Interview Assignment Task

Mervin Renie

First Task Email :

First Challenge for Hash Agile Internship/Freshers Drive Inbox ×

Recruitment <recruitment@hashagile.com> 12:12 (9 hours ago)
to Recruitment

Dear Candidate,

Greetings !!

Due to overwhelming responses, we are staring the interview process by 11:55 AM and provided time will be from 12:00-2:00 PM.

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.

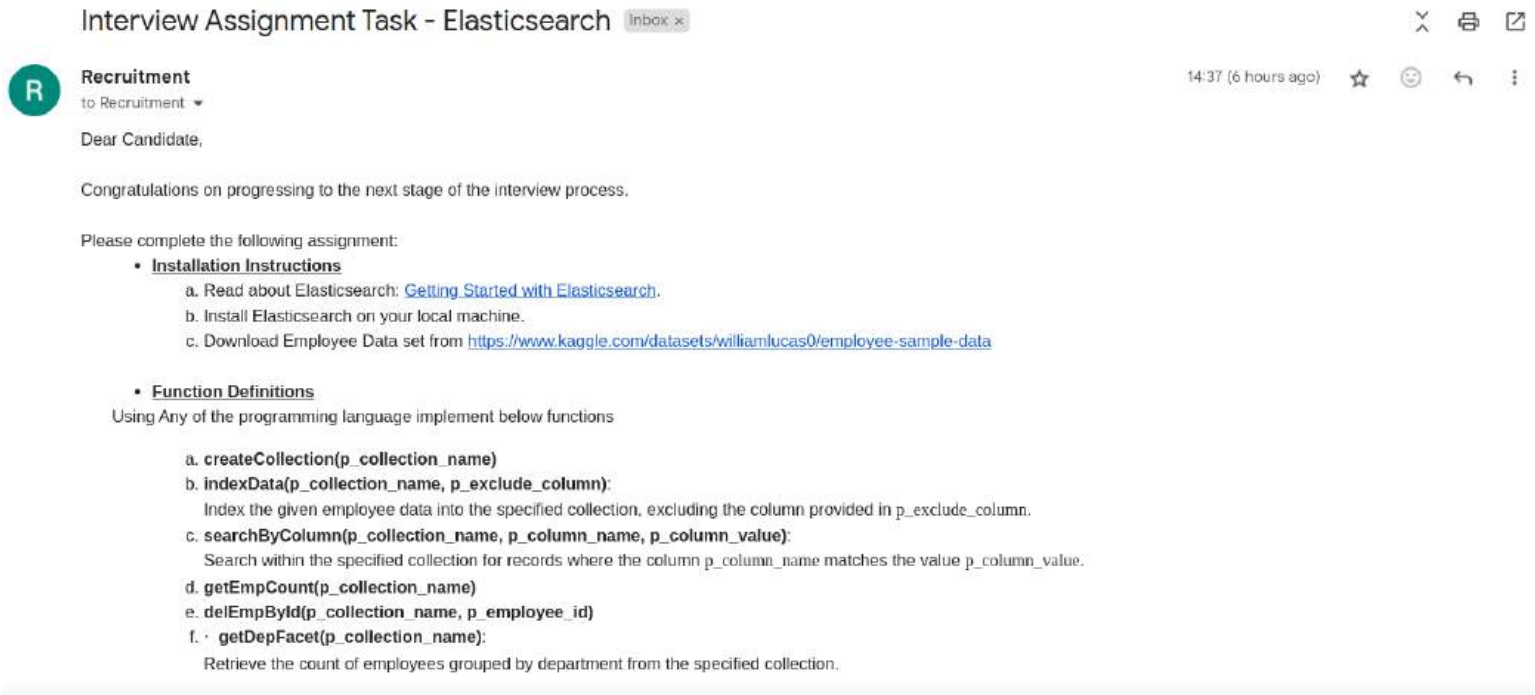
Instructions:

- **Deadline**: You must solve the attached program Type Script Language and submit your code via the provided Google Form link by 0**2:00 PM** today.**.**
- **Submission Form**: https://forms.gle/4xuGnsgdvHyjzogZA
- **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

Second Task Email :



Interview Assignment Task - Elasticsearch Inbox ×

Recruitment
to Recruitment

14:37 (6 hours ago)

Dear Candidate,

Congratulations on progressing to the next stage of the interview process.

Please complete the following assignment:

- **Installation Instructions**
  a. Read about Elasticsearch: Getting Started with Elasticsearch.
  b. Install Elasticsearch on your local machine.
  c. Download Employee Data set from https://www.kaggle.com/datasets/williamlucas0/employee-sample-data

- **Function Definitions**

Using Any of the programming language implement below functions

a. **createCollection(p_collection_name)**
b. **indexData(p_collection_name, p_exclude_column)**:
   Index the given employee data into the specified collection, excluding the column provided in p_exclude_column.
c. **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.
d. **getEmpCount(p_collection_name)**
e. **delEmpById(p_collection_name, p_employee_id)**
f. · **getDepFacet(p_collection_name)**:
   Retrieve the count of employees grouped by department from the specified collection.

Github :

Round -1 : hash-github-round-1

Assignment : hash-github-assignment

Procedure :

- Create a client class from @elastic/elasticsearch package which is used to interact with elasticsearch cluster/nodes, allowing to perform operations like creating indices, indexing documents, data searching, etc.

hash_tech > Round_2 > JS elasticSearch.js > ...

```
1  const { Client } = require('@elastic/elasticsearch');
2
3  const client = new Client({ node: 'http://localhost:9200' });
4
5  module.exports = client;
6
```

- The createCollection function is designed to create a index in elasticsearch and also handles naming conventions to prevent invalid naming errors.

hash_tech > Round_2 > JS collection.js > createCollection

```
1  const client = require('./elasticSearch');
2
3  // Function to create a collection
4  async function createCollection(collectionName) {
5    const lowerCaseCollectionName = collectionName.toLowerCase();
6    try {
7      await client.indices.create({
8        index: lowerCaseCollectionName,
9      });
10     console.log(`Index created: ${lowerCaseCollectionName}`);
11   } catch (error) {
12     console.error('Error creating collection:', error);
13   }
14 }
15
```

- **createCollection(p_collection_name)**

hash_tech > Round_2 > JS app.js > ...

```
const { createCollection} = require('./collection');

let v_nameCollection = 'Hash_Mervin_renie';
let v_phoneCollection = 'Hash-8377';

// Execution
async function execute(){

  await createCollection(v_nameCollection);
  await createCollection(v_phoneCollection);
}
execute();

```

```
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$ node app.js
Index created: hash_mervin_renie
Index created: hash-8377
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$
```

- For verification, before data indexing total count will be empty.

hash_tech > Round_2 > JS searchOperations.js > getEmpCount

```
1
2   const client = require('./elasticSearch');
3
4   // Function to get employee count
5   async function getEmpCount(collectionName) {
6     const lowerCaseCollectionName = collectionName.toLowerCase();
7     try {
8       const response = await client.count({
9         index: lowerCaseCollectionName,
10      });
11      console.log(`Total employees: ${response.count}`);
12    } catch (error) {
13      console.error('Error getting employee count:', error);
14    }
15  }
16
```

```
const { createCollection} = require('./collection');
const {getEmpCount} = require('./searchOperations');

let v_nameCollection = 'Hash_Mervin_renie';
let v_phoneCollection = 'Hash-8377';

// Execution
async function execute(){

  //await createCollection(v_nameCollection);
  //await createCollection(v_phoneCollection);

  await getEmpCount(v_nameCollection);
}
execute();
```

```
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$ node app.js
Total employees: 0
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$
```

- The function indexData reads employee data from a CSV file, processes the data by excluding a specified column and then indexes the processed data into an Elasticsearch collection.

hash_tech > Round_2 > JS dataIndexing.js > indexData

```
1   const fs = require('fs');
2   const csv = require('csv-parser');
3   const client = require('./elasticSearch');
4   const path = require('path');
5
6   // Use path.join to create the absolute path
7   const filePath = path.join(__dirname, 'Employee_data.csv');
8
9   // Function to index data, excluding a specific column
10  async function indexData(collectionName, excludeColumn) {
11    const lowerCaseCollectionName = collectionName.toLowerCase();
12    const results = [];
13
14    fs.createReadStream(filePath)
15      .pipe(csv())
16      .on('data', (row) => {
17        delete row[excludeColumn];
18        results.push(row);
19      })
20      .on('end', async () => {
21        for (let employee of results) {
22          await client.index({
23            index: lowerCaseCollectionName,
24            body: employee,
25          });
26        }
27        console.log('Data indexed successfully');
28      });
29  }
```

- **indexData(p_collection_name, p_exclude_column)**

```
const { createCollection} = require('./collection');
const { indexData } = require('./dataIndexing');
const {getEmpCount} = require('./searchOperations');

let v_nameCollection = 'Hash_Mervin_renie';
let v_phoneCollection = 'Hash-8377';

// Execution
async function execute(){

  //await createCollection(v_nameCollection);
  //await createCollection(v_phoneCollection);

  await getEmpCount(v_nameCollection);

  await indexData(v_nameCollection, 'Department');
  await indexData(v_phoneCollection, 'Gender');
}
execute();
```

```
Data indexed successfully
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$
```

➢ The function deleteEmployeeId will match the required employee id with Elasticsearch collection and returns a value > 0 to delete the document.

hash_tech > Round_2 > JS collection.js > delEmpByEmployeeId

```
18  async function delEmpByEmployeeId(collectionName, employeeId) {
21      // First, search for the document by Employee ID
22      const searchResponse = await client.search({
23        index: lowerCaseCollectionName,
24        body: {
25          query: {
26            match: {
27              'Employee ID': employeeId,
28            },
29          },
30        },
31      });
32
33      console.log(searchResponse.hits.hits);
34
35      // Check if any documents were found
36      if (searchResponse.hits.total.value > 0) {
37        const docId = searchResponse.hits.hits[0]._id; // Get the document ID
38
39        // Now delete the document using the document ID
40        await client.delete({
41          index: lowerCaseCollectionName,
42          id: docId,
43        });
44        console.log(`Employee with ID ${employeeId} deleted successfully`);
45      } else {
46        console.log(`Employee with ID ${employeeId} not found`);
47      }
48    } catch (error) {
```

- **delEmpById(p_collection_name, p_employee_id)**

```
// Execution
async function execute(){

  //await createCollection(v_nameCollection);
  //await createCollection(v_phoneCollection);

  //await getEmpCount(v_nameCollection);

//  await indexData(v_nameCollection, 'Department');
//  await indexData(v_phoneCollection, 'Gender');

  await delEmpByEmployeeId(v_nameCollection, 'E02003');

}
execute();
```

```
  },
  {
    _index: 'hash_mervin_renie',
    _id: '6OXgUpIBIVrNWX48cnPu',
    _score: 5.985494,
    _source: {
      'Employee ID': 'E02003',
      'Full Name': 'Robert Patel',
      'Job Title': 'Analyst',
      'Business Unit': 'Corporate',
      Gender: 'Male',
      Ethnicity: 'Asian',
      Age: '58',
      'Hire Date': '10/23/2013',
      'Annual Salary': '$45,703 ',
      'Bonus %': '0%',
      Country: 'United States',
      City: 'Chicago',
      'Exit Date': ''
    }
  }
]
Employee with ID E02003 deleted successfully
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$
```

➢ Check total count of employees in Elasticsearch collection.

hash_tech > Round_2 > JS searchOperations.js > getEmpCount

```
1
2   const client = require('./elasticSearch');
3
4   // Function to get employee count
5   async function getEmpCount(collectionName) {
6     const lowerCaseCollectionName = collectionName.toLowerCase();
7     try {
8       const response = await client.count({
9         index: lowerCaseCollectionName,
10      });
11      console.log(`Total employees: ${response.count}`);
12    } catch (error) {
13      console.error('Error getting employee count:', error);
14    }
15  }
16
```

- **getEmpCount(p_collection_name)**

```
// Execution
async function execute(){

  //await createCollection(v_nameCollection);
  //await createCollection(v_phoneCollection);

  //await getEmpCount(v_nameCollection);

//  await indexData(v_nameCollection, 'Department');
//  await indexData(v_phoneCollection, 'Gender');

  //await delEmpByEmployeeId(v_nameCollection, 'E02003');
  await getEmpCount(v_nameCollection);
}
execute();
```

```
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$ node app.
Total employees: 4230
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$
```

- The searchByColumn search within the specified collection of records where column matches the value. It returns id, name, job, salary for a neat presentation rather than exposing all details.

```
18  // Function to search by a specific column value
19  async function searchByColumn(collectionName, columnName, columnValue) {
20    const lowerCaseCollectionName = collectionName.toLowerCase();
21    try {
22      const response = await client.search({
23        index: lowerCaseCollectionName,
24        body: {
25          query: {
26            match: {
27              [columnName]: columnValue,
28            },
29
30          },
31        },
32      });
33      //For neat appearence of all data matches.
34      const results = response.hits.hits.map(hit => ({
35        employeeid: hit._source['Employee ID'],
36        fullname: hit._source['Full Name'],
37        job:hit._source['Job Title'],
38        salary:hit._source['Annual Salary']
39      }));
40      console.log(results);
41    }
42    catch (error) {
43      console.error('Error searching data:', error);
44    }
45  }
46
```

- **searchByColumn(p_collection_name, p_column_name, p_column_value)**

```
async function execute(){
  //await createCollection(v_nameCollection);
  //await createCollection(v_phoneCollection);

  //await getEmpCount(v_nameCollection);

  //await indexData(v_nameCollection, 'Department');
  //await indexData(v_phoneCollection, 'Gender');

  //await delEmpByEmployeeId(v_nameCollection, 'E02003');
  //await getEmpCount(v_nameCollection);

  await searchByColumn(v_nameCollection, 'Department', 'IT');

}
execute();
```

```
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$ node app.js
[]
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$
```

```
async function execute(){
  //await createCollection(v_nameCollection);
  //await createCollection(v_phoneCollection);

  //await getEmpCount(v_nameCollection);

  //await indexData(v_nameCollection, 'Department');
  //await indexData(v_phoneCollection, 'Gender');

  //await delEmpByEmployeeId(v_nameCollection, 'E02003');
  //await getEmpCount(v_nameCollection);

  //await searchByColumn(v_nameCollection, 'Department', 'IT');
  await searchByColumn(v_nameCollection, 'Gender', 'Male');

}
execute();
```


```
{
  employeeid: 'E02701',
  fullname: 'Miles Ross',
  job: 'IT Coordinator',
  salary: '$50,022 '
},
{
  employeeid: 'E02702',
  fullname: 'Jonathan Santos',
  job: 'Vice President',
  salary: '$204,534 '
},
{
  employeeid: 'E02703',
  fullname: 'Joshua Maldonado',
  job: 'Sr. Account Representative',
  salary: '$75,814 '
},
{
  employeeid: 'E02704',
  fullname: 'Santiago f Vo',
  job: 'Director',
  salary: '$169,487 '
},
```

```
async function execute(){
  //await createCollection(v_nameCollection);
  //await createCollection(v_phoneCollection);

  //await getEmpCount(v_nameCollection);

  //await indexData(v_nameCollection, 'Department');
  //await indexData(v_phoneCollection, 'Gender');

  //await delEmpByEmployeeId(v_nameCollection, 'E02003');
  //await getEmpCount(v_nameCollection);

  //await searchByColumn(v_nameCollection, 'Department', 'IT');
  //await searchByColumn(v_nameCollection, 'Gender', 'Male');
  await searchByColumn(v_phoneCollection, 'Department', 'IT');
}
execute();
```


```
},
{
  employeeid: 'E02012',
  fullname: 'Jameson Pena',
  job: 'Systems Analyst',
  salary: '$40,499 '
},
{
  employeeid: 'E02014',
  fullname: 'Jose Wong',
  job: 'Director',
  salary: '$150,558 '
},
{
  employeeid: 'E02017',
  fullname: 'Luna Lu',
  job: 'IT Systems Architect',
  salary: '$64,208 '
},
{
  employeeid: 'E02020',
  fullname: 'Jordan Kumar',
  job: 'Service Desk Analyst',
  salary: '$95,729 '
},
```

➢ The function getDepFacet returns the count of employees grouped by department from the specified collection.

hash_tech > Round_2 > JS searchOperations.js > getDepFacet

```
47
48
49  // Function to get facet of departments
50  async function getDepFacet(collectionName) {
51    const lowerCaseCollectionName = collectionName.toLowerCase();
52    try {
53      const response = await client.search({
54        index: lowerCaseCollectionName,
55        body: {
56          aggs: {
57            departments: {
58              terms: {
59                field: 'Department.keyword',
60              },
61            },
62          },
63          size: 0,
64        },
65      });
66      console.log(response.aggregations.departments.buckets);
67    } catch (error) {
68      console.error('Error getting department facet:', error);
69    }
70  }
71
72  module.exports = { searchByColumn, getEmpCount, getDepFacet };
73
```

- **getDepFacet(p_collection_name)**

```
async function execute(){
  //await createCollection(v_phoneCollection);

  //await getEmpCount(v_nameCollection);

  //await indexData(v_nameCollection, 'Department');
  //await indexData(v_phoneCollection, 'Gender');

  //await delEmpByEmployeeId(v_nameCollection, 'E02003');
  //await getEmpCount(v_nameCollection);

  //await searchByColumn(v_nameCollection, 'Department', 'IT');
  //await searchByColumn(v_nameCollection, 'Gender', 'Male');
  //await searchByColumn(v_phoneCollection, 'Department', 'IT');
  await getDepFacet(v_nameCollection);
  await getDepFacet(v_phoneCollection);
}
execute();
```

```
mervin@tux:/media/mervin/DATA/hash_tech/Round_2$ node app.js
[]
[
  { key: 'IT', doc_count: 1170 },
  { key: 'Sales', doc_count: 633 },
  { key: 'Engineering', doc_count: 537 },
  { key: 'Marketing', doc_count: 460 },
  { key: 'Accounting', doc_count: 445 },
  { key: 'Finance', doc_count: 439 },
  { key: 'Human Resources', doc_count: 416 },
  { key: '', doc_count: 141 }
]
```

❖ Elasticsearch Config (Optional) :

```
curl -X GET 'http://localhost:9200'
{
  "name" : "tux",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "VUrvc2ErRfi7jf9nmLkfvA",
  "version" : {
    "number" : "8.15.2",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "98adf7bf6bb69b66ab95b761c9e5aadb0bb059a3",
    "build_date" : "2024-09-19T10:06:03.564235954Z",
    "build_snapshot" : false,
    "lucene_version" : "9.11.1",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```