

# ELK Task

Rana Hassan Hosny

GIZA SYSTEMS | [RANA.HOSNY@GIZASYSTEMS.ONMICROSOFT.COM](mailto:RANA.HOSNY@GIZASYSTEMS.ONMICROSOFT.COM)

# Elasticsearch Endpoints Documentation

## Query list:

1. Search for a specific class:

### Purpose:

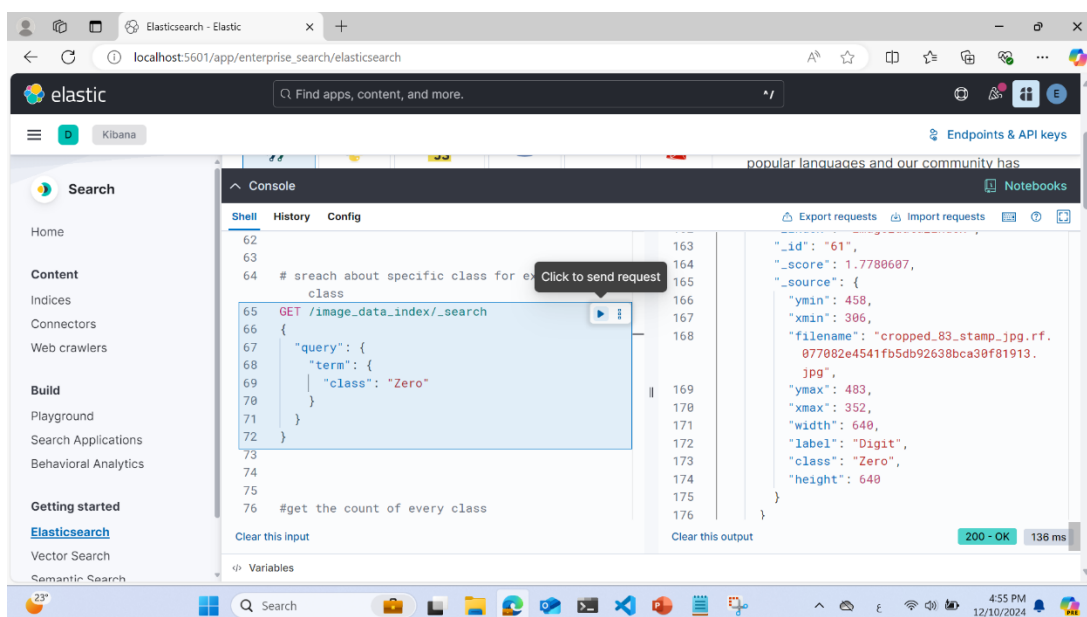
- This query retrieves documents belonging to a specific class, such as "Zero". It is useful for filtering and analyzing data specific to one class.

```
GET /image_data_index/_search
{
  "query": {
    "term": {
      "class": "Zero"
    }
  }
}
```

### Usefulness:

- Filters data by class.
- Help to understand the distribution and details of specific classes.

### Example:



The screenshot shows the Elasticsearch Kibana interface. In the left sidebar, the 'Search' section is active. The main console area displays a REST client with a query for the 'Zero' class. The query is as follows:

```
GET /image_data_index/_search
{
  "query": {
    "term": {
      "class": "Zero"
    }
  }
}
```

The console shows the response for this query, which includes a single document with the following fields:

```
{
  "_id": "61",
  "_score": 1.7780607,
  "_source": {
    "ymin": 458,
    "xmin": 306,
    "filename": "cropped_83_stamp.jpg.rf.077082e4541fb5db92638bca30f81913.jpg",
    "ymax": 483,
    "xmax": 352,
    "width": 640,
    "label": "Digit",
    "class": "Zero",
    "height": 640
  }
}
```

The status bar at the bottom of the console indicates a 200 OK response with a 138 ms execution time.

## 2. Get the count of every class:

### Purpose:

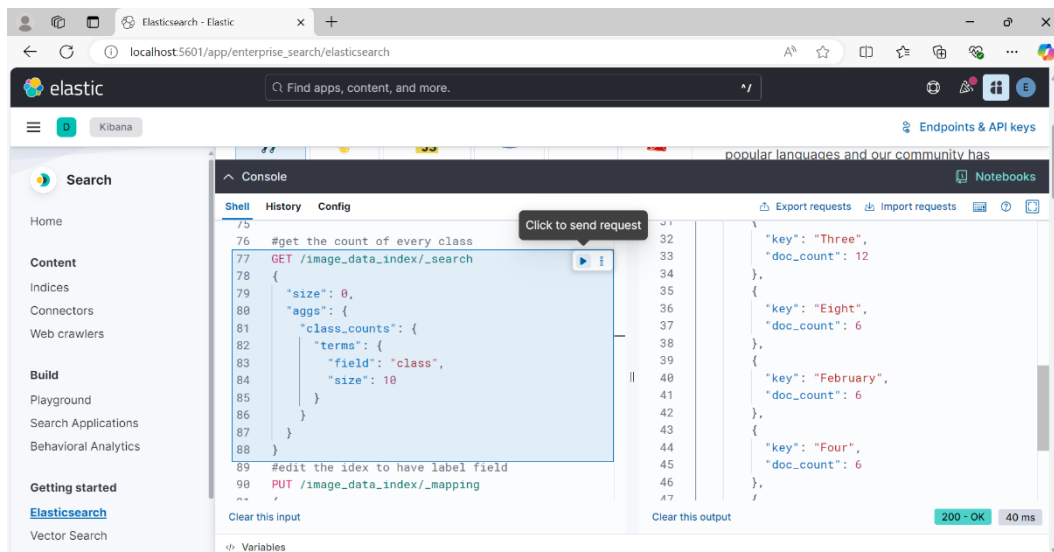
This query aggregates the number of documents for each class. It's helpful for visualizing the distribution of classes.

```
GET /image_data_index/_search
{
  "size": 0,
  "aggs": {
    "class_counts": {
      "terms": {
        "field": "class",
        "size": 10
      }
    }
  }
}
```

### Usefulness:

- Provides insights into data distribution.
- Helps in debugging and balancing datasets.

### Example:



The screenshot shows the Elasticsearch console interface. On the left, there's a sidebar with navigation options like Home, Content, Indices, Connectors, Web crawlers, Build, Playground, Search Applications, Behavioral Analytics, Getting started, Elasticsearch, Vector Search, and Semantic Search. The main area is divided into a 'Console' tab and a 'Config' tab. The 'Console' tab shows a list of requests. The first request is a GET request to `/image_data_index/_search` with a JSON body: `{ "size": 0, "aggs": { "class_counts": { "terms": { "field": "class", "size": 10 } } } }`. The response is a JSON array of objects, each containing a 'key' and a 'doc\_count'. The response is: `[ { "key": "Three", "doc_count": 12 }, { "key": "Eight", "doc_count": 6 }, { "key": "February", "doc_count": 6 }, { "key": "Four", "doc_count": 6 } ]`. The status is 200 - OK and the time taken is 40 ms.

3. Edit the index to add a `label` field:

### Purpose:

This updates the index mapping to include a new field `label`. It allows the addition of structured data for categorization.

```
PUT /image_data_index/_mapping
{
  "properties": {
    "label": {
      "type": "keyword"
    }
  }
}
```

### Usefulness:

- Prepares the index for additional categorization.
- Supports flexible data enrichment.

### Example:

The screenshot shows the Kibana interface for the `image_data_index`. The 'Mappings' tab is selected, displaying a list of fields and their types. The `label` field is highlighted with a blue background, indicating it is the selected field. The field type is `keyword`. Other fields include `class` (keyword), `coordinates` (object), `filename` (keyword), `height` (integer), `width` (integer), `xmax` (long), `xmin` (long), `ymax` (long), and `ymin` (long). A sidebar on the right contains information about index mappings and a link to 'Transform your searchable content'.

#### 4. Map class to label and add label field

##### **Purpose:**

Adds a label field with values "Digit" or "Month" based on the class. It helps in grouping and querying data more efficiently.

---

```
POST /image_data_index/_update_by_query
{
  "script": {
    "source": """
      if (ctx._source.class == 'January' || ctx._source.class == 'February' ||
      ctx._source.class == 'March' ||
      ctx._source.class == 'April' || ctx._source.class == 'May' ||
      ctx._source.class == 'June' ||
      ctx._source.class == 'July' || ctx._source.class == 'August' ||
      ctx._source.class == 'September' ||
      ctx._source.class == 'October' || ctx._source.class == 'November' ||
      ctx._source.class == 'December') {
        ctx._source.label = 'Month';
      } else if (ctx._source.class == 'Zero' || ctx._source.class == 'One' ||
      ctx._source.class == 'Two' ||
      ctx._source.class == 'Three' || ctx._source.class == 'Four' ||
      ctx._source.class == 'Five' ||
      ctx._source.class == 'Six' || ctx._source.class == 'Seven' ||
      ctx._source.class == 'Eight' ||
      ctx._source.class == 'Nine') {
        ctx._source.label = 'Digit';
      }
    """,
    "lang": "painless"
  }
}
```

##### **Usefulness:**

- Simplifies querying based on high-level categories.
- Facilitate analytics by grouping similar data.

- **Example:**

## image\_data\_index

[View in Playground](#)

Overview Documents Mappings Pipelines






🔍 Search documents in this index

< 1 2 3 4 5 6 >

Showing 25 of 144. Search results maxed at 10,000 documents.

Document id: 1

Show 6 fewer fields

Field	Contents
 ymin	452
 xmin	273
 filename	"cropped_83_stamp_jpg.rf.42b15bc2d884f6f0b38f2868341f7745.jpg"
 ymax	486
 xmax	393
 width	640
 label	"Digit"
 class	"Eight"
 height	640

Document id: 2

🔍

Field	Contents
 ymin	483
 xmin	288

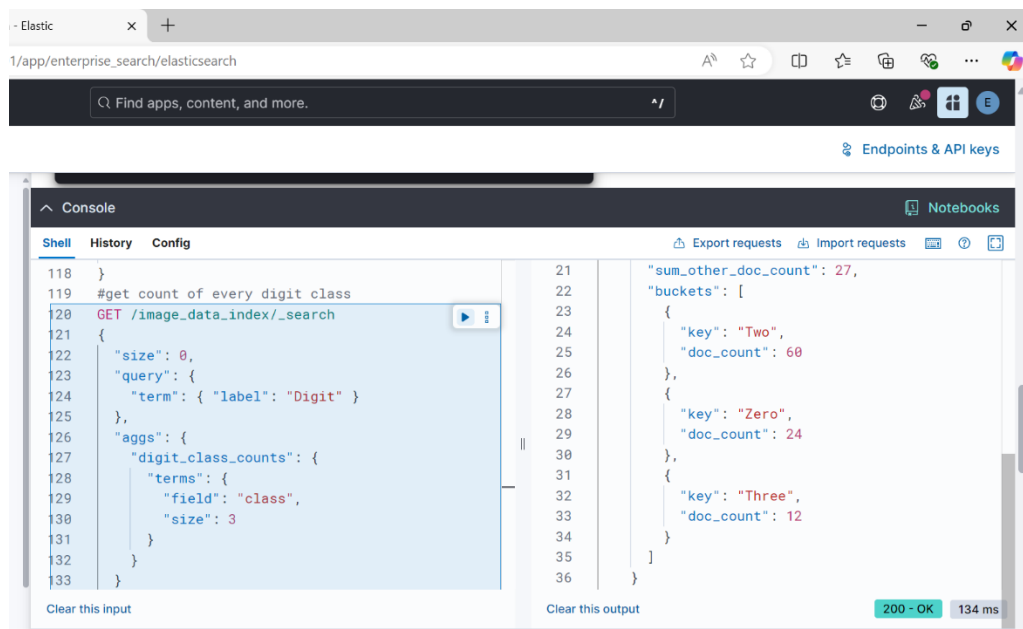
5. Get count of every digit class:

### Purpose:

This query retrieves the counts of all classes labeled as "Digit".

```
GET /image_data_index/_search
{
  "size": 0,
  "query": {
    "term": { "label": "Digit" }
  },
  "aggs": {
    "digit_class_counts": {
      "terms": {
        "field": "class",
        "size": 3
      }
    }
  }
}
```

### Example:



The screenshot shows the Elastic Search console interface. The left pane displays the following query:

```
118 }
119 #get count of every digit class
120 GET /image_data_index/_search
121 {
122   "size": 0,
123   "query": {
124     "term": { "label": "Digit" }
125   },
126   "aggs": {
127     "digit_class_counts": {
128       "terms": {
129         "field": "class",
130         "size": 3
131       }
132     }
133   }
134 }
```

The right pane shows the JSON response:

```
21 {
22   "sum_other_doc_count": 27,
23   "buckets": [
24     {
25       "key": "Two",
26       "doc_count": 60
27     },
28     {
29       "key": "Zero",
30       "doc_count": 24
31     },
32     {
33       "key": "Three",
34       "doc_count": 12
35     }
36   ]
37 }
```

At the bottom right, the status bar indicates "200 - OK" and "134 ms".

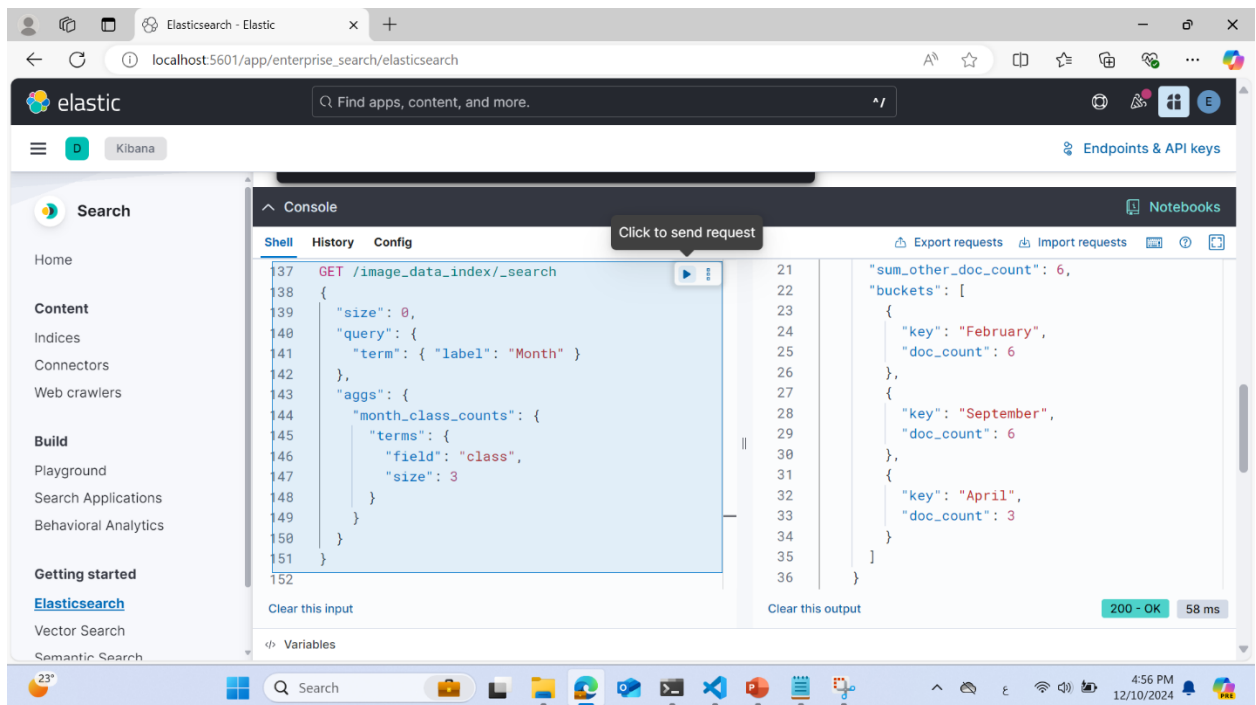
6. Get count of every month class:

**Purpose:**

This query retrieves the counts of all classes labeled as "Month".

```
GET /image_data_index/_search
{
  "size": 0,
  "query": {
    "term": { "label": "Month" }
  },
  "aggs": {
    "month_class_counts": {
      "terms": {
        "field": "class",
        "size": 3
      }
    }
  }
}
```

**Example:**



The screenshot shows the Elastic Kibana interface. The left sidebar contains navigation links for Search, Content, Indices, Connectors, Web crawlers, Build, Playground, Search Applications, Behavioral Analytics, Getting started, Elasticsearch, Vector Search, and Semantic Search. The main console area is titled 'Console' and has tabs for Shell, History, and Config. The Shell tab is active, showing a REST client interface. The input field contains a GET request to `/image_data_index/_search` with a JSON body that filters for documents where `label` is `Month` and aggregates the `class` field using a `terms` aggregation with a size of 3. A 'Click to send request' button is visible. The output field shows the response, which includes a `sum_other_doc_count` of 6 and a `buckets` array with three entries: `February` (6), `September` (6), and `April` (3). The status bar at the bottom indicates a 200 OK response with a 58 ms execution time.

```
GET /image_data_index/_search
{
  "size": 0,
  "query": {
    "term": { "label": "Month" }
  },
  "aggs": {
    "month_class_counts": {
      "terms": {
        "field": "class",
        "size": 3
      }
    }
  }
}
```

```
{
  "sum_other_doc_count": 6,
  "buckets": [
    {
      "key": "February",
      "doc_count": 6
    },
    {
      "key": "September",
      "doc_count": 6
    },
    {
      "key": "April",
      "doc_count": 3
    }
  ]
}
```



## 7. Get the max width and height:

### Purpose:

Finds the maximum width and height values in the dataset. Useful for understanding image dimensions.

```
GET /image_data_index/_search
```

```
{
  "size": 0,
  "aggs": {
    "max_width": {
      "max": {
        "field": "width"
      }
    },
    "max_height": {
      "max": {
        "field": "height"
      }
    }
  }
}
```

### Example:

The screenshot shows the Elastic Kibana interface. The left sidebar contains navigation links: Home, Content, Indices, Connectors, Web crawlers, Build, Playground, Search Applications, Behavioral Analytics, and Getting started. The main area is the 'Console' tab, which displays a REST client interface. The 'Shell' tab is active, showing a search request in the input area and the corresponding JSON response in the output area. The request is a GET to `/image_data_index/_search` with a `size` of 0 and two aggregations: `max_width` (max of `width`) and `max_height` (max of `height`). The response shows `failed: 0`, `hits: []`, and aggregations where `max_width` has a value of 144 and `max_height` has a value of 640. The status bar at the bottom indicates a 200 OK response in 131 ms.

```
GET /image_data_index/_search
{
  "size": 0,
  "aggs": {
    "max_width": {
      "max": {
        "field": "width"
      }
    },
    "max_height": {
      "max": {
        "field": "height"
      }
    }
  }
}
```

```
{
  "failed": 0,
  "hits": {
    "total": {
      "value": 144,
      "relation": "eq"
    },
    "max_score": null,
    "hits": []
  },
  "aggregations": {
    "max_width": {
      "value": 640
    },
    "max_height": {
      "value": 640
    }
  }
}
```

## 8. Calculate the area of shapes

### Purpose:

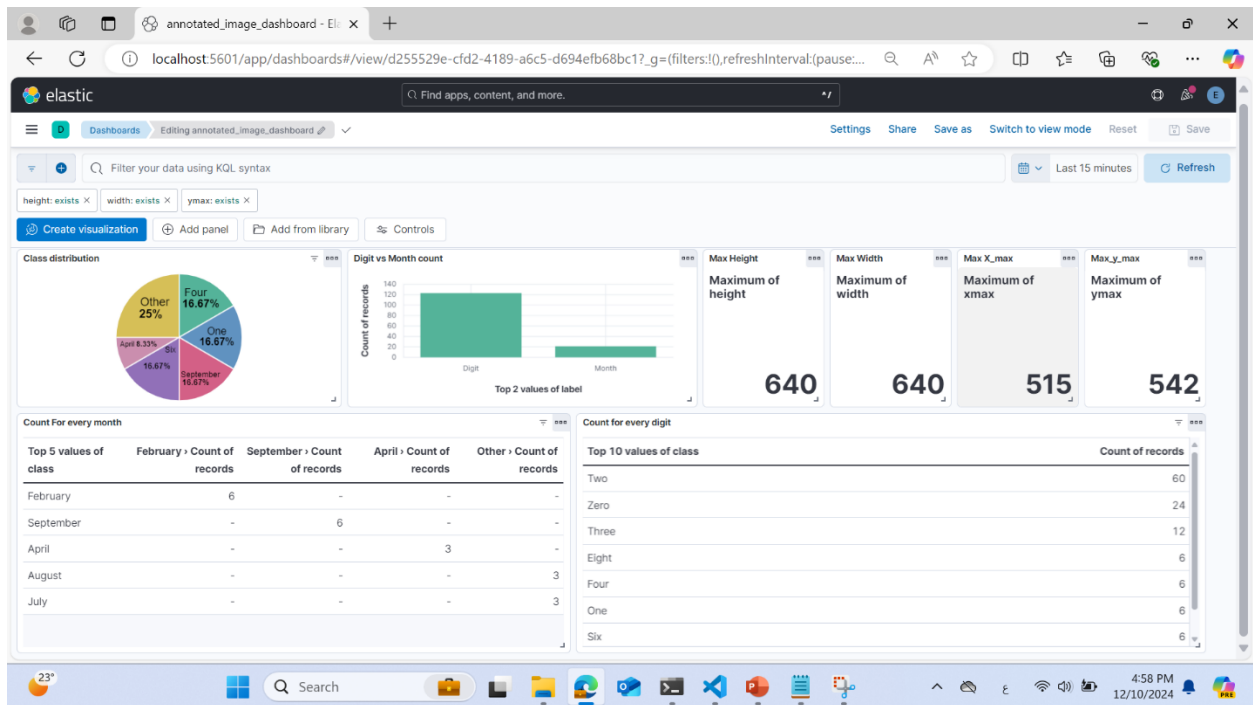
Calculates the area of bounding boxes using the formula  $(x_{\max} - x_{\min}) * (y_{\max} - y_{\min})$ .

```
GET /image_data_index/_search
{
  "_source": ["filename", "class", "xmin", "ymin",
"xmax", "ymax"],
  "query": {
    "match_all": {}
  },
  "script_fields": {
    "bounding_box_area": {
      "script": {
        "lang": "painless",
        "source": "(doc['xmax'].value -
doc['xmin'].value) * (doc['ymax'].value -
doc['ymin'].value)"
      }
    }
  }
}
```

### Example:

The screenshot shows the Elastic Kibana interface. The left sidebar contains navigation links: Home, Content, Indices, Connectors, Web crawlers, Build, Playground, Search Applications, Behavioral Analytics, and Getting started. The main area is titled 'elastic' and contains a search bar. Below the search bar, there's a 'Console' tab with a 'Shell' sub-tab. The console shows a GET request to `/image_data_index/_search` with a query object containing `match_all` and a script field named `bounding_box_area`. The script calculates the area of a bounding box using the formula  $(x_{\max} - x_{\min}) * (y_{\max} - y_{\min})$ . The response shows the calculated area for a document with filename `cropped_83_stamp.jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg` and class `Eight`, resulting in an area of `4080`.

## Kibana Dashboards:



### Bonus:

In the bar tables I did filter by label one by label = "Month" and other by label = "Digit"



## Implementation Documentation:

### Data Ingestion:

- First I have downloaded the annotated images data as TensorFlow csv format and transform this to Json (as I wanted flattened Json not nested one)
- I want to use Bulk API, so I use this code to transform the Json to the format which Bulk API gets:

```
# Prepare the bulk request body with an incrementing index
bulk_data = []
for i, doc in enumerate(documents, start=1): # Starting index at 1
    bulk_data.append({ "index": { "_index": "image_data_index",
    "_id": str(i) } }) # Add index action with incremental _id
    bulk_data.append(doc) # Add the document itself

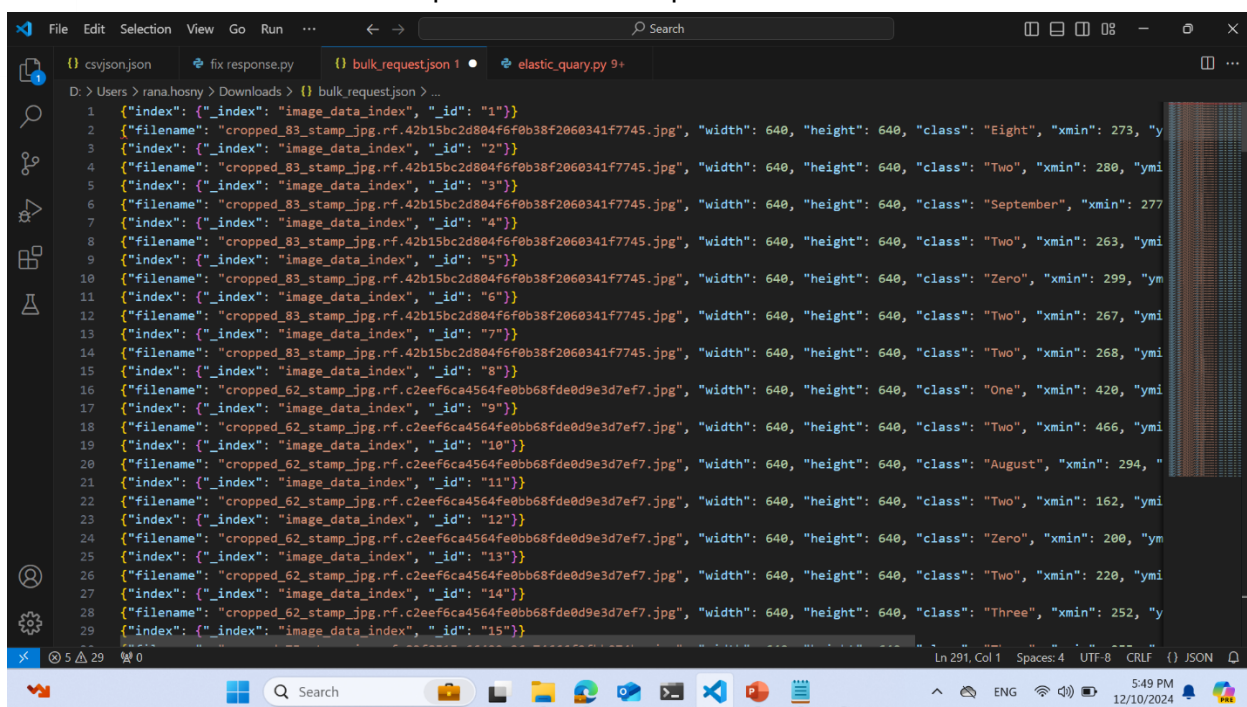
# Convert the bulk data to JSON format and join it with newline
characters
bulk_request = '\n'.join(json.dumps(item) for item in bulk_data)

# Specify the file path
file_path = r"D:\Users\rana.hosny\Downloads\bulk_request.json"

# Save to the file
with open(file_path, 'w') as f:
    f.write(bulk_request)

print(f"Bulk request saved to {file_path}")
```

## Sample of the bulk request file:



```
1 {"index": {"_index": "image_data_index", "_id": "1"}}
2 {"filename": "cropped_83_stamp_jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg", "width": 640, "height": 640, "class": "Eight", "xmin": 273, "ymin": 247, "xmax": 473, "ymax": 346}
3 {"index": {"_index": "image_data_index", "_id": "2"}}
4 {"filename": "cropped_83_stamp_jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg", "width": 640, "height": 640, "class": "Two", "xmin": 280, "ymin": 247, "xmax": 473, "ymax": 346}
5 {"index": {"_index": "image_data_index", "_id": "3"}}
6 {"filename": "cropped_83_stamp_jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg", "width": 640, "height": 640, "class": "September", "xmin": 277, "ymin": 259, "xmax": 371, "ymax": 350}
7 {"index": {"_index": "image_data_index", "_id": "4"}}
8 {"filename": "cropped_83_stamp_jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg", "width": 640, "height": 640, "class": "Two", "xmin": 263, "ymin": 247, "xmax": 473, "ymax": 346}
9 {"index": {"_index": "image_data_index", "_id": "5"}}
10 {"filename": "cropped_83_stamp_jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg", "width": 640, "height": 640, "class": "Zero", "xmin": 299, "ymin": 247, "xmax": 473, "ymax": 346}
11 {"index": {"_index": "image_data_index", "_id": "6"}}
12 {"filename": "cropped_83_stamp_jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg", "width": 640, "height": 640, "class": "Two", "xmin": 267, "ymin": 247, "xmax": 473, "ymax": 346}
13 {"index": {"_index": "image_data_index", "_id": "7"}}
14 {"filename": "cropped_83_stamp_jpg.rf.42b15bc2d804f6f0b38f2060341f7745.jpg", "width": 640, "height": 640, "class": "Two", "xmin": 268, "ymin": 247, "xmax": 473, "ymax": 346}
15 {"index": {"_index": "image_data_index", "_id": "8"}}
16 {"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "One", "xmin": 420, "ymin": 247, "xmax": 473, "ymax": 346}
17 {"index": {"_index": "image_data_index", "_id": "9"}}
18 {"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Two", "xmin": 466, "ymin": 247, "xmax": 473, "ymax": 346}
19 {"index": {"_index": "image_data_index", "_id": "10"}}
20 {"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "August", "xmin": 294, "ymin": 247, "xmax": 473, "ymax": 346}
21 {"index": {"_index": "image_data_index", "_id": "11"}}
22 {"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Two", "xmin": 162, "ymin": 247, "xmax": 473, "ymax": 346}
23 {"index": {"_index": "image_data_index", "_id": "12"}}
24 {"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Zero", "xmin": 200, "ymin": 247, "xmax": 473, "ymax": 346}
25 {"index": {"_index": "image_data_index", "_id": "13"}}
26 {"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Two", "xmin": 220, "ymin": 247, "xmax": 473, "ymax": 346}
27 {"index": {"_index": "image_data_index", "_id": "14"}}
28 {"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Three", "xmin": 252, "ymin": 247, "xmax": 473, "ymax": 346}
29 {"index": {"_index": "image_data_index", "_id": "15"}}
```

- Then insert the data like this example:(I have inserted it in chunks as the command has limited size input)

```
curl -X PUT https://172.18.0.2:9200/_bulk -k -u "elastic:MT2t34+MmwG6rouktA0E" -H
"Content-Type: application/json" -d '
{"index": {"_index": "image_data_index", "_id": "139"}}
{"filename": "cropped_69_stamp_jpg.rf.16bcae47cb7897f4754c41ebfe358519.jpg",
"width": 640, "height": 640, "class": "Eight", "xmin": 439, "ymin": 247, "xmax":
473, "ymax": 346}
{"index": {"_index": "image_data_index", "_id": "140"}}
{"filename": "cropped_69_stamp_jpg.rf.16bcae47cb7897f4754c41ebfe358519.jpg",
"width": 640, "height": 640, "class": "September", "xmin": 264, "ymin": 259, "xmax":
371, "ymax": 350}
```

```

{"index": {"_index": "image_data_index", "_id": "141"}}
{"filename": "cropped_69_stamp_jpg.rf.16bcae47cb7897f4754c41ebfe358519.jpg",
"width": 640, "height": 640, "class": "Two", "xmin": 226, "ymin": 266, "xmax":
245, "ymax": 363}
{"index": {"_index": "image_data_index", "_id": "142"}}
{"filename": "cropped_69_stamp_jpg.rf.16bcae47cb7897f4754c41ebfe358519.jpg",
"width": 640, "height": 640, "class": "Two", "xmin": 144, "ymin": 269, "xmax":
175, "ymax": 370}
{"index": {"_index": "image_data_index", "_id": "143"}}
{"filename": "cropped_69_stamp_jpg.rf.16bcae47cb7897f4754c41ebfe358519.jpg",
"width": 640, "height": 640, "class": "Zero", "xmin": 171, "ymin": 291,
"xmax": 197, "ymax": 341}
{"index": {"_index": "image_data_index", "_id": "144"}}
{"filename": "cropped_69_stamp_jpg.rf.16bcae47cb7897f4754c41ebfe358519.jpg",
"width": 640, "height": 640, "class": "Two", "xmin": 195, "ymin": 263, "xmax":
223, "ymax": 367}
,

```

```

File ranahossny@rana-hosny: ~ x + v
{
  "xmin": 466, "ymin": 290, "xmax": 506, "ymax": 420}
{"index": {"_index": "image_data_index", "_id": "10"}}
{"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Augu
st", "xmin": 294, "ymin": 299, "xmax": 408, "ymax": 432}
{"index": {"_index": "image_data_index", "_id": "11"}}
{"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Two"
, "xmin": 162, "ymin": 315, "xmax": 200, "ymax": 438}
{"index": {"_index": "image_data_index", "_id": "12"}}
{"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Zero
", "xmin": 200, "ymin": 355, "xmax": 221, "ymax": 400}
{"index": {"_index": "image_data_index", "_id": "13"}}
{"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Two"
, "xmin": 220, "ymin": 310, "xmax": 247, "ymax": 427}
{"index": {"_index": "image_data_index", "_id": "14"}}
{"filename": "cropped_62_stamp_jpg.rf.c2eef6ca4564fe0bb68fde0d9e3d7ef7.jpg", "width": 640, "height": 640, "class": "Thre
e", "xmin": 252, "ymin": 319, "xmax": 276, "ymax": 427}
,
{"errors": false, "took": 0, "items": [{"index": {"_index": "image_data_index", "_id": "7", "_version": 1, "result": "created", "_shar
ds": {"total": 2, "successful": 1, "failed": 0}, "_seq_no": 10, "_primary_term": 1, "status": 201}}, {"index": {"_index": "image_data_i
ndex", "_id": "8", "_version": 1, "result": "created", "_shards": {"total": 2, "successful": 1, "failed": 0}, "_seq_no": 11, "_primary_t
erm": 1, "status": 201}}, {"index": {"_index": "image_data_index", "_id": "9", "_version": 1, "result": "created", "_shards": {"total"
: 2, "successful": 1, "failed": 0}, "_seq_no": 12, "_primary_term": 1, "status": 201}}, {"index": {"_index": "image_data_index", "_id":
"10", "_version": 1, "result": "created", "_shards": {"total": 2, "successful": 1, "failed": 0}, "_seq_no": 13, "_primary_term": 1, "sta
tus": 201}}, {"index": {"_index": "image_data_index", "_id": "11", "_version": 1, "result": "created", "_shards": {"total": 2, "succes
sful": 1, "failed": 0}, "_seq_no": 14, "_primary_term": 1, "status": 201}}, {"index": {"_index": "image_data_index", "_id": "12", "_ver
sion": 1, "result": "created", "_shards": {"total": 2, "successful": 1, "failed": 0}, "_seq_no": 15, "_primary_term": 1, "status": 201}},
{"index": {"_index": "image_data_index", "_id": "13", "_version": 1, "result": "created", "_shards": {"total": 2, "successful": 1, "f
ailed": 0}, "_seq_no": 16, "_primary_term": 1, "status": 201}}, {"index": {"_index": "image_data_index", "_id": "14", "_version": 1, "r
esult": "created", "_shards": {"total": 2, "successful": 1, "failed": 0}, "_seq_no": 17, "_primary_term": 1, "status": 201}}]}
ranahoss
ny@rana-hosny: ~$ |
bulk_request_send_to_elasticsearch: 2025-10-10 10:00:00.000000 bulk_request_send

```



## Mapping Decisions:

When creating the index `image_data_index` and defining its mapping, the choice of field types was driven by the structure and usage of the data.

---

```
PUT /image_data_index
{
  "mappings": {
    "properties": {
      "filename": {
        "type": "keyword"
      },
      "width": {
        "type": "integer"
      },
      "height": {
        "type": "integer"
      },
      "class": {
        "type": "keyword"
      },
      "coordinates": {
        "properties": {
          "xmin": {
            "type": "integer"
          },
          "ymin": {
            "type": "integer"
          },
          "xmax": {
            "type": "integer"
          },
          "ymax": {
            "type": "integer"
          }
        }
      }
    }
  }
}
```

---

- filename (keyword): For exact searches by unique file names.
- width and height (integer): Store image dimensions for filtering or aggregation.
- class (keyword): Categorical data for grouping and counting classes.
- coordinates (nested integers): Bounding box dimensions for area
- calculations.

## Adding the label Field:

```
PUT /image_data_index/_mapping
{
  "properties": {
    "label": {
      "type": "keyword"
    }
  }
}
```

**label (keyword):** Represents static high-level categories like "Digit" or "Month" for efficient filtering and grouping.

## Query Design:

- **Search Specific Class:** Retrieves documents matching a specific class for precise filtering.
- **Count Classes:** Aggregates the distribution of class occurrences to understand data composition.
- **Count by Label:** Filters and counts classes within specific high-level labels for insights.
- **Find Max Dimensions:** Identifies the largest image dimensions for analysis.
- **Calculate Bounding Box Area:** Computes areas to evaluate object sizes within images.

## Visualization Choices:

- Pie plot: to show the total percentage of distribution of every class
- 4 Kpis to show the max values of X\_min,X\_max,Y\_min and Y\_max
- The table shows the count of instances in every digit class
- The table shows the count of instances in every month class





[Video Demo:](#)