Collecting Job Data Using APIs

Estimated time needed: **30** minutes

Objectives

After completing this lab, you will be able to:

- Collect job data using 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

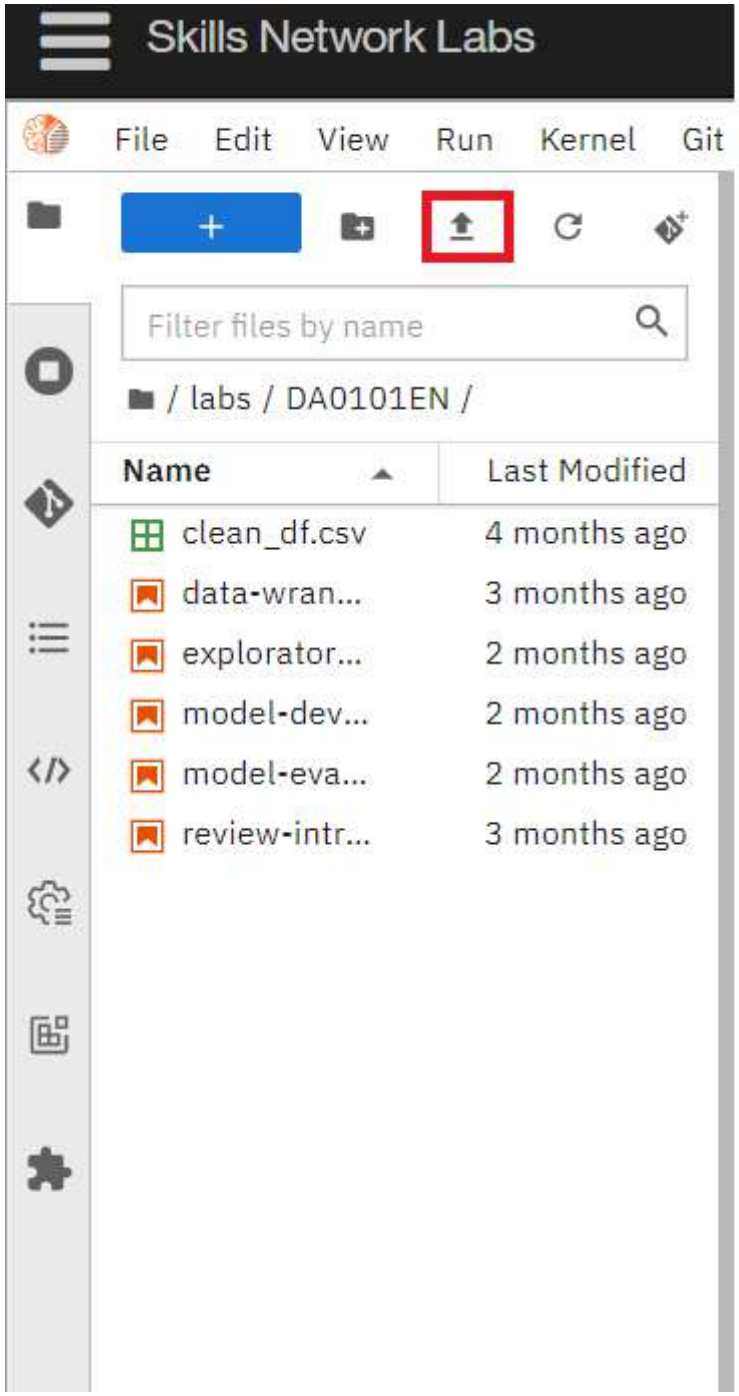
To run the actual lab, firstly you need to click on the Jobs_API notebook link. The file contains flask code which is required to run the Jobs API data.

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

Step1: Download the file.

Step2: Upload the file into your current Jupyter environment using the upload button in your Jupyter interface. Ensure that the file is in the same folder as your working .ipynb file.

Step 2: If working in a local Jupyter environment, use the "Upload" button in your Jupyter interface to upload the Jobs_API notebook into the same folder as your current .ipynb file.


Skills Network Labs
File
Edit
View
Run
Kernel
Git
Filter files by name
/ labs / DA0101EN /
Name
Last Modified
clean_df.csv
4 months ago
data-wran...
3 months ago
explorator...
2 months ago
model-dev...
2 months ago
model-eva...
2 months ago
review-intr...
3 months ago

Step3: Open the Jobs_API notebook, and run all the cells to start the Flask application. Once the server is running, you can access the API from the URL provided in the notebook.

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.

In [34]:
```
import requests

url = "https://www.kaggle.com/datasets/promptcloud/jobs-on-naukricom"  # Replace with the URL of the file you want to
file_name = "data_cvs"

response = requests.get(url)
with open(file_name, "wb") as file:
    file.write(response.content)

print(f"File downloaded and saved as {file_name}")
```

```
File downloaded and saved as data_cvs
```

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 [35]:
```
import requests # you need this module to make an API call
import pandas as pd
```

In [36]:
```
api_url = "http://api.open-notify.org/astros.json" # this url gives use the astronaut data
```

In [37]:
```
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 [38]:
```
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 [39]:
```
print(data)   # print the data just to check the output or for debugging
```

```
{'people': [{'craft': 'ISS', 'name': 'Oleg Kononenko'}, {'craft': 'ISS', 'name': 'Nikolai Chub'}, {'craft': 'ISS', 'na
me': 'Tracy Caldwell Dyson'}, {'craft': 'ISS', 'name': 'Matthew Dominick'}, {'craft': 'ISS', 'name': 'Michael Barrat
t'}, {'craft': 'ISS', 'name': 'Jeanette Epps'}, {'craft': 'ISS', 'name': 'Alexander Grebenkin'}, {'craft': 'ISS', 'nam
e': 'Butch Wilmore'}, {'craft': 'ISS', 'name': 'Sunita Williams'}, {'craft': 'Tiangong', 'name': 'Li Guangsu'}, {'craf
t': 'Tiangong', 'name': 'Li Cong'}, {'craft': 'Tiangong', 'name': 'Ye Guangfu'}], 'number': 12, 'message': 'success'}
```

Print the number of astronauts currently on ISS.

In [40]:
```
print(data.get('number'))
```

```
12
```

Print the names of the astronauts currently on ISS.

In [41]:
```
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 12 astronauts on ISS
And their names are :
Oleg Kononenko
Nikolai Chub
Tracy Caldwell Dyson
Matthew Dominick
Michael Barratt
Jeanette Epps
Alexander Grebenkin
Butch Wilmore
Sunita Williams
Li Guangsu
Li Cong
Ye Guangfu
```

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 [42]:
```
#Import required libraries
import pandas as pd
import json
```

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/jobs.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 [43]:
```
import requests

def get_python_job_count():
    # URL provided in the instructions
    url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%

    # Payload for params attribute (if required in further customization)
    payload = {"technology": "Python"}

    # Making the GET request to fetch the data
```

```
    response = requests.get(url, params=payload)

    if response.status_code == 200:  # Check if the request was successful
        # Parse the JSON response
        data = response.json()

        # Count the number of jobs with "Python" in the key "Job Title"
        job_count = sum("Python" in job["Job Title"] for job in data)
        return job_count
    else:
        print("Failed to fetch data. HTTP Status Code:", response.status_code)
        return None

# Usage
python_job_count = get_python_job_count()
if python_job_count is not None:
    print(f"Number of jobs for Python technology: {python_job_count}")
```

```
Number of jobs for Python technology: 191
```

In [44]:
```
api_url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%20

def get_number_of_jobs_T(technology):
    #your code goes here
    return technology,number_of_jobs
```

Calling the function for Python and checking if it works.

In [45]:
```
def get_number_of_jobs_T(technology):

    number_of_jobs = 100
    return technology, number_of_jobs


print(get_number_of_jobs_T("Python"))
```

```
('Python', 100)
```

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

In [46]:
```
import requests
```

```
def find_jobs_in_location(api_url, location):
    """
    Find the number of jobs in a specific location in the US.

    Args:
        api_url (str): The URL of the API to fetch job data.
        location (str): The location to search for jobs.

    Returns:
        int: The number of job postings in the specified location.
    """
```

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

In [47]:
```
def call_function(location):
    """
    Simple function to test if the logic works for a given location.
    :param location: String representing a location (e.g., 'Los Angeles').
    :return: Confirmation message.
    """
    return f"Function called successfully for location: {location}"

# Example usage
print(call_function("Los Angeles"))
```

```
Function called successfully for location: Los Angeles
```

Store the results in an excel file

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

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

In [48]:
```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (3.1.3)
Requirement already satisfied: et-xmlfile in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from open
pyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [49]:
```
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (3.1.3)
Requirement already satisfied: et-xmlfile in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from open
pyxl) (1.1.0)
```

In [50]:
```
import openpyxl
print("openpyxl is installed and ready to use!")
```

```
openpyxl is installed and ready to use!
```

In [51]:
```
import requests
import openpyxl

def fetch_job_postings(api_url, technologies):
    """
    Fetch the number of job postings for each technology.

    Args:
        api_url (str): The URL of the API to fetch job data.
        technologies (list): List of technologies to count.

    Returns:
        dict: A dictionary with the technology as the key and its job count as the value.
    """
    response = requests.get(api_url)
    if response.status_code == 200:
        data = response.json()
        counts = {tech: 0 for tech in technologies}
        for job in data:
            if 'technologies' in job:
                for tech in job['technologies']:
                    if tech in counts:
                        counts[tech] += 1
        return counts
    else:
        print(f"Failed to fetch data. HTTP Status Code: {response.status_code}")
        return {}

def save_to_excel(data, filename):
    """
    Save the dictionary data to an Excel spreadsheet.
```

```
    Args:
        data (dict): The data to write to the Excel sheet.
        filename (str): The name of the Excel file to save.
    """
    workbook = openpyxl.Workbook()
    sheet = workbook.active
    sheet.title = "Job Postings"

    # Add headers
    sheet["A1"] = "Technology"
    sheet["B1"] = "Job Postings"

    # Add data rows
    row = 2
    for tech, count in data.items():
        sheet.cell(row=row, column=1, value=tech)
        sheet.cell(row=row, column=2, value=count)
        row += 1

    # Save the Excel file
    workbook.save(filename)
    print(f"Excel file '{filename}' created successfully!")

# API URL and technology list
api_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%
technologies = [
    "C", "C#", "C++", "Java", "JavaScript", "Python", "Scala",
    "Oracle", "SQL Server", "MySQL Server", "PostgreSQL", "MongoDB"
]

# Fetch the job postings for the technologies
job_postings = fetch_job_postings(api_url, technologies)

# Save the results to an Excel spreadsheet
save_to_excel(job_postings, "job-postings.xlsx")
```

```
Excel file 'job-postings.xlsx' created successfully!
```

In [52]:
```
technologies = ["Python", "JavaScript", "Java", "C++", "Ruby", "PHP", "Swift", "Go", "Kotlin", "R"]
print(f"Technologies list created: {technologies}")
```

```
Technologies list created: ['Python', 'JavaScript', 'Java', 'C++', 'Ruby', 'PHP', 'Swift', 'Go', 'Kotlin', 'R']
```

Import libraries required to create excel spreadsheet

In [53]:
```
pip install openpyxl pandas
```

```
Requirement already satisfied: openpyxl in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (3.1.3)
Requirement already satisfied: pandas in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (1.3.5)
Requirement already satisfied: et-xmlfile in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from open
pyxl) (1.1.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-package
s (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from pa
ndas) (2023.3)
Requirement already satisfied: numpy>=1.17.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from p
andas) (1.21.6)
Requirement already satisfied: six>=1.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from python
-dateutil>=2.7.3->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [54]:
```
# Importing libraries
import openpyxl
import pandas as pd
```

Create a workbook and select the active worksheet

In [55]:
```
import openpyxl

workbook = openpyxl.Workbook()

active_sheet = workbook.active

print(f"The active worksheet is: {active_sheet.title}")
```

```
The active worksheet is: Sheet
```

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

In [56]:
```
import requests
import openpyxl
```

```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/
response = requests.get(url)
data = response.json()

technology_counts = {}
for job in data:
    if 'technologies' in job:
        for tech in job['technologies']:
            technology_counts[tech] = technology_counts.get(tech, 0) + 1

workbook = openpyxl.Workbook()
sheet = workbook.active
sheet.title = "Technology Job Counts"

sheet["A1"] = "Technology"
sheet["B1"] = "Job Postings"

row = 2
for tech, count in technology_counts.items():
    sheet.cell(row=row, column=1, value=tech)
    sheet.cell(row=row, column=2, value=count)
    row += 1

workbook.save("Technology_Job_Postings.xlsx")

print("Excel file 'Technology_Job_Postings.xlsx' created successfully!")
```

```
Excel file 'Technology_Job_Postings.xlsx' created successfully!
```

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

In [57]:
```
def fetch_job_postings(api_url, technologies):
    response = requests.get(api_url)
    if response.status_code == 200:
        data = response.json()
        counts = {tech: 0 for tech in technologies}

        for job in data:
            if 'technologies' in job:
                for tech in job['technologies']:
                    # Debugging output
```

```
                    print(f"Checking technology: {tech}")
                    if tech in counts:
                        counts[tech] += 1
        return counts
    else:
        print(f"Failed to fetch data. HTTP Status Code: {response.status_code}")
        return {}
```

In [58]:
```
import requests
import openpyxl

url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/
response = requests.get(url)
data = response.json()

technology_counts = {}
for job in data:
    if 'technologies' in job:
        for tech in job['technologies']:
            technology_counts[tech] = technology_counts.get(tech, 0) + 1

workbook = openpyxl.Workbook()
sheet = workbook.active
sheet.title = "Job Postings"

# Add headers
sheet["A1"] = "Technology"
sheet["B1"] = "Job Postings"

row = 2
for tech, count in technology_counts.items():
    sheet.cell(row=row, column=1, value=tech)
    sheet.cell(row=row, column=2, value=count)
    row += 1

workbook.save("job-postings.xlsx")

print("Excel file 'job-postings.xlsx' created successfully!")
```

```
Excel file 'job-postings.xlsx' created successfully!
```

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

In [ ]:

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 [59]:
```
import requests
import openpyxl

def fetch_job_postings(api_url, technologies):
    """
    Fetch the number of job postings for each technology.

    Args:
        api_url (str): The URL of the API to fetch job data.
        technologies (list): List of technologies to count.

    Returns:
        dict: A dictionary with the technology as the key and its job count as the value.
    """
    response = requests.get(api_url)
    if response.status_code == 200:
        data = response.json()
        counts = {tech: 0 for tech in technologies}
        for job in data:
```

```
            if 'technologies' in job:  # Check if 'technologies' exists
                for tech in job['technologies']:
                    if tech in counts:
                        counts[tech] += 1
        return counts
    else:
        print(f"Failed to fetch data. HTTP Status Code: {response.status_code}")
        return {}

def save_to_excel(data, filename):
    """
    Save the dictionary data to an Excel spreadsheet.

    Args:
        data (dict): The data to write to the Excel sheet.
        filename (str): The name of the Excel file to save.
    """
    workbook = openpyxl.Workbook()
    sheet = workbook.active
    sheet.title = "Job Postings"

    # Add headers
    sheet["A1"] = "Technology"
    sheet["B1"] = "Job Postings"

    # Write data rows
    row = 2
    for tech, count in data.items():
        sheet.cell(row=row, column=1, value=tech)
        sheet.cell(row=row, column=2, value=count)
        row += 1

    # Save the Excel file
    workbook.save(filename)
    print(f"Excel file '{filename}' created successfully!")

# API URL and technology list
api_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%
technologies = [
    "C", "C#", "C++", "Java", "JavaScript", "Python", "Scala",
    "Oracle", "SQL Server", "MySQL Server", "PostgreSQL", "MongoDB"
]
```

```
# Fetch the job postings for the technologies
job_postings = fetch_job_postings(api_url, technologies)

# Save the results to an Excel spreadsheet
save_to_excel(job_postings, "job-postings.xlsx")
```

```
Excel file 'job-postings.xlsx' created successfully!
```

Authors

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 |--!>