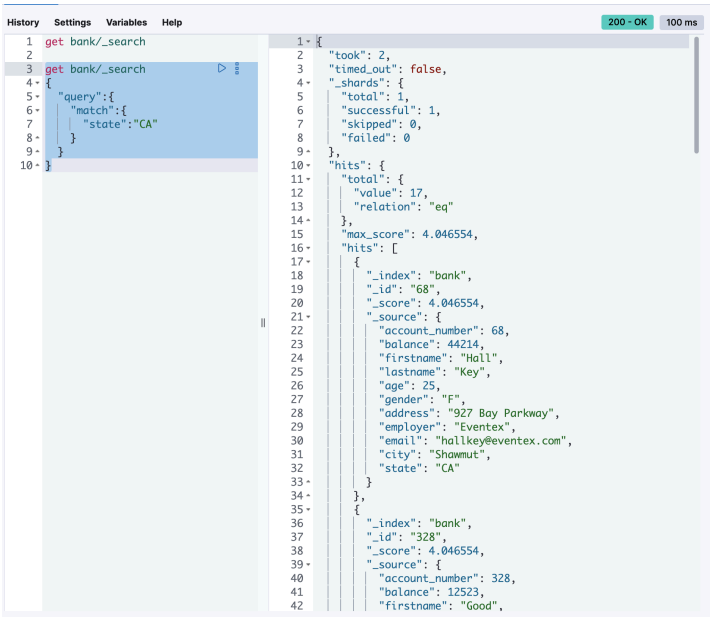
**SIMPLE QUERIES:**

1. Write a query to get bank accounts that are in California:

```
get bank/_search
{
 "query":{
  "match":{
   "state":"CA"
  }
 }
}
```




2.Write a query to get bank accounts that are in California and work in Techade

```
GET bank/_search
{
 "query": {
  "bool": {
   "must": [
    { "match": {"state": "CA"} },
    { "match": {"employer": "Techade"}}
   ]
  }
 }
}
```

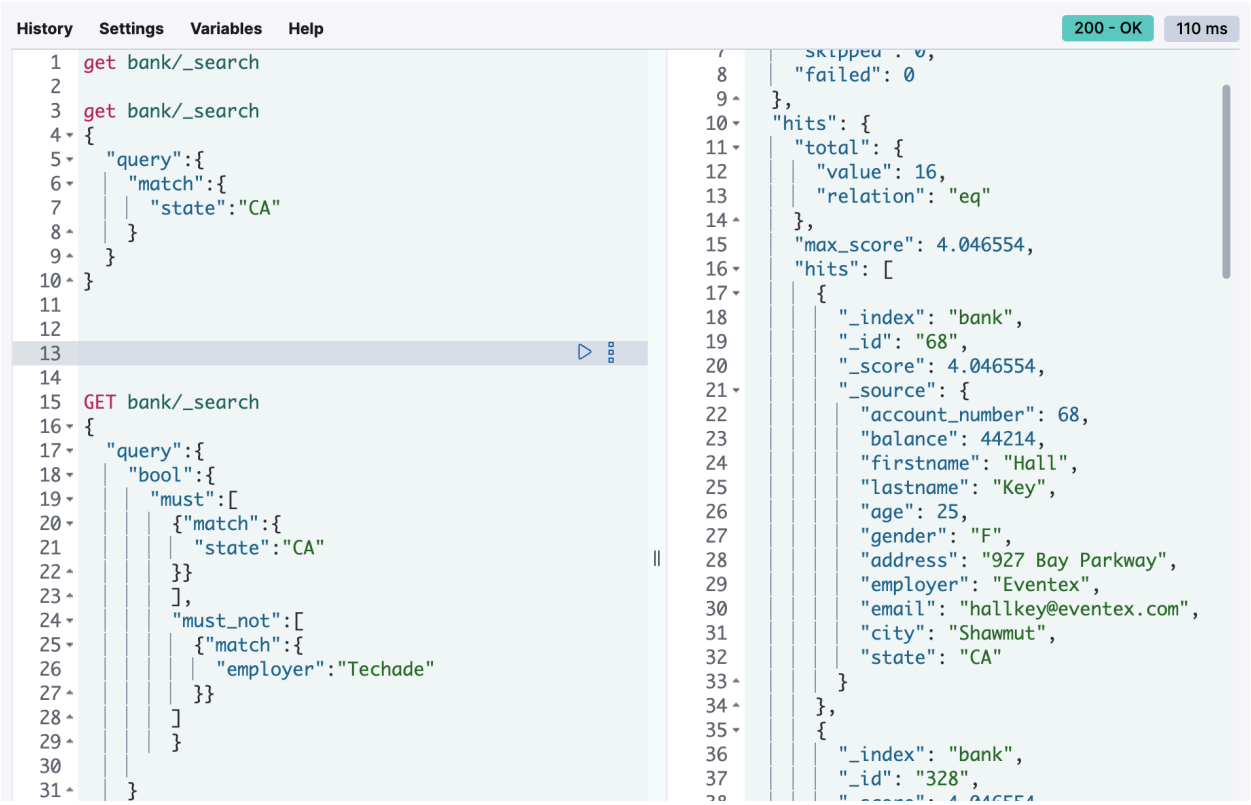
```
get bank/_search

get bank/_search
{
  "query":{
    "match":{
      "state":"CA"
    }
  }
}


# find "Techade" accounts in CA only
GET bank/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": {"state": "CA"} },
        { "match": {"employer": "Techade"}}
      ]
    }
  }
}
```


```
{
  "took": 5,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 10.552503,
    "hits": [
      {
        "_index": "bank",
        "_id": "413",
        "_score": 10.552503,
        "_source": {
          "account_number": 413,
          "balance": 15631,
          "firstname": "Pugh",
          "lastname": "Hamilton",
          "age": 39,
          "gender": "F",
          "address": "124 Euclid Avenue",
          "employer": "Techade",
          "email": "pughhamilton@techade
            .com",
          "city": "Beaulieu",
          "state": "CA"
        }
      }
    ]
  }
}
```


3. Get bank details in California and not employed in Techade

```
GET bank/_search
{
 "query":{
  "bool":{
   "must":[
    {"match":{
     "state":"CA"
    }}
    ],
    "must_not":[
     {"match":{
      "employer":"Techade"
```

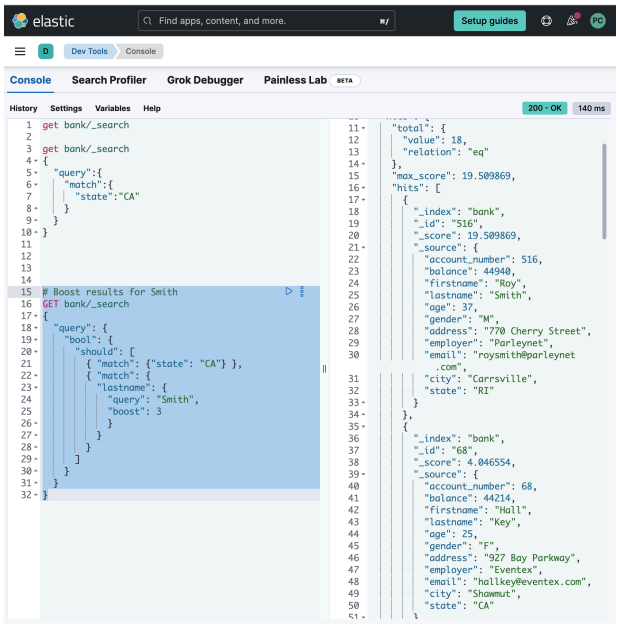
```
      }}
    ]
    }

  }
}
```




4. Writing a query to use the “boost” command to give certain keywords more importance- especially used in search engines:
Query to get results for Smith who work in California but the last name Smith takes more importance over Califronia. Therefore, the output contains “Smith’s” who do not work in California as well.

```
# Boost results for Smith
GET bank/_search
{
  "query": {
    "bool": {
      "should": [
        { "match": {"state": "CA"} },
```

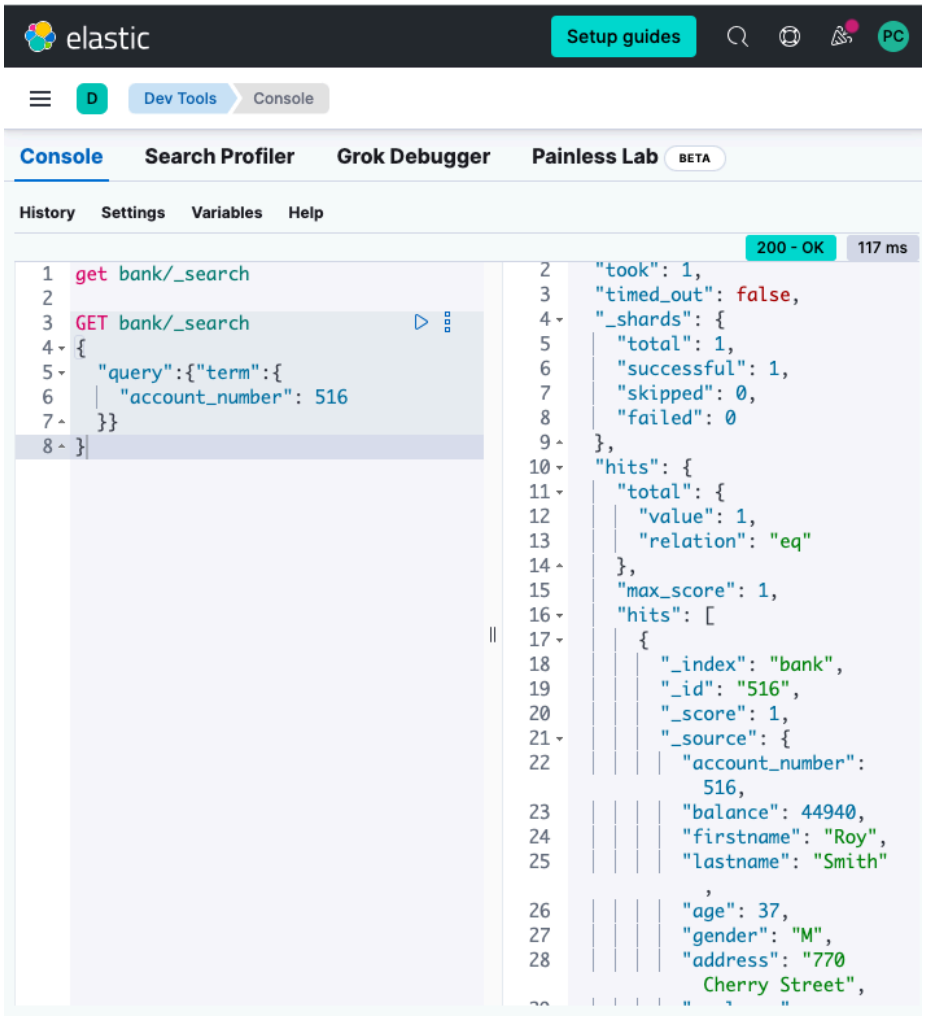
```
        { "match": {
          "lastname": {
            "query": "Smith",
            "boost": 3
            }
          }
        }
      ]
    }
  }
}
```




5. Writing a query using term

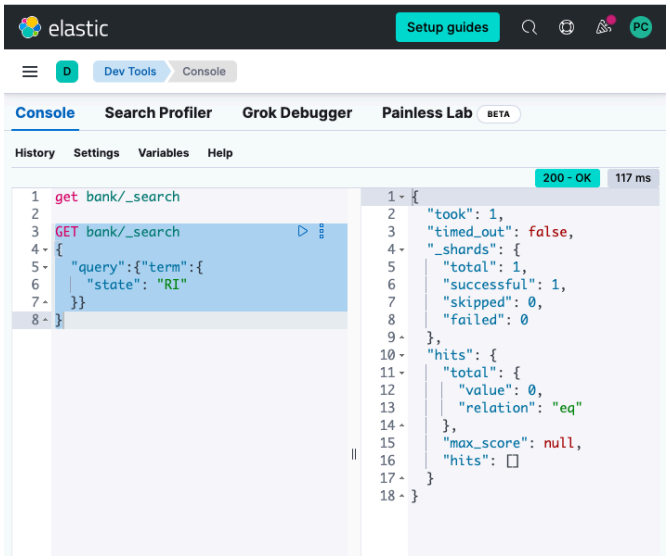
“Term” is used to look up numeric values and keywords

Text does not do well with term field as see below:

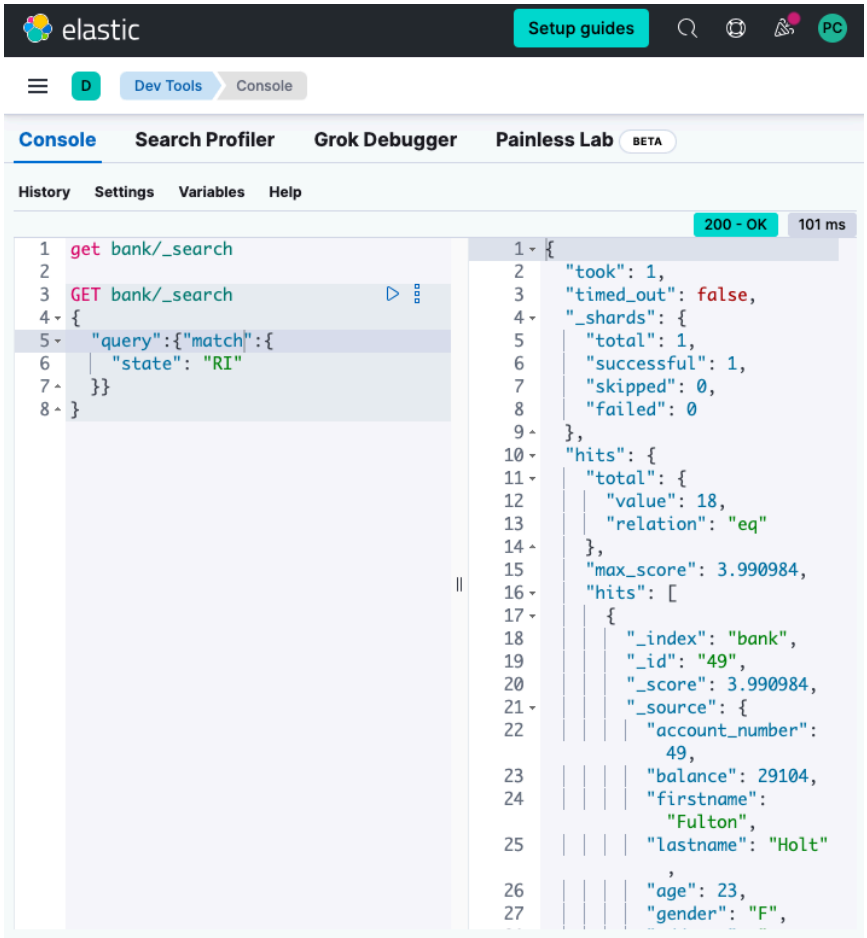



**While using term with text: This query returns no documents.**

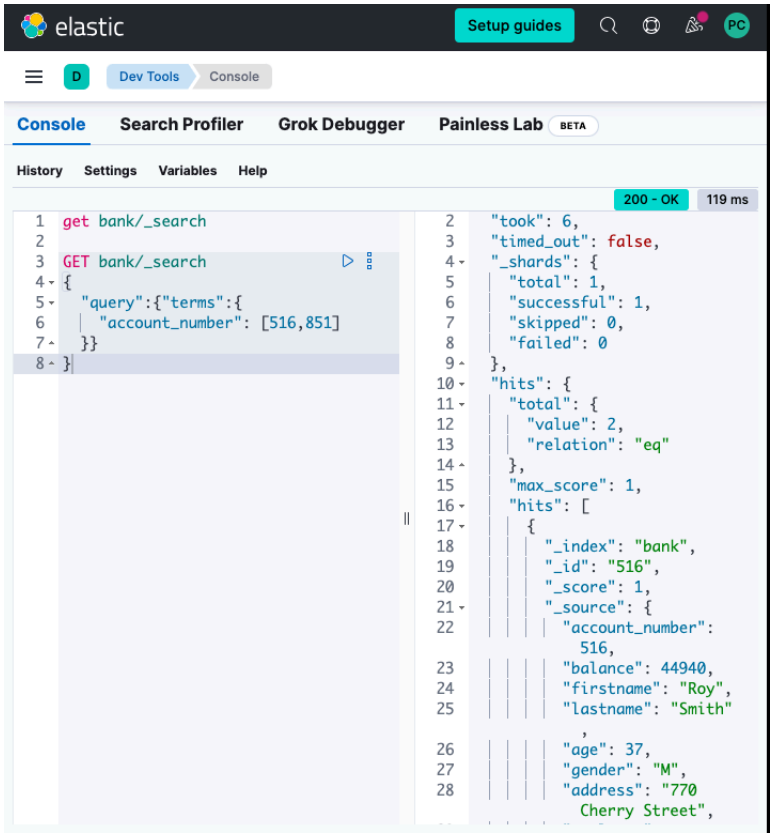
```
GET bank/_search
{
  "query":{"term":{
    "state": "RI"
  }}
}
```




**But, when match is used it shows 18 documents.**



6. Query to retrieve documents using an  array to get multiple results



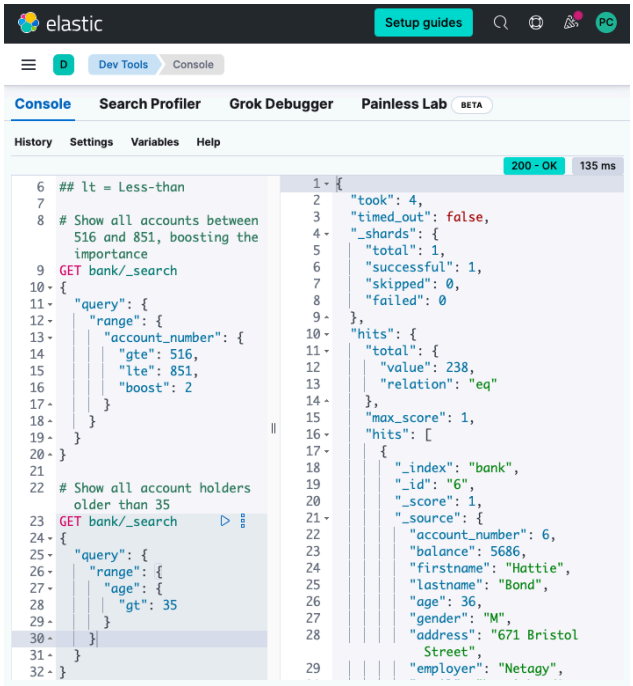

7. Query using gte, gt, lte and lt -

This query searches for documents where the account number is in the range of 516 and 851. Boost is used to increased the relevance score of the documents that match the condition by 2, which makes in relevant in the search results.

```
GET bank/_search
{
  "query": {
    "range": {
      "account_number": {
        "gte": 516,
        "lte": 851,
        "boost": 2
      }
    }
  }
}
```

8. Query to show all users above the age of 35

```
GET bank/_search
{
 "query": {
  "range": {
   "age": {
    "gt": 35
   }
  }
 }
}
```




**Tokenizer:**

- Analyzes a query
- Handles a query

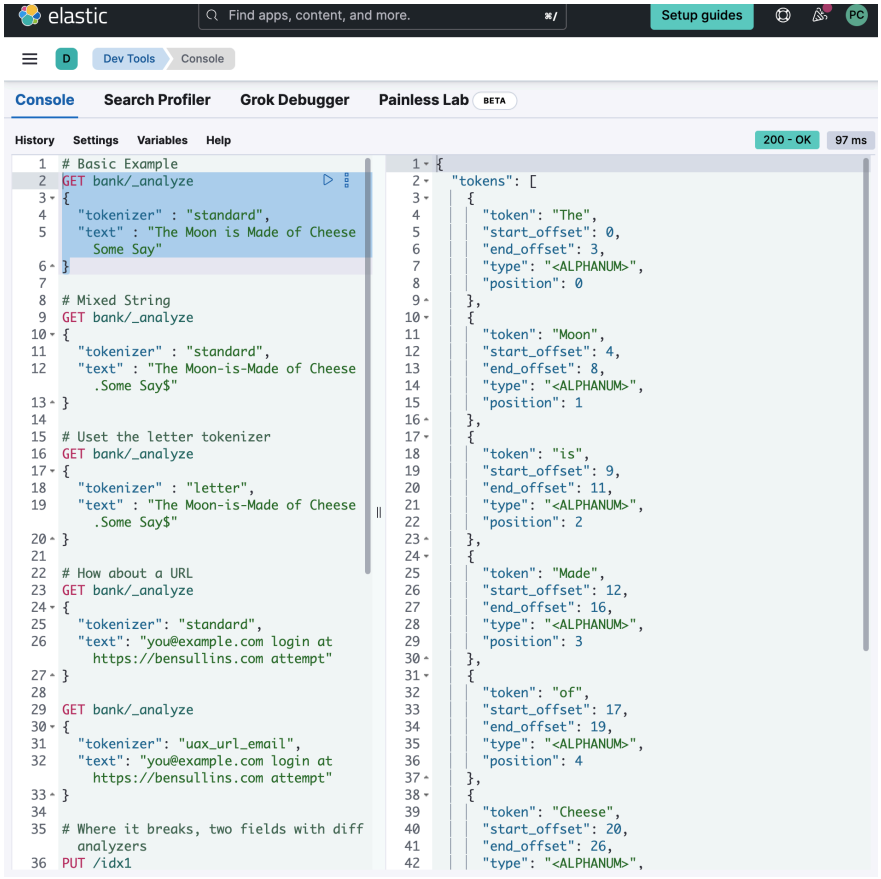
Here, I have used different tokenizers to see how it works;

When “standard” tokenizer is used for the text “The Moon is Made of Cheese Some Say”:

Every word is a token and is of the type “Alphanumeric”

```
GET bank/_analyze
{
 "tokenizer" : "standard",
```

"text" : "The Moon is Made of Cheese Some Say"
}

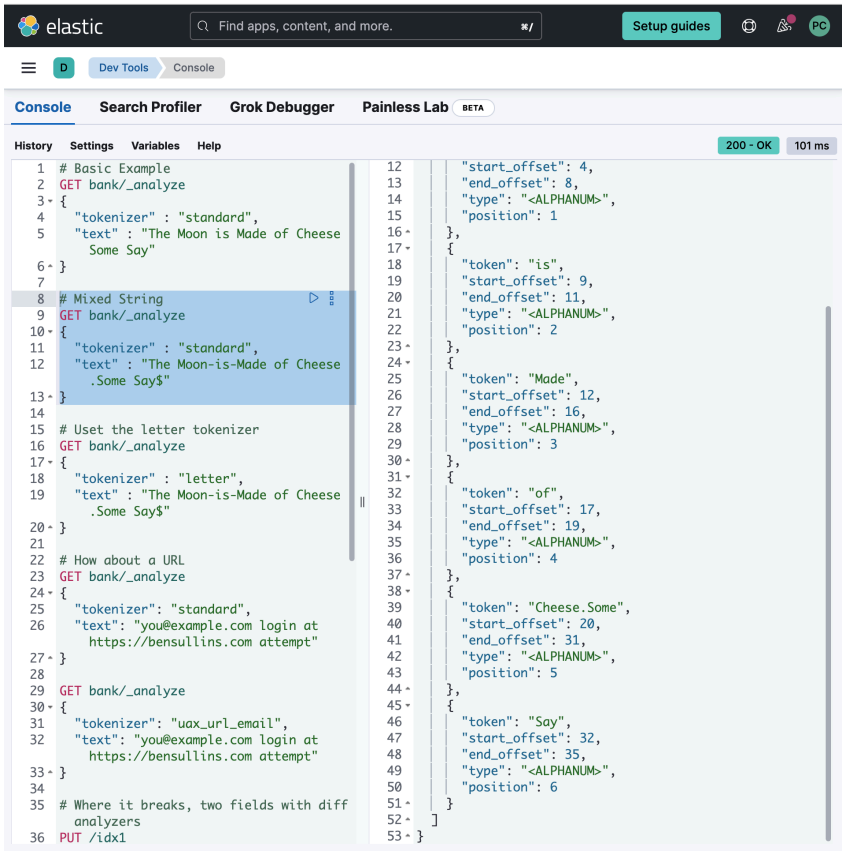



A “standard” tokenizer is used with the query “ The Moon-is-Made of Cheese.Some Say$"

Every word is a token except ofr “Cheese.Some” is considered as one token and the “$” is removed.

Type is Alphanumeric

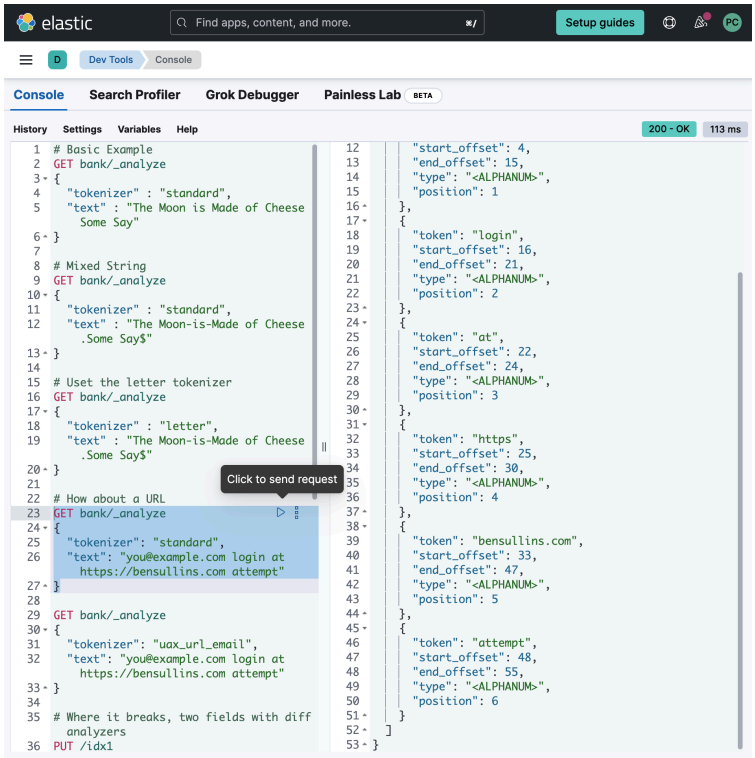
# Mixed String
GET bank/_analyze
{
"tokenizer" : "standard",
"text" : "The Moon-is-Made of Cheese.Some Say$"
}




The tokenizer “letter” is used with the text “The Moon-is-Made of Cheese.Some Say$”.
Every word is a token and the” $” and “.” is removed.
The type is “word”

To parse a url and the tokenizer used is “standard”
Every word is a token and is of type alphanumeric

```
GET bank/_analyze
{
  "tokenizer": "standard",
  "text": "you@example.com login at https://bensullins.com attempt"
}
```




```
# Basic Example
GET bank/_analyze
{
  "tokenizer" : "standard",
  "text" : "The Moon is Made of Cheese
    Some Say"
}

# Mixed String
GET bank/_analyze
{
  "tokenizer" : "standard",
  "text" : "The Moon-is-Made of Cheese
    .Some Say$"
}

# Uset the letter tokenizer
GET bank/_analyze
{
  "tokenizer" : "letter",
  "text" : "The Moon-is-Made of Cheese
    .Some Say$"
}

# How about a URL
GET bank/_analyze
{
  "tokenizer": "standard",
  "text": "you@example.com login at
    https://bensullins.com attempt"
}

GET bank/_analyze
{
  "tokenizer": "uax_url_email",
  "text": "you@example.com login at
    https://bensullins.com attempt"
}

# Where it breaks, two fields with diff
  analyzers
PUT /idx1
```

```
    "start_offset": 4,
    "end_offset": 15,
    "type": "<ALPHANUM>",
    "position": 1
  },
  {
    "token": "login",
    "start_offset": 16,
    "end_offset": 21,
    "type": "<ALPHANUM>",
    "position": 2
  },
  {
    "token": "at",
    "start_offset": 22,
    "end_offset": 24,
    "type": "<ALPHANUM>",
    "position": 3
  },
  {
    "token": "https",
    "start_offset": 25,
    "end_offset": 30,
    "type": "<ALPHANUM>",
    "position": 4
  },
  {
    "token": "bensullins.com",
    "start_offset": 33,
    "end_offset": 47,
    "type": "<ALPHANUM>",
    "position": 5
  },
  {
    "token": "attempt",
    "start_offset": 48,
    "end_offset": 55,
    "type": "<ALPHANUM>",
    "position": 6
  }
]
}
```

While parsing a url with the tokezier “uax_url_email”

The email id is considered as a token and is of type “EMAIL”

URL is considered as type URL and the rest is alphanumeric

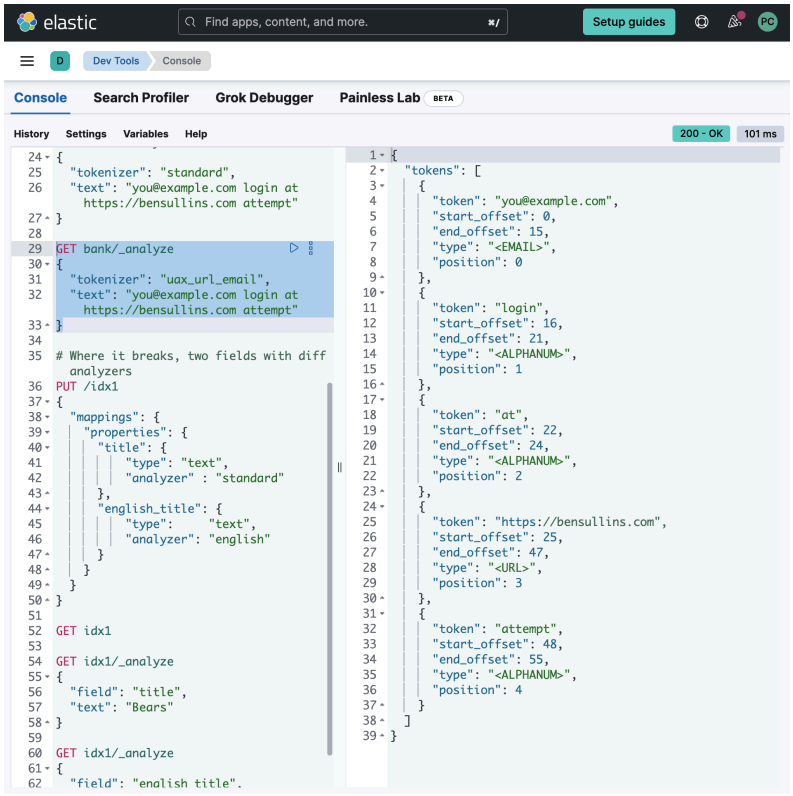
GET bank/_analyze

{

"tokenizer": "uax_url_email",

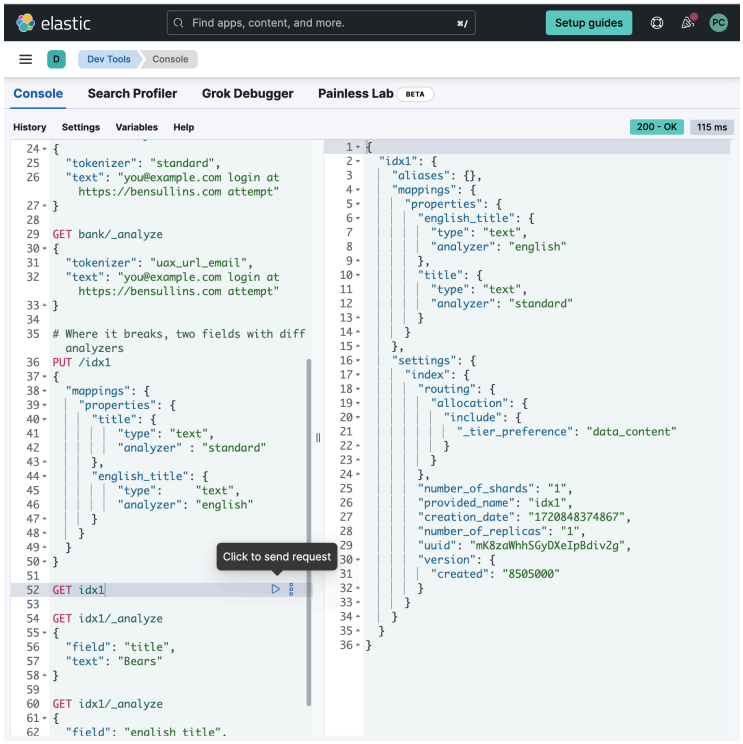
"text": "you@example.com login at https://bensullins.com attempt"

}




Creating a new index:

```
PUT /idx1
{
  "mappings": {
    "properties": {
      "title": {
          "type": "text",
          "analyzer" : "standard"
      },
      "english_title": {
          "type":     "text",
          "analyzer": "english"
      }
    }
  }
}
```




Send  data to the index to see the handling of data

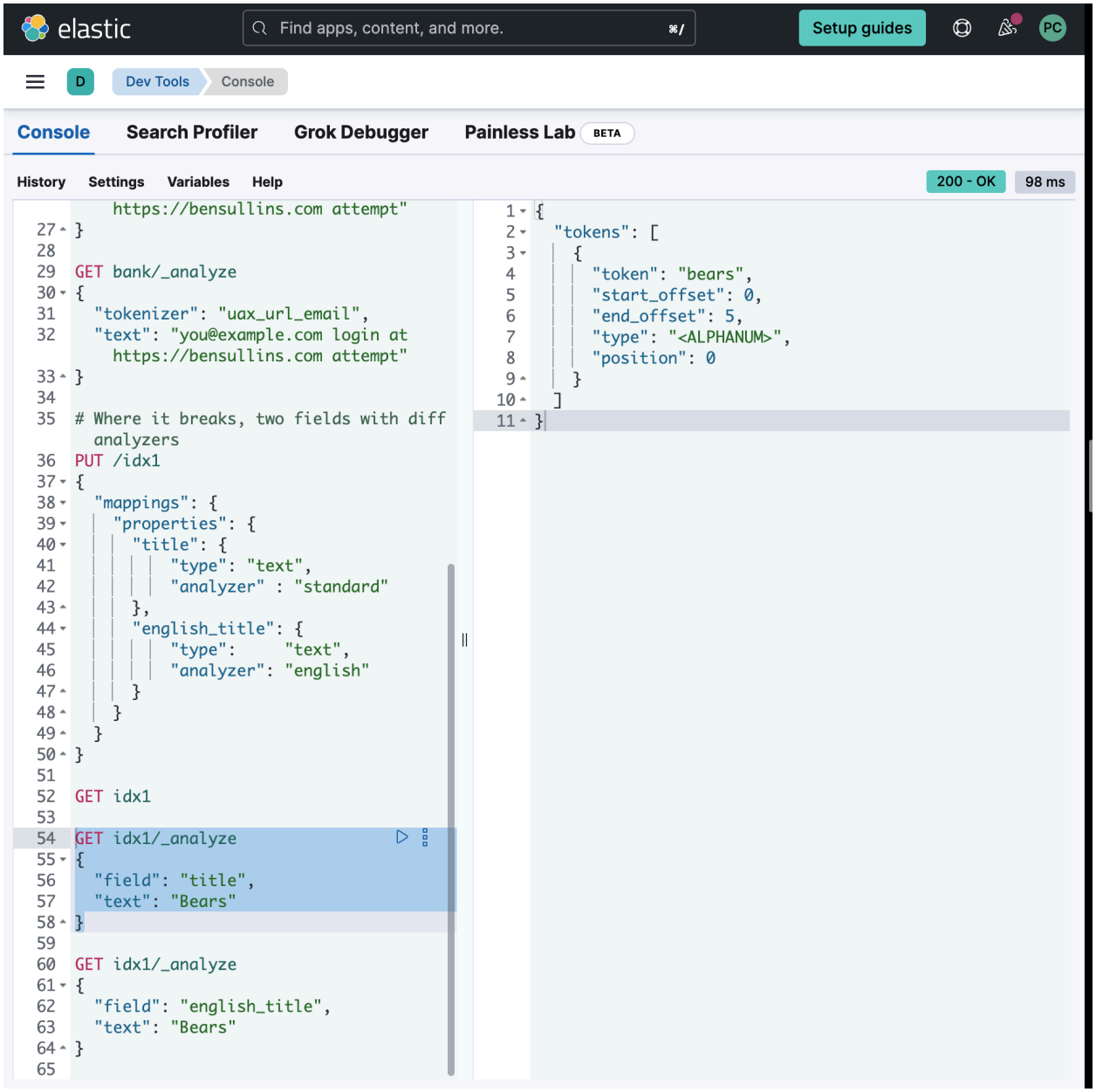
GET idx1/_analyze

{

  "field": "title",

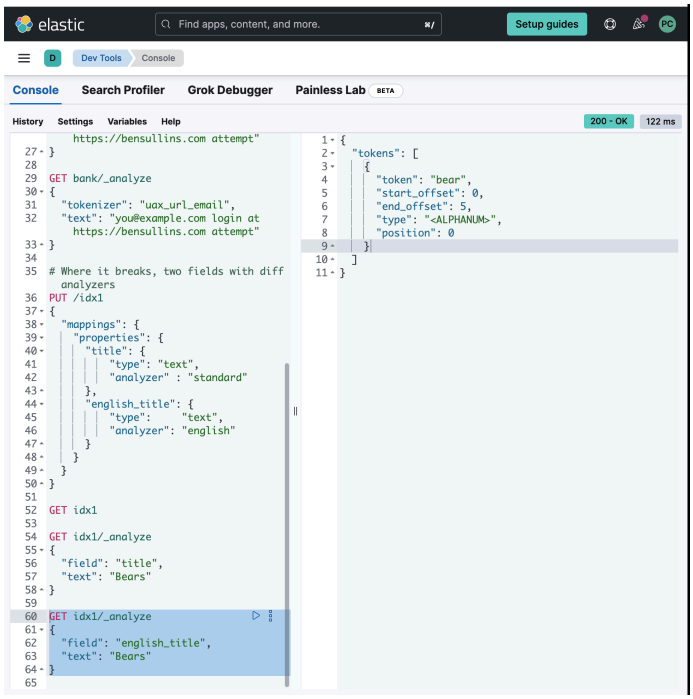
  "text": "Bears"

It is going to use standard analyzer and the result contains a token “bear” but the capital is dropped




While using english_title it dropped the capital letter and the plural fom so the token is “bear” and not Bears

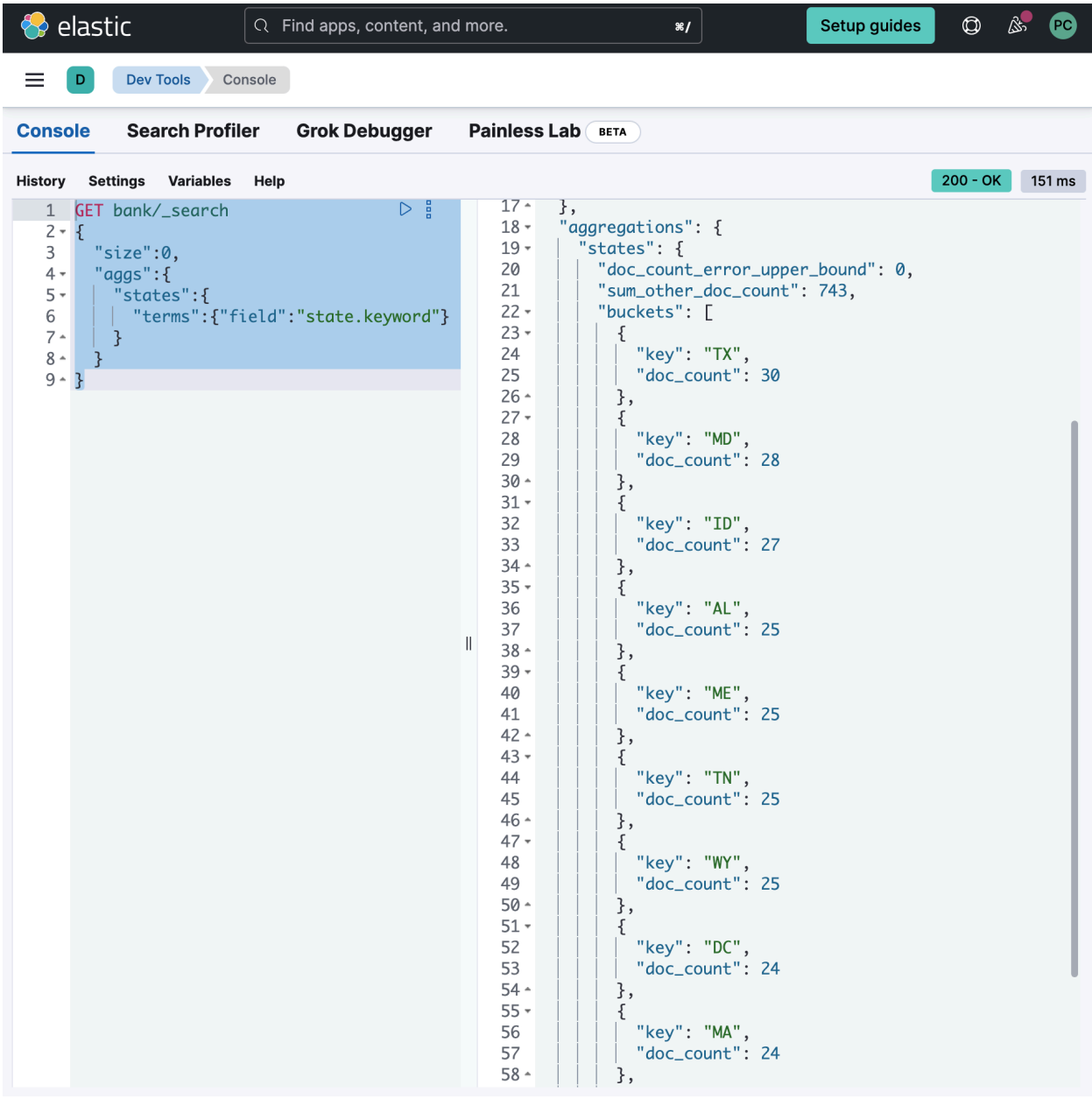
GET idx1/_analyze
{
"field": "english_title",
"text": "Bears"
}




## ANALYSIS: USING AGGREGATE FUNCTIONS

1. See state count in aggregation as number of documents.

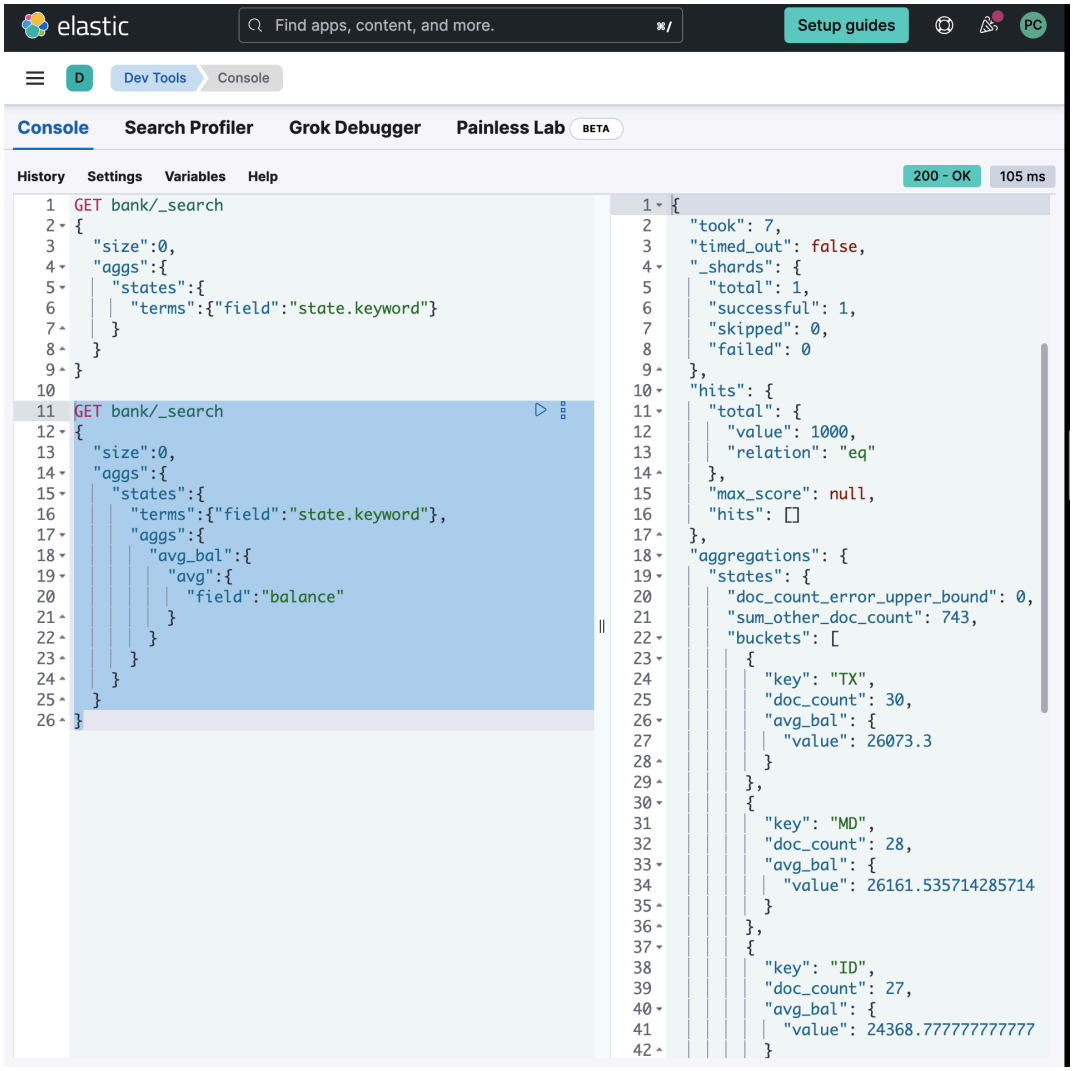
```
GET bank/_search
{
  "size":0,
  "aggs":{
    "states":{
      "terms":{"field":"state.keyword"}
    }
  }
}
```




2. Average Balance using nested aggregations - Query to get average balance of accounts in states and also mention the number of documents in each state.

```
GET bank/_search
{
  "size":0,
  "aggs":{
    "states":{
      "terms":{"field":"state.keyword"},
      "aggs":{
        "avg_bal":{
```

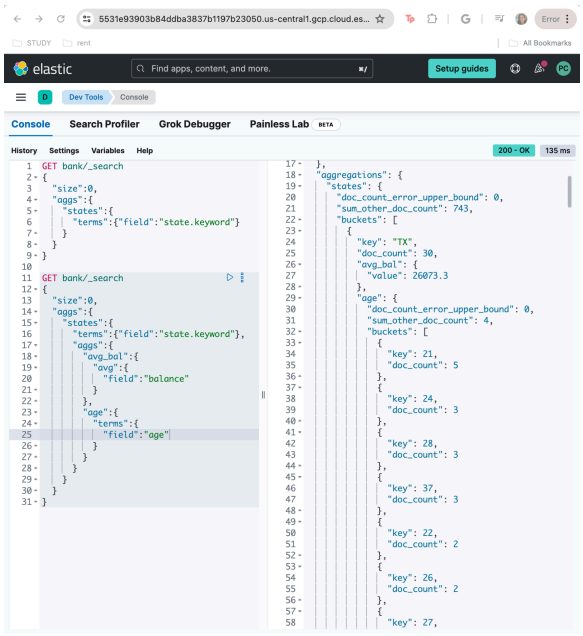
```
      "avg":{
        "field":"balance"
      }
    }
  }
 }
}
```




3.Add a further aggregation to get the number of accounts in each state, avg account balance and for every age- how many documents exist?

```
GET bank/_search
{
 "size":0,
 "aggs":{
  "states":{
```

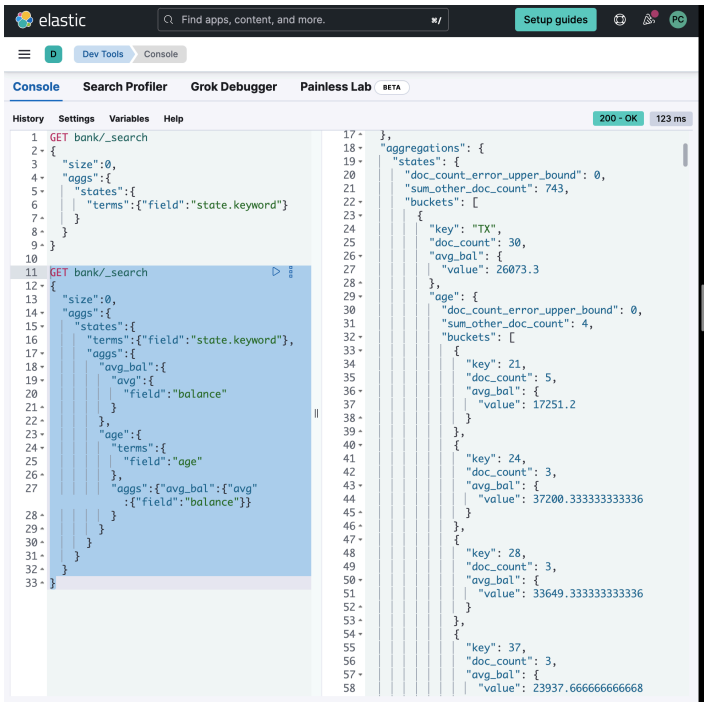
```
    "terms":{"field":"state.keyword"},
    "aggs":{
     "avg_bal":{
       "avg":{
         "field":"balance"
       }
     },
     "age":{
       "terms":{
         "field":"age"
       }
     }
    }
   }
  }
}
```




4.Creating a drill down with calculating avg_bal of accounts for every age with all the above mentioned fields.

```
GET bank/_search
{
 "size":0,
 "aggs":{
  "states":{
```

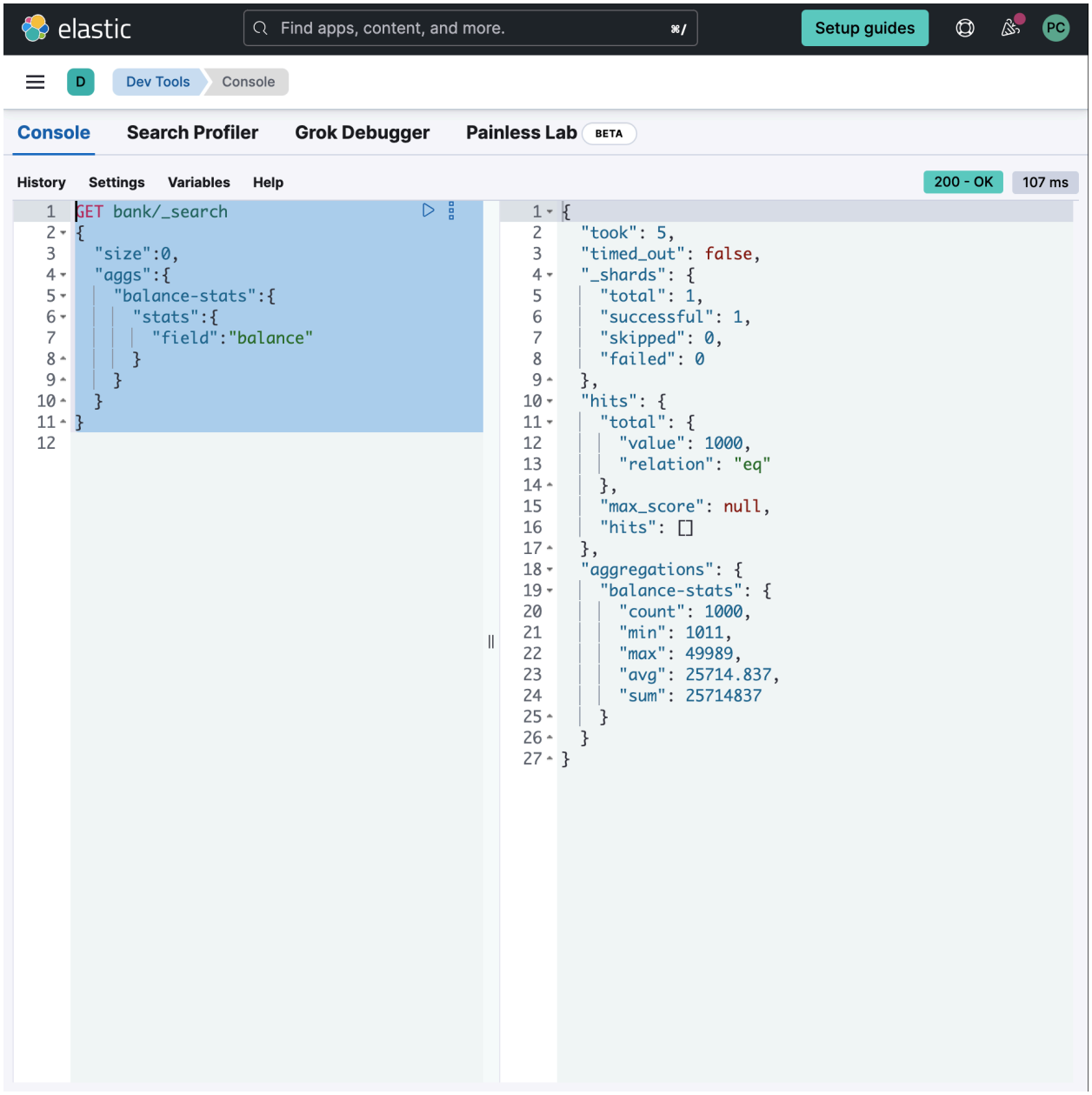
```
    "terms":{"field":"state.keyword"},
    "aggs":{
     "avg_bal":{
      "avg":{
       "field":"balance"
      }
     },
     "age":{
      "terms":{
       "field":"age"
      },
      "aggs":{"avg_bal":{"avg":{"field":"balance"}}}
      }
     }
    }
   }
}
```




3. Get stats on the dataset present in the cluster

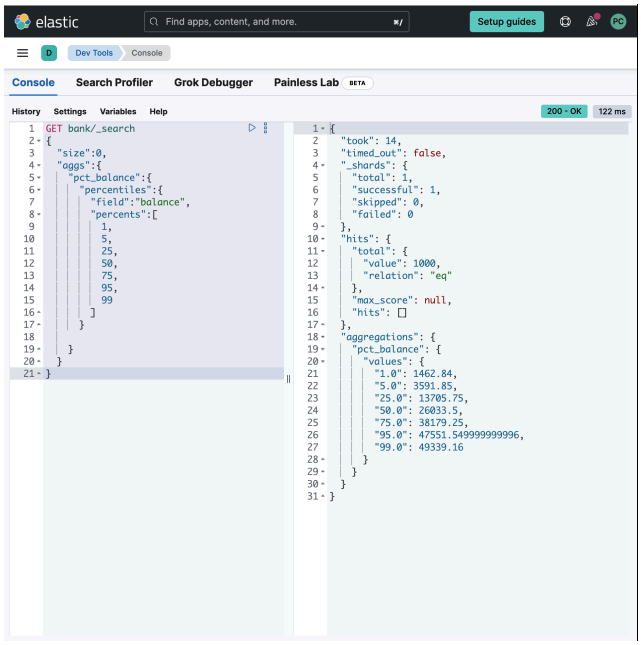
```
GET bank/_search
{
 "size":0,
 "aggs":{
```

```
  "balance-stats":{
    "stats":{
      "field":"balance"
    }
  }
 }
}
```



4.What number of accounts are percentiles? This query gives us the distribution

```
{
 "size":0,
 "aggs":{
  "pct_balance":{
   "percentiles":{
    "field":"balance",
    "percents":[
     1,
     5,
     25,
     50,
     75,
     95,
     99
    ]
   }

  }
 }
}
```

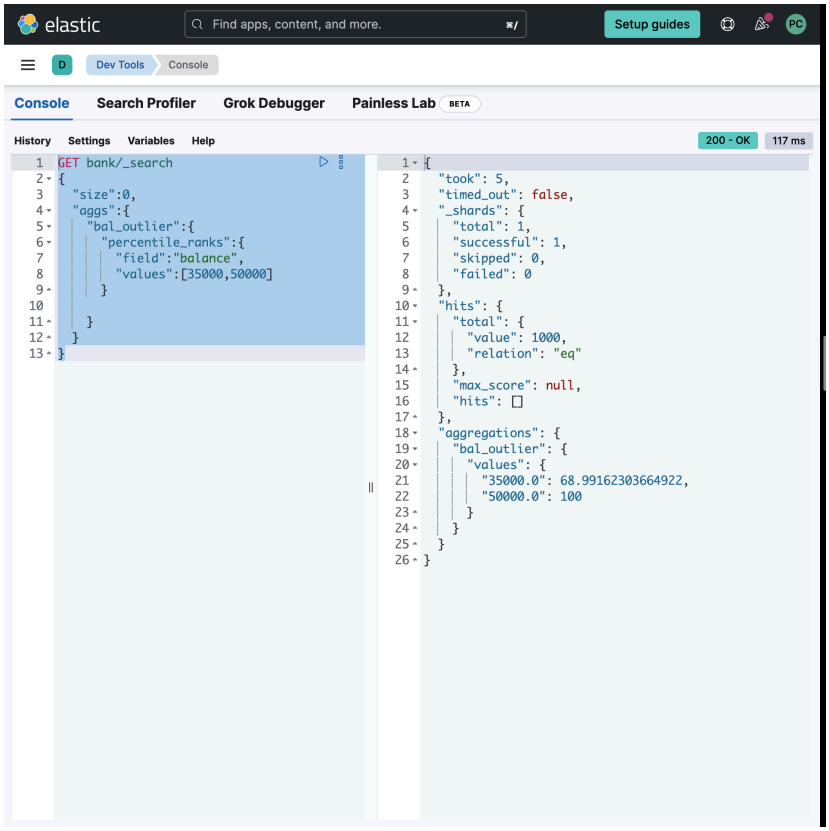



4. What percentile of accounts contain a balance of 35,000 to 50,000. See if there are any outliers present and can be used for fraud detection .

Accounts with 35,000 is in 68 percentile and 50,000 are in the 100 percentile

GET bank/_search

```
{
 "size":0,
 "aggs":{
  "bal_outlier":{
   "percentile_ranks":{
    "field":"balance",
    "values":[35000,50000]
   }

  }
 }
}
```




5. This query shows the distribution of accounts and its balance

```
GET bank/_search
{
 "size":0,
 "aggs":{
  "bals":{
   "histogram":{
    "field":"balance",
```

```
        "interval":500
      }

    }
  }
}
```