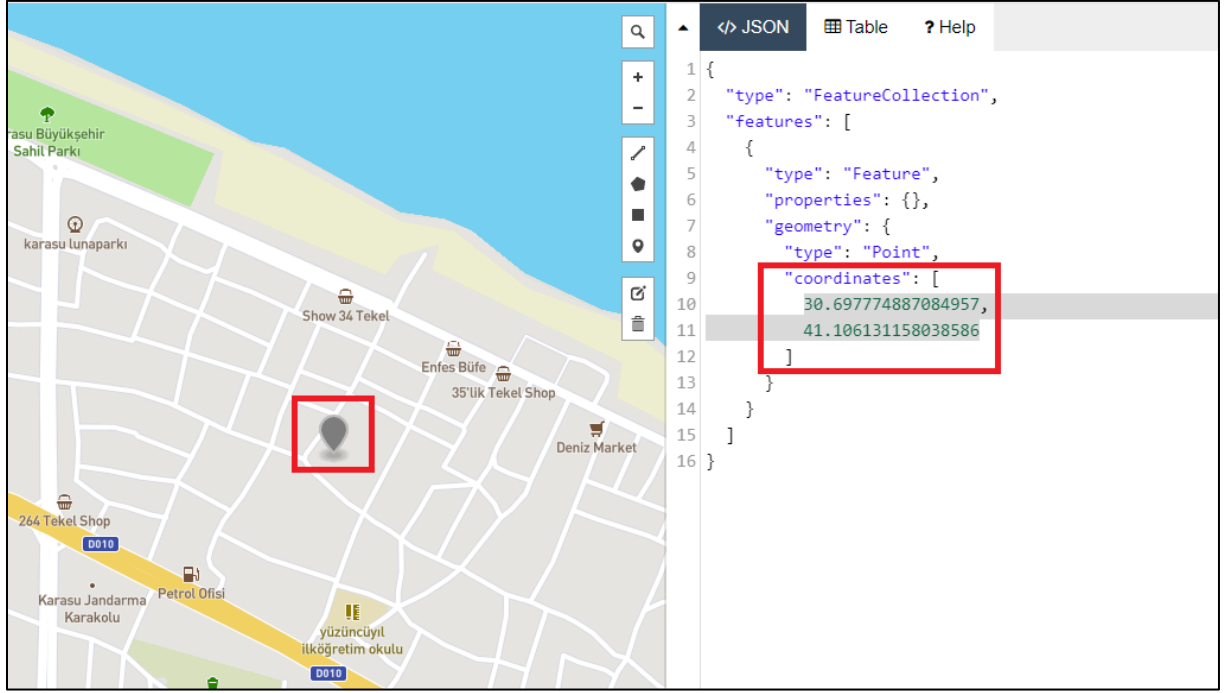


COĞRAFI UYGULAMALAR

Coğrafik sorgular, e-ticaret, kargo takip, bankalar arası iletişim gibi birçok sektörde kullanılır. Çünkü fiziksel bir yerin adreslenebilmesi için latitud(enlem) ve longitud(boylam) denen coğrafik bilgiler gereklidir bu bilgilerin bulunmasında coğrafik sorgular rol oynar. Bu bilgilere göre bırakılan pointler lad ve lod denilen değerlerden oluşur ve bu değerler hangi sisteme verilirse doğrudan bu lokasyona ulaşım sağlanır. GPSlerde de bu mantıkla kullanılabilir.

Aşağıdaki örnekte seçili nokta için yukarıda bahsedilen lad ve lod değerleri görülmektedir.



Neden Coğrafik Sorgulara Gerek Duyulur?

1. Bankacılık

Yukarıda da bahsedildiği gibi bankacılık sektöründe de coğrafik sorgulardan yararlanılır. Örneğin iki farklı noktada bir bankanın iki şubesi bulunuyor. Bu iki şube arasında oturan kişinin hangi şube müşterisi olacağına karar vermek için iki nokta arasındaki uzaklık bilgisine bakılır ve böylece müşteri direkt olarak içerisinde bulunduğu semte ait şubeye yönlendirilmek yerine farklı semtte bulunan fakat belki de kendisine daha yakın bir şubeye yönlendirilerek işlemlerini daha rahat yapması sağlanabiliyor. İşte bu örnekte bahsedilen iki nokta arasındaki uzaklık hesaplamasına göre müşteri yönlendirme işlemi coğrafik sorgular ile yapılıyor ve müşteri memnuniyetinin maksimum seviyede olmasına fayda sağlıyor.

2. E-Ticaret

E-ticaret sektöründe de aynı şekilde coğrafik sorgulardan yararlanılarak müşteri memnuniyeti, firmanın karı gibi parametrelerde artış sağlanabilir. Örneğin online bir yemek şirketi sadece Üsküdar içerisindeki müşterilere hizmet sunabileceğini söylüyor ve semt dışına servis yapmıyor. Örneğin bulunduğu noktadan 5km ötesinde Üsküdar sınırları içerisindeki müşteriye servise çıkarken, 500 metre yakınındaki müşterilere Kadıköy sınırları içerisinde olduğu için servis yapmıyor ve müşteri kaybediyor. İşte bu gibi durumlarda coğrafik sorgular kullanılarak mesafe hesaplamaları ile müşteri memnuniyeti ve şirketin karı gözetilebilir

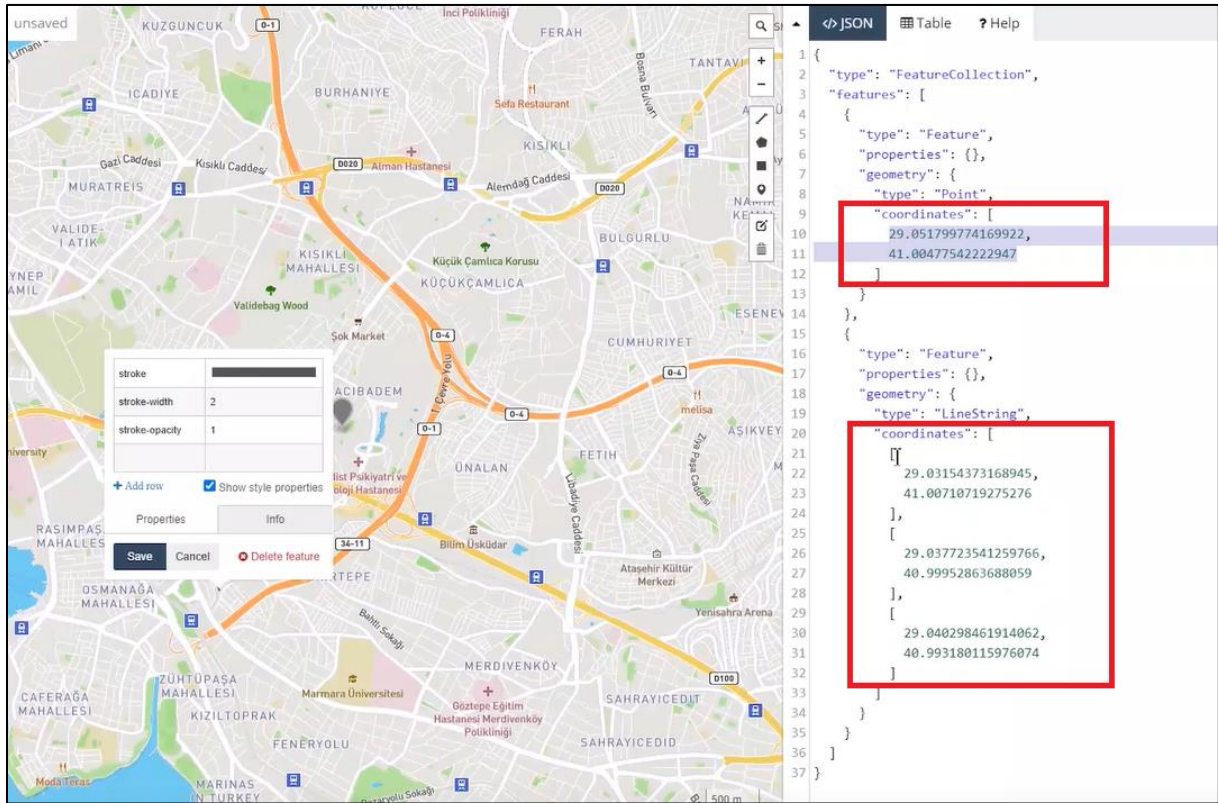
3. Kargo-Taşıma

Kargo şirketleri için de coğrafik sorguların kullanılması kritik bir konudur. Örneğin, bir kurye kargo şubesinden çıkarak kargo dağıtımını yapmaya başladı. Eğer bu dağıtım için coğrafik sorgular kullanılırsa daha kısa sürede daha karlı bir şekilde dağıtım işlemlerinin bitmesi sağlanabilir. Aksi halde işlerin aksamaması, kuryenin kullandığı aracın yakıt ihtiyacından dolayı şirket karı gibi parametrelerde olumsuz etki gözlenebilir. Bu sebepten ötürü kargoculuk-taşıma sektörlerinde de coğrafik sorgular kritik rol oynar.

Coğrafi Uygulamalar Neden Big Data Konusu?

Bu konunun Big Data tarafındaki yerinden bahsedecek olursak buradaki bazı veriler **semi-structured** yapıda bulunmaktadır. Bilinen ilişkisel veri tabanlarında bu bilgiler tutulmuyor ya da tutulsa bile kompleks işlemler gerektirebiliyor ve performans açısından kullanışlı olmayabiliyor. Bu tip durumlara NoSql veri tabanlarından olan MongoDB ve Elasticsearch rahatlıkla çözüm sağlayabiliyor.

Aşağıda coğrafik sorgularda **semi-structured** yapılara ait bir örnek bulunmaktadır.



Görüldüğü gibi 1. ve 2. nokta için coordinates arrayleri farklı eleman sayılarına sahip bu da verilerin **semi-structured** yapıda olabileceğini gösteriyor.

SİTEMİ BUL UYGULAMASI

Kullanılan Teknolojiler:

- **Elasticsearch:** NoSql veri tabanı
- **Kibana:** Elasticsearch ile çalışan bir görselleştirme toolu
- **Postman:** http requestleri atmayı sağlayan rest client
- **DigitalOcean:** Cloude'da server oluşturmayı sağlıyor
- **PuTTY:** Uzaktaki makine bağlanmak için kullanılan program
- **MobaXterm:** Uzaktaki Makine bağlanmak için bir diğer kullanılan program

Elimizde İstanbul'daki bazı apartmanların lokasyon bilgilerini veren bir json data bulunuyor. Bu data kullanarak siteler arasındaki mesafe, bulunan konuma en yakın sitenin bulunması ya da çizilen kapalı alan içerisinde kaç tane site bulunuyor gibi işlemleri gerçekleştirdik.

Bu uygulamada iki veritabanı arasından Elasticsearch'ü tercih ettik. Elasticsearch'te lad ve lod noktalarını Kibana üzerinde görselleştirebiliyoruz. Böylece dataları daha anlamlı halde sunabiliyoruz. Aynı şekilde coğrafik sorguları atabilmek için MongoDB'de kullanılabilir fakat görselleştirme yapabilmek için bir servise ihtiyaç duyuluyor ve bu servise bir frontend hazırlamak gerekiyor. Bu sebepten ötürü elasticsearch kullanmayı tercih ettik.

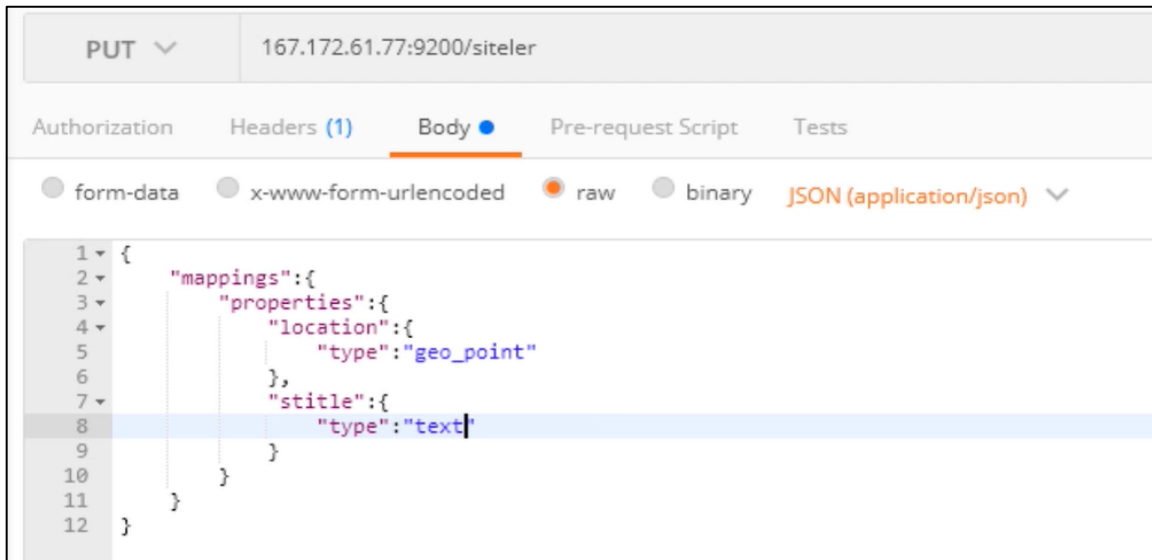
Bunun yanı sıra Elasticsearch'te mapping işlemleri, veri gönderme ve veriler üzerinde gerekli işlemlerin yapılmasını da gerçekleştirdik.

1.Adım: Data Elasticsearch'e atılır.

Elimizde bulunan json datasını elasticsearch'e atmamız gerekiyor.

1.1) Mapping İşlemi

İlk olarak elasticsearch içerisinde "siteler" isminde bir index oluşturduk. Elasticsearchteki indeksler RDBMS'deki veri tabanları gibi düşünülebilir. Oluşturulan bu indekste mapping işlemi yapıldı. Mapping sayesinde yüklenecek olan fieldlerin tipleri belirtiliyor. Coğrafik sorguların hangi fieldlerde yapılacağı belirtiliyor ki queryler performanslı çalışsın.



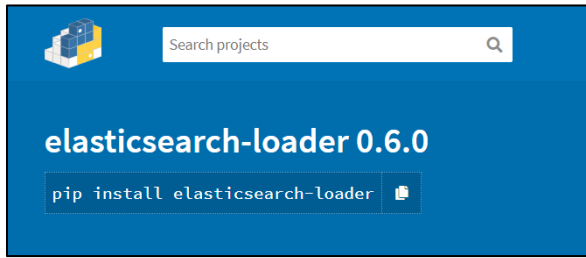
Elimizdeki apartman verisinde “location” ve “stitle” adında 2 adet field bulunuyor. Bu fieldler mapping işleminde tanımlandı ve tipleri belirtildi. Location üzerinde coğrafi sorgular yapılacağı için geo_point tipi, stitle ise string tipinde verilere sahip olduğu için string’e karşılık gelen text tipi yazıldı.



Send denildikten sonra işlem sonucu true döndü ve mapping gerçekleştirildi.

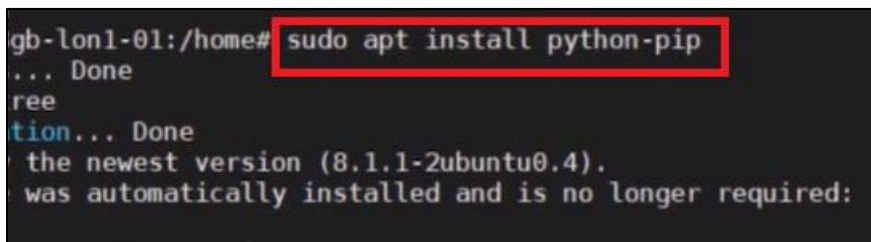
1.2) Elasticsearch-loader Kütüphanesini Kullanarak Json Verisini Elasticserach’e Atma

Verileri elasticsearch’e tek tek manual olarak gönderebilmek mümkün fakat veri sayısı arttıkça bu yöntem kullanılamaz hale gelir. Bir programlama dilinde (java, patyhon gibi) verilerin bulunduğu json

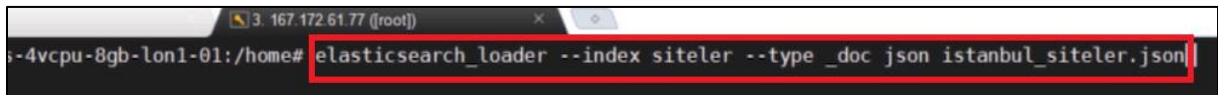


dosyası okunup kod ile gönderilebilir ya da ara bir tool kullanılabilir. Bu noktada open source bir küphane olan elasticsearch-loader devreye girer. Bu kütüphane json, csv tarzı dataları elasticsearch’e kolay bir şekilde göndermeyi sağlıyor. Pip paketi kurulur ve elaticsearch-loader indirilir.

- Önce pip paketi kurulur.



- Sonra elasticsearch-loader indirilir.



Yukarıdaki paket kurulum komutundan sonra apartman listelerinin bulunduğu json dosyası elasticserach’e gönderilmiş olur.

```

1 {
2   "took": 915,
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": "siteler",
19        "_type": "_doc",
20        "_id": "1",
21        "_score": 1,
22        "_source": {
23          "title": "İstanbul Topkapı",
24          "location": {
25            "lat": 41.020845,
26            "lon": 28.901259
27          }
28        }
29      }
30    ]
31  }
32 }

```

Yukarıda görüldüğü gibi 10000 adet datanın hepsi başarılı bir şekilde elasticsearch'e gönderildi.

2.Adım: Data Üzerinde Coğrafi Sorgular Yapılmaya Başlanılır

İşaretlenen bir konumu circle içine alarak o circle'da bulunan tüm sitelerin getirmesi işlemi gerçekleştirilecek.

2.1) Bir Konum İşaretlenerek Enlem ve Boylam Bilgilerini Alma

The screenshot shows a map of Istanbul with a red box highlighting a location in Fikirtepe. The location is marked with a black dot and labeled 'FİKİRTEPE'. The map interface includes various controls like zoom in/out, pan, and a search bar. On the right side, there is a JSON viewer showing the coordinates of the selected location:

```

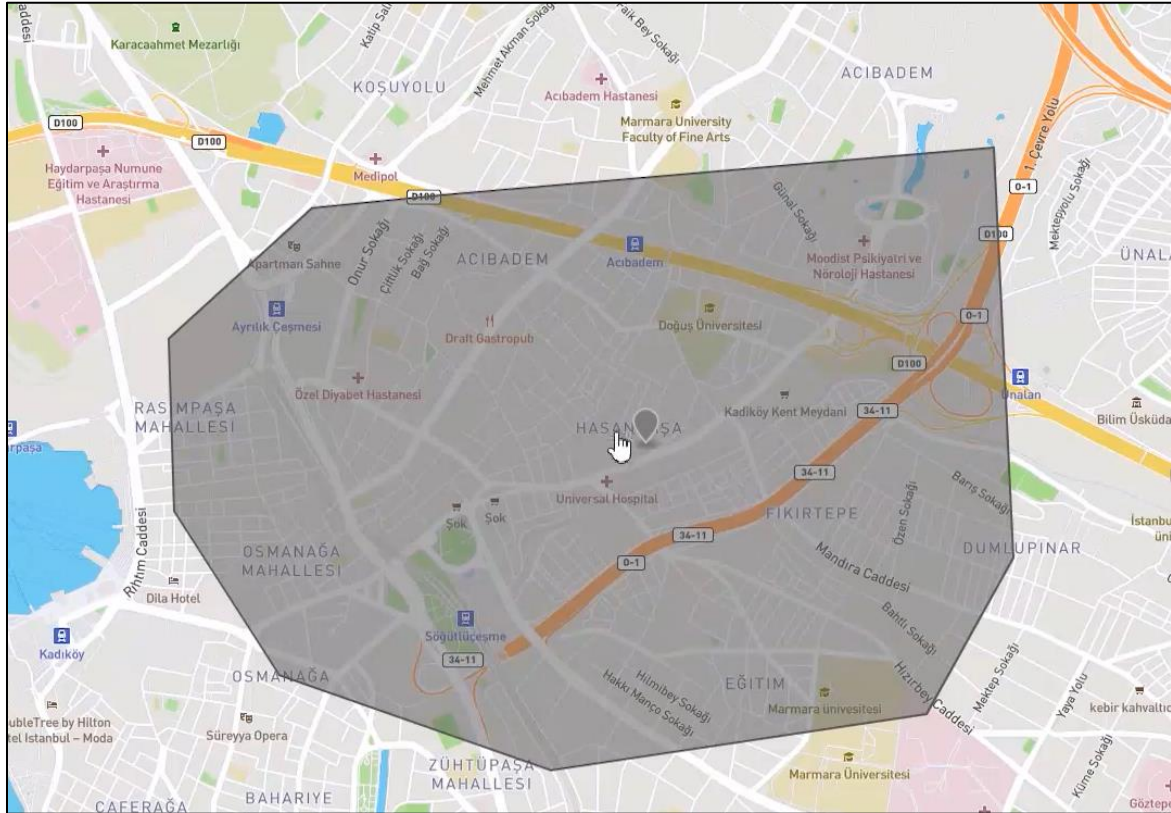
1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {},
7       "geometry": {
8         "type": "Point",
9         "coordinates": [
10          29.04982566833494,
11          40.994929259285605
12        ]
13      }
14    }
15  ]
16 }

```


2.2) Seçili Konuma Göre Belirlenmiş Mesafedeki Diğer Konumların Tespiti

- Önce sorguları yazacağımız bir query field'i oluşturulur. Daha sonra **match_all** komutu ile tüm eşleşen noktaları getirmesini, **distance** ile seçili noktadan 250 metre çapında bir alan çizilmesi sağlanır. **Location** kısmına ise 2.1 maddesinde seçilen konumun enlem ve boylam bilgileri girilir.

```
Authorization Headers (1) Body Pre-request Script Tests
form-data x-www-form-urlencoded raw binary JSON (application/json)
1 {
2   "query":{
3     "bool":{
4       "must":{
5         "match_all":{}
6       },
7       "filter":{
8         "geo_distance":{
9           "distance":"250m",
10          "location":{
11            "lon":29.04982566833496,
12            "lat":40.994929259285605
13          }
14        }
15      }
16    }
17  }
18 }
```



- Atılan bu sorguya göre seçili nokta etrafında 250 metre çapında bir tarama yapıldı ve data setimiz içerisindeki 2 site bilgisi bulundu.

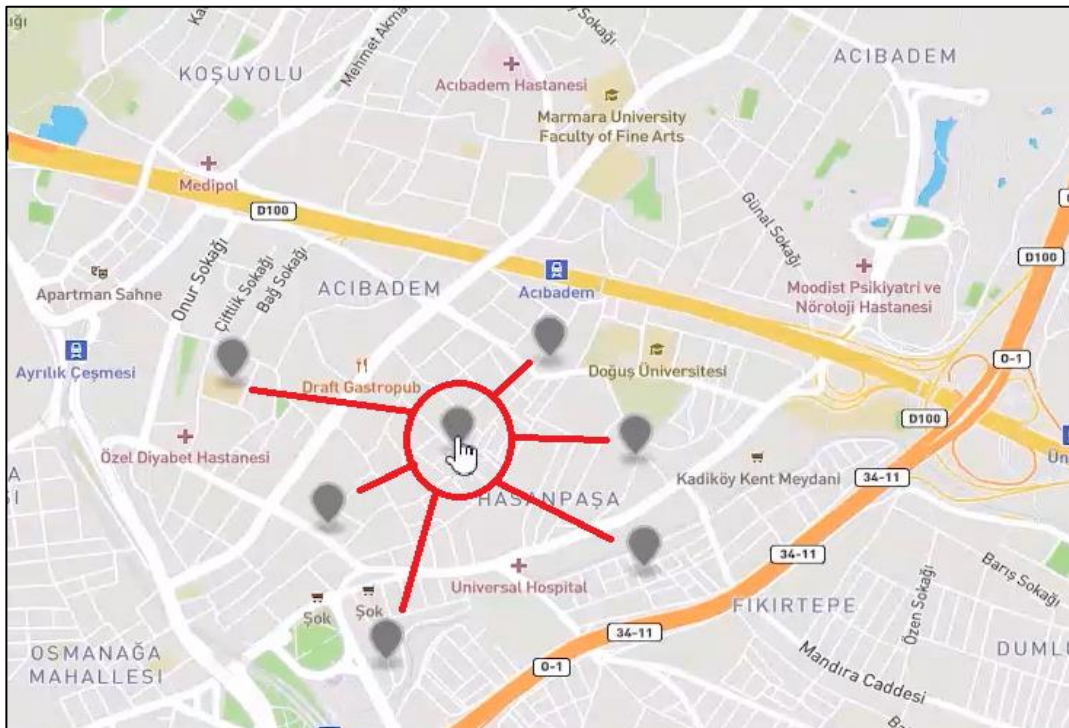
```
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

},
"hits": {
  "total": {
    "value": 2,
    "relation": "eq"
  },
  "max_score": 1,
  "hits": [
    {
      "_index": "siteler",
      "_type": "_doc",
      "_id": "1B5VRHMBFji1Ekwh5Pdb",
      "_score": 1,
      "_source": {
        "stitle": "The Mandarin Acıbadem",
        "location": {
          "lat": 40.996679,
          "lon": 29.050921
        }
      }
    },
    {
      "_index": "siteler",
      "_type": "_doc",
      "_id": "4R5VRHMBFji1Ekwh5Pdb",
      "_score": 1,
      "_source": {
        "stitle": "Fortis Sinanlı",
        "location": {
          "lat": 40.99412,
          "lon": 29.051301
        }
      }
    }
  ]
}
```

Distance değeri değiştirilerek arama çapı artırılarak daha fazla site bilgisi bulunabilir.

3.ADIM: Sorting Kullanarak En Yakından En Uzağa Doğru Site Konum Sıralaması

Bir konum belirlenip, çevresindeki noktaları en yakından uzağa doğru sıralama işlemi yapıldı.



- Sıralama için **sorting** yapıldı. Seçilen merkez nokta için **location** alanına enlem ve boylam bilgileri eklendikten sonra **order** komutu ile sıralama yap denilir. Order'a eklenen **ascending** özelliği ile en yakından uzağa doğru sıralama yapılmış olur. **Unit** ile uzaklığın dönüş tipi, **distance type** ile de uzaklığın nasıl ölçüleceği belirtildi.

```
form-data x-www-form-urlencoded raw binary JSON (application/json)
1 {
2   "query": {
3     "bool": {
4       "must": {
5         "match_all": {}
6       }
7     }
8   },
9   "sort": [
10    {
11      "_geo_distance": {
12        "location": {
13          "lon": 29.041414260864254,
14          "lat": 40.997876784676244
15        },
16        "order": "asc",
17        "unit": "m",
18        "distance_type": "plane"
19      }
20    }
21  ]
22 }
```

- Sorgu sonucunda, belirlenen merkez konuma en yakın olan siteden en uzak olana göre verilerin sıralanışı aşağıdaki gibi olmuştur:

1) Acıbadem Konutları, 124 metre

```
Pretty Raw Preview JSON
10 {
11   "hits": {
12     "total": {
13       "value": 10000,
14       "relation": "gte"
15     },
16     "max_score": null,
17     "hits": [
18       {
19         "_index": "siteler",
20         "_type": "_doc",
21         "_id": "ex5VRHMBFjilEkwh5Pdb",
22         "_score": null,
23         "_source": {
24           "stitle": "Acıbadem Konutları",
25           "location": {
26             "lat": 40.998598,
27             "lon": 29.042551
28           }
29         },
30         "sort": [
31           124.62235311718294
32         ]
33       },
34       {
35         "_index": "siteler",
36         "_type": "_doc",
37         "_id": "HB5VRHMBFjilEkwh5Pdb",
38         "_score": null,
39         "_source": {
40           "stitle": "HB5VRHMBFjilEkwh5Pdb",
41           "location": {
42             "lat": 40.998598,
43             "lon": 29.042551
44           }
45         },
46         "sort": [
47           124.62235311718294
48         ]
49       }
50     ]
51   }
52 }
```


2) Polat Sitesi, 135 metre

```
32      },
33      {
34          "_index": "siteler",
35          "_type": "_doc",
36          "_id": "HB5VRHMBFji1Ekwh5Pdb",
37          "_score": null,
38          "_source": {
39              "stitle": "Polat Sitesi",
40              "location": {
41                  "lat": 40.997081,
42                  "lon": 29.040187
43              }
44          },
45          "sort": [
46              135.78889820739158
47          ]
48      },
49  ],
50 }
```

3) Acıbadem Konutları, 150 metre

```
48      },
49      {
50          "_index": "siteler",
51          "_type": "_doc",
52          "_id": "tR5VRHMBFji1Ekwh5PZb",
53          "_score": null,
54          "_source": {
55              "stitle": "Acıbadem Konutları",
56              "location": {
57                  "lat": 40.997988,
58                  "lon": 29.043203
59              }
60          },
61          "sort": [
62              150.62073278253794
63          ]
64      },
65  ],
66 }
```

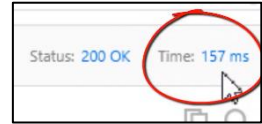
4) Çamlıca Sitesi, 244 metre

```
64      },
65      {
66          "_index": "siteler",
67          "_type": "_doc",
68          "_id": "HR5VRHMBFji1Ekwh5Pdb",
69          "_score": null,
70          "_source": {
71              "stitle": "Çamlı Köşk",
72              "location": {
73                  "lat": 40.999904,
74                  "lon": 29.040273
75              }
76          },
77          "sort": [
78              244.11739997435448
79          ]
80      },
81  ],
82 }
```

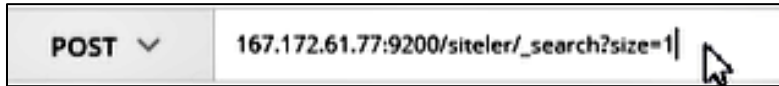
5) Denizciler Sitesi, 368 metre

```
95 }
96 }
97 {
98   "_index": "siteler",
99   "_type": "_doc",
100   "_id": "hR5VRHMBFji1Ekwh5Pdb",
101   "_score": null,
102   "_source": {
103     "stitle": "Denizciler Sitesi",
104     "location": {
105       "lat": 40.999705,
106       "lon": 29.045082
107     }
108   },
109   "sort": [
110     368.8688590245692
111   ]
112 }
113 }
```

NOT-I: Bu işlemleri ilişkisel veri tabanları ile yapmaya kalksaydık hem çok kompleks sorgular yazmak zorunda kalacaktık hem de performans açısından yeterli verimi alamayacaktık. Elasticsearch üzerinden yaptığımız bu sorgular ise hem kolay ve anlaşılır hem de 157 saniye gibi kısa bir sürede performansı yüksek sonuçlar üretti.



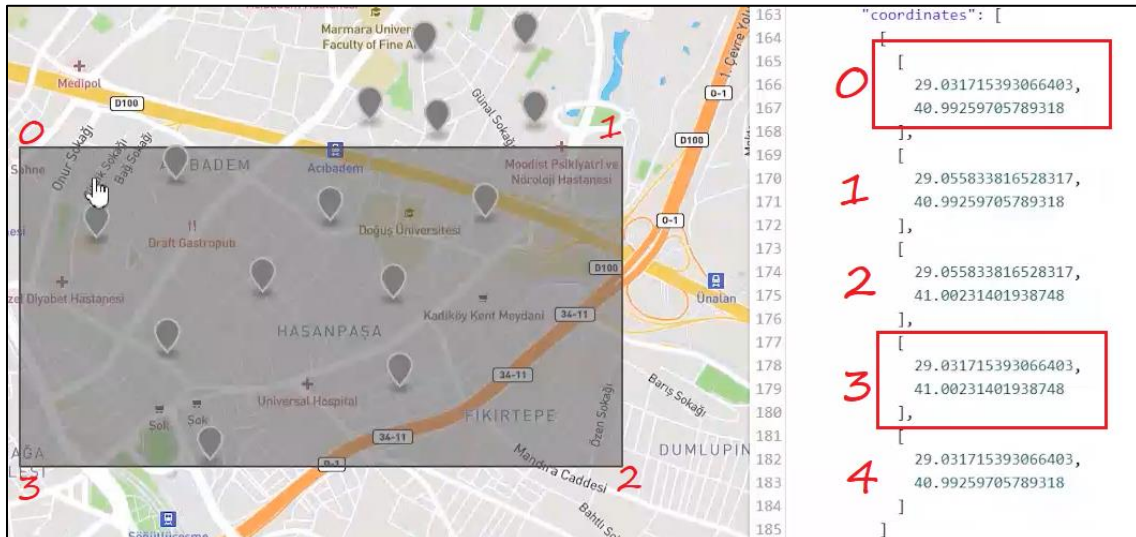
NOT-II: Arama sonuçlarında sadece en yakın noktayı bulmak için; URL de search'ün yanına “?size=1” yazılır ve böylece tek değer döndürülmüş olur:



Bu işlem ile mevcut konumdan “Bana en yakın ATM’yi bul” gibi istekler için kullanılabilir.

4.Adım: Belirlenen Kapalı Alan İçerisinde Bulunan Siteler Belirlenir

Bırakılan pointler içerisinde bir kapalı alan oluşturuldu ve alan dahilinde kalan siteler bulundu.



- Bunun için kapalı alan anlamındaki bounding işlemi yapıldı. Kapalı bir alan olduğu için **geo_bounding_box** parametresi yazıldı ve **location** field'ına **top_left** ve **bottom_right** adındaki iki alan eklendi. Bu alanlar sayesinde elasticsearch kapalı alanı oluşturabiliyor. Sol en üst(0 numaralı) ve sağ en alt(3 numaralı) enlem ve boylam bilgileri yazıldı.

```

form-data x-www-form-urlencoded raw binary JSON (application/json)
{
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "geo_bounding_box": {
          "location": {
            "top_left": {
              "lat": 40.99259705789318,
              "lon": 41.00231401938748
            },
            "bottom_right": {
              "lat": 29.031715393066403,
              "lon": 29.055833816528317
            }
          }
        }
      }
    }
  }
}

```

- Sorgu sonucunda, çizilen kapalı alan içerisinde elimizdeki veride bulunan 41 sitenin mevcut olduğu görüldü.

```

Body Cookies Headers (3) Test Results
Pretty Raw Preview JSON
{
  "total": 1,
  "successful": 1,
  "skipped": 0,
  "failed": 0
},
"hits": {
  "total": {
    "value": 41,
    "relation": "eq"
  },
  "max_score": 1,
  "hits": [
    {
      "_index": "siteler",
      "_type": "_doc",
      "_id": "3x5VRHMjBFj1Ekwh4-k3",
      "_score": 1,
      "_source": {
        "stitle": "Güneş Sitesi",
        "location": {
          "lat": 40.896998,
          "lon": 29.182
        }
      }
    },
    {
      "_index": "siteler",
      "_type": "_doc",
      "_id": "5R5VRHMjBFj1Ekwh4-k3",
      "_score": 1,
      "_source": {
        "stitle": "Özenel Sitesi",
        "location": {
          "lat": 40.899933,

```

NOT-I: Bu işlem sayesinde; belirli bir kapalı alan içinde kaç restoran var, kaç eczane var ya da müşteri GPS'leri takip edilerek bu alanda kaç tane müşteriniz var gibi büyük verilere ulaşmak mümkündür.

NOT-II: Elasticsearch’ün bu gibi işlemlerde ne kadar performanslı olduğundan yukarıdaki sorguda bahsetmiştik, bu işlem sonucu ise yine 132ms gibi kısa bir zamanda yüksek performans ile gerçekleştirildi.

NOT-III: İçerisinde geçen kelimelere göre bu kapalı alan içerisinde bulunan siteleri bulabilmek de mümkün. Örneğin; belirlenen kapalı alan içerisinde adında “Güneşli” geçen 1 site bulunuyor.

```
1 {
2   "query": {
3     "bool": {
4       "must": {
5         "match": {"title": "Güneş"}
6       },
7       "filter": {
8         "geo_bounding_box": {
9           "location": {
10            "top_left": {
11              "lat": 40.90053886975936.
```

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON

```
1 {
2   "took": 1,
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": 1,
13      "relation": "eq"
14    },
15    "max_score": 6.3480697,
16    "hits": [
17      {
18        "_index": "siteler",
19        "_type": "_doc",
20        "_id": "3x5VRHM8Fji1Ekwh4-k3",
21        "_score": 6.3480697,
22        "_source": {
23          "title": "Güneş Sitesi",
24          "location": {
25            "lat": 40.896998,
26            "lon": 29.182
27          }
28        }
29      }
30    ]
31  }
32 }
```

5.Adım: Elasticsearch Üzerine Yüklenen Site Verilerini ve Sorguları Kibana ile Görselleştirme

Kibana elasticsearch ile çalışan bir görselleştirme dashboardudur.

Elasticsearch

Ingest Node Pipelines
Index Management
Index Lifecycle Policies
Transforms
Rollup Jobs
Snapshot and Restore
License Management
Remote Clusters
8.0 Upgrade Assistant

Kibana

Index Patterns
Alerts and Actions
Saved Objects
Spaces
Reporting
Advanced Settings

Index Management

Index Management docs

Indices Index Templates

Update your Elasticsearch indices individually or in bulk.

Include rollout indices Include system indices

Search Lifecycle status Lifecycle phase Reload indices

Name	Health	Status	Primaries	Replicas	Docs count	Storage size
siteler	yellow	open	1	1	10065	728.9kb

Rows per page: 10

5.1) Elasticsearch'teki verileri görselleştirmek için ilk olarak index pattern oluşturulur.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

☐ Include system indices

Step 1 of 2: Define index pattern

Index pattern

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, *, <, >, |.

✓ **Success!** Your index pattern matches **1 index**.

siteler

Rows per page: 10

[Next step](#)

Elasticsearch ve kibana entegrasyonu sağlandı ve veri bilgileri aşağıdaki gibi geldi.

★ siteler

Default

This page lists every field in the **siteler** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the [Elasticsearch Mapping API](#)

Fields (7) Scripted fields (0) Source filters (0)

All field types

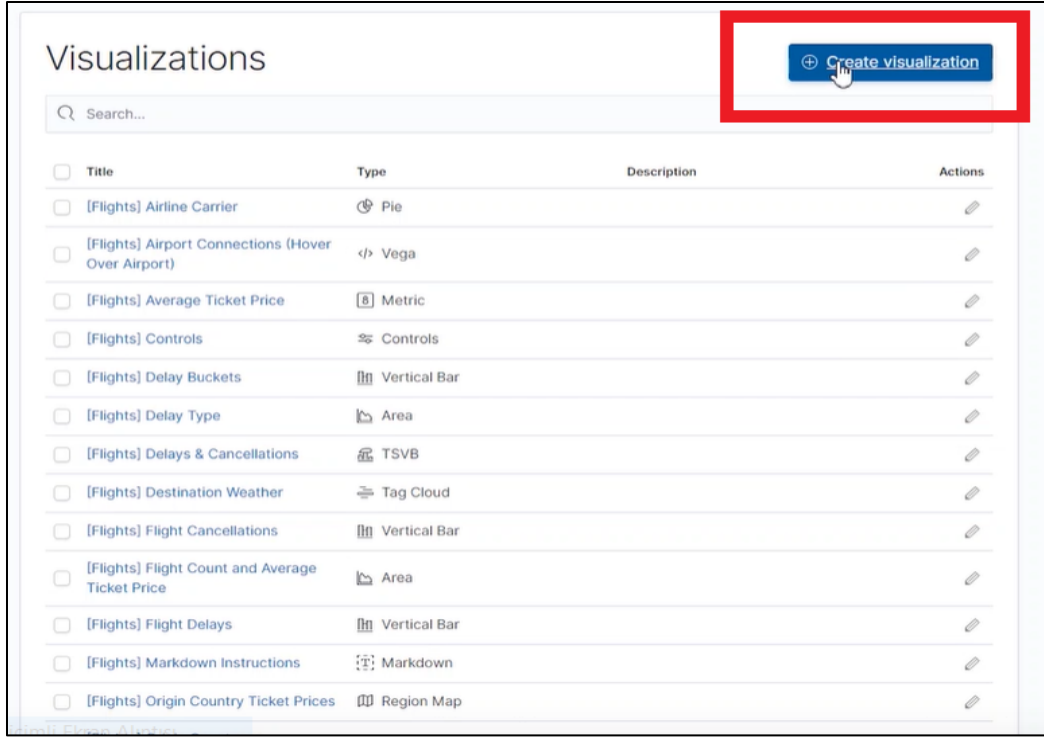
Name	Type	Format	Searchable	Aggregatable	Excluded
_id	string		•	•	
_index	string		•	•	
_score	number				
_source	_source				
_type	string		•	•	
location	geo_point		•	•	
stitle	string		•		

Rows per page: 10

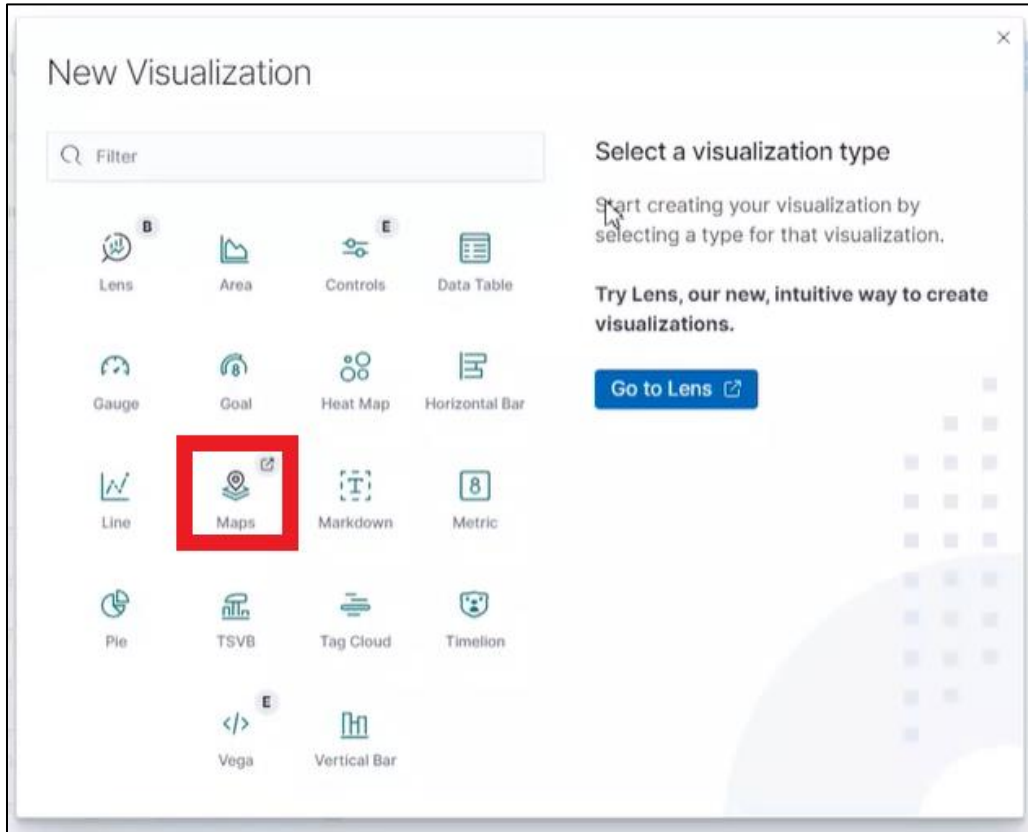
< 1 >

5.2) Visualization seçeneği ile görselleştirme işlemine başlanılır.

Aşağıdaki görselde de görüldüğü gibi ilk açıldığında Elasticsearch içinde sample datalar bulunmaktadır. Create Visualization seçeneği ile görselleştirme oluşturma işlemine başlanılır.

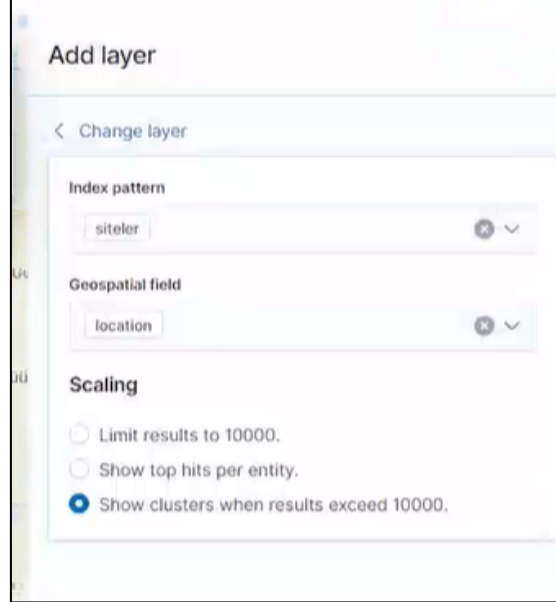


Karşımıza visualization seçenekleri çıkar ve buradan nasıl bir görselleştirme yapacağımız seçilebilir. Coğrafi sorgular yaptığımız için **Maps**'i seçmek daha uygun olacaktır.

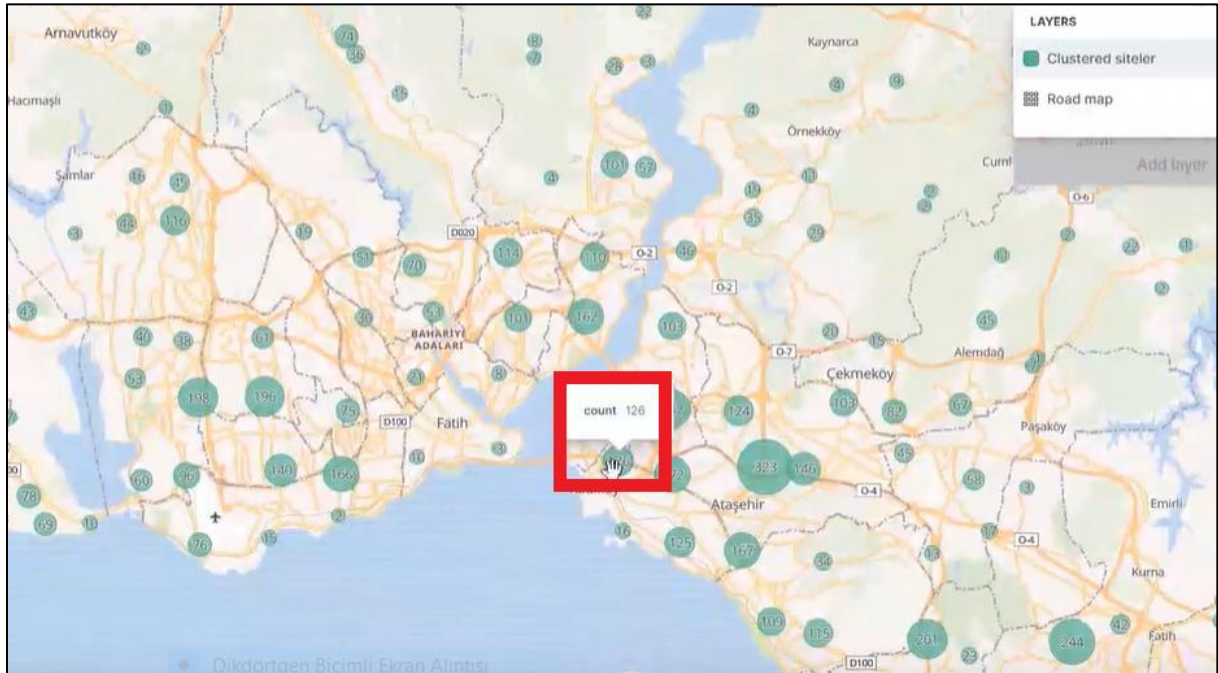


Map' e tıkladığında karşımıza bir dünya haritası çıkar. Bu harita; OpenStreetMap üzerinden render ediliyor. OpenStreetMap açık kaynak kodlu bir harita katmanıdır. Bunu kullanarak harita bazlı işlemler yapılabilir.

İstanbul bölgesine yaklaşılarak Add Layer diyoruz. Daha sonra JSON buraya Upload da edilebilir veya Documents diyerek Elasticsearch'ten de alınabilir. Biz Documents seçeneği ile layerımızı eklemeyi tercih ediyoruz. Ardından 5.1'de oluşturulan Kibana pattern'ı alınır. Kibana Geospatial field'i otomatik olarak görür ve harita üzerinde görselleştirir.



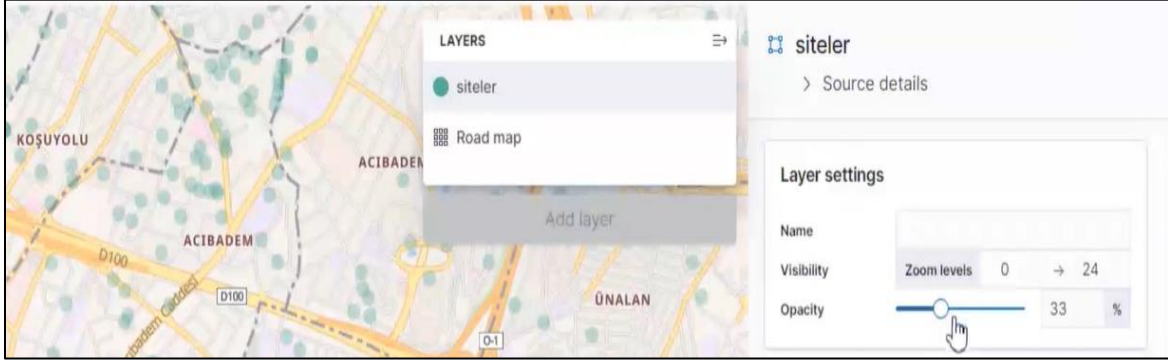
Aşağıdaki resimde görüldüğü üzere görselleştirme yapılmış ve Kadıköy point ine bakınca 126 site olduğu bilgisi verilmektedir.



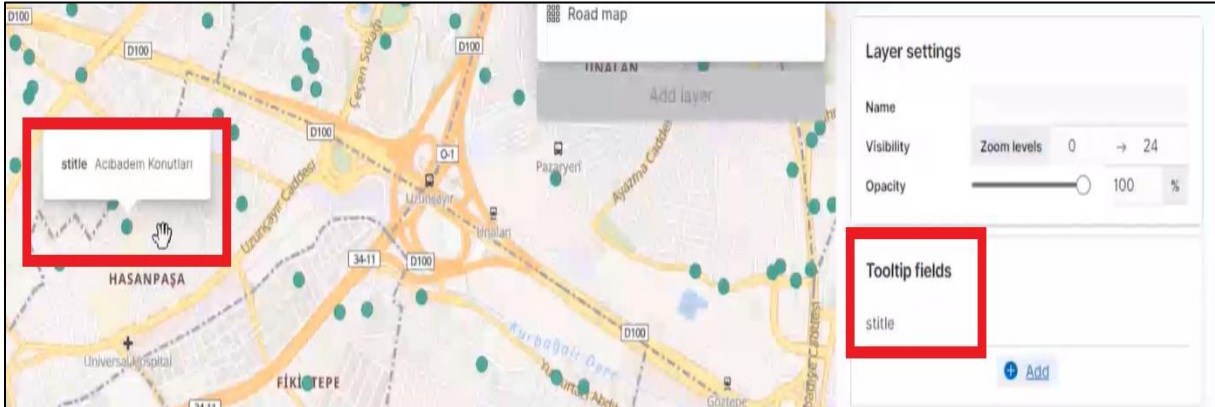
Haritaya yaklařıldığında siteler tek tek görülebilir fakat sitelere tıklanınca henüz bir işlem yapılmıyor çünkü layer ekleme işlemi devam ediyor. Bu nedenle tekrar Add Layer diyoruz.

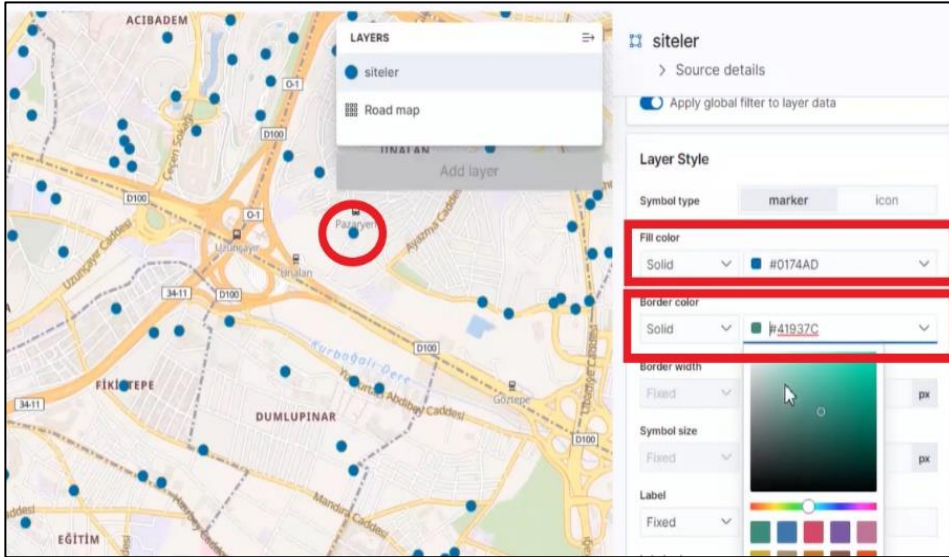
5.3) Layer ayarları yapılandırılır

Sağ tarafta bazı layer ayarları bulunmaktadır. Orada **Opacitiy** seçeneđi ile point'lerin görünürlüğü yarı saydam veya net olarak ayarlanabilir. Ayrıca **Zoomlevel** seçeneđi ile belirlenen bir yakınlařtırma seviyesine ulařana kadar sitelerin gözükmemesi sağlanabilir. Bunu 0 yaparak en uzak seviyede bile sitelerin gözükmesini sağladık.

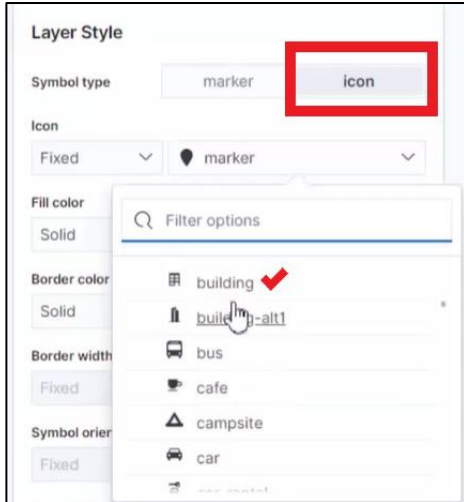


Haritanın sağ tarafında **Tooltip Fields** alanı bulunur. Burada Add diyerek **stitle** field'ini ekliyoruz. Böylece point üzerine tıklandığında hangi site olduđu bilgisi görselleřtirilmiř olur. Ařađıda görüldüđu gibi bir pointe tıklanmıř ve Acıbadem Konutları olduđuına dair bilgi verilerek görselleřtirme sağlanmıřtır.

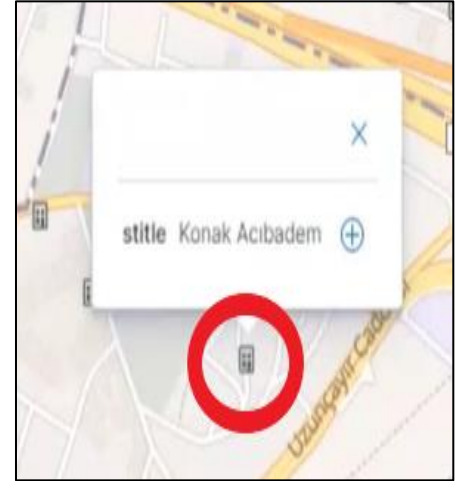




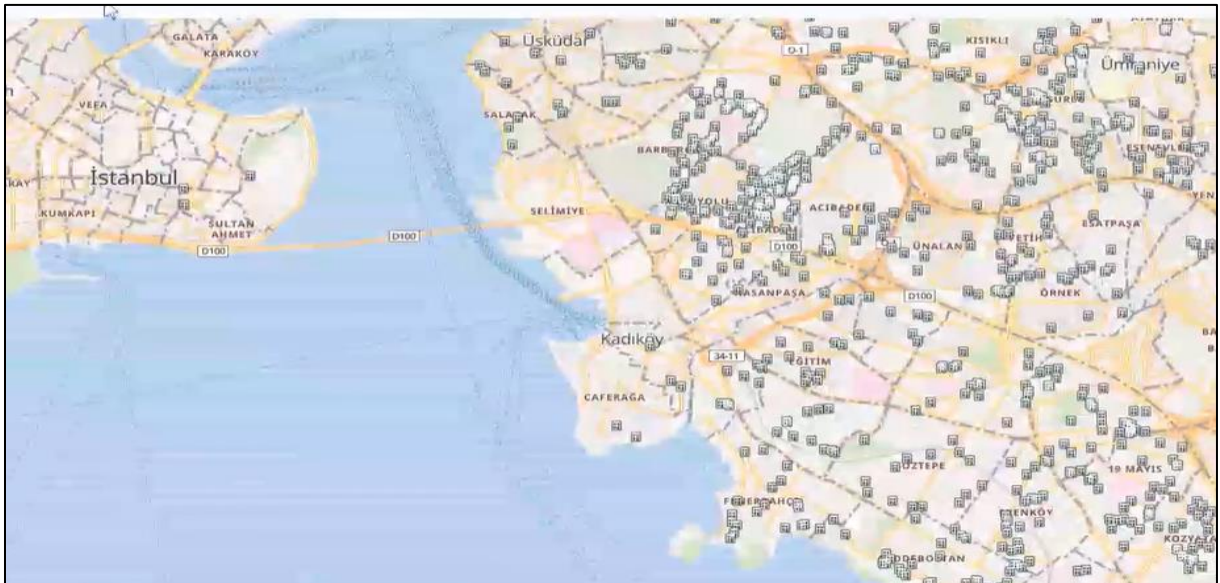
Yandaki gibi Layer Style özelliği ile pointlerin rengi değiştirilebilir ve pointlere border eklenebilir. Böylece istediğimiz görsellik sağlanır.



Yanda görüldüğü gibi **Icon** seçeneğine tıklanarak pointler bir icona dönüştürülebilir. Site görünümünü sağlamak için **building** seçilebilir.

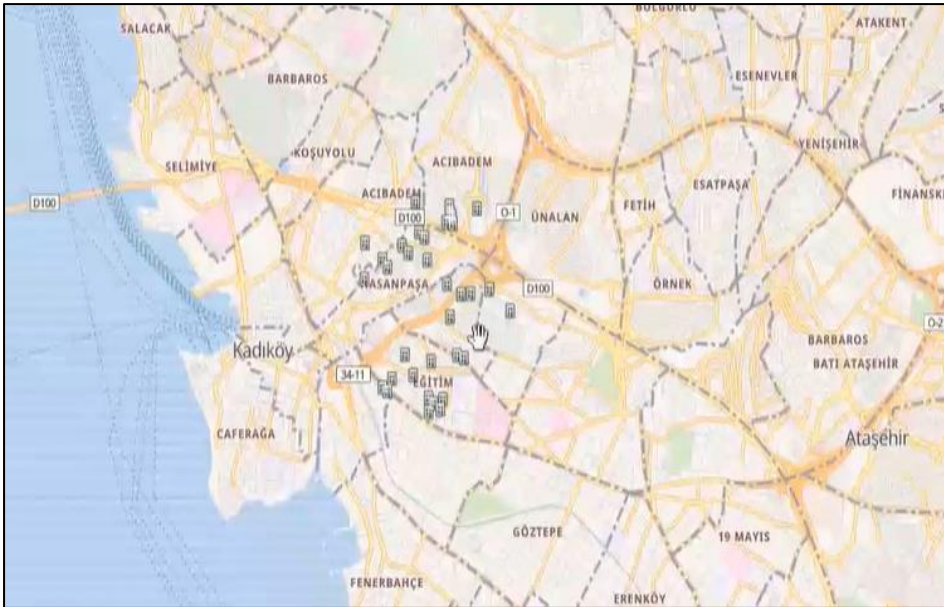
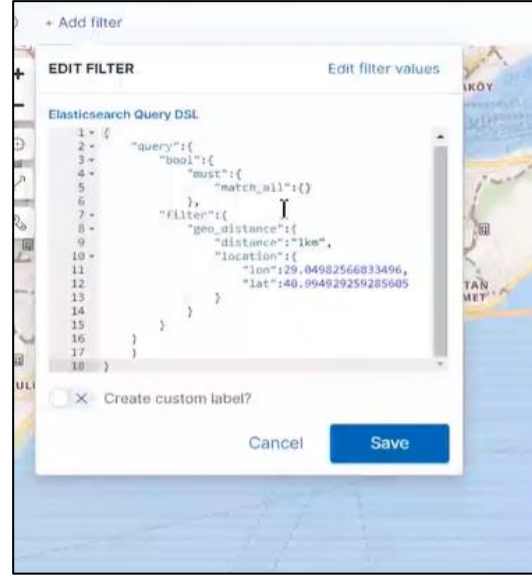
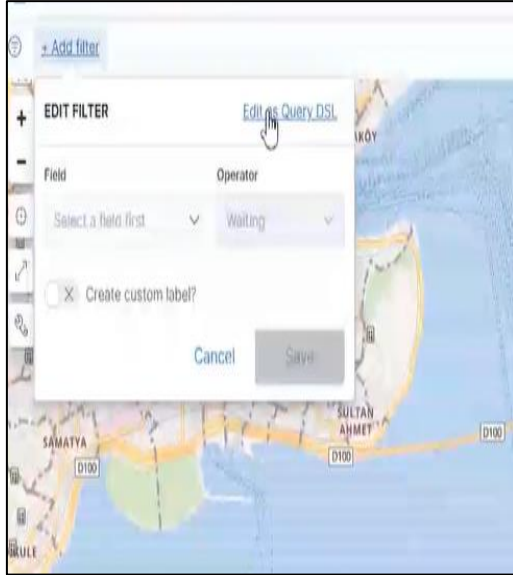


Bu işlemler sonucunda aşağıdaki gibi bir görselleştirme sağlanmış olur.



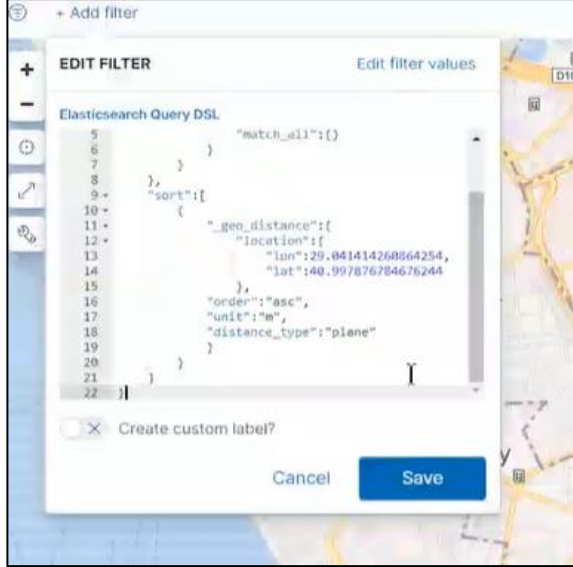
- **Seçili Konuma Göre Belirlenmiş Mesafedeki Diğer Konumları Görselleştirme**

2.2 de yazılan kod (işaretlenen konumdan 1 km uzaklıkta ki konumların listesi) haritada **Add filter** ve ardından **Edit as Query DSL** diyerek çıkan boş alana kopyalanır.



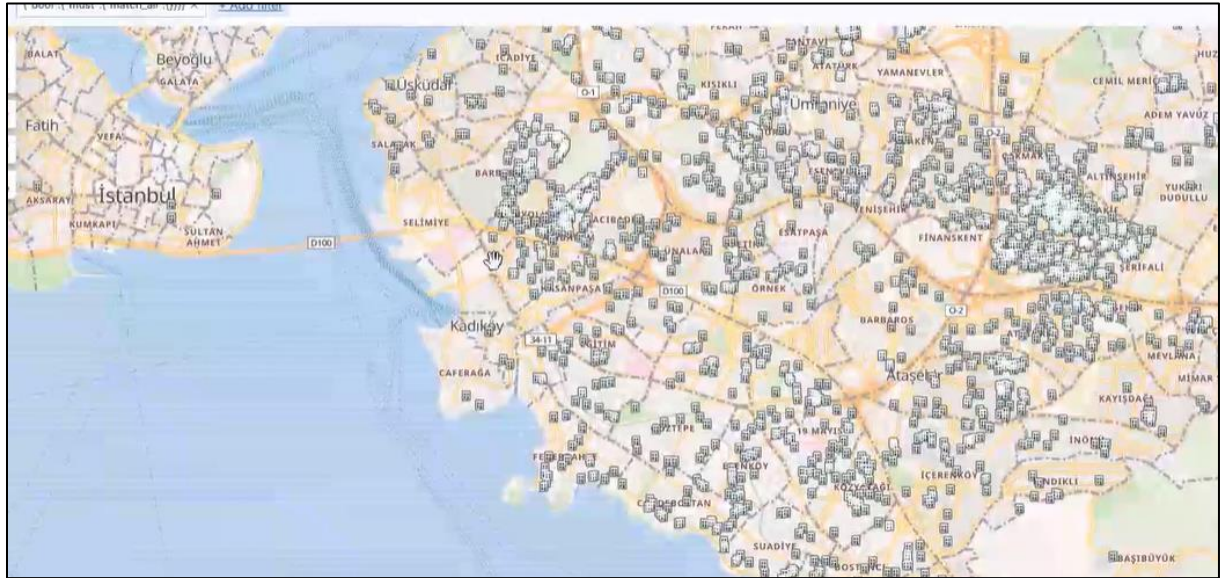
Ve yanda görüldüğü gibi sadece belirlenmiş konumdan 1 km mesafedeki siteler görsel olarak başarılı bir şekilde getirilmiş olur.

- **Sorting Kullanarak En Yakından En Uzağa Doğru Site Konum Sıralaması Görselleştirme**

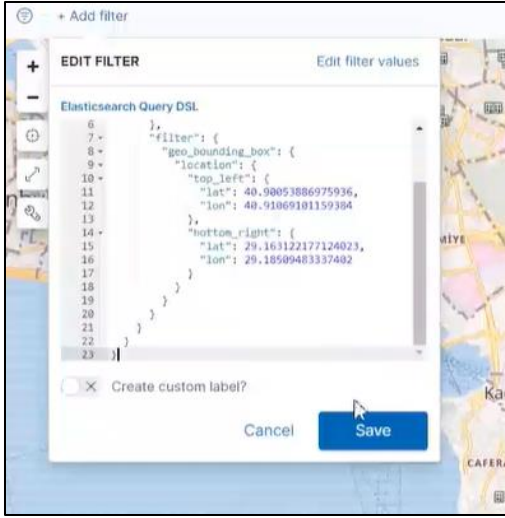


3. Adım da yazılan kod (Bir konum belirlenip, çevresindeki noktaları en yakından uzağa doğru sıralama işlemi) harita da Add filter ve ardından Edit as Query DSL diyerek çıkan boş alana kopyalanır.

Burada sıralama yapıldığından sonuç olarak bütün binaları getirir. Bu nedenle görsellik açısından çok bir anlamı olmaz.



- **Belirlenen Kapalı Alan İçerisinde Bulunan Sitelerin Görselleştirilmesi**



4. Adım da yazılan kod (Bırakılan pointler içerisinde bir kapalı alan oluşturulması ve alan dahilinde kalan siteler bulunması işlemi) harita da Add filter ve ardından Edit as Query DSL diyerek çıkan boş alana kopyalanır.

Burada sonuç olarak çizdiğimiz alanda ki siteler getirilir. Diğer siteler gözükmez.

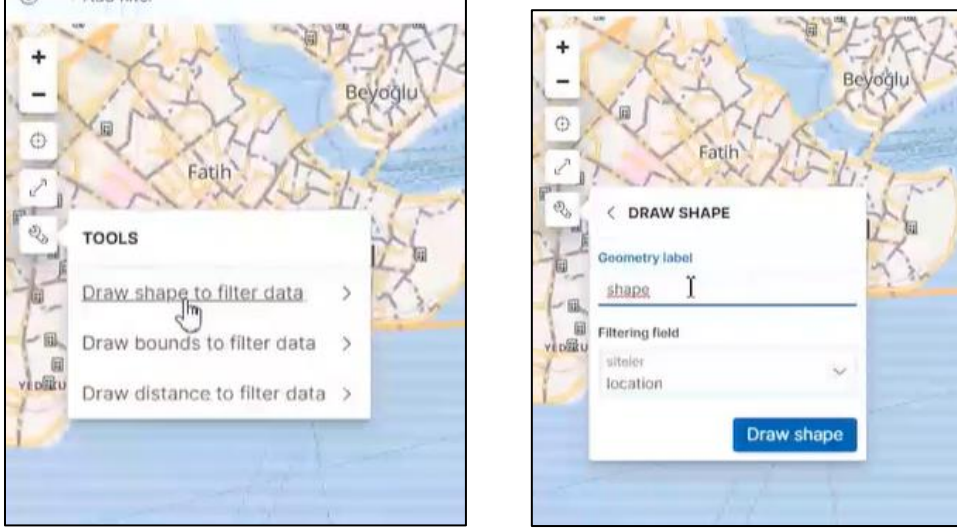


Bu ve bunun gibi örneklerde görüldüğü üzere Elasticsearch'te yazılan sorguların burda da çalıştırılabildiği görülür.

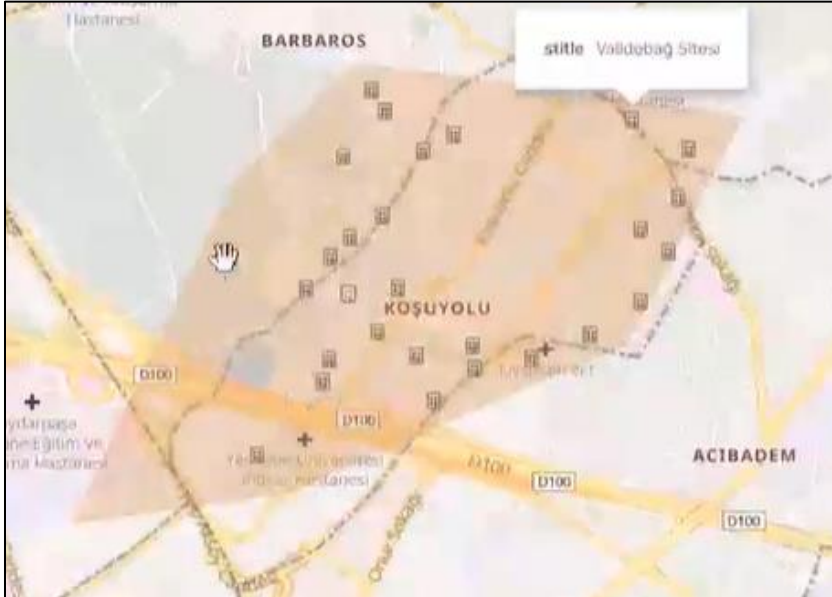
5.4) Kibana ile query olmadan kapalı alan oluşturmak

- Draw shape to filter data ile kapalı alan oluşturmak

Sağ tarafta bulunan Tools seçeneğine tıklanır, Draw Shape to filter data seçilerek shape denilir ve istenilen alanın köşeleri tek tık ile sırasıyla belirlenir. Son köşe ise double tık ile belirlenerek kapalı alan oluşturulur.

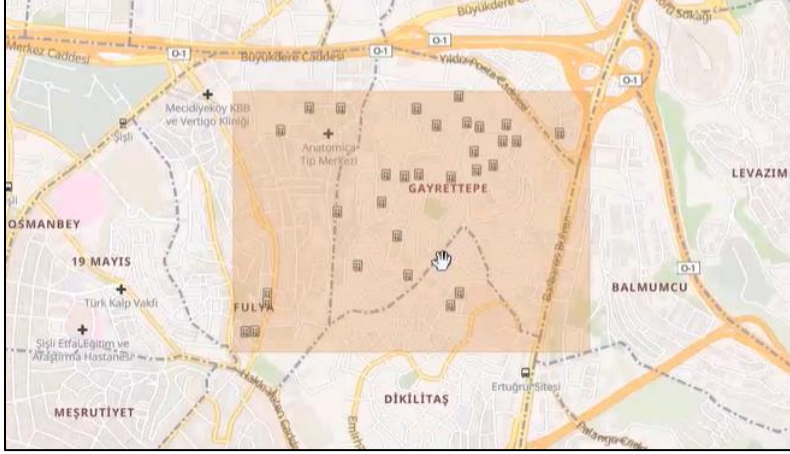


Aşağıda görüldüğü gibi başarılı bir kapalı alan kodsuz, query'siz Kibana ile görselleştirilmiş olur.



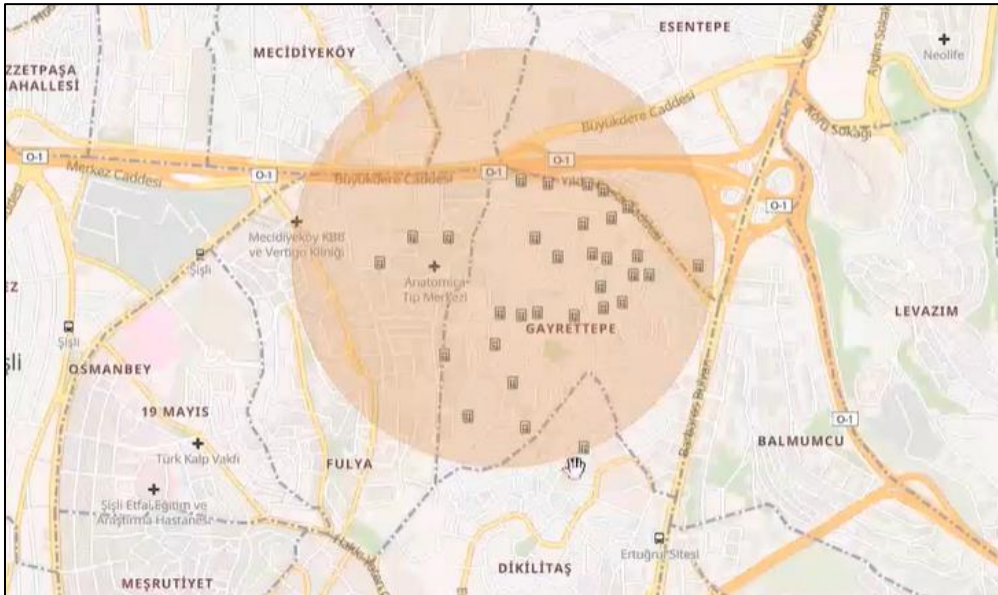
- **Draw bounds to filter data ile kapalı alan oluşturmak**

Sağ tarafta bulunan Tools seçeneğine tıklanır, oradan Draw bounds to filter data seçilerek bound denilir ve kare veya dikdörtgen şeklinde bir kapalı alan belirlenir.



- **Draw distance to filter data ile kapalı alan oluşturmak**

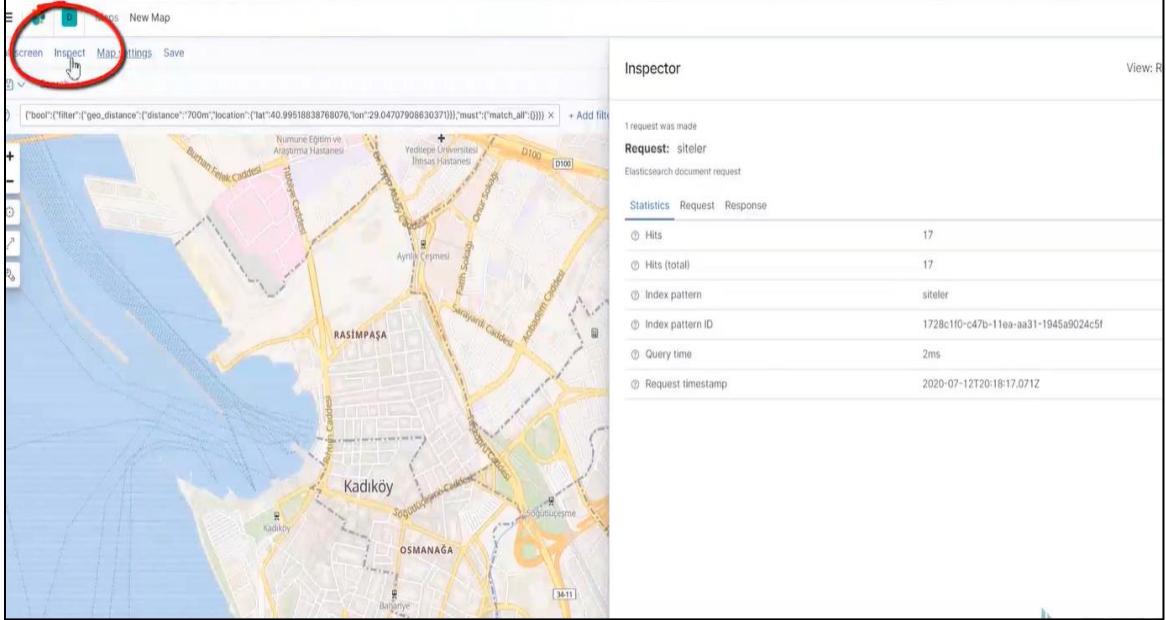
Sağ tarafta bulunan Tools seçeneğine tıklanır, oradan Draw distance to filter data seçilerek bir yere bıraktığımız bir lokasyondan Mouse ile çekilerek circle çizilir. Bu da 2.2 de yazılan kod gibi belli mesafedeki sitelerin görselleştirilmiş halidir.



Bu şekilde geospatal sorgular Kibana ile görselleştirilmiş olur.

5.5) Inspect ile Query nin istatistiklerini görmek

- Burada Statistic bölümünde yazılan query nin ne kadar sürede çalıştığını, kaç datayı getirdiğini, index pattern larını, ID lerini, timestamplerni verir.

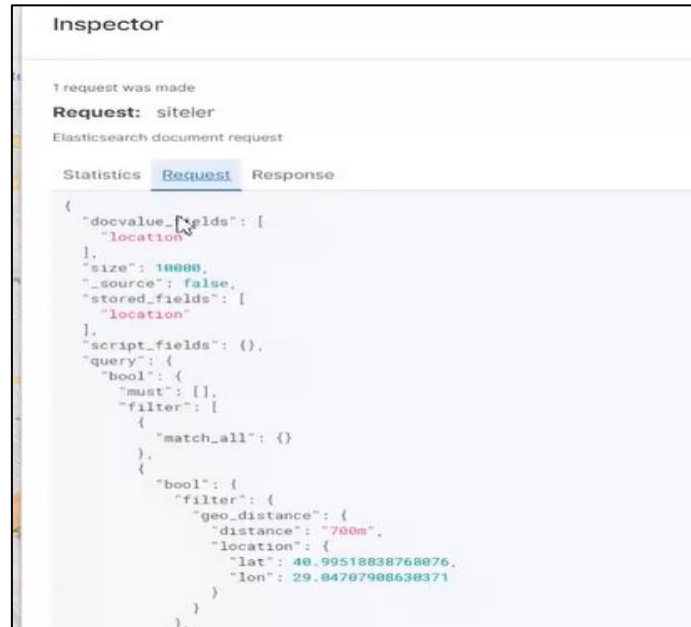


The screenshot shows the Kibana interface. On the left, a map of Kadıköy, Istanbul, is displayed. A red circle highlights the 'Inspect' button in the top-left corner of the map. On the right, the 'Inspector' panel is open, showing the following information:

- 1 request was made
- Request:** siteler
- Elasticsearch document request
- Statistics: Hits, Hits (total), Index pattern, Index pattern ID, Query time, Request timestamp

Statistics	Request	Response
Hits		17
Hits (total)		17
Index pattern		siteler
Index pattern ID		1728c1f0-c47b-11ea-aa31-1945a9024c5f
Query time		2ms
Request timestamp		2020-07-12T20:19:17.071Z

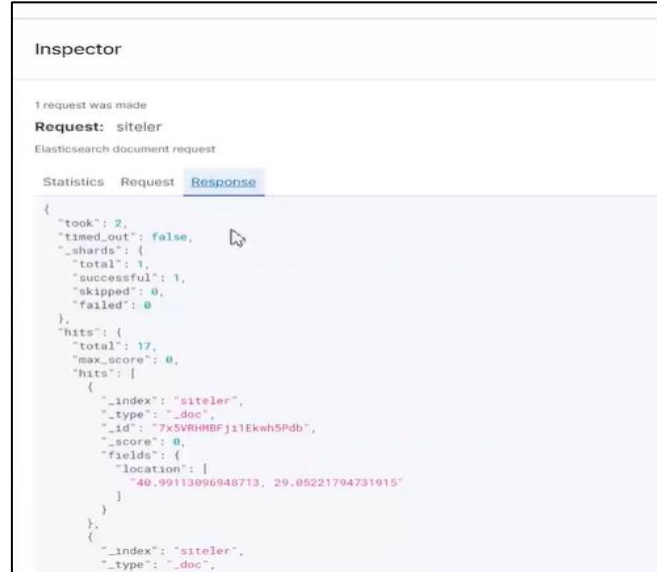
- Request kısmında kod olarak alınabilir yani kopyalanıp kullanılabilir.



The screenshot shows the 'Inspector' panel with the 'Request' tab selected. The request is an Elasticsearch document request for the 'siteler' index pattern. The request body is as follows:

```
{
  "docvalue_fields": [
    "location"
  ],
  "size": 10000,
  "_source": false,
  "stored_fields": [
    "location"
  ],
  "script_fields": {},
  "query": {
    "bool": {
      "must": [],
      "filter": [
        {
          "match_all": {}
        },
        {
          "bool": {
            "filter": [
              {
                "geo_distance": {
                  "distance": "700m",
                  "location": {
                    "lat": 40.99518838768076,
                    "lon": 29.84787988638371
                  }
                }
              }
            ]
          }
        }
      ]
    }
  }
}
```

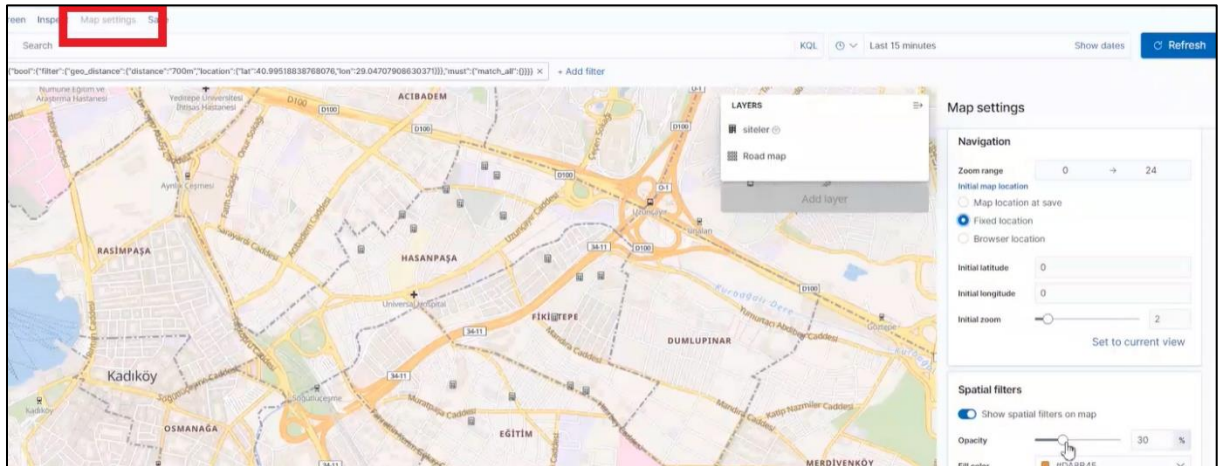
- Response ile ham data şeklinde Postman'e alınmış halde görülebilir.



Kibana sonuç olarak yapılan işlemleri alıp bir Dashboard oluşturmuş olur.

5.6) Map Setting ile Bazı Ayarlamaları Yapılandırmak

Burada yine Zoomrange ayarları yapılabilir. Fixed location ile dahili yerler verilebilir. Özel filtreler koyulabilir, renkler değiştirilebilir. Ve tüm bu işlemler sonunda **Save** diyerek yapılan işlemler kaydedilerek Dashboard da da bu map görülebilir.



Kısaca Kibana Elasticsearch için hazırlanmış, zengin raporlar ve görsel çıktılar almanızı sağlar. Elasticsearch'e kaydedilen logları anlık olarak izlemenizi sağlar ve ihtiyaç duyduğunuzda grafiksel istatistikler çıkartabilen bir web uygulamasıdır.

SONUÇ:

Elimizde json formatında bulunan İstanbul'daki bazı siteler verisini **elasticsearch**'e gönderdik ve veri üzerinde çeşitli coğrafik sorgular yaparak istenilen amaca hizmet etmesini sağladık. Bu işlem için elasticsearch seçmemizin sebebi ise coğrafik sorguların **semi-structured** yapıda bulunmasıydı. İlişkisel veri tabanlarını kullanarak bu sorguları atmak hem çok karmaşık hem de performans açısından yetersiz olacaktı. Elasticsearch ile 100-150 ms gibi kısa bir sürede veri üzerinde sorgular oluşturabildiğimizi gördük, bu da semi-structure yapılarda bir NoSql veri tabanının ne kadar performanslı çalıştığını gösterdi. Ardından elasticsearch üzerinde bir veri görselleştirme aracı olan **Kibana**'yı kullanarak veri setimizi görselleştirdik.

Sitemi Bul Uygulaması; seçilen bir noktaya oturduğumuz sitenin uzaklıklarının bulunması ve görselleştirilmesi, oluşturulan kapalı bir alan içerisinde kaç adet site bulunuyor ve oturduğumuz site bu alanın neresinde kalıyor belirlemesi ve görselleştirilmesi, diğer sitelerin oturduğumuz siteye olan uzaklıklarının sıralanması ve görselleştirilmesi amaçlarına hizmet ediyor.

Referanslar:

- <https://pypi.org/project/elasticsearch-loader/>
- <https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html>
- <http://geojson.io/#map=2/20.0/0.0>
- <http://www.gokmeneskin.com/blog/python-pip-nedir-ve-nasil-kurulur/>