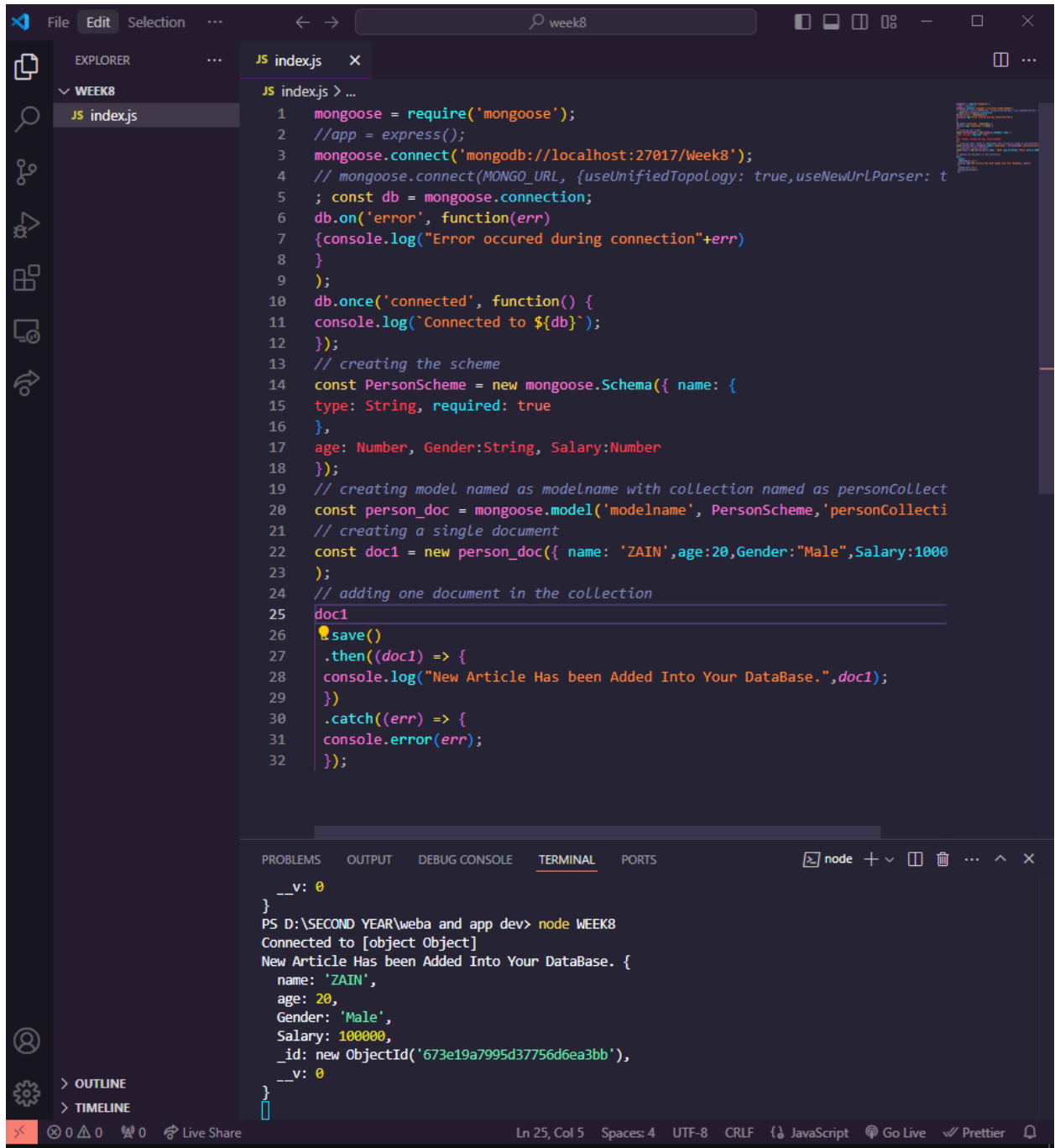


## Week 8



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'WEEK8' with a file named 'index.js'. The main editor area displays the content of 'index.js', which is a Node.js script using Mongoose to connect to a MongoDB database, create a schema, and save a document. The script includes comments and uses ES6 syntax like arrow functions and destructuring. The terminal at the bottom shows the execution of the script, with output messages indicating a successful connection and document saving.

```
JS index.js > ...
1  mongoose = require('mongoose');
2  //app = express();
3  mongoose.connect('mongodb://localhost:27017/Week8');
4  // mongoose.connect(MONGO_URL, {useUnifiedTopology: true, useNewUrlParser: t
5  ; const db = mongoose.connection;
6  db.on('error', function(err)
7  {console.log("Error occurred during connection"+err)
8  }
9  );
10 db.once('connected', function() {
11 console.log(`Connected to ${db}`);
12 });
13 // creating the scheme
14 const PersonScheme = new mongoose.Schema({ name: {
15 type: String, required: true
16 },
17 age: Number, Gender:String, Salary:Number
18 });
19 // creating model named as modelname with collection named as personCollect
20 const person_doc = mongoose.model('modelname', PersonScheme,'personCollecti
21 // creating a single document
22 const doc1 = new person_doc({ name: 'ZAIN',age:20,Gender:"Male",Salary:1000
23 });
24 // adding one document in the collection
25 doc1
26 save()
27 .then((doc1) => {
28 console.log("New Article Has been Added Into Your DataBase.",doc1);
29 })
30 .catch((err) => {
31 console.error(err);
32 });
```

Terminal Output:

```
__v: 0
}
PS D:\SECOND YEAR\weba and app dev> node WEEK8
Connected to [object Object]
New Article Has been Added Into Your DataBase. {
  name: 'ZAIN',
  age: 20,
  Gender: 'Male',
  Salary: 100000,
  _id: new ObjectId('673e19a7995d37756d6ea3bb'),
  __v: 0
}
```

## Task 1

: In your portfolio for this week, explain the purpose of `doc1.save` and `.then` in the code. It is used to handle promises, ensuring that the database connection or any asynchronous operation is successfully completed before executing the subsequent logic. This helps manage flow control and handle results or errors effectively.

`Doc.save` function is a mongoose method used to save a document into a connected MongoDB collection. It takes the `doc1` OBJ and attempts to save it in the `personCollection` in the database and if that is successful it returns a promise that resolves with the saved document.

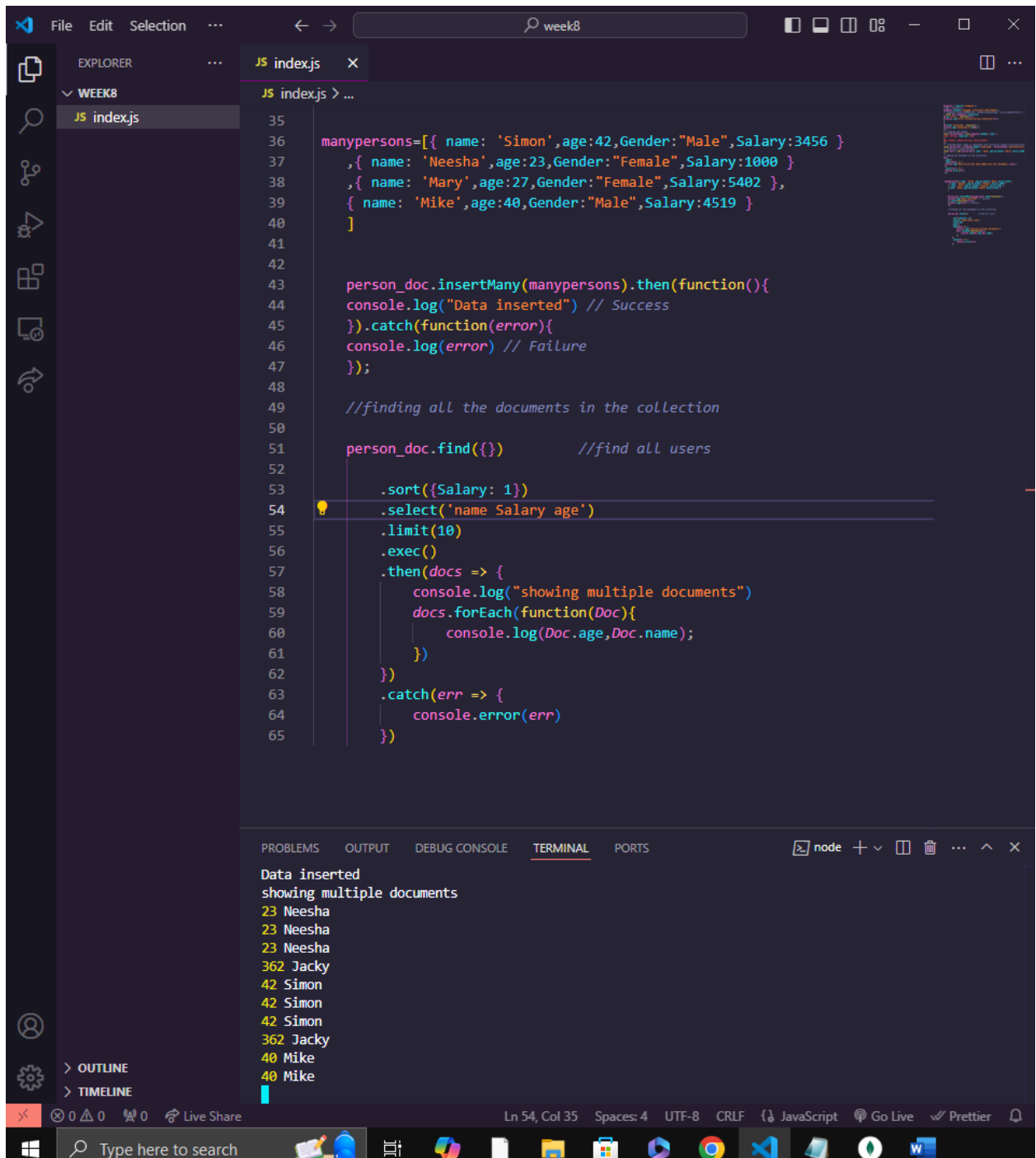
The `.then()` method is used to handle the successful completion of asynchronous operation initiated by `doc1.save()`. The subsequent logic executes only when the operation is successful. This allows you to access the result of the operation and perform follow up actions such as logging, triggering or updating the UI

## Task 2

The code that creates multiple documents is `insertMany()` which is a mongoose method used to insert multiple documents in one operation which is good for large datasets.

It works by taking an array of objects where each object represents a document to be inserted then once run returns a promise that resolves when all documents are successfully inserted or rejects if an error occurs

## Task 3



```
File Edit Selection ... week8
EXPLORER
WEEK8
JS index.js
JS index.js > ...
35
36 manypersons=[{ name: 'Simon',age:42,Gender:"Male",Salary:3456 }
37 ,{ name: 'Neesha',age:23,Gender:"Female",Salary:1000 }
38 ,{ name: 'Mary',age:27,Gender:"Female",Salary:5402 },
39 { name: 'Mike',age:40,Gender:"Male",Salary:4519 }
40 ]
41
42
43 person_doc.insertMany(manypersons).then(function(){
44 console.log("Data inserted") // Success
45 }).catch(function(error){
46 console.log(error) // Failure
47 });
48
49 //finding all the documents in the collection
50
51 person_doc.find({}) //find all users
52
53 .sort({Salary: 1})
54 .select('name Salary age')
55 .limit(10)
56 .exec()
57 .then(docs => {
58 console.log("showing multiple documents")
59 docs.forEach(function(Doc){
60 console.log(Doc.age,Doc.name);
61 })
62 })
63 .catch(err => {
64 console.error(err)
65 })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
node
Data inserted
showing multiple documents
23 Neesha
23 Neesha
23 Neesha
362 Jacky
42 Simon
42 Simon
42 Simon
362 Jacky
40 Mike
40 Mike
```

Ln 54, Col 35 Spaces: 4 UTF-8 CRLF JavaScript Go Live Prettier

## Task 4

The screenshot shows a VS Code editor window with a file named `index.js` open. The code is written in JavaScript and uses Mongoose to interact with a database. The code is as follows:

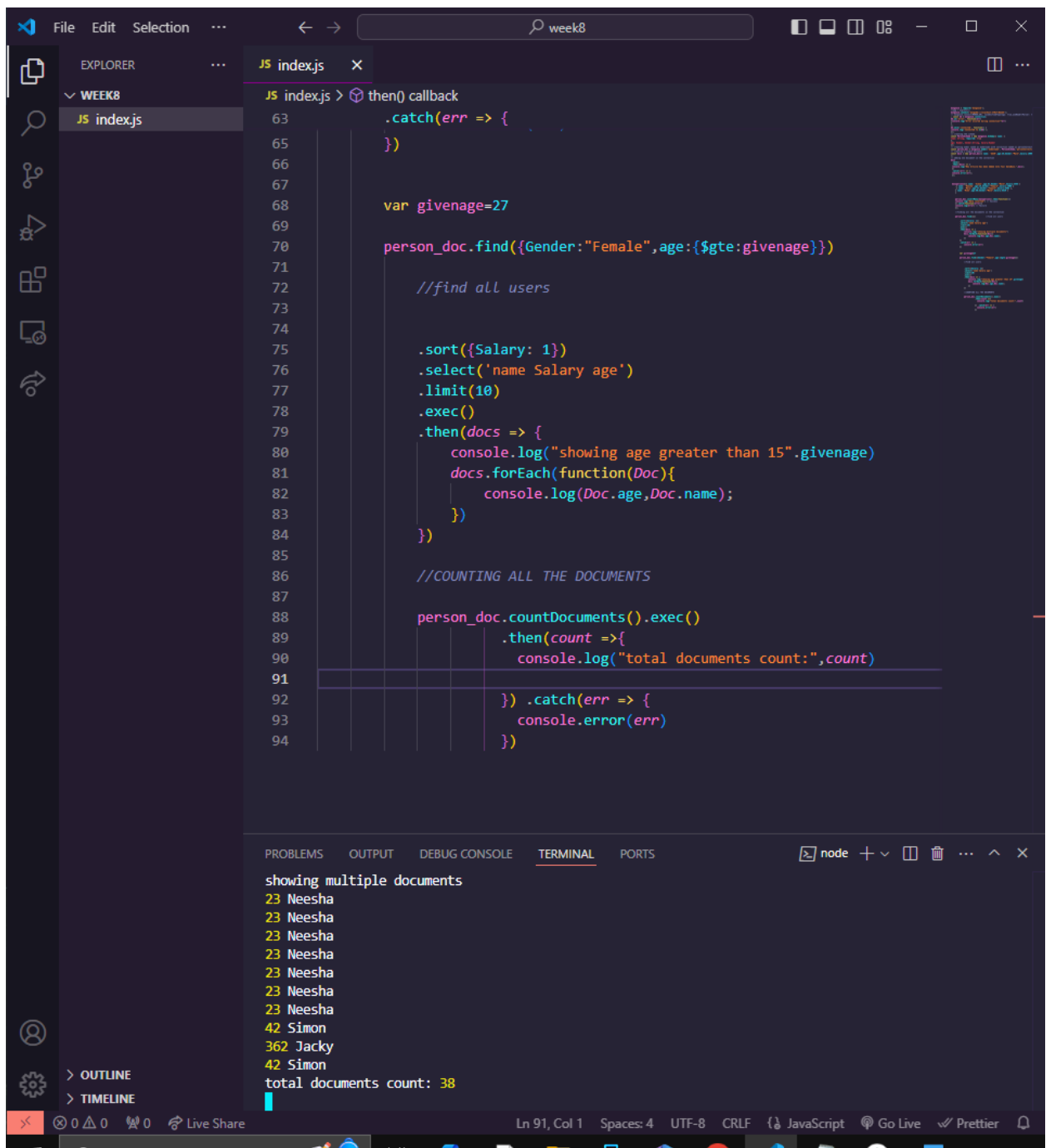
```
55 .limit(10)
56 .exec()
57 .then(docs => {
58   console.log("showing multiple documents")
59   docs.forEach(function(Doc){
60     console.log(Doc.age, Doc.name);
61   })
62 })
63 .catch(err => {
64   console.error(err)
65 })
66
67
68 var givenage=27
69
70 person_doc.find({Gender:"Female",age:{$gte:givenage}})
71
72 //find all users
73
74
75 .sort({Salary: 1})
76 .select('name Salary age')
77 .limit(10)
78 .exec()
79 .then(docs => {
80   console.log("showing age greater than 15".givenage)
81   docs.forEach(function(Doc){
82     console.log(Doc.age, Doc.name);
83   })
84 })
85
86
```

The terminal window at the bottom shows the output of the code:

```
23 Neesha
42 Simon
362 Jacky
42 Simon
42 Simon
undefined
27 Mary
27 Mary
27 Mary
27 Mary
27 Mary
27 Mary
```

The status bar at the bottom indicates the file is `index.js`, line 68, column 24, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax. The editor also shows the Explorer sidebar with the file `index.js` selected under the `WEEK8` folder.

## Task 5



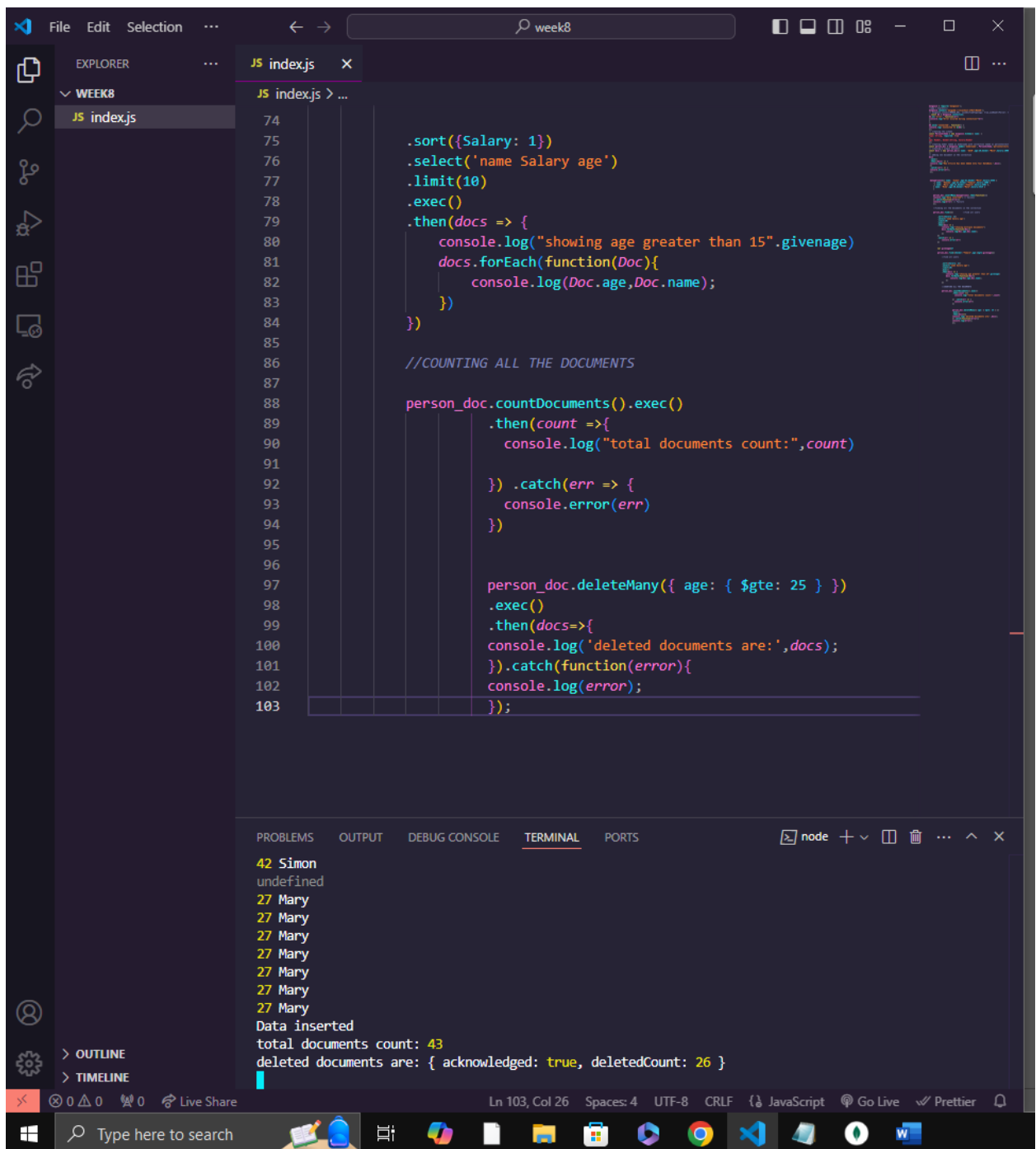
The screenshot shows a Visual Studio Code editor with a file named `index.js` open. The file contains JavaScript code that interacts with a database (likely MongoDB) to find documents and count them. The code is as follows:

```
63     .catch(err => {
64     })
65
66
67
68     var givenage=27
69
70     person_doc.find({Gender:"Female",age:{$gte:givenage}})
71
72     //find all users
73
74
75     .sort({Salary: 1})
76     .select('name Salary age')
77     .limit(10)
78     .exec()
79     .then(docs => {
80         console.log("showing age greater than 15".givenage)
81         docs.forEach(function(Doc){
82             console.log(Doc.age,Doc.name);
83         })
84     })
85
86     //COUNTING ALL THE DOCUMENTS
87
88     person_doc.countDocuments().exec()
89     .then(count =>{
90         console.log("total documents count:",count)
91     })
92     .catch(err => {
93         console.error(err)
94     })
```

The terminal output at the bottom shows the execution results:

```
showing multiple documents
23 Neesha
23 Neesha
23 Neesha
23 Neesha
23 Neesha
23 Neesha
42 Simon
362 Jacky
42 Simon
total documents count: 38
```

## Task 6



## Task 7

```
84      })
85
86      //COUNTING ALL THE DOCUMENTS
87
88      person_doc.countDocuments().exec()
89      .then(count =>{
90          console.log("total documents count:",count)
91      }) .catch(err => {
92          console.error(err)
93      })
94
95
96
97      person_doc.deleteMany({ age: { $gte: 25 } })
98      .exec()
99      .then(docs=>{
100          console.log('deleted documents are:',docs);
101      }).catch(function(error){
102          console.log(error);
103      });
104
105
106      person_doc.updateMany({ Gender: "Female" },{Salay:555
107      .exec()
108      .then(docs=>{
109          console.log("update")
110          console.log(docs); // Success
111      }).catch(function(error){
112          console.log(error); // Failure
113      });
```

23 Neesha  
23 Neesha  
23 Neesha  
23 Neesha  
23 Neesha  
23 Neesha  
20 ZAIN  
20 ZAIN  
undefined  
Data inserted  
total documents count: 22  
deleted documents are: { acknowledged: true, deletedCount: 3 }

## Functions identified and purpose

Function	Purpose	Example usage
Mongoose.connect	Makes the connection to the MongoDB	Mongoose.connect('address')

	database	
Save()	Saves a single document in the database	Doc1.save
insertMany()	Inserts multiple documents into a collection	Model.insertMany(data)
Find()	Finds data matching to a query	Model.find({query})
countDocuments()	Counts documents in a collection	Model.countDocuments()
deleteMany()	Deletes multiple documents matching a query	Model.deleteMany({query})
updateMany()	Updates multiple documents matching a query	Model.updateMany({query},{updateData})