

# Examen - Rune D'Haluwin

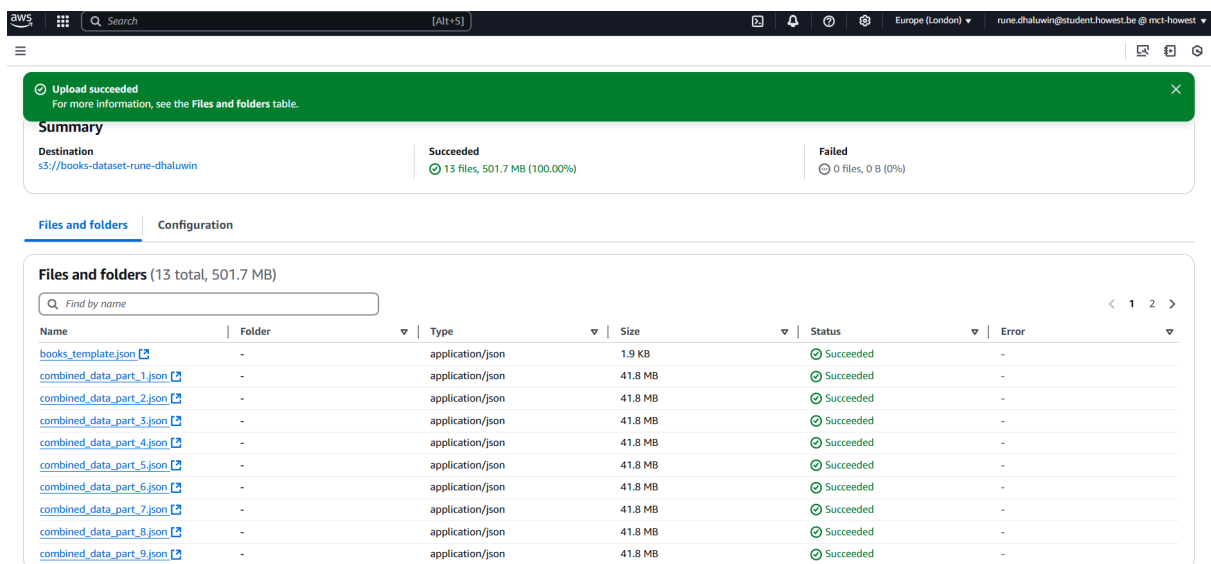
## Opdracht

Er is al een werkend docker compose project voorzien, je hoeft in principe niets te wijzigen aan wat er al gegeven is! In mijn repo is er een mapje logstash met daarin 2 mapjes: data en pipeline. In het mapje pipeline zit het bestand books.conf (Die jij zelf zult moeten maken). In het mapje data zitten verschillende json bestanden (combined\_data\_part\_1.json, combined\_data\_part\_2.json, ... , tot en met combined\_data\_part\_12.json) en een bestand books\_template.json (De combined\_data\_part\_1.json tot en met combined\_data\_part\_12.json bevat dus de hele dataset over boeken.

## 1. Gebruik cloud storage:

Kies een cloud oplossing en zorg dat de data in de cloud word opgeslagen in plaats van lokaal

Mijn oplossing: AWS S3 Bucket aangemaakt en alle json files daarin geüpload



The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various icons. Below the navigation bar, a green banner indicates "Upload succeeded" for 13 files (501.7 MB) to the bucket "s3://books-dataset-rune-dhaluwin". Below this, the "Files and folders" tab is selected, showing a table of the uploaded files. The table has columns for Name, Folder, Type, Size, Status, and Error. All files are listed as "application/json" and "Succeeded".

Name	Folder	Type	Size	Status	Error
books_template.json	-	application/json	1.9 KB	Succeeded	-
combined_data_part_1.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_2.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_3.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_4.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_5.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_6.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_7.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_8.json	-	application/json	41.8 MB	Succeeded	-
combined_data_part_9.json	-	application/json	41.8 MB	Succeeded	-

## 2. Importeer de data:

Zoek uit hoe je de data kunt verwerken met de Elastic Stack. De data moet uiteraard opgeslagen worden in Elasticsearch! De verwerking van de data moet

voldoen aan enkele specifieke vereisten:

- De data moet worden opgeslagen in een index genaamd: books\_dataset.
- Maak gebruik van de meegeleverde template:
  - template\_name: "books"
  - template ⇒ "<path>books\_template.json" (<path> = placeholder )
  - template\_overwrite ⇒ true
- Bekijk de dataset en zorg ervoor dat jouw output overeenkomt met de gegeven voorbeelden. Je data moet exact hetzelfde formaat hebben! Dit betekent dat je enkele filters nodig zult hebben om dit te bereiken! Wees kritisch en let goed op de details! het is prima als de volgorde van de velden licht verschilt, maar uiteindelijk moet je wel dezelfde velden hebben als in de voorbeelden!
- Het is belangrijk dat je pipeline efficiënt is.
- TIPS! : gebruik meerdere mutate filters, maak het op zo'n eenvoudige manier mogelijk.

Mijn oplossing:

ik heb nu een basic pipeline die de data uit mijn amazon bucket haalt en hem weer geeft en een .env bestand gemaakt waarbij mijn geheime keys etc komen, deze staat in mijn .gitignore en .dockerignore

```
input {
  s3 {
    bucket ⇒ "books-dataset-rune-dhaluwin"
    prefix ⇒ ""
    region ⇒ "${AWS_REGION}"
    access_key_id ⇒ "${AWS_ACCESS_KEY_ID}"
    secret_access_key ⇒ "${AWS_SECRET_ACCESS_KEY}"
    codec ⇒ "json_lines"
  }
}

filter {
  # hier komen de mutaties
```

```

}

output {
  stdout { codec => rubydebug }
}

```

Als ik dan de logstash al run via deze commands krijg ik dit resultaat (dus de data komt nu binnen vanuit mijn amazon bucket):

```
docker compose exec logstash /bin/bash
```

```
logstash -f /usr/share/logstash/pipeline/books.conf --config.reload.automatic
```

```

      "user" => {
        "location" => "atlanta, georgia, usa",
        "age" => "25.0",
        "user_id" => "152958"
      }
    }
    {
      "book" => {
        "isbn" => "0060184728",
        "image_urls" => {
          "small" => "http://images.amazon.com/images/P/0060184728.01.THUMBZZZ.jpg",
          "large" => "http://images.amazon.com/images/P/0060184728.01.LZZZZZZZ.jpg",
          "medium" => "http://images.amazon.com/images/P/0060184728.01.MZZZZZZZ.jpg"
        },
        "author" => "Michael S. Sanders",
        "year_of_publication" => 2002.0,
        "publisher" => "HarperCollins",
        "title" => "From Here, You Can't See Paris : Seasons of a French Village and Its Restaurant"
      },
      "book_rating" => 0,
      "@version" => "1",
      "@timestamp" => 2025-05-12T12:58:47.503048645Z,
      "user" => {
        "location" => "felixstowe, suffolk, united kingdom",
        "age" => "57.0",
        "user_id" => "209516"
      }
    }
  }
[2025-05-12T12:58:48,906][INFO ][logstash.javapipeline] [[main] Pipeline terminated {"pipeline.id"=>"main"}
^C[2025-05-12T12:58:49,382][FATAL][logstash.runner] ] SIGINT received. Terminating immediately..
[2025-05-12T12:58:49,808][FATAL][org.logstash.Logstash] ]
org.jruby.exceptions.ThreadKill: null
root@31ddc8ecf111:/usr/share/logstash#

```

Nu heb ik gezorgd dat het correct word gefilterd en de output goed staat:

```

input {
  s3 {
    bucket => "books-dataset-rune-dhaluwin"
    prefix => ""
    region => "${AWS_REGION}"
    access_key_id => "${AWS_ACCESS_KEY_ID}"
  }
}

```

```

    secret_access_key ⇒ "${AWS_SECRET_ACCESS_KEY}"
    codec ⇒ "json_lines"
  }
}

filter {
  # Step 1: Split user location into separate fields (city, state, country)
  mutate {
    split ⇒ { "[user][location]" ⇒ ", " }
    add_field ⇒ {
      "[user][city]" ⇒ "%{[user][location][0]}"
      "[user][state]" ⇒ "%{[user][location][1]}"
      "[user][country]" ⇒ "%{[user][location][2]}"
    }
  }

  # Step 2: Clean up - remove the original location field
  mutate {
    remove_field ⇒ "[user][location]"
  }

  # Step 3: Convert numeric fields to proper types
  mutate {
    convert ⇒ {
      "[user][user_id]" ⇒ "integer"
      "book_rating" ⇒ "integer"
    }
  }

  # Step 4: Handle age field - convert to integer or null
  if [user][age] == "" or [user][age] == "null" {
    mutate { replace ⇒ { "[user][age]" ⇒ nil } }
  } else {
    mutate { convert ⇒ { "[user][age]" ⇒ "integer" } }
  }
}

output {

```

```

elasticsearch {
  hosts => ["http://elasticsearch:9200"]
  index => "books_dataset"
  template => "/usr/share/logstash/data/books_template.json"
  template_name => "books"
  template_overwrite => true
}

stdout { codec => rubydebug }
}

```

Dit zie ik nu in kibana (in Dev tools) als ik deze query doe: GET books\_dataset/\_search

```

1 {
2   "took": 578,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 10000,
13      "relation": "gte"
14    },
15    "max_score": 1,
16    "hits": [
17      {
18        "_index": "books_dataset",
19        "_id": "MuhxjBt2KSSThrigNF",
20        "_score": 1,
21        "_source": {
22          "version": "1",
23          "book_rating": 0,
24          "book": {
25            "author": "Erica Spindler",
26            "image_urls": {
27              "small": "http://images.amazon.com/images/P/1551669145.01.THUMBZZZ.jpg",
28              "medium": "http://images.amazon.com/images/P/1551669145.01.MCZZZZZZZ.jpg",
29              "large": "http://images.amazon.com/images/P/1551669145.01.LZZZZZZZZZ.jpg"
30            },
31            "year_of_publication": 2002,
32            "title": "Dead Run",
33            "publisher": "Kura",
34            "isbn": "1551669145"
35          },
36          "timestamp": "2025-05-12T13:14:06.229472451Z",
37          "user": {
38            "state": "california",
39            "country": "usa",
40            "..."

```

## 3. Queries en aggregaties

Beantwoord de volgende vragen met behulp van de Elasticsearch DSL

### 3.1 De heel eenvoudige:

- Hoeveel documenten zijn er in totaal? ⇒ 945287

```
GET books_dataset/_count
```

```

1  {
2    "count": 945287,
3    "_shards": {
4      "total": 1,
5      "successful": 1,
6      "skipped": 0,
7      "failed": 0
8    }
9  }

```

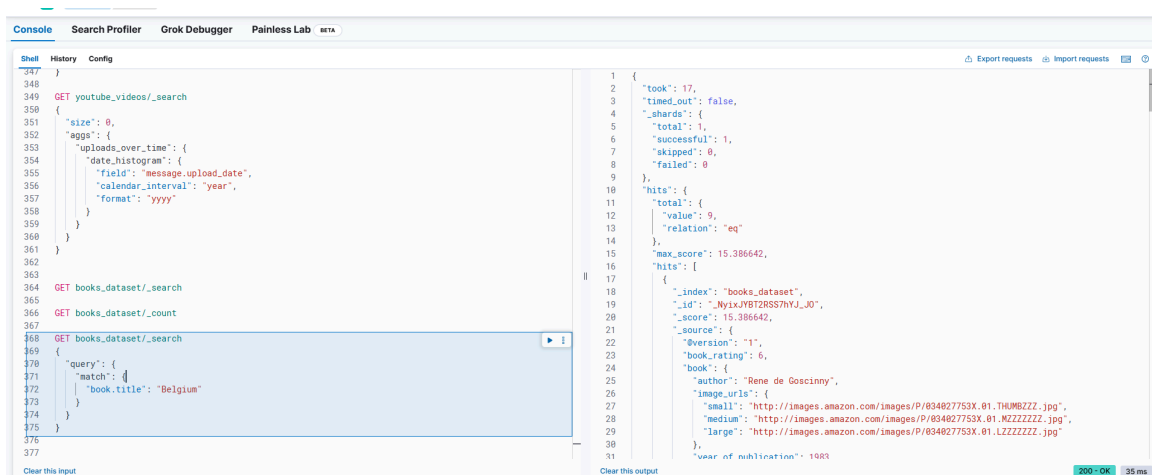
- Zoek alle documenten waarvan de titel "Belgium" bevat. ⇒ 9 documenten bevatten Belgium in de titel.

GET books\_dataset/\_search

```

{
  "query": {
    "match": {
      "book.title": "Belgium"
    }
  }
}

```

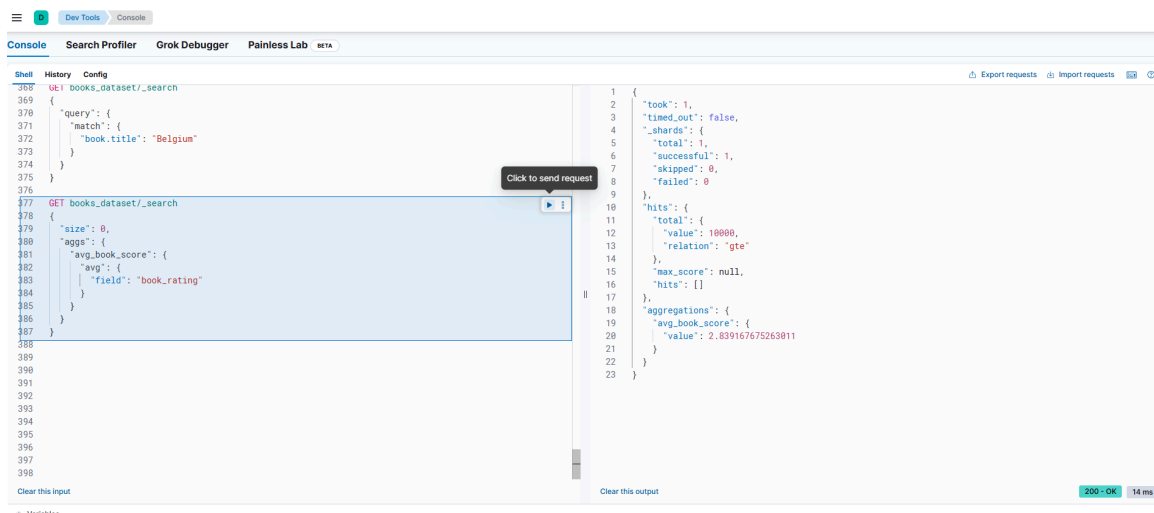


## 3.2 De eenvoudige:

- Wat is de gemiddelde score van een boek?  $\Rightarrow 2.839167675263011$

GET books\_dataset/\_search

```
{
  "size": 0,
  "aggs": {
    "avg_book_score": {
      "avg": {
        "field": "book_rating"
      }
    }
  }
}
```



- Wat is de mediaanscore van een boek?  $\Rightarrow 0.2591136226760329$

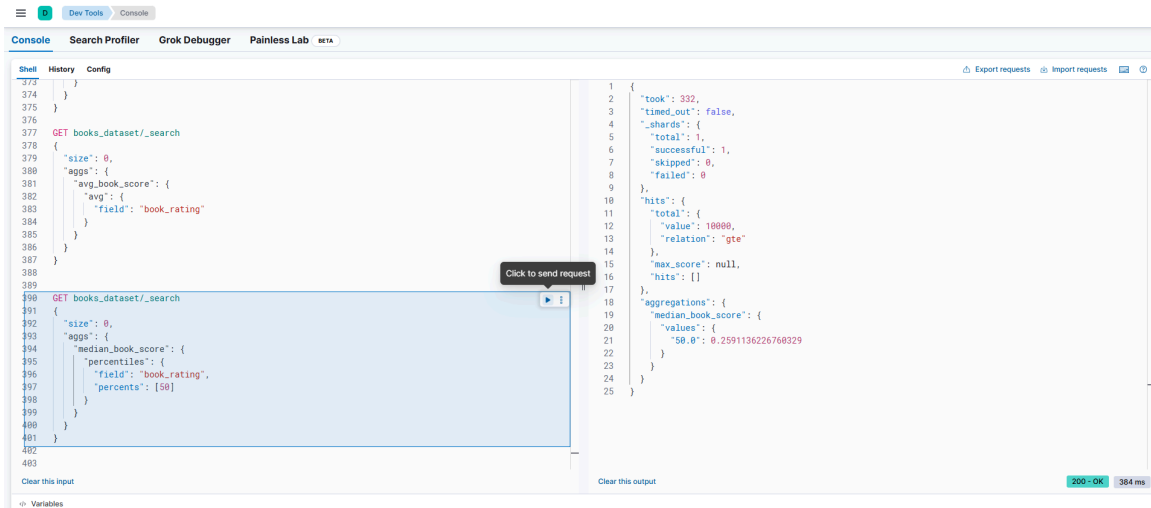
GET books\_dataset/\_search

```
{
  "size": 0,
  "aggs": {
    "median_book_score": {
      "percentiles": {
        "field": "book_rating",
        "percents": [50]
      }
    }
  }
}
```

```

    }
  }
}

```



### 3.3 Iets moeilijker:

- Welke gebruiker (`user_id`) heeft in totaal de meeste scores gepost? Uit welk land komt deze gebruiker? Probeer overbodige velden uit het resultaat te filteren, kijk naar de (voorbeeld) hits sectie hieronder:
- `user_id: 11676` en Land van deze gebruiker: "n/a" dus het land is onbekend van deze gebruiker.

```

"hits": [
  {
    "_index": "books_dataset",
    "_id": "0123456789-123456",
    "_score": 1,
    "_source": {
      "user": {
        "country": "Examnia",
        "user_id": 123456
      }
    }
  }
]

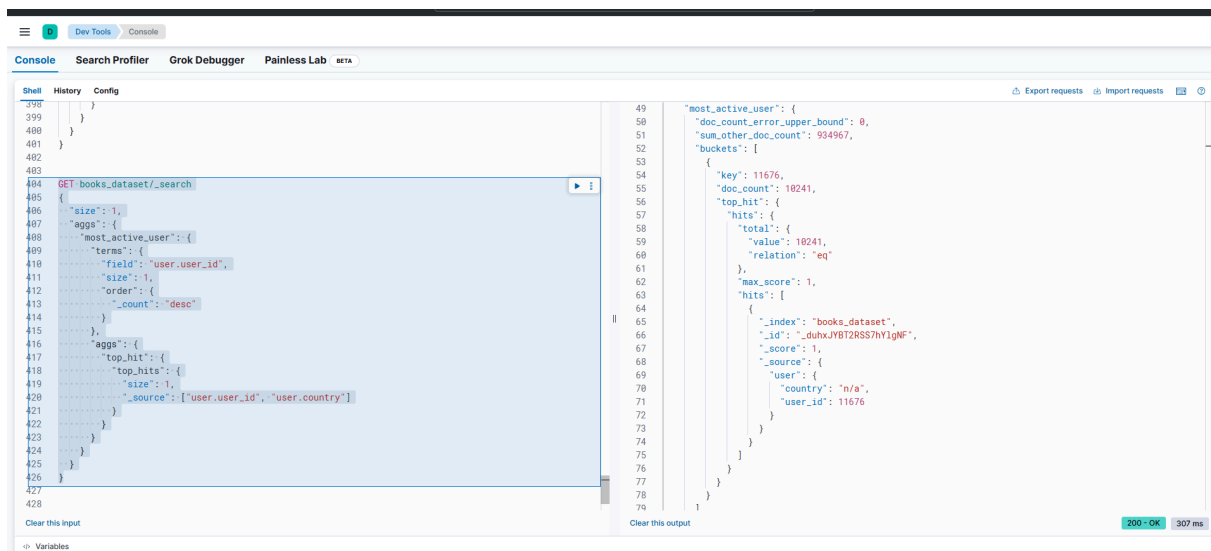
```



```
}  
]
```

GET books\_dataset/\_search

```
{  
  "size": 1,  
  "aggs": {  
    "most_active_user": {  
      "terms": {  
        "field": "user.user_id",  
        "size": 1,  
        "order": {  
          "_count": "desc"  
        }  
      },  
    },  
    "aggs": {  
      "top_hit": {  
        "top_hits": {  
          "size": 1,  
          "_source": ["user.user_id", "user.country"]  
        }  
      }  
    }  
  }  
}
```



### 3.3 Na aanpassingen:

# 3.3 na aanpassingen:

GET books\_dataset/\_search

```
{
  "size": 1,
  "_source": ["user.user_id", "user.country"],
  "aggs": {
    "most_active_user": {
      "terms": {
        "field": "user.user_id",
        "size": 1,
        "order": {
          "_count": "desc"
        }
      },
    },
    "aggs": {
      "top_hit": {
        "top_hits": {
          "size": 1,
          "_source": ["user.user_id", "user.country"]
        }
      }
    }
  }
}
```

```
}
}
```

Resultaat: 20 overbodige lijntjes minder

The screenshot shows a REST client interface with a 'Console' tab. On the left, a REST API call is defined with the following details:

- Method: GET
- URL: books\_dataset/\_search
- Headers: Content-Type: application/json
- Body: A JSON object with the following structure:
 

```
{
  "size": 1,
  "_source": ["user.user_id", "user.country"],
  "aggs": {
    "most_active_user": {
      "terms": {
        "field": "user.user_id",
        "size": 1,
        "order": {
          "_count": "desc"
        }
      }
    }
  },
  "aggs": {
    "top_hits": {
      "top_hits": {
        "size": 1,

```

On the right, the response is shown as a JSON object:

```
{
  "key": 11676,
  "doc_count": 18241,
  "top_hit": {
    "total": {
      "value": 18241,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "books_dataset",
        "_id": "_duhxJyBT2R5S7hYlgNF",
        "_score": 1,
        "_source": {
          "user": {
            "country": "n/a",
            "user_id": 11676
          }
        }
      }
    ]
  }
}
```

### 3.4 Moeilijker:

- Welk boek heeft de hoogste gemiddelde score binnen de 10 meest beoordeelde boeken? De output zou er ongeveer als volgt kunnen uitzien:

```
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "succesfull": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 10000,
      "relation": "gte"
    },
    "max_score": null,
    "hits": []
  }
}
```

```

"<replaced>": {
  "<some_name>": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 1021954,
    "b...": [
      {
        "key": "0123456789",
        "doc_count": 123,
        "avg_score": {
          "value": 4.567
        }
      }
    ]
  }
}

```

```

GET books_dataset/_search
{
  "size": 0,
  "aggs": {
    "most_reviewed_books": {
      "terms": {
        "field": "book.title.keyword",
        "size": 10,
        "order": {
          "_count": "desc" // Sorteer op aantal beoordelingen (doc_count)
        }
      },
      "aggs": {
        "avg_score": {
          "avg": {
            "field": "book_rating"
          }
        }
      }
    }
  }
}

```

ConsoleSearch ProfilerGrok DebuggerPainless LabGETA

ShellHistoryConfig

```
447 }
448 }
449 }
450 }
451 }
452 }
453
454 GET /books/_search
455 {
456   "size": 0,
457   "aggs": {
458     "most_reviewed_books": {
459       "terms": {
460         "field": "book_title.keyword",
461         "size": 10,
462         "order": {
463           "_count": { "desc" } // Sorter op aantal beoordelingen (doc_count)
464         }
465       },
466       "aggs": {
467         "avg_score": {
468           "avg": {
469             "field": "book_rating"
470           }
471         }
472       }
473     }
474   }
475 }
476
477
```

Clear this input

Variables

Export requestsImport requests

```
1 {
2   "took": 125,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 10000,
13      "relation": "gte"
14    },
15    "max_score": null,
16    "hits": []
17  }
18  "aggregations": {
19    "most_reviewed_books": {
20      "doc_count_error_upper_bound": 0,
21      "sum_other_doc_count": 935940,
22      "buckets": [
23        {
24          "key": "Wild Animals",
25          "doc_count": 2311,
26          "avg_score": {
27            "value": 1.025097364580215
28          }
29        },
30        {
31          "key": "The Inevitable Rites of Nature"
```

200 - OK161 ms

dus het is boek: **The Da Vinci Code** die 818 keer beoordeeld is en de gemiddelde score **4.68**

```
{
  "key": "The Da Vinci Code",
  "doc_count": 818,
  "avg_score": {
    "value": 4.677261613691932
  }
},
{
```

Boek	Aantal beoordelingen	Gemiddelde score
Wild Animus	2311	1.03
The Lovely Bones: A Novel	1191	4.48
<b>The Da Vinci Code</b>	818	<b>4.68</b>
Bridget Jones's Diary	762	3.58
The Nanny Diaries: A Novel	760	3.58
A Painted House	755	3.16
The Secret Life of Bees	708	4.46

Divine Secrets of the Ya-Ya Sisterhood: A Novel	686	3.42
The Red Tent (Bestselling Backlist)	661	4.32
Angels & Demons	616	3.72

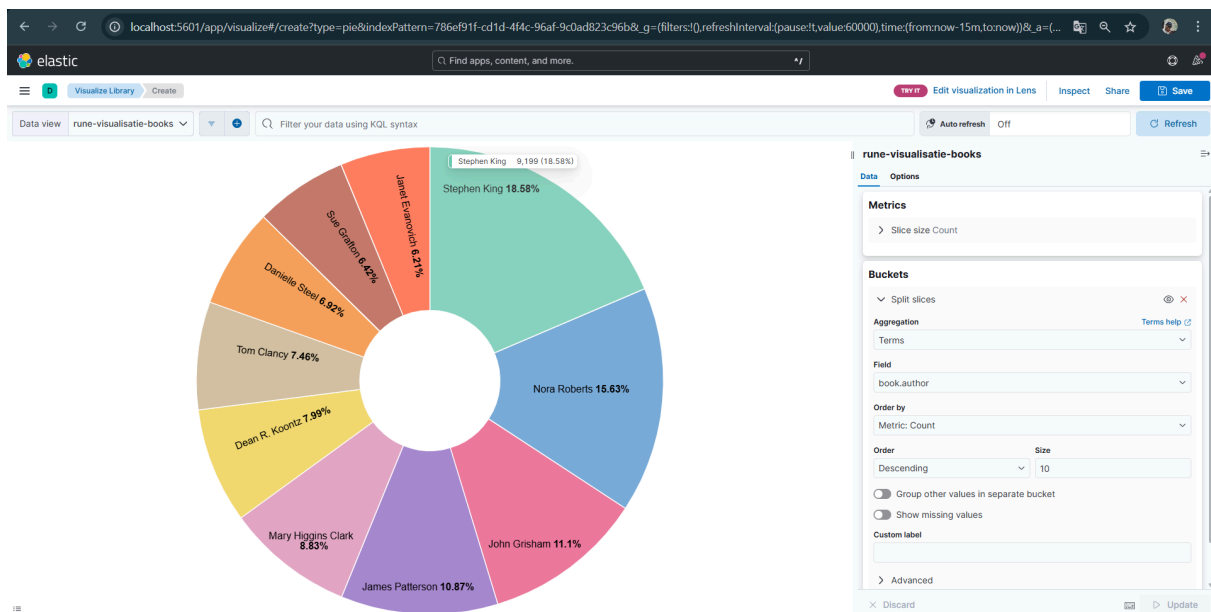
## 4. Visualisatie

Probeer onderstaande visualisatie na te maken. Wat toont deze eigenlijk?

stap 1: ik heb hem aangemaakt in "Visualize library" dan "Create data view"  
dan `books_dataset*`. Daarna heb ik geen time filter field name gedaan. en daarna Create data view

stap 2: alle stappen om dus je pi op te splitsen via terms aggregation per author en dan 10 geselecteerd:

**Deze toont welke auteur de meeste boeken heeft geschreven.**



## 5. Clusterstatus

Wat is de status van de cluster? Let uit!

Ik doe deze queries:

GET \_cluster/health

```
{
  "cluster_name": "docker-cluster",
  "status": "yellow",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "active_primary_shards": 32,
  "active_shards": 32,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 2,
  "unassigned_primary_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 94.11764705882352
}
```

GET \_cat/indices

green	open	.internal.alerts-transform.health.alerts-default-000001	iJZ\
green	open	.internal.alerts-observability.logs.alerts-default-000001	xNF
green	open	.internal.alerts-observability.uptime.alerts-default-000001	pV
green	open	.internal.alerts-ml.anomaly-detection.alerts-default-000001	r
green	open	.internal.alerts-observability.slo.alerts-default-000001	Cbvl
green	open	.internal.alerts-observability.apm.alerts-default-000001	4a2
green	open	.internal.alerts-default.alerts-default-000001	ZS4-x1c
green	open	.internal.alerts-observability.metrics.alerts-default-000001	MI
green	open	.internal.alerts-ml.anomaly-detection-health.alerts-default-000001	MI
green	open	.internal.alerts-observability.threshold.alerts-default-000001	N

green	open	.internal.alerts-security.alerts-default-000001	DRchjZi
yellow	open	books_dataset	HDYB_SrEQgC_aTz
green	open	.internal.alerts-stack.alerts-default-000001	do5Uqh

GET \_cat/shards

.kibana_entities-definitions-1	0 p	STARTED	44	26.7k
.internal.alerts-observability.slo.alerts-default-000001	0 p	STARTED		
.kibana_usage_counters_8.18.1_001	0 p	STARTED	75	1
.internal.alerts-default.alerts-default-000001	0 p	STARTED	0	
.internal.alerts-observability.uptime.alerts-default-000001	0 p	STARTED		
.internal.alerts-stack.alerts-default-000001	0 p	STARTED	0	
.internal.alerts-ml.anomaly-detection-health.alerts-default-000001	0 p	STARTED		
.slo-observability.summary-v3.4	0 p	STARTED	0	2
.kibana-siem-rule-migrations-prebuilt.rules	0 p	STARTED	0	
.kibana_task_manager_8.18.1_001	0 p	STARTED	34	12
.internal.alerts-observability.metrics.alerts-default-000001	0 p	STARTED		
.internal.alerts-observability.threshold.alerts-default-000001	0 p	STARTED		
.slo-observability.sli-v3.4	0 p	STARTED	0	249b
.kibana_ingest_8.18.1_001	0 p	STARTED	0	249b
.kibana_alerting_cases_8.18.1_001	0 p	STARTED	1	6.9
.kibana-siem-rule-migrations-integrations	0 p	STARTED	0	
.kibana_analytics_8.18.1_001	0 p	STARTED	7	2.3ml
.internal.alerts-observability.apm.alerts-default-000001	0 p	STARTED		
.ds-.logs-deprecation.elasticsearch-default-2025.05.12-000001	0 p	STARTED		
.internal.alerts-security.alerts-default-000001	0 p	STARTED	0	
.ds-ilm-history-7-2025.05.12-000001	0 p	STARTED	39	
.apm-source-map	0 p	STARTED	0	249b
.internal.alerts-observability.logs.alerts-default-000001	0 p	STARTED		
.kibana_security_solution_8.18.1_001	0 p	STARTED	3	3
.kibana_8.18.1_001	0 p	STARTED	30	98.3kb
books_dataset	0 p	STARTED	945287	293m
books_dataset	0 r	UNASSIGNED		
books_dataset	0 r	UNASSIGNED		
.internal.alerts-transform.health.alerts-default-000001	0 p	STARTED		
.slo-observability.summary-v3.4.temp	0 p	STARTED	0	



.apm-agent-configuration	0 p STARTED	0	249
.internal.alerts-ml.anomaly-detection.alerts-default-000001	0 p STARTED		
.apm-custom-link	0 p STARTED	0	249b
.async-search	0 p STARTED	7	25.4kb 25
.ds-.kibana-event-log-ds-2025.05.12-000001	0 p STARTED		

## ✓ Clusterstatus en Uitleg

### 📋 Samenvatting van de Gegevens

- **Cluster naam:** `docker-cluster`
- **Status:** `yellow`
- **Aantal nodes:** 1
- **Aantal datanodes:** 1
- **Aantal actieve primaire shards:** 32
- **Aantal actieve shards in totaal:** 32
- **Aantal unassigned shards:** 2  
(2 onverdeelde replica shards van `books_dataset` index)

## 🟡 Betekenis van Status `yellow`

De status `yellow` betekent:

- **Alle primaire shards zijn beschikbaar en actief.**
  - De data is volledig leesbaar en toegankelijk.
- **Niet alle replica shards zijn beschikbaar.**
  - Er zijn replica's die niet kunnen worden toegewezen, meestal omdat er **maar één node** actief is.

Dit is **normaal gedrag** voor een **single-node cluster**:

- Elasticsearch kan replica's niet toewijzen aan dezelfde node als de primaire shard.
- Dit voorkomt dat een fout in die ene node ook meteen de replica's uitschakelt, wat geen echte failover zou zijn.



## Details uit *cat/indices* en *cat/shards*

- De index `books_dataset` heeft:
  - 1 primaire shard ( `p STARTED` ) – Actief en gezond.
  - 2 replica shards ( `r UNASSIGNED` ) – Niet toegewezen omdat er maar 1 node is.

Alle andere interne indices hebben geen replica's (1 primaire shard, 0 replica's), en staan daarom **op** `green`, wat correct is.

---



## Is dit een probleem?

- Voor een **single-node test- of ontwikkelomgeving** is dit **geen probleem**.
- Voor een **productieomgeving** zou dit wél een risico zijn, omdat je **geen redundantie** hebt.