READ ME Vincenzo Coppola - 001823744

This document contains the information pertaining to the addition of a "Jobs in Sports" repository as part of the AI Skunks database. The questions that I was hoping to be answered initially are:

What kind of jobs are most available in the sports world?

Can I find data related jobs in sports easily? (Personal question but can apply to any discipline)

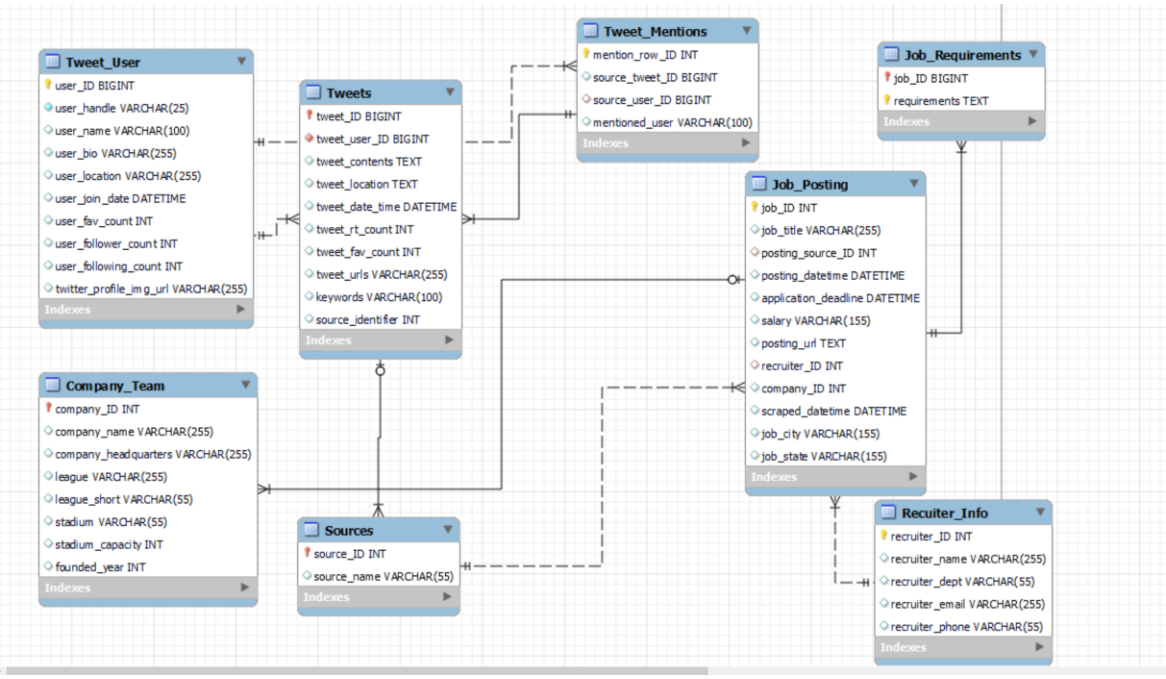
Who can I contact to network and inquire about the job?

In order to do this, the database was planned to consist of at least, Job Postings, Required Skills, Company Information, and ideally Recruiter Information. These can be better described below:

> Job Postings from sites such as Teamwork Online/LinkedIn/indeed etc. The info here will likely be Job ID's, Team/Company Name, Team/Company Designation (Sports teams will be kept differently than large scale sports firms), Salary Info, Date Posted, Deadline Date, Location, Remote option, etc.
>
> Company Information from the company site (Likely teams). This data will include at a minimum, Team/Company Name/ID, League, Size, Market Size, etc.
>
> Required Skills parsed from the job postings to aid with matching job seekers to the jobs and to assist them with learning necessary skills to enter the sports world.
>
> Recruiter Information in hopes that job seekers can have a foot forward by having a contact to reach out to and make a connection. This will likely include Name (First and Last), Team Email, Phone (if Possible), Department, etc.

The project was made with the aim to attempt and mimic other repositories in the database in hopes to match job seekers with the right teams for them and provide guidance on the skills that are in demand as well as a contact so that they can better enter one of the harder markets out there. First, as previously mentioned existing databases like LinkedIn and Teamwork online needed to be better researched to know what data is needed. They then needed to be scraped and data collected prior to cleaning and wrangling. Once it is at the point for database insertion, MYSQL was preferred and used via python to enter data and be referenced after initial scraping.

While the aim of the project was trivial this was an individual effort and far from simple. The true process consisted of researching the familiar databases around and understanding the data available. Then an ER Diagram was designed in order to provide a base line. This was done at the point of the twitter API checkpoint as seen below.



At this point countless hours were allocated to learning the Twitter API which was an extremely useful skill and so the twitter data was kept in cleaned form under a series of twitter tables. Twitter however was not the best site to use as it is one of if not the least formal sites around and does not have the ideal platform for job seekers.

Moving on from that point, the actual web scraping was to be done. Initially, the ideal platform to scrape was Indeed however after days and weeks of attempting to learn how to web scrape and apply it on Indeed.Com it was determined improbable if not impossible due to increased security measures in place. Once Indeed was abandoned, Teamwork Online, a well-known job database was successfully scraped using BeautifulSoup in Python. This data was used as the baseline for the SQL ER diagram and was the bulk of the project.

Following suit, Linkedin was scraped and inserted into the database in a single day illustrating exponential growth in data collection and storage knowledge. With the data collected, several steps were taken to ensure the database was as normalized as possible. Some issues in the

table creations were found in these steps and addressed using altering comments in parallel with adjusting the initial create table statements [Therefore Alter Statements should not be needed].

While the initial goals of this project were big, they turned out not to be feasible for a group of 1. Indeed's Security, Linkedin's legal standing on web scraping, and locked recruiter information internet-wide proved to be some of many roadblocks. That being said, the database as completed is an illustration of what was able to be accomplished in a short amount of time by one individual. Moving Forward there are things that will be ideally fixed. More time will go into Indeed scraping / potential API usage as the information on that site is the best seen to this point. The data there will ideally be cleaned up and more scripts will be developed for future scraping. Additionally, the twitter data will be extracted further to narrow down real job opportunities only and not just tweets that could be jobs. The concept of a job database requires continuous refreshes and will be implemented in the best-case scenario. Below is a description of all files included for ease of reader.

Files Included:

Twitter Scrape.IPYNB – Python Twitter API (TWEEPY) code to scrape twitter and insert into Tweets, Tweet_Mentions and Tweet_user tables

Teamwork Online Scrape.IPYNB – Python script to make the original scrape of Teamwork Online and create all tables for rest of database. Data inserted into job_postings, Company_Team, sources, job_requirements, etc

Teamwork Online Scrape Refreshes.IPYNB – Python Script for all additional scrapes following the initial teamwork scrape. Should avoid duplicating current database before inserts

Linkedin Scrape.IPYNB – Python Script to scrape Linkedin with some keywords (league names) specified. Should also be aware of current database

JobsInSports_Schema - ER Diagram for database including future tables such as recruiter info that weren't able to be made as data was not able to be found yet

Table Alter Statements / Table Creation Statements / jobsinsports_views - Other files including SQL Create Statements, use cases, Views & Alters (ignore Alters assuming no errors)

References – Referenced sites and links for data and information during the project

Other Images:

USE CASES:

1.

Use Case: Search for Analyst Jobs in the NHL

Description: User wants to find job posts for the National Hockey League

Actors: User

Precondition: User must have access to the database

Actor action – User queries for job postings joined with companies based in the NHL

System Responses – Database results with company league_short as NHL are displayed

2.

Use Case: Search for job requirements to job postings in Massachusetts

Description: User wants to find requirements of jobs in Massachusetts that are in the database

Actors: User

Precondition: User must have access to the database

Actor action: User queries for job postings with locations in MA

System Responses: A list of all job postings and their state are displayed with requirements for them

3.

Use Case: Find the league with the most postings available

Description: User wants to find out which league is looking for the most help

Actors: User

Precondition: User must have access to the database

Actor action: User queries for a count of all job postings by league

System Responses: Each league and a count of postings is displayed

4.

Use Case: Find remote jobs not on a sport team but in the sports domain

Description: User wants to find jobs that are not on sports teams but still sports related and offered remotely

Actors: User

Precondition: User must have access to the database

Actor action: User queries for jobs with locations that mention remote and have NULL or NA leagues

System Responses: job postings are displayed for the met criteria

5.

Use Case: Find links to job postings for the Spring specifically in the MLB

Description: User wants to find Spring jobs for professional baseball teams

Actors: User

Precondition: User must have access to the database

Actor action: User Queries for job postings with company league of Major League Baseball and a job title like spring

System Responses: Job titles containing spring and their respective location is displayed

SQL FOR USE CASES:

1.

```
CREATE VIEW NHL_Analysts AS

SELECT c.company_name, p.job_title, c.league_short, p.job_city, p.job_state

FROM job_posting p

JOIN company_team c

ON p.company_ID = c.company_ID

WHERE p.job_title LIKE "%analyst%" and c.league_short = "NHL" OR c.company_name =
'National Hockey League';
```

2.

```
CREATE VIEW MA_job_postings AS

SELECT p.job_ID, p.job_title, p.job_city, p.job_state, r.requirements

FROM job_posting p

JOIN job_requirements r

ON p.job_ID = r.job_ID

WHERE p.job_state LIKE "%MA";
```

3.

```
CREATE VIEW league_listings AS

SELECT  c.league_short, COUNT(p.job_ID) as league_postings

FROM job_posting p

JOIN company_team c

ON p.company_ID = c.company_ID

GROUP BY c.league_short

ORDER BY league_postings DESC;
```

4.

CREATE VIEW remote_non_league AS

SELECT p.job_ID, p.job_title, p.scraped_datetime, p.job_city, p.posting_url, c.company_ID, c.company_name

FROM job_posting p

JOIN company_team c

ON p.company_ID = c.company_ID

WHERE c.league IS NULL AND (p.job_city LIKE "%remote%" OR p.job_state LIKE "%remote%");

5.

CREATE VIEW spring_jobs AS

SELECT c.company_name, p.job_title, c.league_short, p.posting_url

FROM job_posting p

JOIN company_team c

ON p.company_ID = c.company_ID

WHERE p.job_title LIKE "%Spring%";

1.

```
37 •  SELECT * FROM league_listings;
```

Result Grid | Filter Rows: | Export

league_short	league_postings
NULL	335
MLB	40
NFL	15
NBA	14
NHL	11
MLS	10

2.

```
37 •  SELECT * FROM ma_job_postings;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_ID	job_title	job_city	job_state	requirements
266	Red Sox Foundation 50/50 Raffle Sales Team M...	Boston	MA	Ability to work nights, weekends and holidays, ...
266	Red Sox Foundation 50/50 Raffle Sales Team M...	Boston	MA	Highly motivated, compassionate, respectful an...
266	Red Sox Foundation 50/50 Raffle Sales Team M...	Boston	MA	Must be detail-oriented and demonstrate strong...
266	Red Sox Foundation 50/50 Raffle Sales Team M...	Boston	MA	Interact positively with fans and deliver the bes...
266	Red Sox Foundation 50/50 Raffle Sales Team M...	Boston	MA	Interest in sports philanthropy or non-profit wo...
266	Red Sox Foundation 50/50 Raffle Sales Team M...	Boston	MA	Prior sales and customer service experience, pr...
269	Kids Crew 2023	Boston	MA	Greeting and engaging with our young fans ent...
269	Kids Crew 2023	Boston	MA	Ability to work special events outside of the ball...
269	Kids Crew 2023	Boston	MA	Maintaining a professional and friendly demean...

ma_job_postings27

3.

```
SELECT * FROM nhl_analysts;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name	job_title	league_short	job_city	job_state
National Hockey League	Business Analyst	NULL	New York	NY
Vegas Golden Knights	CRM & Database Analyst	NHL	Las Vegas	NV
National Hockey League	Director, Finance - Sponsorship & Partnership M...	NULL	New York	NY
National Hockey League	Off Ice Offical (Part Time /Seasonal)	NULL	Dallas	TX
National Hockey League	Sr. Manager, Innovation	NULL	New York	NY
National Hockey League	Director, Club Business Affairs	NULL	New York	NY
National Hockey League	Senior Manager, Club Business Affairs	NULL	New York	NY
National Hockey League	Coordinator, Business Development - Internatio...	NULL	New York	NY
National Hockey League	Manager, Event Communications and Player De...	NULL	New York	NY
National Hockey League	Assistant, Social Impact, Growth Initiatives and	NULL	New York	NY

nhl_analysts 28

4.

```
SELECT * FROM remote_non_league;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_ID	job_title	scraped_datetime	job_city	posting_url	company_ID	company_name
3	Senior Director, Strategic Partnerships	2022-12-11 22:05:38	Open to Remote	https://www.teamworkonline.com/multiple-prop...	105	OneTeam Partners
6	Sponsorship & Athlete Marketing Coordinator	2022-12-11 22:05:39	Open to Remote	https://www.teamworkonline.com/multiple-prop...	105	OneTeam Partners
21	Coordinator, Social Media	2022-12-11 22:05:43	Remote	https://www.teamworkonline.com/other-sports...	116	Professional Bull Riders
132	Bilingual (Spanish) Web Coordinator	2022-12-11 22:06:08	Kansas City	https://www.teamworkonline.com/soccer-jobs/...	130	Sporting KC
226	Game Day Social Media & Highlights Coordinator	2022-12-11 22:06:29	Carlsbad	https://www.teamworkonline.com/soccer-jobs/...	82	Major Arena Soccer League
255	Regional Development Officer—Texas, West Co...	2022-12-11 22:06:37	Remote: Texas	https://www.teamworkonline.com/olympic-jobs/...	143	USA Triathlon

5.

```
SELECT * FROM spring_jobs;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name	job_title	league_short	posting_url
500 Festival	500 Festival Post-Race Party, Course Entertain...	NULL	https://www.teamworkonline.com/live-event-jo...
United Soccer League	USL League Two Operations Internship - Spring ...	NULL	https://www.teamworkonline.com/soccer-jobs/...
FC Tulsa	Merchandise Internship, Spring 2023	NULL	https://www.teamworkonline.com/soccer-jobs/...
Pittsburgh Penguins	Corporate Social Responsibility and Diversity, E...	NHL	https://www.teamworkonline.com/hockey-jobs/...
National Football League	2023 Spring Law Clerk	NULL	https://www.linkedin.com/jobs/view/2023-sprin...

4 Main Tables

1. Job_posting – contains all job titles, posting dates, scraped dates, application deadlines, salary estimates, location (city and state) source_ID, company_ID, and posting website.

```
SELECT *
FROM job_posting;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

job_ID	job_title	company_ID	posting_datetime	scraped_datetime	applic	salary	job_city	job_state	posting_url
3313...	Senior Manager, Club Business Affairs	275	2022-11-30 21:36:31	2022-12-14 21:36:31	NULL	$80,000-...	New York	NY	https://www.linkedin.com/jobs/view/se
3314...	Coordinator, Business Development - Internatio...	275	2022-11-30 21:36:20	2022-12-14 21:36:20	NULL	$50,000-...	New York	NY	https://www.linkedin.com/jobs/view/co
3321...	Manager, Event Communications and Player De...	275	2022-11-30 21:36:23	2022-12-14 21:36:23	NULL	$70,000-...	New York	NY	https://www.linkedin.com/jobs/view/ma
3322...	Assistant - Social Impact, Growth Initiatives and...	275	2022-11-30 21:36:23	2022-12-14 21:36:23	NULL	$50,000-...	New York	NY	https://www.linkedin.com/jobs/view/as
3323...	Royalty Analyst	275	2022-11-30 21:36:30	2022-12-14 21:36:30	NULL	$50,000-...	New York	NY	https://www.linkedin.com/jobs/view/ro
3323...	Product Designer	275	2022-12-09 21:36:31	2022-12-14 21:36:31	NULL	$120,000...	New York	NY	https://www.linkedin.com/jobs/view/pr
3327...	Director, Club Innovation & Business Development	275	2022-11-30 21:36:21	2022-12-14 21:36:21	NULL	$140,000...	New York	NY	https://www.linkedin.com/jobs/view/di
3333...	Project Employee, Social Content - Graphic Design	276	2022-12-10 21:36:43	2022-12-14 21:36:43	NULL	$1,153pe...	Secaucus	NJ	https://www.linkedin.com/jobs/view/pr
3334...	Senior Coordinator, Club Marketing	273	2001-08-29 21:24:51	2022-12-14 21:24:51	NULL	$60,000-...	New York	NY	https://www.linkedin.com/jobs/view/se
3334...	Manager, Media Strategy & Business Development	274	2022-12-11 21:25:05	2022-12-14 21:25:05	NULL	NA	New York	NY	https://www.linkedin.com/jobs/view/ma
3338...	Senior Director, E-Commerce	275	2022-11-30 21:36:22	2022-12-14 21:36:22	NULL	$140,000...	New York	NY	https://www.linkedin.com/jobs/view/se
3339...	Part-Time Designer/Animator	274	2022-11-23 21:25:09	2022-12-14 21:25:09	NULL	NA	Inglewood	CA	https://www.linkedin.com/jobs/view/pa
3340...	Analyst - Travel & Procurement	275	2022-11-30 21:36:25	2022-12-14 21:36:25	NULL	$55,000-...	New York	NY	https://www.linkedin.com/jobs/view/ar
3341...	Director, Employee Relations	274	2022-11-30 21:25:02	2022-12-14 21:25:02	NULL	NA	New York	NY	https://www.linkedin.com/jobs/view/di
3341...	Product Designer	276	2022-11-30 21:36:40	2022-12-14 21:36:40	NULL	$110,000...	New York	NY	https://www.linkedin.com/jobs/view/pr
3341...	Project Manager, Multicultural Marketing	275	2022-12-08 21:36:29	2022-12-14 21:36:29	NULL	$75,000-...	New York	NY	https://www.linkedin.com/jobs/view/pr
3341...	Director, Multicultural Engagement & Integration	275	2022-12-08 21:36:27	2022-12-14 21:36:27	NULL	$115,000...	New York	NY	https://www.linkedin.com/jobs/view/di
3344...	Coordinator, Ticket Operations	275	2022-12-08 21:36:24	2022-12-14 21:36:24	NULL	$52,000-...	New York	NY	https://www.linkedin.com/jobs/view/co
3350...	Director, NFL+ Content	274	2022-11-30 21:25:07	2022-12-14 21:25:07	NULL	NA	Inglewood	CA	https://www.linkedin.com/jobs/view/di
3350...	Illustrator, Social Media (External Agency Emplo...	274	2022-11-30 21:25:12	2022-12-14 21:25:12	NULL	NA	Inglewood	CA	https://www.linkedin.com/jobs/view/ill

2. Company_Team – Company_ID, company_name, headquarters (~location city and state), league,league_short (abbreviation), stadium (name), capacity, founded_year.

```
SELECT * FROM Company_Team;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

company_ID	company_name	company_headquarters_city	company_headquarters_state	league	league_short	stadium	stadium_capacity	founded_year
157	Washington Wizards	Washington	D.C.	National Baseball Assoc...	NBA	Capital One ...	20356	1961
158	Wilmington Blue Rocks	NULL	NULL	NULL	NULL	NULL	NULL	NULL
159	Worcester Red Sox	NULL	NULL	NULL	NULL	NULL	NULL	NULL
160	3Step Soccer	NULL	NULL	NULL	NULL	NULL	NULL	NULL
161	Baltimore Orioles	Baltimore	Maryland	Major League Baseball	MLB	Oriole Park a...	45971	1901
162	New York Yankees	New York City	New York	Major League Baseball	MLB	Yankee [illegible]	[illegible]	1901
163	Toronto Blue Jays	Toronto	Ontario	Major League Baseball	MLB	Rogers [illegible]	[illegible]	1977
164	Cleveland Guardians	Cleveland	Ohio	Major League Baseball	MLB	Progressive ...	34830	1901
165	Minnesota Twins	Minneapolis	Minnesota	Major League Baseball	MLB	Target Field	38871	1901
166	Los Angeles Angels	Anaheim	California	Major League Baseball	MLB	Angel Stadium	45957	1961
167	Oakland Athletics	Oakland	California	Major League Baseball	MLB	RingCentral ...	35067	1901
168	Seattle Mariners	Seattle	Washington	Major League Baseball	MLB	T-Mobile Park	47943	1977
169	Texas Rangers	Arlington	Texas	Major League Baseball	MLB	Globe Life Field	40300	1961
170	Atlanta Braves	Cumberland	Georgia	Major League Baseball	MLB	Truist Park	41500	1871
171	Miami Marlins	Miami	Florida	Major League Baseball	MLB	LoanDepot P...	36742	1993
172	New York Mets	New York City	New York	Major League Baseball	MLB	Citi Field	41922	1962

Yankee Stadium

Company_Team 32 ×

3. Job_requirements

```
SELECT *
FROM job_requirements;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_ID	requirements
3384...	Hires, develops, and retains diverse talent that makes a strong, positive impact on the organization
3384...	Sets goals and manages the performance of direct report(s) using the established performance review process
3384...	Supports learning and development of direct report(s)
3384...	Facilitates regular, ongoing communication with direct report(s)
3384...	Promotes adherence to internal policies, holding self and direct report(s) accountable to those policies
3302...	Prepare journal entry support
3302...	Assist various MLB entities with accounts payable and prepare related bank reconciliations
3302...	Assist with day to day accounts receivable activities including drafting bills for international broadcast, publishing and productions
3302...	Assist with reconciliations of Balance Sheet and Profit & Loss accounts
3302...	Assist with the financial close within the accounting system
3302...	Collect data from various foreign offices
3302...	Prepare analyses and reporting of financial data at the end of each month
3302...	Assist with the dissemination of information to the auditors

4. Sources – simple source reference table to identify source name where each job posting was found and includes source_ID, source_name

```
37 •  SELECT * FROM sources;
```

Result Grid | Filter Rows:

source_ID	source_name
1	Twitter
2	Teamwork Online
3	Linkedin
NULL	NULL

Finally, below are some of the Less used tables in the database that are included due to the twitter API assignment being a prominent piece of the semester. These tables were not the focus of this database but will be improved moving forward.

5. Tweet_user – user profile information for all users of scraped tweets.

```
49 •  SELECT *
50    FROM tweet_user;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

user_ID	user_handle	user_name	user_bio	user_location	user_join_date	user_fav_count	user_follower_count	user_following_co
22837487	NYSBA	New York State Bar Association	Serving attorneys and the community since 187...	1 Elk Street, Albany, NY	2009-03-04 16:28:45	5901	19401	1632
22875075	CA_ATL_HRTA2	Hospitality/Restaurant Jobs in Atlan...	Follow this account for geo-targeted Hospitality...	Atlanta, GA	2009-03-04 22:44:08	0	355	191
24065241	onetwocross	Shingo Kohara	Go Warriors and Golden State TKD!	iPhone: 37.570454,-1...	2009-03-12 19:16:38	699	541	1986
24854155	vmdcruz	Maryann Virginia Dcruz	finding the horizon	India	2009-03-17 03:41:19	42887	224	867
25312400	SyracuseMets	Syracuse Mets	The Official Twitter of the Syracuse Mets | Tripl...	Syracuse, NY	2009-03-19 11:05:49	12734	30590	798
25396117	themediamentor	Jobs by Media Mentor	Jobs & advice for media professionals & student...	51.422919,-0.685714	2009-03-19 18:16:30	14067	18530	1752
25482527	cirencoll	Cirencester College	Your specialist Sixth Form provider. Follow us o...	Cirencester	2009-03-20 05:31:53	853	4764	335
25517844	Dbetras	David Betras	I'm a father, Husband, attorney I love boating. ...	Mahoning Valley	2009-03-20 11:10:02	1981	2257	1377
26005689	annarbornews	The Ann Arbor News	Local news covering Ann Arbor and the Washte...	Ann Arbor, Michigan	2009-03-23 10:04:24	178	60951	1109
27090815	darthvinson	tarkheena	Unimportant Indo-Brit abroad. Musician, if anyo...		2009-03-27 16:35:25	7030	135	1374
27374841	dark__memer	dark_memer			2009-03-28 23:51:10	14149	109	415
28345003	SBCubs	South Bend Cubs	Chicago Cubs High-A affiliate 🦁 x2 Voted Best ...	Four Winds Field - Sou...	2009-04-02 10:16:20	9687	53493	412
28564292	tmj_ukc_retail	TMJ-UKC Retail Jobs	Follow this account for geo-targeted Retail job t...	Cardiff	2009-04-03 09:42:13	0	173	137
28652066	GVIrish	Gerald Irish	Senior Software Engineer @ Microsoft Opinions ...	Arlington, VA	2009-04-03 17:02:07	30957	245	560
28815243	epouvantail	John	Baseball, music, films. #cufc	Northern England	2009-04-04 11:49:28	8591	1510	1372
29108008	300ZXNA	Chad Schobert, O.D.	Former fundamentalist trying to stop the GOP's ...	Idaho	2009-04-05 20:52:26	175702	684	945
29554784	bmarcello	Brandon Marcello	College Football for @247Sports | Previously: Bi...	United States	2009-04-07 17:47:17	9836	38852	2110
30022599	apache14	Bball Pundit¤	Founder Basketball254 • African Basketball Is T...	• Khwiroro • Kimilili City	2009-04-09 13:10:35	1555	2856	2069
30090883	Justinsua	Justin Su'a	Wherever you see consistency, a system is in pl...		2009-04-09 19:01:27	22509	26452	937
30803366	KevinMcCaff	Kevin McCaffrey	Comedian (Letterman/truTV). My album is titled ...	New York City	2009-04-13 00:04:35	72860	9900	978

6. Tweets – data for each tweet scraped based off of specific keywords included in the database.

```
SELECT *
FROM tweets;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

tweet_ID	tweet_user_ID	tweet_contents	tweet_date_time	tweet_location	tweet_rt_count	tweet_fav_count	tweet_urls	keywords	sou
172727374	5167462	The Boston Celtics are searching for a new sale...	NULL	Boston, MA	0	0	THISisAfakeLINKBC.org	NULL	1
1592081187224403968	1168169980921339905	NEW JOB ALERT! FC Eindhoven NL C...	2022-11-14 04:05:09	None	0	0	https://t.co/O3hP4igrYN	SoccerJobs	1
1592096281324838913	1168169980921339905	NEW JOB ALERT! Brisbane Roar FC ...	2022-11-14 05:05:07	None	1	1	https://t.co/O3hP4igrYN	SoccerJobs	1
1592111378210455555	1168169980921339905	NEW JOB ALERT! Brisbane City FC A...	2022-11-14 06:05:07	None	0	0	https://t.co/O3hP4igrYN	SoccerJobs	1
1592126483518169088	1168169980921339905	NEW JOB ALERT! Paris Saint-Germai...	2022-11-14 07:05:08	None	0	0	https://t.co/O3hP4igrYN	SoccerJobs	1
1592141686729236480	1168169980921339905	NEW JOB ALERT! Nashville Soccer Cl...	2022-11-14 08:05:33	None	0	0	https://t.co/O3hP4igrYN	SoccerJobs	1
1592159151152857089	1045280698238676993	Job Opportunity Association: Leicester City P...	2022-11-14 09:14:57	None	1	2	https://t.co/QuLHyref5y	FootballJobs	1
1592159650300190721	1045280698238676993	Job Opportunity Association: Exeter City Pos...	2022-11-14 09:16:56	None	0	1	https://t.co/I1PEtx2WCd	FootballJobs	1
1592160519754252289	1045280698238676993	Job Opportunity Association: Stoke City Posi...	2022-11-14 09:20:23	None	0	1	https://t.co/ZssKnKrP3n	FootballJobs	1
1592162528280608770	1045280698238676993	Job Opportunity Association: Swansea City P...	2022-11-14 09:28:22	None	5	9	https://t.co/3Kmq1HEJOB	FootballJobs	1
1592165299629195271	1045280698238676993	Job Opportunity Association: In Netherlands ...	2022-11-14 09:39:23	None	0	2	https://t.co/nc1HBnHZqJ	FootballJobs	1
1592169053749006336	1045280698238676993	Job Appointment Delighted for Lee Baxter a...	2022-11-14 09:54:18	None	0	7	NO URLS	FootballJobs	1
1592185596788760577	1372641309484331013	Job opportunity! Swansea City (@SwansOfficia...	2022-11-14 11:00:02	None	2	2	https://t.co/cYC3OZH0vb	FootballJobs	1
1592221861857091585	63135225	Director, Game Presentation - @RaysBaseball (...	2022-11-14 13:24:08	None	1	1	https://t.co/vEPpaNfgn0	BaseballJobs	1
1592226367684235264	982292682641936385	New Job Alert: @MLB is looking for a Senior ...	2022-11-14 13:42:02	None	0	0	https://t.co/jZ6h7r7wWd	Job Alert MLB	1
1592226401641299970	982292682641936385	New Job Alert: @MLB is looking for a Vice Pr...	2022-11-14 13:42:10	None	0	0	https://t.co/0npRYcZgGp	Job Alert MLB	1
1592226421400813568	982292682641936385	New Job Alert: @MLB is looking for a Tech F...	2022-11-14 13:42:15	None	0	0	https://t.co/lM01KCspK5	Job Alert MLB	1
1592239228926431232	63135225	Peoria Sports Complex Field Maintenance Super...	2022-11-14 14:33:09	None	1	2	https://t.co/FKSq7PNbTv	BaseballJobs	1
1592283124406562818	63135225	Field Maintenance/Crew Member - Salt River Fie...	2022-11-14 17:27:34	None	0	0	https://t.co/JFO8c0Tq88	BaseballJobs	1
1592283130203111426	63135225	Manager, Premium Partnerships - @Jets (Florha...	2022-11-14 17:27:36	None	0	0	https://t.co/HwnCZqIjSp	FootballJobs	1

7. Tweet_mentions – all mentioned user ids from scraped tweets and the tweet ID and user ID of the tweeter.

```
SELECT *
FROM tweet_mentions;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

mention_row_ID	source_tweet_ID	source_user_ID	mentioned_user
1009	1600886185198759937	63135225	No Mentions
1010	1600882529695059968	63135225	No Mentions
1012	1600832622757695488	146082269986...	No Mentions
1014	1600634595531100160	156950291001...	PBSCCS
1016	1600624981204905984	143921563463...	MLB
1017	1600624978604408832	143921563463...	Pirates
1018	1600611384223764480	63135225	Cubs
1019	1600607855060856832	63135225	No Mentions
1022	1600594640151146496	63135225	astros
1023	1600591864146821120	2203526911	No Mentions
1024	1600590366033731584	63135225	Brewers
1025	1600575143528771584	63135225	Brewers
1026	1600575138218795008	63135225	Brewers
1027	1600572863223840827	996112653796...	MLB
1028	1600567969155022848	63135225	Reds
1029	1600567963505287168	63135225	No Mentions
1030	1600566072914345985	63135225	No Mentions
1031	1600556506457714689	63135225	Reds
1033	1600541663998640128	63135225	No Mentions
1034	1600539645703786502	63135225	No Mentions
1035	1600530597335953410	63135225	No Mentions

Personal Statement:

My knowledge of Twitter API, BeautifulSoup, HTML, Python, SQL, and many more have significantly improved, and the database is something I am proud of. I hope to continue this project as I outlined over the winter break and into the future.

The following pages are code PDFs for reference. In order Twitter Scrape, Teamwork Scrape, Teamwork Refreshes, Linkedin Scrape

ABSTRACT

Contained below is a functional twitter bot that accesses tweets via the Tweepy library and Twitter API. The connection is first intialized using API, Bearer, and Clent keys/IDs. Once verified, two functions are created.

1. Search_Tweets()

Used to scrape all tweets in the explore page that contain one of the keywords specified in the argument list. For this project's purpose those keywords were centered around finding job posts via twitter for jobs in the sports market. This is not a perfect method of finding job postings nor is twitter the ideal platform as many tweets are subjective and opinionated thus resulting in an endless amount of variations of verbiage. In other words, the texts scraped from twitter could use a combination of the words job, hire, and resume for purposes other than actual employment. This function returns three arrays:

1. tweet_results - Contains information regarding individual tweets that meet the keyword criteria. Some info includes tweet id, tweet date/time, tweet contents and URLS if any.

2. twitter_user_results - Contains information regarding the users of each tweet including but not limited to user id/name/username, user location, and user join date.

3. tweet_mention_results - Contains information regarding users mentioned in tweets in case a recruiter or imortant figure is mentioned. These fields are tweet id, tweet user id, and the username of the mentioned account.

4. User_Timeline()

Defined to pull tweets from all unique users pulled via Search_Tweets(). This function returns an identical array to tweet_results so that the two can be later combined into one table of tweets and tweet info. The one returned field is:

1. user_tweets - Contains identical info to tweet_results for all tweets by all users gathered in function 1 but only from the past 24 hours.

Finally a tibble is created containing the value 1 and the string "Twitter". This tibble called Sources is created with future data scraping in mind as job postings can be referenced by a source ID which states where the post was pulled from.

Once gathered, the data is passed into the SQL database which is created via the Python/MYSQL connection established using pymysql. The execute_query function is the only function defined for this portion to allow each query to be executed and a message to display if succesful.

This bot works appropriately however as previously mentioned is not ideal for job searching. Improvements could be expanding the search for greater than 24 hours and to clean out irrelvant or subjective tweets. For an example of a subjective tweet issue read the below scenario:

Scenario 1

If a twitter user did not like an NFL player or coach's performace in a game they may tweet, "Wow what a terrible call by THIS SPECIFIC NFL COACH, they deserve to be out of a Job!! @NFLTEAM let me know when you are hiring for a new head coach because Id like to apply and my resume is astounding!". While that twitter user may not be even remotely qualified nor serious about the tweet, this specific tweet would trigger keywords such as 'Job','hiring','apply', and even 'resume'. This is just one example of how difficult twitter scraping can be when looking for serious objective content.

In [ ]:

```
## Importing necessary libraries and storing API Keys as a comments for reference


# API KEY: FB7QAHPem56r0Mv83RKuPJR4Y
# API KEY SECRET: r7j3GMhjRIXtXjkNnlypjqQW60jDTkojPBPcieQlahdTHr3Q9h
# BEARER TOKEN: AAAAAAAAAAAAAAAAAAAAAMuRiQEAAAAASfWfjQlkVEO7bLR%2B2LI%2FS8s42Eo%3D4dXlyGNYxVfKk2QD1kW4C3GmDZgSZuuFbF8XYbB
# Client ID: UkxpeUtFN0pDanNWRS1tSk9ZTXY6MTpjaQ
# Client Secret: wTSnz2QkfIittU70Vw0rhKFm6FTjkTbeMTHzZXUUwHK4rcd7S_

import pandas as pd
import numpy as np
import csv
import os
import datetime, time
import pytz
#!pip install wget
import wget
#!pip install -- tweepy
import tweepy
## !pip install mysql.connector
# import mysql.connector
## !pip install pymysql
import pymysql
pymysql.install_as_MySQLdb()
import MySQLdb
import xlrd
```

In [ ]:

```
## Installation / Access to Twitter Account, Prints Authentification Succesful if connection is valid

consumer_key = 'FB7QAHPem56r0Mv83RKuPJR4Y'
consumer_secret = 'r7j3GMhjRIXtXjkNnlypjqQW60jDTkojPBPcieQlahdTHr3Q9h'
access_token = '1583139235342811138-CNcRKakrtwYsJdRhjNvUqXsrE6mha8'
access_token_secret = '1kpR1zSr9wlE1wk8MexeytDHlTzUgBM31qzNsSmF4mzlX'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit = True)

try:
    api.verify_credentials()
    print("Authentication Succesful")
except:
    print("Error during authentication, check keys and try again!!")
```

Once Authenticated Functions are Defined as seen below:

In [ ]:

```
## Function to search for the keywords and retrieve tweet contents and properties as well as user information. Returns tw
## lists one for tweet info and one for user info.

def search_tweets(keywords):
    tweet_results = []
    twitter_user_results = []
    tweet_mention_results = []
    i = 0
    for query in keywords:

        for tweet in tweepy.Cursor(api.search_tweets, q = query, count=5,
                                   tweet_mode = 'extended').items():

            if tweet.full_text.startswith('RT @'):
                pass
                #text = tweet.retweeted_status.full_text
                #tweet_results.append(s + 'RT @' + text)
            else:
                i += 1
                s = (i)

                if(tweet.entities['user_mentions'] == []):
```

```
                    if tweet.entities['urls'] == []:
                        tweet_results.append([s, tweet.id, tweet.full_text, tweet.created_at, tweet.place, tweet.retweet_

                        twitter_user_results.append([tweet.user.id, tweet.user.screen_name, tweet.user.name, tweet.user.c
                                     tweet.user.location, tweet.user.created_at, tweet.user.favourites_count, tweet.user
                                     tweet.user.profile_image_url])

                        tweet_mention_results.append([tweet.id,tweet.user.id, 'No Mentions'])

                    else:
                        tweet_results.append([s, tweet.id, tweet.full_text, tweet.created_at, tweet.place, tweet.retweet_

                        twitter_user_results.append([tweet.user.id, tweet.user.screen_name, tweet.user.name, tweet.user.c
                                     tweet.user.location, tweet.user.created_at, tweet.user.favourites_count, tweet.user
                                     tweet.user.profile_image_url])

                        tweet_mention_results.append([s, tweet.id,tweet.user.id, 'No Mentions'])


                else:
                    if tweet.entities['urls'] == []:
                        tweet_results.append([s, tweet.id, tweet.full_text, tweet.created_at, tweet.place, tweet.retweet_

                        twitter_user_results.append([tweet.user.id, tweet.user.screen_name, tweet.user.name, tweet.user.c
                                     tweet.user.location, tweet.user.created_at, tweet.user.favourites_count, tweet.user
                                     tweet.user.profile_image_url])

                        tweet_mention_results.append([tweet.id,tweet.user.id, tweet.entities['user_mentions'][0]['screen_

                    else:
                        tweet_results.append([s, tweet.id, tweet.full_text, tweet.created_at, tweet.place, tweet.retweet_

                        twitter_user_results.append([tweet.user.id, tweet.user.screen_name, tweet.user.name, tweet.user.c
                                     tweet.user.location, tweet.user.created_at, tweet.user.favourites_count, tweet.user
                                     tweet.user.profile_image_url])

                        tweet_mention_results.append([s, tweet.id, tweet.user.id, tweet.entities['user_mentions'][0]['scr

    return(tweet_results,twitter_user_results,tweet_mention_results)
```

In [ ]:

```
## Function to search all users and pull their tweets over the last 24 hours. Returns same info as tweet table in other f

def user_timeline(user_names):
    user_tweets = []
```

```
    print(user_names)

    now = datetime.datetime.today()
    date_since = datetime.datetime.strptime(time.strftime("%Y-%m-%d"), "%Y-%m-%d")
    day_ago_pre = now - datetime.timedelta(hours=24)
    day_ago = day_ago_pre.replace(tzinfo=pytz.utc)

    for name in user_names:
        count = 0
        try:
            for tweet in tweepy.Cursor(api.user_timeline, screen_name = name,
                                       tweet_mode = 'extended').items():

                if tweet.full_text.startswith('RT @'):
                    pass

                elif (tweet.created_at < day_ago):
                    break

                else:
                    count = count + 1
                    if tweet.entities['urls'] == []:
                        user_tweets.append([count, tweet.id, tweet.full_text, tweet.created_at, tweet.place, tweet.retwee
                    else:
                        user_tweets.append([count, tweet.id, tweet.full_text, tweet.created_at, tweet.place, tweet.retwee
        except:
            pass
    return(user_tweets)
```

Initializing dataframes # UPDATE: REMOVED BECAUSE DEEMED NOT NECESSARY #Tweets_df = pd.DataFrame(columns = ['index', 'tweet_ID', 'tweet_contents', 'tweet_date_time', 'tweet_location', 'keywords']) #Tweet_User_df = pd.DataFrame(columns = ['user_id', 'user_Handle','user_name','user_bio','user_location','user_follower_count','user_friend_count','twitter_profile_img_url'])

Once Functions are completed, Data is Collected as seen below:

In [ ]:

```
## Works 100% with few keywords, Rate Limit Not accounted for Yet
keywords_list = ["following roles sports","Job Opportunity Sports", "Jobs in Sports","SoccerJobs"]
tweet_search_rs,tweet_user_rs, tweet_mention_rs = search_tweets(keywords_list)
##"Football Jobs", "Basketball Jobs", "Baseball Jobs", "Job Alert MLS", "Job Alert NFL", "Job Alert NBA", "Job Alert MLB"
```

In [ ]:
```
Tweets_df = pd.DataFrame(tweet_search_rs, columns = ['index', 'tweet_ID', 'tweet_contents', 'tweet_date_time', 'tweet_loc

Tweet_User_df = pd.DataFrame(tweet_user_rs,columns = ['user_ID', 'user_handle','user_name','user_bio','user_location','us

Tweet_Mentions_df = pd.DataFrame(tweet_mention_rs, columns = ['mention_row_ID','tweet_ID', 'source_user_ID', 'mentioned_u

Tweets_df['tweet_user_ID'] = (Tweet_User_df['user_ID'])
```

In [ ]:
```
user_tweets_rs = user_timeline(Tweet_User_df.user_handle.unique())
```

In [ ]:
```
User_Tweets_df = pd.DataFrame(user_tweets_rs, columns = ['index', 'tweet_ID', 'tweet_contents', 'tweet_date_time', 'tweet
User_Tweets_df['keywords'] = np.nan
```

In [ ]:
```
User_Tweets_df = User_Tweets_df.reindex(columns = ['tweet_ID', 'tweet_user_ID', 'tweet_contents', 'tweet_date_time', 'twe
Tweets_df = Tweets_df.reindex(columns = ['tweet_ID', 'tweet_user_ID', 'tweet_contents', 'tweet_date_time', 'tweet_locatic
```

In [ ]:
```
Tweets_Final_df = pd.concat([Tweets_df,User_Tweets_df])
```

In [ ]:
```
Tweets_Final_df['source_identifier'] = 1
Tweets_Final_df = Tweets_Final_df[(Tweets_Final_df.duplicated(['tweet_ID'])) == False]
Tweet_Mentions_df = Tweet_Mentions_df[(Tweet_Mentions_df.duplicated(['tweet_ID'])) == False]
Tweet_User_df = Tweet_User_df[(Tweet_User_df.duplicated(['user_ID'])) == False]
```

In [ ]:
```
Tweets_Final_df['tweet_contents'] = Tweets_Final_df['tweet_contents'].str.replace("'","").str.replace('"','')
```

In [ ]:
```
Sources = pd.DataFrame({'source_ID': [1], 'source_name': ['Twitter']})
```

With the Data Collected, the database connection is established and all data is imported as seen below:

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(query_statement = ("CREATE Database IF NOT EXISTS JobsinSports"))
```

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

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

In [ ]:
```
execute_query("CREATE TABLE IF NOT EXISTS Tweets(tweet_ID BIGINT PRIMARY KEY NOT NULL UNIQUE, tweet_user_ID BIGINT NOT NU
```

In [ ]:
```
execute_query("CREATE TABLE IF NOT EXISTS Sources(source_ID BIGINT PRIMARY KEY NOT NULL UNIQUE, source_name VARCHAR(55));
```

In [ ]:
```
execute_query("CREATE TABLE IF NOT EXISTS Tweet_User(user_ID BIGINT PRIMARY KEY NOT NULL UNIQUE, user_handle VARCHAR(25)
```

In [ ]:
```
execute_query("CREATE TABLE IF NOT EXISTS Tweet_Mentions(mention_row_ID INT PRIMARY KEY NOT NULL UNIQUE, source_tweet_ID
```

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 Tweet_User_df.iterrows():
    execute_query('INSERT INTO Tweet_User (user_ID, user_handle, user_name, user_bio, user_location, user_join_date, user
```

In [ ]:
```
# Tweets_df.shape

for i,j in Tweets_Final_df.iterrows():
    execute_query('INSERT INTO Tweets (tweet_ID, tweet_user_ID, tweet_contents, tweet_date_time, tweet_location, tweet_rt
```

In [ ]:
```
for i,j in Tweet_Mentions_df.iterrows():
    try:
        execute_query('INSERT INTO Tweet_Mentions (mention_row_ID, source_tweet_ID, source_user_ID, mentioned_user) VALUE
    except:
        pass
```

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

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', 'Heac
```

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()
```

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()
```

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()
```

REFERENCES

Scraped Data

https://en.wikipedia.org/wiki/Major_League_Soccer

https://en.wikipedia.org/wiki/National_Basketball_Association

https://en.wikipedia.org/wiki/Major_League_Baseball

https://en.wikipedia.org/wiki/National_Football_League

https://en.wikipedia.org/wiki/National_Hockey_League

https://www.teamworkonline.com/jobs-in-sports?page=1

https://www.linkedin.com/jobs/search?keywords=National%20Basketball%20Association&location=United%20States&geoId=103644278&trk=public_jobs_jobs-search-bar_search-submit&position=1&pageNum=0

Sites Referenced for Coding

https://sparkbyexamples.com/pandas/pandas-drop-first-row-of-dataframe/#:~:text=Use%20DataFrame.,-drop()%20to&text=drop()%20function%20is%20used,change%20to%20the%20existing%20DataFrame.

https://www.statology.org/case-statement-pandas/

https://note.nkmk.me/en/python-pandas-dataframe-for-iteration/

https://cmdlinetips.com/2020/06/pandas-explode-convert-list-like-column-elements-to-separate-rows/#:~:text=Pandas%20explode()%20to%20separate,row%20for%20each%20of%20them.

https://stackoverflow.com/questions/61349328/python-pandas-dataframe-doesnt-update-value-each-for-loop-cycle-why

https://pythontic.com/pandas/serialization/mysql

https://docs.python.org/3/library/datetime.html