Topics

1. MongoDb Review
2. More on Arrays
3. Aggregation Framework
4. Aggregation Commands

Review

- If you want to remove the key value pair **“age” 30**, what keyword goes in the modifier document?
- {"$unset" : {“age" : 1}})
- What does $set provide with updateOne or updateMany
- Change the existing value for a key value pair or
- Include a new key value pair that wasn’t’ in the document
- What message do you get if you use Update instead of UpdateOne?
- Deprecated – a recommendation to not use it in the future
- If you want a different document for document id 10 in the cars collection what comes before the parenthesis db.cars??(
- 3 new key value pairs, one changed value for a key value pair
- db.cars.replaceOne(

Operators

- What are 3 query operators used with arrays?
- $all
- Matches arrays that contain all elements specified in the query, may also match a non array
- $elemMatch
- Selects documents if the element in the array field matches all the specified $elemMatch conditions ignores non arrays
- $size
- Selects documents based on the number of array elements

Arrays

- You can also use $type to identify your arrays
- db.aB.find({narrator : {$type: "array"}});
- Will return all documents where the narrator was set up as an array
- db.aB.find({narrator : {$type: "string"}});?
- Returns arrays and non arrays that are string

- How can you return just the documents with narrator key value pairs that are not using arrays?
- db.aB.find({narrator : {$not : {$type: "array"}}});
- { "_id": 25,
- "title": "The Burning Room",
- "author": { "first": "Michael", "last": "Connelly"},
- "duration": 611,
- "publisher": "Hachette Audio",
- "narrator": "Titus Welliver",
- "datepublished": {
- "$date": "2011-03-14T14:10:30Z"
- } }

Review

- You have arrays for more than one narrator in a key value pair. What keyword do you use to locate the audiobooks using “Meg Smith” as one of the narrators excluding the audiobooks where she is the only narrator

- db.audioBooks.find(
- { narrator: { $elemMatch: { $eq: “Meg Smith" } } }
-);

Running a MongoDB statement in the lab

- Close the playground window if the run arrow does not show
- Move the dividing line between windows
- Select the statement and press CTRL+ALT+S
- Don’t use comment method
- Don’t use multiple windows – all code for the lab in one window

MongoDB Demonstration

- Array modifiers
- There are modifiers to update arrays:
- "$push“
- "$each“
- "$slice“
- "$sort“
- “$addToSet”

MongoDB Demonstration

- “$push”
- The "$push" operator adds elements to the end of an array if the array exists in a key value pair.
- If the key value array pair does not exist it will be created with the given elements.
- db.blog.posts.findOne() { "_id" : ObjectId("4b2d75476cc613d5ee930164"), "title" : "A blog post", "content" : "..." }
- The following command adds an array comment to the existing document.
- db.blog.posts.update({"title" : "A blog post"},
- {"$push" : {"comments" :
- {"name" : "joe", "email" : "joe@example.com",
- "content" : "nice post."}}})
- Since the “comments” key does not exist, it will be created.

MongoDB Demonstration

- Added Array
- See the document from the previous slide, with key “comments” added to it.
- db.blog.posts.findOne()
- { "_id" : ObjectId("4b2d75476cc613d5ee930164"),
- "title" : "A blog post",
- "content" : "...",
- "comments" : [
- {
- "name" : "joe",
- "email" : "joe@example.com",
- "content" : "nice post." }
-]
- }

MongoDB Demonstration

- “$each”
- The “$each” operator is used to push multiple values to an array on one “$push” operation.
- db.stock.ticker.update({"_id" : "GOOG"},
- {"$push" :
- {"hourly" : {"$each" : [562.776, 562.790, 559.123]}}})
- Three elements are pushed into the array.

- db.MyCustomers.updateOne({_id:141},
- {$push :{stores:{ $each :["Barcelona", "Madrid",
- "Seville"]}}});
- No stores key value pair existed, but command above will be successful in producing an array for stores
- db.MyCustomers.updateOne({_id:141}, {$set :{stores: "Madrid"}});
- db.MyCustomers.updateOne({_id:141},
- {$push :{stores:{ $each :["Barcelona", "Seville"]}}});
- Fails with: The field 'stores' must be an array but is of type string in document

- db.MyCustomers.updateOne({_id:141}, {$set :{stores: ["Madrid"]}});
- This next statement does not fail since we started with an array
- db.MyCustomers.updateOne({_id:141},
- {$push :{stores:{ $each :["Seville“, "Barcelona",]}}});
- $sort
- db.MyCustomers.updateOne({_id:141},
- {$push :{stores:{ $each :["Seville","Barcelona"],$sort: 1} }});
- The elements are sorted to Barcelona, Madrid, Seville order

- Controlling for duplicate elements
- db.MyCustomers.updateOne({_id:141},
- {$push :{stores:{ $each :["Seville","Barcelona"],$sort: 1} }});
- The second time the above statement is run we get
- "stores": [
- "Barcelona",
- "Barcelona",
- "Madrid",
- "Seville",
- "Seville"
-]

- If we have Barcelona, Madrid and Saville in the stores array and we run:
- db.MyCustomers.updateOne({_id:141},
- {$addToSet :{stores:{ $each :
- ["Seville","Barcelona","Granada"] }}});
- It will produce:
- "stores": [
- "Barcelona",
- "Madrid",
- "Seville",
- "Granada"]

MongoDB Demonstration

- “$slice”
- The “$slice” operator is used with the “$push” operator to make sure that an array will not grow bigger that a certain size.
- db.movies.find({"genre" : "horror"}, ...
- {"$push" : {"top10" : {
- "$each" : ["Nightmare on Elm Street", "Saw"],
- "$slice" : 10}}})
- The value of the “$slice” limits the array to hold 10 elements. If the number of pushing elements violates the size, only the last 10 elements added to the array will be kept.

MongoDB Demonstration

- "$sort“
- You can sort an array before pushing more elements to the array.
- db.movies.find({"genre" : "horror"},
- {"$push" : {"top10" : {
- "$each" : [{"name" : "Nightmare on Elm Street", "rating" : 6.6},
- {"name" : "Saw", "rating" : 4.3}],
- "$slice" : -10, ... "$sort" : {"rating" : -1}}}})
- Before pushing the new elements, all elements existing in the array and the new ones is sorted by the “rating” key and the first 10 elements from the sorted list will be stored into the array.

- Stores has Barcelona,Madrid,Seville
- db.MyCustomers.updateOne({_id:141},
- {$push :{stores:{ $each :["Aa","Zz"],$sort: 1,$slice:3}
- }});
- Will cause stores to have:
- "stores": [
- "Aa",
- "Barcelona",
- "Madrid"
-]

MongoDB Demonstration

- Arrays as Sets
- If you want to avoid duplicates in your array, you can use “$ne” operator when pushing elements into an array.
- db.papers.updateOne({"authors cited" : {"$ne" : "Richie"}},
- {$push : {"authors cited" : "Richie"}})
- The above command pushes a new author to the array if the element “Richie” does not exist in the array.

MongoDB Demonstration

- $addToSet
- To avoid duplicates in array, the “$addToSet” can also be used.
- db.users.findOne({"_id" : ObjectId("4b2d75476cc613d5ee930164")})
- { "_id" : ObjectId("4b2d75476cc613d5ee930164"),
- "username" : "joe",
- "emails" : [
- "joe@example.com",
- "joe@gmail.com",
- "joe@yahoo.com"
-] }
- To add another email, we use “$addToSet” to avoid pushing duplicates into the array.

MongoDB Demonstration

- Adding Elements using “$addToSet”
- db.users.update({"_id" : ObjectId("4b2d75476cc613d5ee930164")},
- {"$addToSet" : {"emails" : "joe@gmail.com"}})
- The new element will not be added to the “emails” array because this email does already exist in the array.
- To add another email:
- db.users.update({"_id" : ObjectId("4b2d75476cc613d5ee930164")},
- {"$addToSet" : {"emails" : "joe@hotmail.com"}})
- This email will be added because the same email does not exist in the array.

- Both $push and $addToSet will fail if the key value pair exists with a non array value.

The Aggregation Framework

- The aggregation framework lets you transform and combine documents in a collection to do more complex analysis on your documents.
- It supports:
- Filtering
- Projecting
- Grouping
- Sorting
- Limiting
- Skipping

The Aggregation Framework

- With an article collection:
- The Aggregation Framework allows you to:
- Project the authors out of each article document.
- Group the authors by name,
- count the number of occurrences.
- Sort the authors by the occurrence count, descending.
- Limit results to the first five.

MongoDB Demonstration

- Pipeline Operations
- Each operator:
- Receives a stream of documents
- Applies some transformations
- Passes the result of the transformation
- Operators can be combined in any order and repeated as needed.
- $match
- $project
- $group
- $sort
- $limit
- $skip
- you could combine "$match", "$group", and then "$match" with different criteria

MongoDB Demonstration

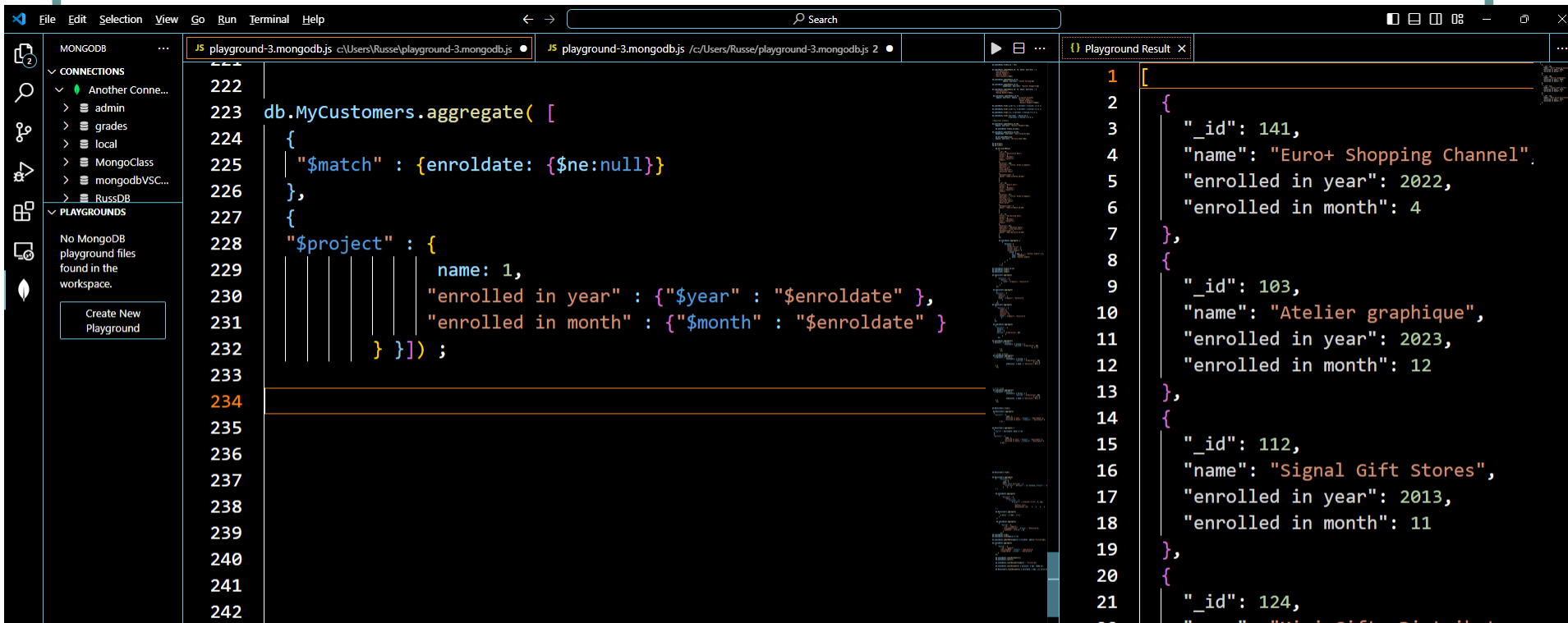
- $match
- $match is used to filter documents for applying aggregation on a subset of documents.
- It is better to put "$match" expressions as early as possible in the pipeline. $match limits the total number of documents in the aggregation pipeline, earlier $match operations minimize the amount of processing down the pipe.
- {$match : {“province" : "ONT"}}
- This expression filters documents where the value of the “province” field is “ONT”.
- "$match" can be used with the following operators
- "$gt“
- "$lt“
- "$in …

MongoDB Demonstration

- $project
- Project allows you to select a field, rename a field, and apply some operations on the selected fields from documents.
- It is used for
- field inclusion
- field exclusion,
- field names,
- and creating a virtual field based on existing fields
- db.articles.aggregate({"$project" : {"author" : 1})
- It returns documents containing one field.
- The “_id” field is also returned by default.
- You can use “$project” to exclude some fields:
- db.articles.aggregate({"$project" : {"author" : 1, "_id" : 0}})
- The above command excludes the “_id” field from the result.

Filtering documents and fields

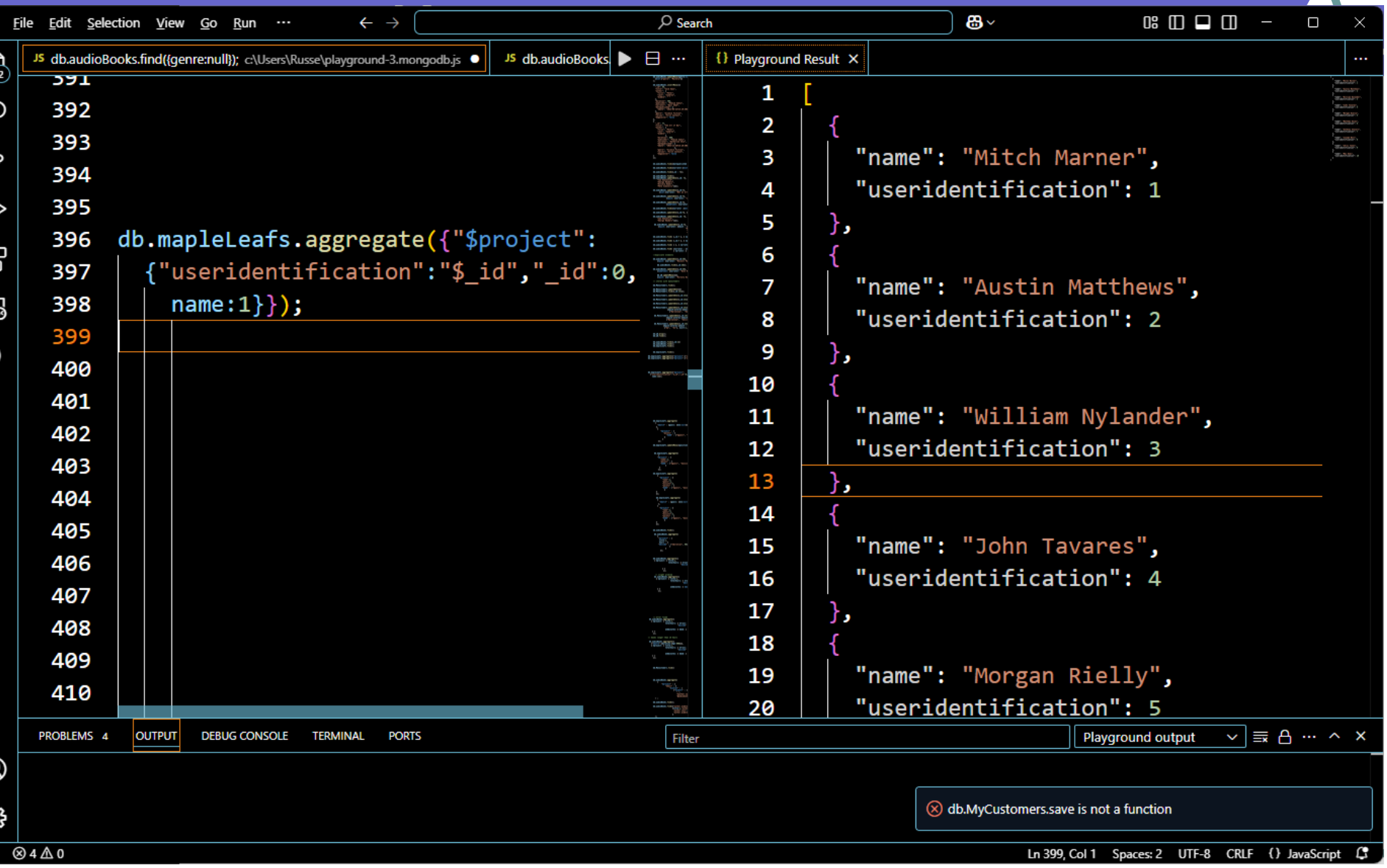
- $match used to filter out documents
- $project used to filter out fields and add fields from the matched documents.



MongoDB Demonstration

- Rename a Field
- You can use “$project” to rename a field:
- db.users.aggregate({"$project" : {"userId" : "$_id", "_id" : 0}})
- { "result" : [
- { "userId" : ObjectId("50e4b32427b160e099ddbee7")},
- { "userId" : ObjectId("50e4b32527b160e099ddbee8")}
- …],
- "ok" : 1 }
- The "$fieldname“ syntax is used to rename a field: “userId”: “$_id”
- The content of the “_id” field is projected for “userId.
- In this example, the field “_id” will be projected as “userId”.
- We need to exclude the “_id” from the document to avoid having duplicate “_id” fields.
- You can use this technique to project multiple copies of a field.

Projection


File Edit Selection View Go Run
Search
db.audioBooks.find({genre:null}); c:\Users\Russe\playground-3.mongodb.js
db.audioBooks.
Playground Result
391
392
393
394
395
396 db.mapleLeafs.aggregate({"$project":
397 {"useridentification":"$_id","_id":0,
398 name:1}});
399
400
401
402
403
404
405
406
407
408
409
410
[
{
"name": "Mitch Marner",
"useridentification": 1
},
{
"name": "Austin Matthews",
"useridentification": 2
},
{
"name": "William Nylander",
"useridentification": 3
},
{
"name": "John Tavares",
"useridentification": 4
},
{
"name": "Morgan Rielly",
"useridentification": 5
PROBLEMS 4
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter
Playground output
db.MyCustomers.save is not a function
4 0
Ln 399, Col 1
Spaces: 2
UTF-8
CRLF
JavaScript

MongoDB Demonstration

- Mathematical Expressions

Operation	Expression	Description
Add	"$add" : [expr1[, expr2, ..., exprN]]	Takes one or more expressions and adds them together.
Subtract	"$subtract" : [expr1, expr2]	Takes two expressions and subtracts the second from the first.
Multiplication	"$multiply" : [expr1[, expr2, ..., exprN]]	Takes one or more expressions and multiplies them together.
Division	"$divide" : [expr1, expr2]	Takes two expressions and divides the first by the second.
Mod	"$mod" : [expr1, expr2]	Takes two expressions and returns the remainder of dividing the first by the second.

MongoDB Demonstration

- Mathematical Expression Example
- We want to add bonus to the employee's salary when displaying the total pay.
- db.employees.aggregate(
- {
- "$project" : {
- "totalPay" : {
- "$add" : ["$salary", "$bonus"]
- }
- }
- })

MongoDB Demonstration

- More Complex Expressions
- Assume that we want to deduct an amount from the total pay:
- db.employees.aggregate(
- {
- "$project" : {
- "totalPay" : {
- "$subtract" : [{"$add" : ["$salary", "$bonus"]}, "$401k"]
- }
- }
- })

$add with mapleLeafs

- Goals and Assists are stored, points can always be calculated so they are not stored.
- db.mapleLeafs.aggregate(
- { "$project" : {
- "name":1,
- "goals":1,
- "assists":1,
- "points" : {
- "$add" : ["$goals", "$assists"] }
- } });

$divide with audioBooks

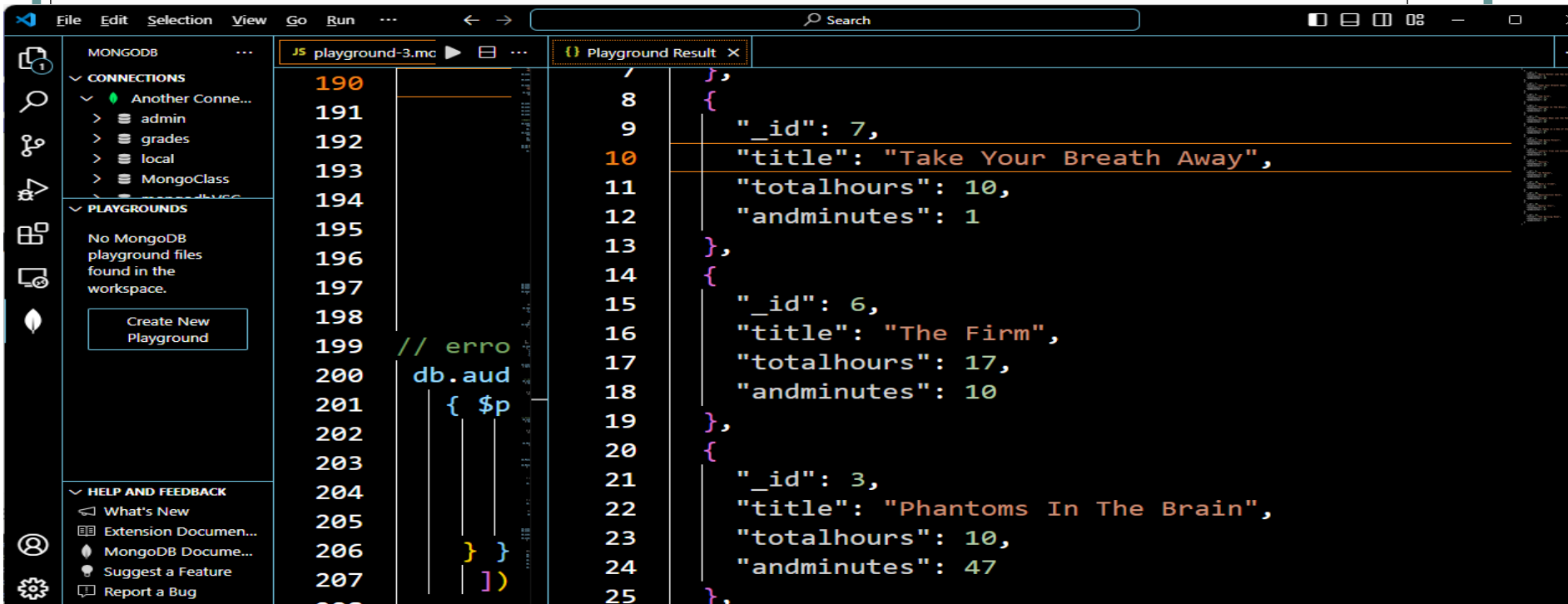
- Listening time for audio books is stored as minutes, some may want this information in hours
- db.audioBooks.aggregate(
- {
- "$project" : {
- title:1,
- hours : {
- "$divide" : ["$duration", 60] }
- } });

Including $trunc and $mod

Mongo Lecture 11 with Rounding.t

File Edit View

```
db.audioBooks.aggregate([
  { $project: { title:1,
                totalhours: { $trunc: [ {
                              "$divide" : ["$duration", 60]
                                                }, 0 ] },
                andminutes: { $mod: [ "$duration", 60 ] }
   } }
    ]);
```



MongoDB Date

- Date Expression
- There are many time-based aggregations:
- The total pay in a year
- Total sales in a month
- There are operators to extract date information. These operators take a date expression and return a number.
- "$year", "$month", "$week",
- "$dayOfMonth",
- "$dayOfWeek",
- "$dayOfYear",
- "$hour", "$minute", and "$second“
- Date operations can be applied on fields with the date type.

MongoDB Demonstration

Date Expression Example

The following returns the month that the employee was hired:

```
db.employees.aggregate(
  {
    "$project" : {
      "hiredIn" : {"$month" : "$hireDate"}
    } })
```

The following command calculates the number of years that the employee has been working in the company:

```
db.employees.aggregate(
{   "$project" : {
      "tenure" : {
        "$subtract" : [{"$year" : new Date()}, {"$year" : "$hireDate"}]
      }   }   }
)
```

MongoDB Demonstration

- String Expressions

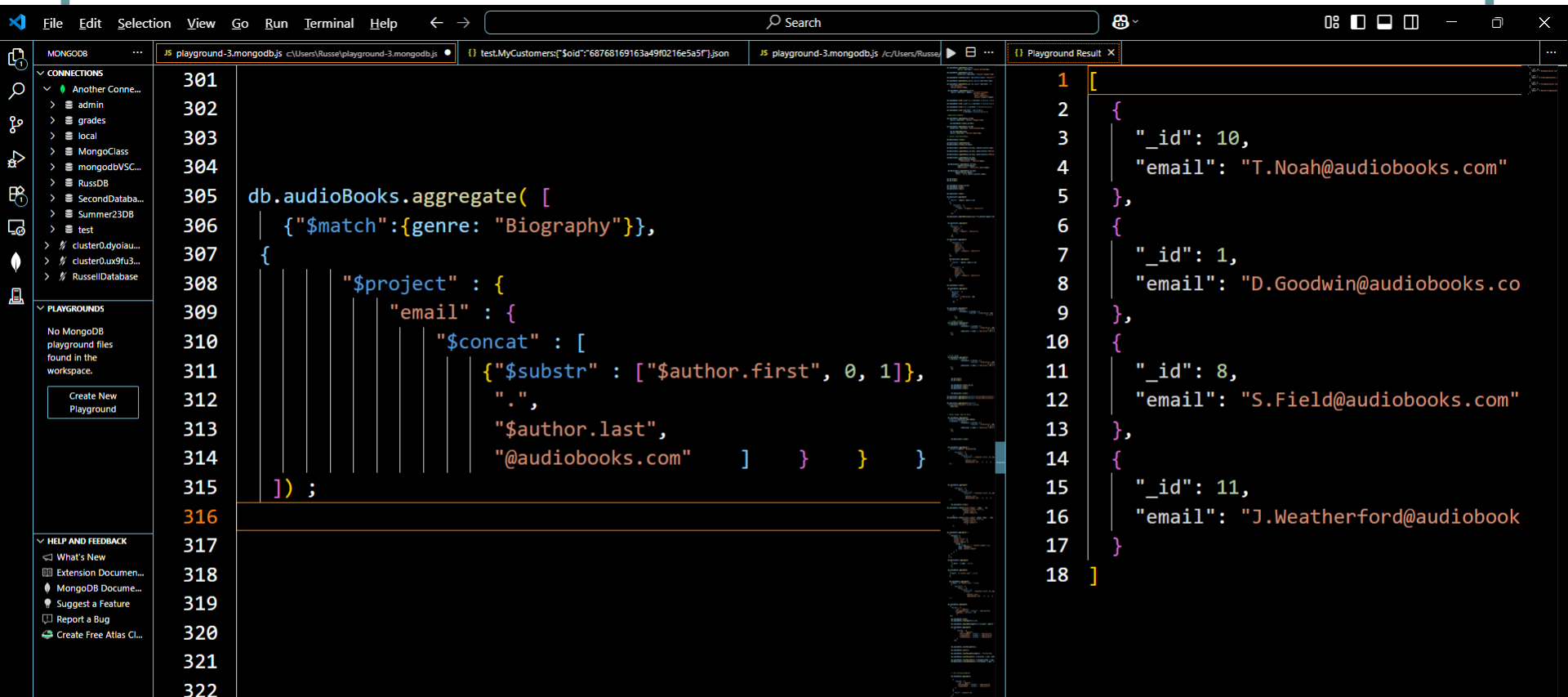
Operation	Expression	Description
substring	"$substr" : [expr, startOffset, numToReturn]	The startOffset and numToReturn values are measured in bytes.
concatenation	“$concat" : [expr1[, expr2, ..., exprN]]	Concatenates given string expressions.
lowercase	"$toLower" : expr	Returns the string in lower case.
uppercase	"$toUpper" : expr	Returns the string in upper case.

MongoDB Demonstration

- String Expression Example
- We want to generate email addresses with the following format:
- f.last@email.com
- f is the first letter of the first name, last is the last name added to “@email.com”
- db.employees.aggregate(
- {
- "$project" : {
- "email" : {
- "$concat" : [
- {"$substr" : ["$firstName", 0, 1]},
- ".",
- "$lastName",
- "@email.com"] } } }
-)

Pipeline

- $match
- $project



MongoDB Demonstration

- Comparison Expressions

Operation	Expression	Description
Comparison	"$cmp" : [expr1, expr2]	Returns • 0: if two expressions are equal • 1: if expr1 is greater than expr2
Comparison (case insensitive)	"$strcasecmp" : [string1, string2]	Case insensitive comparison between string1 and string2
Comparison	"$eq"/"$ne"/"$gt"/"$gte"/"$lt"/"$lte" : [expr1, expr2]	Returns true or false

MongoDB Demonstration

- Logical Operators

Operation	Expression	Description
Not	"$not" : expr	Returns the opposite Boolean of expr. NOT (true) → false NOT (false) → true
And	"$and" : [expr1 [, expr2 …, exprN]]	Returns true if all expressions are true
Or	"$or« : [expr1 [, expr2 …, exprN]]	Returns true if at least one expression is true

MongoDB Demonstration

- Control Statements

Operation	Expression	Description
Conditional Expression	"$cond" : [booleanExpr, trueExpr, falseExpr]	If booleanExpr is true: • trueExpr is returned Otherwise: • falseExpr is returned
IsNull	“$ifNull" : [expr, replacementExpr]	If the value of expr is not null: • expr is returned Otherwise: • replacementExpr is returned

MongoDB Demonstration

- Complex Projection Example

```
db.students.aggregate(
  {
    "$project" : {
      "grade" : {
        "$cond" : [
          "$teachersPet",
          100, // if
              { // else
                "$add" : [
                  {"$multiply" : [.1, "$attendanceAvg"]},
                  {"$multiply" : [.3, "$quizzAvg"]},
                  {"$multiply" : [.6, "$testAvg"]}
                ]
              }
          ]   }   }   })
```

MongoDB Demonstration

- “$group”
- • The “$group” operator groups documents based on a certain field.
- {"$group" : {"_id" : "$day"}}
- It groups documents based on the “$day” field.
- {"$group" : {"_id" : "$grade"}}
- It groups documents based on the “$grade” field.
- {"$group" : {"_id" : {“province" : "$province", "city" : "$city"}}}
- It groups documents based on multiple fields: “$province” and then “$city”.

MongoDB Demonstration

- Grouping Operators
- Aggregation expressions:
- $sum
- $avg
- $min
- $max
- $first
- $last

MongoDB Demonstration

- “$sum”
- The “$sum” operator returns the total of a certain documents’ field in a group.
- db.sales.aggregate(
- {
- "$group" : {
- "_id" : "$country",
- "totalRevenue" : {"$sum" : "$revenue"}
- }
- })
- The above query returns the total revenue of documents in each country

MongoDB Demonstration

- “$avg”
- The “$avg” operator returns the average value of a certain documents’ field in n a group.
- db.sales.aggregate(
- {
- "$group" : {
- "_id" : "$country",
- "averageRevenue" : {"$avg" : "$revenue"},
- "numSales" : {"$sum" : 1}
- }
- })
- It returns the average revenue and the number of sales for each country. Each document is assumed to be a sale document (a sale transaction).

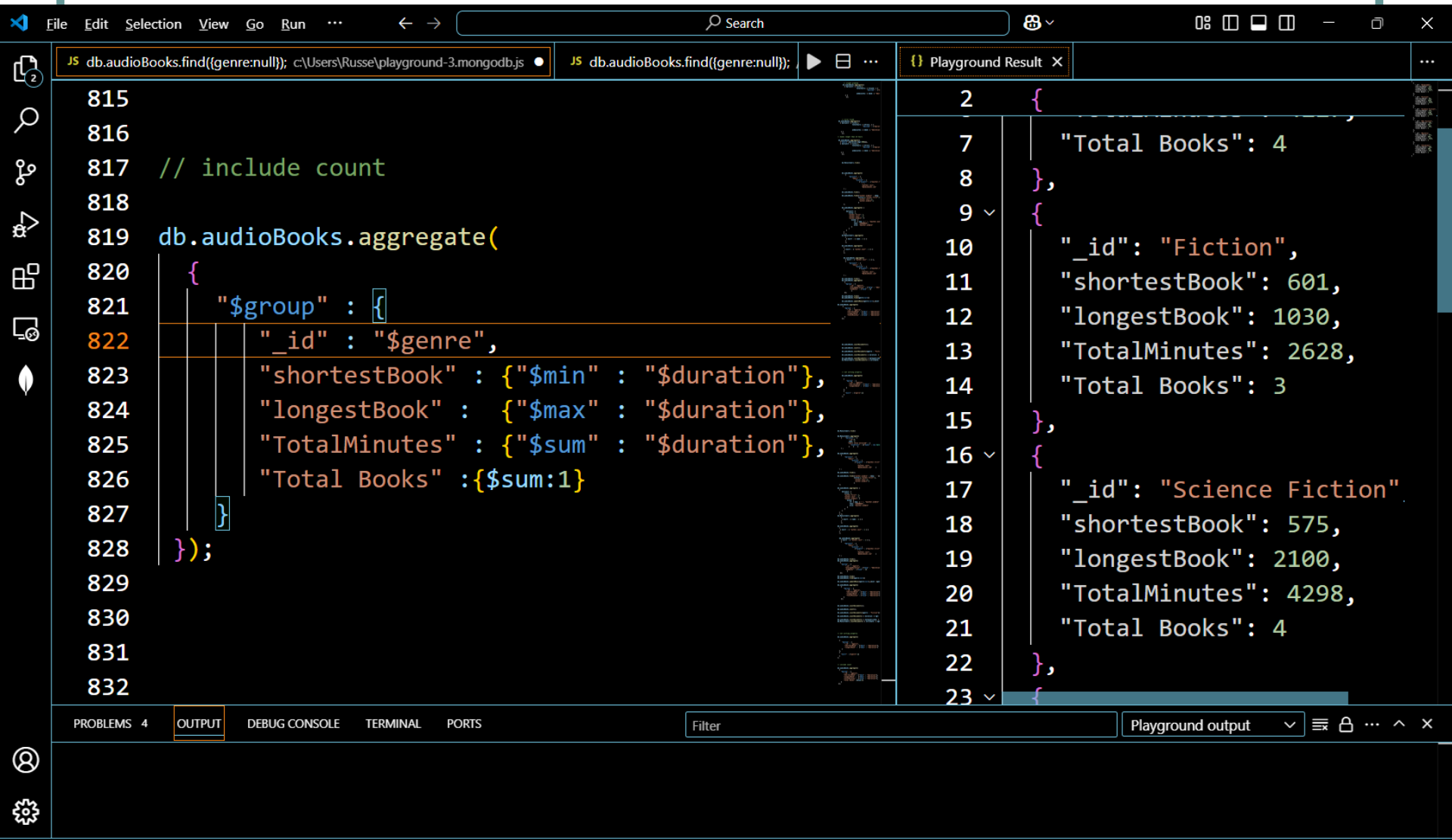
MongoDB Demonstration

- “$max”/“$min”
- The “$max”/”$min” operator returns the greatest/smallest value of a document’s field among all documents in a collection or in a group.
- db.scores.aggregate(
- {
- "$group" : {
- "_id" : "$grade",
- "lowestScore" : {"$min" : "$score"},
- "highestScore" : {"$max" : "$score"}
- }
- })
- The above query finds the maximum and minimum scores for each group of grades.

MongoDB Demonstration

- “$first”/“$last”
- When data is sorted, the “$first” and “$last” operators can be used to find the smallest and greatest values in a group of documents.
- If the data is not sorted it is not efficient to use “$first” and “$last” operators to find the min and max because a sorting operation is needed.
- db.scores.aggregate(
- {
- "$sort" : {"score" : 1}
- },
- {
- "$group" : {
- "_id" : "$grade",
- "lowestScore" : {"$first" : "$score"},
- "highestScore" : {"$last" : "$score"}
- } })

$Group


File Edit Selection View Go Run
Search
db.audioBooks.find({genre:null}); c:\Users\Russe\playground-3.mongodb.js
db.audioBooks.find({genre:null});
Playground Result
// include count
db.audioBooks.aggregate(
{
"$group" : {
"_id" : "$genre",
"shortestBook" : {"$min" : "$duration"},
"longestBook" : {"$max" : "$duration"},
"TotalMinutes" : {"$sum" : "$duration"},
"Total Books" :{$sum:1}
}
});
"Total Books": 4
},
{
"_id": "Fiction",
"shortestBook": 601,
"longestBook": 1030,
"TotalMinutes": 2628,
"Total Books": 3
},
{
"_id": "Science Fiction"
"shortestBook": 575,
"longestBook": 2100,
"TotalMinutes": 4298,
"Total Books": 4
},
PROBLEMS 4
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter
Playground output

MongoDB Demonstration

- “$limit”
- The “$limit” operator returns first n documents as a result of a query.
- The syntax:
- “$limit" : integer
- “$skip”
- The “$skip” operator discards the first n documents from the query result.

MongoDB Demonstration

- Count
- The count is a simple aggregation operator that returns the number of documents in a collection.
- db.book.count()
- 0
- db.book.insert(“title”: “Blue Sky”)
- db.book.count()
- 1
- You can also count the result of a query using the count function.
- db.book.insert(“title”: “Vanilla Sky”)
- db.book.count() -> total number of documents
- 2
- db.book.count({“title”: “Blue Sky”}) –> documents with title “Blue Sky
- 1

Count is Deprecated – Use countDocuments

- db.audioBooks.countDocuments();
- 1 16
- db.audioBooks.countDocuments({genre : "Fiction"});
- 1 3
- db.audioBooks.countDocuments({ datepublished: { $gt: new Date('01/01/2018') } });
- 1 4
- But it was not 4 – check your results

Lab

- The final Lab has been been posted
- Review next week in lecture period
- Final test week of December 8 in lecture period
- A review document will be released on Friday and used in the lecture period review
- Attempt some of the questions

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