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

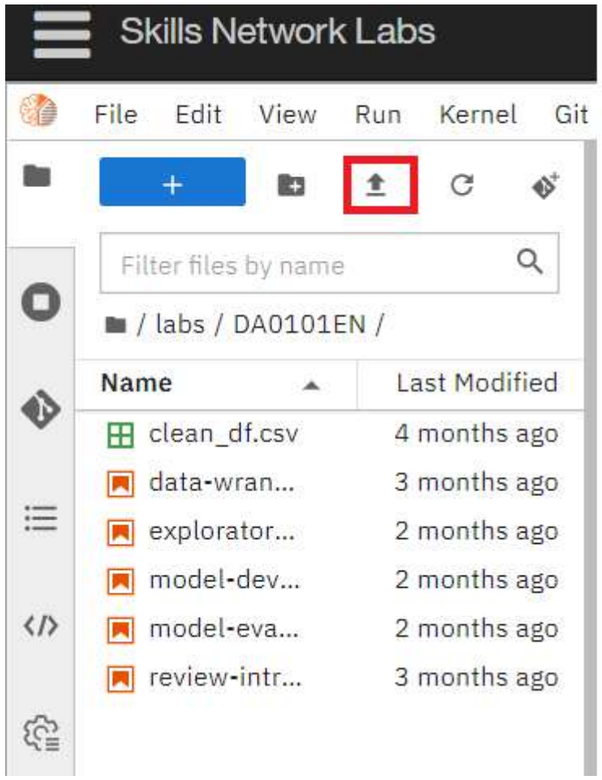
To run the actual lab, firstly you need to click on the Jobs_API notebook link. The file contains flask code which is required to run the Jobs API data.

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



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 [1]:
```
import requests # you need this module to make an API call
import pandas as pd
```

In [2]:
```
api_url = "http://api.open-notify.org/astros.json" # this url gives use the astronaut data
```

In [3]:
```
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 [4]:
```
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 [5]:
```
data
```

Out[5]:
```
{'people': [{'craft': 'ISS', 'name': 'Mark Vande Hei'},
  {'craft': 'ISS', 'name': 'Pyotr Dubrov'},
  {'craft': 'ISS', 'name': 'Anton Shkaplerov'},
  {'craft': 'Shenzhou 13', 'name': 'Zhai Zhigang'},
  {'craft': 'Shenzhou 13', 'name': 'Wang Yaping'},
  {'craft': 'Shenzhou 13', 'name': 'Ye Guangfu'},
  {'craft': 'ISS', 'name': 'Raja Chari'},
  {'craft': 'ISS', 'name': 'Tom Marshburn'},
  {'craft': 'ISS', 'name': 'Kayla Barron'},
  {'craft': 'ISS', 'name': 'Matthias Maurer'}],
 'message': 'success',
 'number': 10}
```

In [6]:
```
print(data)   # print the data just to check the output or for debugging
```

```
{'people': [{'craft': 'ISS', 'name': 'Mark Vande Hei'}, {'craft': 'ISS', 'name': 'Pyotr Dubrov'}, {'craft': 'ISS', 'name': 'Anton Shkaplerov'}, {'craft': 'Shenzhou 13', 'name': 'Zhai Zhigan
g'}, {'craft': 'Shenzhou 13', 'name': 'Wang Yaping'}, {'craft': 'Shenzhou 13', 'name': 'Ye Guangfu'}, {'craft': 'ISS', 'name': 'Raja Chari'}, {'craft': 'ISS', 'name': 'Tom Marshburn'}, {'craf
t': 'ISS', 'name': 'Kayla Barron'}, {'craft': 'ISS', 'name': 'Matthias Maurer'}], 'message': 'success', 'number': 10}
```

Print the number of astronauts currently on ISS.

In [11]:
```
print(data.get('number'))

print(data.get('people'))
```

```
10
[{'craft': 'ISS', 'name': 'Mark Vande Hei'}, {'craft': 'ISS', 'name': 'Pyotr Dubrov'}, {'craft': 'ISS', 'name': 'Anton Shkaplerov'}, {'craft': 'Shenzhou 13', 'name': 'Zhai Zhigang'}, {'craft':
'Shenzhou 13', 'name': 'Wang Yaping'}, {'craft': 'Shenzhou 13', 'name': 'Ye Guangfu'}, {'craft': 'ISS', 'name': 'Raja Chari'}, {'craft': 'ISS', 'name': 'Tom Marshburn'}, {'craft': 'ISS', 'nam
e': 'Kayla Barron'}, {'craft': 'ISS', 'name': 'Matthias Maurer'}]
```

Print the names of the astronauts currently on ISS.

In [8]:
```
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'))
```

```
There are 10 astronauts on ISS
And their names are :
Mark Vande Hei
Pyotr Dubrov
Anton Shkaplerov
Zhai Zhigang
Wang Yaping
Ye Guangfu
Raja Chari
Tom Marshburn
Kayla Barron
Matthias Maurer
```

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 [3]:
```
#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 [4]:
```
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:
        data=response.json()
```

```
#         print(data)
        number_of_jobs = len(data)

    return technology,number_of_jobs
```

Calling the function for Python and checking if it works.

In [10]:
```
get_number_of_jobs_T("Python")
```

Out[10]:
```
('Python', 1173)
```

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

In [11]:
```
def get_number_of_jobs_L(location):
    payload={"Location":location}
    response=requests.get(api_url, params=payload)
    if response.ok:
        data=response.json()
#         print(data)
        number_of_jobs = len(data)

    return location,number_of_jobs
```

In [12]:
```
get_number_of_jobs_L("Los Angeles")
```

Out[12]:
```
('Los Angeles', 640)
```

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

In [13]:
```
locations=["Los Angeles", "New York", "San Francisco", "Washington DC", "Seattle"]

def get_number_of_jobs_Loc_list(locations):
    number_of_jobs_list = []
    for location in locations:
        payload={"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({location: number_of_jobs})

    return number_of_jobs_list
```

In [14]:
```
#your code goes here
get_number_of_jobs_Loc_list(locations)
```

Out[14]:
```
[{'Los Angeles': 640},
 {'New York': 3226},
 {'San Francisco': 435},
 {'Washington DC': 5316},
 {'Seattle': 3375}]
```

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 [15]:
```
#your code goes here
countries = ['Los Angeles', 'New York', 'San Francisco', 'Washington DC', 'Seattle', 'Austin', 'Detroit']
```

Import libraries required to create excel spreadsheet

In [19]:
```
# your code goes here
from openpyxl import Workbook
```

Create a workbook and select the active worksheet

In [57]:
```
# your code goes here
wb=Workbook()
ws=wb.active
ws.append(countries)
```

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 [16]:
```
#your code goes here
def get_number_of_jobs_TL(technology, countries):
    number_of_jobs_list = []
    for location in countries:
        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", countries)
```

Out[16]: [24, 143, 17, 258, 133, 15, 170]

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

In [14]:
```
#your code goes here
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 [22]:

```
# your code goes here
technologies = [' C ', ' C|', 'C#', 'c#', 'C++', 'c++', 'Java ', 'Java|', 'java|','java ', 'JAVA ', 'JAVA|', 'JavaScript', 'Javascript','javascript', 'Python', 'Scala', 'SCALA', 'scala', 'Orac
                ' SQL Server', ' sql', ' SQL', '|sql', '|SQL', 'MySQL', 'mysql', 'Mysql','PostgreSQL', 'PostGreSQL','postgresql','MongoDB', 'mongodb', 'MongoDb']

def get_number_of_jobs_TL(technologies, countries):
    final_list = []
    for technology in technologies:
        number_of_jobs_list = [technology]
        for location in countries:
            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

get_number_of_jobs_TL(technologies, countries)
```

Out[22]:

```
[[' C ', 0, 0, 0, 1, 1, 0, 1],
 [' C|', 9, 68, 12, 111, 62, 18, 71],
 ['C#', 5, 41, 3, 68, 49, 5, 60],
 ['c#', 2, 21, 6, 40, 25, 4, 27],
 ['C++', 3, 43, 3, 55, 41, 4, 32],
 ['c++', 4, 24, 5, 37, 22, 4, 27],
 ['Java ', 5, 12, 2, 21, 17, 2, 11],
 ['Java|', 17, 121, 12, 209, 132, 9, 143],
 ['java|', 15, 55, 7, 83, 63, 5, 71],
 ['java ', 0, 3, 0, 7, 6, 1, 10],
 ['JAVA ', 0, 0, 0, 1, 0, 0, 0],
 ['JAVA|', 0, 4, 0, 7, 5, 0, 4],
 ['JavaScript', 7, 51, 7, 61, 52, 5, 41],
 ['Javascript', 26, 183, 22, 290, 192, 19, 189],
 ['javascript', 14, 52, 7, 76, 57, 5, 63],
 ['Python', 24, 143, 17, 258, 133, 15, 170],
 ['Scala', 0, 8, 0, 3, 4, 1, 5],
 ['SCALA', 0, 6, 1, 6, 7, 2, 13],
 ['scala', 1, 5, 0, 12, 1, 0, 7],
 ['Oracle', 17, 95, 19, 143, 110, 11, 115],
 ['oracle', 6, 18, 0, 21, 16, 2, 16],
 [' SQL Server', 2, 34, 2, 47, 26, 5, 28],
 [' sql', 14, 55, 7, 72, 45, 4, 46],
 [' SQL', 28, 214, 22, 333, 197, 30, 216],
 ['|sql', 0, 4, 0, 6, 1, 0, 3],
 ['|SQL', 1, 8, 1, 22, 9, 0, 16],
 ['MySQL', 20, 93, 20, 132, 97, 13, 93],
 ['mysql', 2, 25, 4, 29, 24, 4, 30],
```

```
['Mysql', 0, 0, 0, 0, 2, 0, 1],
['PostgreSQL', 0, 1, 0, 3, 1, 0, 2],
['PostGreSQL', 0, 0, 0, 0, 1, 0, 0],
['postgresql', 0, 1, 1, 3, 3, 0, 1],
['MongoDB', 2, 25, 2, 32, 21, 1, 25],
['mongodb', 2, 4, 0, 3, 3, 0, 5],
['MongoDb', 0, 1, 0, 1, 0, 0, 1]]
```

In [23]:

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

Out[23]:

	Technology	Los Angeles	New York	San Francisco	Washington DC	Seattle	Austin	Detroit
0	C	0	0	0	1	1	0	1
1	C|	9	68	12	111	62	18	71
2	C#	5	41	3	68	49	5	60
3	c#	2	21	6	40	25	4	27
4	C++	3	43	3	55	41	4	32
5	c++	4	24	5	37	22	4	27
6	Java	5	12	2	21	17	2	11
7	Java|	17	121	12	209	132	9	143
8	java|	15	55	7	83	63	5	71
9	java	0	3	0	7	6	1	10
10	JAVA	0	0	0	1	0	0	0
11	JAVA|	0	4	0	7	5	0	4
12	JavaScript	7	51	7	61	52	5	41
13	Javascript	26	183	22	290	192	19	189
14	javascript	14	52	7	76	57	5	63
15	Python	24	143	17	258	133	15	170
16	Scala	0	8	0	3	4	1	5
17	SCALA	0	6	1	6	7	2	13
18	scala	1	5	0	12	1	0	7
19	Oracle	17	95	19	143	110	11	115
20	oracle	6	18	0	21	16	2	16
21	SQL Server	2	34	2	47	26	5	28
22	sql	14	55	7	72	45	4	46
23	SQL	28	214	22	333	197	30	216
24	|sql	0	4	0	6	1	0	3
25	|SQL	1	8	1	22	9	0	16
26	MySQL	20	93	20	132	97	13	93
27	mysql	2	25	4	29	24	4	30

	Technology	Los Angeles	New York	San Francisco	Washington DC	Seattle	Austin	Detroit
28	Mysql	0	0	0	0	2	0	1
29	PostgreSQL	0	1	0	3	1	0	2
30	PostGreSQL	0	0	0	0	1	0	0
31	postgresql	0	1	1	3	3	0	1
32	MongoDB	2	25	2	32	21	1	25
33	mongodb	2	4	0	3	3	0	5
34	MongoDb	0	1	0	1	0	0	1

In [62]:
```
file_name = "job-postings.xlsx"
df_data.to_excel(file_name)
print("Dataframe is written to Excel")
```

```
Dataframe is written to Excel
```

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