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:

<u>Job Postings</u> 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.

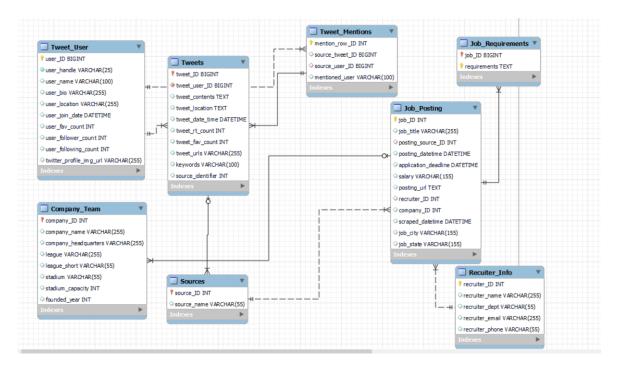
<u>Company Information</u> from the company site (Likely teams). This data will include at a minimum, Team/Company Name/ID, League, Size, Market Size, etc.

<u>Required Skills</u> 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.

<u>Recruiter Information</u> 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

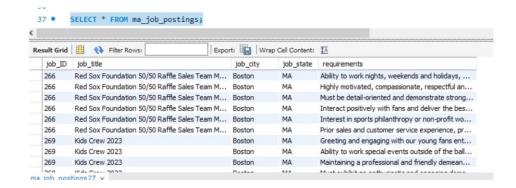
ON p.company_ID = c.company_ID

WHERE p.job_title LIKE "%Spring%";

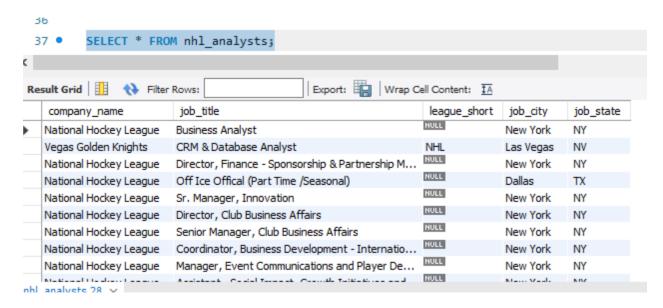
1.

R	esult Grid	Name of the Filter Rows:	[
	league_short	league_postings	
•	NULL	335	
	MLB	40	
	NFL	15	
	NBA	14	
	NHL	11	
	MLS	10	

2.



3.



4.

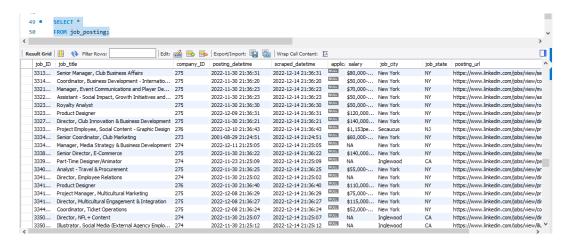


5.

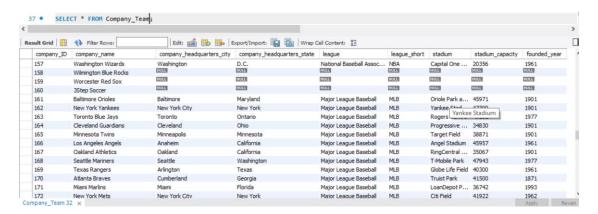


4 Main Tables

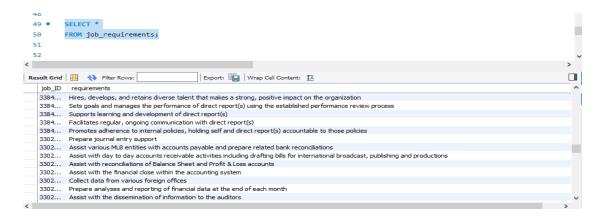
 Job_posting – contains all job titles, posting dates, scraped dates, application deadlines, salary estimates, location (city and state) source_ID, company_ID, and posting website.



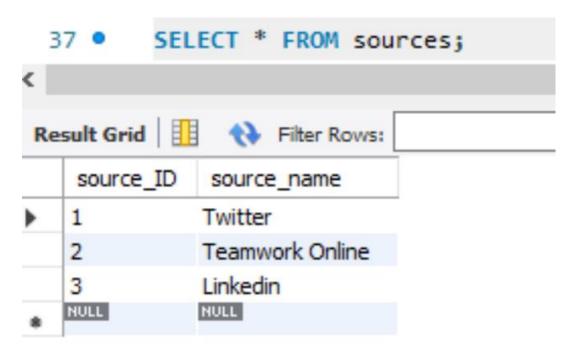
 Company_Team - Company_ID, company_name, headquarters (~location city and state), league,league_short (abbreviation), stadium (name), capacity, founded_year.



3. Job_requirements

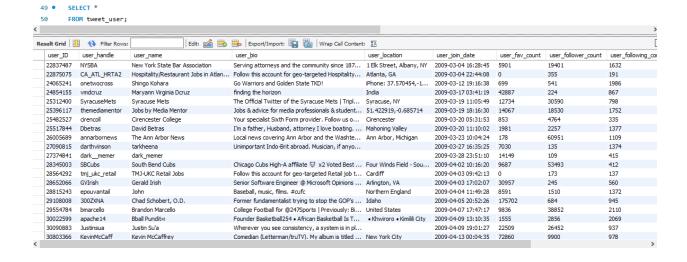


4. Sources – simple source reference table to identify source name where each job posting was found and includes source_ID, source_name

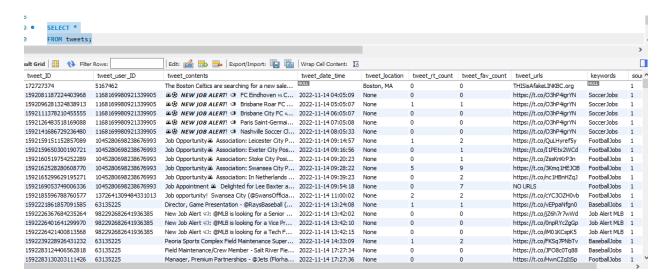


Finally, below are some of the Less used tables in the database that are included due to the twitter API assignment being a prominent piece of the semester. These tables were not the focus of this database but will be improved moving forward.

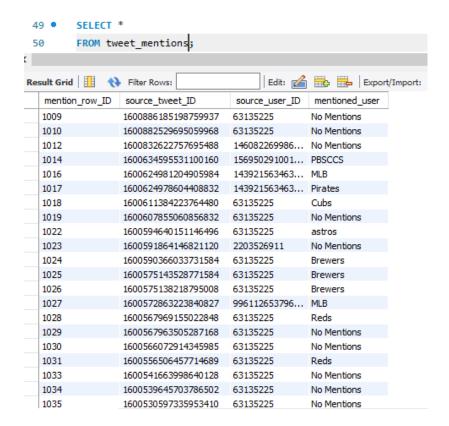
5. Tweet_user – user profile information for all users of scraped tweets.



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



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



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