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 [ ]:

```
## Function to merge two dictionaries
def merge(dict1, dict2):
    return(dict2.update(dict1))

## Function to extract the beautiful soup from link + pagenumber(s)
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)

## Function to extract Beautiful Soup from inner links after scraped from header
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)

## Function to perform html parsing
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 [ ]:

```
############################################## 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"
    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(1, len(job_posting_teamwork)+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_ID",'posting_source_ID','

# 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 [ ]:

```
## Scraping leagues from wikipedia to get big team information
url_page = requests.get('https://en.wikipedia.org/wiki/Major_League_Soccer')
soup = BeautifulSoup(url_page.content,'html.parser')

table_sec = soup.find('table',class_="wikitable sortable")
table_mls = table_sec.find_all('tr')
company_mls_list = []

for team in table_mls:
    team_info = team.find_all('td')
    company_info_mls = []
    for info in team_info:
        company_info_mls.append(info.text.strip())
    company_mls = {
        'total_info': company_info_mls
    }

    company_mls_list.append(company_mls)
```

In [ ]:

```
df1 = pd.DataFrame(company_mls_list)
company_mls_df = pd.DataFrame(df1.total_info.tolist(), columns = ['team_name','Headquarters','Stadium','capacity','founde
company_mls_df['league'] = 'Major League Soccer'
company_mls_df['league_short'] = 'MLS'
company_mls_df = company_mls_df.reindex(columns = ['team_name','Headquarters','league','league_short','Stadium','capacity
company_mls_df.loc[company_mls_df['team_name'] == 'LA Galaxy','team_name'] = 'Los Angeles Galaxy'
```

In [ ]:

```
url_page = requests.get('https://en.wikipedia.org/wiki/Major_League_Baseball')
soup = BeautifulSoup(url_page.content,'html.parser')
table_sec = soup.find('table',class_="wikitable sortable")
table_mlb = table_sec.find_all('tr')
company_mlb_list = []

for team in table_mlb:
    team_info = team.find_all('td')
    company_info_mlb = []
    for info in team_info:
        company_info_mlb.append(info.text.strip())
    company_mlb = {
        'total_info': company_info_mlb
    }

    company_mlb_list.append(company_mlb)
```

In [ ]:

```
df1 = pd.DataFrame(company_mlb_list)
company_mlb_df = pd.DataFrame(df1.total_info.tolist(), columns = ['team_name','Headquarters','Stadium','capacity','coordi
company_mlb_df['league'] = 'Major League Baseball'
company_mlb_df['league_short'] = 'MLB'
company_mlb_df = company_mlb_df.reindex(columns = ['team_name','Headquarters','league','league_short','Stadium','capacity
company_mlb_df['founded'] = company_mlb_df['founded'].str.replace(r"\(..\)",'')
company_mlb_df['founded'] = company_mlb_df['founded'].str.replace("*",'')
```

In [ ]:

```
url_page = requests.get("https://en.wikipedia.org/wiki/National_Football_League")
soup = BeautifulSoup(url_page.content,'html.parser')
table_sec = soup.find('table',class_="wikitable sortable plainrowheaders")
table_nfl = table_sec.find_all('tr')
company_nfl_list = []

for team in table_nfl:
    team_info = team.find_all('td')
    company_info_nfl = []
    for info in team_info:
        company_info_nfl.append(info.text.strip())
    company_nfl = {
        'total_info': company_info_nfl
    }

    company_nfl_list.append(company_nfl)
```

In [ ]:
```
df1 = pd.DataFrame(company_nfl_list)
company_nfl_df = pd.DataFrame(df1.total_info.tolist(), columns = ['team_name','Headquarters','Stadium','capacity','coordi
company_nfl_df['league'] = 'National Football League'
company_nfl_df['league_short'] = 'NFL'
company_nfl_df = company_nfl_df.reindex(columns = ['team_name','Headquarters','league','league_short','Stadium','capacity
company_nfl_df['founded'] = company_nfl_df['founded'].str.partition('(')[0]
company_nfl_df['Stadium'] = company_nfl_df['Stadium'].str.replace(r"\[.\]",'')
company_nfl_df['founded'] = company_nfl_df['founded'].str.replace(r"\[.\]",'')
company_nfl_df['team_name'] = company_nfl_df['team_name'].str.replace("*",'')
```

In [ ]:
```
url_page = requests.get("https://en.wikipedia.org/wiki/National_Hockey_League")
soup = BeautifulSoup(url_page.content,'html.parser')
table_sec = soup.find('table',class_="wikitable")
table_nhl = table_sec.find_all('tr')
company_nhl_list = []

for team in table_nhl:
    team_info = team.find_all('td')
    company_info_nhl = []
    for info in team_info:
        company_info_nhl.append(info.text.strip())
    company_nhl = {
        'total_info': company_info_nhl
    }

    company_nhl_list.append(company_nhl)
```

In [ ]:
```
df1 = pd.DataFrame(company_nhl_list)
company_nhl_df = pd.DataFrame(df1.total_info.tolist(), columns = ['team_name','Headquarters','Stadium','capacity','founde
company_nhl_df['league'] = 'National Hockey League'
company_nhl_df['league_short'] = 'NHL'
company_nhl_df = company_nhl_df.reindex(columns = ['team_name','Headquarters','league','league_short','Stadium','capacity
company_nhl_df['founded'] = company_nhl_df['founded'].str.replace("*",'')
```

In [ ]:
```
url_page = requests.get("https://en.wikipedia.org/wiki/National_Basketball_Association")
soup = BeautifulSoup(url_page.content,'html.parser')
table_sec = soup.find('table',class_="wikitable")
table_nba = table_sec.find_all('tr')
company_nba_list = []
```

```
for team in table_nba:
    team_info = team.find_all('td')
    company_info_nba = []
    for info in team_info:
        company_info_nba.append(info.text.strip())
    company_nba = {
        'total_info': company_info_nba
    }

    company_nba_list.append(company_nba)
```

In [ ]:

```
df1 = pd.DataFrame(company_nba_list)
company_nba_df = pd.DataFrame(df1.total_info.tolist(), columns = ['team_name','Headquarters','Stadium','capacity','coordi
company_nba_df['league'] = 'National Baseball Association'
company_nba_df['league_short'] = 'NBA'
company_nba_df = company_nba_df.reindex(columns = ['team_name','Headquarters','league','league_short','Stadium','capacity

company_nba_df.loc[4,'capacity'] = '19,812'
company_nba_df.loc[4,'Stadium'] = 'Madison Square Garden'
company_nba_df.loc[4,'Headquarters'] = 'New York City, New York'
company_nba_df.loc[4,'founded'] = '1946'

company_nba_df.loc[25,'capacity'] = '19,079'
company_nba_df.loc[25,'Stadium'] = 'Crypto.com Arena'
company_nba_df.loc[25,'Headquarters'] = 'Los Angeles, California'
company_nba_df.loc[25,'founded'] = '1947'
company_nba_df['founded'] = company_nba_df['founded'].str.replace("*",'')
```

In [ ]:

```
### ONLY RUN ONCE!!!!! ###
company_mlb_df.drop(index=company_mlb_df.index[[0,1,17]], axis=0, inplace=True)
company_nfl_df.drop(index=company_nfl_df.index[[0,1,18,35]], axis=0, inplace=True)
company_nba_df.drop(index=company_nba_df.index[[0,1,17]], axis=0, inplace=True)
company_nhl_df.drop(index=company_nhl_df.index[[0,1,18]], axis=0, inplace=True)
company_mls_df.drop(index=company_mls_df.index[[0,1,17]], axis=0, inplace=True)
```

In [ ]:

```
company_teams_df_temp = pd.concat([company_mlb_df,
company_nfl_df,
company_nba_df,
```

In [ ]:
```
Company_Team_df_temp2 = pd.merge(Company_Team_df, company_teams_df_temp, left_on="Company", right_on="team_name", how='ou
```

In [ ]:
```
count = max(Company_Team_df['company_ID'])
new_ID = count + 1

Company_Team_df_temp2['company_ID'] = np.where(Company_Team_df_temp2['company_ID']>0,Company_Team_df_temp2['company_ID'],
Company_Team_df_temp2['Company'] = np.where(Company_Team_df_temp2['Company'].isnull(),'None',Company_Team_df_temp2['Compa
Company_Team_df_temp2['capacity'] = Company_Team_df_temp2['capacity'].str.replace(",",'')
Company_Team_df_temp2['capacity'] = np.where(Company_Team_df_temp2['capacity'].isnull(),0,Company_Team_df_temp2['capacity
Company_Team_df_temp2['founded'] = np.where(Company_Team_df_temp2['founded'].isnull(),0,Company_Team_df_temp2['founded'])

for i,j in Company_Team_df_temp2.iterrows():
    if (j['company_ID'] == 0.0):
        Company_Team_df_temp2.at[i,'company_ID'] = new_ID
        new_ID = new_ID + 1
    else:
        pass

for i,j in Company_Team_df_temp2.iterrows():
    if (j['Company']=='None'):
        Company_Team_df_temp2.at[i,'Company'] = j['team_name']
    else:
        pass

Company_Team_df_temp2['company_ID'] = Company_Team_df_temp2['company_ID'].astype(int)
Company_Team_df_temp2['founded'] = Company_Team_df_temp2['founded'].astype(int)
Company_Team_df_temp2['capacity'] = Company_Team_df_temp2['capacity'].astype(int)

Company_Team_df_final = Company_Team_df_temp2.drop('team_name',axis = 1)
Company_Team_df_final = Company_Team_df_final.fillna('NA')
Company_Team_df_final['Headquarters_city'] = Company_Team_df_final['Headquarters'].str.partition(', ')[0]
Company_Team_df_final['Headquarters_state'] = Company_Team_df_final['Headquarters'].str.partition(', ')[2]
Company_Team_df_final.drop('Headquarters',axis = 1,inplace = True)
Company_Team_df_final = Company_Team_df_final.reindex(columns = ['company_ID', 'Company', 'league', 'league_short', 'Head
```

In [ ]:
```
## SQL Command Execution Begins Here
```

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 [ ]: execute_query("CREATE TABLE IF NOT EXISTS Job_Posting(job_ID BIGINT PRIMARY KEY NOT NULL UNIQUE, job_title VARCHAR(255),
```

```
In [ ]: execute_query("CREATE TABLE IF NOT EXISTS Company_Team(company_ID INT PRIMARY KEY NOT NULL UNIQUE, company_name VARCHAR(2
```

```
In [ ]: execute_query("CREATE TABLE IF NOT EXISTS Job_Requirements(job_ID BIGINT, requirements TEXT,PRIMARY KEY(job_ID,requiremen
```

```
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_final.iterrows():
            execute_query('INSERT INTO Company_Team (company_ID, company_name, company_headquarters_city, company_headquarters_st
```

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 [ ]:
```
## Only used to delete and edit schema during Debugging phase
#execute_query("DROP TABLE Job_Requirements")
```

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