W9 Activity: Create / Insert / Update MongoDB

- Due Sunday by 11:59pm
- Points 24

Directions

For each of the following exercises, submit:

- The **mongosh query command** you used
  - (a formatted code block would be neat, but however you need to format it is fine)
- A screenshot of your terminal window that shows **clearly** what the response from your DB connection was

Exercise 1: Create Database

The `use` keyword allows us to either *switch to* or *create* a database

- Create a new database named `chirper` with the following command:
  - `use chirper`

Exercise 2: Create Collection

We can create a new collection one of two ways, this one is less accident prone.

- Use the `createCollection` function to create a new collection:
  - `db.createCollection("posts")`

Exercise 3: Insert into collection

Now we have a `posts` collection with no data in it. Let's put a post in there.

```
db.posts.insertOne({
    text: "Wow this is such a good microblog",
    category: "tech",
    likes: 0,
    date: Date()
})
```

Exercise 4: Insert Many

We can do a batch insert if we use the `insertMany` function.

- Use the `db.posts.insertMany([])` function to insert at least 3 posts.
- You can use whatever object formatting you like, but should include the fields in the following:

```
db.posts.insertMany([
  {
    text: "This morning a miracle happened as promised; the rising of the world's closest star.",
    category: "news",
    likes: 2,
    date: Date()
  },
  {
    text: "The almanacs warned us that the fast coming weather might blow us away like dandelion flo
wers",
    category: "events",
    likes: 3,
    date: Date()
  },
  {
    text: "I've been trying not to think before my third cup of coffee."
    category: "personal",
    likes: 4,
    date: Date()
  }
])
```

Exercise 5: Create on Insert

Once we have set the database we are working under, we have to dot notation to access all of the sub-groups of objects within the database.

If we use dot notation to perform an action on a collection that does not yet exist, the database will first create a collection with that name and then insert the data passed into it as the first thing in the collection.

- Use the `insertOne` function to both create a new collection, and insert a piece of data into the collection like so:
  - `db.users.insertOne({username:"lilfrog", displayname: "Toadlet"})`
  - You can use whatever you want for the username and display name inside the double quotes
- We can use `show collections` on mongosh to get a list of all of the collections so far. Type in `show collections` to show that your second collection was successfully initiated.

Exercise 6: Update One

The `updateOne` method will update the first matching object from a query.

- Use the `db.users.find( {username : <...> })` to show the information for the user you inserted on exercise 5.
- We're going to pass two things to the updateOne function:
  - An object used to query, the first object
    - `{username: <...>}`
  - An object used to target, and then update, a particular field.
    - `{$set : { field : value }}`
    - This object uses the `$set` operator, which signals a particular process to update

- Use `db.posts.updateOne( { username: "lilfrog" }, {$set : {displayname : "Pumpkin Toadlet"}} )`

Exercise 7: Upsert

We can pass in an object to signal to mongodb to both either update an existing document, or if the document doesn't already exist, to create one.

- We we're going to use 3 objects:
  - An object used to query, the first object
    - `{ category : "comedy"}`
  - An object to target, and then update, a particular set of fields
    - `{ $set: { field: ...} }`
  - An object used to signal the "update if doesn't exist", aka upsert.
    - `{ upsert : true}`

```
db.posts.updateOne(
  { category : "comedy"},
  {
    $set:
      {
        text: "3 NoSQL databases walk into a bar. They leave, because there's nowhere to sit. They c
ouldn't find a table."
        category: "comedy",
        likes: 5,
        date: Date()
      }
  },
  { upsert: true }
)
```

Exercise 8: Update Many

Remember that we can query *all* documents in a collection using `db.collectionname.find()`.
Because the update function is more critical, we have to be specific on the number to make a change to.

We also have to use specific **operators** , starting with a `$`, in order to perform certain specific update features.

- We can update all of the posts likes by one unity using the `$inc` update operator
- To the first parameter, we're going to pass in the *empty document*, `{}`, to return all documents in posts.
- To the second parameter we're going pass in an object that has the update operator, and then as the value we will pass in the field and the amount to increment.
  - `{ $inc : { likes : 1 } }`
- Use the following:
  - `db.posts.updateMany({}, { $inc: { likes : 1 } } )`