# Linkedin Job Posting Data Analysis using NoSQL

Aryama Ray
Ananya Varma Mudunuri
Vaishnavi Mocherla
Sai Sahithi Bethapudi
Vijay Rama Raju Penmetsa

***Abstract*—**

**LinkedIn is being one of the most popular platforms for job searching, posting, making professional connections for past few years. The objective for our application is to build NoSQL database architecture and perform analysis on that data which can be helpful for job seekers, potential employers and stake holders in the firms to make decision.**

## I. PROBLEM STATEMENT

With respect to the latest job market, it is always beneficial to have a platform which is helpful to analyze the latest trends, patterns and insights in the job market. This project provides all kinds of analyzing and create visualizations which are going to create a seamless experience to the users to get a clear idea on the trends in the job market. There are three kinds of users on the platform, Type 1: job seekers (who just view the analysis) Type 2: the researchers (who perform the analysis on the data) Type 3: the linked admin (who will scrap data from LinkedIn).

## II. SOLUTION REQUIREMENT

The approach of this project is that it provides this platform where all kinds of analyzing are readily available.

The LinkedIn admin is going to be responsible for entering the data into the NoSQL. He is also in charge of updating the new job postings. The researchers on the other hand are the one performing various analysis and visualizations with respect to the data provided by the LinkedIn Admin. Now, the users, that is the job seekers are the one that are going to be viewing these insights created by the researchers. Moreover, other than managing the data, the admin of LinkedIn will also be able to observe the patterns of the job postings thought this application as well.

We are going to use MongoDB for the insertion and analyzing the data. The input data, that is the data that is web scraped by the LinkedIn admin. The researchers, using the MongoDB are going to retrieve various analytics from the data and additionally create dashboards to visualize this analysis. And finally, the users are going to be the one viewing these various kinds of insights to find out the trends and the patterns in the job market However, this does not mean that only users can view this portal, all kinds of users, including LinkedIn admin will be able to view this portal. The job seekers are going to be in the reading mode whereas the researchers and the admin are going to be in both reading and writing mode.

The data web scraped is going to play a huge part in the making of this project. This data is initially preprocessed by dropping a few columns, filling null values with mean or mode, having a uniform datatype, and so on to fit the requirements of the analysis. Since we are going to be working on the job postings from the United States, we have normalized the state names that were inconsistent into one format. Since we are going to use MongoDB which by default creates id for each object, we have removed the id column for the tables. This data is then entered in an embedded format where the company details are going to contain all the information about the company and the benefit values can be a list of arrays since one single job can have number of benefits.

The limitation of this specific approach is that it is a complex process to insert the LinkedIn data since the schema is complex. The analysis is mainly for the jobs in the USA since the dataset is mainly focused with the job postings in USA. In addition, this project does not let users interact but only just lets them view the analysis, i.e., it is not interactive with the job seekers.

## III. CONCEPTUAL DATABASE DESIGN

The seven datasets can be divided into three categories of dataset:

1: companies.csv

company_industries.csv
company_specialities.csv
employee_counts.csv

This category contains all the details related to the company. This includes its name, company id, specialties, which industry it is from, the number of employees and so on. All the details with respect to the companies can be found here in this category.

2: job_details:

benefits.csv
job_industries.csv
job_skills.csv

Similarly like the company details category, this category also represents all the details of the job. These details include the required skills, which it is from, salary details, work period, which industry is this job associated with and so on.

3: job_postings:

job_postings.csv

The job postings basically contain information with request to the postings like the URL link of the post, which company is the job posting is for, views, applies, etc.

However, for NoSQL database we will be designing denormalised tables/Collections. NoSQL databases are more flexible in terms of data representation and storage.So these catagories can be made into two other collections:

CompanyJobDetails:This collection is for all the job requirements from different companies.

- Document Structure of this collection is designed from JOB table schema created in Lab1 along with Company table schema.

- Company table schema is inserted as Embedded document under Company details.

- This embedded document also has company industry and specialty added as a list element to reduce data redundancy.

- Also, this document structure has Jobskills added as a list element to reduce load of data redundancy.

- Entire collection is limited to these tables to optimize query performance and data load.

Here is a snippet of CompanyJobDetails JSON Schema structure for one document.


```
_id: ObjectId('6555245df157517edb9b3fbc')
job_title: "Civil Engineer"
company: Object
  company_name: "Eric L. Davis Engineering Inc."
  description: "Eric L. Davis Engineering Inc. headquartered in the Dallas-Fort Worth …"
  company_size: 2
  state: "TX"
  country: "US"
  city: "Forney"
  zip_code: "75087"
  url: "https://www.linkedin.com/company/eric-l.-davis-engineering"
  industry: Array (1)
    0: "Architecture & Planning"
  speciality: Array (4)
    0: "Residential Foundation Design"
    1: "Residential Frame Design"
    2: "Lateral/Shear Design"
    3: "Residential Inspection Services"
max_salary: 82000
min_salary: 60000
pay_period: "YEARLY"
formatted_work_type: "Full-time"
job_location: "Forney, TX"
application_type: "ComplexOnsiteApply"
formatted_experience_level: "Mid-Senior level"
sponsored: 1
work_type: "FULL_TIME"
skill_abr: Array (2)
  0: "ENG"
  1: "IT"
```


JobPostFromCompany : This collection is for all the job post details posted on LinkedIn Website from different companies.

- Document Structure of this collection is designed from JOBPOST table schema created in Lab1.

- Company name and id from Company table schema is inserted as Embedded document under Company details.

- This embedded document also has employee count and follower count from EMP_CNT table schema added from Lab1.

- Also, this document structure has Benefits added as a list element to reduce load of data redundancy.

- Entire collection is limited to these tables to optimize query performance and data load.

Here is a snippet of CompanyJobDetails JSON Schema structure for one document.

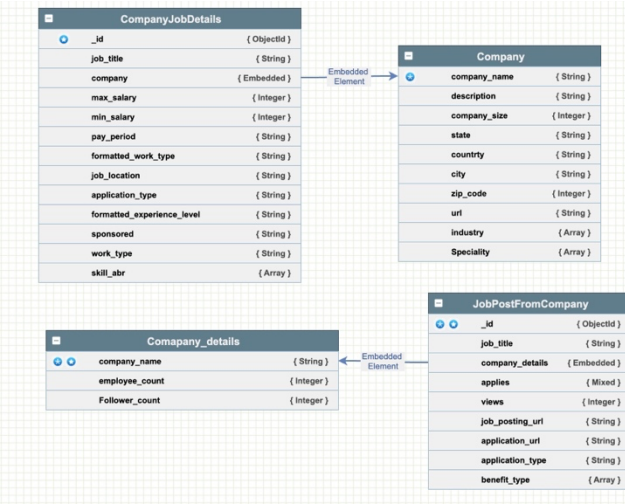

```
_id: ObjectId('65557e9ff157517edb9b7c57')
job_title: "Sales Manager"
company_details: Object
  company_name: "CargoLogin."
  employee_count: 15
  follower_count: 159
applies: 6
views: 25
job_posting_url: "https://www.linkedin.com/jobs/view/133114754/?trk=jobs_biz_prem_srch"
application_url: "unknown"
application_type: "ComplexOnsiteApply"
benefit_type: Array (3)
  0: "Medical insurance"
  1: "401(k)"
  2: "Vision insurance"
```

Conversion of the data into JSON and schema building for NoSQL database:

To create Collections in MongoDB database, we converted our csv data into JSON format through pandas Dataframe and developed the schema structure using Jupyter notebook.

Then Clusters and Database are created in MongoDB. And uploaded JSON data into two collections:

- CompanyJobDetail: This collection has 15520 documents.
- JobPostFromCompany: This collection has 15470 documents.




IV. FUNCTIONAL ANALYSIS

Functional analysis is used to identify the functional elements of the project.

Here in this project

a. **Job Seeker**: A job seeker can have multiple requirements from the system.

a job seeker can search for:

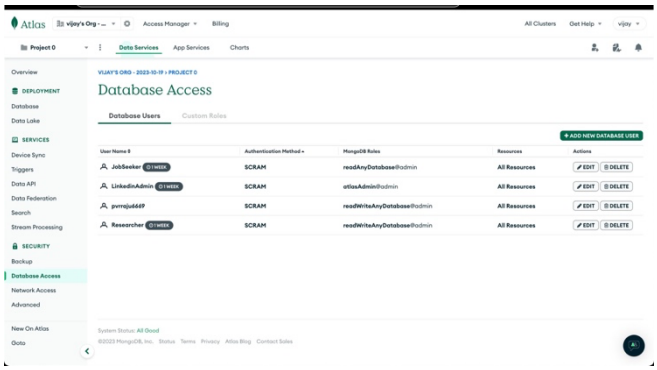
i. a job on the platform based on skills, experience level, job title, job location, salary etc.

ii. a company on the platform based on company name, company size, benefits etc.

iii. if the job post has application link posted on the platform job seeker can apply.

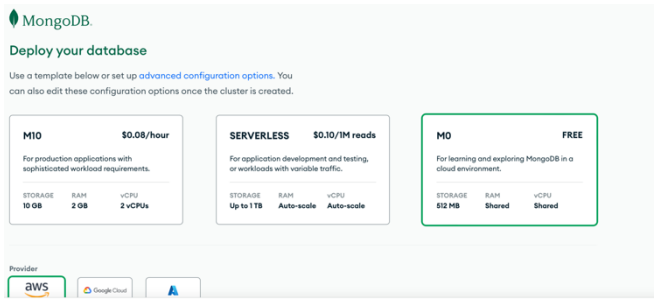
b. **LinkedIn Admin:** This admin is responsible for adding the data into the database and also get an picture on the patterns in the postings

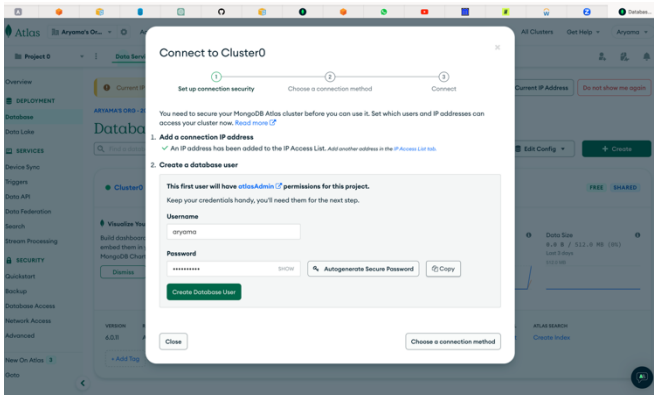
c. **Researchers:** A researcher helps with creating various analysis based on the data in the database. Additionally, they even create data visualization dashboards.

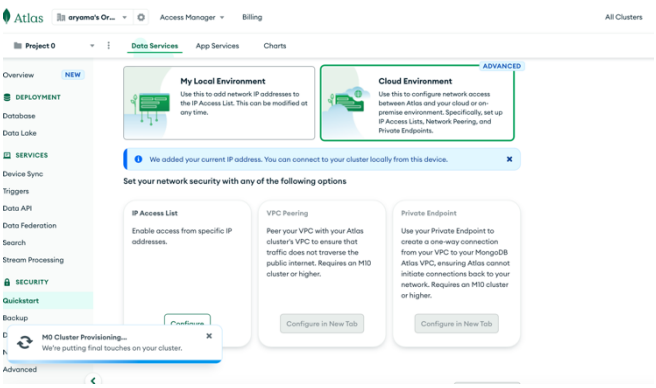



V. DATABASE CONNECTION

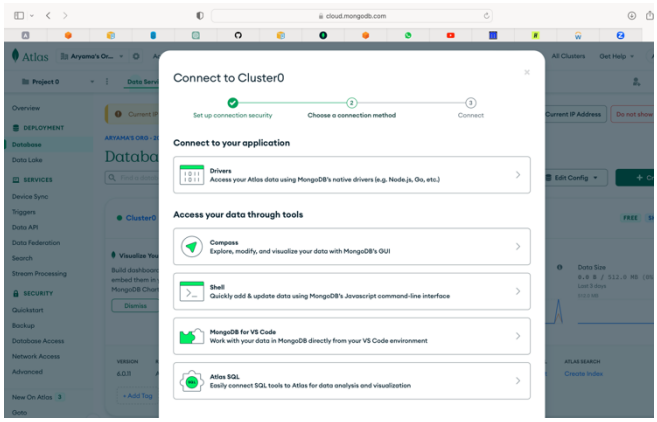
MongoDB cluster creation in cloud environment



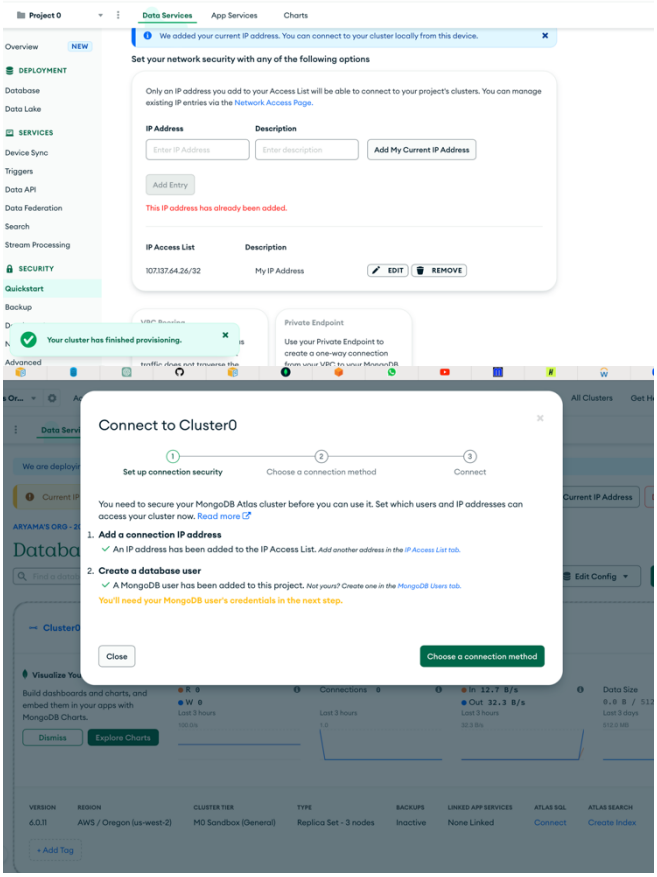





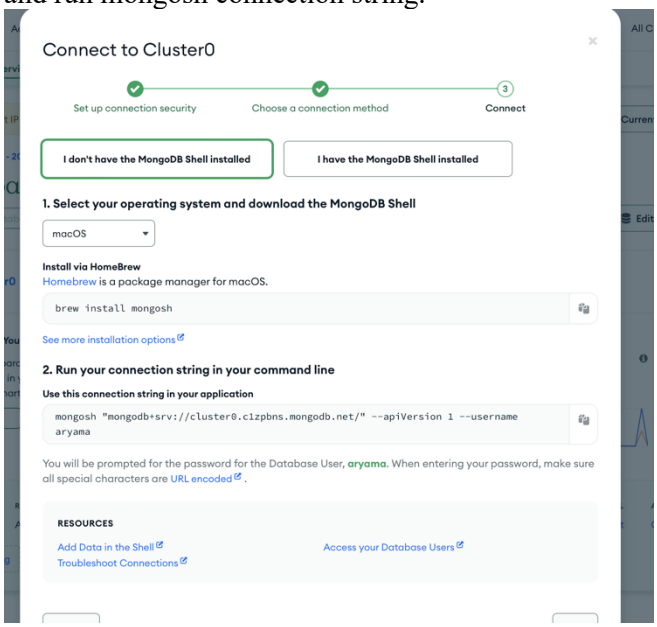

Cluster created using Cloud Environment Data Services.




Using MongoDB shell. We will be installing mongosh and run mongosh connection string.

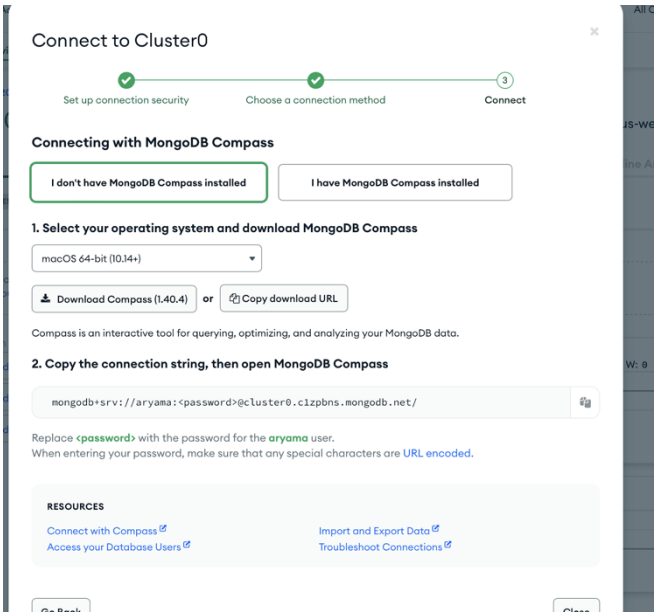



Connecting the database with MongoDB shell and MongoDB compass




Creating first database

Also, we can connect Cluster using connection string in MongoDB compass:




```
from pymongo import MongoClient
client = MongoClient(f" your connection string ")
db = client["Lab_2"]
collection = db['Company_T']
results = collection.find().limit(10)
print(results)
```

This Python connects to a MongoDB database using the pymongo module. It establishes a connection to the Lab_2 database and pulls the first ten documents from the Company_T collection. The snippet of code retrieves the documents using the find () function and sets a limit on the amount of documents returned using the limit() method. Lastly, the results are printed using the print () function.

## VI. QUERY AND ANALYSIS

1: Finding the average salary for each combination of formatted_experience_level and work_type.

```
db.LinkedinJobAnalysisData.aggregate([
  {
    $group: {
      _id: {
        experience_level: "$formatted_experience_level",
        work_type: "$work_type"
      },
      avg_salary: {$avg: {$add: ["$max_salary",
"$min_salary"] } }
    }
  },
{
$sort: {avg_salary: -1}
}
]);
```

This query can be used to find the average salary for each combination of experience level and work type. From this query we can say that experience level Director seems to be having the highest average salary with work type as other, followed by executive experience level.

2: the most common specialty and which companies have these speciality.

```
db.LinkedinJobAnalysisData.aggregate([
{
   $unwind: "$company.speciality"
  },
  {
   $group: {
     _id: "$company.speciality",
     companyCount: { $sum: 1 },
     companies: {$addToSet:
"$company.company_name" }
   }
  },
  {
   $sort: { companyCount: -1 }
  },
  {
   $limit: 1
  },
  {
   $project: {
     _id: 0,
     mostCommonSpeciality: "$_id",
     companyCount: 1,
     companies: 1
   }
  }
]);
```

The above query can be used to find which specialty is most in demand and what all companies have this speciality.From the output , community is the most common speciality with companies like Lyft ,The Moms project, etc. having this specialty

3:10 Companies with the Most Job Listings.

```
db.LinkedinJobAnalysisData.aggregate([
 {
 $group: {
 _id: "$company_details.company_name",
 jobCount: { $sum: 1 }
 }
```

```
},
{
$sort: { jobCount: -1 }
},
{
$limit: 10
},
{
$project: {
_id: 0, // Exclude the default _id field
company_name: "$_id",
jobCount: 1
}
}
]);
```

The company that has the greatest number of postings are from the company 'null'. From this we can tell that there are lot of job postings where this company name is not mentioned. From the second company in the output is City Lifestyle which a total posting of 161 offers.

4: Ratio of the employee count to the applications.

```
db.LinkedinJobAnalysisData.aggregate([
  {
    $match: {
      $and: [
        { "company_details.employee_count": { $ne: null } },
        { "applies": { $ne: null } }
      ]
    }
  },
  {
    $group: {
      _id: "$company_details.company_name",
      total_employee_count: { $first: "$company_details.employee_count" },
      total_applications: { $sum: "$applies" }
    }
  },
  {
    $project: {
      _id: 0,
      company_name: "$_id",
      employee_to_application_ratio: { $divide: ["$total_employee_count", "$total_applications"] }
    }
  },
  {
    $sort: { employee_to_application_ratio: -1 }
  }
]);
```

This calculates the ratio of the employee count with the applications, here the company PWC has the highest ration with 273293, followed by Ericsson, McDonald and Starbucks. However, it is important to keep in mind that if there are lot of job postings this subsequently leads to a lot of job applications.

5:Job Postings with High Application-to-View Ratio.

```
db.LinkedinJobAnalysisData.aggregate([
  {
    $match: {
      "views": { $gt: 0 },
      "applies": { $gt: 0 },
      $expr: { $gt: ["$views", "$applies"] }
    }
  },
  {
    $project: {
      _id: 0,
      job_posting_url: 1,
      company_name: "$company_details.company_name",
      views: 1,
      applies: 1,
      application_to_view_ratio: { $divide: ["$applies", "$views"] }
    }
  },
  {
    $sort: { application_to_view_ratio: -1 }
  }
]);
```

Similarly, like the above query this find outs the ration of Application to view of the job postings. Here we can understand the relationship of the views and the applies for that particular job posting.

6. Which state generates more jobs?

```
db.CompanyJobDetail.aggregate([{
  $unwind: "$company"
}, {
  $group: {
    _id: "$company.state",
jobcount: {
```

```
    $count: {}
   }
  }
}]).sort({
 jobcount: -1
})
```

The Linkedin Data set shows California alone has the highest job requirement in the USA, and this requirement is greater than the combined requirement from Texas and New York, which are from the East coast.

7. Let us check which Job title has highest job requirement in California

```
db.CompanyJobDetail.aggregate([{
 $match: {
  "job_location": {
   $regex: 'CA'
  }
 }
}, {
 "$group": {
  _id: "$job_title",
  "jobcount": {
   $count: {}
  }
 }
}, {
 "$project": {
  "_id": 1,
  "jobcount": 1
 }
}]).sort({
 jobcount: -1
})
```

So the query result shows Sales Director(owner/operator) title has the highest job requirement in California state followed by Executive Assistant title.

8. Number of jobs posted from companies having highest followers. Are the companies having a high volume of followers generating the required amount of jobs?

```
{
 db.JobPostFromCompany.aggregate([{
  $unwind: "$company_details"
 }, {
  "$group": {
   _id: "$company_details.company_name",
   maxfollower: {
    $max: "$company_details.follower_count"
   }
  }
 }, {
  $sort: {
   "maxfollower": -1
  }
 }, {
  $limit: 5
 }])
}
```

So, Amazon has the highest followers followed by Google and Unilever. We will explore how many jobs were created by these top following companies.

```
db.JobPostFromCompany.aggregate([{
  $unwind: "$company_details"
 }, {
  $match: {
   "company_details.company_name": {
    $in: ["Amazon", "Google", "Unilever", "Apple"]
   }
  }
 }, {
  "$group": {
   _id: "$company_details.company_name",
   "jobcount": {
    $count: {}
   }
  }
 }])[{
   _id: 'Apple',
   jobcount: 15
  }, {
   _id: 'Amazon',
   jobcount: 93
  }, {
   _id: 'Google',
   jobcount: 93
  }, {
   _id: 'Unilever',
   jobcount: 2
  }
```

9. This query calculates the average salary for each experience

```
level.db.LinkedinJobAnalysisData.aggregate([
 {
  $group: {
   _id: "$formatted_experience_level",
   avg_salary: { $avg: { $add: ["$max_salary",
"$min_salary"] } }
```

```
    }
  },
  {
    $project: {
      _id: 0,
      experience_level: "$_id",
      avg_salary: 1
    }
  }
]);
```

This query uses the LinkedinJobAnalysisData collection to calculate the average salary for each experience level. The average salary is determined by taking the mean of the minimum and maximum incomes for each level, and it groups job posts based on formatted_experience_level. Following that, a projection of the results shows the average salary and level of experience. This approach offers insight into the ways in which salary expectations vary among various job market experience levels.

10. This query finds the most common job title

```
db.LinkedinJobAnalysisData.aggregate([
  {
    $group: {
      _id: "$job_title",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { count: -1 }
  },
  {
    $limit: 1
  },
  {
    $project: {
      _id: 0,
      most_common_job: "$_id",
      postings_count: "$count"
    }
  }
]);
```

The above query identifies the most common job title in the `LinkedinJobAnalysisData` collection. It counts the occurrences of each title and groups the data by 'job_title'. This count is then used to arrange the results in decreasing order. In order to highlight the job title that appears most frequently in the dataset, the query restricts the output to the single most frequent job title and provides it along with the total number of postings for that title.

11. Query to Identify Job Titles with the Most Significant Salary Differences:

This query finds job titles with the greatest difference between their maximum and minimum salary offerings. This can highlight roles with highly variable pay scales.

```
db.LinkedinJobAnalysisData.aggregate([
  {
    $project: {
      job_title: 1,
      salary_difference: { $subtract: ["$max_salary", "$min_salary"] }
    }
  },
  {
    $sort: { salary_difference: -1 }
  },
  {
    $limit: 10
  },
  {
    $project: {
      _id: 0,
      job_title: 1,
      salary_difference: 1
    }
  }
]);
```

In order to determine which job titles, have the biggest disparities between their maximum and minimum salary offers, this query examines the 'LinkedinJobAnalysisData' collection. For every job title, the wage differential is computed and sorted in descending order. By limiting the output to the top 10, the query highlights the job titles with the greatest salary variability. This approach helps identify the roles with the greatest salary ranges available in the job market.

12: Calculate the average salary for jobs in each country and work type combination.

```
db.LinkedinJobAnalysis.aggregate([
  {
    $group: {
      _id: { country: "$company_details.country", work_type: "$formatted_work_type" },
```

```
            avg_salary: { $avg: "$max_salary"
}
        }
    },
    {
        $project: {
            country: "$_id.country",
            work_type: "$_id.work_type",
            avg_salary: 1,
            _id: 0
        }
    }
]);
```

This query performs an aggregation operation on the `LinkedinJobAnalysis` collection. It groups the data by `country` and `work_type`, and calculates the average `max_salary` for each group. The result is a list of countries, work types, and their corresponding average salaries.

13: Identify companies with the highest average number of job views for their postings.

```
db.LinkedinJobAnalysis.aggregate([
    {
        $group: {
            _id:
"$company_details.company_name",
            avg_views: { $avg: "$views" }
        }
    },
    {
        $sort: { avg_views: -1 }
    },
    {
        $limit: 5
    }
]);
```

The query groups the data by company_name and calculates the average number of views for each company. The results are then sorted in descending order of avg_views. Finally, it limits the output to the top 5 companies with the highest average views.

14:Find job titles and locations with the highest median salary, ordered by median salary in descending order.

```
db.LinkedinJobAnalysis.aggregate([
    {
        $sort: { "med_salary": -1 }
    },
    {
        $project: {
            job_title: 1,
            location: 1,
            med_salary: 1
        }
    },
    {
        $limit: 10
    }
```

```
]);
```

The query sorts the data in descending order based on the `med_salary` field. The query then projects or selects the `job_title`, `location`, and `med_salary` fields from the sorted data. Finally, it limits the output to the top 10 records with the highest median salaries.

15:List Job Titles with High Views-to-Applications Ratio (Retrieve job titles with a high ratio of views to applications, indicating high interest relative to the number of applications.)

```
db.LinkedinJobAnalysis.aggregate([
  {
    $match: {
      applies: { $gt: 0 }
    }
  },
  {
    $project: {
      job_title: 1,
      views_to_applications_ratio: { $divide: ["$views", "$applies"] }
    }
  },
  {
    $sort: { views_to_applications_ratio: -1 }
  },
  {
    $limit: 10
  }
]);
```

This query first filters the data to include only those records where `applies` is greater than 0. Then, it calculates the ratio of `views` to `applies` for each job and includes the `job_title` in the output. The results are sorted in descending order based on the `views_to_applications_ratio`. Finally, it limits the output to the top 10 records with the highest views-to-applications ratio.

16:Identify Job Titles with the Highest Increase in Applications (Find job titles with the highest percentage increase in the number of applications compared to the previous data collection.)

```
db.LinkedinJobAnalysis.aggregate([
  {
    $sort: { "applies": -1 }
  },
  {
    $group: {
      _id: "$job_title",
      previous_applies: { $first: "$applies" },
      current_applies: { $first: "$applies" }
    }
  },
  {
    $project: {
      job_title: "$_id",
      percentage_increase: { $multiply: [{ $divide: [{ $subtract: ["$current_applies", "$previous_applies"] }, "$previous_applies"] }, 100] }
    }
  },
  {
    $sort: { percentage_increase: -1 }
  },
  {
```

```
            $limit: 10

        }

    ]);
```

The query sorts the documents based on the number of job applications in descending order. Then, it groups the documents by job title and retains the first document's number of applies as both the previous and current applies for each group. After that, it calculates the percentage increase in job applications for each job title. The documents are then sorted in descending order based on this calculated percentage increase. Finally, the output is limited to the top 10 documents with the highest percentage increase in job applications.

17. Display Companies names job titles that have least views and applies ( for HR to know how a company should not be and for candidates to choose jobs wid least competition ) ( table : jobs_T + company_T ) :

```
db.Jobs_T.aggregate([
  {
    $lookup:
    {
      from: "Company_T",
      localField: "company.company_name",
      foreignField:
"company_details.company_name",
      as: "company_info"
    }
  },
  {
    $unwind: "$company_info"
  },
  {
    $project:
    {
      _id: 0,
      company_name:
"$company.company_name",
      job_title: 1,
      views: "$company_info.views",
      applies: "$company_info.applies"
    }
  },
  {
    $sort:
    {
      views: 1,
      applies: 1
    }
  }
]).pretty()
```

18. To know which unique titles are costing the most from sum of their max salarys in all the companies (who is eating most of the money (write it in diplomatic way ) ) (table: jobs_T ) :

```
db.Jobs_T.aggregate([
  {
    $group:
    {
      _id: "$job_title",
      total_max_salary: { $sum: "$max_salary" }
    }
  },
  {
    $sort:
    {
      total_max_salary: -1
    }
  }
]). pretty()
```

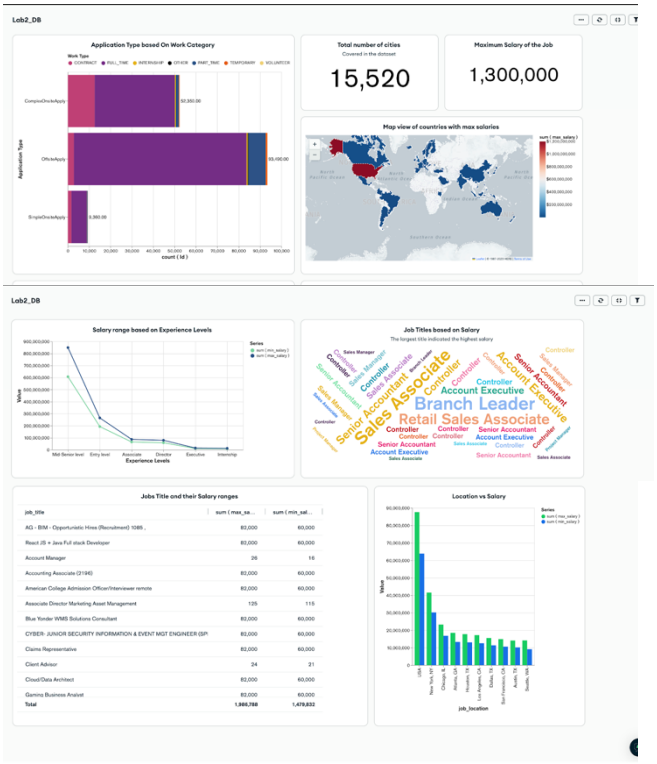
19. Which job titles are least posted in all the companies (these workers can be outsourced )( table : jobs_T ) :

```
db.Jobs_T.aggregate([
  {
    $group:
    {
      _id: "$job_title",
      count: { $sum: 1 }
    }
  },
  {
    $sort:
    {
      count: 1
    }
  }
]).pretty()
```

Data Visualization:

**DASHBOARD**

This dashboard is a comprehensive tool designed for project managers to track and analyze their project's progress and performance. It provides a visual representation of various key metrics related to job applications, job vacancies, and salary ranges.




## Link:

https://charts.mongodb.com/charts-project-0-hocab/public/dashboards/6ac354c5-8fb2-4b65-afa3-78a64c99edad

## Features:

- **Application Type based on Work Category**: This bar graph provides insights into the number of applications received for different work categories.

- **Map view of countries with max vacancies**: This map gives a geographical representation of job vacancies, highlighting the countries with the most job openings.

- **Salary Range based on Experience Level:** This line graph shows the salary range for different positions, providing a clear view of compensation based on experience levels.

- **Project Manager Position based on Salary Range:** This bar graph presents the salary range for different positions within the project management domain.

## Usage

To use this dashboard, simply navigate through the various charts and graphs. Hover over specific data points for more detailed information. This tool is designed to aid project managers in making informed decisions based on real-time data.

## VII. PERFORMANCE MEASUREMENTS

SQL:

1. Query 1:
   - Load time: 389 ms
   - Connect Time: 378 ms
   - Latency: 387 ms
   - Size in bytes: 135
   - Sent bytes: 0
2. Query 2:
   - Load time: 253 ms
   - Connect Time: 240 ms
   - Latency: 250 ms
   - Size in bytes: 285
   - Sent bytes: 0
3. Query 3:
   - Load time: 195 ms
   - Connect Time: 188 ms
   - Latency: 193 ms
   - Size in bytes: 450
   - Sent bytes: 0

MongoDB:

1. Query 1:
   - Execution Success: true
   - nReturned: 5977
   - Execution Time: 37 ms
   - Total Keys Examined: 0

- Total Docs Examined: 15520

2. Query 2:

- Execution Success: true
- nReturned: 5976
- Execution Time: 38 ms
- Total Keys Examined: 0
- Total Docs Examined: 15470

3. Query 3:

- Execution Success: true
- nReturned: 12948
- Execution Time: 75 ms
- Total Keys Examined: 0
- Total Docs Examined: 15520

Now, we can conclude that MongoDB is much faster than MySQL. And this report is formed with the help of JMeter to perform the performance measurement of MySQL and used explain () method to measure performance in MongoDB.

VIII. REFERENCES:


https://dev.mysql.com/doc/refman/8.0/en/stored-objects- security.html

https://dba.stackexchange.com/questions/60651/how-do-i- best-measure-the-query-performance

https://dev.mysql.com/doc/refman/8.0/en/trigger-syntax.html

https://stackoverflow.com/questions/37159201/which-one- is-er-diagram
https://www.lucidchart.com/pages/uml-use-case-diagram

https://www.mongodb.com/docs/compass/current/documents/

https://www.knowi.com/blog/visualization-solutions-for-mongodb/

https://www.mongodb.com/docs/atlas/

https://www.kenwalger.com/blog/nosql/mongodb/mongodb-atlas/

https://www.mydbops.com/blog/tag/mongodb-atlas/