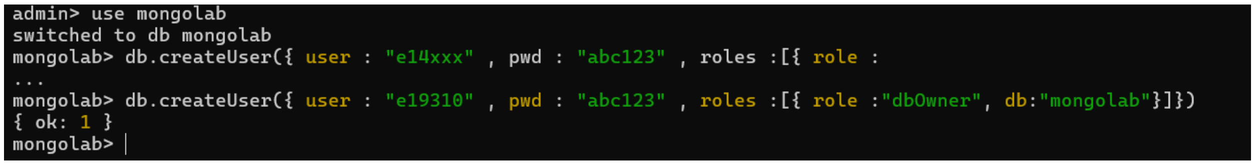
CO527: Advanced Database Systems
Lab 06 - Introduction to NoSQL databases using MongoDB

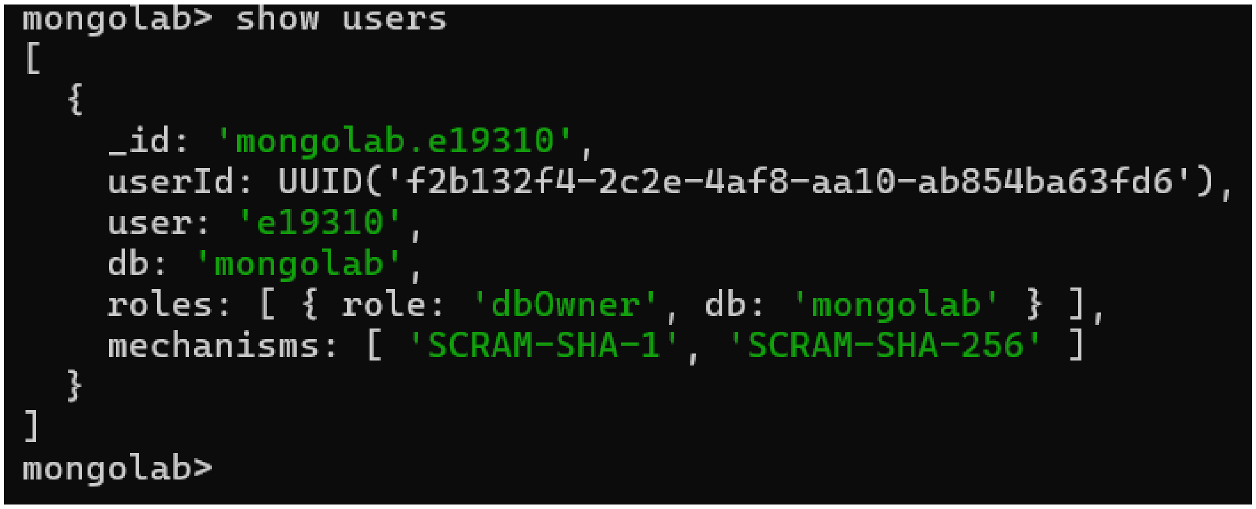
Ranage R.D.P.R.
E/19/310

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.

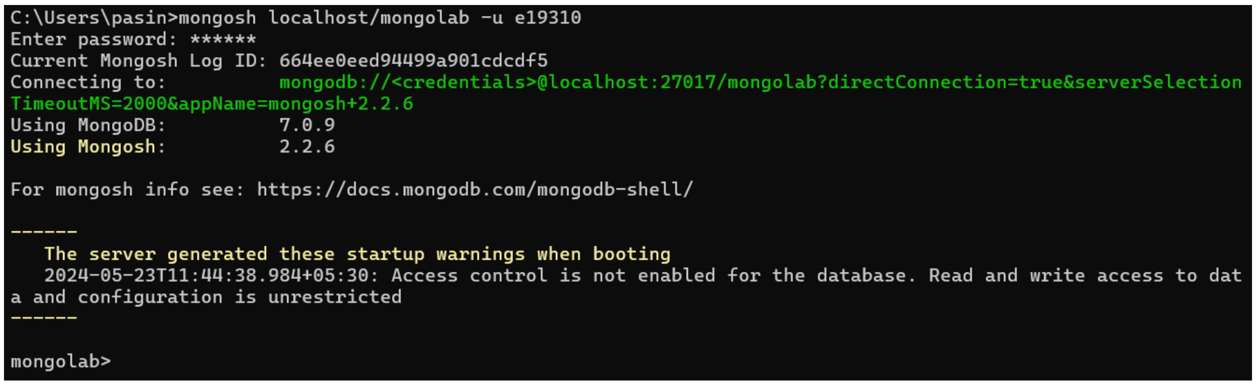


```
admin> use mongolab
switched to db mongolab
mongolab> db.createUser({ user : "e14xxx" , pwd : "abc123" , roles :[{ role :
...
mongolab> db.createUser({ user : "e19310" , pwd : "abc123" , roles :[{ role :"dbOwner", db:"mongolab"}]})
{ ok: 1 }
mongolab>
```



```
mongolab> show users
[
  {
    _id: 'mongolab.e19310',
    userId: UUID('f2b132f4-2c2e-4af8-aa10-ab854ba63fd6'),
    user: 'e19310',
    db: 'mongolab',
    roles: [ { role: 'dbOwner', db: 'mongolab' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
]
mongolab>
```

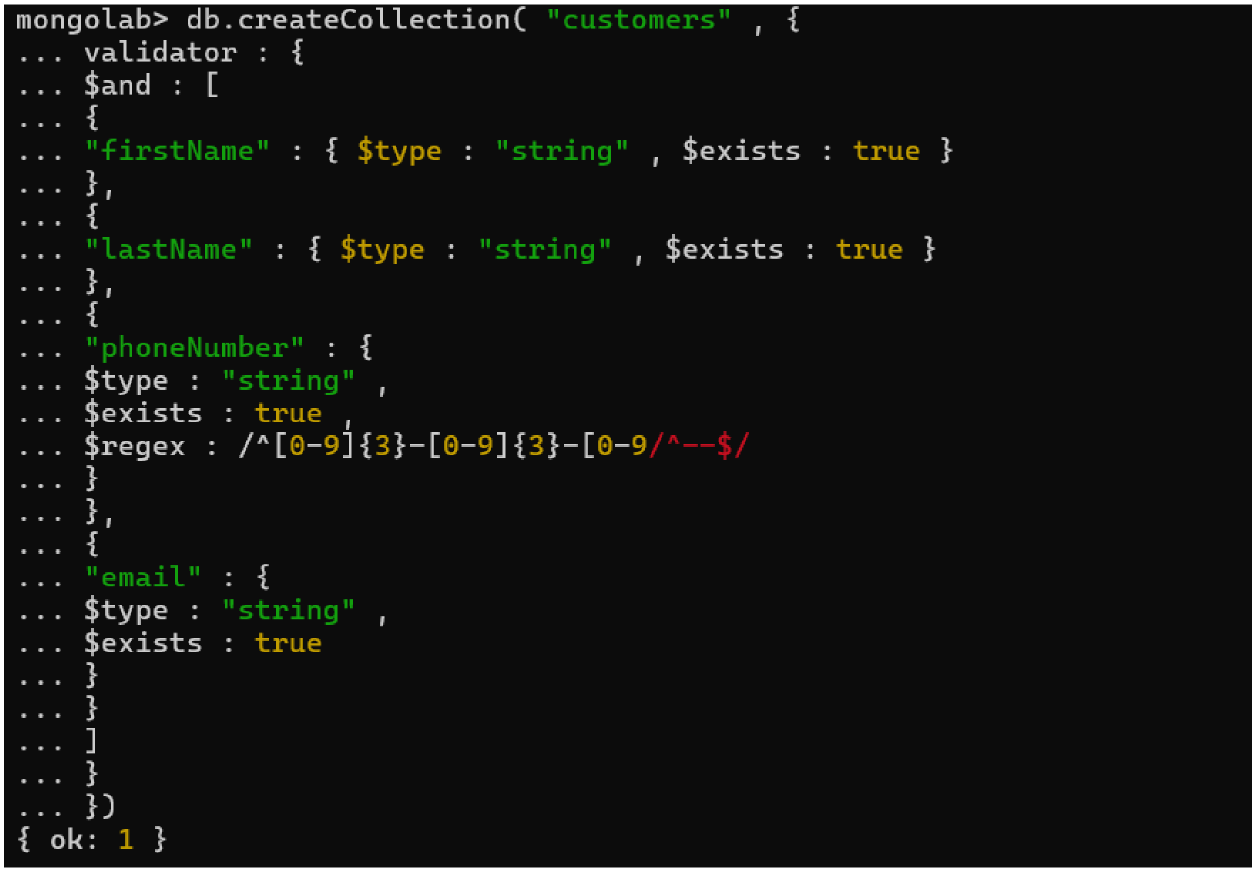
4. Log out of the mongo shell and log back in using the user you created.



```
C:\Users\pasin>mongosh localhost/mongolab -u e19310
Enter password: ******
Current Mongosh Log ID: 664ee0eed94499a901cdcdf5
Connecting to:          mongodb://<credentials>@localhost:27017/mongolab?directConnection=true&serverSelection
TimeoutMS=2000&appName=mongosh+2.2.6
Using MongoDB:          7.0.9
Using Mongosh:          2.2.6

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-05-23T11:44:38.984+05:30: Access control is not enabled for the database. Read and write access to dat
a and configuration is unrestricted
------

mongolab>
```

Exercises

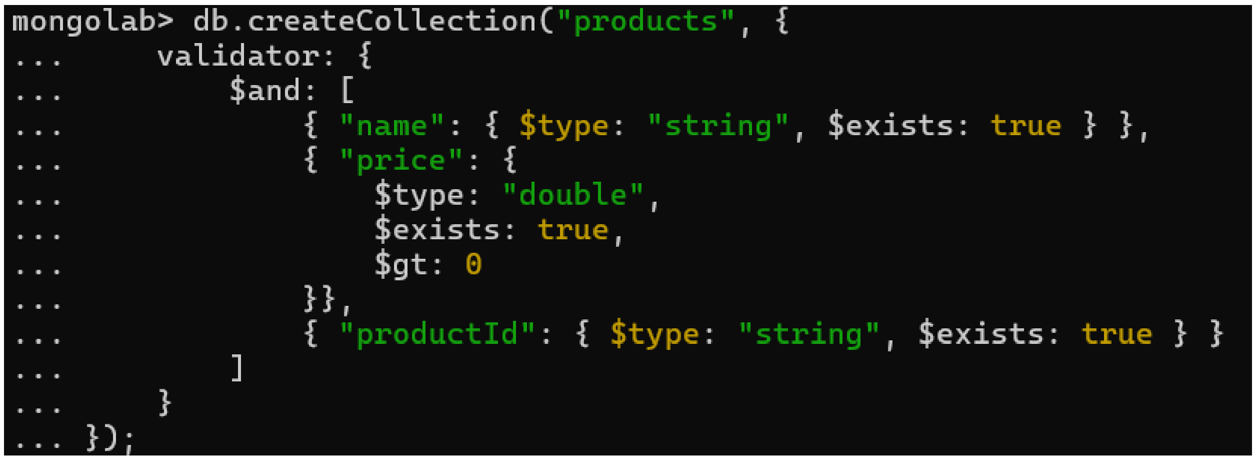
Exercise 1: Data Validation
Creating the customer collection with a custom data validation function



```
mongolab> db.createCollection( "customers" , {
... validator : {
... $and : [
... {
... "firstName" : { $type : "string" , $exists : true }
... },
... {
... "lastName" : { $type : "string" , $exists : true }
... },
... {
... "phoneNumber" : {
... $type : "string" ,
... $exists : true ,
... $regex : /^[0-9]{3}-[0-9]{3}-[0-9/^--$/
... }
... },
... {
... "email" : {
... $type : "string" ,
... $exists : true
... }
... }
... ]
... }
... })
{ ok: 1 }
```

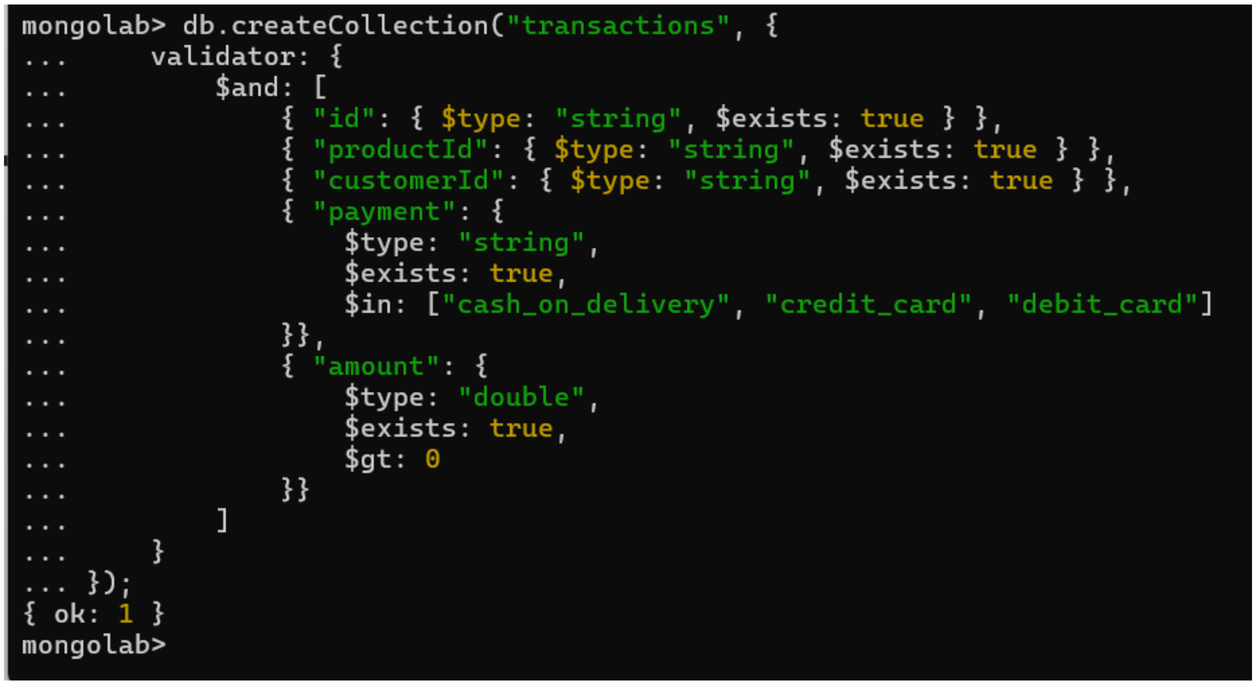
Write code to create collections for Product and Transaction implementing the following validation rules.

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



```
mongolab> db.createCollection("products", {
...     validator: {
...         $and: [
...             { "name": { $type: "string", $exists: true } },
...             { "price": {
...                 $type: "double",
...                 $exists: true,
...                 $gt: 0
...             }},
...             { "productId": { $type: "string", $exists: true } }
...         ]
...     }
... });
```

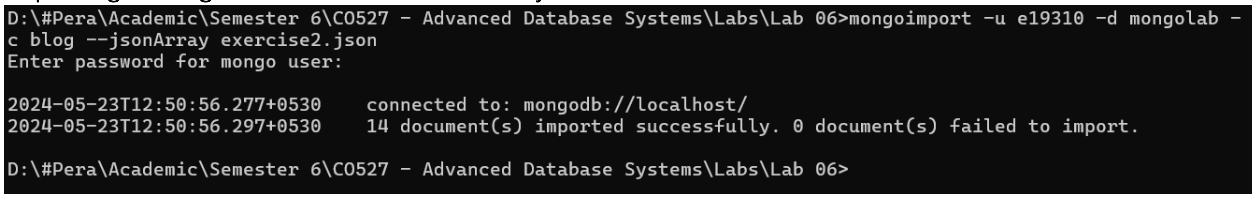
2. Transaction: all fields are required and must have the proper type. Payment should be one of “cash_on_delivery”, “credit_card” or “debit_card”. Amount must be greater than 0.



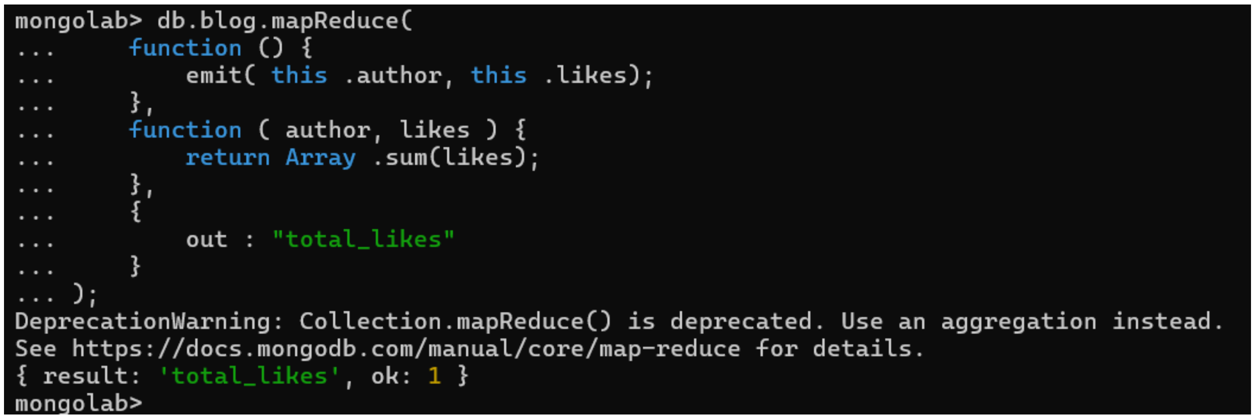
```
mongolab> db.createCollection("transactions", {
...     validator: {
...         $and: [
...             { "id": { $type: "string", $exists: true } },
...             { "productId": { $type: "string", $exists: true } },
...             { "customerId": { $type: "string", $exists: true } },
...             { "payment": {
...                 $type: "string",
...                 $exists: true,
...                 $in: ["cash_on_delivery", "credit_card", "debit_card"]
...             }},
...             { "amount": {
...                 $type: "double",
...                 $exists: true,
...                 $gt: 0
...             }}
...         ]
...     }
... });
{ ok: 1 }
mongolab>
```

Exercise 2: Aggregation & Map-Reduce

Importing some generated data as exercise2.json



```
D:\#Pera\Academic\Semester 6\CO527 - Advanced Database Systems\Labs\Lab 06>mongoimport -u e19310 -d mongolab -
c blog --jsonArray exercise2.json
Enter password for mongo user:

2024-05-23T12:50:56.277+0530    connected to: mongodb://localhost/
2024-05-23T12:50:56.297+0530    14 document(s) imported successfully. 0 document(s) failed to import.

D:\#Pera\Academic\Semester 6\CO527 - Advanced Database Systems\Labs\Lab 06>
```

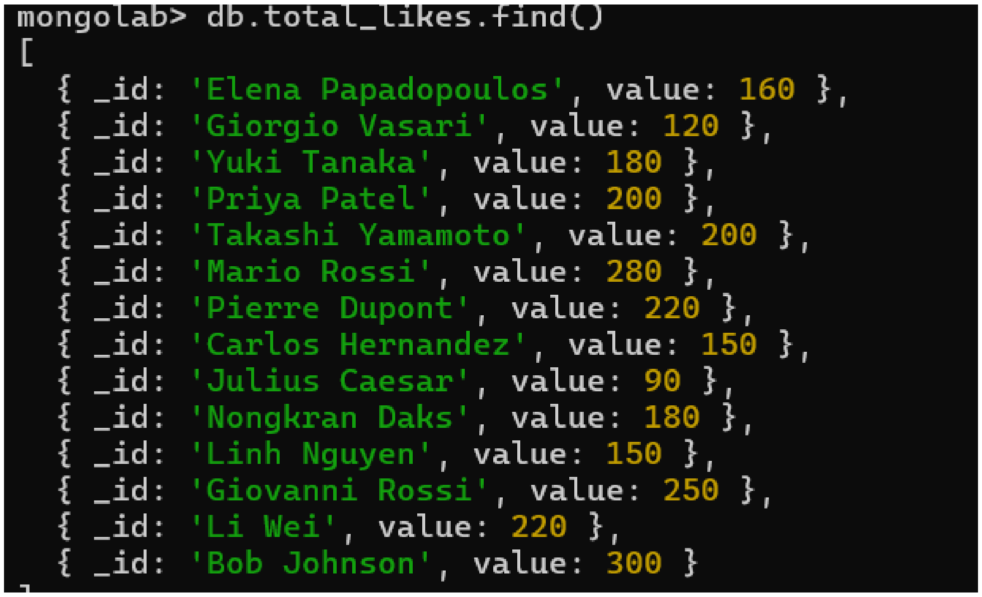


```
mongolab> db.blog.mapReduce(
...     function () {
...         emit( this .author, this .likes);
...     },
...     function ( author, likes ) {
...         return Array .sum(likes);
...     },
...     {
...         out : "total_likes"
...     }
... );
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'total_likes', ok: 1 }
mongolab>
```

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.

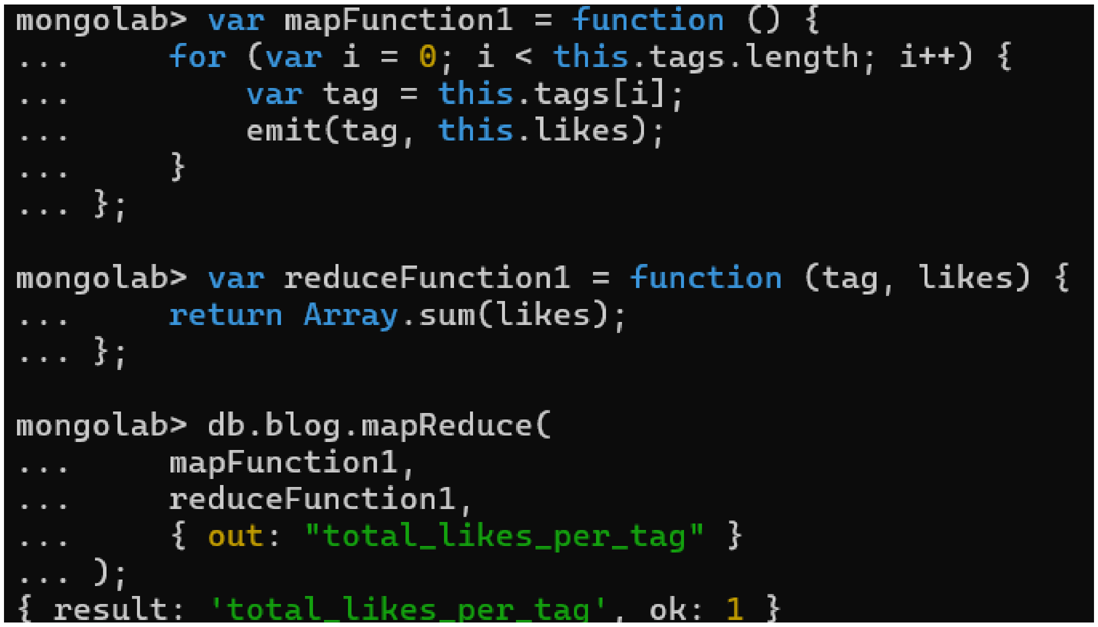
The reduce function is invoked for each unique key emitted by the map function. It receives the key (author) and an array of values (likes) associated with that key. In this case, it calculates the total number of likes for each author by summing up the values (likes) using `Array.sum()`.

Result



```
mongolab> db.total_likes.find()
[
  { _id: 'Elena Papadopoulos', value: 160 },
  { _id: 'Giorgio Vasari', value: 120 },
  { _id: 'Yuki Tanaka', value: 180 },
  { _id: 'Priya Patel', value: 200 },
  { _id: 'Takashi Yamamoto', value: 200 },
  { _id: 'Mario Rossi', value: 280 },
  { _id: 'Pierre Dupont', value: 220 },
  { _id: 'Carlos Hernandez', value: 150 },
  { _id: 'Julius Caesar', value: 90 },
  { _id: 'Nongkran Daks', value: 180 },
  { _id: 'Linh Nguyen', value: 150 },
  { _id: 'Giovanni Rossi', value: 250 },
  { _id: 'Li Wei', value: 220 },
  { _id: 'Bob Johnson', value: 300 }
```

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.

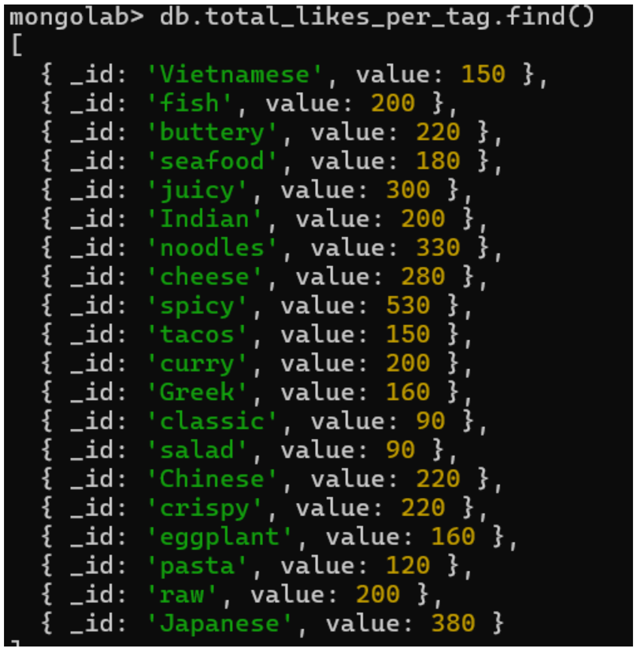


```
mongolab> var mapFunction1 = function () {
...     for (var i = 0; i < this.tags.length; i++) {
...         var tag = this.tags[i];
...         emit(tag, this.likes);
...     }
... };

mongolab> var reduceFunction1 = function (tag, likes) {
...     return Array.sum(likes);
... };

mongolab> db.blog.mapReduce(
...     mapFunction1,
...     reduceFunction1,
...     { out: "total_likes_per_tag" }
... );
{ result: 'total_likes_per_tag', ok: 1 }
```

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.

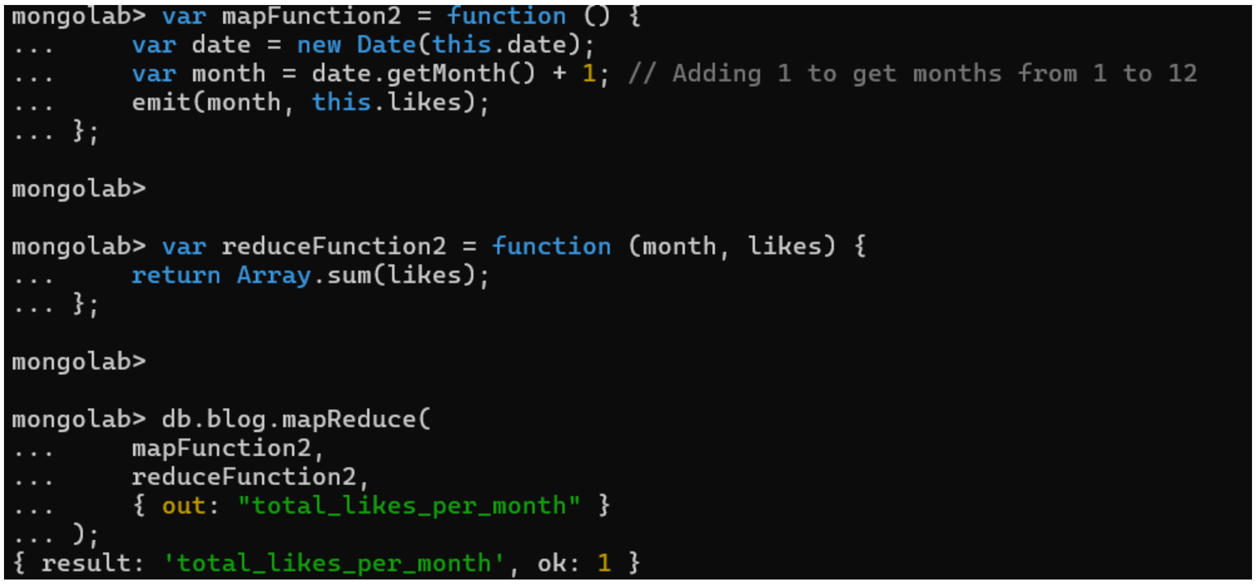
The reduce function is invoked for each unique key emitted by the map function. It receives the key (tag) and an array of values (likes) associated with that key. In this case, it calculates the total number of likes for each tag by summing up the values (likes) using `Array.sum()`.

Result



```
mongolab> db.total_likes_per_tag.find()
[
  { _id: 'Vietnamese', value: 150 },
  { _id: 'fish', value: 200 },
  { _id: 'buttery', value: 220 },
  { _id: 'seafood', value: 180 },
  { _id: 'juicy', value: 300 },
  { _id: 'Indian', value: 200 },
  { _id: 'noodles', value: 330 },
  { _id: 'cheese', value: 280 },
  { _id: 'spicy', value: 530 },
  { _id: 'tacos', value: 150 },
  { _id: 'curry', value: 200 },
  { _id: 'Greek', value: 160 },
  { _id: 'classic', value: 90 },
  { _id: 'salad', value: 90 },
  { _id: 'Chinese', value: 220 },
  { _id: 'crispy', value: 220 },
  { _id: 'eggplant', value: 160 },
  { _id: 'pasta', value: 120 },
  { _id: 'raw', value: 200 },
  { _id: 'Japanese', value: 380 }
```

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.)

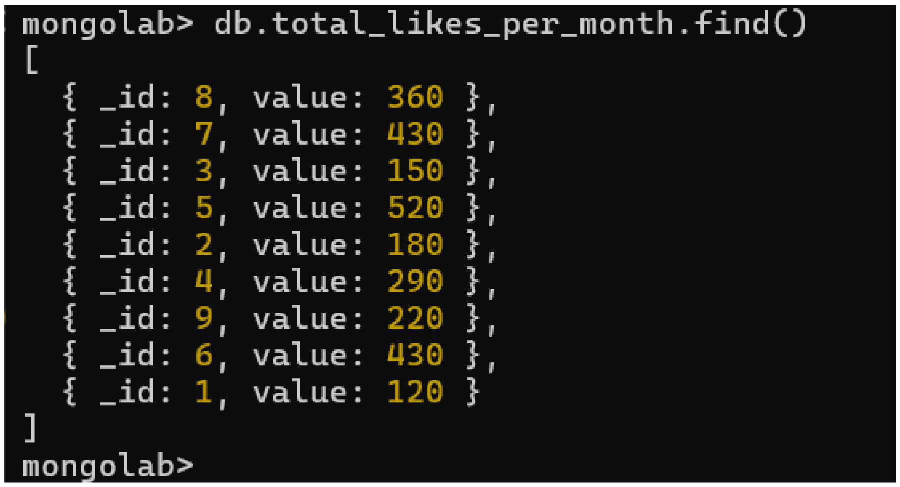


```
mongolab> var mapFunction2 = function () {
...     var date = new Date(this.date);
...     var month = date.getMonth() + 1; // Adding 1 to get months from 1 to 12
...     emit(month, this.likes);
... };

mongolab>

mongolab> var reduceFunction2 = function (month, likes) {
...     return Array.sum(likes);
... };

mongolab>

mongolab> db.blog.mapReduce(
...     mapFunction2,
...     reduceFunction2,
...     { out: "total_likes_per_month" }
... );
{ result: 'total_likes_per_month', ok: 1 }
```

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.

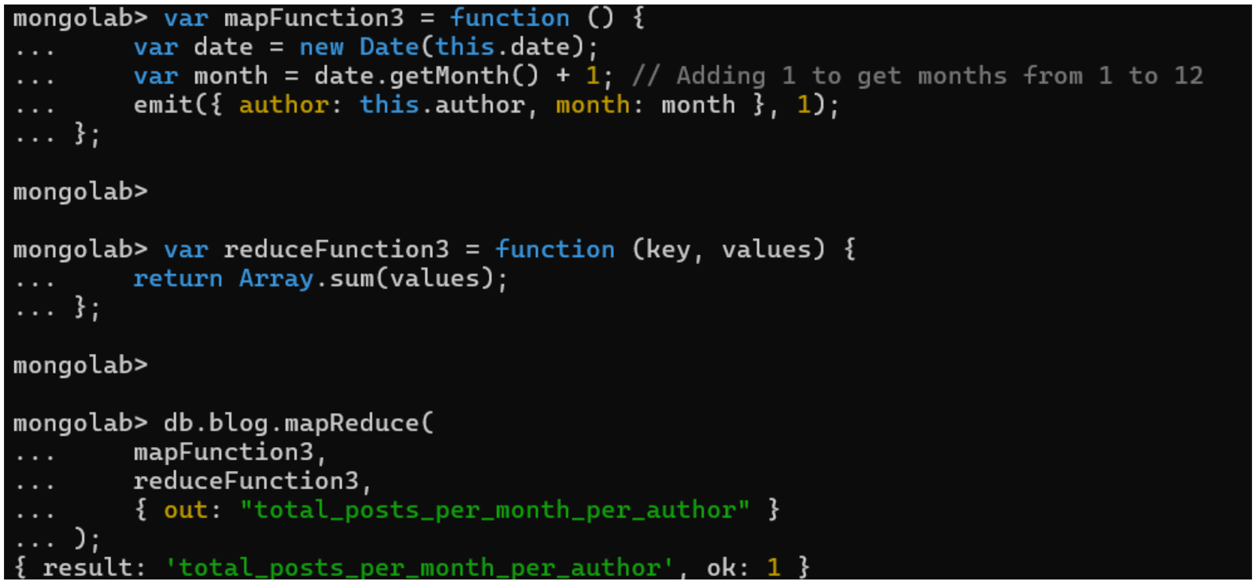
The reduce function receives the key (month) and an array of values (likes) associated with that key. In this case, it calculates the total number of likes for each month by summing up the values (likes) using `Array.sum()`.

Result



```
mongolab> db.total_likes_per_month.find()
[
  { _id: 8, value: 360 },
  { _id: 7, value: 430 },
  { _id: 3, value: 150 },
  { _id: 5, value: 520 },
  { _id: 2, value: 180 },
  { _id: 4, value: 290 },
  { _id: 9, value: 220 },
  { _id: 6, value: 430 },
  { _id: 1, value: 120 }
]
mongolab>
```

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

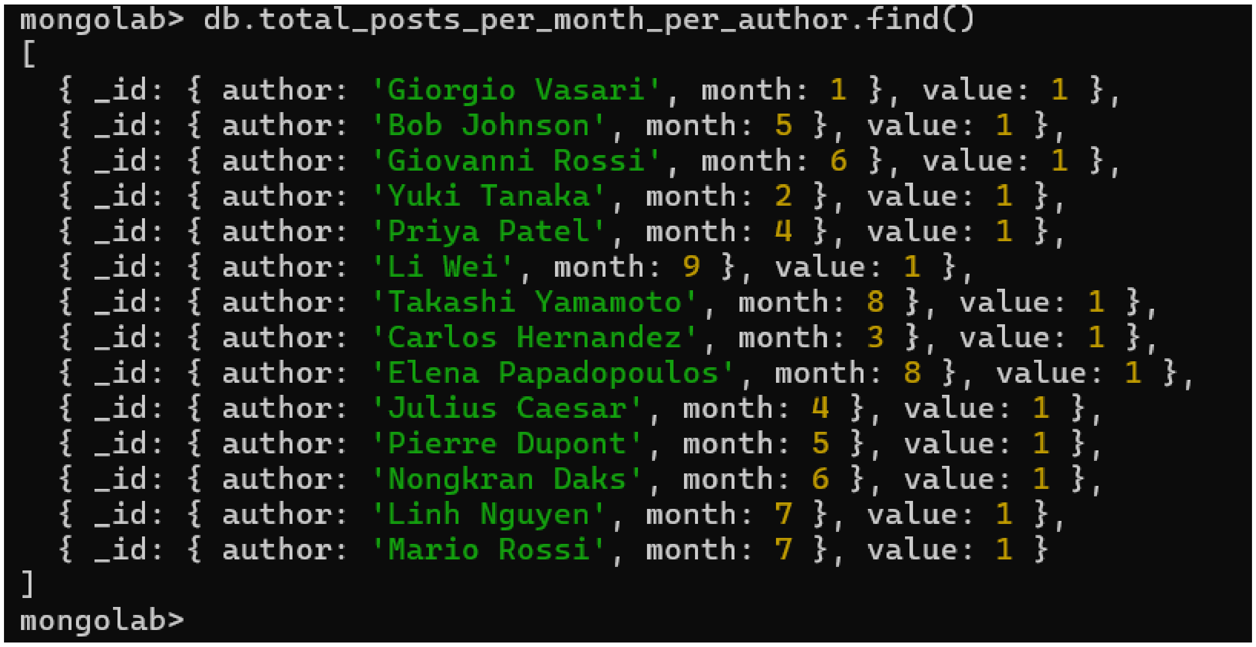


```
mongolab> var mapFunction3 = function () {
...     var date = new Date(this.date);
...     var month = date.getMonth() + 1; // Adding 1 to get months from 1 to 12
...     emit({ author: this.author, month: month }, 1);
... };

mongolab>

mongolab> var reduceFunction3 = function (key, values) {
...     return Array.sum(values);
... };

mongolab>

mongolab> db.blog.mapReduce(
...     mapFunction3,
...     reduceFunction3,
...     { out: "total_posts_per_month_per_author" }
... );
{ result: 'total_posts_per_month_per_author', ok: 1 }
```

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



```
mongolab> db.total_posts_per_month_per_author.find()
[
  { _id: { author: 'Giorgio Vasari', month: 1 }, value: 1 },
  { _id: { author: 'Bob Johnson', month: 5 }, value: 1 },
  { _id: { author: 'Giovanni Rossi', month: 6 }, value: 1 },
  { _id: { author: 'Yuki Tanaka', month: 2 }, value: 1 },
  { _id: { author: 'Priya Patel', month: 4 }, value: 1 },
  { _id: { author: 'Li Wei', month: 9 }, value: 1 },
  { _id: { author: 'Takashi Yamamoto', month: 8 }, value: 1 },
  { _id: { author: 'Carlos Hernandez', month: 3 }, value: 1 },
  { _id: { author: 'Elena Papadopoulos', month: 8 }, value: 1 },
  { _id: { author: 'Julius Caesar', month: 4 }, value: 1 },
  { _id: { author: 'Pierre Dupont', month: 5 }, value: 1 },
  { _id: { author: 'Nongkran Daks', month: 6 }, value: 1 },
  { _id: { author: 'Linh Nguyen', month: 7 }, value: 1 },
  { _id: { author: 'Mario Rossi', month: 7 }, value: 1 }
]
mongolab>
```