Portfolio Week 8

Task 1:

Example 1:

```
// Define a new record
const rec1 = new personDoc(
{
    name: 'Jacky',
    age: 36,
    gender: 'Male',
    salary: 3456
});
```

```
_id: ObjectId('673a09ca74966bab7126cc24')
name : "Jacky"
age : 36
gender : "Male"
salary : 3456
__v : 0
```

Example 2:

```
// Define a new record
const rec1 = new personDoc(
{
    name: 'Frodo',
    age: 21,
    gender: 'Male',
    salary: 2500
});
```

```
_id: ObjectId('673b86b5cc88796e811ad5b6')
name : "Frodo"
age : 21
gender : "Male"
salary : 2500
__v : 0
```

Example 3:

```
// Define a new record
const rec1 = new personDoc(
{
    name: 'Arwen',
    age: 45,
    gender: 'Female',
    salary: 6780
});
```

```
_id: ObjectId('673b871e85bf8006e523acd9')
name : "Arwen"
age : 45
gender : "Female"
salary : 6780
__v : 0
```

Explanation:

I created three new records within my collection using the Mongoose module. ‘doc1’ is an instance of the model I created. It represents the record that is to be saved into the database. ‘.save’ is an asynchronous method which saves the record to the MongoDB database and returns a promise object. Once the record is saved, ‘.then’ executes. However, if an error occurs, ‘.catch’ is executed.

Task 2:

```
const manyPeople =
[
    {name: 'Gandalf', age: 70, gender: 'Male', salary: 3500},
    {name: 'Sam', age: 23, gender: 'Male', salary: 2900},
    {name: 'Aragorn', age: 50, gender: 'Male', salary: 8600},
    {name: 'Rose', age: 25, gender: 'Female', salary: 4600},
    {name: 'Eowyn', age: 33, gender: 'Female', salary: 6350}
];
```

```
_id: ObjectId('673b8c8d8b1386f37cb6222f')
name : "Gandalf"
age : 70
gender : "Male"
salary : 3500
__v : 0

_id: ObjectId('673b8c8d8b1386f37cb62230')
name : "Sam"
age : 23
gender : "Male"
salary : 2900
__v : 0

_id: ObjectId('673b8c8d8b1386f37cb62231')
name : "Aragorn"
age : 50
gender : "Male"
salary : 8600
__v : 0

_id: ObjectId('673b8c8d8b1386f37cb62233')
name : "Eowyn"
age : 33
gender : "Female"
salary : 6350
__v : 0

_id: ObjectId('673b8c8d8b1386f37cb62232')
name : "Rose"
age : 25
gender : "Female"
salary : 4600
__v : 0
```

Explanation:

I used an array to hold multiple JSON records, which were then inserted into the database using the ‘.insertMany’ method. The array was passed as an argument to this method, which returned a promise object. ‘.then’ handled the successful execution, whilst ‘.catch’ handled any unsuccessful execution.

Task 3:

Output:

```
Displaying the requested documents:
Frodo 21 2500
Sam 23 2900
Jacky 36 3456
Gandalf 70 3500
Rose 25 4600
Eowyn 33 6350
Arwen 45 6780
Aragorn 50 8600
```

Explanation:

I used the '.sort' method to sort salary by ascending order, and used the '.select' method to display the name, age, and salary fields. The '.limit' method was used to limit the number of records printed to the console. The '.exec' method then executed the query, which was followed by the '.then' and '.catch' methods.

Task 4:

Output:

Example 1:

```
Displaying Females with age greater than or equal to 30
Arwen 45
Eowyn 33
```

Example 2:

```
Displaying Females with age greater than or equal to 40
Arwen 45
```

Explanation:

Similar to task 3 except filtering was used within the '.find' method. I filtered by female gender and age, sorted by name, displayed only name and age, and limited displayed documents to 10.

Task 5:

Output:

```
Total document count: 8
```

Explanation:

I used the '.countDocuments' method to count the total number of documents and print them to the console. After '.exec', the promise object is returned and either '.then' or '.catch' is executed.

Task 6:

Output:

```
Deleted documents: { acknowledged: true, deletedCount: 6 }
```

Explanation:

I deleted 6 documents using the '.deleteMany' method. I filtered the delete query with ages greater than or equal to 25. In the '.then' method, the total count is printed to the console.

Task 7:

Output:

```
Update
{
  acknowledged: true,
  modifiedCount: 3,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 3
}
```

Explanation:

I updated 3 records of gender female with a salary of 5555 using the '.updateMany' method. The '.then' method prints the results to the console.

Reflection:

Overall, I learned how to connect a part of the backend to a MongoDB database. This was done using Mongoose which provided various methods for defining a connection, schema, schema wrapper and interface (model), and CRUD operations. I also learned of the importance of the promise object that is returned during the execution of a query/command; and of the success and failure of handling asynchronous operations.