ABSTRACT

Contained below is a functional script that webscrapes wikipedia and Teamwork Online - the best sports database available. The script is as follows:

First the libraries are loaded for scraping and data cleaning. Then Teamwork Online is scraped through a variety of user -defined functions and passed into the list: job_list.

This list is then made into a dataframe and validated through a series of agile development cycles which included visualizing the data table at each step. In the end the final table is saved into job_posting_teamwork_df. Cleaning included many partions, replacements, typecasting and reindexing as well as other steps.

Some of the data was passed into other dataframes such as job_requirements_df_final which contains an exploded list of job requirements and qualifications scraped from Teamwork Online. Another dataframe made was called Company_Team_df and contained the distinct companies and an encoded ID number.

Further scraping came into play when all major leagues' (MLS, MLB, NFL, NHL, and NBA) wiki pages were scraped to get all team information. This data was then cleaned and merged with the actual companies so that those that did have a team match would have that info. Many NULLS occured and were cleaned as well as possible.

Finally, the database was connected to and all data was succesfully imported.

In [ ]:

```
import pandas as pd
import numpy as np
from datetime import datetime
from lxml import html

import requests
from bs4 import BeautifulSoup
#!pip install requests_html
#from requests_html import HTMLSession
import random
import re
#from nltk import bigrams
#from nltk.corpus import stopwords
#from nltk.stem import WordNetLemmatizer
#from nltk.tokenize import word_tokenize
import string
```

```
import matplotlib as mlt
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.preprocessing import LabelEncoder

import pymysql
pymysql.install_as_MySQLdb()
import MySQLdb

#! pip install wordcloud
#from subprocess import check_output
#from wordcloud import WordCloud, STOPWORDS
```

In [ ]:

```
def merge(dict1, dict2):
    return(dict2.update(dict1))

def extract(page):
    url = f'https://www.teamworkonline.com/jobs-in-sports?page={page}'
    user_agents_list = [
    'Mozilla/5.0 (iPad; CPU OS 12_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile/15E148',
    'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.83 Safari/53
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36'
    ]

    headers = {'User-Agent': random.choice(user_agents_list)}

    r = requests.get(url,headers)
    soup = BeautifulSoup(r.content,'html.parser')
    return(soup)

def extract_inner(link_ext):
    url_inner = 'https://www.teamworkonline.com' + link_ext

    user_agents_list = [
    'Mozilla/5.0 (iPad; CPU OS 12_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile/15E148',
    'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.83 Safari/53
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36'
    ]

    headers = {'User-Agent': random.choice(user_agents_list)}

    r_inner = requests.get(url_inner,headers)
    soup_inner = BeautifulSoup(r_inner.content,'html.parser')
```

```
    return(soup_inner)

def transform(soup):
    divs = soup.find_all('div',class_ = 'result-item recent-job')
    for job in divs:
        title = job.find('h3',class_='base-font').text.strip()
        job_exception = job.find_all('span',class_ = 'icon-bullet__content icon-bullet__content--recent-job-card')
        for i in job_exception:

            if i.text.endswith('Jobs'):
                company_temp = i.text.replace(' Jobs','')
            else:
                location_temp = i.text.replace('Jobs in ',' (')

        link_ext = job.a['href']

        #details = []
        more_info = extract_inner(link_ext)
        try:
            divs_inner_1 = more_info.find('div',class_ = 'opportunity-preview__body').find_all('ul')
            details = []

            for info in divs_inner_1:
                for i in (info.find_all('li')):
                    details.append(i.text.strip())

        except:
            details= []

        try:
            full_job = (more_info.find('h1',class_ = 'opportunity-preview__title').text)
        except:
            full_job = (title + '-' + company_temp + location_temp + ')')
            # Up until - is job, after dash to ( is company and (INSIDE parenthese is location)

        job = {
            'title': title,
            'job_info': full_job,
            'url': 'https://www.teamworkonline.com' + link_ext,
            'details': details,
            'scrape_datetime': datetime.now().strftime("%m/%d/%Y %H:%M:%S")
        }

        joblist.append(job)
```

```
    return

joblist = []
```

In [ ]:

```
pages = ((1,3),(3,5),(5,7),(7,9),(9,11))
for i in pages:
    for j in range(i[0],i[1]):
        c=extract(j)
        transform(c)
```

In [ ]:

```
database = MySQLdb.connect(host="localhost" , user="root" , passwd="Pps11844")
cursor = database.cursor()

def execute_query(query_statement):
    try:
        cursor.execute(query_statement);
        database.commit();
        print("Data is Succefully Inserted")

    except Exception as e :
        database.rollback();
        print("The  Exception Occured : ", e)

execute_query("USE JobsinSports")
```

In [ ]:

```
SQL_df_posting = pd.read_sql('select * from job_posting',database)
```

In [ ]:

```
SQL_df_companies = pd.read_sql('select * from company_team',database)
```

In [ ]:

```
cursor.execute("SELECT MAX(company_ID) FROM company_team;")
result = cursor.fetchone();
max_comp_ID = result[0]
```

In [ ]:

```
cursor.execute("SELECT MAX(job_ID) FROM job_posting;")
result2 = cursor.fetchone();
max_job_ID = result2[0]
```

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

In [ ]:
```
SQL_df_companies
```

In [ ]:
```
############################################## USED FOR INITIAL SCRAPE ##################################################
# Creating and cleaning job data table
job_posting_teamwork = pd.DataFrame(joblist)

for i,j in job_posting_teamwork.iterrows():
    if j['title'] in (j['job_info']):
        j['job_info'] = j['job_info'].replace(j['title'],'')

job_posting_teamwork["Location"] = (job_posting_teamwork["job_info"].str.partition("(")[2]).str.replace(")","").str.repla
job_posting_teamwork["Company"] = job_posting_teamwork["job_info"].str.partition("(")[0].str.partition("-")[2].str.strip(

job_posting_teamwork['job_city'] = job_posting_teamwork['Location'].str.partition(",")[0]
job_posting_teamwork['job_state'] = job_posting_teamwork['Location'].str.partition(",")[2]

job_posting_teamwork = job_posting_teamwork.drop(["job_info","Location"],axis=1)
for i,j in job_posting_teamwork.iterrows():
    if(j["Company"] == "Oakland A's"):
        j["Company"] = "Oakland Athletics"

    elif(j["Company"] == "NYCFC"):
            j["Company"] = "New York City FC"

    else:
        pass

for i,j in job_posting_teamwork.iterrows():
    if((j['title'] in SQL_df_posting['job_title'].values) and (j['Company'] in SQL_df_companies['company_name'].values)):
        job_posting_teamwork = job_posting_teamwork.drop(index = i,axis = 1)
    else:
        pass

number = LabelEncoder()

job_posting_teamwork["company_ID"] = number.fit_transform(job_posting_teamwork["Company"].astype('str'))
job_posting_teamwork.loc[job_posting_teamwork['company_ID'] == 0,'company_ID'] = (max(job_posting_teamwork['company_ID'])

job_posting_teamwork['job_ID'] = np.arange(max_job_ID + 1, len(job_posting_teamwork) + max_job_ID+1)
```

```
job_posting_teamwork['posting_source_ID'] = 2
job_posting_teamwork['posting_datetime'] = 'NA'
job_posting_teamwork['application_deadline'] = 'Unknown'
job_posting_teamwork['salary'] = 'Unknown'
job_posting_teamwork['scrape_datetime'] = pd.to_datetime(job_posting_teamwork['scrape_datetime'])

job_posting_teamwork = job_posting_teamwork.rename(columns = {'title': 'job_title', 'url': 'posting_link'})

job_posting_teamwork_df = job_posting_teamwork.reindex(columns = ['job_ID','job_title','Company',"company_ID",'posting_sc

# Creating Company Table
Company_Team = pd.DataFrame(job_posting_teamwork[['company_ID','Company']])
Company_Team_df = Company_Team.drop_duplicates()

# Creating the requirements table
job_requirements_df = pd.DataFrame(job_posting_teamwork_df[['job_ID','details']])
job_requirements_df_final = job_requirements_df.assign(temp = job_requirements_df.details.str.split(",")).explode('detail
job_requirements_df_final['details'] = job_requirements_df_final['details'].str.replace("'","").str.replace('"','')
job_posting_teamwork_df = job_posting_teamwork_df.drop('details',axis = 1)
```

In [ ]:

```
count = 1

for i,j in Company_Team_df.iterrows():
    if((j['Company'] in SQL_df_companies['company_name'].values)):
        Company_Team_df.at[i,'company_ID'] = SQL_df_posting.loc[i,'company_ID']
    else:
        Company_Team_df.at[i,'company_ID'] = max_comp_ID + count
        count = count + 1

job_posting_teamwork_df = pd.merge(job_posting_teamwork_df, Company_Team_df, left_on="Company", right_on="Company", how='
```

In [ ]:

```
job_posting_teamwork_df = job_posting_teamwork_df.rename(columns = {'company_ID_y': 'company_ID'})
job_posting_teamwork_df = job_posting_teamwork_df.drop(['Company','company_ID_x'],axis = 1)
job_posting_teamwork_df
```

In [ ]:

```
for i,j in Company_Team_df.iterrows():
    if((j['company_ID'] in SQL_df_companies['company_ID'].values) and (j['Company'] in SQL_df_companies['company_name'].v
        Company_Team_df = Company_Team_df.drop(index = i,axis = 1)
    else:
        pass
```

In [ ]:
```
Sources = pd.DataFrame({'source_ID': [2], 'source_name': ['Teamwork Online']})
```

In [ ]:
```
## Initialize connection to MYSQL
database = MySQLdb.connect(host="localhost" , user="root" , passwd="Pps11844")
cursor = database.cursor()
```

In [ ]:
```
def execute_query(query_statement):
    try:
        cursor.execute(query_statement);
        database.commit();
        print("Data is Succefully Inserted")

    except Exception as e :
        database.rollback();
        print("The  Exception Occured : ", e)
```

In [ ]:
```
execute_query("USE JobsinSports")
```

In [ ]:
```
for i,j in job_requirements_df_final.iterrows():
    execute_query('INSERT INTO Job_Requirements (job_ID, requirements) VALUES (%d, "%s")' % (j['job_ID'],j['details']))
```

In [ ]:
```
for i,j in Sources.iterrows():
    execute_query('INSERT INTO Sources (source_ID, source_name) VALUES (%d, "%s")' % (j['source_ID'],j['source_name']))
```

In [ ]:
```
for i,j in Company_Team_df.iterrows():
    execute_query('INSERT INTO Company_Team (company_ID, company_name) VALUES (%d, "%s")' % (j['company_ID'], j['Company'
```

In [ ]:
```
for i,j in job_posting_teamwork_df.iterrows():
    execute_query('INSERT INTO Job_Posting (job_ID, job_title, company_ID, scraped_datetime, job_city, job_state, posting
```

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