**CO527: Advanced Database Systems**

**Lab 06 - Introduction to NoSQL databases using MongoDB**

Ranage R.D.P.R.

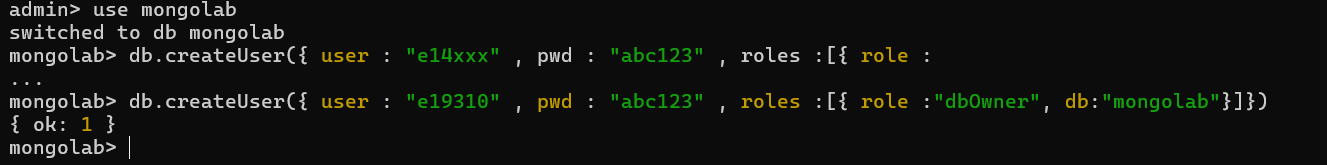
E/19/310

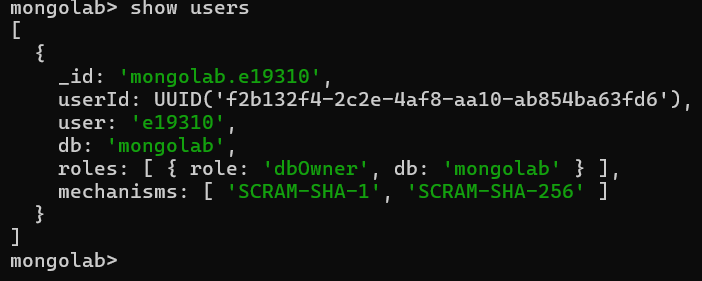
**Preparation**

**1. Once you have installed and started the Mongodb you can log into your server as root.**

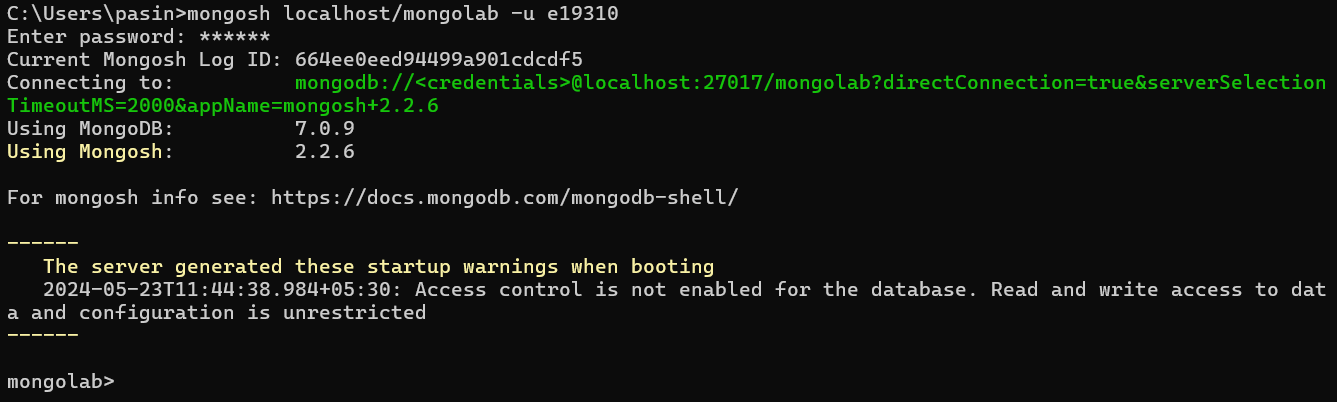
**2. Create a new database.**

**3. Now create a new user to access the database.**

****

****

**4. Log out of the mongo shell and log back in using the user you created.**

****

**Exercises**

**Exercise 1: Data Validation**

Creating the customer collection with a custom data validation function

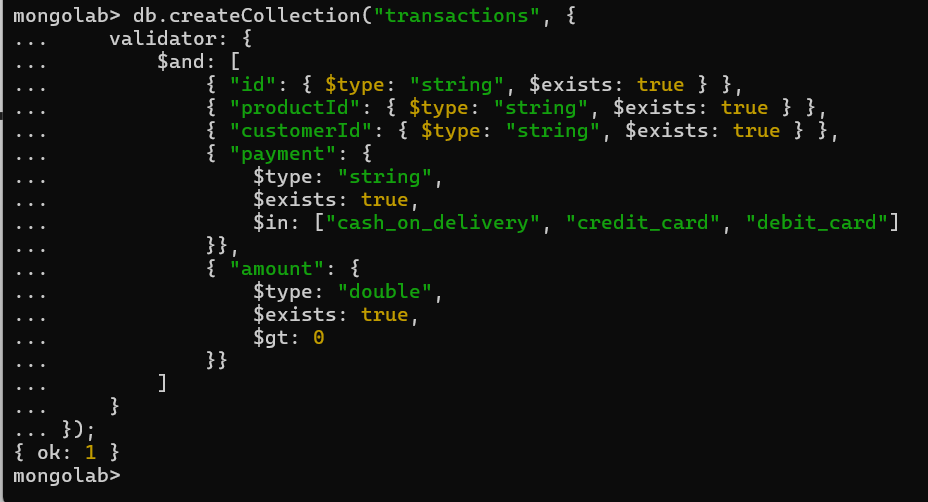


**Write code to create collections for Product and Transaction implementing the following validation rules.**

**1. Product: all fields are required and must have the proper type. Price must be greater than 0.**

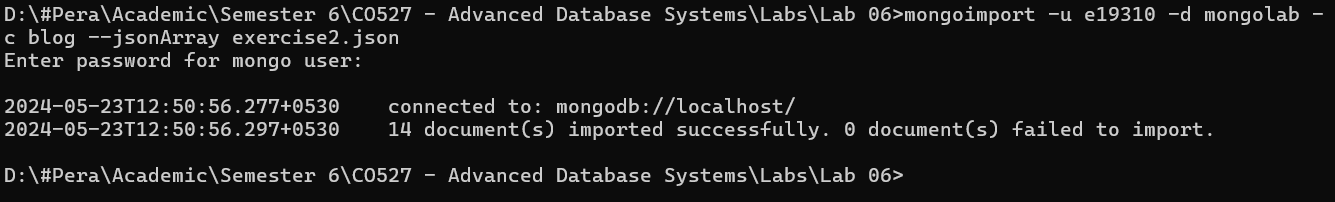
****

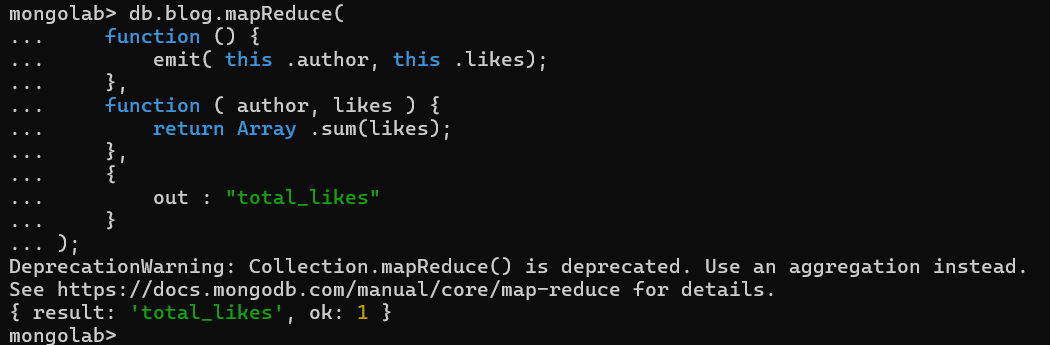
**2. Transaction: all fields are required and must have the proper type. Payment should be one of “cash\_on\_delivery”, “credit\_card” or “debit\_card”. Amount must be greater than 0.**

****

**Exercise 2: Aggregation & Map-Reduce**

Importing some generated data as exercise2.json

****

****

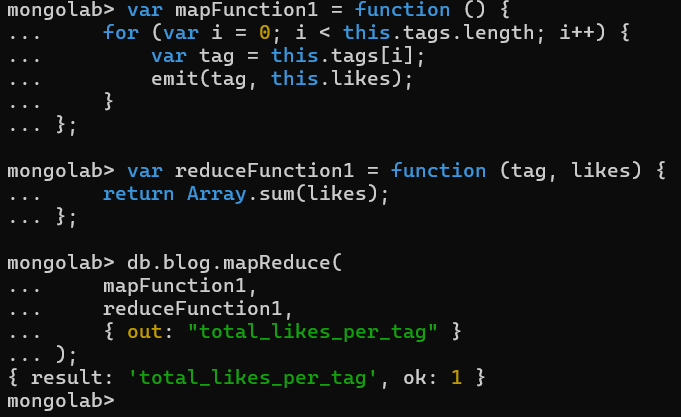
The map function is invoked for each document in the collection (blog). It emits key-value pairs, where the key is the author of the document (this.author) and the value is the number of likes (this.likes). The emit() function is used to emit these key-value pairs.

The reduce function is invoked for each unique key emitted by the map function. It receives the key (author) and an array of values (likes) associated with that key. In this case, it calculates the total number of likes for each author by summing up the values (likes) using Array.sum().

**Result**



**1. The total number of likes for each tag. Suppose a post with 3 likes is tagged [“a”,”b”,”c]. Then each of these tags get 3 likes. Hint: use a Javascript for loop to iterate over tags of a post.**

****

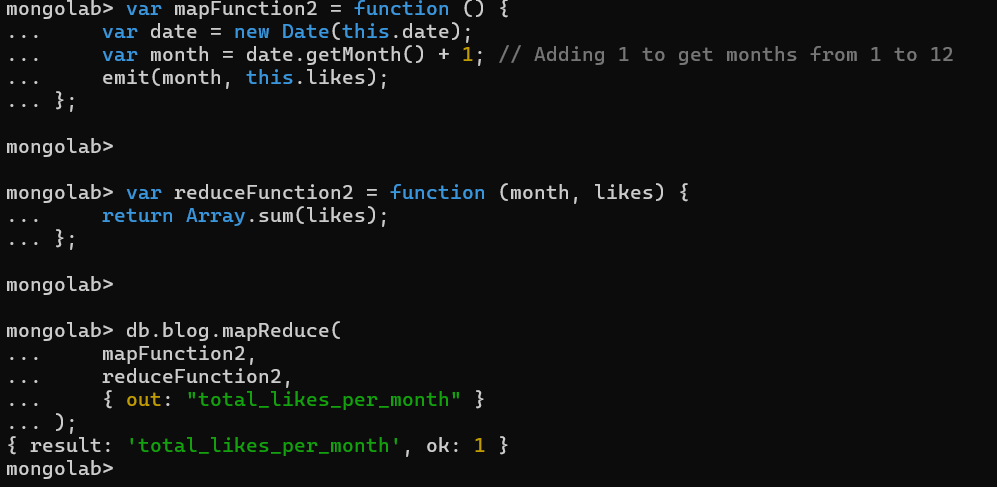
The map function iterates over the tags array of each document in the blog collection. For each tag, it emits a key-value pair where the key is the tag itself (tag) and the value is the number of likes (this.likes) associated with the post. The emit() function is used to emit these key-value pairs.

The reduce function is invoked for each unique key emitted by the map function. It receives the key (tag) and an array of values (likes) associated with that key. In this case, it calculates the total number of likes for each tag by summing up the values (likes) using Array.sum().

**Result**

****

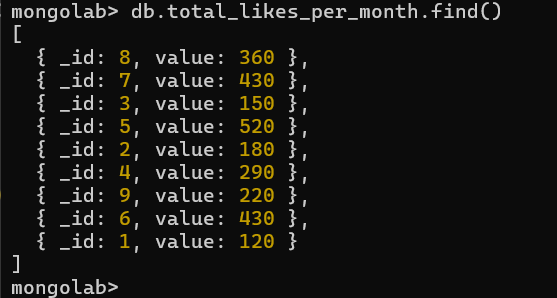
**2. The total number of likes per month (over all years.) Hint: In your map function create a Date object out of the date field. Then use getMonth() to extract the month (which strangely returns a value from 0 to 11.)**

****

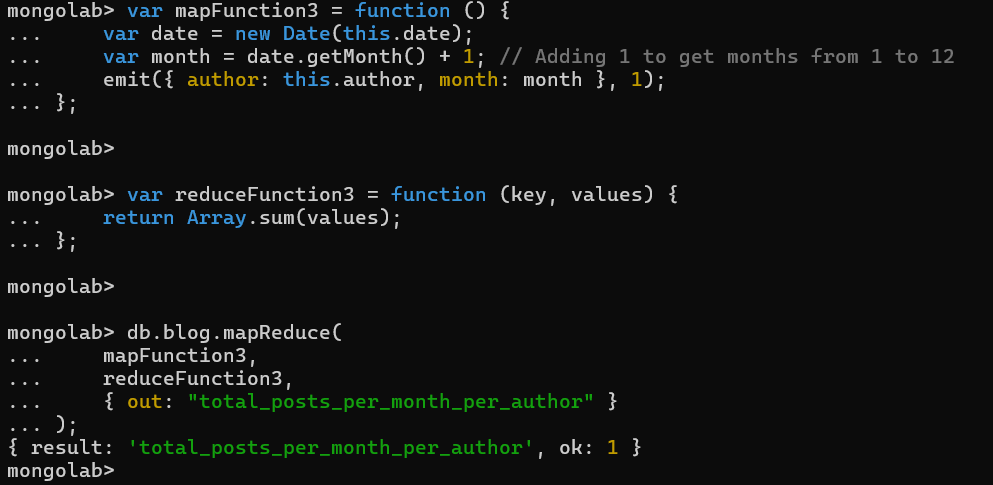
The map function first converts the date field of each document into a JavaScript Date object. Then, it extracts the month from the date object using the getMonth() method, adding 1 to it to get months from 1 to 12 instead of 0 to 11. Finally, it emits key-value pairs where the key is the month (month) and the value is the number of likes (this.likes) associated with the post.

The reduce function receives the key (month) and an array of values (likes) associated with that key. In this case, it calculates the total number of likes for each month by summing up the values (likes) using Array.sum().

**Result**

****

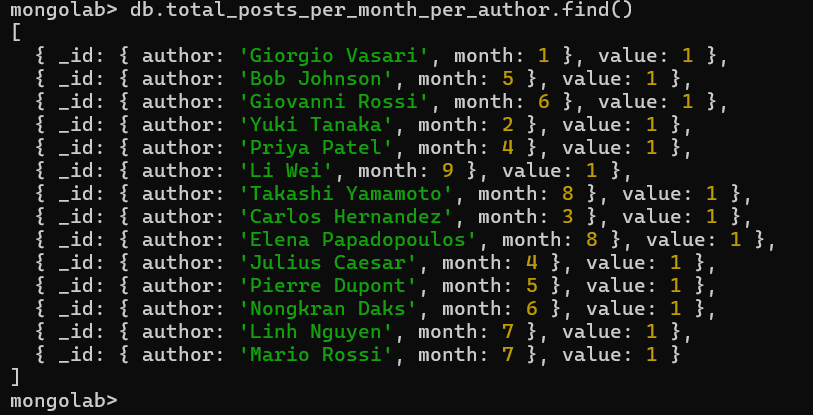
**3. (optional) The total number of posts per month by each author.**

****

The map function converts the date field of each document into a JavaScript Date object. Then, it extracts the month from the date object using the getMonth() method, adding 1 to it to get months from 1 to 12 instead of 0 to 11. After that, it emits key-value pairs where the key is an object containing the author and the month, and the value is 1 (indicating the presence of a post).

The reduce function receives the key (an object containing the author and the month) and an array of values (1 for each post) associated with that key. In this case, it calculates the total number of posts for each author in each month by summing up the values using Array.sum().

**Result**

****