# MongoDB Advanced Queries and Aggregation Pipeline

Victor Udeh

30 June, 2024

## Table of Contents

1. Introduction

2. Data Import

3. Verification of Data Import

4. MongoDB Queries

5. Aggregation Pipeline

6. Conclusion

## Introduction

In this document, we demonstrate the use of advanced query operations in MongoDB. These operations allow for more programmatic data analysis at the database layer. The aggregation pipeline in MongoDB provides a set of operators for performing set theory operations like grouping and intersections within collections.

## Data Import

The following command was used to import the data into the MongoDB database:

mongoimport --host nv-desktop-services.apporto.com --port 31593 --username root --password 61HyRsVrum --authenticationDatabase admin --db companies --collection research --file /usr/local/datasets/companies.json

Result:  
2024-06-30T13:53:30.863+0000 connected to: mongodb://nv-desktop-services.apporto.com:31593/  
2024-06-30T13:53:32.982+0000 18801 document(s) imported successfully. 0 document(s) failed to import.

## Verification of Data Import

To ensure the data was successfully imported, the following queries were executed:

Query: Find the company named "AdventNet"

Command:

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

Result:  
Screenshot provided below

Query: Find companies founded in 1996, displaying only the name field, limiting results to 10

Command:

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

Result:  
Screenshot provided below

## MongoDB Queries

Query 1: Companies Founded After 2010

Command:

db.research.find({"founded\_year": {$gt: 2010}}, {"name": 1, "\_id": 0}).sort({"name": 1}).limit(20).pretty()

Result:

Screenshot provided below   
  
  
  
Query 2: Companies with Offices in CA or TX

Command:

db.research.find({"offices.state\_code": {$in: ["CA", "TX"]}}, {"name": 1, "number\_of\_employees": 1, "\_id": 0}).sort({"number\_of\_employees": -1}).limit(20).pretty()

Result:  
Screenshot provided below

## Aggregation Pipeline

The following aggregation pipeline was designed to show the total number of offices by state for all companies that have offices in the United States:

Command:

db.research.aggregate([  
 { $unwind: "$offices" },  
 { $match: { "offices.country\_code": "USA" } },  
 { $group: { \_id: "$offices.state\_code", totalOffices: { $sum: 1 } } },  
 { $sort: { totalOffices: -1 } }  
])

Result:

Screenshot provided below   
  
  
  
  
Explanation of the Aggregation Pipeline:

- $unwind: Deconstructs the offices array field to output a document for each office.  
- $match: Filters documents to include only those with offices in the USA.  
- $group: Groups the documents by state\_code and calculates the total number of offices for each state.  
- $sort: Sorts the grouped results by the total number of offices in descending order.

## Conclusion

In this document, we successfully demonstrated advanced MongoDB queries and the use of aggregation pipelines. These operations enhance the ability to perform detailed data analysis directly within the database, leveraging MongoDB's powerful querying capabilities.

SCREENSHOTS  
A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer program

Description automatically generated