Collecting Job Data Using APIs

Estimated time needed: **45 to 60** minutes

Objectives

After completing this lab, you will be able to:

- Collect job data from Jobs API
- Store the collected data into an excel spreadsheet.

> **Note: Before starting with the assignment make sure to read all the instructions and then move ahead with the coding part.**

Instructions

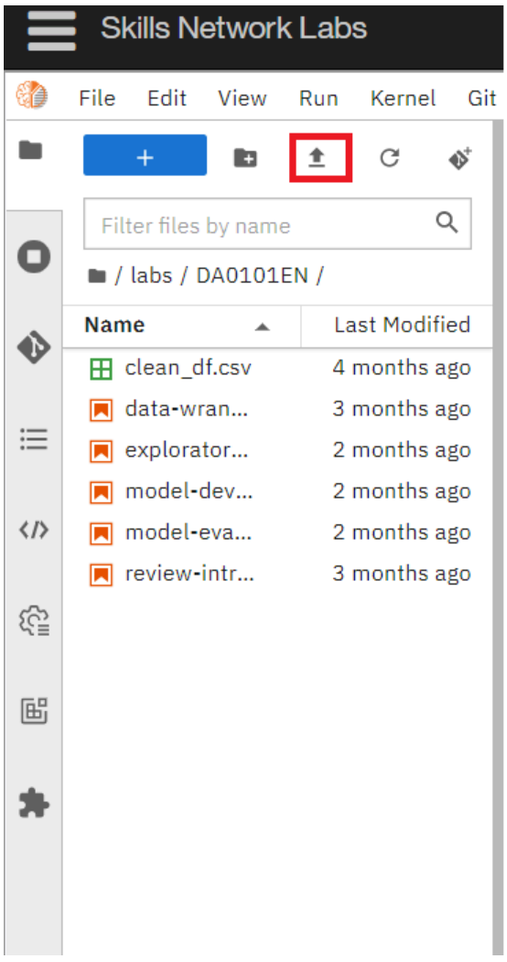
To run the actual lab, firstly you need to click on the Jobs_API notebook link. The file contains flask code which is required to run the Jobs API data.

Now, to run the code in the file that opens up follow the below steps.

Step1: Download the file.

Step2: Upload it on the IBM Watson studio. (If IBM Watson Cloud service does not work in your system, follow the alternate Step 2 below)

Step2(alternate): Upload it in your SN labs environment using the upload button which is highlighted in red in the image below: Remember to upload this Jobs_API file in the same folder as your current .ipynb file



Step3: Run all the cells of the Jobs_API file. (Even if you receive an asterik sign after running the last cell, the code works fine.)

If you want to learn more about flask, which is optional, you can click on this link here.

Once you run the flask code, you can start with your assignment.

Dataset Used in this Assignment

The dataset used in this lab comes from the following source: https://www.kaggle.com/promptcloud/jobs-on-naukricom under the under a **Public Domain license**.

> Note: We are using a modified subset of that dataset for the lab, so to follow the lab instructions successfully please use the dataset provided with the lab, rather than the dataset from the original source.

The original dataset is a csv. We have converted the csv to json as per the requirement of the lab.

Warm-Up Exercise

Before you attempt the actual lab, here is a fully solved warmup exercise that will help you to learn how to access an API.

Using an API, let us find out who currently are on the International Space Station (ISS).
The API at http://api.open-notify.org/astros.json gives us the information of astronauts currently on ISS in json format.
You can read more about this API at http://open-notify.org/Open-Notify-API/People-In-Space/

In [ ]:
```
import requests # you need this module to make an API call
import pandas as pd
```

In [ ]:
```
api_url = "http://api.open-notify.org/astros.json" # this url gives use the astronaut data
```

In [ ]:
```
response = requests.get(api_url) # Call the API using the get method and store the
                                 # output of the API call in a variable called response.
```

In [ ]:
```
if response.ok:              # if all is well() no errors, no network timeouts)
    data = response.json()   # store the result in json format in a variable called data
                             # the variable data is of type dictionary.
```

In [ ]:
```

```

In [ ]:
```
print(data)   # print the data just to check the output or for debugging
```

Print the number of astronauts currently on ISS.

In [ ]:
```
print(data.get('number'))
```

Print the names of the astronauts currently on ISS.

In [ ]:
```
astronauts = data.get('people')
print("There are {} astronauts on ISS".format(len(astronauts)))
print("And their names are :")
for astronaut in astronauts:
    print(astronaut.get('name'))
```

Hope the warmup was helpful. Good luck with your next lab!

Lab: Collect Jobs Data using Jobs API

Objective: Determine the number of jobs currently open for various technologies and for various locations

Collect the number of job postings for the following locations using the API:

- Los Angeles
- New York
- San Francisco
- Washington DC
- Seattle
- Austin
- Detroit

In [ ]:
```
import requests#Import required libraries
import pandas as pd
import json
```

Write a function to get the number of jobs for the Python technology.

> Note: While using the lab you need to pass the **payload** information for the **params** attribute in the form of **key value** pairs.

Refer the ungraded **rest api lab** in the course **Python for Data Science, AI & Development** link

The keys in the json are

- Job Title
- Job Experience Required
- Key Skills
- Role Category
- Location
- Functional Area
- Industry
- Role

You can also view the json file contents from the following json URL.

In [50]:
```
api_url="http://127.0.0.1:5000/data"
def get_number_of_jobs_T(technology):
    payload = {"Key Skills": technology}
    response = requests.get(api_url, params=payload)
    if response.ok:
        job_data = response.json()
        number_of_jobs = len(job_data)
    return technology, number_of_jobs
```

Calling the function for Python and checking if it works.

In [51]:
```
get_number_of_jobs_T("Python")
```

Out[51]:
```
('Python', 1173)
```

Write a function to find number of jobs in US for a location of your choice

In [52]:
```
def get_number_of_jobs_L(location):
    payload = {"Location" : location}
    response = requests.get(api_url,params = payload)
    if response.ok:
        data = response.json()
        number_of_jobs = len(data)
    return location,number_of_jobs
```

Call the function for Los Angeles and check if it is working.

In [53]:
```
get_number_of_jobs_L("Los Angeles")
```

Out[53]:
```
('Los Angeles', 640)
```

Store the results in an excel file

Call the API for all the given technologies above and write the results in an excel spreadsheet.

If you do not know how create excel file using python, double click here for **hints**.

Create a python list of all locations for which you need to find the number of jobs postings.

In [54]:
```
locations = ['Los Angeles', 'New York', 'San Francisco', 'Washington DC', 'Seattle', 'Austin', 'Detroit']
```

In [55]:
```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (3.1.2)
Requirement already satisfied: et-xmlfile in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from openpyxl)
(1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [56]:
```
from openpyxl import Workbook
```

Import libraries required to create excel spreadsheet

Create a workbook and select the active worksheet

In [57]:
```
wb = Workbook()
ws=wb.active
ws.append(locations)
```

Find the number of jobs postings for each of the location in the above list. Write the Location name and the number of jobs postings into the excel spreadsheet.

In [58]:
```
def get_number_of_jobs_TL(technology, locations):
    number_of_jobs_list = []
    for location in locations:
        payload={"Key Skills": technology, "Location": location}
        response=requests.get(api_url, params=payload)
        if response.ok:
            data=response.json()
            number_of_jobs = len(data)
            number_of_jobs_list.append(number_of_jobs)
    return number_of_jobs_list
#     return ws.append(number_of_jobs_list)

get_number_of_jobs_TL("Python", locations)
```

Out[58]:
```
[24, 143, 17, 258, 133, 15, 170]
```

Save into an excel spreadsheet named 'job-postings.xlsx'.

In [59]:
```
wb.save('job-postings.xlsx')
```

In the similar way, you can try for below given technologies and results can be stored in an excel sheet.

Collect the number of job postings for the following languages using the API:

- C
- C#
- C++
- Java
- JavaScript
- Python
- Scala
- Oracle
- SQL Server

- MySQL Server
- PostgreSQL
- MongoDB

In [62]:
```
technologies = [' C ', ' C|', 'C#', 'c#', 'C++', 'c++', 'Java ', 'Java|', 'java|','java ', 'JAVA ', 'JAVA|', 'JavaScript',
                ' SQL Server', ' sql', ' SQL', '|sql', '|SQL', 'MySQL', 'mysql', 'Mysql','PostgreSQL', 'PostGreSQL','postgr

def get_number_of_jobs_TL(technologies, locations):
    final_list = []
    for technology in technologies:
        number_of_jobs_list = [technology]
        for location in locations:
            payload={"Key Skills": technology, "Location": location}
            response=requests.get(api_url, params=payload)
            if response.ok:
                data=response.json()
                number_of_jobs = len(data)
                number_of_jobs_list.append(number_of_jobs)
#         print(number_of_jobs_list)
        final_list.append(number_of_jobs_list)
    return final_list

data = get_number_of_jobs_TL(technologies, locations)
df_data = pd.DataFrame(data, columns=['Technology', 'Los Angeles', 'New York', 'San Francisco', 'Washington DC', 'Seattle',
df_data
```

Out[62]:

	Technology	Los Angeles	New York	San Francisco	Washington DC	Seattle	Austin	Detroit
0	C	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	C|	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	C#	5.0	41.0	3.0	68.0	49.0	5.0	60.0
3	c#	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	C++	3.0	43.0	3.0	55.0	41.0	4.0	32.0
5	c++	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	Java	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	Java|	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	java|	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	java	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	JAVA	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11	JAVA|	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	JavaScript	7.0	51.0	7.0	61.0	52.0	5.0	41.0
13	Javascript	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	javascript	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	Python	24.0	143.0	17.0	258.0	133.0	15.0	170.0
16	Scala	0.0	8.0	0.0	3.0	4.0	1.0	5.0
17	SCALA	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	scala	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	Oracle	17.0	95.0	19.0	143.0	110.0	11.0	115.0
20	oracle	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21	SQL Server	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	sql	NaN	NaN	NaN	NaN	NaN	NaN	NaN
23	SQL	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	|sql	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	|SQL	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26	MySQL	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	mysql	NaN	NaN	NaN	NaN	NaN	NaN	NaN
28	Mysql	NaN	NaN	NaN	NaN	NaN	NaN	NaN
29	PostgreSQL	0.0	1.0	0.0	3.0	1.0	0.0	2.0
30	PostGreSQL	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31	postgresql	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	MongoDB	2.0	25.0	2.0	32.0	21.0	1.0	25.0
33	mongodb	NaN	NaN	NaN	NaN	NaN	NaN	NaN
34	MongoDb	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Author

Ayushi Jain

Other Contributors

Rav Ahuja

Lakshmi Holla

Malika

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-01-19	0.3	Lakshmi Holla	Added changes in the markdown
2021-06-25	0.2	Malika	Updated GitHub job json link
2020-10-17	0.1	Ramesh Sannareddy	Created initial version of the lab

Copyright © 2022 IBM Corporation. All rights reserved.