CS628 Full-Stack Development – Web App

**HOS07A: MongoDB Atlas**

Created by Veerendra Jagatha on Aug 09, 2023

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* Version numbers may not match the most current version at the time of writing. If given the option to choose between the stable release (long-term support) or the most recent, please select the **stable release** rather than the beta-testing version.
* There might be subtle discrepancies along with the steps. Please **use your best judgment** while going through this cookbook-style tutorial to complete each step.
* If you are not familiar with a terminal, command line, and bash scripts, check out this video: <https://youtu.be/Dp7uw9c6QH8>
* All the steps and concepts in this tutorial are from references, so if you encounter problems, please **try to read and compare the references to solve the problem**. If you still can't solve the problem, please contact your course TA.
* **Avoid copy-pasting code from the book or the GitHub repository**. Instead, type out the code yourself. Resort to copy-pasting only when you are stuck and find things not working as expected.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

#### **Readings and Examples:**

* Visit [CS 628 Repository for Examples](https://github.com/samchung0117/cs628-examples).
  + Select the related module.
  + Visit the README.md file.
  + Find examples for your practices.

**Learning Outcomes**

* Section 1: Accessing GitHub Codespaces
* Section 2: SQL vs NoSQL Differences
* Section 3: MongoDB Introduction
* Section 4: Sign up for MongoDB Atlas
* Section 5: Setting up your first MongoDB Database
* Section 6: Setting up environment - MongoDB for VS Code on GitHub Codespace
* Section 7: Learn MongoDB step by step
* Section 8: Pushing your work to GitHub

**Section 1: Accessing GitHub Codespaces**

Refer the steps from [TA Center](https://cityuseattle.github.io/docs/git/github_codepsace/) to get started with this week’s module GitHub Codespace.

**Note: For this week deliverables, you will take the screenshots of MongoDB commands from section 7 and save them in current module folder.**

Before we begin let us try to understand differences between SQL and NoSQL databases and what is MongoDB.

**Section 2: SQL vs NoSQL Differences**

**SQL (Structured Query Language) Databases:**

1. Relational databases with fixed schemas.
2. Data is stored in tables with rows and columns.
3. ACID (Atomicity, Consistency, Isolation, Durability) transactions ensure data integrity.
4. Suitable for structured data and well-defined relationships.
5. Examples: MySQL, PostgreSQL, SQLite, Oracle.

**NoSQL (Not Only SQL) Databases:**

1. Flexible, schema-less databases.
2. Data is stored in various formats like documents, key-value pairs, or graphs.
3. Often designed for horizontal scalability and high performance.
4. Better suited for unstructured or semi-structured data.
5. Examples: MongoDB, Cassandra, Redis, Couchbase.

**Section 3: MongoDB Introduction**

1. **Document-Oriented Database:**

* MongoDB is a NoSQL database that stores data in JSON-like documents.
* Documents can have varying structures, allowing flexibility in data representation.

1. **Key Concepts:**

* **Collections**: Equivalent to tables in relational databases, store related documents.
* **Documents**: Individual data records, represented in BSON format (Binary JSON).
* **Fields**: Data elements within a document, like columns in SQL.

1. **Advantages:**

* Flexible schema accommodates changing data requirements.
* Fast read and write operations.

1. **Basic Operations:**

* CRUD (Create, Read, Update, Delete) operations like SQL, but using MongoDB syntax.

1. **Querying:**

* Use a flexible query language to retrieve documents that match specific criteria.
* Filtering, sorting, and aggregation operations are supported.

1. **Aggregation Framework:**

* Advanced data processing using stages like grouping, filtering, and transforming.

1. **Indexes:**

* Create indexes to improve query performance on specific fields.
* Helps in speeding up read operations.

1. **Geospatial Queries:**

* MongoDB supports geospatial queries for location-based data.

1. **Use Cases:**

* Web applications, content management, cataloging, real-time analytics.

**Section 4: Sign up for MongoDB Atlas**

1. Open this [link](https://www.mongodb.com/atlas/database) and click on Try Free to sign up for MongoDB Atlas.

A screenshot of a computer

Description automatically generated

1. You will see a screen as shown below. You can sign up with your email or Gmail.

A screenshot of a login form

Description automatically generated

1. Use Gmail or your account to sign up and then sign in. You will see a screen as shown below.

A screenshot of a computer

Description automatically generated

1. Select the options as shown in the image and click finish.

**Section 5: Setting up your first MongoDB Database**

1. You will be asked to deploy your first database. Select M0 Free cluster, AWS provider and give a name for your cluster and click on Create.  
     
   A screenshot of a computer

   Description automatically generated
2. Once the above step is done, you will be prompted with a QuickStart under Security. You need to create username and password for your MongoDB Connection.

A screenshot of a computer

Description automatically generated

1. Click on Finish after creating the user account. Remember your credentials. We need them for our connection string later.
2. Next step is to enable access to Atlas from anywhere. Ideally, we will only allow connections to MongoDB Atlas from trusted IP addresses. For now, we can use wild cards and allow from anywhere.
3. Under Security à Network Access, click on ADD IP ADDRESS.

A screenshot of a phone

Description automatically generated

1. Click on à Allow Access from Anywhere and confirm

A screenshot of a computer

Description automatically generated

**Section 6: Setting up environment - MongoDB for VS Code on GitHub Codespace**

1. In this GitHub codespace module, navigate to the Extensions section. Search for the "MongoDB for VS Code" Extension and proceed to install it.

A screenshot of a computer

Description automatically generated

1. Upon installation, you'll observe the inclusion of the MongoDB Extension. Click on it, then select "Add Connection." A prompt will appear, requesting your connection string.

A screenshot of a computer

Description automatically generated

1. Retrieve the connection string from Atlas by navigating to the "Overview" section and then clicking on "Connect."

A screenshot of a computer

Description automatically generated

1. Obtain the connection string from the "MongoDB for VS Code" Extension. Replace the password and username with your account which we created earlier.

A screenshot of a computer

Description automatically generated

1. Input the acquired connection string into the prompt and confirm by pressing the "Enter" key.

A screenshot of a computer

Description automatically generated

1. Your connection will be successfully established.

A screenshot of a computer

Description automatically generated

1. Click on "Create Playground" and subsequently remove the automatically generated script to begin with a blank slate.

A screenshot of a computer screen

Description automatically generated

1. Ensure you save the file in your current HOS directory.

**Note:** As stated [here](https://www.mongodb.com/docs/atlas/troubleshoot-connection/#incorrect-connection-string-format) if your password contains special character, you need to encode them and then use it in the connection string.

**Section 7: Learn MongoDB step by step**

You will learn MongoDB commands by executing one by one in the Playground we created earlier. MongoDB Playgrounds are interactive JavaScript environments designed for prototyping queries, aggregations, and MongoDB commands. They provide useful syntax highlighting to assist your work. In MongoDB for VS Code, playgrounds are recognized by files bearing the “.mongodb.js” extension.

Type the following commands and hit run button on the right corner of the screen to execute them. Capture the screenshots of the Playground Results.

A screenshot of a phone

Description automatically generated

1. The show dbs command is used to display a list of available databases in the MongoDB server instance.

A close-up of a white background

Description automatically generated

1. The use CS628Practice command is used to switch to the specified database named "CS628Practice." If the database doesn't exist, MongoDB will create it.

A close-up of a logo

Description automatically generated

1. The db command is used to display the name of the current selected database in MongoDB.



1. The db.createCollection('users') command is used to create a new collection named "users" within the current selected database.



1. The db.users.insertMany([...]) command is used to insert multiple user profiles into the "users" collection. Each profile includes fields like name, age, and email.

A screenshot of a computer program

Description automatically generated

1. The db.users.find() command is used to retrieve and display all documents (user profiles) from the "users" collection.

A green text on a white background

Description automatically generated

1. The db.users.find({ age: { $lt: 30 } }) command is used to retrieve user profiles from the "users" collection where the age is less than 30.



Other comparison operators similar to $lt (less than) in MongoDB include:

* $gt: Matches values that are greater than a specified value.
* $lte: Matches values that are less than or equal to a specified value.
* $gte: Matches values that are greater than or equal to a specified value.
* $eq: Matches values that are equal to a specified value.
* $ne: Matches values that are not equal to a specified value.
* $in: Matches values that exist in a specified array.
* $nin: Matches values that do not exist in a specified array.
* $exists: Matches documents that have the specified field.
* $type: Matches documents where the value of a field has a specific data type.

1. The db.users.updateOne({ name: 'user1' }, { $set: { age: 31 } }) command is used to update a single user profile in the "users" collection. It finds the document with the name "user1" and sets the "age" field to 31.

A close-up of a white background

Description automatically generated

1. The db.users.deleteOne({ name: 'user2' }) command is used to delete a single user profile from the "users" collection. It removes the document with the name "user2" from the collection.

A close-up of a sign

Description automatically generated

1. The db.users.aggregate([...]) command with the specified aggregation pipeline calculates the average age of users in the "users" collection. It groups all documents (\_id: null) and calculates the average age using the $avg aggregation operator on the 'age' field.

A close-up of words

Description automatically generated

1. The db.users.createIndex({ email: 1 }) command is used to create an ascending index on the "email" field of the "users" collection. This index improves the speed of searches on the "email" field for faster retrieval of data.

A close-up of a message

Description automatically generated

1. Enhance text searches by creating a text index on the "name" and "email" fields in the "users" collection, allowing efficient text-based queries, such as searching for the term "user1". The db.users.find({ $text: { $search: 'user1' } }) command performs a text search on the "users" collection. It searches for documents where the text index matches the search term "user1," potentially across multiple fields.

A close-up of a computer code

Description automatically generated

1. The db.users.find().sort({ age: -1 }) command retrieves user profiles from the "users" collection and sorts them in descending order based on the "age" field.

A close-up of words

Description automatically generated

1. The db.users.find().limit(2).skip(1) command retrieves a limited number of user profiles from the "users" collection, starting from the second document (skipping the first one). This is commonly used for implementing pagination in database queries.

A number of a number

Description automatically generated

1. The db.users.aggregate([...]) command with the specified aggregation pipeline groups and counts user profiles in the "users" collection based on their ages. It uses the $group stage to group documents by the "age" field and calculates the count using the $sum aggregation operator. The results are then sorted in ascending order using the $sort stage.

A computer code with text

Description automatically generated with medium confidence

1. The db.users.aggregate([...]) command with the specified aggregation pipeline finds the oldest and youngest users in the "users" collection. It uses the $group stage to calculate the maximum age using the $max aggregation operator and the minimum age using the $min aggregation operator, both without grouping. The result is a single document with the calculated maximum and minimum ages.

A group of text on a white background

Description automatically generated

1. The following queries explain how to find places within a certain radius of a specified point using geospatial queries, where places locations are stored as coordinates and indexed using a 2dsphere index.

A screenshot of a computer code

Description automatically generated

1. Execute advanced queries to find users whose names start with 'u' or 'C', using the $or operator with regular expressions for case-insensitive matching.

A computer code with text

Description automatically generated with medium confidence

1. Perform aggregation to calculate the total age of users by grouping all documents with a null identifier, summing the ages using the $sum aggregation operator.

A close-up of words

Description automatically generated

1. Perform cleanup by dropping the "users" collection, removing it from the database.

A close-up of a white background

Description automatically generated

1. Display a list of collections in the current database using the show collections command.

A close-up of a white background

Description automatically generated

**Section 8: Pushing your work to GitHub**

* 1. Go to Source Control on your GitHub codespace and observe the pending changes.

Graphical user interface, text, application

Description automatically generated

* 1. Type the Message for your changes in the Message box on the top. For example,” **Submission for Module07 – Your Name**”
  2. Click on the dropdown beside the commit button and select **Commit & Push** to update the changes to your repository main branch.
  3. Select **Yes** when prompted.

Graphical user interface, application

Description automatically generated