Jobs API

In [ ]:
```
!pip install flask
```

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

In [ ]:
```
import flask
from flask import request, jsonify
import requests
import re

def get_data(key,value,current):
    results = list()
    pattern_dict = {
        'C'      : '(C)',
        'C++'    : '(C\+\+)',
        'Java'   :'(Java)',
        'C#'     : '(C\#)',
        'Python' :'(Python)',
        'Scala' : '(Scala)',
        'Oracle' : '(Oracle)',
        'SQL Server': '(SQL Server)',
        'MySQL Server' :'(MySQL Server)',
        'PostgreSQL':'(PostgreSQL)',
        'MongoDB'    : '(MongoDB)',
        'JavaScript'    : '(JavaScript)',
        'Los Angeles' :'(Los Angeles)',
        'New York':'(New York)',
        'San Francisco':'(San Francisco)',
        'Washington DC':'(Washington DC)',
        'Seattle':'(Seattle)',
        'Austin':'(Austin)',
        'Detroit':'(Detroit)',





    }
    for rec in current:
        print(rec[key])
        print(type(rec[key]))
        print(rec[key].find(value))
        #if rec[key].find(value) != -1:
        import re
        #reex_str = """(C)|(C\+\+)|(JavaScript)|(Java)|(C\#)|(Python)|(Scala)|(Oracle)|(SQL Server)|(MySQL Server)|(PostgreSQL)|(MongoDB)"""
        if re.search(pattern_dict[value],rec[key]) != None:
            results.append(rec)
    return results

app = flask.Flask(__name__)

import json
data = None
with open('jobs.json',encoding='utf-8') as f:
    # returns JSON object as
    # a dictionary
    data = json.load(f)
```

```
@app.route('/', methods=['GET'])
def home():

    return '''<h1>Welcome to flask JOB search API</p>'''


@app.route('/data/all', methods=['GET'])
def api_all():
    return jsonify(data)


@app.route('/data', methods=['GET'])
def api_id():
    # Check if keys such as Job Title,KeySkills, Role Category and others  are provided as part of the URL.
    #  Assign the keys to the corresponding variables..
    # If no key is provided, display an error in the browser.
    res = None
    for req in request.args:

        if req == 'Job Title':
            key = 'Job Title'
        elif req == 'Job Experience Required' :
            key='Job Experience Required'
        elif req == 'Key Skills' :
            key='Key Skills'

        elif req == 'Role Category' :
            key='Role Category'
        elif req == 'Location' :
            key='Location'

        elif req == 'Functional Area' :
            key='Functional Area'

        elif req == 'Industry' :
            key='Industry'
        elif req == 'Role' :
            key='Role'
        elif req=="id":
             key="id"
        else:
            pass

        value = request.args[key]
        if (res==None):
            res = get_data(key,value,data)
        else:
            res = get_data(key,value,res)
```

```
    # Use the jsonify function from Flask to convert our list of
    # Python dictionaries to the JSON format.
    return jsonify(res)

app.run()
```

Collect Jobs Data using Jobs API

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

In [ ]:
```
api_url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/jobs
d = requests.get(api_url)
data = d.json()
def get_number_of_jobs_T(technology):

    #your code goes here
    number_of_jobs=0
    for i in data:
        if(technology in i['Key Skills']):
            number_of_jobs+=1


    return technology,number_of_jobs
```

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

In [ ]:
```
def get_number_of_jobs_L(location):

    #your coe goes here
    number_of_jobs=0
    for i in data:
        if(location in i['Location']):
            number_of_jobs+=1

    return location,number_of_jobs
```

In [ ]:
```
print(get_number_of_jobs_L('Los Angeles'))
```

In [ ]:
```
d={}
for i in data:
    d[i['Location']] = d.get(i['Location'],0)+1
location_list = list(d.keys()
```

In [ ]:
```
!pip install openpyxl
from openpyxl import Workbook
```

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

In [ ]:
```
ws.append(['Locations','Number_of_Jobs'])

for i in location_list:
    ws.append(get_number_of_jobs_L(i))
```

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

In [ ]:
```
import pandas as pd
pd.read_excel("job-postings.xlsx")
```

In [ ]:
```
tech_list=['C','C#','C++','Java','JavaScript','Python','Scala','Oracle','SQL Server','MySQL Server','PostgreSQL','MongoDB']
wb_1=Workbook()
ws_1=wb.active
ws_1.append(['Technology','Number_of_Jobs'])

for i in tech_list:
    ws_1.append(get_number_of_jobs_T(i))
wb_1.save("techbased_job-postings.xlsx")
```

In [ ]:
```
#this url contains the data you need to scrape
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/Programming_Languages.html"
```

Web Scraping

In [ ]:
```
import requests as rq
from bs4 import BeautifulSoup as bs
```

In [ ]:
```
es = rq.get('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/Programming_Languages.html').text
```

In [ ]:
```
parsed = bs(res,'html5lib')
```

In [ ]:
```
lan=[]
avg_sal=[]
for i in parsed.find_all('tr')[1:]:
    l = i.find_all('td')[1].string
    s = i.find_all('td')[3].string
    lan.append(l)
    avg_sal.append(s)
lan,avg_sal
```

In [ ]:
```
import pandas as pd

df = pd.DataFrame({'Language':lan,'Average Annual Salary':avg_sal})
df.to_csv('popular-languages.csv',index=False)
```

Survey Dataset Exploration

In [ ]:
```
import pandas as pd
```

In [ ]:
```
dataset_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m1_survey_data.csv"
```

In [ ]:
```
df = pd.read_csv(dataset_url)
```

In [ ]:
```
df.head()
```

In [ ]:
```
df.shape
```

In [ ]:
```
df.info()
```

In [ ]:
```
df['Age'].mean()
```

In [ ]:
```
df['Country'].nunique()
```

Data Wrangling

In [ ]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m1_survey_data.csv")
```

In [ ]:
```
df.duplicated().sum()
```

In [ ]:
```
df.drop_duplicates(inplace=True)
```

In [ ]:
```
df.duplicated().sum()
```

In [ ]:
```
df.info()
```

In [ ]:
```
df.isna().sum()
```

In [ ]:
```
df['WorkLoc'].isna().sum()
```

In [ ]:
```
df['WorkLoc'].value_counts()
```

In [ ]:
```
maj = df['WorkLoc'].value_counts().index[0]
```

In [ ]:
```
df['WorkLoc'].fillna(maj,inplace=True)
```

In [ ]:
```
df['WorkLoc'].isna().sum()
```

In [ ]:
```
df['CompFreq'].unique()
```

In [ ]:
```
def change(x):

  if(x['CompFreq']=='Yearly'):
    return x['CompTotal']
  elif(x['CompFreq']=='Monthly'):
    return 12*x['CompTotal']
  elif(x['CompFreq']=='Weekly'):
    return 52*x['CompTotal']

df['NormalizedAnnualCompensation'] = df.apply(change,axis=1)
```

In [ ]:
```
df['NormalizedAnnualCompensation'].describe()
```

Exploratory Data Analysis

In [ ]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m2_survey_data.csv")
```

In [ ]:
```
import seaborn as sns
```

In [ ]:
```
sns.displot(df['ConvertedComp'], kde = True)
```

In [ ]:
```
sns.histplot(data=df,x='ConvertedComp')
```

In [ ]:
```
df['ConvertedComp'].median()
```

In [ ]:
```
df['Gender'].value_counts()['Man']
```

In [ ]:
```
df_woman = df[df['Gender']=='Woman']
print("Woman Median: ", df_woman['ConvertedComp'].median())
```

In [ ]:
```
df['Age'].describe()
```

In [ ]:
```
sns.histplot(data=df,x='Age')
```

In [ ]:
```
sns.boxplot(data=df,x='ConvertedComp')
```

In [ ]:
```
Q1,Q3 = df['ConvertedComp'].quantile(.25),df['ConvertedComp'].quantile(.75)
IQR = Q3 - Q1
print('The Inter Quartile Range for ConvertedComp: ', IQR)
```

In [ ]:
```
upper = Q3+(IQR*1.5)
lower = Q1-(IQR*1.5)

print('Upper bound: ', upper)
print('Lower bound: ', lower)
```

In [ ]:
```
(df['ConvertedComp'] < lower) | (df['ConvertedComp'] > upper)
```

In [ ]:
```
df2 = df.loc[(df['ConvertedComp']<=upper)&(df['ConvertedComp']>=lower)]
```

In [ ]:
```
df2.describe()
```

In [ ]:
```
df.corr()['Age']
```

Data Visualization

In [ ]:
```
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m4_survey_data.sqlite
```

In [ ]:
```
import sqlite3
conn = sqlite3.connect("m4_survey_data.sqlite") # open a database connection
```

In [ ]:
```
QUERY = """
SELECT * FROM master
"""
df = pd.read_sql_query(QUERY,conn)
df.hist(column='ConvertedComp')
```

In [ ]:
```
QUERY = """
SELECT * FROM master
"""
df = pd.read_sql_query(QUERY,conn)
df.boxplot(column='Age')
```

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

QUERY = """
SELECT * FROM master
"""
df = pd.read_sql_query(QUERY,conn)
plot = sns.scatterplot(x='Age', y='WorkWeekHrs', data=df)
```

In [ ]:
```
QUERY = """
SELECT WorkWeekHrs, CodeRevHrs, Age FROM master
"""
df1=pd.read_sql_query(QUERY,conn)

sns.scatterplot(data=df1, x='WorkWeekHrs', y='CodeRevHrs', size='Age', hue='Age', alpha=0.5, sizes=(10, 500))

plt.title('WorkWeekHrs and CodeRevHrs By Age', size=14)
plt.xlabel('WorkWeekHrs', size=10)
plt.ylabel('CodeRevHrs', size=10)

plt.show()
```

In [ ]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt

QUERY = """
SELECT DevType, COUNT(*) as count
from DevType
group by DevType
order by count(DevType) DESC LIMIT 5
"""
df=pd.read_sql_query(QUERY,conn)
df.set_index('DevType', inplace=True)

colors_list=['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'lightgreen', 'pink']

df['count'].plot(kind='pie', figsize=(20,6), autopct='%1.1f%%', labels=None, startangle=90, colors=colors_list, shadow=True, pctdistance=1.12)

plt.legend(labels=df.index, loc='upper right')
plt.title('Top 5 DevType Respondents Wish To Learn')
plt.axis('equal')
plt.show()
```

In [ ]:
```
QUERY = """
SELECT WorkWeekHrs, CodeRevHrs, Age FROM master
WHERE Age BETWEEN 30 AND 35
"""
df = pd.read_sql_query(QUERY,conn)
df1 = df.groupby('Age').median()

df1.plot(kind='bar', figsize=(10, 6), stacked=True)

plt.title('Stacked Bar Chart of Median WorkWeekHrs and CodeRevHrs for Those Age 30 to 35')
plt.show()
```

In [ ]:
```
QUERY = """
SELECT ConvertedComp, Age FROM master
WHERE Age BETWEEN 25 AND 30
"""
df = pd.read_sql_query(QUERY,conn)
df1 = df.groupby('Age').median()

df1.plot(kind='line', figsize=(20, 6))

plt.title('Median ConvertedComp for Those Age 25 to 30')
plt.ylabel('ConvertedComp')
plt.show()
```

In [ ]:
```
QUERY = """
SELECT MainBranch, COUNT(*) as MainBranch
from master
group by MainBranch
"""

df=pd.read_sql_query(QUERY,conn)

df.plot(kind='barh', figsize=(10,6), color='lightskyblue')
plt.xlabel('Count')
plt.ylabel('MainBranch')
plt.show()
```

In [ ]:
```
conn.close()
```