID '{p_employee_id}' deleted from collection
'{p_collection_name}'.") #printing this msg after deleting employee

    else: #employee id is not exist means,

        print(f"Employee with ID '{p_employee_id}' not found. Cannot delete.") #printing this
msg to show that specified employee id is not found


# 6.get Department Facet (Count of employees grouped by department)

def getDepFacet(p_collection_name): #getDepFacet function created given format

    p_collection_name = p_collection_name.lower()  #convert to lowercase

    query = {

        "size": 0,

        "aggs": {

            "departments": {

                "terms": {

                    "field": "Department.keyword"

                }

            }

        }

    }#this query to aggregate data by the "department" field without returning any actual
documents (size is set to 0).
```

response = es.search(index=p_collection_name, body=query) #executes the aggregation query on the specified collection and stores the response.

print(f"Department Facet for collection '{p_collection_name}':") #Printing the message indicating that the department facet results are being displayed for the specified collection.

for bucket in response['aggregations']['departments']['buckets']: #looping each bucket in the aggregation results, where each bucket represents a department.

print(f"Department: {bucket['key']}, Count: {bucket['doc_count']}") # printing the department name (key) and the count of documents in that department (doc_count) for each bucket.

```
#function executions in given format
v_nameCollection = 'Hash_Praveenkumar'
v_phoneCollection = 'Hash_5364'

#executing the functions in the specified order
createCollection(v_nameCollection)
createCollection(v_phoneCollection)

getEmpCount(v_nameCollection)

indexData(v_nameCollection, 'Department')
indexData(v_phoneCollection, 'Gender')

delEmpById(v_nameCollection, 'E02003')

getEmpCount(v_nameCollection)

searchByColumn(v_nameCollection, 'Department', 'IT')
searchByColumn(v_nameCollection, 'Gender', 'Male')
searchByColumn(v_phoneCollection, 'Department', 'IT')

getDepFacet(v_nameCollection)
```

getDepFacet(v_phoneCollection)