**Assignment 2**

**MongoDB Create and update operations**

Please work in **groups** to complete this assignment. This assignment is worth **15% of the total course grade** and will be evaluated through your written submission. Each day being late will result in 10% mark penalty.

Please submit the following files through Blackboard:

* Assign2.docx

1. Add this declaration on the top of your Assign2.docx file.

We, Group 1 **[Kabir Narula] [Maksym Volkovynskyi**] ,declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) **or distributed to other students.**

**Specify below what each member has done towards the completion of this work:**

|  |  |  |
| --- | --- | --- |
|  | **Name** | **Task(s)** |
| **1-** | **Kabir Narula** | **Everything** |
| **2-** | **Maksym Volkovynskyi** | **Everything** |

**We did each question individually so that we can practice better and then later compared and matched our answers…**

Instructions

Answer each of the following questions with a mongodb command, and the corresponding output. You can use mongosh within mongodb compass.

# marking:

10 questions, 3pts each. Total: 30pts

# Part 2: insert and update

1. Use insert() to insert the following book in the books collection. Authors’s value is an array. Categories’s values is an array.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| \_id | title | isbn | publishedDate | authors | categories |
| 20 | MongoDB: The Definitive Guide | 9781491954461 | 2019-12-01 | Kristina Chodorow | Next Generation Databases |

Command:

db.books.insertOne({

\_id: 20,

title: 'MongoDB: The Definitive Guide',

isbn: '9781491954461',

publishedDate: new Date('2019-12-01'),

authors: ['Kristina Chodorow'],

categories: ['Next Generation Databases']

})

Output:

A screen shot of a computer code

Description automatically generated

1. Add a field named pageCount equal 544 to book id 20. Use the $set modifier. Show \_id and pageCount fields for book id 20.

Command:

db.books.updateOne(

{\_id:20},

{$set:{pageCount:544}}

)

Output:

A screen shot of a computer code

Description automatically generated

1. Update the pageCount for bookid 20 to 500.

Show the \_id and pageCount fields for book id 20 after the update.

Command:

db.books.updateOne(

{\_id:20},

{$set: {pageCount: 500}}

)

Output:

A screen shot of a computer code

Description automatically generated

1. Update the title for book id 20 by adding the text “Third edition”. Use update() function, the first argument is to identify the document to update, the second argument holds the update. For the second argument, use aggregation pipeline and enclose it with square brackets []. use $set and $concat. $title holds the current value in the key “title”.

Show the \_id and title fields for book id 20 after the update.

reference: <https://stackoverflow.com/questions/23868963/append-a-string-to-the-end-of-an-existing-field-in-mongodb>

Command:

db.books.updateOne(

{\_id:20},

[{$set:{

title: {$concat:['$title', ' Third edition']}

}}]

)

Output:



1. Add a value “NoSQL databases” to the categories field for book id 20. Use $addToSet modifier.

Show the \_id and the categories fields for book id 20 after the update.

Command:

db.books.updateOne(

{\_id:20},

{

$addToSet:{categories: 'NoSQL databases'}

}

)

Output:

A screen shot of a computer code

Description automatically generated

1. Push the following categories “mongodb”, “advanced databases” in the categories field for book id 20. Use $push and $each modifiers.

Show the \_id and categories fields for book id 20 after the update.

Command:

db.books.updateOne(

{\_id:20},

{

$push: {categories: {$each: [

'mongodb', 'advanced databases'

]}}

}

)

Output:

A screen shot of a computer code

Description automatically generated

1. Pull the element “advanced DB” from the categories array for book id 20.

Show the \_id and categories fields for book id 20 after the update.

Command:

db.books.updateOne(

{\_id:20},

{

$pull:{categories: 'advanced DB'}

}

)

Output:

A screen shot of a computer code

Description automatically generated

1. Roll a new field named “pubyear” for all books in the collection and assign the value as the year extracted from the publishedDate field. Use update() function, the second argument should be enclosed with square brackets to refer to aggregation pipeline.

Show the \_id, publishedDate and pubyear fields for all books after the update.

Command:

db.books.updateMany(

{},

[

{$set: {pubyear: {

$year: '$publishedDate'

}}}

]

)

Output:

A screenshot of a computer code

Description automatically generated A screenshot of a computer program

Description automatically generated

A computer code with numbers and digits

Description automatically generated

1. Remove the long description fields from all books.

show one document of the books collection after the update.

Command:

db.books.updateMany(

{},

{

$unset: {longDescription: ''}

}

)

Output:

A screenshot of a computer

Description automatically generated

1. Add an array comments to book id 20. Push the two comments for the book together in one update command. Show documents 20 after the update.

|  |  |
| --- | --- |
| Book id | comments |
| 19 | Name: "Joe"  Content: "great content"  Rating: 5 |
| 20 | Name: "Joe"  Content: "lacks some topics"  Rating: 3 |
| Name: "Sam"  Content: "excellent book. Contains illustratives"  rating: 5 |

Command:

db.books.bulkWrite([

{

updateOne: {

filter: {\_id: 19},

update: {

$set: {

comments: [

{

name: 'Joe',

content: 'great content',

rating: 5

}

]

}

}

}

},

{

updateOne: {

filter: {\_id: 20},

update: {

$set: {

comments: [

{

name: 'Joe',

content: 'lacks some topics',

rating: 3

},

{

name: 'Sam',

content: 'excellent book. Contains illustratives',

rating: 5

}

]

}

}

}

}

])

Output:

A screenshot of a computer program

Description automatically generated A screen shot of a computer code

Description automatically generated