

Solution

Code (JavaScript)

```
const { Client } = require('@elastic/elasticsearch');
const fs = require('fs');
const csv = require('csv-parser');
const client = new Client({
  node: 'http://localhost:9200',
  auth: {
    username: 'elastic',
    password: 'RIOOO'
  }
});
const indexName = 'employee_data_new1';
async function createIndex() {
  // Delete the index if it already exists
  const exists = await client.indices.exists({ index: indexName });
  if (exists.body) {
    await client.indices.delete({ index: indexName });
    console.log(`Index ${indexName} deleted.`);
  }
  await client.indices.create({
    index: indexName,
    body: {
      settings: {
        number_of_shards: 3,
        number_of_replicas: 2
      },
      mappings: {
        properties: {
          EmployeeID: { type: 'integer' },
```

```

        FirstName: { type: 'text' },
        LastName: { type: 'text' },
        Gender: { type: 'keyword' },
        Age: { type: 'integer' },
        Salary: { type: 'float' },
        Department: { type: 'text' },
        Position: { type: 'text' },
        DateOfHire: { type: 'date', format: 'yyyy-MM-dd' }
    }
}
});

console.log(`Index ${indexName} created successfully.`);
}

async function loadDataToElasticsearch() {
    const bulkOperations = [];
    fs.createReadStream('C:\\Users\\Naren\\Downloads\\employee_data_new.csv')
        .pipe(csv())
        .on('data', (row) => {
            bulkOperations.push({
                index: { _index: indexName }
            });
            bulkOperations.push({
                EmployeeID: parseInt(row.EmployeeID),
                FirstName: row.FirstName,
                LastName: row.LastName,
                Gender: row.Gender,
                Age: parseInt(row.Age),
                Salary: parseFloat(row.Salary),
                Department: row.Department,

```

```

    Position: row.Position,
    DateOfHire: row.DateOfHire
  });
})
.on('end', async () => {
  console.log(`Indexing ${bulkOperations.length / 2} employee records...`);

  try {
    const bulkResponse = await client.bulk({ refresh: true, body: bulkOperations });
    if (bulkResponse && bulkResponse.errors) {
      console.error('Errors occurred during bulk indexing:', bulkResponse.errors);
    } else {
      console.log('All employee records indexed successfully.');
```

```
    }
```

```
  } catch (err) {
```

```
    console.error('Error during bulk indexing:', err);
```

```
  }
```

```
});
```

```
}
```

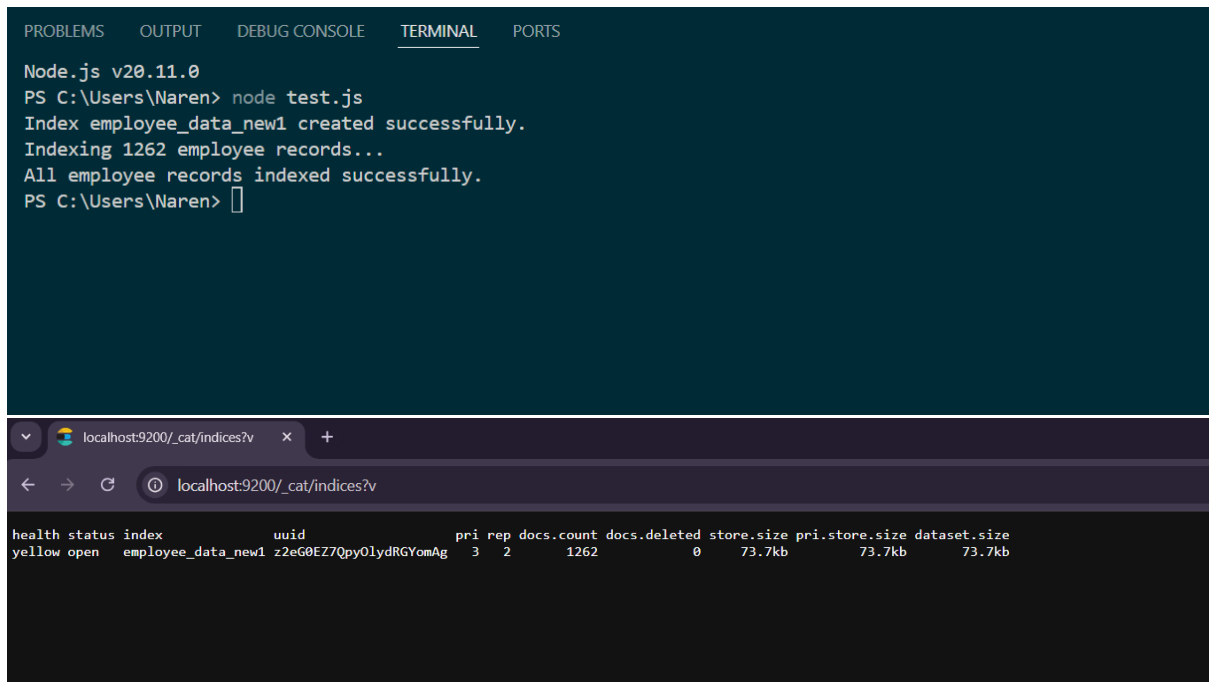
```
(async () => {
```

```
  await createIndex();
```

```
  await loadDataToElasticsearch();
```

```
})();
```

Answer Screenshot :



The screenshot displays a development environment with a terminal window and a web browser. The terminal window, titled 'Node.js v20.11.0', shows the execution of a Node.js script. The command 'node test.js' is run, resulting in the following output: 'Index employee_data_new1 created successfully.', 'Indexing 1262 employee records...', and 'All employee records indexed successfully.'. The prompt 'PS C:\Users\Naren>' is visible at the end of the output.

Below the terminal, a web browser window is open to the URL 'localhost:9200/_cat/indices?v'. The browser displays a table of Elasticsearch indices. The table has columns for health, status, index, uuid, pri, rep, docs.count, docs.deleted, store.size, pri.store.size, and dataset.size. The data row shows a 'yellow open' index named 'employee_data_new1' with a UUID of 'z2eG0EZ7QpyOlydRGYomAg', 3 primary shards, 2 replicas, 1262 documents, 0 deleted documents, and a store size of 73.7kb.

health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size	dataset.size
yellow	open	employee_data_new1	z2eG0EZ7QpyOlydRGYomAg	3	2	1262	0	73.7kb	73.7kb	73.7kb