Dinesh Karamchandani

A20407484

Source code for Phase 3 ipynb files

1. Import_issues.ipynb

```
#Author : DSP18SCM84K

import github3

import json

GITHUB_TOKEN = '03a0975c4877eba8924b0fe03f056cbbd09fe3e5'

ORG = 'SPM587SP18'

REPO = 'SCM587SP18'

FILENAME_ISSUES = ORG + 'issues.json'

gh = github3.login(token=GITHUB_TOKEN)


f = open(FILENAME_ISSUES, 'w')

for issue in gh.search_issues('type:issue repo:SPM587SP18/SCM587SP18'):        # Find issues from given Repo

        label_name=[]

        data={}


        current = issue.as_json()

        current_issue =json.loads(current)


        data['issue_number']=current_issue["number"]                    # Get issue number

        data['created_at']= current_issue["created_at"][0:10]             # Get created date of issue

        if current_issue["closed_at"] == None:

           data['closed_at']= current_issue["closed_at"]

        else:

           data['closed_at']= current_issue["closed_at"][0:10]           # Get closed date of issue
```

```
        for label in current_issue["labels"]:
            label_name.append(label["name"])                    # Get label name of issue
        data['labels']= label_name
        data['State'] = current_issue["state"]                  # It gives state of issue like closed or open
        data['Author'] = current_issue["user"]["login"]         # Get Author of issue
        out=json.dumps(data)                                    # save this all information to a JSON file
        f.write(out+ '\n')
f.close()
```

2. charting_issues.ipynb

Author: DSP18SCM84K
created_at: 2018-03-31,
labels: bug,
State: closed,
issue_number: 1,
closed_at: 2017-04-06

```
import os
import _pickle as pickle
import pandas as pd                     # panda's nickname is pd
import numpy as np                      # numpy as np
from pandas import DataFrame, Series    # for convenience
import matplotlib.pyplot as plt
%matplotlib inline

# Read the JSON file into a list of dictionaries
import json
list_of_issues_dict_data = [json.loads(line) for line in open('SPM587SP18issues.json')]
# Create the DataFrame object for the list_of_issues_dict_data object
issues_df = DataFrame(list_of_issues_dict_data)
# Sanity test: print first 10 rows in our DataFrame
issues_df
# Prepare and Clean the dataframe object
wrangled_issues_df = issues_df[['Author','State','closed_at','created_at','issue_number','labels']]
wrangled_issues_df.loc[0:len(wrangled_issues_df), 'OriginationPhase']= np.NaN
wrangled_issues_df.loc[0:len(wrangled_issues_df),'DetectionPhase']= np.NaN
wrangled_issues_df.loc[0:len(wrangled_issues_df),'Category']= np.NaN
wrangled_issues_df.loc[0:len(wrangled_issues_df),'Priority']= np.NaN
```

```
wrangled_issues_df.loc[0:len(wrangled_issues_df),'Status']= np.NaN

wrangled_issues_df.at[4,'issue_number']
for i in range(0, len(wrangled_issues_df)):
   print(i,wrangled_issues_df.iloc[i]['issue_number'])
   if wrangled_issues_df.iloc[i]['labels']:
      for label in wrangled_issues_df.iloc[i]['labels']:
         label_name= (label.split(':'))[0]
         label_value= (label.split(':'))[1]
         wrangled_issues_df.loc[i, label_name]=label_value
wrangled_issues_df

# Plot in Bar Chart the total number of issues created every day for every Detaction Phase
LabelsReviewedByDate =
wrangled_issues_df.groupby(['created_at','DetectionPhase']).created_at.count()
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue','yellow',
'purple', 'red', 'green'], grid=False)

# Plot in Bar Chart the total number of issues created for every Phase based on thier priorites
LabelsReviewedByDate = wrangled_issues_df.groupby(['Priority','DetectionPhase']).created_at.count()
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue','yellow',
'purple', 'red', 'green'], grid=False)

# Plot in Bar Chart the total number of issues closed every day for every Category
LabelsReviewedByDate = wrangled_issues_df.groupby(['closed_at','Category']).closed_at.count()
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue', 'purple',
'red'], grid=False)

# Requirement #1: Add your code here
LabelsReviewedByDate =
wrangled_issues_df.groupby(['created_at','OriginationPhase']).created_at.count()
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue','yellow',
'purple', 'red', 'green'], grid=False)

# Requirement #2: Add your code here
LabelsReviewedByDate = wrangled_issues_df.groupby(['Status','OriginationPhase']).created_at.count()
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue','yellow',
'purple', 'red', 'green'], grid=False)
```

3. Heatmap_issues.ipynb

```
#Dinesh Karamchandani A20407484
import pandas as pd                          # panda's nickname is pd
import json                        # numpy as np
```

```
df1 = [json.loads(line) for line in open('SPM587SP18issues.json')]
from elasticsearch import Elasticsearch, helpers
es = Elasticsearch()
#print 1st record of dataframe of json
df1[:1]
#build the index
actions = []
for data in df1:
   action = {
      "_index": "issues_1",
      "_type": "issues",
      "_id": data["issue_number"],
      "_source":data


      # iterate on keys and if keys is list then split

   }
   actions.append(action)
#actions
helpers.bulk(es,actions)

#Part 1) query to get all the records
doc = {
      'size': 10000,
      'query':{
         'match_all': {}
      }
}

all_results = es.search(index ='issues_1', body =doc,scroll = '1h')


#getting scroll id and scroll size
sid = all_results['_scroll_id']
scroll_size = all_results['hits']['total']

sid
scroll_size
all_results

#writing function to get list of locations for folium heatmap from the results
def get_final_location(all_results, scroll_size,sid):
   count = 0
```

```
    count2,count3 =0,0
    final_location = []
    final_issue_no = []
    final_issue_no2 = []
    while(scroll_size>0):
        for doc in all_results['hits']['hits']: #[1]['_source']['labels']:
            #print((i.split(':'))[0])
            location_ll = []
            location_dict = {}
            results = doc['_source']['labels']
            count = count+1
            for i in results:
                #print(i)
                #print("In here")
                label_name = (i.split(':'))[0].strip().title()
                #label_value = (i.split(':'))[1]
                if (label_name == 'Latitude' or label_name == 'Longitude' and (i.split(':'))[1] != None):
                    label_value = (i.split(':'))[1].strip()
                    #print(label_name,label_value)
                    location_dict[label_name] = float(label_value)
                    final_issue_no.append(doc['_source']['issue_number'])
                    #count2=count2+1
                    #location.append(float(label_value))
                    #print("Inside for location_dict", location_dict)
            #print("Outside for")
            if('Longitude' in location_dict and 'Latitude' in location_dict):
                count3=count3+1
                final_issue_no2.append(doc['_source']['issue_number'])
                final_location.append([location_dict['Latitude'],location_dict['Longitude']])
                #print("Final location", final_location)
            #print("Outside for location-dict", location_dict)
        all_results = es.scroll(scroll_id = sid, scroll = '2m')
        sid = all_results['_scroll_id']
        scroll_size = len(all_results['hits']['hits'])
    #print("records fetched", count)
    #print("records fetched", count2)
    print("records fetched", count3)
    return final_location

final_location = get_final_location(all_results, scroll_size,sid)
#although issues are 253, only 142 have location against them hence we will only 142 records in the
heatmap

#printing final_locations first 10 records
```

```
final_location[:10]

import folium
from folium import plugins
print(folium.__version__)

#define function for getting folium heatmap

def get_folium_heatmap(final_location):
   without_condition_heatmap = folium.Map([41.878693, -87.638924],zoom_start = 11)
   return without_condition_heatmap.add_child(plugins.HeatMap(final_location,radius=15))
   #return without_condition_heatmap

heatmap = get_folium_heatmap(final_location)
```

#Part1 Folium heatmap for all the issues

```
heatmap
```

#Part2. 1) DetectionPhase is Field AND Priority is Critical

```
        doc = {
              'size': 10000,
              'query':{
                 'bool': {
                       'must' : [

                          {'match' : {'labels':'DetectionPhase:Field'}},
                          {'match' : {'labels':'Priority:Critical'}}
                       ]
                 }

              }
        }

all_results = es.search(index ='issues_1', body =doc,scroll = '1h')

sid = all_results['_scroll_id']
scroll_size = all_results['hits']['total']
print("Sid:", sid)
print("Scroll size:", scroll_size)

final_location = get_final_location(all_results, scroll_size,sid)
```

#only 3 records were collected as one of the records only has only latitude and no longitude - issue#12
final_location

#Part 2. 1) Heatmap for DetectionPhase is Field AND Priority is Critical

```
heatmap = get_folium_heatmap(final_location)
heatmap

doc = {
      'size': 10000,
      'query':{
         'bool': {
               'must' : [

                  {'match' : {'labels':'DetectionPhase:Field'}},
                  {'match' : {'labels':'Status:Completed'}}
               ]
         }

      }
}

all_results = es.search(index ='issues_1', body =doc,scroll = '1h')

sid = all_results['_scroll_id']
scroll_size = all_results['hits']['total']
scroll_size

final_location = get_final_location(all_results, scroll_size,sid)

final_location
#40.170101, -92.177847 located far from chicago hence not in map
```

#Part 2. 2) Heatmap for DetectionPhase is Field AND Status is Completed

```
heatmap = get_folium_heatmap(final_location)
heatmap
```

Part 2. 3)DetectionPhase is Field AND Priority is Critical AND Status is Approved

```
doc = {
      'size': 10000,
      'query':{
```

```
        'bool': {
              'must':[
                {'match' : {'labels':'DetectionPhase:Field'}},
                {'match':{'labels': 'Priority:Critical'}},
                {'match':{'labels':'Status:Approved'}}
              ]
        }

    }
}

all_results = es.search(index ='issues_1', body =doc,scroll = '1h')

sid = all_results['_scroll_id']
scroll_size = all_results['hits']['total']

scroll_size

final_location = get_final_location(all_results, scroll_size,sid)
final_location
#issue 12 with no longitude hence only 2 records inside locations list
```

#Part 2. 3)Heatmap for DetectionPhase is Field AND Priority is Critical AND Status is Approved

```
heatmap = get_folium_heatmap(final_location)
heatmap
```

#Part 2. 4)DetectionPhase is Field AND Priority is Critical or High AND Status is Approved or inProgress

```
doc = {
  "query": {
    "bool": {
      "must": [
       {
         "match": {
           "labels": "DetectionPhase:Field"
         }
       },
       {
         "bool": {
           "should": [
            {
```

```
            "match": {
              "labels": "Priority:Critical"
            }
           },
           {
            "match": {
              "labels": "Priority:High"
            }
           }
          ],
          "minimum_should_match": "1"
        }
       },
       {
        "bool": {
          "should": [
           {
            "match": {
              "labels": "Status:Approved"
            }
           },
           {
            "match": {
              "labels": "Status:inProgress"
            }
           }
          ],
          "minimum_should_match": "1"
        }
       }
     ]
    }
  }
}

all_results = es.search(index ='issues_1', body =doc,scroll = '1h')

sid = all_results['_scroll_id']
scroll_size = all_results['hits']['total']

scroll_size

final_location = get_final_location(all_results, scroll_size,sid)
#although 23 records are there but only 4 issues have longitude and latitude
```

#issue#12 had only latitude and no longitude
final_location

#Part 2. 4)DetectionPhase is Field AND Priority is Critical or High AND Status is Approved or inProgress

```
heatmap = get_folium_heatmap(final_location)
heatmap
```

#Part 2. 5) All locations that got at least 5 issues on the same location

```
#creating a new index since we would need Latitude and Longitude fields outside of the labels array
final_list = []
for data in df1:
   data2 = data
   temp2 = {}
   for key, values in data2.items():
      if(isinstance(values,list)):
         for i in values:
            if(i.split(":")[0].strip().lower()=='longitude' or i.split(":")[0].strip().lower() == 'latitude'):
               temp2[i.split(":")[0].strip()]=i.split(":")[1].strip()

   data.update(temp2)
   action = {
   "_index": "issues_3",
   "_type": "issues",
   "_id": data["issue_number"],
   "_source":data          #temp2
   }
   final_list.append(action)

helpers.bulk(es,final_list)

doc = {
   "aggs": {
      "top_tags": {
         "terms": {
            "field": "Latitude.keyword",
            #"size": 5
            "min_doc_count" : 5
         },
         "aggs": {
            "top_sales_hits": {
               "top_hits": {
```

```
                "_source": {
                   "includes": [ "Latitude", "Longitude" ]
                },
                "size" : 1
             }
          }
       }
    }
  }
}

all_results = es.search(index ='issues_3', body =doc,scroll = '1h')
#all_results

final_location_ll = []
latitude,longitude = 0, 0
for i in all_results['aggregations']['top_tags']['buckets']:
   latitude = float(i['top_sales_hits']['hits']['hits'][0]['_source']['Latitude'])
   longitude = float(i['top_sales_hits']['hits']['hits'][0]['_source']['Longitude'])
   final_location_ll.append([latitude,longitude])
final_location_ll[:10]
```

#Part 2. 5) All locations that got at least 5 issues on the same location

```
heatmap = get_folium_heatmap(final_location_ll)
heatmap
```