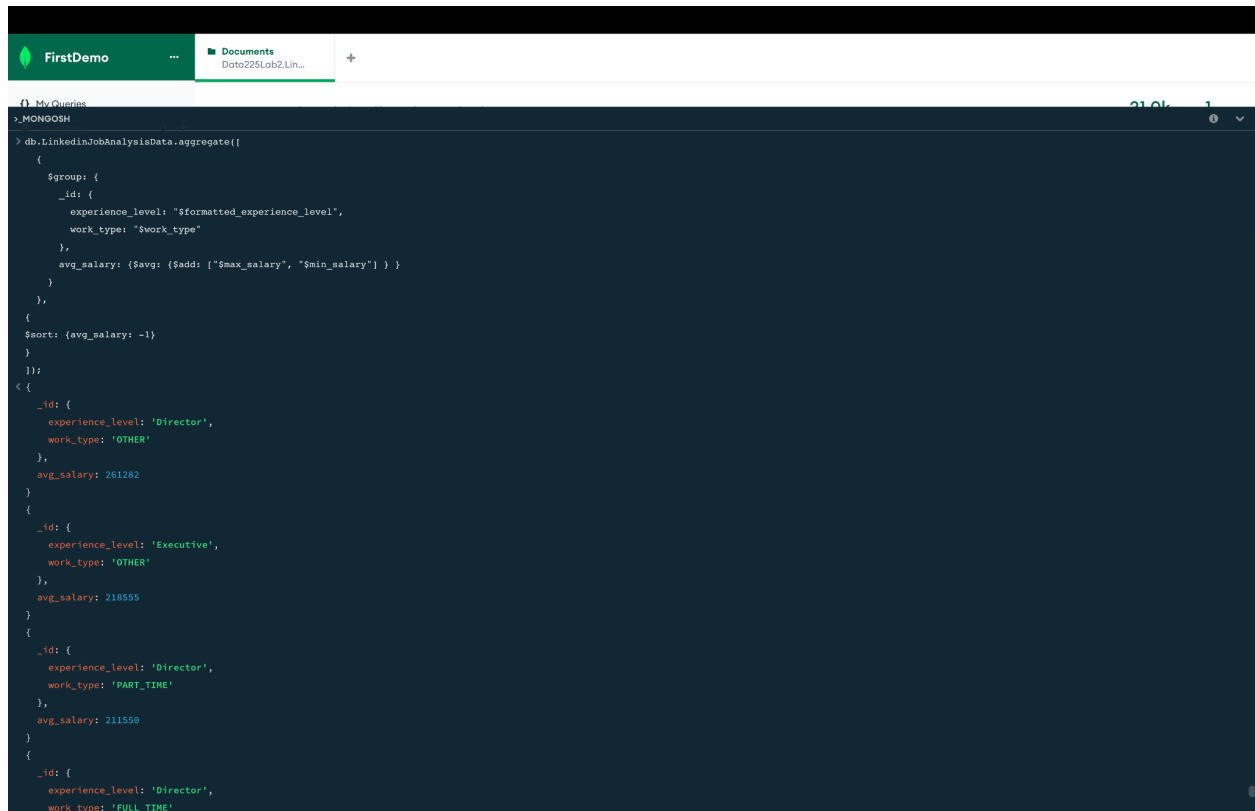


MongoDB queries

1: Find 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.



The screenshot shows a MongoDB query interface with a dark theme. At the top, there's a header with 'FirstDemo' and a 'Documents' tab. The main area displays a MongoDB aggregation query and its results. The query is as follows:

```
> 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 }
  }
])
```

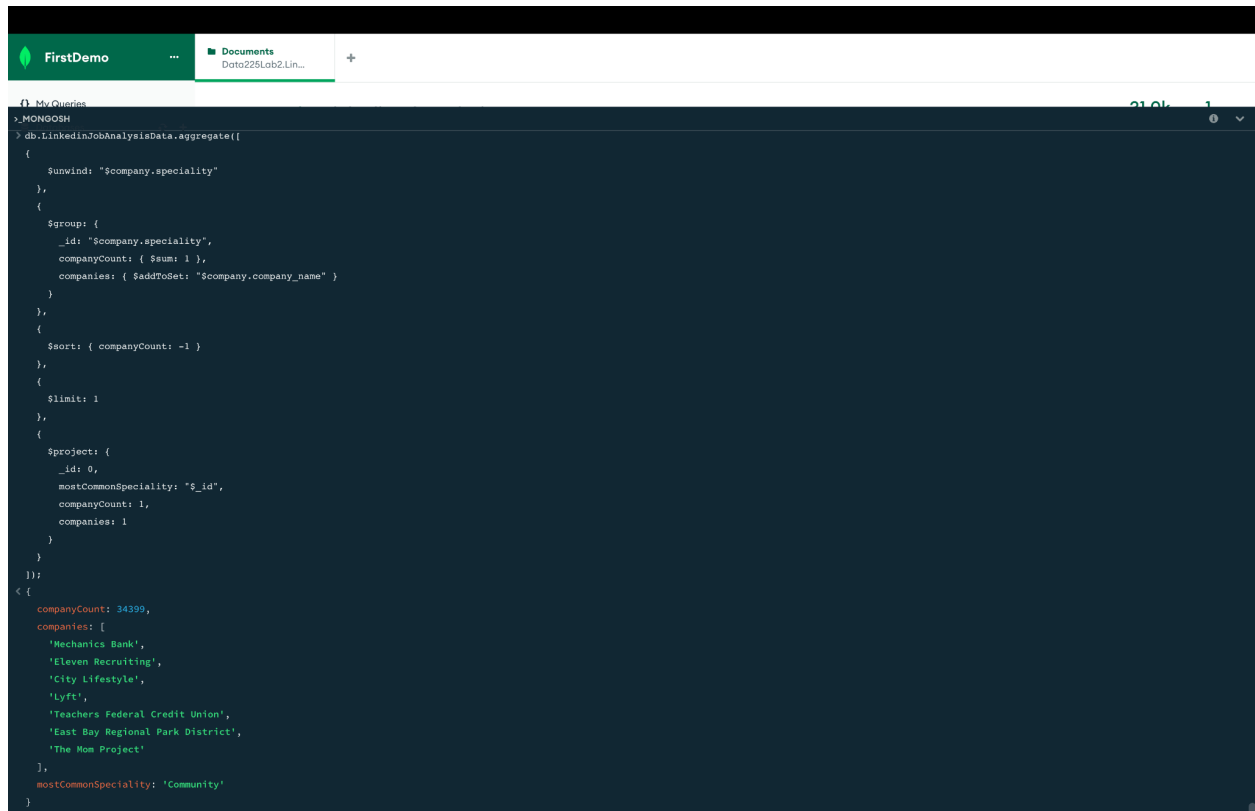
The results are displayed in a JSON-like format, showing the average salary for different combinations of experience level and work type. The results are sorted by average salary in descending order.

```
{
  "_id": {
    "experience_level": "Director",
    "work_type": "OTHER"
  },
  "avg_salary": 261282
},
{
  "_id": {
    "experience_level": "Executive",
    "work_type": "OTHER"
  },
  "avg_salary": 218555
},
{
  "_id": {
    "experience_level": "Director",
    "work_type": "PART_TIME"
  },
  "avg_salary": 211559
},
{
  "_id": {
    "experience_level": "Director",
    "work_type": "FULL_TIME"
  },
  "avg_salary": 211559
}
```

2: Most common specialty and which companies

```
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 specialty. From the output, community is the most common speciality with companies like Lyft, The Moms project, etc. having this specialty



The screenshot shows a MongoDB query interface with a green header bar containing 'FirstDemo' and a 'Documents' tab. The query is executed in the 'My Queries' section, showing the following aggregation query:

```
> 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
    }
  }
]);
```

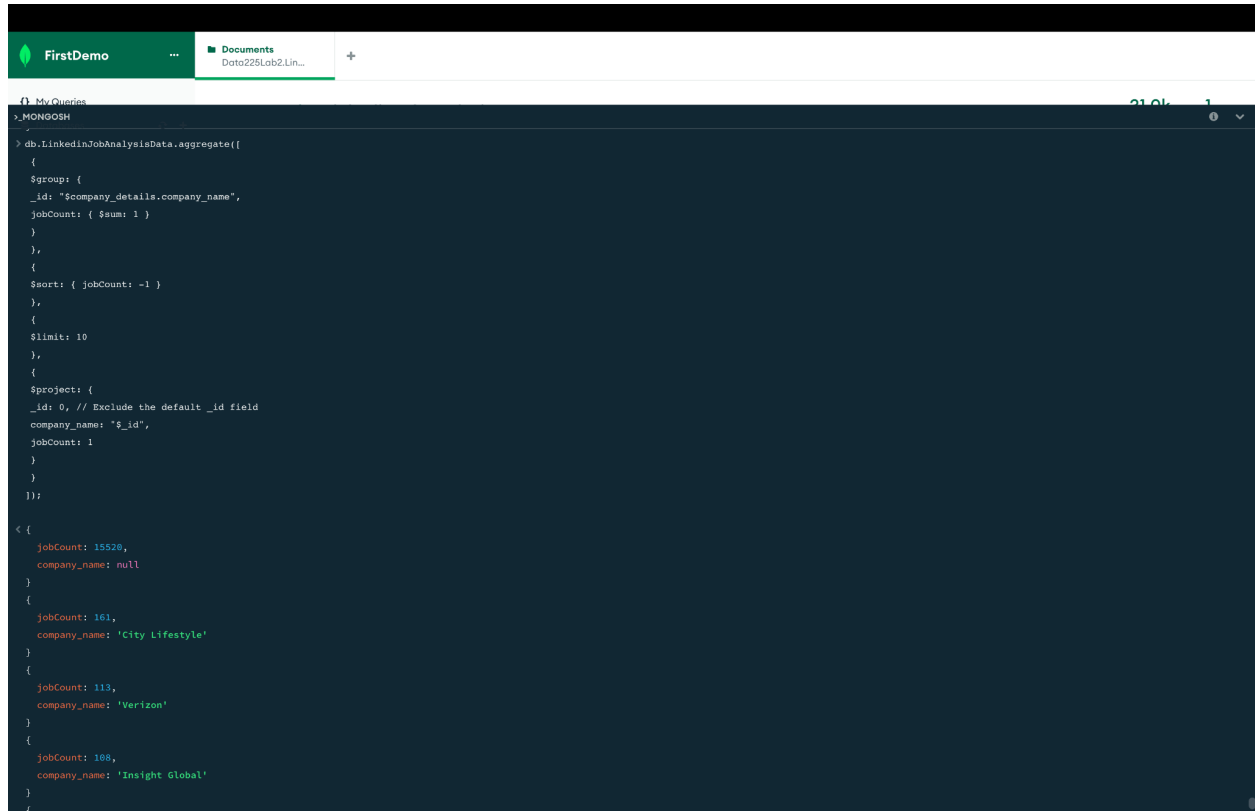
The output of the query is displayed below the query, showing the following JSON document:

```
{
  companyCount: 34399,
  companies: [
    'Mechanics Bank',
    'Eleven Recruiting',
    'City Lifestyle',
    'Lyft',
    'Teachers Federal Credit Union',
    'East Bay Regional Park District',
    'The Mom Project'
  ],
  mostCommonSpeciality: 'Community'
}
```

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 a lot of job postings where this company name is not mentioned. From the second company in the output is City Lifestyle which has a total postings of 161 offers.



The screenshot shows a MongoDB shell interface with a dark theme. The top bar includes a logo, the text 'FirstDemo', and a tab labeled 'Documents' with the file path 'Data225Lab2.Lin...'. The main area displays a MongoDB aggregation query and its output.

```
> use myqueries
> use myqueries
> 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
    }
  }
])
```

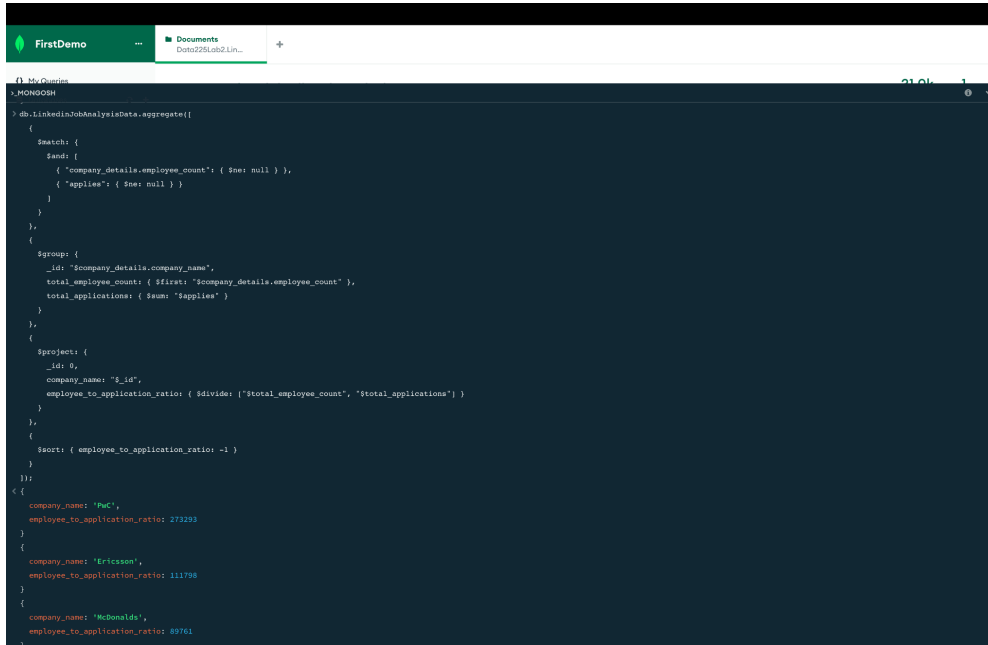
The output shows the top 10 companies by job count:

```
< {
  jobCount: 15528,
  company_name: null
}
{
  jobCount: 161,
  company_name: 'City Lifestyle'
}
{
  jobCount: 113,
  company_name: 'Verizon'
}
{
  jobCount: 188,
  company_name: 'Insight Global'
}
```

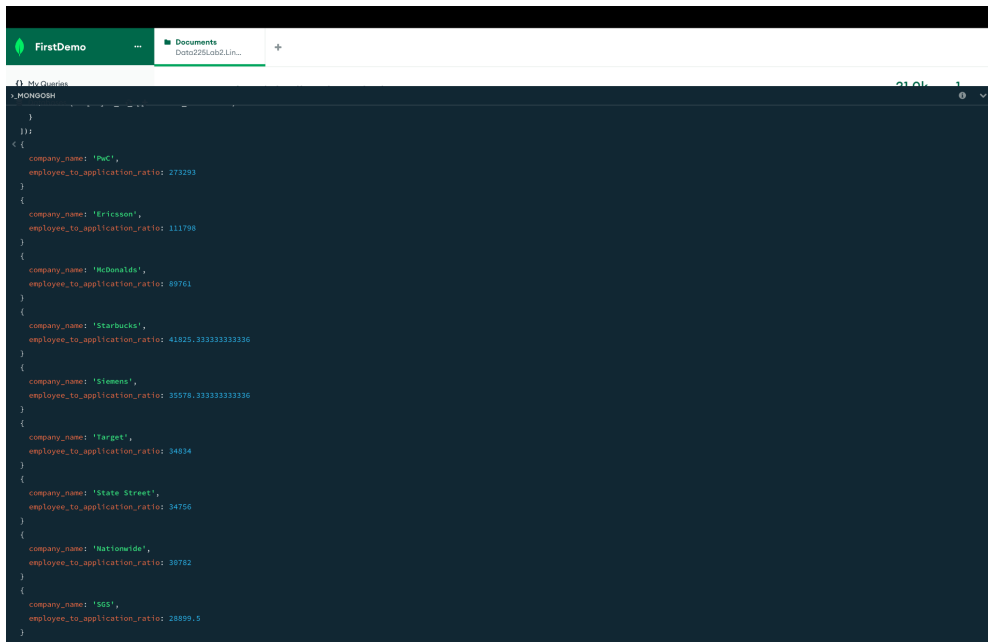
4:Ratio of the employee count to the applications

```
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 ratio with 273293, followed by Ericsson, McDonald and Starbucks. However, it is important to keep in mind that if there are a lot of job postings this subsequently leads to a lot of job applications.



```
x.MONGOSH
> db.linkedInJobAnalysisData.aggregate([
  {
    $match: {
      $and: [
        { "company_details.employee_count": { $ne: null } },
        { "applications": { $ne: null } }
      ]
    },
    {
      $group: {
        _id: "$company_details.company_name",
        total_employee_count: { $first: "$company_details.employee_count" },
        total_applications: { $sum: "$applications" }
      }
    },
    {
      $project: {
        _id: 0,
        company_name: "$_id",
        employee_to_application_ratio: { $divide: [ "$total_employee_count", "$total_applications" ] }
      }
    },
    {
      $sort: { employee_to_application_ratio: -1 }
    }
  ]
});
{
  company_name: 'PwC',
  employee_to_application_ratio: 273293
}
{
  company_name: 'Ericsson',
  employee_to_application_ratio: 111798
}
{
  company_name: 'McDonalds',
  employee_to_application_ratio: 89761
}
```



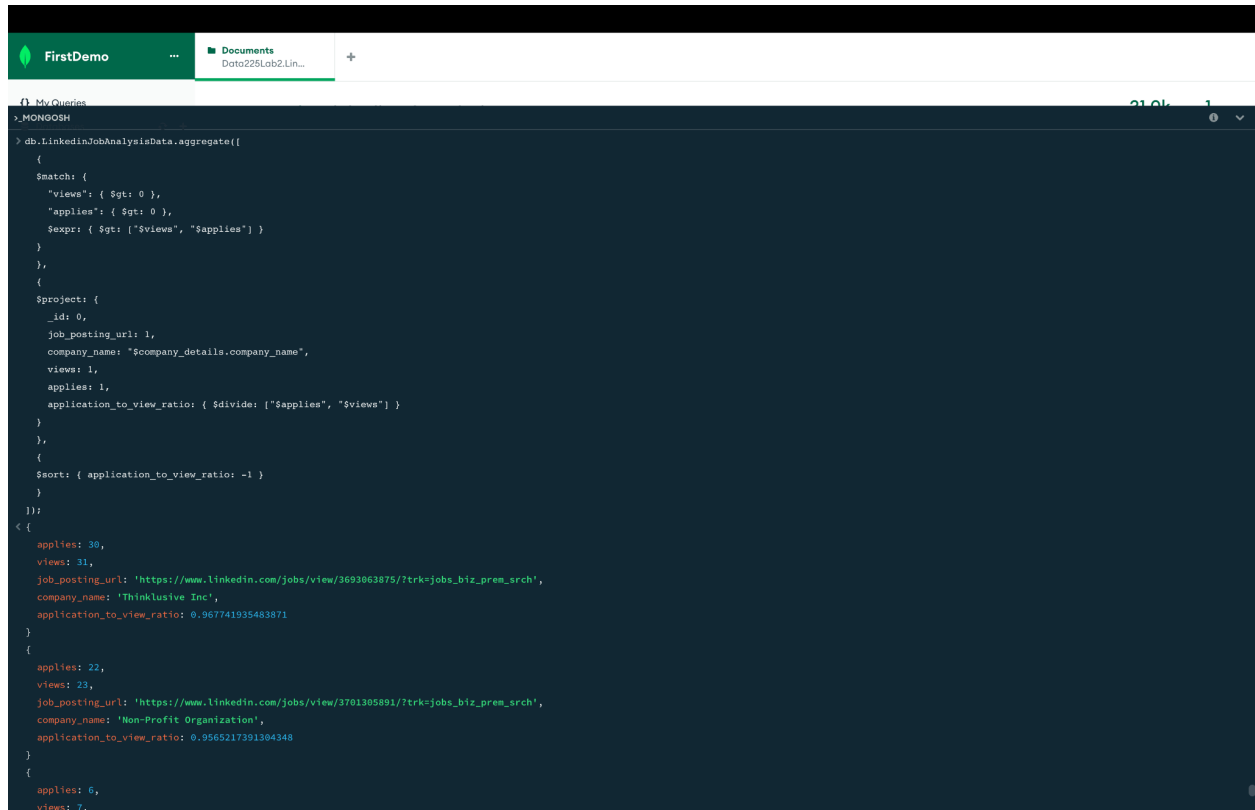
```

{
  company_name: 'Starbucks',
  employee_to_application_ratio: 41825.333333333336
}
{
  company_name: 'Siemens',
  employee_to_application_ratio: 35578.333333333336
}
{
  company_name: 'Target',
  employee_to_application_ratio: 34834
}
{
  company_name: 'State Street',
  employee_to_application_ratio: 34756
}
{
  company_name: 'Nationwide',
  employee_to_application_ratio: 38782
}
{
  company_name: 'BGS',
  employee_to_application_ratio: 28899.5
}
}
```


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 finds the ratio of Application to view of the job postings. Here we can understand the relationship of the views and the applications for that particular job posting.



The screenshot shows a MongoDB shell interface with a green header bar containing 'FirstDemo' and a 'Documents' tab. The main area is a dark-themed code editor displaying a MongoDB aggregation query. The query uses the `$match`, `$project`, and `$sort` operators to filter documents, project specific fields, and sort them by the `application_to_view_ratio` in descending order. The results are displayed in a light blue monospace font, showing three documents with their respective fields and the calculated 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 }
  }
]);
< {
  applies: 38,
  views: 31,
  job_posting_url: 'https://www.linkedin.com/jobs/view/3693063875/?trk=jobs_biz_prem_src',
  company_name: 'Thinklusive Inc',
  application_to_view_ratio: 0.967741935483871
}
{
  applies: 22,
  views: 23,
  job_posting_url: 'https://www.linkedin.com/jobs/view/3701385891/?trk=jobs_biz_prem_src',
  company_name: 'Non-Profit Organization',
  application_to_view_ratio: 0.9565217391304348
}
{
  applies: 6,
  views: 7,
```