SAN JOSE STATE UNIVERSITY

Database Systems for Analytics (DATA 225)

Instructor – Simon Shim

Lab Project - 2

Report on

LinkedIn Job Posting Analysis

Group 8

Submitted By

Aiswarya Raghavadesikan

Neha Thakur

Saumya Varshney

Shikha Singh

Venkata Sai Sreelekha Gollu

TABLE OF CONTENTS

I. Introduction

The project focuses on analyzing LinkedIn job postings, offering valuable insights into the ever-changing job market. LinkedIn is an employment focused social media platform which caters to the requirements of both employers and employees. With the dynamic job market, LinkedIn is offering insights into the skills necessary to excel in the contemporary professional world. With over 800 million users worldwide, LinkedIn provides a vast source of data on job opportunities. Each individual posting contains several valuable attributes, including the title, job description, salary, location, application URL, and work-types (remote, contract, etc.), in addition to separate files containing the benefits, skills, and industries associated with each posting.

This project aims to extract meaningful information, including job trends by industry, location, skills in demand, and changes over time. LinkedIn Database offers the distribution of job postings across various companies and industries. Skill distribution and emerging job roles could provide great insights into the changing and evolving needs of companies. This database helps to tackle real-world employment challenges, ultimately enhancing our understanding of today's job market and assisting professionals in making informed business decisions.

II. Problem Statement

The analysis and management of the LinkedIn job postings dataset pose a significant challenge due to the vast and dynamic nature of the data. This dataset encompasses a diverse range of job categories, industries, and geographic locations, making it challenging to extract meaningful insights and trends. Key issues include the identification of emerging job roles, the assessment of skill sets in demand, the determination of competitive salary ranges, and the discernment of dynamic market trends. Consequently, addressing these challenges is critical to enabling informed decision-making in the job market and enhancing the effectiveness of job search and talent acquisition processes.

III. Solution Requirement

In order to efficiently understand the dataset, the first step is to clean the given dataset to more meaningful data.

Elucidating a given job posting will immensely help a job seeker to further understand the company and the current job market trends. Our data insights provides a job seeker with quick and valuable information regarding an active job posting, its associated salary range, work mode and work type. Analyzing the filtered data will help a job seeker to make an efficient decision based on the dynamic job market trends.

Job companies and recruiters can also simultaneously benefit from this which includes viewing the number of applicants for a given job posting, identifying the potential candidates, closing out the job posting promptly.

Functional analysis helps in processing the given dataset by fetching the appropriate and relevant information from the table and delivering profound insights to the job poster and a job seeker. However, the recruiter's contact information, current employers of a company for internal referrals, job seeker information are some of the aspects which are out of the scope of this project.

IV. Limitations

Scaling Considerations: As dataset grows, considerations for horizontal scaling (sharding) become more important. Ensure our MongoDB deployment is appropriately scaled to handle increased data volumes.

Limited Historical Data Access to historical job postings data on LinkedIn can be restricted, making it challenging to conduct long-term studies or track trends over time. Job postings can change rapidly, affecting the accuracy of historical analysis.

Privacy Concerns: Handling LinkedIn data raises privacy concerns. Respecting the privacy of individuals and organizations within the dataset and adhering to LinkedIn's terms of service is crucial to maintain ethical data analysis practices.

Frequency of Data Updates: Frequent updates and deletions of data on LinkedIn necessitate investigation into how often the data can be updated and retrieved. Regular updates are crucial for maintaining the relevance and accuracy of the analysis.

In some cases, company details are not present completely corresponding to job postings. This might lead to inaccurate analysis for companies.

This dataset lacks information about job seekers. Due to which it is difficult to draw insights for aspiring candidates.

V. Conceptual Design

To implement LinkedIn job posting analysis, Here are some of the key entities mentioned which will be helpful in drawing meaning full insights.

Getting to know the raw data.

- Job Postings
- Job industries
- Job skills
- Benefits
- Companies
- Company Specialties
- Company Industries
- Employee counts
- **Job Postings:** It is a central repository for job listings and is commonly used in job search platforms. The attributes used in this table are job_id , company_id , title, description, skills_desc, work_type, location, currency, remote_allowed, sponsored, max_salary, med_salary, min_salary, pay_period, compensation_type, formatted_work_type, formatted_experience_level, applies, views, original_listed_time, listed_time, expiry, closed_time, posting_domain, job_posting_url, application_url , application_type
- **Job industries:** The job_industries table is used to associate job postings with specific industries. It serves as a bridge between job postings and industries, allowing

users to categorize job listings based on the industry they belong to. The attributes used in this table are, job_id, industry_id

- **Job skills:** This table contains information about the skills required for various job postings. The attributes in this table are, job_id, skill_abr
- **Benefits:** The benefits table stores data related to benefits offered by companies in job postings. It includes information about the types of benefits (e.g., 401K, Medical Insurance) and whether these benefits are explicitly tagged or inferred from the job posting text. The attributes in this table are, job_id, type, inferred
- **Companies:**The companies table contains data about various companies and serves as a reference for job postings. It includes details about the company's name, description, size, location, and more. This table is typically linked to the job_postings table to associate job postings with specific companies. The attributes in this table are, company_id, name, description, company_size, address, city, state, country, zip_code, url.
- **Employee counts:** This table tracks the number of employees at each company. It often includes information about the company's follower count on a platform. Employee counts are crucial for understanding the size and workforce of different companies. The attributes in this table are, company_id, time_recorded, employee_count, follower_count.
- **Company specialities:** This table associates companies with their specializations or areas of expertise. The attributes in this table are, company_id, speciality.
- **Company_industries:** The company_industries table links companies with the industries they are associated with. The attributes in this table are company_id, industry.

A. Data Cleaning

As part of the data cleaning process, we inspected each collection for duplicate rows in the raw data.

*1) **Duplicates**:* Every job posting and company should have unique rows in all the collections. By removing the duplicates we are reducing the ambiguity from the dataset.

- **job_skills** - There were no duplicate rows.
- **job_industries** - There were no duplicate rows.
- **benefits** - There were no duplicate rows.
- **job_postings** - There were no duplicate rows.
- **companies** - There were 33 duplicate rows.
- **employee_counts** - There were 3356 duplicate rows.
- **company_industries** - There were 9877 duplicate rows.
- **company_specialities** - There were 85844 duplicate rows.

B. Enforcing constraints

Job skills, job industries and benefits were associated with job postings with foreign key job_id. There were some job skills and job industries document whose job_id didn't exist in job postings collection. These documents were removed because there cannot be skills and industries without job posting.
Similarly, Employee counts, company industries and company specialties are associated with company collection with foreign key company_id. There were some industries, specialties and employee counts whose company_id didn't exist in company collection. These documents were removed because there cannot be specialties, employee count and industries without company details.

- **job_skills** - There were 932 rows with job_id that were not present in job_postings.
- **job_industries** - There are 1040 rows with job_id that were not present in job_postings.
- **benefits** - There are no rows where the job_id is not present in job_postings.
- **job_postings** - There were 50 company_ids in job_postings that didn't exist in companies table. Since a job_posting can exist without a company, we replaced these 50 company_ids with NULL.
- **companies** - There are no foreign keys in companies.
- employee_counts - There were 50 rows with company_id that were not present in companies.
- **company_industries** - There were 50 rows with company_id that were not present in companies.
- **company_specialities** - There were 384 rows with company_id that were not present in companies.

C. Denormalization

The process of strategically introducing redundancy in the data to improve the read performance of the database. This is called denormalization.
There were 8 collections in our raw data:
job skills
job industries
benefits
job postings
companies
employee counts
company industries
company specialties

We have denormalized these 8 collections into 2 – job postings and companies.
Job skills, job industries, benefits were combined inside job postings as embedded documents because they were all connected with the job_id. It would be simple to combine all these collections into 1 to avoid expensive look ups (joins). Similarly, employee counts, company industries, company specialties were combined inside companies as embedded documents because they were

all connected with the company_id. It would be simple to combine all these collections into 1 to avoid expensive look ups (joins).
We now have 3 options:

1) job_postings within company one collection
2) company within job_postings one collection
3) two separate collections company and job_postings

If we combined companies within job postings or job postings inside companies to make 1 collection then in either case we were losing some company details or some job posting details. Also, if we go with option 1 then it becomes primarily focused on companies and if we go with option 2, then it is more job postings centric. Hence, we proceeded with option 3 and have 2 final collections post denormalization.

D. Data Model

To implement LinkedIn job posting analysis, here are some of the key entities mentioned which will be helpful in drawing meaning full insights. The two major tables used after denormalization are,

1) Job postings
2) Companies

1) **job_postings** It is a central repository for job listings and is commonly used in job search platforms. The attributes used in this table are job id , company id , title, description, work_type, location, remote allowed, sponsored, salary details: currency, max salary, med salary, min salary, pay_period and compensation type, benefits: inferred and types, formatted work type, views, original listed time, listed time, expiry, job posting url, application type, industry_ids, skills, skills desc, formatted experience level, applies, application url ,closed time, posting domain.
PRIMARY KEY: job_id
FOREIGN KEY: company_id

- **id (ObjectId):** The unique identifier for each document in the collection. It's an automatically generated ObjectId by MongoDB.
- **job_id (Number):** A numerical identifier for the job posting, unique for each job.
- **title (String):** The job title or position name.
- **description (String):** A text field containing the job description, providing details about the responsibilities, requirements, and other relevant information about the job.
- **work_type (String):** Describes the type of work associated with the job (e.x., full-time, part-time, contract).
- **location (String):** Represents the geographical location where the job is based.
- **remote_allowed (Number):** Indicates whether remote work is allowed or not. It's represented as a number, boolean with 1 for true and 0 for false.
- **sponsored (Number):** Indicates whether the job posting is sponsored. Like remote_allowed, it's represented as a number.
- **formatted_work_type (String):** A formatted or human-readable representation of the work type. Ex: Full-time
- **views (Number):** The number of views or impressions the job posting has received.
- **original_listed_time (String):** The original time when the job was listed.
- **listed_time (String):** The current listed time for the job.
- **expiry (String):** Represents the expiration date or time for the job posting.
- **job_posting_url (String):** The URL or link to apply for a job posting.
- **application_type (String):** Describes the type of application process (example, OffsiteApply, ComplexOnsiteApply).
- **salary_details (Object):** An object containing details about the salary, such as currency, maximum salary, minimum salary, pay period, compensation type, and median salary.
- **currency (String):** Represents the currency in which the salary is denominated. For example, it could be "USD" for U.S. dollars or "EUR" for Euros.
- **max_salary (Number):** Represents the maximum salary offered for the job posting. It indicates the highest amount an employee can earn for this position.
- **min_salary (Number):** Represents the minimum salary offered for the job posting. It indicates the lowest amount an employee can earn for this position.
- **pay_period (String):** Describes the period at which the salary is paid. Common values might

include per year, per month,per week or per hour. (e.x., yearly, monthly, hourly).

- **compensation_type (String):** Describes the type of compensation structure associated with the job posting. ex: BASE_SALARY.

- **med_salary (Number):** Represents the median salary for the job posting. The median is the middle value in a set of numbers and is useful for understanding the central tendency of the salary distribution.

- **industry_ids (Array of Numbers):** An array of numerical identifiers representing the industries associated with the job.

- **skills (Array of Strings):** An array of strings representing the skills required for the job.

- **benefits (Array of Objects):** An array of objects representing benefits associated with the job. Each object may have inferred and types fields.

- **Inferred (Number):** Whether the benefit was explicitly tagged or inferred through text by LinkedIn type(Array of Strings): Type of benefit provided (401K, Medical Insurance, etc)

- **company_id (Number):** A numerical identifier representing the company associated with the job posting.

- **applies (Number):** The number of job applications received for the job posting.

- **application_url (String):** The URL or link to the job application.

- **formatted_experience_level (String):** formatted representation of the experience level required for the job. Ex: entry, associate, executive,

- **posting_domain (String):** The domain where the job posting is originally posted.

- **closed_time (String):** The time when the job posting was closed.

- **skills_desc (String):** A description details related to the required skills.

2) **Companies:**

The companies table contains data about various companies and serves as a reference for job postings. This table is typically linked to the job postings table to associate job postings with specific companies. It includes details about the company_id, company's name, description, company size, url, address_details: address, city, state, country and zip_code, employe_counts: employee_count, follower_count, time_recorded, specialities, industries.

PRIMARY KEY: company id

Along with these primary key's while executing queries in mongodb it by default creates the primary key "_id" for the tables.

- **_id (ObjectId):** The unique identifier for each document in the collection. It's an automatically generated ObjectId by MongoDB.

- **company_id (Number):** A numerical unique identifier representing the company.

- **name (String):** The name of the company.

- **description (String):** A text field containing a description of the company.

- **company_size (Number):**
- Indicates the size of the company, represents the number of employees.

- **url (String):** The URL associated with the company, which could be the company's website.

- **address_details (Object):** An object containing details about the company's address, including fields like address, city, state, country, and zip code.

- **address (String):** Represents the street address or location of the company.

- **city (String):** Represents the city where the company is located.

- **state (String):** Represents the state or province where the company is located.

- **country (String):** Represents the country where the company is located.

- **zip_code (String):** Represents the ZIP code associated with the company's address.

- **specialities (Array of Mixed):** An array of mixed types representing the specialities of expertise associated with the company. Ex: advertising, recruiting etc. industries (Array of Strings): An array of strings representing the industries in which the company operates.

- **employee_counts (Array of Objects):** An array of objects representing employee counts and follower

counts for the company. Each object may have fields such as employee_count and follower_count.

- **employee_count (Number):** Number of employees at company.
- **time_recorded (String):** A string representing the time when the record was recorded. Unix time of data collection.
- **follower_count (Number):** Number of company followers on LinkedIn.

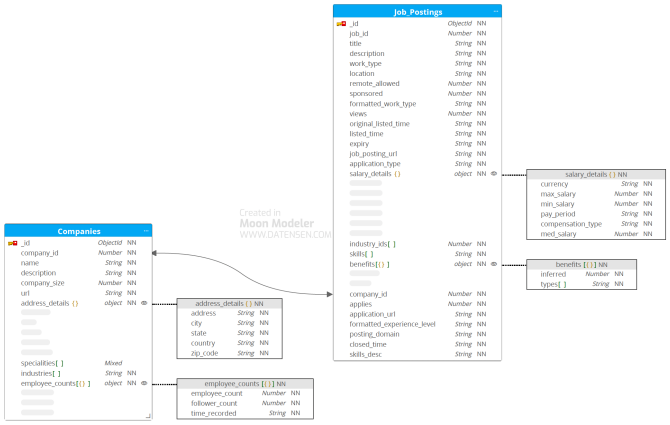

Fig. 1. Data Model

VI. Functional Analysis and Components

Linkedin Job posting analysis system can be used by a wide range of users, including job seekers, data analysts, researchers, and HR representatives. Its various functionalities can be accessed by different individuals. Following are details explaining how it works for different people:

- Job seekers can refine their job search by considering factors such as the compensation structure (Hourly, Monthly, or Yearly) and the salary range, enabling them to make well-informed decisions when applying for positions. In addition, job seekers are empowered with a versatile array of filters, allowing them to tailor their search based on work arrangements (Full-Time, Part-Time, Internship, or Contract) and the spectrum of benefits offered by the organizations.
- Detailed insights into the companies, including their core competencies and industry involvement, provide job seekers with a comprehensive understanding of potential employers.
- Human Resources (HR) representatives can conduct a comprehensive analysis of analogous job listings offered by various organizations. This analysis encompasses an evaluation of the requisite skill sets and qualifications necessary for the creation of new job postings.
- Data analyst/Researcher can analyze emerging job roles and the requisite skill sets that are in high demand. Furthermore, they undertake the evaluation of prevalent salary ranges offered by various companies in the job market.

VII. MQL QUERIES

1) **Count of job postings by job industry.**

db.job_postings.aggregate([$unwind:"$industry_ids",$group: _id:"$industry_ids", count: $sum :1]);

2) **Job postings with a specific salary range.**

db.job_postings.find("salary_details.min_salary" :$gte:30000,"salary_details.max_salary": $lte:100000,job_id:1,"salary_details.min_salary":1, "salary_details.max_salary":1);

3) **Job postings having 'senior' positions with details of job location and salary.**

db.job_postings.find(title: /senior/i, _id: 0, job_id: 1, location: 1, title: 1, salary_details: 1)

4) **Top 3 job postings with highest number of views.**

db.job_postings.aggregate([$sort:"views":-1,$limit: 3, $project:"job_id":1,"title":1,"views":1,"work_type":1, "location":1,"expiry":1,"application_type":1, "job_posting_url": 1,"_id": 0])

5) **Locations with the Highest Number of Remote Work Opportunities.**

db.job_postings.aggregate([$match:remote_allowed:1 ,$group: _id: "$location", totalRemoteJobs: $sum: 1 ,$sort: totalRemoteJobs: -1 ,$limit:5,$project:_id: 0, location: "$_id", totalRemoteJobs: 1])

6) **Top 5 Locations with the Sponsored Job Postings.**

db.job_postings.aggregate([$match: sponsored: 1 ,$group: _id: "$location", total_sponsored_jobs: $sum: 1 ,$sort: total_sponsored_jobs: -1 ,$limit: 5 , $project: _id: 0, location: "$_id", total_sponsored_jobs: 1])

7) **Company with highest employee count where specialities is "analytics".**

db.companies.aggregate([$match:"specialities":"analytics", $sort:"employee_counts":-1, $limit:1,$project:_id:0,company_name:"$name", company_size:"$company_size",employeeCount: "$employee_counts"])

8) **Find top 10 company industries having highest job postings.**

```
db.getCollection('job_postings').aggregate([$lookup: from: 'companies',localField: 'company_id', foreignField: 'company_id',as: 'CompanyDetails' , $unwind: path: '$CompanyDetails' , $unwind: path: '$CompanyDetails.industries' , $group: _id: '$CompanyDetails.industries',countInd: $sum: 1 , $sort: countInd: -1, $limit: 10 ]);
```

9) **Finding the top 3 companies possessing maximum employee counts.**

```
db.companies.aggregate([$unwind:"$employee_counts", $sort:"employee_counts.employee_count":-1,$group:_id:"$name",maxEmployeeCount:$max:"$employee_counts.employee_count", $sort:maxEmployeeCount:-1,$limit:3,$project:_id:0,name:"$_id", maxEmployeeCount:1]);
```

10) **Finding the top 20 specialities and sorted based on the maximum count of specialities.**

```
db.companies.aggregate([$unwind:"$specialities" ,$group: _id: "$specialities",count:$sum: 1,$sort:count: -1, $limit:20 ]);
```

11) **Finding the companies that are offering FULL TIME positions as remote opportunities along with the company name, title.**

```
db.job_postings.aggregate([$match:"work_type": "FULL_TIME","remote_allowed":1,$lookup: from:"companies",localField:"company_id", foreignField:"company_id", as:"company", $unwind:"$company",$project:_id:0,title:1,work_type:1, remote_allowed:1,company_name:"$company.name"])
```

12) **Count of companies for the respective specialties: Cloud, IOT, AI and Network.**

```
db.companies.aggregate([$match:specialities: "internet of things",$group:_id:null, count:$sum:1]);
db.companies.aggregate([$match:specialities: "cloud",$group:_id:null,count:$sum:1]);
db.companies.aggregate([$match:specialities: "iot",$group:_id:null,count:$sum:1]);
db.companies.aggregate([$match:specialities: "ai",$group:_id:null,count:$sum:1]);
db.companies.aggregate([$match:specialities: "network",$group:_id:null,count:$sum:1]);
```

VIII. VISUALIZATION

1) **Distribution of job postings based on industries.**

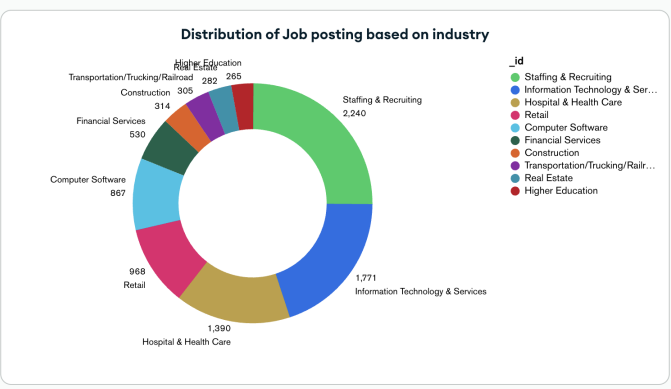

Fig. 2. Distribution of job postings based on industries

This Donut chart displays the number of job postings based on industry. Here we can observe that staffing and recruiting has highest number of job posting. It can help the job seeker to identify the emerging industries, it allows them to target their career goals.

2) **Visualizing Count of jobs based on location**

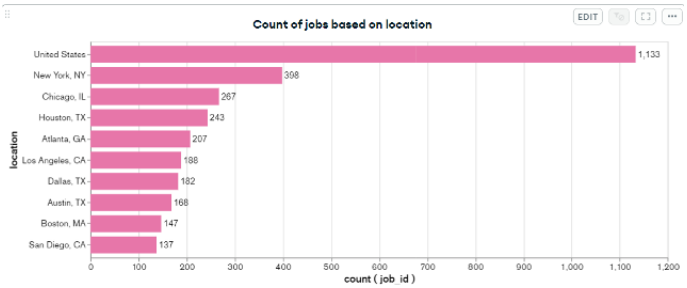

Fig. 3. Visualizing Count of jobs based on location

The visualization focuses on the count of jobs based on location and retrieves the top 10 locations with the highest job counts in MongoDB Atlas. Here job id is taken in X axis with an aggregation of count and location is taken on Y axis with a limit of 10. The visualization provides valuable insights into the geographical concentration of job opportunities helping in the strategic decision-making for job seekers and employers looking to understand and target high-demand regions within the dataset.

3) **Visualizing number of jobs belong to each pay period and their respective max salary**

In this representation, X axis represents the pay period and Y axis represents the Values denoting the pay period and the max salary. The line graph represents the max salary. The least amount of job postings are observed in the MONTHLY pay period category and

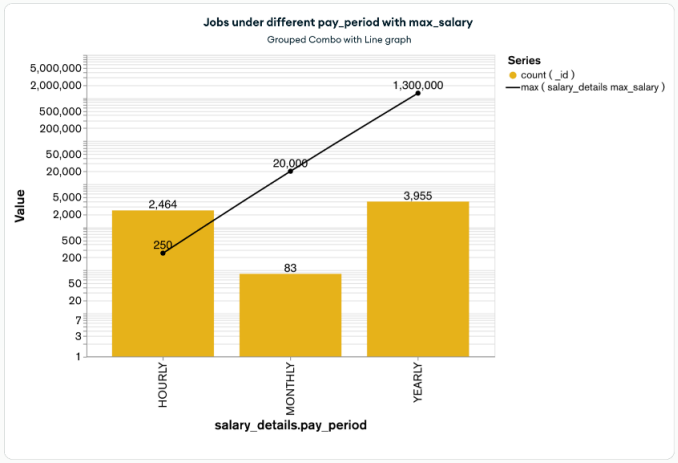

Fig. 4. Visualizing number of jobs belong to each pay period and their respective max salary

its corresponding max salary of $20,000. The highest number of job postings were for Yearly pay period and its corresponding max salary is $1,300,000.

4) **Visualizing the distribution of top 10 skills**

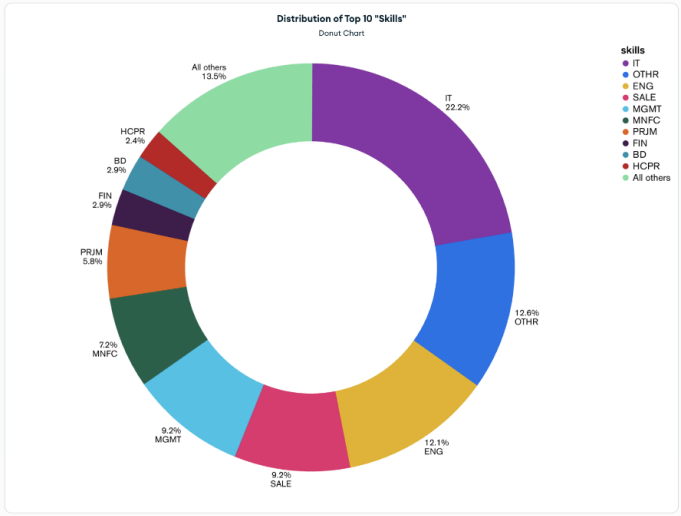

Fig. 5. Visualizing the distribution of top 10 skills

Here we can observe the distribution of top 10 skills along with the data points percentage. Here the majority of the job postings requires IT skills having 22.2%

5) **Country wise representation of job postings**
Here the Geo choropleth explains that the maximum job postings are from the country United States (North American continent) and the least count of jobs are from the countries Honduras (Central American Continent), Brazil (South American Continent), Germany (European Continent) and India (Asian continent)

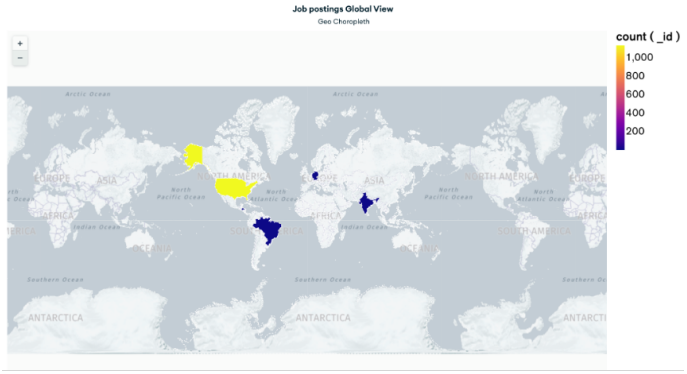

Fig. 6. Country wise representation of job postings

6) **Number of jobs based on experience level and work type**

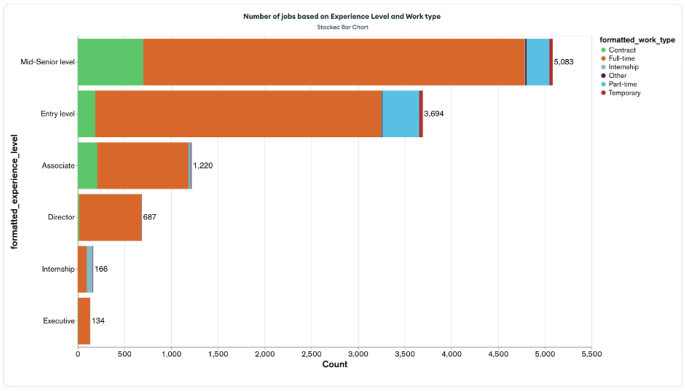

Fig. 7. Number of jobs based on experience level and work type

Here we can observe that almost across all the experience level categories the majority of work type is of Full-Time. Very minimal counts of 'Contract' positions can be observed in the Director level experience category. For the internship experience level, work types of Full-Time are more than the work types of internship

7) **Industries spread across various companies**
Here the Word Cloud feature is used to understand the Industries spread across various companies. We observe the Top 3 major industries from various companies are "Information Technology Services", "Staffing & Recruiting" and "Hospital & Health Care". (Refer fig 8)

8) **Top 20 specialities associated with companies**
From the visualization we are able to understand that specialty staffing has the highest count of company id's with almost 250 companies, followed by recruiting with over 240 companies. This information is valuable for understanding the prevalent areas of expertise among

Fig. 8. Number of jobs based on experience level and work type

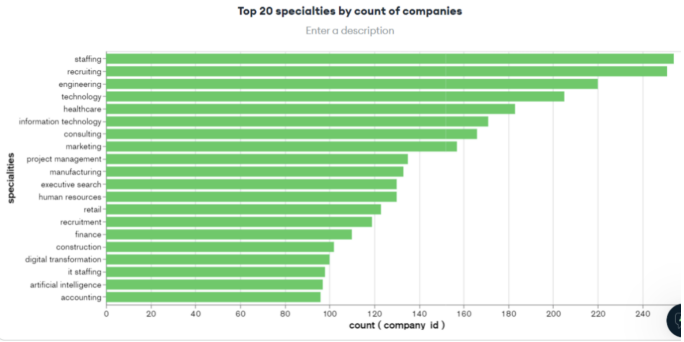

Fig. 9. Number of jobs based on experience level and work type

companies and helps in decision-making for both businesses and professionals navigating the job market. (Refer fig 9)

IX. Connection To MongoDB in Cloud

- **Create a MongoDB Atlas Account:** Go to the MongoDB Atlas website (https://www.mongodb.com/cloud/atlas). Use Free tier to create an account.
- **Create a New Cluster:** In database deployments click on create cluster button, Choose the cloud provider, region, and cluster configuration according to requirements. Then click create cluster.
- **Configure Security Settings:** Go to Database access, click on the ”ADD NEW DATABASE USER” button to create a user with the necessary permissions.
- **Get Connection String:** In the database dashboard, click on the ”Connect” button. Choose ”Connect Your Application.” Using compass. Choose the version of compass and copy the connection string, then open MongoDB Compass. The connection string should look like: mongodb+srv://saumyavarshney: ”password” @linkedinjobanalysis.lv1hgyp.mongodb.net/ (Refer fig. 10)
- **Test the Connection:** Run compass application and check, it successfully connects to MongoDB Atlas cluster. (Refer fig. 11)

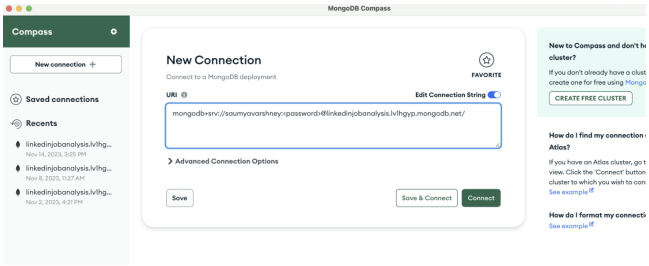

Fig. 10. Prompt for new connection link

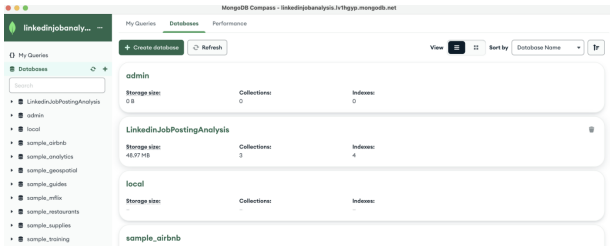

Fig. 11. Successful DB connection

X. Performance Measurement

The performance measurement in MongoDB is quite different compared to the MySQL performance measurements. The MongoDB Compass helps to visualize the processing time of a query to fetch the documents using the option Explain. It also produces a little more vital information about the performance measurements as a Query Performance Summary that includes number of documents returned, documents examined, execution time, sorted in memory and examined index key. While the shell interface provides a comprehensive information about the performance measurements for the provided query by processing every stages of the aggregation pipeline using the query command .explain(“executionStatistics”).

Performance Measurement comparisons: MySQL vs MongoDB queries:

1) **Locations with the Highest Number of Remote Work Opportunities**
db.job_postings.aggregate([$match: remote_allowed: 1 , $group: _id: ”$location”, totalRemoteJobs: $sum: 1, $sort: totalRemoteJobs: -1 , $limit: 5 ,$project: _id: 0, location: ”$_id”, totalRemoteJobs: 1])

Checking performance on MongoDB Compass

Explain Plan : Visual Tree view

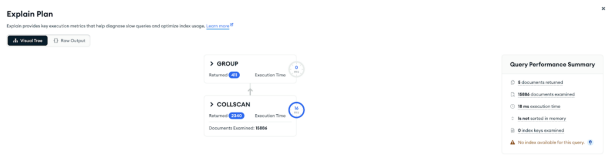

Fig. 12. Visual Tree

Query execution in MySql
select location, sum(remote_allowed) AS TotalRemoteJobs from job_postings where remote_allowed = 1 GROUP by location ORDER BY 2 desc, location desc limit 5

Fig. 13. SQL Query performance

2) **Top 5 Locations with the Sponsored Job Postings**
db.Job_postings.aggregate([$match: sponsored: 1 ,$group: _id: ”$location”, total_sponsored_jobs:$sum: 1 ,$sort: total_sponsored_jobs: -1 ,$limit: 5 , $project:_id: 0, location: ”$_id”, total_sponsored_jobs: 1])

Checking performance on MongoDB Compass
Explain Plan : Visual Tree view

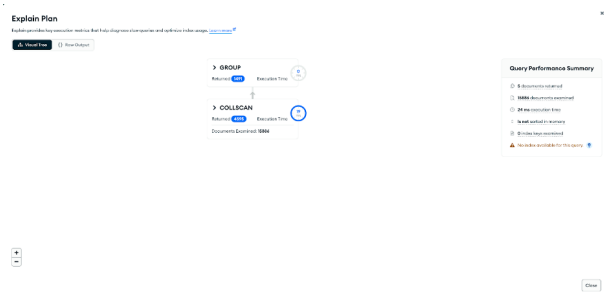

Fig. 14. Visual Tree

Query execution in MySql
select location, sum(sponsored) AS TotalSponsoredJobs from job_postings where sponsored = 1 GROUP by location ORDER BY 2 desc, location desc limit 5

Fig. 15. SQL Query performance

3) **Company with highest employee count where specialties is “analytics”.**
db.Companies.aggregate([$match: ”specialities”: ”analytics” ,$sort: ”employee_counts”: -1 ,$limit: 1,$project: _id: 0,company_name: ”$name”,company_size: ”$company_size”,employeeCount: ”$employee_counts”])

Checking performance on MongoDB Compass

Explain Plan : Visual Tree view

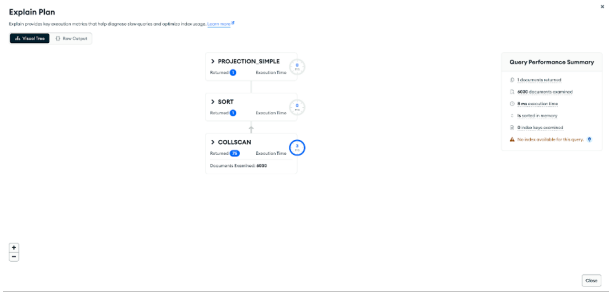

Fig. 16. Visual Tree

Query execution in MySql
SELECT DISTINCT C.name, EC.employee_count AS CompanySize, CS.speciality FROM employee_counts EC JOIN companies C ON EC.company_id = C.company_id JOIN company_specialities CS ON CS.company_id = C.company_id WHERE CS.speciality = ’analytics’

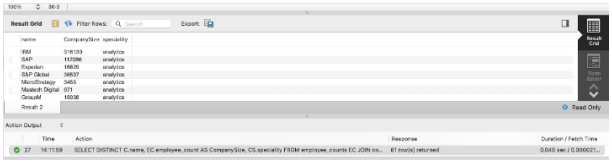

Fig. 17. SQL Query performance

Summary of Performance comparision: MySQL vs MongoDB: Comparison of the performance and speed between MySQL and MongoDB queries are purely based on the query’s requirement and the amount of background process to fetch the contents from the tables/collections respectively. It is highly project centric and the time taken to traverse the volume of data to retrieve the necessary information. Here, the Mongodb

query performances (using MongoDB Atlas Shell and Compass) were much faster in retrieving the results than the MySQL queries (using MySQL workbench). MySQL queries showed slower performance over MongoDB queries to fetch data even though the collscan was internally used by the application.

Query Requirement	MySQL performance	MongoDB performance
Query 1	41 ms	16 ms
Query 2	41 ms	19 ms
Query 3	40 ms	3 ms

Fig. 18. Performance comparison of MongoDB and MySql

XI. GitHub Repository

https://github.com/svarshneysjsu/LinkedIn_JobPostingAnalysis_NoSQL

XII. CONCLUSION

The purpose of the LinkedIn dataset conveyed the essence of a huge business process established between the job seeker and the job recruiter. The dataset helped to come up with various interpretations and drawing possible outcomes along with some limitations. Our functional analysis demonstrated the most accurate rendering of the dataset to a job seeker and a job recruiter.