In [ ]:

```
import pandas as pd
import numpy as np
from datetime import datetime,timedelta
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(league):
    url = f'https://www.linkedin.com/jobs/search?keywords={league}&location=United%20States&geoId=103644278&trk=public_jc
    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 = 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_ = 'base-card relative w-full hover:no-underline focus:no-underline base-card--link

    for job in divs:
        title = job.find('h3',class_='base-search-card__title').text.strip()
        company = job.find('h4',class_ = 'base-search-card__subtitle').text.strip()
        location = job.find('span',class_ = 'job-search-card__location').text.strip()

        scrape_date = datetime.now()
        try:
            posting_delta = job.find('time',class_ = 'job-search-card__listdate').text.strip().partition(' ')
            delta = posting_delta[0]

            if(posting_delta[2].partition(' ')[0] == 'hours'):
                post_date = scrape_date - timedelta(hours = int(delta))

            elif(posting_delta[2].partition(' ')[0] == 'days'):
                post_date = scrape_date - timedelta(days = int(delta))

            elif(posting_delta[2].partition(' ')[0] == 'weeks'):
                post_date = scrape_date - timedelta(weeks = int(delta))

            else:
                delta = delta * 4
                post_date = scrape_date - timedelta(weeks = int(delta))
```

```
        except:
            post_date = datetime.now()

        link_ext = job.a['href']
        job_ID = link_ext.partition('?refId')[0].split('-')[-1]

        #details = []
        more_info = extract_inner(link_ext)
        try:
            divs_inner = more_info.find('div',class_ = 'show-more-less-html__markup')
            divs_inner_1 = divs_inner.find_all('ul')
            base_text = divs_inner.prettify()
            details = []

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

        except:
            details= []

        job = {
            'job_ID': job_ID,
            'title': title,
            'Location': location,
            'Company': company,
            'details': details,
            'url': link_ext,
            'posting_datetime': post_date.strftime("%m/%d/%Y %H:%M:%S"),
            'scrape_datetime': scrape_date.strftime("%m/%d/%Y %H:%M:%S"),
            'additionals': base_text
        }

        joblist.append(job)

    return
```

In [ ]:
```
joblist = []
leagues = ['Major%20League%20Soccer', 'Major%20League%20Baseball','National%20Football%20League']
for league in leagues:
    c=extract(league)
    transform(c)

joblist1 = joblist
```

In [ ]:
```
joblist = []
leagues_again = ['National%20Hockey%20League', 'National%20Basketball%20Association']
for league in leagues_again:
    c=extract(league)
    transform(c)

joblist2 = joblist
```

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")

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

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

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

database.close()
```

In [ ]:

```
job_posting_linkedin_1 = pd.DataFrame(joblist1)
job_posting_linkedin_2 = pd.DataFrame(joblist2)
job_posting_linkedin = pd.concat([job_posting_linkedin_1, job_posting_linkedin_2])
job_posting_linkedin.reset_index(drop=True, inplace=True)

job_posting_linkedin['job_ID'] = job_posting_linkedin['job_ID'].astype(float)

for i,j in job_posting_linkedin.iterrows():
    if(re.findall(r'\$',j['additionals'])):
        job_posting_linkedin.at[i,'salary'] = '$'+(j['additionals'].partition('$')[2].partition('.')[0].partition('<')[0]
    else:
        job_posting_linkedin.at[i,'salary'] = 'NA'

for i,j in job_posting_linkedin.iterrows():
    if(len(j['salary'])==3):
        job_posting_linkedin.at[i,'salary'] = (j['salary'] + '/hr')
    else:
        pass

for i,j in job_posting_linkedin.iterrows():
    if(re.findall(r'New York City',j['Location'])):
        job_posting_linkedin.at[i,'Location'] = 'New York, NY'
    else:
        pass

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

job_posting_linkedin['Company'] = job_posting_linkedin['Company'].str.partition('(')[0].str.replace('Football Club ','FC'
job_posting_linkedin['Company'] = job_posting_linkedin['Company'].str.strip()
job_posting_linkedin['posting_source_ID'] = 3
job_posting_linkedin['application_deadline'] = 'Unknown'
job_posting_linkedin['scrape_datetime'] = pd.to_datetime(job_posting_linkedin['scrape_datetime'])
job_posting_linkedin['posting_datetime'] = pd.to_datetime(job_posting_linkedin['posting_datetime'])


job_requirements_df = pd.DataFrame(job_posting_linkedin[['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('"','')
```

In [ ]:

```
job_posting_linkedin.drop(['Location','details','additionals'],axis = 1,inplace = True)
```

In [ ]:
```
```

In [ ]:
```
Company_Team = pd.DataFrame(job_posting_linkedin['Company'])
Company_Team_df = Company_Team.drop_duplicates()
```

In [ ]:
```
Company_Team_df['Company_temp'] = [1,2,3,4,5,6,7,8]
Company_Team_df.loc[Company_Team_df['Company_temp'] == 1,'company_ID'] = int(max_comp_ID + 1)
Company_Team_df.loc[Company_Team_df['Company_temp'] == 5,'company_ID'] = int(max_comp_ID + 2)
Company_Team_df.loc[Company_Team_df['Company_temp'] == 6,'company_ID'] = int(max_comp_ID + 3)
Company_Team_df.loc[Company_Team_df['Company_temp'] == 7,'company_ID'] = int(max_comp_ID + 4)
Company_Team_df.loc[Company_Team_df['Company_temp'] == 8,'company_ID'] = int(max_comp_ID + 5)
Company_Team_df.loc[Company_Team_df['Company_temp'] == 2,'company_ID'] = 258
Company_Team_df.loc[Company_Team_df['Company_temp'] == 3,'company_ID'] = 257
Company_Team_df.loc[Company_Team_df['Company_temp'] == 4,'company_ID'] = 255
Company_Team_df.drop('Company_temp',inplace=True,axis=1)
```

In [ ]:
```
Company_Team_df
```

In [ ]:
```
job_posting_linkedin_df = pd.merge(job_posting_linkedin, Company_Team_df, left_on="Company", right_on="Company", how='lef
```

In [ ]:
```
job_posting_linkedin_df = job_posting_linkedin_df.reindex(columns = ['job_ID','title',"company_ID",'posting_source_ID','p
```

In [ ]:
```
Sources = pd.DataFrame({'source_ID': [3], 'source_name': ['Linkedin']})
```

In [ ]:
```
# Tested but not perfected yet -- IGNORE
#job_posting_linkedin_df
#count = 1

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

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_linkedin_df.iterrows():
    execute_query('INSERT INTO Job_Posting (job_ID, job_title, company_ID, posting_datetime, scraped_datetime, salary, jo
```

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