# **CS 340 Module Eight Assignment**

## **Overview**

In this assignment, we demonstrate advanced MongoDB query operations. The goal is to create a database, run specific queries, and implement an aggregation pipeline to analyze data from a NoSQL database.

## **Task 1: Database Creation**

### **Step 1: Importing the Database**

Using the **mongoimport** tool, the **companies** database is created by loading the provided **companies.json** file.

mongoimport --uri ‘{URL OF DB}’ --db companies --collection research --file /usr/local/datasets/companies.json

### **Step 2: Query Verification**

After the import, the following MongoDB queries are used to verify that the data has been loaded correctly:

#### **a. Find a company named AdventNet:**

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

This query searches the database for the company "AdventNet" to verify the correct loading of data.

#### **b. Find companies founded in the year 1996:**

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

This query lists the first 10 companies founded in 1996, showing only the company names.

## **Task 2: MongoDB Queries**

### **a. Companies Founded After 2010**

This query lists the first 20 companies founded after the year 2010, ordered alphabetically.

db.research

.find({ founded\_year: { $gt: 2010 } }, { name: 1, \_id: 0 })

.sort({ name: 1 })

.limit(20)

* **Description:** This query filters companies founded after 2010 (**"$gt": 2010**) and sorts them alphabetically (**sort({"name": 1})**). The **limit(20)** restricts the results to 20 companies.

### **b. Companies with Offices in California or Texas**

This query lists the first 20 companies with offices in either California or Texas, ordered by the number of employees and alphabetically.

db.research

.find(

{ "offices.state\_code": { $in: ["CA", "TX"] } },

{ name: 1, number\_of\_employees: 1, \_id: 0 }

)

.sort({ number\_of\_employees: -1, name: 1 })

.limit(20)

* **Description:** This query uses **"$in"** to filter companies with offices in California or Texas. The results are then sorted first by the number of employees (**"number\_of\_employees": -1** for descending) and second by company name (**"name": 1**). The **limit(20)** restricts the results to 20.

## **Task 3: MongoDB Aggregation Pipeline**

### **Aggregation Pipeline to Count Offices by State**

This pipeline counts the total number of offices by state for companies with multiple offices.

db.research.aggregate([

{ $unwind: "$offices" }, // Unwind the offices array

{ $group: { \_id: "$offices.state\_code", office\_count: { $sum: 1 } } },

{ $sort: { office\_count: -1 } } // Sort by office count in descending order

])

* **Description:**
  + **$unwind** deconstructs the **offices** array, creating a separate document for each office.
  + **$group** groups the documents by state (**"$offices.state\_code"**) and calculates the number of offices per state using **"$sum": 1**.
  + **$sort** orders the states by the number of offices in descending order (**"$office\_count": -1**).