**Assignment 2**

**MongoDB Aggregation**

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, Mansoor Ahmad Zafar 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.**

1. Specify what each member has done towards the completion of this work:

|  |  |  |
| --- | --- | --- |
|  | Name | Task(s) |
| 1 | Mansoor Ahmad Zafar | Everything |
| 2 |  |  |
| 3 |  |  |

Instructions

Answer each of the following questions with a mongodb command, and the corresponding output.

# marking:

10 questions, 4pts each. Total: 40pts

# Part 3: aggregate

1. Show the book ids, title and publication year of all books. Extract the publication year from publishedDate. Do NOT use the pubyear key that exists in the document. Use aggregate() function and $project operator.

Show \_id, title, and pubyear for each book. Sort the books in ascending order of their \_id.

**Command:**

db.books.aggregate

(

{"$project" :

{

"\_id" : 1,

"title" : 1,

"pubyear" : {"$year" : "$publishedDate"}

}

}

)

**Output:**

**** **A screen shot of a computer code

Description automatically generated** A screen shot of a computer code

Description automatically generated A computer screen shot of code

Description automatically generated A computer screen shot of a program

Description automatically generated

1. Show the book ids, title and publication year for those published after 2010. Extract the publication year from publishedDate. Do NOT use the pubyear key that exists in the document. Use aggregate() function, $project operator to select only title and year of publication in pubyear key and $match to keep only those with year of publication greater than or equal 2011.

**Command:**

db.books.aggregate

(

{"$project" :

{

"\_id" : 1,

"title" : 1,

"pubyear" : {"$year" : "$publishedDate"}

}

},

{"$match" : {"pubyear" : {"$gt" : 2010} } }

)

output:

[

{ \_id: 2, title: 'Android in Action, Second Edition', pubyear: 2011 },

{ \_id: 9, title: 'Griffon in Action', pubyear: 2012 },

{ \_id: 10, title: 'OSGi in Depth', pubyear: 2011 },

{ \_id: 17, title: 'MongoDB in Action', pubyear: 2011 },

{ \_id: 3, title: 'Specification by Example', pubyear: 2011 },

{ \_id: 20, title: 'MongoDB: The Definitive Guide', pubyear: 2019 }

]

1. Show all books with the title in capital letters. Use aggregate() function, $project and $toUpper operators. Sort the books in ascending order of their \_id.

reference: <https://docs.mongodb.com/manual/reference/operator/aggregation/toUpper/>

**Command:**

db.books.aggregate

(

{"$project" :

{

"title" : {"$toUpper" : "$title"}

}

},

{"$sort" : {"\_id" : 1}}

)

**Output:**

A screen shot of a computer code

Description automatically generated A screen shot of a computer program

Description automatically generated A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Show the number of books published per each year. Use aggregate(), $group and $sum operators.

**Command:**

db.books.aggregate

(

{"$group" :

{

"\_id" : "$pubyear",

"nbBooks" : {"$sum" : 1}

}

}

)

Output:

[

{ \_id: 2012, nbBooks: 1 },

{ \_id: 2010, nbBooks: 3 },

{ \_id: 1998, nbBooks: 1 },

{ \_id: 1999, nbBooks: 1 },

{ \_id: 2019, nbBooks: 1 },

{ \_id: 1997, nbBooks: 1 },

{ \_id: 2009, nbBooks: 3 },

{ \_id: 2008, nbBooks: 4 },

{ \_id: 2011, nbBooks: 4 }

]

For more examples, refer to:

<https://docs.mongodb.com/manual/reference/operator/aggregation/sum/>

1. Show the book publication status and the number of books per status. Exclude null status. Use $group and $match in the aggregate() pipeline.

**Command:**

db.books.aggregate

(

{"$match" : {"status" : {"$exists" : true}}},

{"$group" :

{

"\_id" : "$status",

"nbBooks" : {"$sum" : 1}

}

}

)

output:

{ "\_id" : "PUBLISH", "nbBooks" : 18 }

1. Show the book title and the number of comments given for each book. Exclude the books that do not have any comments. Show the book title and the number of comments. Use aggregate, $project, $cond, $isArray and $size in your command.

Some books do not have comments field, $isArray checks the type of the comments field, if it does not exist, $isArray is false and you must assign the value 0 for the number of comments.

**Command:**

db.books.aggregate

({"$project" :

{

"\_id" : 1,

"title" : 1,

"nbComments" : {"$cond" : [{"$isArray" : "$comments"}, {"$size" : "$comments"}, 0 ]}

}

})

output:

[

{

\_id: 18,

title: 'Distributed Application Development with PowerBuilder 6.0',

nbComments: 2

},

{

\_id: 19,

title: 'Jaguar Development with PowerBuilder 7',

nbComments: 2

},

{ \_id: 20, title: 'MongoDB: The Definitive Guide', nbComments: 1 }

]

1. Show the books with isbn and title concatenated. Use $project and $concat operators in the aggregate() function pipeline. Sort the books in ascending order of their \_id.

**Command:**

db.books.aggregate

(

{"$project" :

{

"\_id" : 1,

"book" : { "$concat" : [{"$toString" : "$isbn"}, ", ", "$title"]}

}

}

)

output:

[

{ \_id: 1, book: '1933988673, Unlocking Android' },

{ \_id: 2, book: '1935182722, Android in Action, Second Edition' },

{ \_id: 3, book: '1617290084, Specification by Example' },

{ \_id: 4, book: '1933988746, Flex 3 in Action' },

{ \_id: 5, book: '1935182420, Flex 4 in Action' },

{ \_id: 6, book: '1933988312, Collective Intelligence in Action' },

{ \_id: 7, book: '1933988320, Zend Framework in Action' },

{ \_id: 8, book: '1933988797, Flex on Java' },

{ \_id: 9, book: '1935182234, Griffon in Action' },

{ \_id: 10, book: '193518217X, OSGi in Depth' },

{ \_id: 11, book: '1933988509, Flexible Rails' },

{ \_id: 13, book: '1933988762, Hello! Flex 4' },

{ \_id: 14, book: '1884777384, Coffeehouse' },

{

\_id: 15,

book: '1933988592, Team Foundation Server 2008 in Action'

},

{

\_id: 16,

book: '1933988711, Brownfield Application Development in .NET'

},

{ \_id: 17, book: '1935182870, MongoDB in Action' },

{

\_id: 18,

book: '1884777686, Distributed Application Development with PowerBuilder 6.0'

},

{

\_id: 19,

book: '1884777864, Jaguar Development with PowerBuilder 7'

},

{ \_id: 20, book: '9781491954461, MongoDB: The Definitive Guide' }

]

**Unwind**

1. Find the average rating given for each book.

**Command:**

db.books.aggregate

(

{"$match" : {"comments" : {"$exists" : true}}},

{"$project" :

{

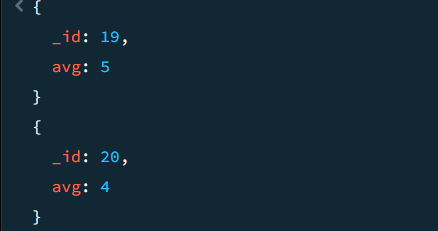
"avg" : {"$avg" : "$comments.Rating"}

}

}

)

**Output:**



1. Find the average rating given by each user.

**Command:**

db.books.aggregate

(

{"$unwind" : "$comments"},

{"$group" :

{

"\_id" : "$comments.Name",

"avg" : {"$avg" : "$comments.Rating"}

}

}

)

**Output:**

A computer screen shot of text

Description automatically generated

1. Show the user and its lowest and highest rating.

**Command:**

db.books.aggregate

(

{"$match" : {"comments" : {"$exists" : true}}},

{"$project" :

{

"lowRate" : {"$min" : "$comments.Rating"},

"highRate" : {"$max" : "$comments.Rating"}

}

}

)

**Output:**

