Department of Applied Data Science

DATA 225

Database Systems for Analytics

Instructor: Simon Shim

LAB GROUP PROJECT REPORT 2

LinkedIn Job Postings

Group 4

Group Members

Anshul Yadav

Sakshi Jain

Shreya Sree Matta

Vatsalya Pooja Chintapalli

Veena Ramesh Beknal

INDEX

PROBLEM STATEMENT:

In the post-pandemic job market, job seekers and recruiters must navigate multiple obstacles. There is a critical need for candidates to understand job trends, skills, and compensation trends to make informed discussions while recruiters need to attract the right talent that fits the company culture with the right skills and right compensation. Various platforms are available for job listings, but LinkedIn is the primary platform for networking and job postings since it has close to 900 million members on the platform and offers a telescopic view of job-related information along with company and individual updates. Using the dataset "LinkedIn Job Posting Analysis" available on Kaggle provides an extensive collection of job listings from the platform for 2023. This dataset contains a nearly comprehensive record of 15,000+ job postings listed over the course of 2 days. Each individual posting contains 27 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. The majority of jobs are also linked to a company, which are all listed in another csv file containing attributes such as the company description, headquarters location, and number of employees, and follower count.

Data from LinkedIn is a rich source for job seekers to identify industry hiring trends and recruiters to find the ideal candidates. Storing all these data points in a structured manner like a database with real-time updates can be invaluable and the potential for exploration of this dataset is vast. This database can be used to identify current open job positions, in-demand skills, compensation trends, companies/industries primed for growth, etc.

Analysis of data from this database combined with other datasets can also offer macro and micro trends of the job market scenario and aid job seekers in improving their respective careers. Connecting the database to a cloud platform, Amazon AWS in this case, ensures reliability, scalability, access-controlled and consistency, and ensures that the access is location-agnostic.

SOLUTION REQUIREMENTS:

We propose using the popular NoSQL database, MongoDB which is an open-source database developed on an architecture that is a horizontal scale-out, and uses a data storage schema that can be customized. Datasets in MongoDB are stored as collections, collections are essentially a group of documents. They are similar to tables in a relational database system, but without any enforcement of a specific kind of document as a schema does. A collection exists within a single database and can have different fields in each document. We will use 2 applications to work with the MongoDB database – Atlas and Compass. Atlas is a cloud database service that focuses on managing the deployment and operation of MongoDB databases along with functionalities like access control, visualization, etc., while Compass is a graphical user interface tool for interacting with MongoDB databases. The database will be created and hosted in a cloud instance of AWS to facilitate querying and analysis. This database facilitates in-depth analysis, ensures data integrity, and empowers job seekers and companies to make better-informed decisions in this competitive job market. We have also explored:

- Key use case queries to show how to navigate the data and answer questions that would typically exist in the mind of an individual entering the job market or

someone looking to make a switch from an existing job
- Visualization/dashboarding of data in Atlas
- Performance comparison between RDBMS (MySQL) and MongoDB

LIMITATIONS:

While this database is robust and can deliver as intended, there are limitations that must be addressed with respect to the data:

- Data integrity: The database is as good as the data it contains and since there is no mandate to update all the data fields, completeness of the data is not in our control. In several cases, the imputation of missing data would be incorrect since said imputation may not reflect the real world (like compensation)
- Normalization/standardization: Since job postings are posted by individuals working for companies or hiring firms who post on behalf of the aforementioned firms, fields like country, city, state of the job may not be standardized (like NY, NYC, New York City for job location)
- Lack of historical data: Since the current data considered is a limited timeframe snapshot, historical trends and comparisons aren't possible – this makes it challenging to understand if a current job market scenario is relatively normal or an outlier

Below are few shortcomings of MongoDB (in some cases, as compared to RDBMS):

- Inability to directly join tables: MongoDB doesn't support joins like a relational database. Although joins between collections can be achieved by manually coding aggregate functions, this may affect performance
- High memory usage: MongoDB stores key names for each value pair resulting in data redundancy which may result in higher memory/storage
- Limited Nesting: Nesting of documents > 100 levels is not possible

DENORMALIZATION:

One of the advantages of using MongoDB over a relational database like MySQL is that unstructured data can be stored in MongoDB. This along with the scalability facilitated by the embedded document structure of data and scale-out architecture results in low effort to set up, unlike MySQL which requires thorough schema design with primary keys, foreign keys, constraints and so on. This advantage of MongoDB allows us to design denormalized databases. Denormalization is based on the simple principle of "*Data that is accessed together should be stored together*". Denormalized databases can improve read performance and query performance in a variety of cases, such as:

- A recurring query requires a few fields from a large document in another collection. We can choose to maintain a copy of those fields in an embedded document in the collection that the recurring query targets to avoid merging two distinct collections or performing frequent $lookup operations
- An average value of some field in a collection is frequently requested. We can choose to create a derived field in a separate collection that is updated as part of your writes and maintains a running average for that field

While embedding documents or arrays without data duplication is preferred for grouping related data, denormalization can improve read performance when separate collections must be maintained. To summarize, denormalization makes sense when we have a high read-to-write ratio [Link to official guide].

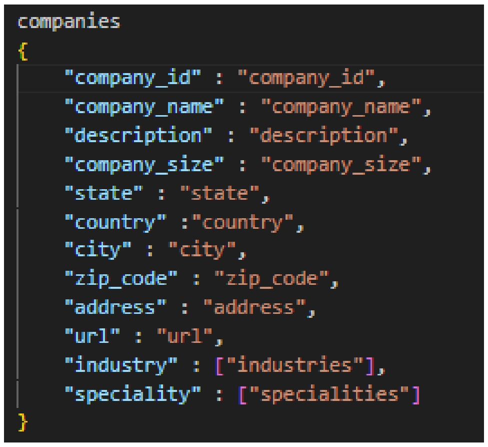
In the case of our database, unlike our previous proposed architecture in Lab 1 wherein we had designed an elaborate ER diagram for a relational database with detailed relationships between the entities (i.e. tables), in the MongoDB database case, we can use the principle of denormalization to consolidate multiple tables into 3 collections:

- Companies collection –
    - Combination of the companies, company_industries and company_specialties datasets, all joined together by company_id
    - Industries and specialties are embedded into arrays to reduce redundancy
    - Snapshots of Python code used to create the underlying dataframe for companies collection below:
- Job postings –
    - This collection combines the job_postings, job skills, job_benefits and job_indsutries datasets all joined together by job_id
    - Skills, benefits and industries are embedded into arrays to reduce redundancy
- Employee counts –
    - This dataset will remain as is since it contains date recorded
    - The only change is that we're creating 2 additional date & time columns, extracted from time_recorded column to make it easier to query

These dataframes once created can be exported into MongoDB collections with conceptual database design as shown in the next section.

CONCEPTUAL DATABASE DESIGN:

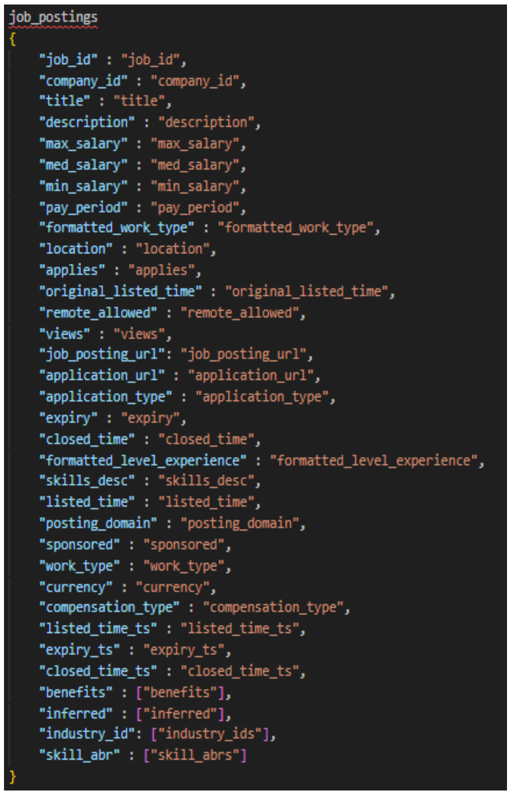
1. **Companies**: This collection stores all the documents about the companies that list jobs on LinkedIn and is unique at company_id key level. This collection is a combination of the companies, company_industries and company_specialties datasets. A snapshot of the collection is shown below:



```
companies
{
    "company_id" : "company_id",
    "company_name" : "company_name",
    "description" : "description",
    "company_size" : "company_size",
    "state" : "state",
    "country" :"country",
    "city" : "city",
    "zip_code" : "zip_code",
    "address" : "address",
    "url" : "url",
    "industry" : ["industries"],
    "speciality" : ["specialities"]
}
```

Figure 1 Companies Schema

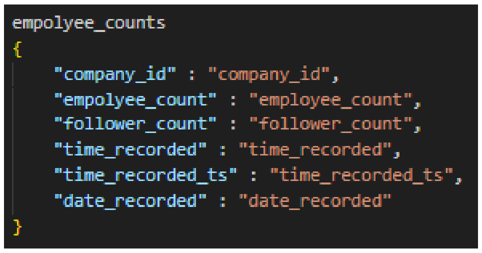
2. **Job Postings:** This is the master collection of job postings containing multiple columns pertaining to the posted job. This collection combines the job_postings, job skills, job_benefits and job_indsutries datasets and is unique at job_id level. A snapshot of the collection is shown below:



```
job_postings
{
    "job_id" : "job_id",
    "company_id" : "company_id",
    "title" : "title",
    "description" : "description",
    "max_salary" : "max_salary",
    "med_salary" : "med_salary",
    "min_salary" : "min_salary",
    "pay_period" : "pay_period",
    "formatted_work_type" : "formatted_work_type",
    "location" : "location",
    "applies" : "applies",
    "original_listed_time" : "original_listed_time",
    "remote_allowed" : "remote_allowed",
    "views" : "views",
    "job_posting_url": "job_posting_url",
    "application_url" : "application_url",
    "application_type" : "application_type",
    "expiry" : "expiry",
    "closed_time" : "closed_time",
    "formatted_level_experience" : "formatted_level_experience",
    "skills_desc" : "skills_desc",
    "listed_time" : "listed_time",
    "posting_domain" : "posting_domain",
    "sponsored" : "sponsored",
    "work_type" : "work_type",
    "currency" : "currency",
    "compensation_type" : "compensation_type",
    "listed_time_ts" : "listed_time_ts",
    "expiry_ts" : "expiry_ts",
    "closed_time_ts" : "closed_time_ts",
    "benefits" : ["benefits"],
    "inferred" : ["inferred"],
    "industry_id": ["industry_ids"],
    "skill_abr" : ["skill_abrs"]
}
```

Figure 2 Job posting Schema

3. **Employee Counts**: This collection stores the details of employee count and followers over different timeframes. Since this collection contains a timestamp of when the headcount was recorded, a single company can have multiple employee_count and follower_count values.



```
empolyee_counts
{
    "company_id" : "company_id",
    "empolyee_count" : "employee_count",
    "follower_count" : "follower_count",
    "time_recorded" : "time_recorded",
    "time_recorded_ts" : "time_recorded_ts",
    "date_recorded" : "date_recorded"
}
```

Figure 3 Employee Count Schema

SCHEMA DESIGN:

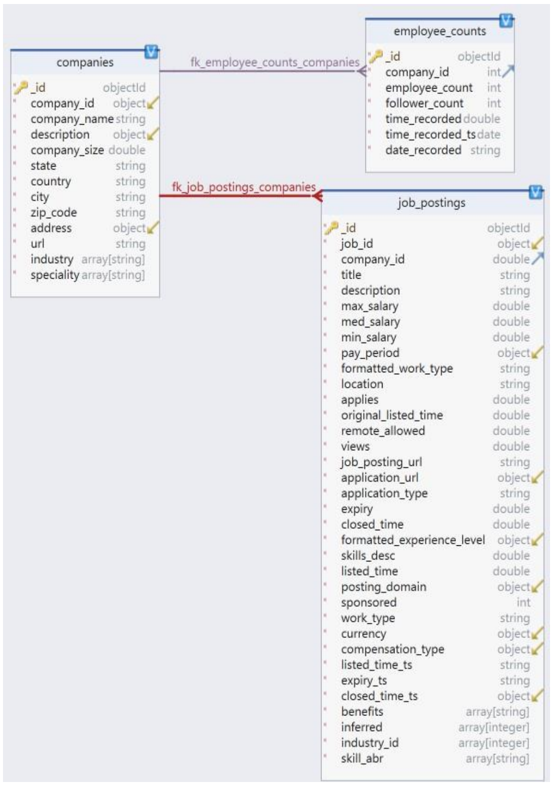
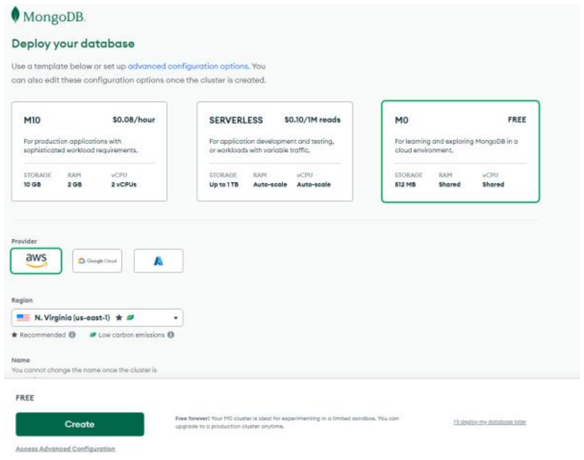


Figure 4 Schema design diagram

CONNECTIVITY TO MONGODB IN CLOUD:

Post denormalization as explained in the above steps, we can create a cloud database instance which will contain the 3 collections. The cloud instance can be accessed through the Atlas interface (https://cloud.mongodb.com/). For this project, we are using the AWS provider's free tier as shown the screenshot below:



MongoDB

Deploy your database

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

M10	$0.08/hour	For production applications with sophisticated workload requirements.	STORAGE 10 GB	RAM 2 GB	vCPU 2 vCPUs
SERVERLESS	$0.10/1M reads	For application development and testing, or workloads with variable traffic.	STORAGE Up to 1 TB	RAM Auto-scale	vCPU Auto-scale
M0	FREE	For learning and exploring MongoDB in a cloud environment.	STORAGE 512 MB	RAM Shared	vCPU Shared

Provider: aws

Region: N. Virginia (us-east-1)

★ Recommended · Low carbon emissions

Name: You cannot change the name once the cluster is

FREE

Create

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

I'll deploy my database later

Access Advanced Configuration

Figure 5 Database in Mongo DB

Once we have set up credentials, we can deploy a cluster with the required configuration. Our cluster named **LinkedInjobpostingsDBcluster** can be accessed using the following connection string:

mongodb+srv://veenabeknal:<password>@linkedinjobpostingsdbcl.59k7czo.mongodb.net/

Please note that password has been masked for security reasons in the above connection string.

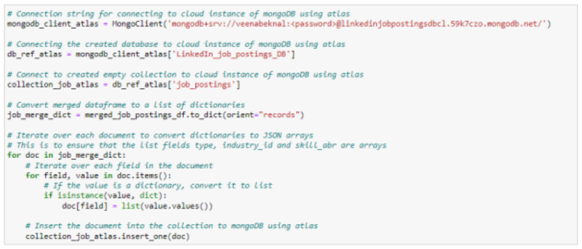
Snapshot of our cluster with the 3 collections is shown below:

LinkedInjobpostingsDBCluster
LinkedIn_job_postings_DB.companies
companies
job_postings

Figure 6 LinkedIn Database

There are 3 ways to input data into this cloud instance once the empty collections are created:

- Write out the 3 dataframes as csv files and input into the appropriate MongoDB collections using the Compass tool
- Same as step 1 except with json files
- Using the pymongo python library, we can directly import our denormalized dataframes into the cloud instance using the connection string; code used for job postings collection is shown below:



```python
# Connection string for connecting to cloud instance of mongoDB using atlas
mongodb_client_atlas = MongoClient('mongodb+srv://veenabeknal:<password>@linkedinjobpostingsdbcl.59k7czo.mongodb.net/')

# Connecting the created database to cloud instance of mongoDB using atlas
db_ref_atlas = mongodb_client_atlas['LinkedIn_job_postings_DB']

# Connect to created empty collection to cloud instance of mongoDB using atlas
collection_job_atlas = db_ref_atlas['job_postings']

# Convert merged dataframe to a list of dictionaries
job_merge_dict = merged_job_postings_df.to_dict(orient="records")

# Iterate over each document to convert dictionaries to JSON arrays
# This is to ensure that the list fields type, industry_id and skill_abr are arrays
for doc in job_merge_dict:
    # Iterate over each field in the document
    for field, value in doc.items():
        # If the value is a dictionary, convert it to list
        if isinstance(value, dict):
            doc[field] = list(value.values())

    # Insert the document into the collection to mongoDB using atlas
    collection_job_atlas.insert_one(doc)
```

Figure 7 Python snippet to connect to MongoDB

FUNCTIONAL ANALYSIS:

From the perspective of a job seeker, we can use the various entities detailed earlier, link them through queries and arrive at interesting data-driven insights to enhance one's professional career. For candidates, the key criteria while looking for a job are (in no particular order of priority) – company prospects, growth prospects, benefits, compensation, etc. We have listed multiple MQL query snippets that can help a candidate evaluate the job market and take an informed decision. Below are some use cases for candidates to use this dataset and derive value:

1. Finding companies that don't offer 401(k) as a benefit
2. Finding jobs in the state of California with max salary < $200K and min salary > 125K
3. Identifying the companies in New York state that fall under IT industry and speciality as cloud/security/analytics?
4. Finding the top 5 paying jobs by max salary across all jobs
5. Identifying companies that have the keywords "artificial intelligence" in their description
6. Find the companies that don't offer 401(k) as a benefit.

7. Find jobs in California where the max salary is greater than 200K and min salary is greater than 125K
8. Companies with the most common job skills
9. List all companies in a specific city (ex: New York) and their average employee counts
10. Identifying companies with the highest average maximum salaries for job postings.
11. Which companies have the highest number of specialties by industry?

MQL CODE SNIPPETS/QUERIES:

1. db.job_postings.find({ title: /analyst/i, formatted_work_type: /fulltime/i, location: /San jose/i});

2. db.employee_counts.aggregate([{ $sort: { time_recorded: -1 } }, { $group: { _id: "$company_id", follower_count: { $first: "$follower_count" } } }, { $sort: { follower_count: -1 } }, { $limit: 5 }]);

3. db.job_postings.aggregate([{$lookup: {from: "companies",localField: "company_id",foreignField: "company_id",as: "company"}},{$unwind: "$company"},{$match: {$and: [{ "company.state": /New York/i },{$or: [{ "company.speciality": /data analytics/i }, { "company.speciality": /big data/i }]}]}},{$sort: {med_salary: -1}}]);

4. db.job_postings.aggregate([{$group: {_id: "$title",salaryVariability: {$stdDevSamp: { $subtract: ["$max_salary", "$min_salary"]}}}},{ $sort: { salaryVariability: -1 } },{ $limit: 5 }]);

5. db.job_postings.aggregate([{ $match: { sponsored: 1, title: "Data Analyst" } }, { $group: { _id: "$location", count: { $sum: 1 } } }]);

6. db.job_postings.aggregate([{$match: {benefits: {$ne: "401k"}}}, {$project: {company_id: 1, title: 1, benefits: 1}}]);

7. db.job_postings.aggregate([{$match: {location: {$regex: "CA"}, max_salary: {$lt: 200000}, min_salary: {$gt: 125000}}}, {$project: {_id: 0, job_id: 1, company_id: 1, max_salary: 1, min_salary: 1}}]);

8. db.job_postings.aggregate([{ $unwind: "$skill_abr" }, { $group: { _id: "$skill_abr", count: { $sum: 1 } } }, { $sort: { count: -1 } }, { $limit: 10 }]);

9. db.getCollection('companies').aggregate([{$lookup: {from: 'employee_counts', localField: 'company_id', foreignField: 'company_id', as: 'employee_counts'}}, {$unwind: {path: '$employee_counts', includeArrayIndex: 'string', preserveNullAndEmptyArrays: true}}, {$group: {_id: {company_name: '$company_name', city: '$city'}, avg_employee_count: {$avg: '$employee_counts.employee_count'}}} , {$match: {'_id.city': 'New York'}}, {$project: {_id: 0, company_name: '$_id.company_name', city: '$_id.city', avg_employee_count: 1}}]);

10. db.job_postings.aggregate([{$lookup: {from: "companies",localField: "company_id", foreignField: "company_id",as: "company_info"}},{$group: {_id:

"$company_id",avgMaxSalary: {$avg: {$toDouble: {$ifNull: ["$max_salary", 0]}}},company_info: {$push: "$company_info"}}},{$sort: {avgMaxSalary: -1}}]);

11. db.companies.aggregate([{$unwind:"$speciality"},{$group:{_id:{company_id:"$company_id",industry:"$industry",company_name:"$company_name"},specialtiesCount:{$sum:1}}},{$sort:{"_id.industry":1,specialtiesCount:-1}},{$group:{_id:"$_id.industry",topCompany:{$first:{company_id:"$_id.company_id",company_name:"$_id.company_name",specialtiesCount:"$specialtiesCount"}}}},{$project:{_id:1,topCompany:1}}]);

VISUALIZATION:

Figure 8 shows the overall visualization of the LinkedIn job postings data.

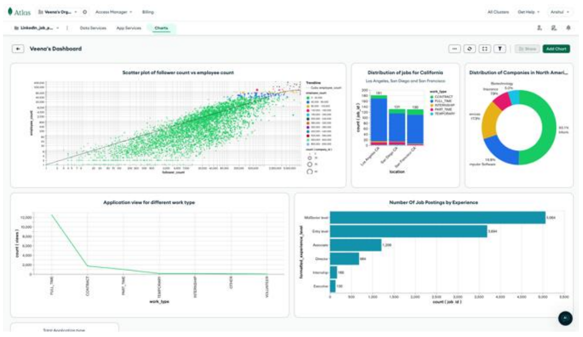


Figure 8 Visualization using Atlas

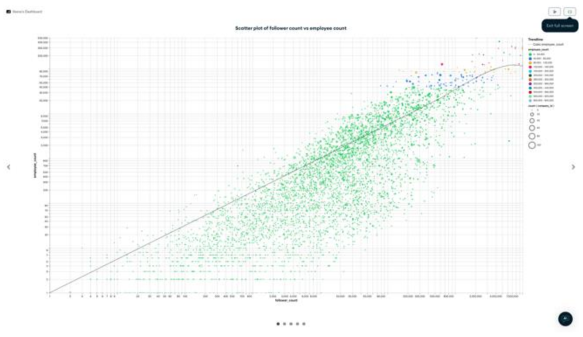


Figure 9 Scatter plot of follower count vs employee count

The scatter plot in Figure 9 shows a clear linear relationship between the number of LinkedIn followers and the number of employees for various companies. Most data points lie below the line, indicating that companies generally have fewer employees relative to their follower count. This indicates that having a large online following is not necessarily correlated with having a large workforce. Additionally, the plot shows a saturation effect, where the growth in employees diminishes as follower count increases.

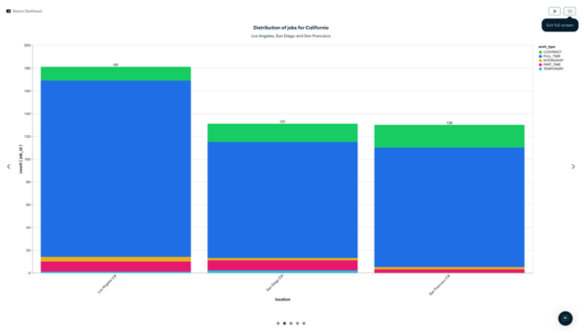


Figure 10 Distribution of jobs in California cities

This bar chart that provides a comprehensive view of the job market in Los Angeles, San Diego, and San Francisco. The chart reveals the distribution of job types in each city. In Los Angeles, full-time key positions dominate, accounting for 85.6% of jobs, followed by part-time roles (4.97%), contract jobs (6.63%), internships (2.21%), and temporary positions (0.55%). San Diego shows a similar trend albeit with a slightly higher share of contract roles (12.21%) and part-time jobs (6.87%). San Francisco, however, has a different job market structure, with contract jobs being the highest of the 3 cities (15.38%) while full-time positions form the majority (80.77%). The findings of this analysis can be valuable for job seekers and businesses navigating these cities' job markets.

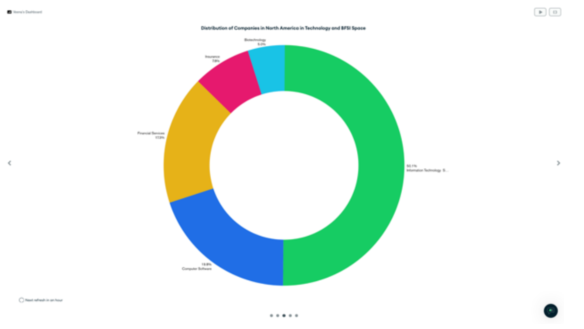


Figure 11 Distribution of Companies in North America in Technology and BFSI Space

The job market in North America is constantly evolving, with the technology and BSFI sectors leading the way. A graph depicting the distribution of companies in these sectors reveals the dominance of IT services, accounting for half of the job distribution. Software development roles are also prominent, representing 19.8%. The financial services industry plays a critical role, making up 17.3% of the job distribution. Additionally, the insurance sector (7.8%) and biotechnology (5%) contribute form the minority but have a meaningful share.

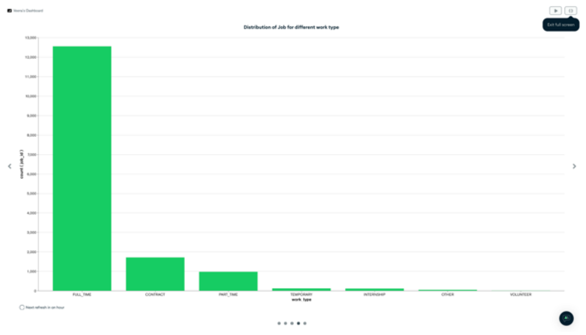


Figure 12 Distribution of jobs across different work types

Analyzing job applications can offer insights into candidate preferences and engagement levels. A graph shows that full-time positions are the most popular, with over 12,000, while contract roles have around 1,700. Part-time and internship positions have fewer than 1,000. These trends suggest a strong inclination towards full-time roles. Employers and recruiters can use this information to augment recruitment efforts accordingly to attract top talent.

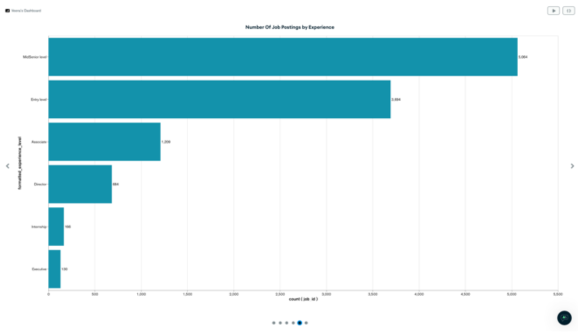
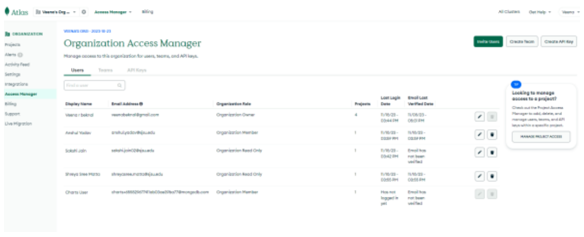


Figure 13 Number of Job Postings by Experience

The chart reveals that mid-senior level positions have the highest count at 5,064, indicating a strong demand for experienced professionals. Entry-level positions closely follow with 3,694 job postings, a good sign for those starting their careers. Associate roles cater to those with intermediate experience levels, while director-level positions demonstrate a demand for seasoned professionals in leadership roles. Internships offer skill development and entry into the workforce, while executive positions represent a niche segment with specialized requirements. This analysis helps inform candidates across all experience levels about the prevalence of opportunities across the economy.

ACCESS PRIVILEGES:

Read/Write access to tables and dashboard can be controlled from the Access Manager in Atlas as shown in the snapshot below:



VEENA'S ORG - 2023-11-23

Organization Access Manager

Manage access to this organization for users, teams, and API keys.

Display Name	Email Address	Organization Role	Projects	Last Login Date	Email Last Verified Date
Veena r bekmal	veenabekmal@gmail.com	Organization Owner	4	11/16/23 - 03:44 PM	11/08/23 - 05:51 PM
Anshul Yadav	anshul.yadav@sjsu.edu	Organization Member	1	11/16/23 - 03:39 PM	11/16/23 - 03:39 PM
Sakshi Jain	sakshi.jain02@sjsu.edu	Organization Read Only	1	11/16/23 - 03:42 PM	Email has not been verified
Shreya Sree Marta	shreyasree.marta@sjsu.edu	Organization Read Only	1	11/16/23 - 03:55 PM	11/16/23 - 03:55 PM
Charts User	charts+[illegible]@mongodb.com	Organization Member	1	Has not logged in yet	Email has not been verified

Looking to manage access to a project?

Check out the Project Access Manager to add, delete, and manage users, teams, and API keys within a specific project.

Figure 14 Access Privileges

PERFORMANCE MEASUREMENT:

	Functional Requirements	Execution Time (milliseconds)
1	List of jobs that have ‘analyst’ in the title are full-time and based in San Jose, CA	198
2	Companies with the highest increase in follower count (top 5)	40
3	List of jobs by companies that have ‘big data’ or ‘data analytics’ as a specialty in NY state ordered by descending median salary	21
4	Highest salary ranges for different roles in (one particular industry)	6
5	Count of sponsored "Data Analyst" job postings for each location	65
6	Find the companies that don’t offer 401(k) as a benefit.	254
7	Find jobs in California where the max salary is greater than 200K and min salary is greater than 125K	187
8	Companies with most common job skills	70
9	List all companies in a specific city (ex: New York) and their average employee counts	36 (MySQL Query Performance: 180)
10	Identifying companies with the highest average maximum salaries for job postings.	45
11	Which companies have the highest number of specialties by industry?	41 (MySQL Query Performance: 1047)

Queries 9 and query 11 perform better in Mongo DB compared to MySQL.

GITHUB REPO:

LinkedIn Project.