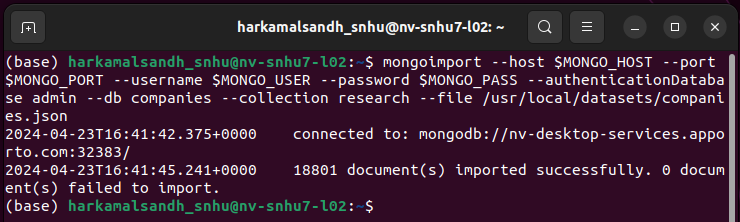
Harkamal Sandhu

CS-499

**Enhancement-3 SQL to NoSQL**

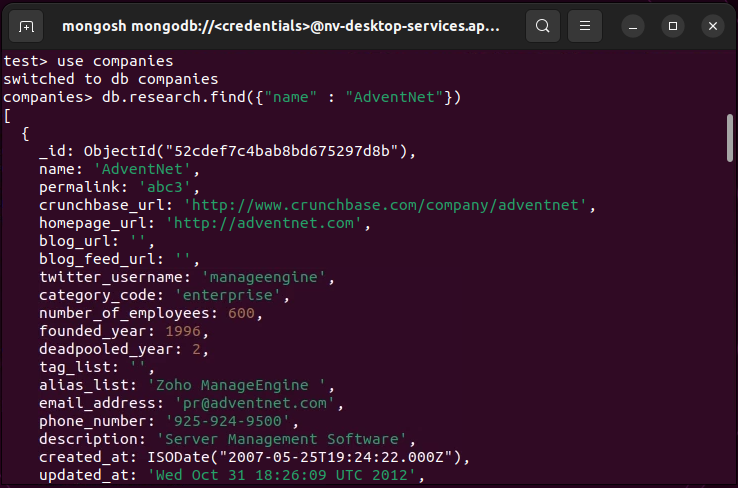
Data Aggregation using MongoDB

1. Using the mongoimport tool, created the database “companies” by loading the documents found in the “companies.json” file into the “research” collection.



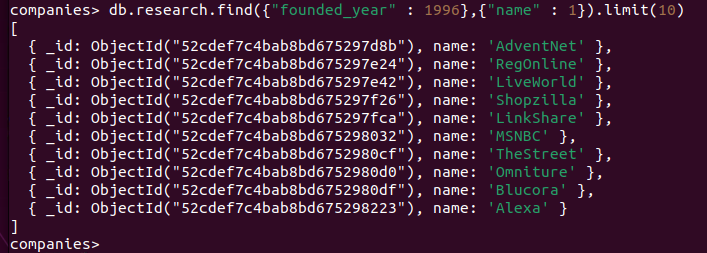
1. After importing, verified the data load using the following MongoDB queries:

* db.research.find({"name" : "AdventNet"})



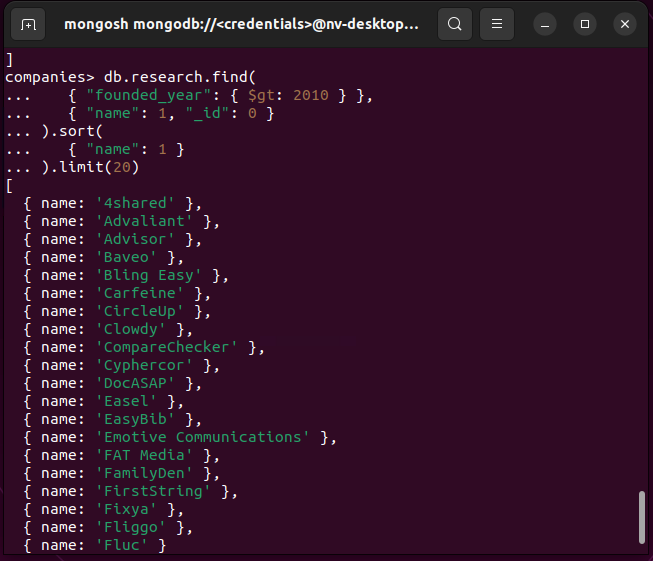


* db.research.find({"founded\_year" : 1996},{"name" : 1}).limit(10)

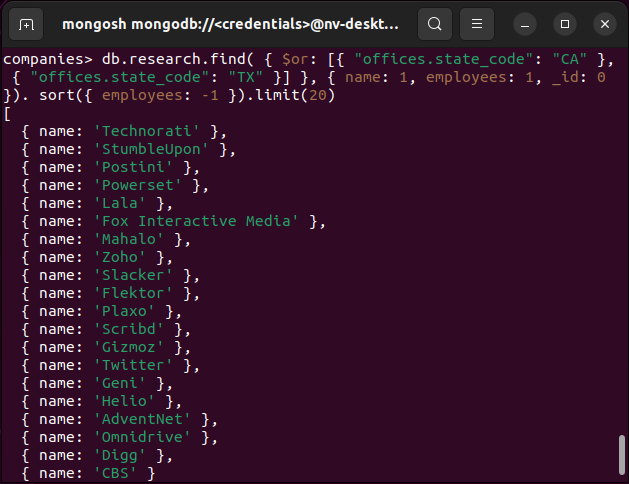


1. Analysed the data using MongoDB queries:

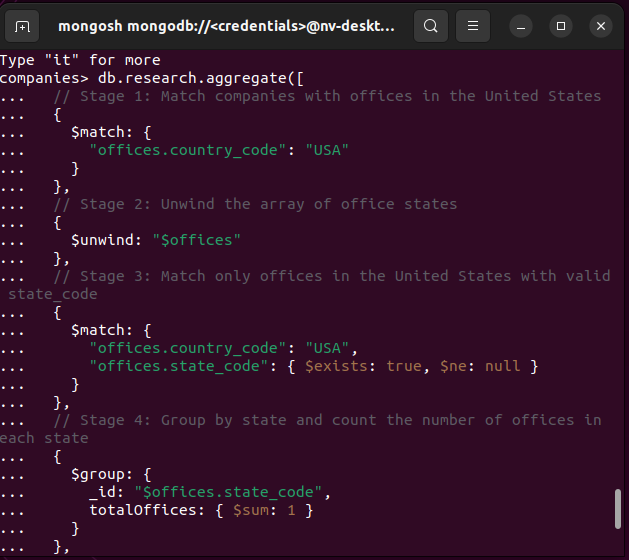
* List only the first 20 names of companies founded after the year 2010, ordered alphabetically.

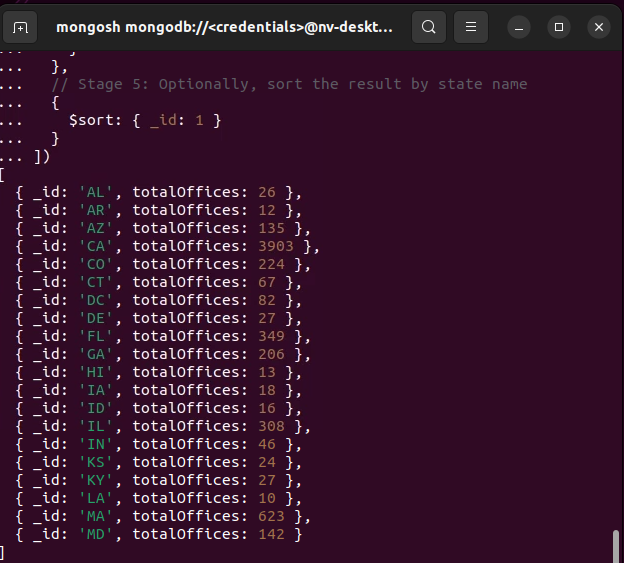


* List only the first 20 names of companies with offices in either California or Texas, ordered by the number of employees and sorted largest to smallest.



1. Implemented a MongoDB aggregation pipeline to show the total number of offices by state for all companies that have offices in the United States.





The provided aggregation pipeline performs a series of operations to count the number of offices in each state within the United States for companies in the research collection. Here's an explanation of each stage:

1. **Stage 1: $match**

* This stage filters the documents (companies) to include only those that have at least one office located in the United States ("offices.country\_code": "USA").

1. **Stage 2: $unwind**

* The $unwind stage deconstructs the offices array field, creating a separate document for each element in the array. This allows us to process each office object individually in subsequent stages.

1. **Stage 3: $match**

* This second $match stage further filters the documents to include only those offices that are located in the United States ("offices.country\_code": "USA") and have a non-null, existing state\_code field ("offices.state\_code": { $exists: true, $ne: null }).

1. **Stage 4: $group**

* The $group stage groups the documents by the state\_code field (\_id: "$offices.state\_code"). For each group (unique state\_code), it counts the number of documents (offices) in that group using the $sum accumulator (totalOffices: { $sum: 1 }).

1. **Stage 5: $sort**

* The final $sort stage orders the resulting documents by the \_id field (which corresponds to the state\_code) in ascending order ({ \_id: 1 }). This effectively sorts the output by state code.

Note: It is mandatory to regularly update MongoDB and other software to patch known vulnerabilities to ensure security of data. Database is restricted to authorized users only; those have valid username and password.