Topics

1. MongoDb Basic Operations
2. More on Searching for Documents
3. Replacing Documents
4. Updating Documents

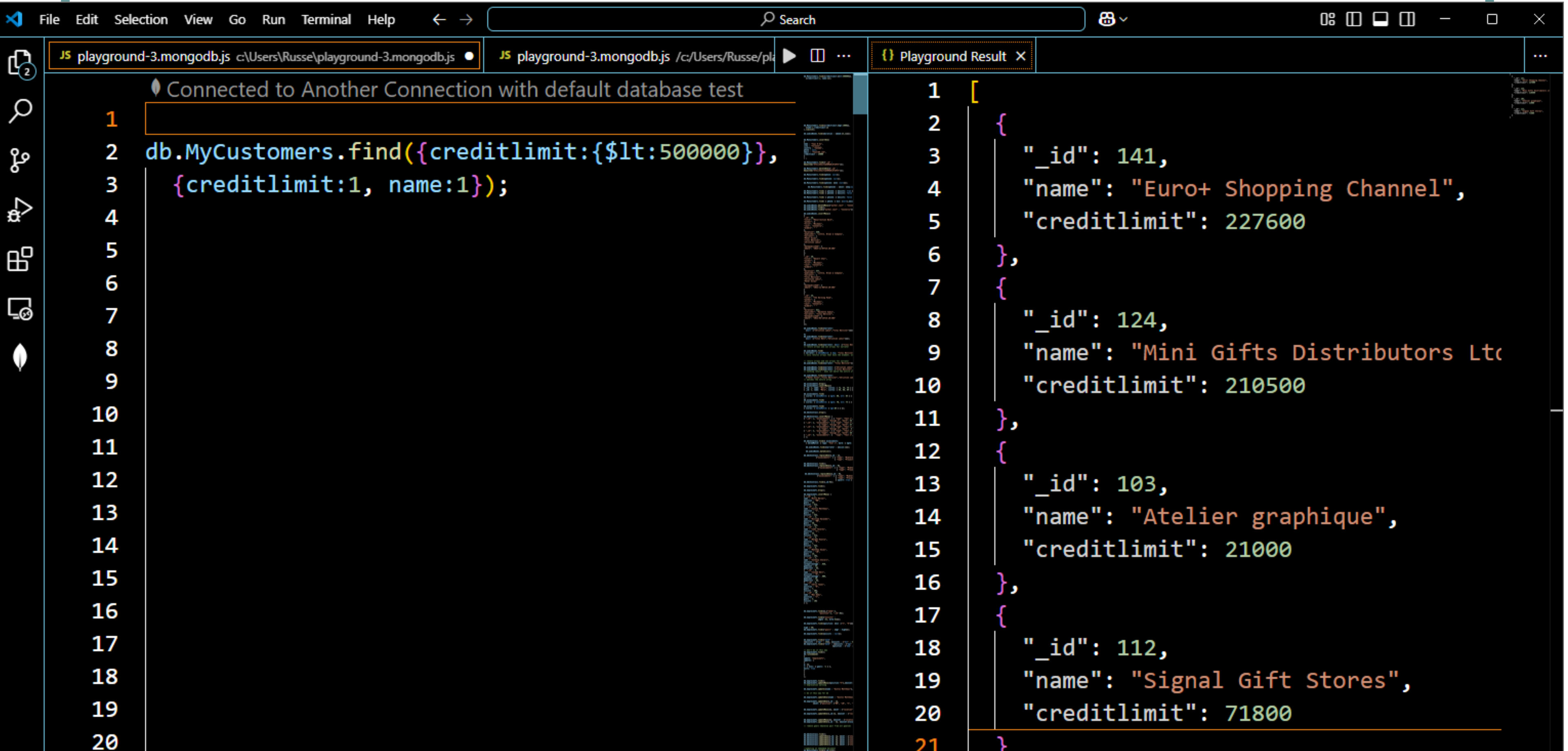
MongoDB

- 4 Basic Operations of Persistent Storage?
- CRUD
- CREATE, READ, UPDATE, DELETE
- SQL equivalent?
- INSERT, SELECT, UPDATE, DELETE
- MongoDB equivalent?

- Create
- db.collection.insertOne()
- db.collection.insertMany()
- Read
- db.collection.find()
- Can include a query criteria, projection and a cursor modifier

Projection

- db.MyCustomers.find({creditlimit:{$lt:500000}},
- {creditlimit:1, name:1});


File Edit Selection View Go Run Terminal Help
Search
playground-3.mongodb.js c:\Users\Russe\playground-3.mongodb.js
playground-3.mongodb.js /c/Users/Russe/pl
Playground Result

Connected to Another Connection with default database test

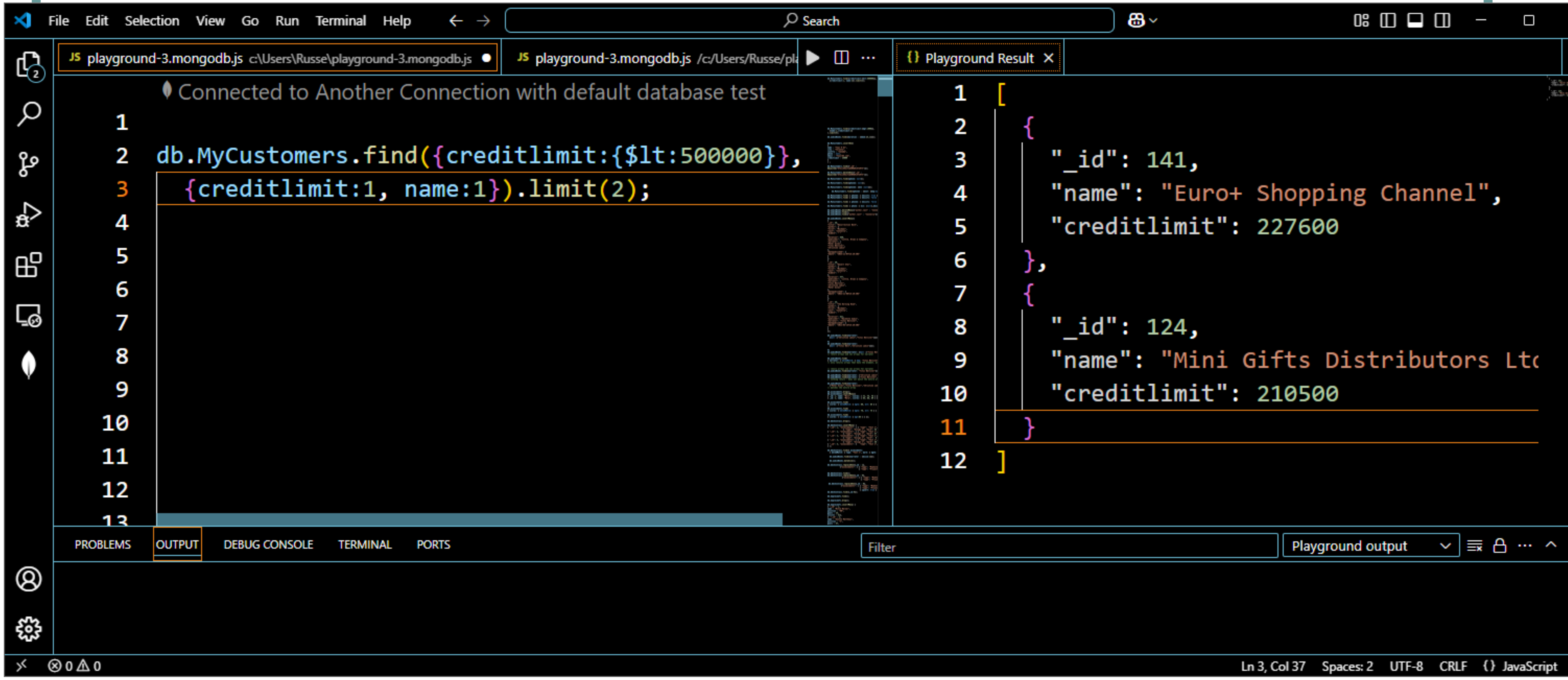
```

db.MyCustomers.find({creditlimit:{$lt:500000}},
  {creditlimit:1, name:1});
```

```
[
  {
    "_id": 141,
    "name": "Euro+ Shopping Channel",
    "creditlimit": 227600
  },
  {
    "_id": 124,
    "name": "Mini Gifts Distributors Ltd
    "creditlimit": 210500
  },
  {
    "_id": 103,
    "name": "Atelier graphique",
    "creditlimit": 21000
  },
  {
    "_id": 112,
    "name": "Signal Gift Stores",
    "creditlimit": 71800
  }
```

Cursor Modifier

- db.MyCustomers.find({creditlimit:{$lt:500000}},
- {creditlimit:1, name:1}).limit(2);



File Edit Selection View Go Run Terminal Help
Search
playground-3.mongodb.js c:\Users\Russe\playground-3.mongodb.js
playground-3.mongodb.js /c/Users/Russe/pl
Playground Result
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter
Playground output
Ln 3, Col 37 Spaces: 2 UTF-8 CRLF {} JavaScript

Connected to Another Connection with default database test

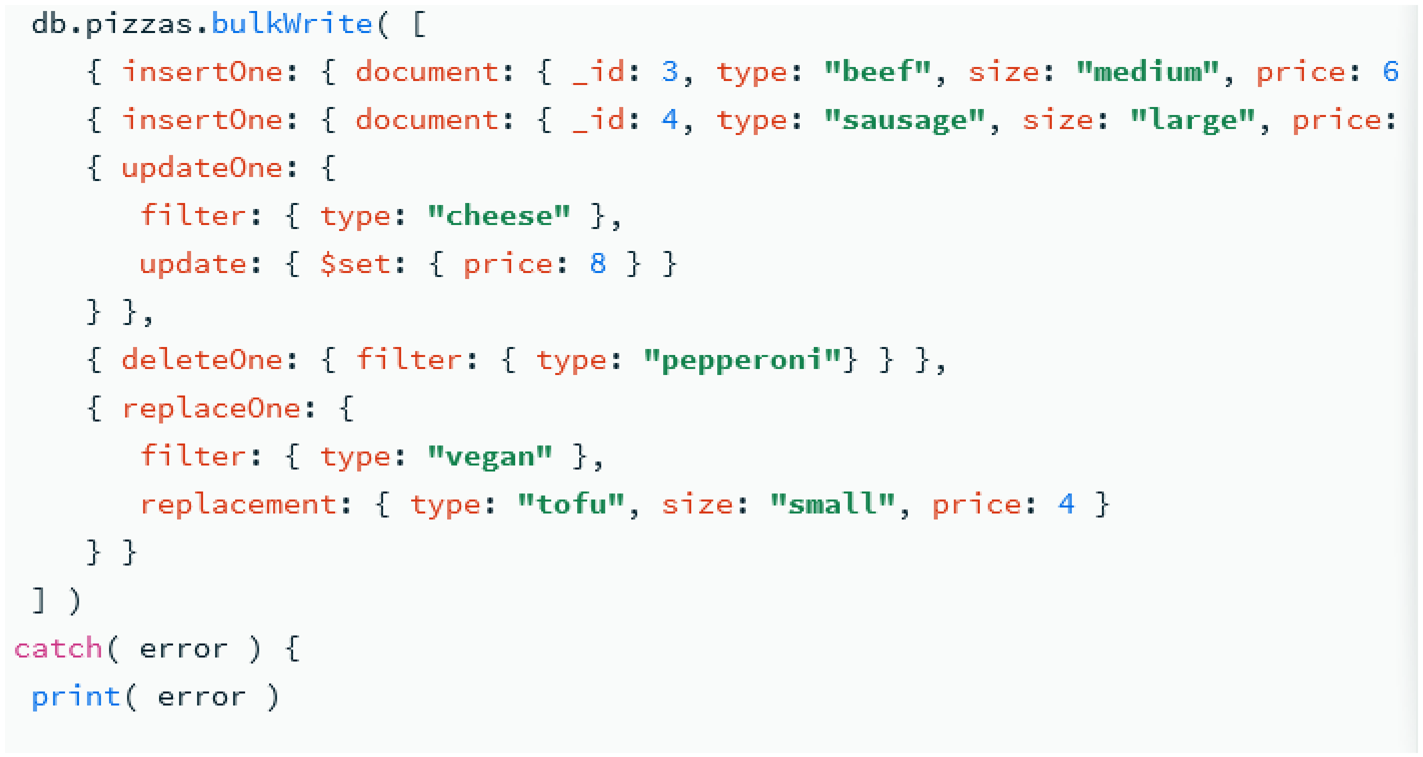
```
db.MyCustomers.find({creditlimit:{$lt:500000}},
  {creditlimit:1, name:1}).limit(2);
```

```
[
  {
    "_id": 141,
    "name": "Euro+ Shopping Channel",
    "creditlimit": 227600
  },
  {
    "_id": 124,
    "name": "Mini Gifts Distributors Ltc
    "creditlimit": 210500
  }
]
```

- MongoDB Updates
- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne

- MongoDB Deletes
- db.collection.deleteOne()
- db.collection.deleteMany()
- There is a MongoDB method that provides the ability to perform bulk insert, update, and delete operations.
- db.collection.bulkWrite()

bulkWrite



```
 db.pizzas.bulkWrite( [
    { insertOne: { document: { _id: 3, type: "beef", size: "medium", price: 6
    { insertOne: { document: { _id: 4, type: "sausage", size: "large", price:
    { updateOne: {
       filter: { type: "cheese" },
       update: { $set: { price: 8 } }
    } },
    { deleteOne: { filter: { type: "pepperoni"} } },
    { replaceOne: {
       filter: { type: "vegan" },
       replacement: { type: "tofu", size: "small", price: 4 }
    } }
 ] )
catch( error ) {
 print( error )
```

MongoDB Demonstration

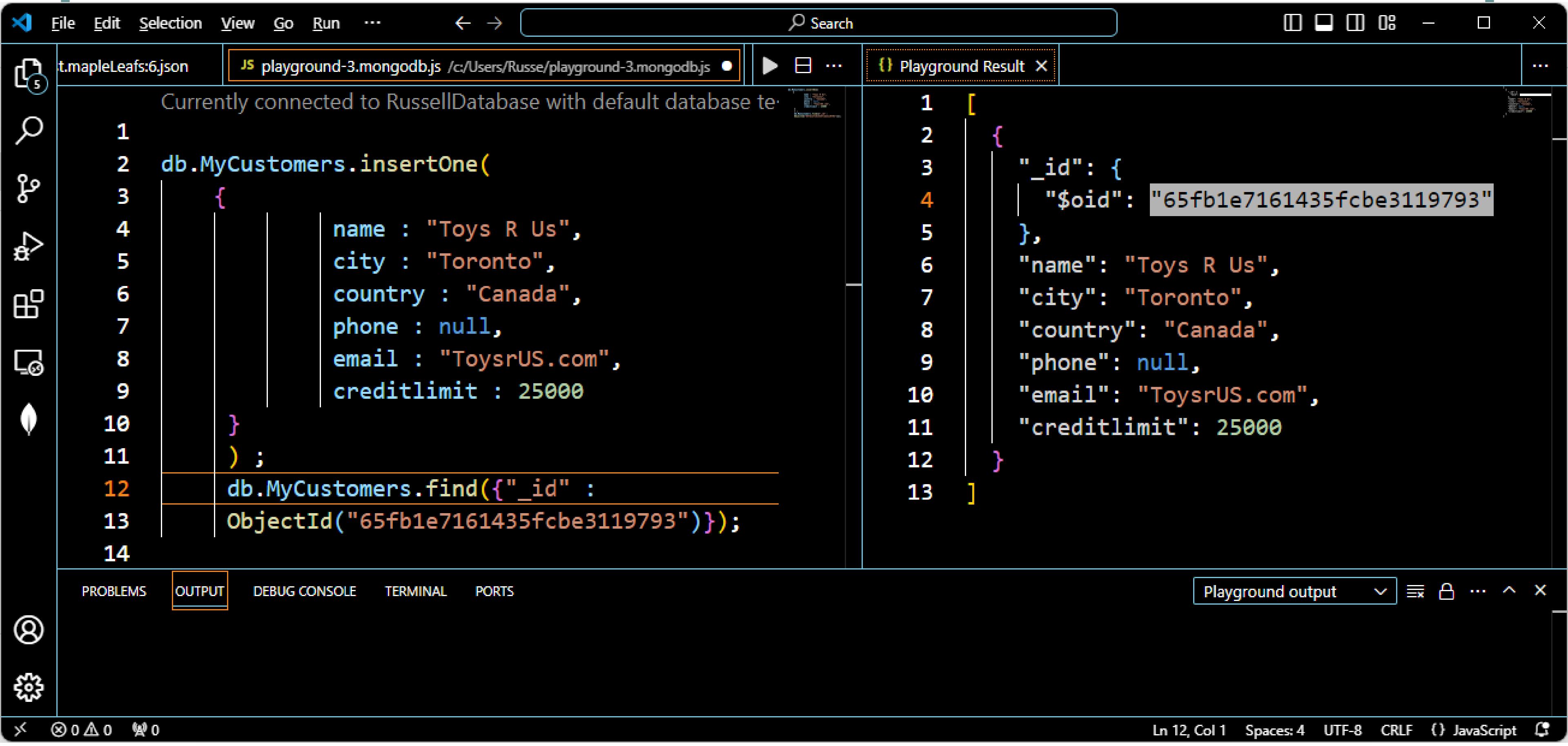
- “$or”/”$and” Operator
- The “$or” operator is used to check an array of possible criteria. The query returns the document if either condition is true.
- db.raffle.find({"$or" : [{"ticket_no" : 725}, {"winner" : true}]})
-
- db.raffle.find({"$or" : [{"ticket_no" : {"$in" : [725, 542, 390]}},
- {"winner" : true}]})

- See the following example of “$and”:
- db.users.find({"$and" : [{"x" : {"$gt" : 1}}, {"x" : {“$lt” : 4}}]})

MongoDB Demonstration

- $mod as a search criteria
- db.users.find({"id_num" : {"$mod" : [5, 1]}})
- This query returns documents if the key “id_num” is 1, 6, 11, or etc.
- "$mod" operator checks if the value of key “id_num” divided by the first value have a remainder of the second value.

- Searching by ObjectId

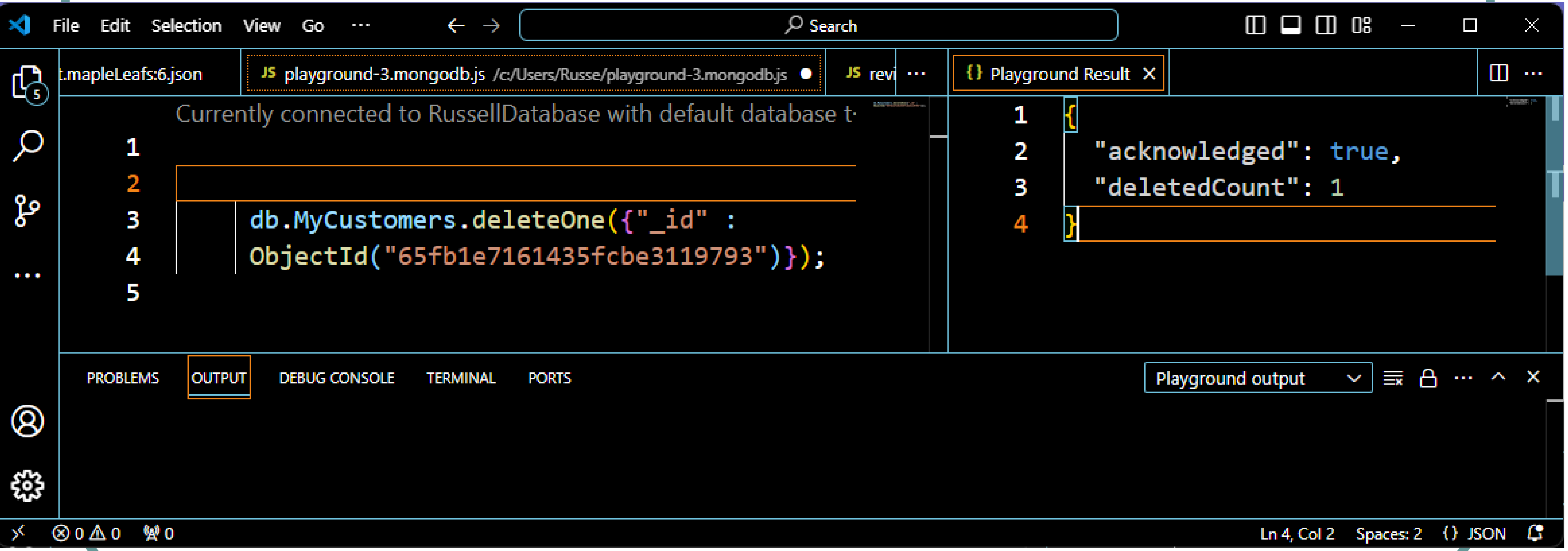

File Edit Selection View Go Run
Search
t.mapleLeafs:6.json
playground-3.mongodb.js /c:/Users/Russe/playground-3.mongodb.js
Playground Result
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Playground output
Ln 12, Col 1 Spaces: 4 UTF-8 CRLF JavaScript

Currently connected to RussellDatabase with default database te

```
db.MyCustomers.insertOne(
    {
            name : "Toys R Us",
            city : "Toronto",
            country : "Canada",
            phone : null,
            email : "ToysrUS.com",
            creditlimit : 25000
    }
    ) ;
    db.MyCustomers.find({"_id" :
    ObjectId("65fb1e7161435fcbe3119793")});
```

```
[
  {
    "_id": {
      "$oid": "65fb1e7161435fcbe3119793"
    },
    "name": "Toys R Us",
    "city": "Toronto",
    "country": "Canada",
    "phone": null,
    "email": "ToysrUS.com",
    "creditlimit": 25000
  }
]
```

- You can remove the document using ObjectId()



Currently connected to RussellDatabase with default database t

```
db.MyCustomers.deleteOne({"_id" :
ObjectId("65fb1e7161435fcbe3119793")});
```

```
{
  "acknowledged": true,
  "deletedCount": 1
}
```

Searching for Null

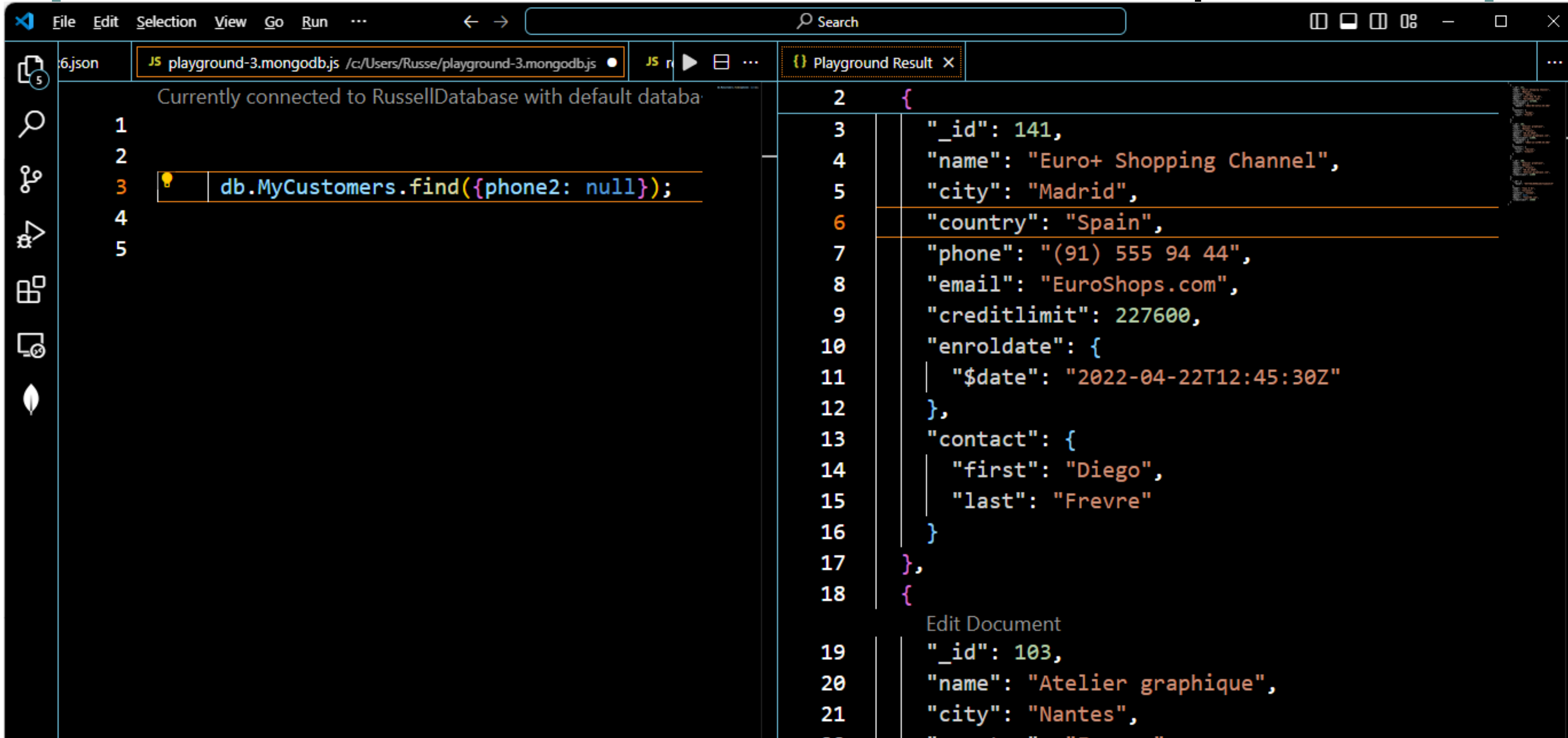
- Null
- Null means the value of a key is unknown.
- Assume the following documents:
- db.c.find()
- { "_id" : ObjectId("4ba0f0dfd22aa494fd523621"), "y" : null }
- { "_id" : ObjectId("4ba0f0dfd22aa494fd523622"), "y" : 1 }
- { "_id" : ObjectId("4ba0f148d22aa494fd523623"), "y" : 2 }
- To find documents with the NULL value for the “y” key:
  - db.c.find({"y" : null})
- { "_id" : ObjectId("4ba0f0dfd22aa494fd523621"), "y" : null }
- To find all documents that a specific key does not exist among their keys.
- db.c.find({"z" : null})
- { "_id" : ObjectId("4ba0f0dfd22aa494fd523621"), "y" : null }
- { "_id" : ObjectId("4ba0f0dfd22aa494fd523622"), "y" : 1 }
- { "_id" : ObjectId("4ba0f148d22aa494fd523623"), "y" : 2

- db.MyCustomers.find({phone: null});

```
{
   "_id": {
     "$oid": "65ff58c05981e81f2adee4c9"
   },
   "name": "Toys R Us",
   "city": "Toronto",
   "country": "Canada",
   "phone": null,
   "email": "ToysrUS.com",
   "creditlimit": 25000
 }
```

db.MyCustomers.find({phone2: null});

- Returns documents where phone2 is set as null and documents without a phone2

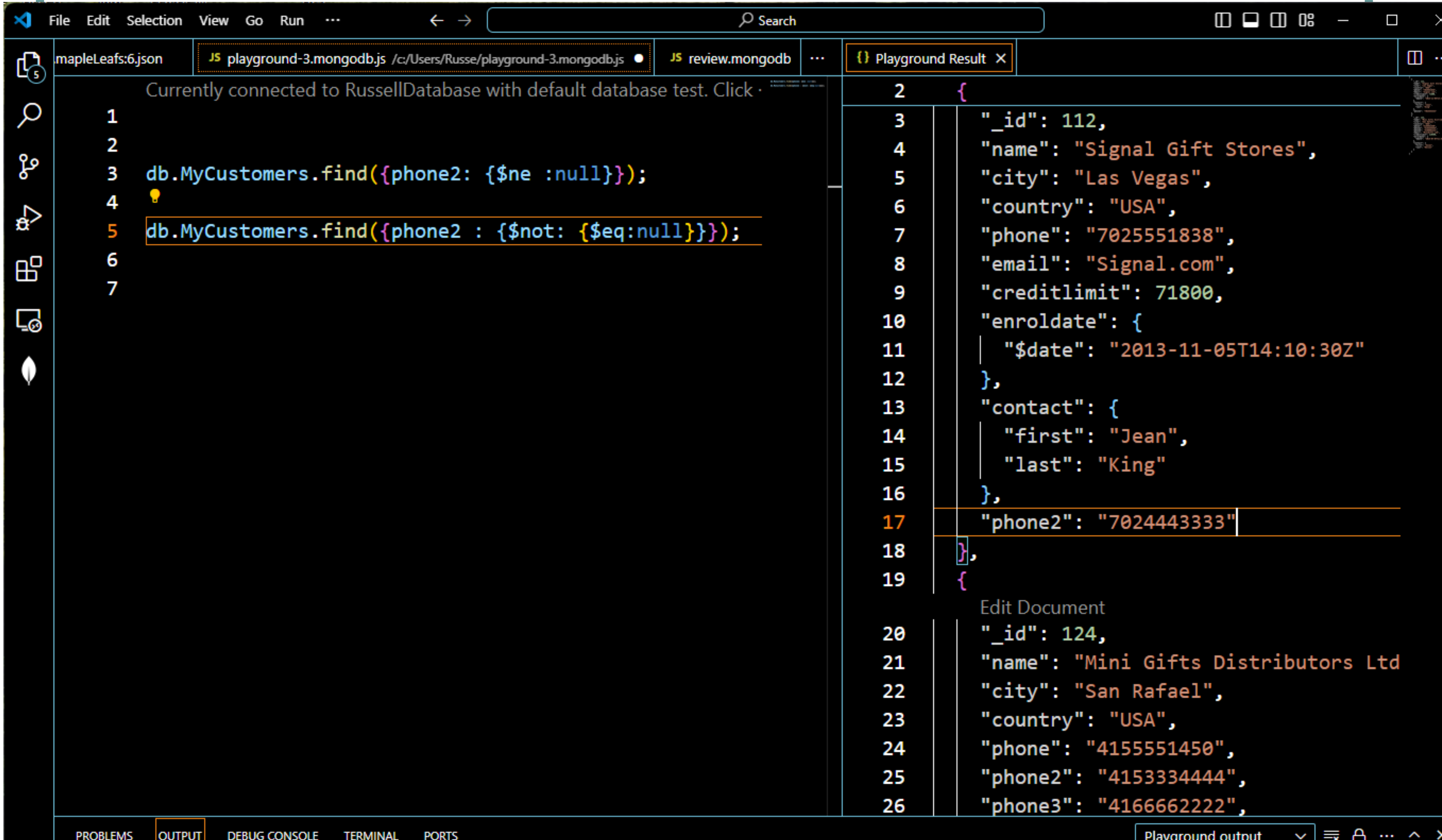

File Edit Selection View Go Run ···
Search
6.json
playground-3.mongodb.js /c:/Users/Russe/playground-3.mongodb.js
Playground Result
Edit Document

Currently connected to RussellDatabase with default databa

```
db.MyCustomers.find({phone2: null});
```

```
  {
    "_id": 141,
    "name": "Euro+ Shopping Channel",
    "city": "Madrid",
    "country": "Spain",
    "phone": "(91) 555 94 44",
    "email": "EuroShops.com",
    "creditlimit": 227600,
    "enroldate": {
      "$date": "2022-04-22T12:45:30Z"
    },
    "contact": {
      "first": "Diego",
      "last": "Frevre"
    }
  },
  {
    "_id": 103,
    "name": "Atelier graphique",
    "city": "Nantes",
```

db.MyCustomers.find({phone2: {$ne :null}});



```
Currently connected to RussellDatabase with default database test. Click ·


db.MyCustomers.find({phone2: {$ne :null}});

db.MyCustomers.find({phone2 : {$not: {$eq:null}}});
```

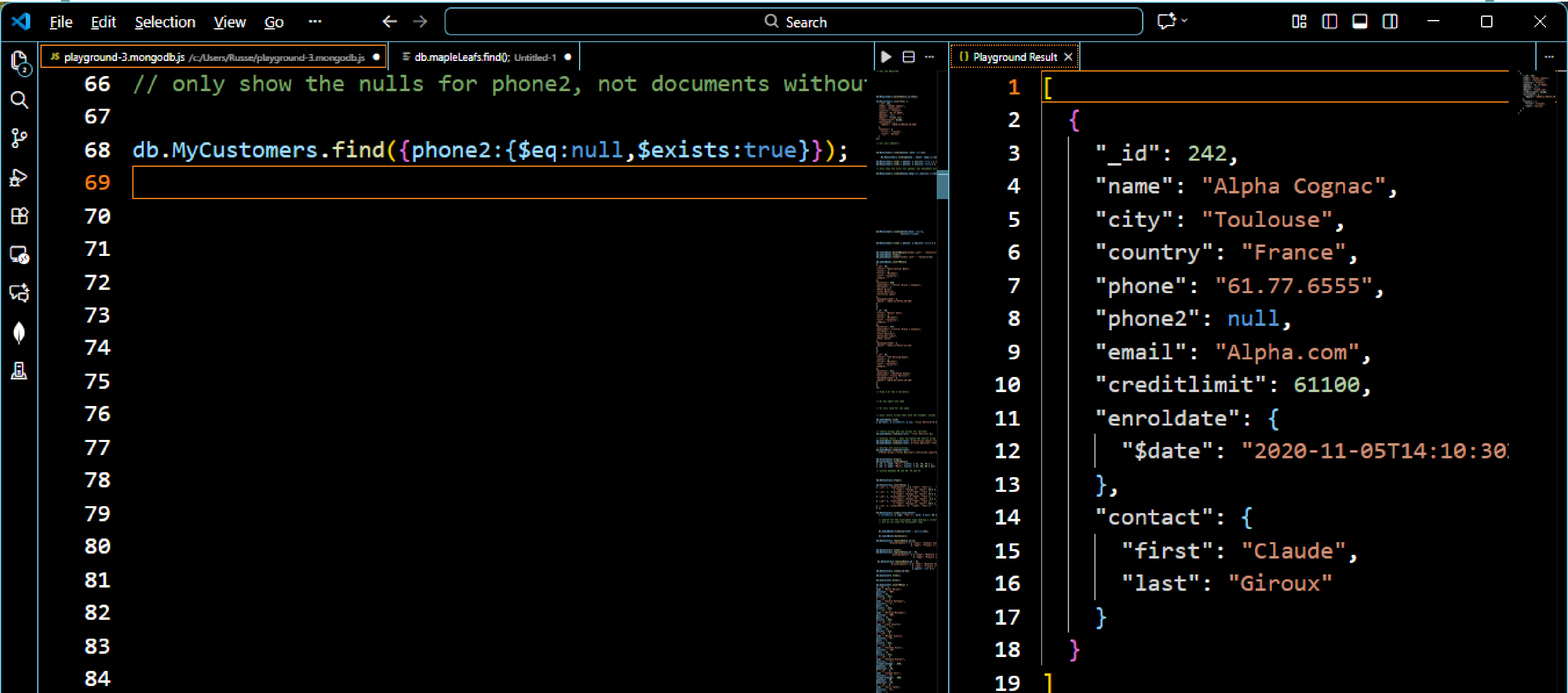
```
  {
    "_id": 112,
    "name": "Signal Gift Stores",
    "city": "Las Vegas",
    "country": "USA",
    "phone": "7025551838",
    "email": "Signal.com",
    "creditlimit": 71800,
    "enroldate": {
      "$date": "2013-11-05T14:10:30Z"
    },
    "contact": {
      "first": "Jean",
      "last": "King"
    },
    "phone2": "7024443333"
  },
  {
    Edit Document
    "_id": 124,
    "name": "Mini Gifts Distributors Ltd
    "city": "San Rafael",
    "country": "USA",
    "phone": "4155551450",
    "phone2": "4153334444",
    "phone3": "4166662222",
```

MongoDB

- “$exists” Operator
- The $exists operator matches documents that contain or do not contain a specified field, including documents where the field value is null.
- Documents that contain the key phone2
- db.MyCustomers.find({ phone2: { $exists: true } })
- and documents that do not
- db.MyCustomers.find({ phone2: { $exists: false } });

Omit documents without a phone2

- db.MyCustomers.find({phone2:{$eq:null, $exists:true}});



File
Edit
Selection
View
Go
Search
playground-3.mongodb.js /c/Users/Russe/playground-3.mongodb.js
db.mapleLeafs.find(); Untitled-1
Playground Result

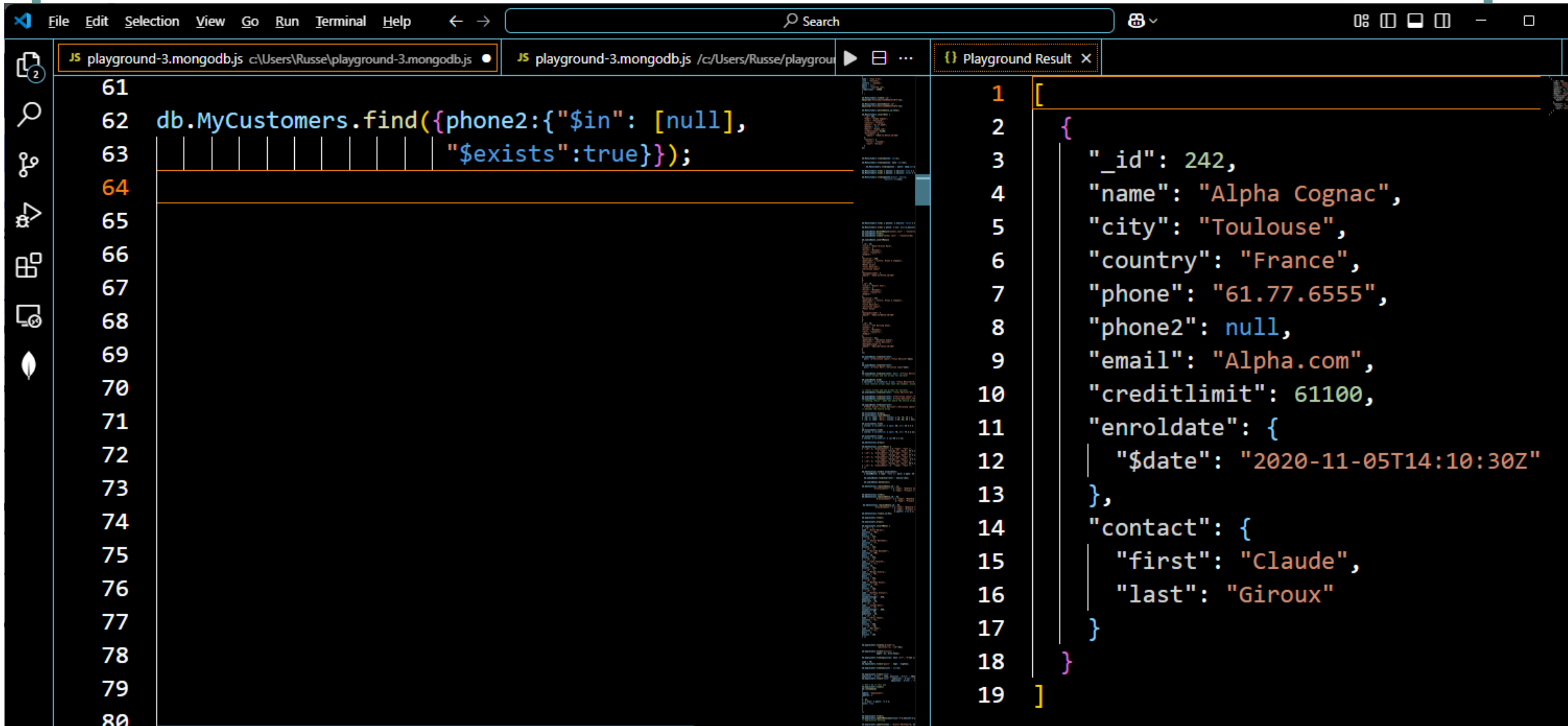
```
// only show the nulls for phone2, not documents withou

db.MyCustomers.find({phone2:{$eq:null,$exists:true}});
```

```
[
  {
    "_id": 242,
    "name": "Alpha Cognac",
    "city": "Toulouse",
    "country": "France",
    "phone": "61.77.6555",
    "phone2": null,
    "email": "Alpha.com",
    "creditlimit": 61100,
    "enroldate": {
      "$date": "2020-11-05T14:10:30
    },
    "contact": {
      "first": "Claude",
      "last": "Giroux"
    }
  }
]
```

Omit documents without a phone2

- db.MyCustomers.find({phone2:{"$in": [null], "$exists":true}});


File Edit Selection View Go Run Terminal Help
Search
playground-3.mongodb.js c:\Users\Russe\playground-3.mongodb.js
playground-3.mongodb.js /c/Users/Russe/playgrou
Playground Result

```
db.MyCustomers.find({phone2:{"$in": [null],
                    "$exists":true}});
```

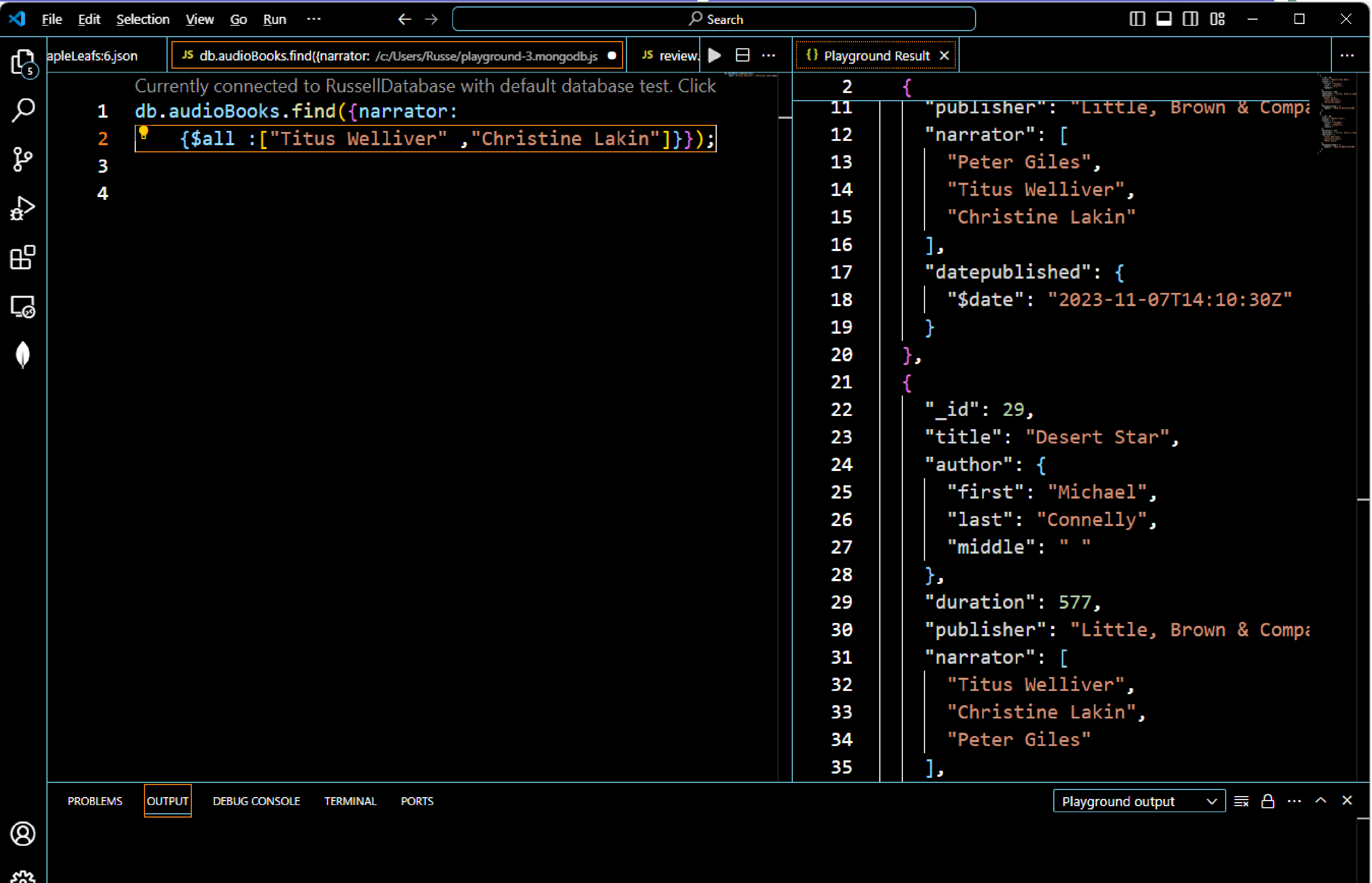
```
[
  {
    "_id": 242,
    "name": "Alpha Cognac",
    "city": "Toulouse",
    "country": "France",
    "phone": "61.77.6555",
    "phone2": null,
    "email": "Alpha.com",
    "creditlimit": 61100,
    "enroldate": {
      "$date": "2020-11-05T14:10:30Z"
    },
    "contact": {
      "first": "Claude",
      "last": "Giroux"
    }
  }
]
```

Searching Arrays

- “$all” Operator
- db.food.insert({"_id" : 1, "fruit" : ["apple", "banana", "peach"]})
- db.food.insert({"_id" : 2, "fruit" : ["apple", "kumquat", "orange"]})
- db.food.insert({"_id" : 3, "fruit" : ["cherry", "banana", "apple"]})
- Let’s say we want or find all documents with both apple and banana elements.
- db.food.find({fruit : {$all : ["apple", "banana"]}})
- {"_id" : 1, "fruit" : ["apple", "banana", "peach"]}
- {"_id" : 3, "fruit" : ["cherry", "banana", "apple"]}
- To check key/values pairs with the exact match does not return the above result. It looks for documents with only values apple and banana.
- db.food.find({"fruit" : ["apple", "banana"]})
- The following query does not return any documents:
- db.food.find({"fruit" : ["banana", "apple", "peach"]})
- Returns 0 documents

- db.audioBooks.find({narrator: {$all :["Christine Lakin","Titus Welliver"]}});
- Vs
- db.audioBooks.find({narrator: {$all :["Titus Welliver","Christine Lakin"]}});
- The order did not matter
- Vs
- db.audioBooks.find({narrator: {$all :["Titus Well","Christine Lakin"]}});
- This does not work, it requires all names match.
- Vs
- db.audioBooks.find({narrator: {$all :["Titus Welliver"]}});
- This matches all documents with array narrators, but it also returns a match to a narrator Titus Welliver where the narrator field does not hold an array of values – just one value.

$all used with arrays


File Edit Selection View Go Run ···
Search
apleLeafs:6.json
db.audioBooks.find({narrator: /c:/Users/Russe/playground-3.mongodb.js
review.
Playground Result
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Playground output

Currently connected to RussellDatabase with default database test. Click

```
db.audioBooks.find({narrator:
    {$all :["Titus Welliver" ,"Christine Lakin"]}});
```

```
    {
      "publisher": "Little, Brown & Compa
      "narrator": [
        "Peter Giles",
        "Titus Welliver",
        "Christine Lakin"
      ],
      "datepublished": {
        "$date": "2023-11-07T14:10:30Z"
      }
    },
    {
      "_id": 29,
      "title": "Desert Star",
      "author": {
        "first": "Michael",
        "last": "Connelly",
        "middle": " "
      },
      "duration": 577,
      "publisher": "Little, Brown & Compa
      "narrator": [
        "Titus Welliver",
        "Christine Lakin",
        "Peter Giles"
      ],
```

- To only include arrays that have a particular narrator name and exclude non arrays with that narrator name
- Returning all the documents where Titus works with other narrators, so exclude the ones where he is alone

```
db.audioBooks.find(
    { narrator: { $elemMatch: { $eq: "Titus Welliver"} } }
 )
```

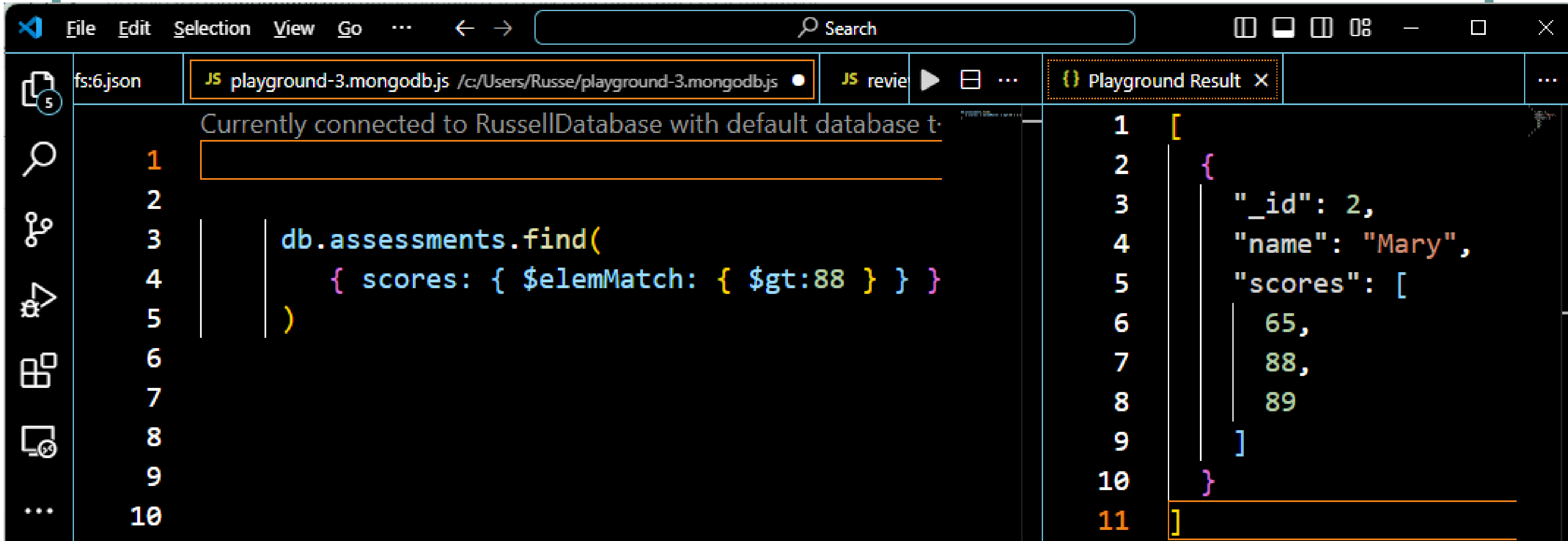
Find without $all or $elemMatch

- This technique returns no matches because there are 3 elements in the narrator array and the query only includes 2 elements.
- db.audioBooks.find({narrator: ["Christine Lakin","Titus Welliver"]});
- db.audioBooks.find({narrator: ["Titus Welliver","Christine Lakin"]});
- This query returns a document because it matches an entire array.
- db.audioBooks.find({narrator:
- ["Peter Giles","Titus Welliver","Christine Lakin"]});

- db.assessments.insertMany([
-     { _id: 1, name: "Bill", scores: [72, 75, 78] },
-     { _id: 2, name: "Mary", scores: [65, 88, 89] }])

- db.assessments.find(
-         { scores: { $elemMatch: { $gte: 80, $lt: 85 } } }
-     )
- Returns []
- db.assessments.find(
-         { scores: { $elemMatch: { $gte: 70, $lt: 75 } } }
-     )
- Returns Bill document

- db.assessments.find(
- { scores: { $elemMatch: { $gt:88 } } })
- One element of Mary’s score array matches, none of Bill’s score array match


File
Edit
Selection
View
Go
Search
fs:6.json
playground-3.mongodb.js /c:/Users/Russe/playground-3.mongodb.js
revie
Playground Result

Currently connected to RussellDatabase with default database t

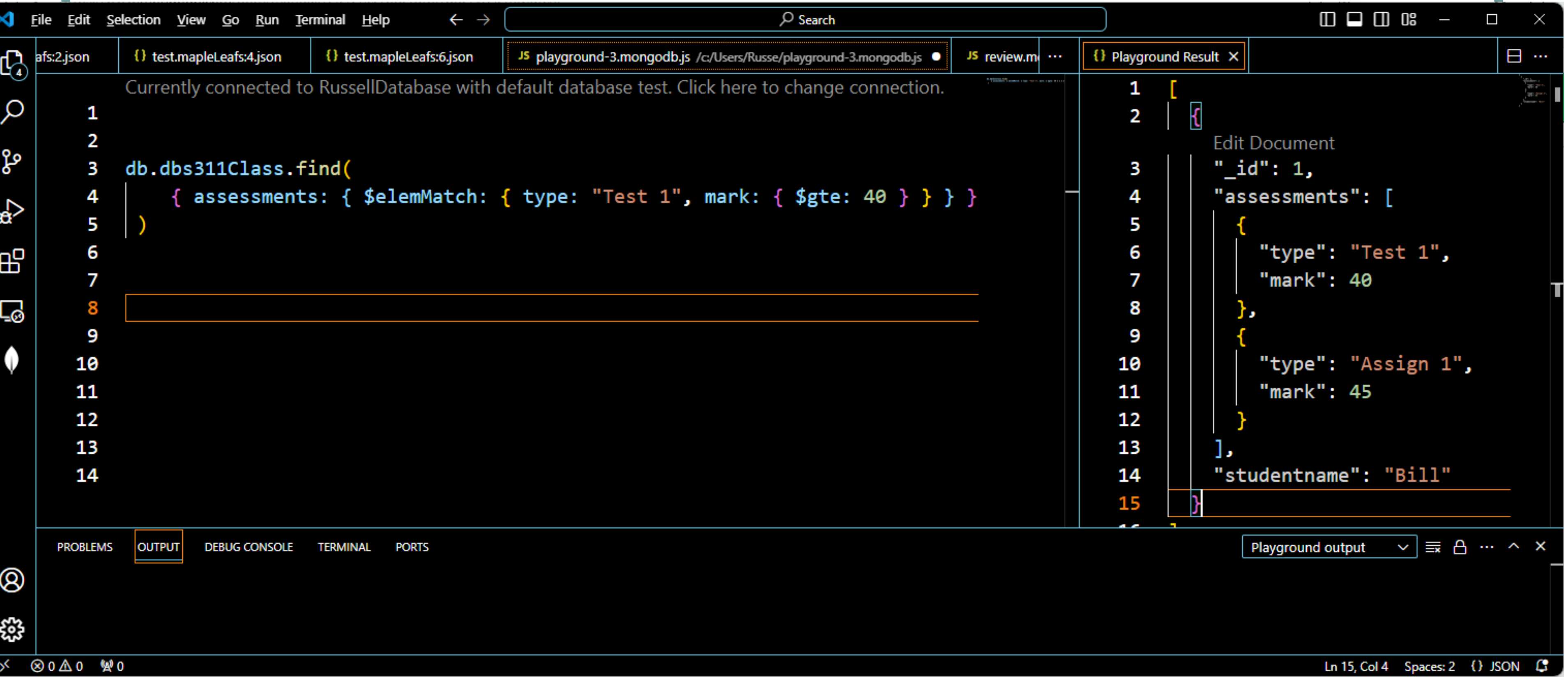
```
db.assessments.find(
    { scores: { $elemMatch: { $gt:88 } } }
)
```

```
[
  {
    "_id": 2,
    "name": "Mary",
    "scores": [
      65,
      88,
      89
    ]
  }
]
```

$elemMatch

```
db.dbs311Class.insertMany( [
   { "_id": 1, "assessments": [ { "type": "Test 1",   "mark": 40 },
                                { "type": "Assign 1", "mark": 45 } ] },
   { "_id": 2, "assessments": [ { "type": "Test 1",   "mark": 32 },
                                { "type": "Assign 1", "mark": 37 } ] },
   { "_id": 3, "assessments": [ { "type": "Test 1",   "mark": 22 },
                                { "type": "Assign 1", "mark":  8 } ] },
   { "_id": 4, "assessments": [ { "type": "Test 1",   "mark": 37 },
                                { "type": "Assign 1", "mark": 38 } ] },
   { "_id": 5, "assessments":  {   "type": "Test 1",   "mark": 35 } }
 ] );
```

- db.dbs311Class.find({ assessments:
- { $elemMatch: { type: "Test 1", mark: { $gte: 40 } } } })
- Only one document matches that search filter criteria



Currently connected to RussellDatabase with default database test. Click here to change connection.

```
db.dbs311Class.find(
    { assessments: { $elemMatch: { type: "Test 1", mark: { $gte: 40 } } } }
)
```

```
[
  {
    "_id": 1,
    "assessments": [
      {
        "type": "Test 1",
        "mark": 40
      },
      {
        "type": "Assign 1",
        "mark": 45
      }
    ],
    "studentname": "Bill"
  }
```

File Edit Selection View Go Run Terminal Help
Search
afs:2.json
test.mapleLeafs:4.json
test.mapleLeafs:6.json
playground-3.mongodb.js /c/Users/Russe/playground-3.mongodb.js
review.m
Playground Result
Edit Document
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Playground output
Ln 15, Col 4 Spaces: 2 {} JSON

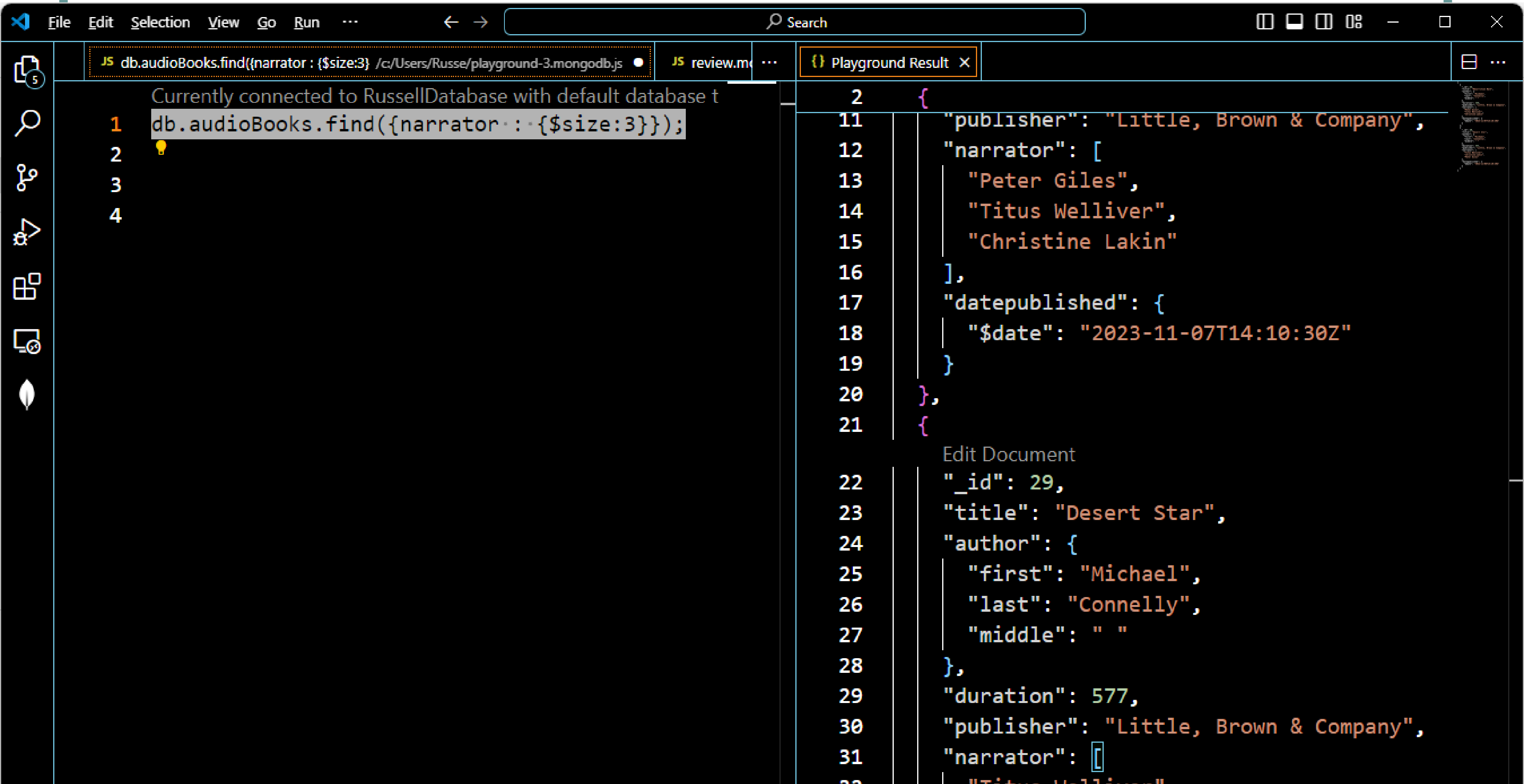
- // search for the assessment type meeting a critera
- // but do not show the assessment type
- db.dbs311Class.find({ assessments:
- { $elemMatch: { type: "Test 1", mark: { $gte: 40 } } } },
- {"assessments.mark":1});
- Or
- db.dbs311Class.find({ assessments:
- { $elemMatch: { type: "Test 1", mark: { $gte: 40 } } } },
- {"assessments.type":0});

- Projection document {"assessments.type":0});
- [{
-     "_id": 1,
-     "assessments": [
-       {
-         "mark": 40
-       },
-       {
-         "mark": 45
-       }
-     ]
-   }]

MongoDB Demonstration

- “$size” Operator
- To query arrays for a given size, the “$size” operator is used.
- db.food.find({"fruit" : {"$size" : 3}})
- You cannot combine the "$size" operator with other $ conditional operators.

- db.audioBooks.find({narrator : {$size:3}});


File
Edit
Selection
View
Go
Run
Search
db.audioBooks.find({narrator : {$size:3}} /c:/Users/Russe/playground-3.mongodb.js
review.m
Playground Result

```
Currently connected to RussellDatabase with default database t
db.audioBooks.find({narrator : {$size:3}});
```

```
{
  "publisher": "Little, Brown & Company",
  "narrator": [
    "Peter Giles",
    "Titus Welliver",
    "Christine Lakin"
  ],
  "datepublished": {
    "$date": "2023-11-07T14:10:30Z"
  }
},
{
Edit Document
  "_id": 29,
  "title": "Desert Star",
  "author": {
    "first": "Michael",
    "last": "Connelly",
    "middle": " "
  },
  "duration": 577,
  "publisher": "Little, Brown & Company",
  "narrator": [
```

- You can also return the size in bytes of the collection
- db.audioBooks.dataSize();
- 1   3787

MongoDB Demonstration

- Document Replacement
- To replace a document with a new one, the replaceOne function is used.
- db.collection.replaceOne()
- Replaces the first matching document in the collection that matches the filter, using the replacement document.
- Replaces at most a single document that match a specified filter even though multiple documents may match the specified filter.
- There is no db.collection.replaceMany option

MongoDB Demonstration

- Document Replacement
- Assume the following user document: { "name" : "joe", }
- Let’s replace this document with the new one
- db.people.replaceOne({ "name" : "joe"}, { "name" : "joe", "friends" : 32, "enemies" : 2})
- If multiple documents have “name”: “joe” the first one found will be replaced
- If you want a specific document – not the first then:
- Use a unique id in your filter to replace a specific document in the collection

replaceOne()

- db.dbs311Class.replaceOne({_id : 1},
- {"assessments": [{ "type": "Midterm Test",   "mark": 40 },
-                                 { "type": "Project 1", "mark": 45 }] });
- Replaces one document
- db.dbs311Class.replaceOne({_id : 9},
- {"assessments": [{ "type": "Midterm Test",   "mark": 40 },
-                                 { "type": "Project 1", "mark": 45 }] });
- "matchedCount": 0,
-   "modifiedCount": 0,
-   "upsertedCount": 0

replaceOne()

- Upsert is a Boolean option that defaults to false,
- when it is true the unmatched document is added

```
db.dbs311Class.replaceOne({_id : 9},
            {"assessments": [ { "type": "Midterm Test",   "mark": 40 },
                             { "type": "Project 1", "mark": 45 } ] },
                             { upsert: true } );
"acknowledged": true,
  "insertedId": 9,
  "matchedCount": 0,
  "modifiedCount": 0,
  "upsertedCount": 1
```

MongoDB Demonstration

- Update Documents
- The update function is used to update the value of a key value in a document.
- update() takes two parameters:
- A query document
- Locates document to update
- Modifier document
- Describes changes to make
- The update operation is atomic:
- If there are two update requests coming to the server, the one reaches the server first will be executed and when it is done the second one will be applied

MongoDB Demonstration

- db.members.update({_id : 1}, {$inc: {"points":10}});
- DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
- The warning means this will be removed in a future version of MongoDB and is replaced by …
- Another method to alter a value in a key value pair
- “$set”
- The $set operator replaces the value of a field with the specified value.
- If the field does not exist, $set will add a new field with the specified value

MongoDB Demonstration

- “$set” (Add a Field)
- db.members.updateOne(
-    {_id : 2},
-    {$set: {"location": "Toronto"}});
- {
-   "acknowledged": true,
-   "insertedId": null,
-   "matchedCount": 1,
-   "modifiedCount": 1,
-   "upsertedCount": 0
- }

- A new collection
- db.mapleLeafs.insertMany([
- { "_id" : 1,
- name : “Nick Robertson",
- position : “LW",
- goals : 25,
- assists : 51},
- { "_id" : 2,
- name : "Austin Matthews",
- position : "C", …

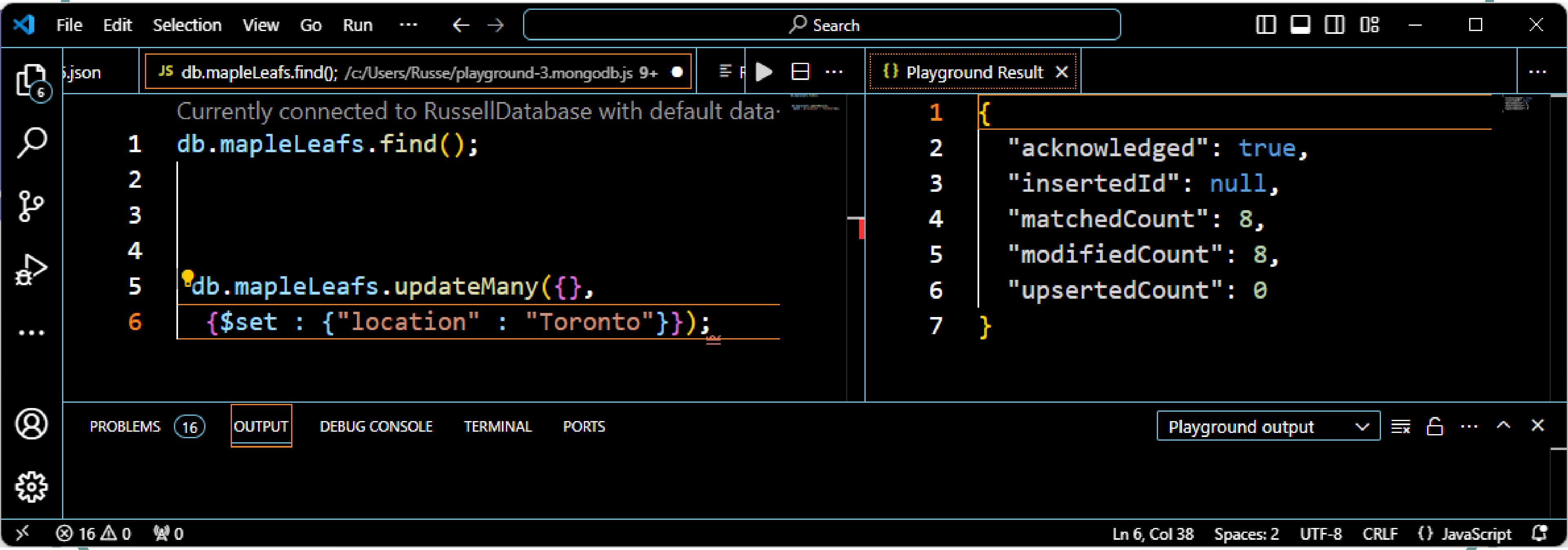
Updates

- Increase all the goals for maple leafs by 1
- db.runCommand(
- {
- update: "mapleLeafs",
- updates: [
- {
- q: {},
- u: { $inc: { goals: 1 } },
- multi: true}]
- });
- Don’t use this method, use updateOne or updateMany

- Another method to add goals to a total for a single document
- db.mapleLeafs.update({name : "Austin Matthews"}, {$inc: {goals:3}});
- DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.

- The best way to add to the goal total
- db.mapleLeafs.updateOne({name : "Austin Matthews"}, {$inc: {goals:3}});
- How do you subtract from a total
- db.mapleLeafs.updateOne({name : "Austin Matthews"}, {$inc: {goals: -3}});

- db.mapleLeafs.updateOne({_id:2}, {$set : {"location" : "Toronto"}})
- db.mapleLeafs.updateMany({}, {$set : {"location" : "Toronto"}});


File Edit Selection View Go Run ···
Search
.json
db.mapleLeafs.find(); /c/Users/Russe/playground-3.mongodb.js 9+
Playground Result
PROBLEMS 16
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Playground output
⊗ 16 ⚠ 0 0
Ln 6, Col 38 Spaces: 2 UTF-8 CRLF {} JavaScript

```
Currently connected to RussellDatabase with default data·
db.mapleLeafs.find();




db.mapleLeafs.updateMany({},
  {$set : {"location" : "Toronto"}});
```

```
{
  "acknowledged": true,
  "insertedId": null,
  "matchedCount": 8,
  "modifiedCount": 8,
  "upsertedCount": 0
}
```

MongoDB Demonstration

- “$set” (Modify a Field)
- The “$set” operator sets the value of a field if the field exists.
- Let’s say we want to change the value of “favorite book”:
- db.users.updateOne({"name" : "joe"},
-   {"$set" : {"favorite book" : "Green Eggs and Ham"}})
- Using the “$set” operator, we can change the value of “favorite book” to an array. The user has different favorite books:
- db.users.updateOne({"name" : "joe"},
-   {"$set" : {"favorite book" :
-     ["Cat's Cradle", "Foundation Trilogy", "Ender's Game"]}})

MongoDB Demonstration

- “$unset” Operator

- used to remove a key value pair from a document.
- Suppose the user does not have any favorite books and we want to remove the “favorite book” key.
- db.users.updateOne({"name" : "joe"},
- ... {"$unset" : {"favorite book" : 1}})
- The document now is
-   db.users.findOne()
- {   “ _id" : ObjectId("4b253b067525f35f94b60a31"),
-       "name" : "joe",
-        "age" : 30,
-       "sex" : "male",
-       "location" : "Wisconsin" }

MongoDB Demonstration

- “$set” (Embedded Documents)
- "_id": 8,
-   "title": "In Pieces",
-   "author": {
-    "first": "Sally",
-    "last": "Field"
-   },
-   "duration": 641,
- db.audioBooks.updateOne({_id : 8}, {$set: {"author.first": "Toronto"}});
- {
- "acknowledged": true,
- "insertedId": null,
- "matchedCount": 1,
- "modifiedCount": 1,
- "upsertedCount": 0
- }

Remaining Schedule

- Lab 9 has been been posted

Week	Lecture	Lab
Nov 17 WK	Online Lecture MongoDB Update Documents	Visual Code Lab 8 Due online
Nov 24 WK	Online Lecture Aggregation	Visual Code Lab 9 due online
Dec 1 WK	Online Lecture Review for Final	Assign2 due online
Dec 8 WK	In Class Final Test	Lab 10 due online