Name – Abhishek Raj

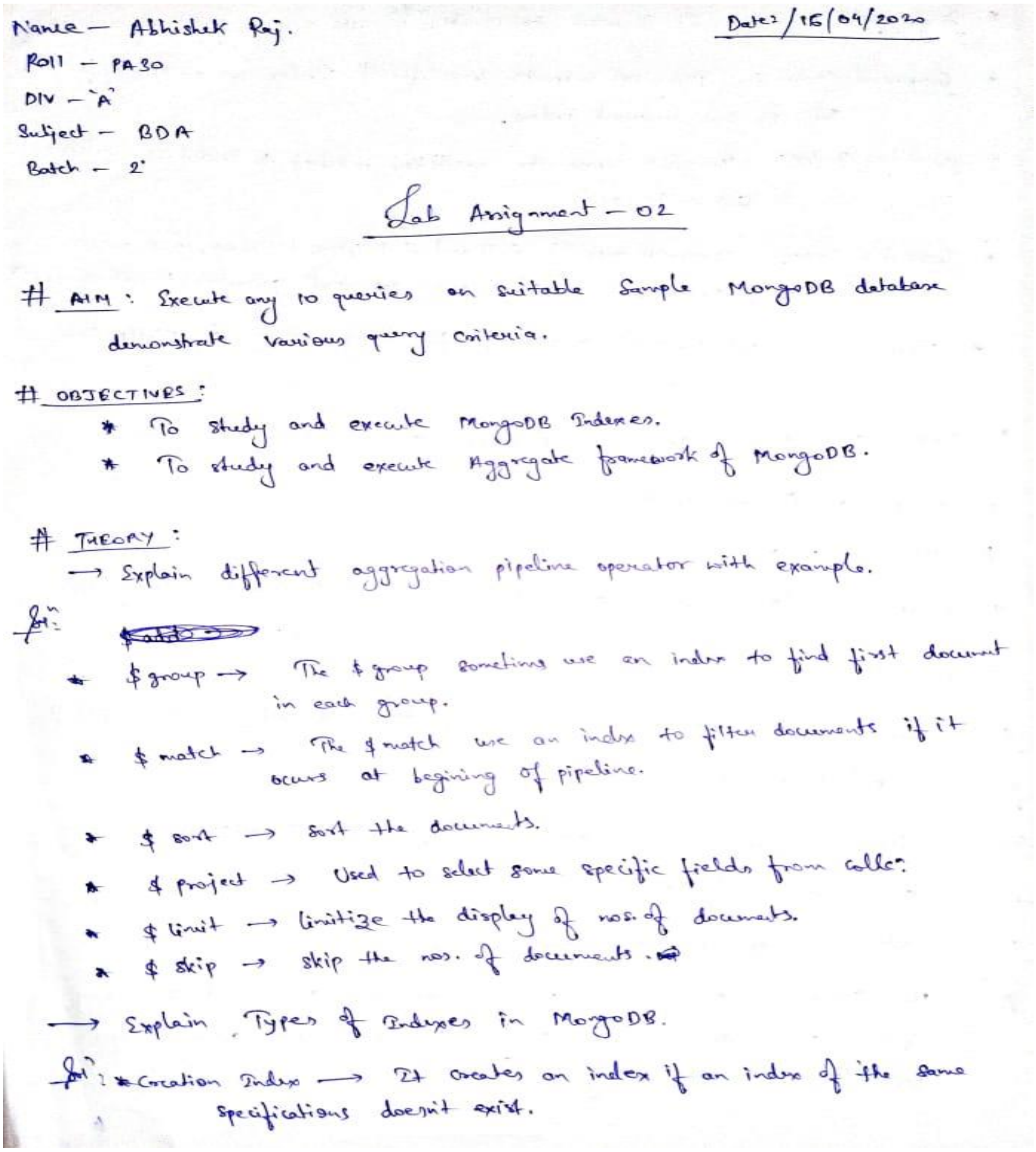
Roll No. – PA30

DIV – 'A'

Batch – 2

Subject – BDA

LAB ASSIGNMENT – 2



Name – Abhishek Raj. Date: /16/04/2020

Roll – PA30

DIV – 'A'

Subject – BDA

Batch – 2

Lab Assignment – 02

AIM: Execute any 10 queries on suitable Sample MongoDB database demonstrate various query criteria.

OBJECTIVES:

* To study and execute MongoDB Indexes.
* To study and execute Aggregate framework of MongoDB.

THEORY:

→ Explain different aggregation pipeline operator with example.

Solⁿ:

* $ group → The $ group sometime use an index to find first document in each group.
* $ match → The $ match use an index to filter documents if it occurs at begining of pipeline.
* $ sort → sort the documents.
* $ project → Used to select some specific fields from collecⁿ.
* $ limit → limitize the display of nos. of documents.
* $ skip → skip the nos. of documents.

→ Explain Types of Indexes in MongoDB.

Solⁿ: * Creation Index → It creates an index if an index of the same specifications doesn't exist.

* Single field Index: It is used to create index on the single field of document.

* Compound Index: MongoDB supports userdefined Index on multiple fields. So, It has compound index.

* Multikey Index: MongoDB uses the multiKey indexes to index the values sorted in array.

* Geospatial Index: MongoDB supports 2-D and 2-D sphere indexes. 2-D uses planar geometry and 2dsphere uses spherical geometry to return results.

* Text Index: It supports searching for string content in a Collection.

INPUT: Sample Collection → Employees and Books.Json

OUTPUT: Execution of all queries.

PLATFORM: Windows

CONCLUSION: Thus, learned about Aggregation and Indexing in MongoDB.

FAQs:

Q.1) Which pipeline operator is similar to where and having in MongoDB?

Sol: $project and $match.

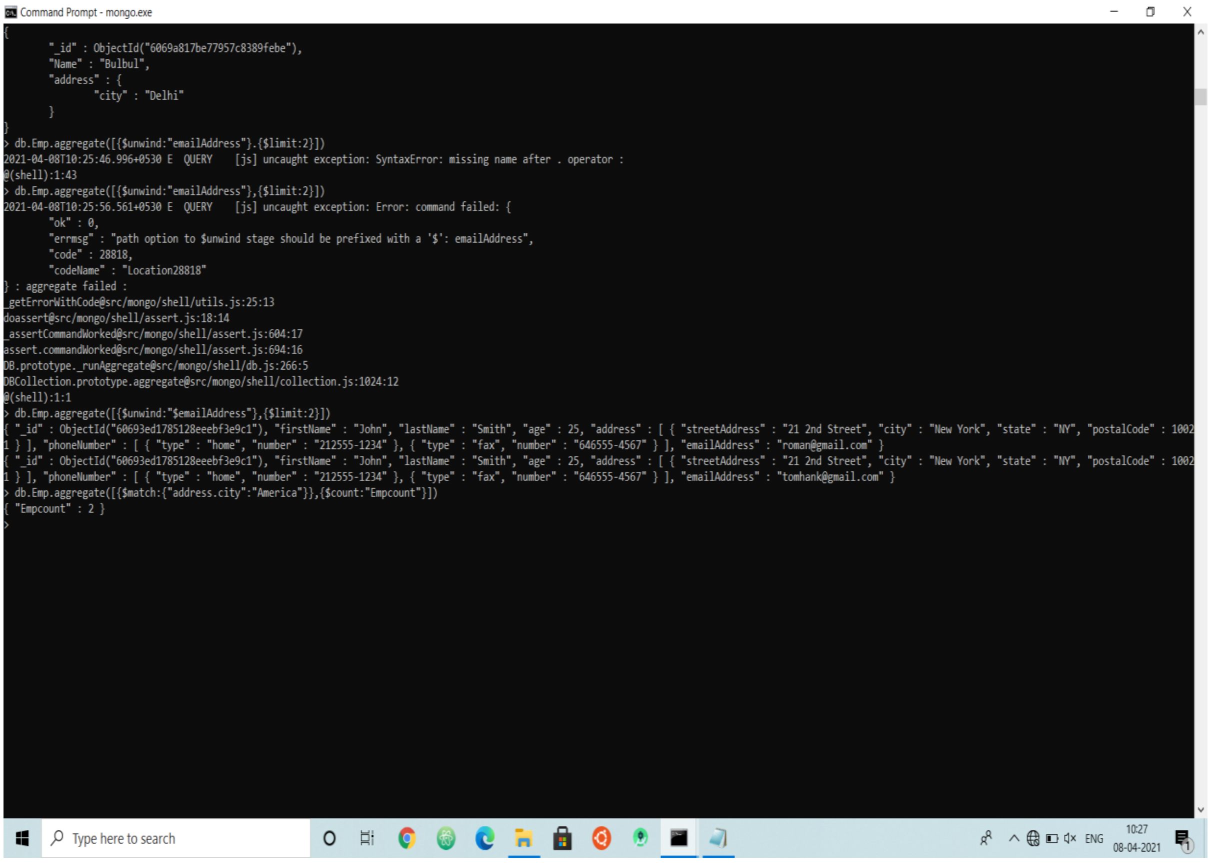
Q.2) How to know which indexes are defined over a Collection?

Sol: db.Collection.getIndexes().

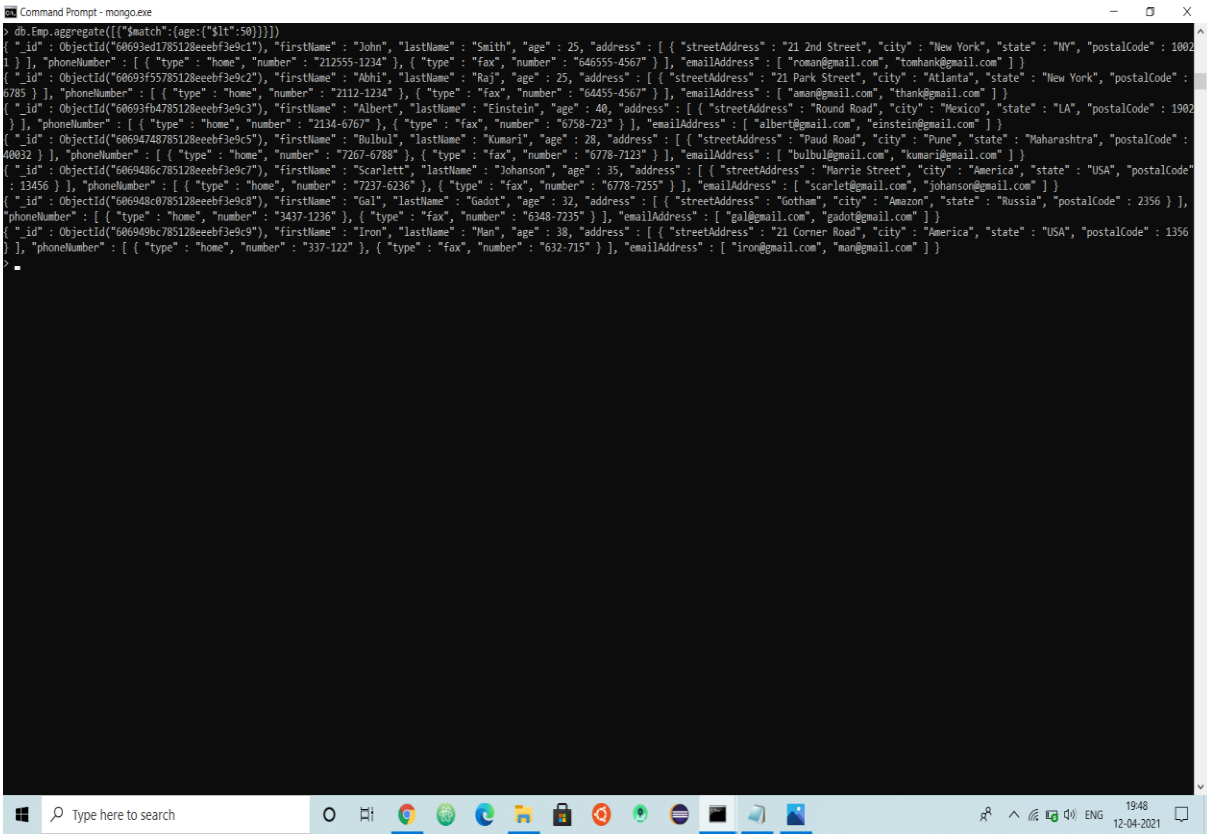
Q.3) What is application of ~~[illegible]~~ Unwind?

Sol: The $unwind operator is used to ~~Document~~ deconstruct an array field in a document and create separate output documents for each item in the array.

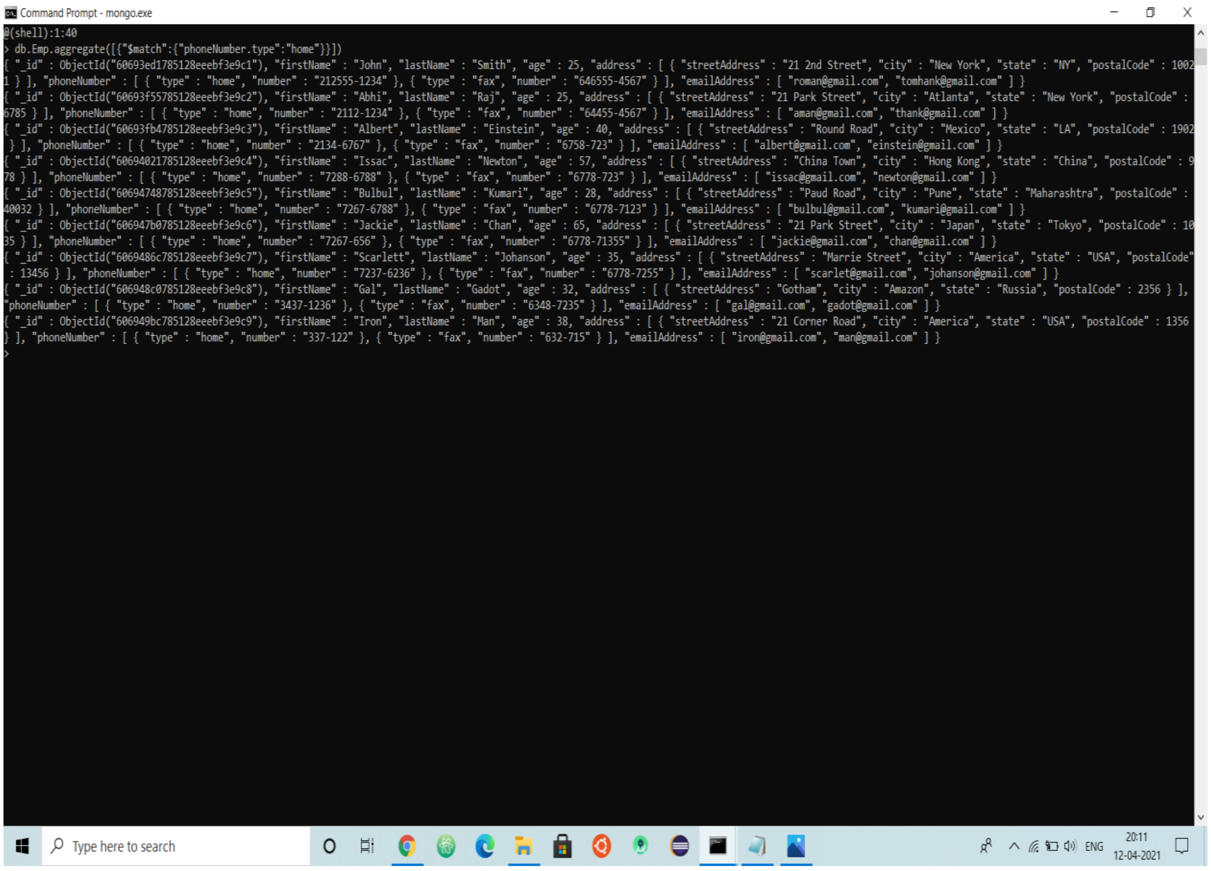
AGGREGATE QUERIES:


Command Prompt - mongo.exe

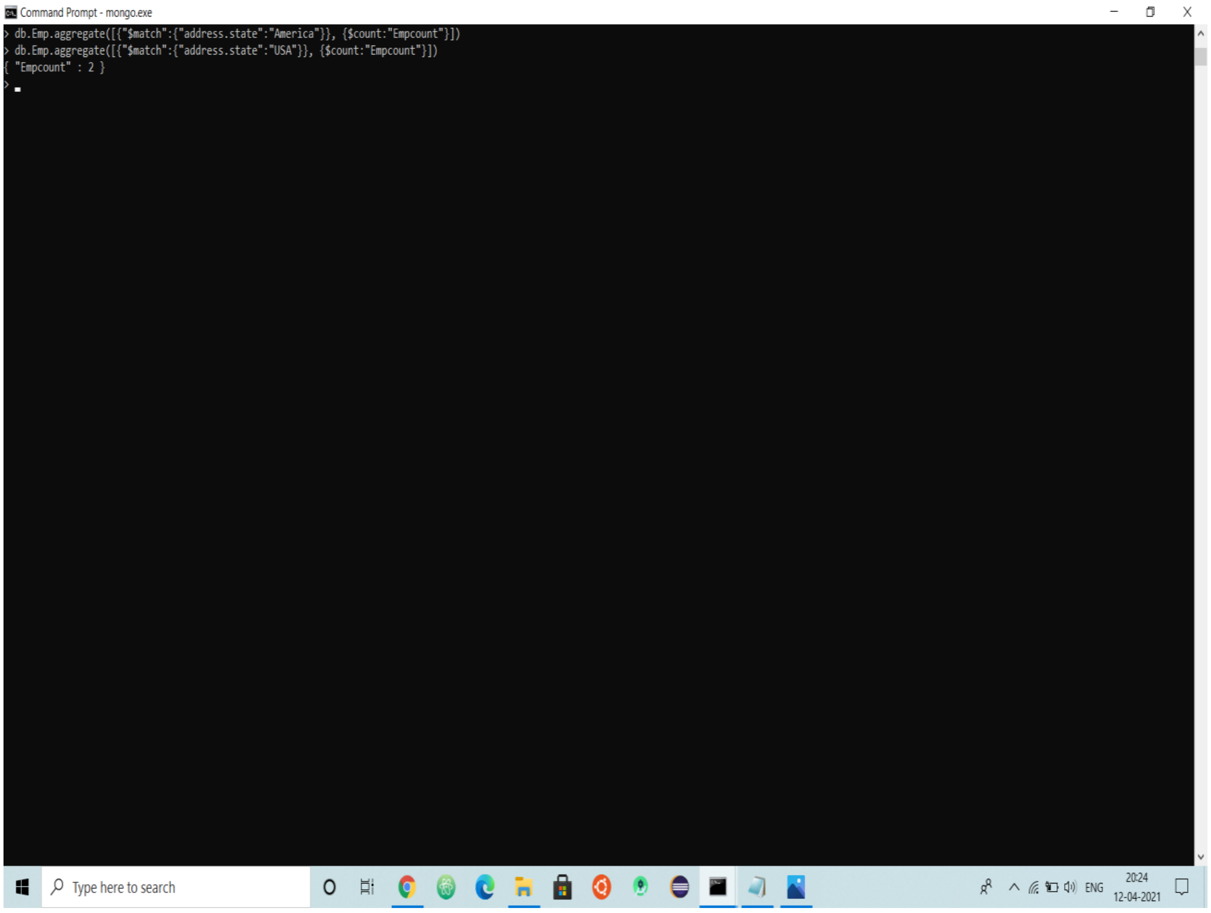
```
{
        "_id" : ObjectId("6069a817be77957c8389febe"),
        "Name" : "Bulbul",
        "address" : {
                "city" : "Delhi"
        }
}
> db.Emp.aggregate([{$unwind:"emailAddress"}.{$limit:2}])
2021-04-08T10:25:46.996+0530 E  QUERY    [js] uncaught exception: SyntaxError: missing name after . operator :
@(shell):1:43
> db.Emp.aggregate([{$unwind:"emailAddress"},{$limit:2}])
2021-04-08T10:25:56.561+0530 E  QUERY    [js] uncaught exception: Error: command failed: {
        "ok" : 0,
        "errmsg" : "path option to $unwind stage should be prefixed with a '$': emailAddress",
        "code" : 28818,
        "codeName" : "Location28818"
} : aggregate failed :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
doassert@src/mongo/shell/assert.js:18:14
_assertCommandWorked@src/mongo/shell/assert.js:604:17
assert.commandWorked@src/mongo/shell/assert.js:694:16
DB.prototype._runAggregate@src/mongo/shell/db.js:266:5
DBCollection.prototype.aggregate@src/mongo/shell/collection.js:1024:12
@(shell):1:1
> db.Emp.aggregate([{$unwind:"$emailAddress"},{$limit:2}])
{ "_id" : ObjectId("60693ed1785128eeebf3e9c1"), "firstName" : "John", "lastName" : "Smith", "age" : 25, "address" : [ { "streetAddress" : "21 2nd Street", "city" : "New York", "state" : "NY", "postalCode" : 1002
1 } ], "phoneNumber" : [ { "type" : "home", "number" : "212555-1234" }, { "type" : "fax", "number" : "646555-4567" } ], "emailAddress" : "roman@gmail.com" }
{ "_id" : ObjectId("60693ed1785128eeebf3e9c1"), "firstName" : "John", "lastName" : "Smith", "age" : 25, "address" : [ { "streetAddress" : "21 2nd Street", "city" : "New York", "state" : "NY", "postalCode" : 1002
1 } ], "phoneNumber" : [ { "type" : "home", "number" : "212555-1234" }, { "type" : "fax", "number" : "646555-4567" } ], "emailAddress" : "tomhank@gmail.com" }
> db.Emp.aggregate([{$match:{"address.city":"America"}},{$count:"Empcount"}])
{ "Empcount" : 2 }
>
```


Command Prompt - mongo.exe

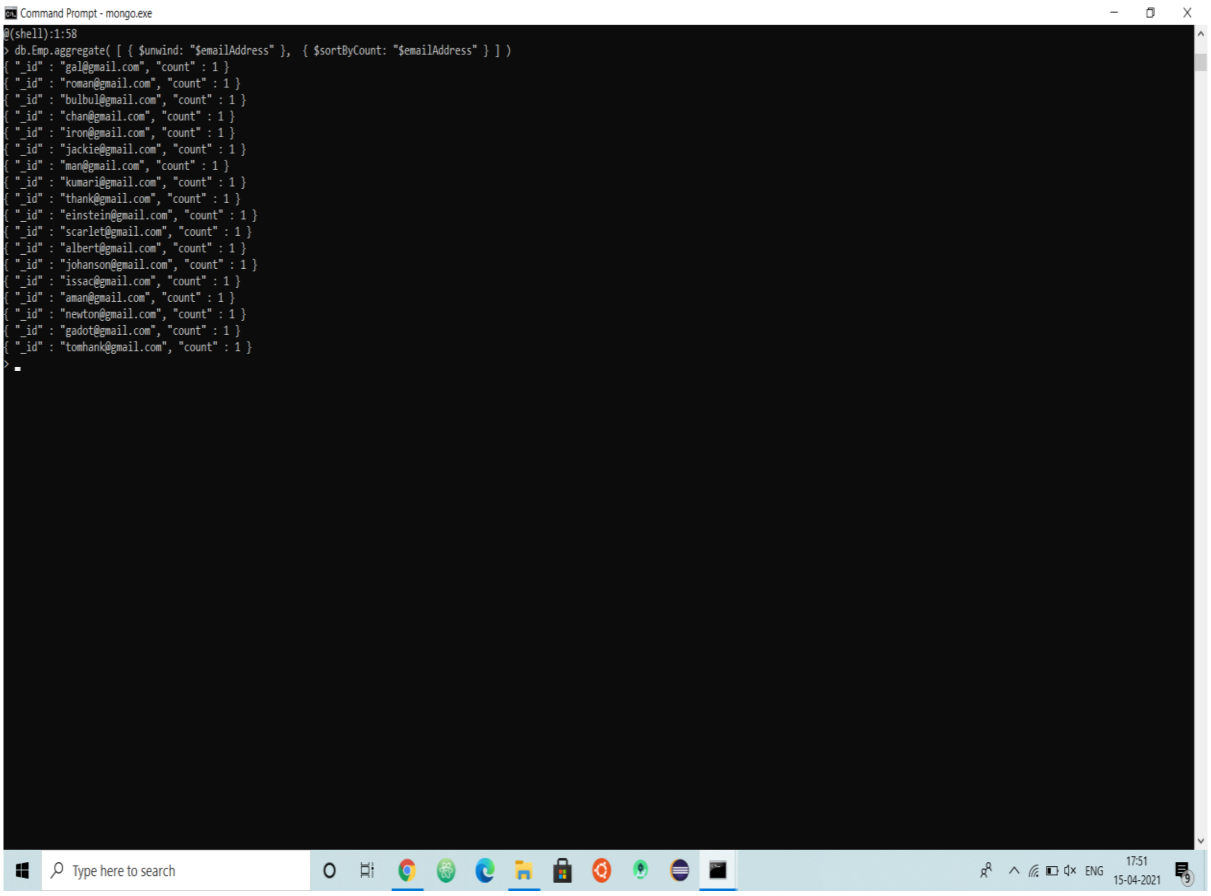
```
> db.Emp.aggregate([{"$match":{age:{"$lt":50}}}])
{ "_id" : ObjectId("60693ed1785128eeebf3e9c1"), "firstName" : "John", "lastName" : "Smith", "age" : 25, "address" : [ { "streetAddress" : "21 2nd Street", "city" : "New York", "state" : "NY", "postalCode" : 1002
1 } ], "phoneNumber" : [ { "type" : "home", "number" : "212555-1234" }, { "type" : "fax", "number" : "646555-4567" } ], "emailAddress" : [ "roman@gmail.com", "tomhank@gmail.com" ] }
{ "_id" : ObjectId("60693f55785128eeebf3e9c2"), "firstName" : "Abhi", "lastName" : "Raj", "age" : 25, "address" : [ { "streetAddress" : "21 Park Street", "city" : "Atlanta", "state" : "New York", "postalCode" :
6785 } ], "phoneNumber" : [ { "type" : "home", "number" : "2112-1234" }, { "type" : "fax", "number" : "64455-4567" } ], "emailAddress" : [ "aman@gmail.com", "thank@gmail.com" ] }
{ "_id" : ObjectId("60693fb4785128eeebf3e9c3"), "firstName" : "Albert", "lastName" : "Einstein", "age" : 40, "address" : [ { "streetAddress" : "Round Road", "city" : "Mexico", "state" : "LA", "postalCode" : 1902
 } ], "phoneNumber" : [ { "type" : "home", "number" : "2134-6767" }, { "type" : "fax", "number" : "6758-723" } ], "emailAddress" : [ "albert@gmail.com", "einstein@gmail.com" ] }
{ "_id" : ObjectId("60694748785128eeebf3e9c5"), "firstName" : "Bulbul", "lastName" : "Kumari", "age" : 28, "address" : [ { "streetAddress" : "Paud Road", "city" : "Pune", "state" : "Maharashtra", "postalCode" :
40032 } ], "phoneNumber" : [ { "type" : "home", "number" : "7267-6788" }, { "type" : "fax", "number" : "6778-7123" } ], "emailAddress" : [ "bulbul@gmail.com", "kumari@gmail.com" ] }
{ "_id" : ObjectId("6069486c785128eeebf3e9c7"), "firstName" : "Scarlett", "lastName" : "Johanson", "age" : 35, "address" : [ { "streetAddress" : "Marrie Street", "city" : "America", "state" : "USA", "postalCode"
 : 13456 } ], "phoneNumber" : [ { "type" : "home", "number" : "7237-6236" }, { "type" : "fax", "number" : "6778-7255" } ], "emailAddress" : [ "scarlet@gmail.com", "johanson@gmail.com" ] }
{ "_id" : ObjectId("606948c0785128eeebf3e9c8"), "firstName" : "Gal", "lastName" : "Gadot", "age" : 32, "address" : [ { "streetAddress" : "Gotham", "city" : "Amazon", "state" : "Russia", "postalCode" : 2356 } ],
"phoneNumber" : [ { "type" : "home", "number" : "3437-1236" }, { "type" : "fax", "number" : "6348-7235" } ], "emailAddress" : [ "gal@gmail.com", "gadot@gmail.com" ] }
{ "_id" : ObjectId("606949bc785128eeebf3e9c9"), "firstName" : "Iron", "lastName" : "Man", "age" : 38, "address" : [ { "streetAddress" : "21 Corner Road", "city" : "America", "state" : "USA", "postalCode" : 1356
} ], "phoneNumber" : [ { "type" : "home", "number" : "337-122" }, { "type" : "fax", "number" : "632-715" } ], "emailAddress" : [ "iron@gmail.com", "man@gmail.com" ] }
>
```



```
@(shell):1:40
> db.Emp.aggregate([{"$match":{"phoneNumber.type":"home"}}])
{ "_id" : ObjectId("60693ed1785128eeebf3e9c1"), "firstName" : "John", "lastName" : "Smith", "age" : 25, "address" : [ { "streetAddress" : "21 2nd Street", "city" : "New York", "state" : "NY", "postalCode" : 1002
1 } ], "phoneNumber" : [ { "type" : "home", "number" : "212555-1234" }, { "type" : "fax", "number" : "646555-4567" } ], "emailAddress" : [ "roman@gmail.com", "tomhank@gmail.com" ] }
{ "_id" : ObjectId("60693f55785128eeebf3e9c2"), "firstName" : "Abhi", "lastName" : "Raj", "age" : 25, "address" : [ { "streetAddress" : "21 Park Street", "city" : "Atlanta", "state" : "New York", "postalCode" :
6785 } ], "phoneNumber" : [ { "type" : "home", "number" : "2112-1234" }, { "type" : "fax", "number" : "64455-4567" } ], "emailAddress" : [ "aman@gmail.com", "thank@gmail.com" ] }
{ "_id" : ObjectId("60693fb4785128eeebf3e9c3"), "firstName" : "Albert", "lastName" : "Einstein", "age" : 40, "address" : [ { "streetAddress" : "Round Road", "city" : "Mexico", "state" : "LA", "postalCode" : 1902
 } ], "phoneNumber" : [ { "type" : "home", "number" : "2134-6767" }, { "type" : "fax", "number" : "6758-723" } ], "emailAddress" : [ "albert@gmail.com", "einstein@gmail.com" ] }
{ "_id" : ObjectId("60694021785128eeebf3e9c4"), "firstName" : "Issac", "lastName" : "Newton", "age" : 57, "address" : [ { "streetAddress" : "China Town", "city" : "Hong Kong", "state" : "China", "postalCode" : 9
78 } ], "phoneNumber" : [ { "type" : "home", "number" : "7288-6788" }, { "type" : "fax", "number" : "6778-723" } ], "emailAddress" : [ "issac@gmail.com", "newton@gmail.com" ] }
{ "_id" : ObjectId("60694748785128eeebf3e9c5"), "firstName" : "Bulbul", "lastName" : "Kumari", "age" : 28, "address" : [ { "streetAddress" : "Paud Road", "city" : "Pune", "state" : "Maharashtra", "postalCode" :
40032 } ], "phoneNumber" : [ { "type" : "home", "number" : "7267-6788" }, { "type" : "fax", "number" : "6778-7123" } ], "emailAddress" : [ "bulbul@gmail.com", "kumari@gmail.com" ] }
{ "_id" : ObjectId("606947b0785128eeebf3e9c6"), "firstName" : "Jackie", "lastName" : "Chan", "age" : 65, "address" : [ { "streetAddress" : "21 Park Street", "city" : "Japan", "state" : "Tokyo", "postalCode" : 10
35 } ], "phoneNumber" : [ { "type" : "home", "number" : "7267-656" }, { "type" : "fax", "number" : "6778-71355" } ], "emailAddress" : [ "jackie@gmail.com", "chan@gmail.com" ] }
{ "_id" : ObjectId("6069486c785128eeebf3e9c7"), "firstName" : "Scarlett", "lastName" : "Johanson", "age" : 35, "address" : [ { "streetAddress" : "Marrie Street", "city" : "America", "state" : "USA", "postalCode"
 : 13456 } ], "phoneNumber" : [ { "type" : "home", "number" : "7237-6236" }, { "type" : "fax", "number" : "6778-7255" } ], "emailAddress" : [ "scarlet@gmail.com", "johanson@gmail.com" ] }
{ "_id" : ObjectId("606948c0785128eeebf3e9c8"), "firstName" : "Gal", "lastName" : "Gadot", "age" : 32, "address" : [ { "streetAddress" : "Gotham", "city" : "Amazon", "state" : "Russia", "postalCode" : 2356 } ],
"phoneNumber" : [ { "type" : "home", "number" : "3437-1236" }, { "type" : "fax", "number" : "6348-7235" } ], "emailAddress" : [ "gal@gmail.com", "gadot@gmail.com" ] }
{ "_id" : ObjectId("606949bc785128eeebf3e9c9"), "firstName" : "Iron", "lastName" : "Man", "age" : 38, "address" : [ { "streetAddress" : "21 Corner Road", "city" : "America", "state" : "USA", "postalCode" : 1356
} ], "phoneNumber" : [ { "type" : "home", "number" : "337-122" }, { "type" : "fax", "number" : "632-715" } ], "emailAddress" : [ "iron@gmail.com", "man@gmail.com" ] }
>
```

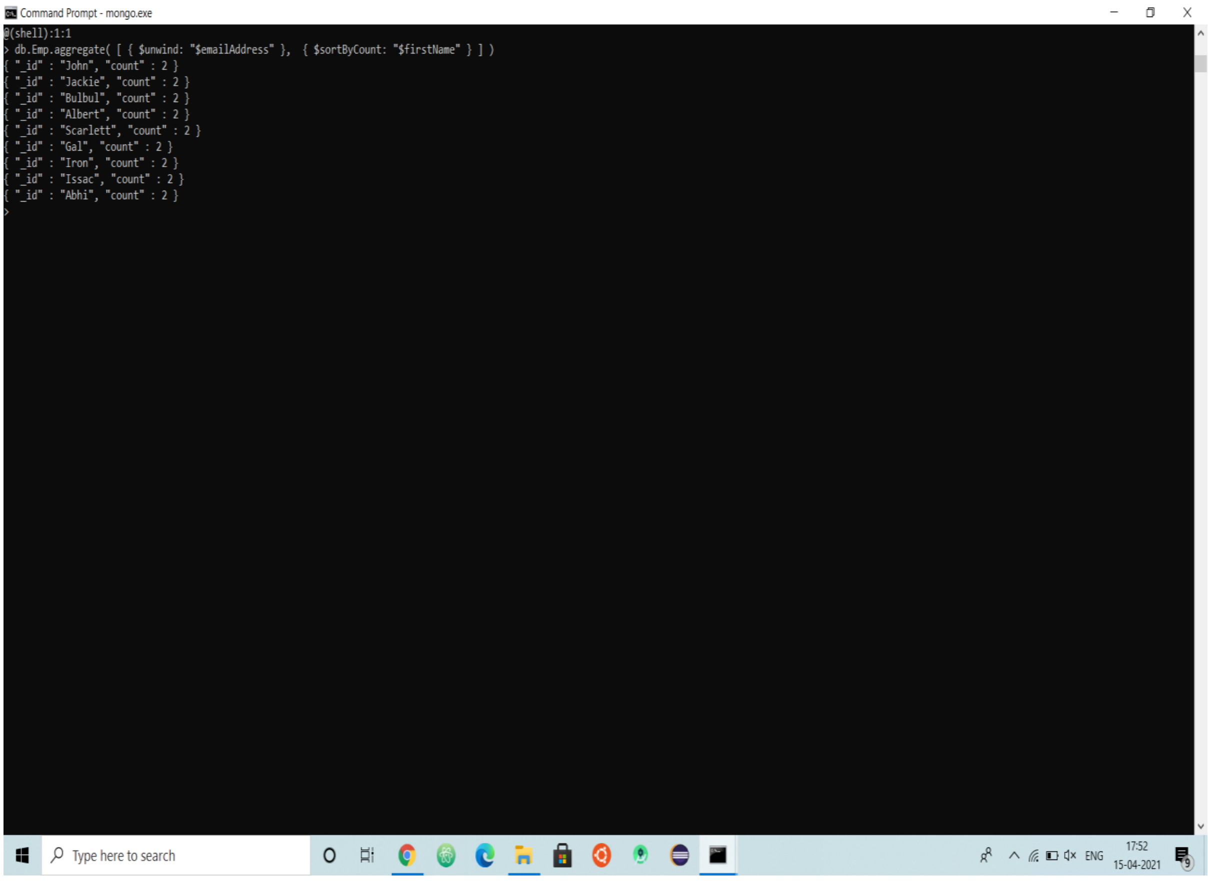


```
> db.Emp.aggregate([{"$match":{"address.state":"America"}}, {$count:"Empcount"}])
> db.Emp.aggregate([{"$match":{"address.state":"USA"}}, {$count:"Empcount"}])
{ "Empcount" : 2 }
>
```



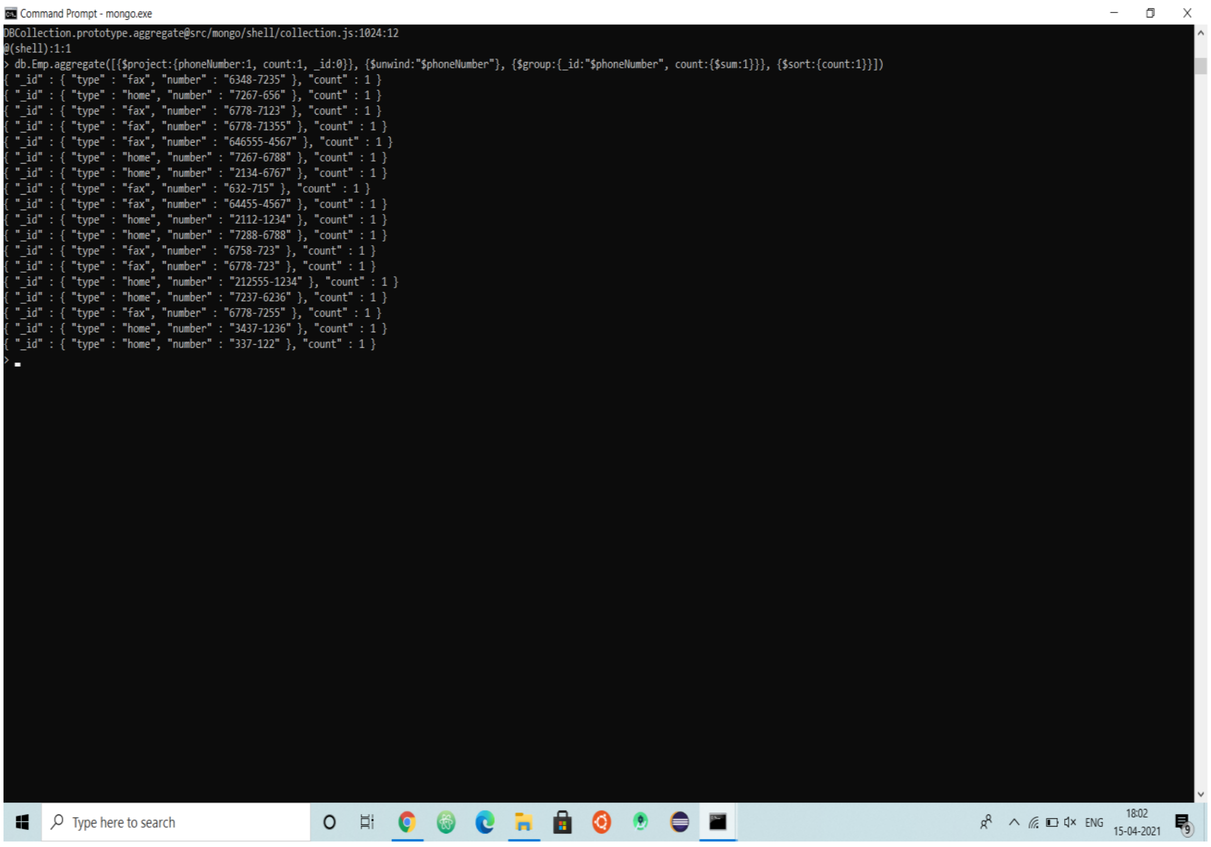
Command Prompt - mongo.exe

```
@(shell):1:58
> db.Emp.aggregate( [ { $unwind: "$emailAddress" },  { $sortByCount: "$emailAddress" } ] )
{ "_id" : "gal@gmail.com", "count" : 1 }
{ "_id" : "roman@gmail.com", "count" : 1 }
{ "_id" : "bulbul@gmail.com", "count" : 1 }
{ "_id" : "chan@gmail.com", "count" : 1 }
{ "_id" : "iron@gmail.com", "count" : 1 }
{ "_id" : "jackie@gmail.com", "count" : 1 }
{ "_id" : "man@gmail.com", "count" : 1 }
{ "_id" : "kumari@gmail.com", "count" : 1 }
{ "_id" : "thank@gmail.com", "count" : 1 }
{ "_id" : "einstein@gmail.com", "count" : 1 }
{ "_id" : "scarlet@gmail.com", "count" : 1 }
{ "_id" : "albert@gmail.com", "count" : 1 }
{ "_id" : "johanson@gmail.com", "count" : 1 }
{ "_id" : "issac@gmail.com", "count" : 1 }
{ "_id" : "aman@gmail.com", "count" : 1 }
{ "_id" : "newton@gmail.com", "count" : 1 }
{ "_id" : "gadot@gmail.com", "count" : 1 }
{ "_id" : "tomhank@gmail.com", "count" : 1 }
>
```



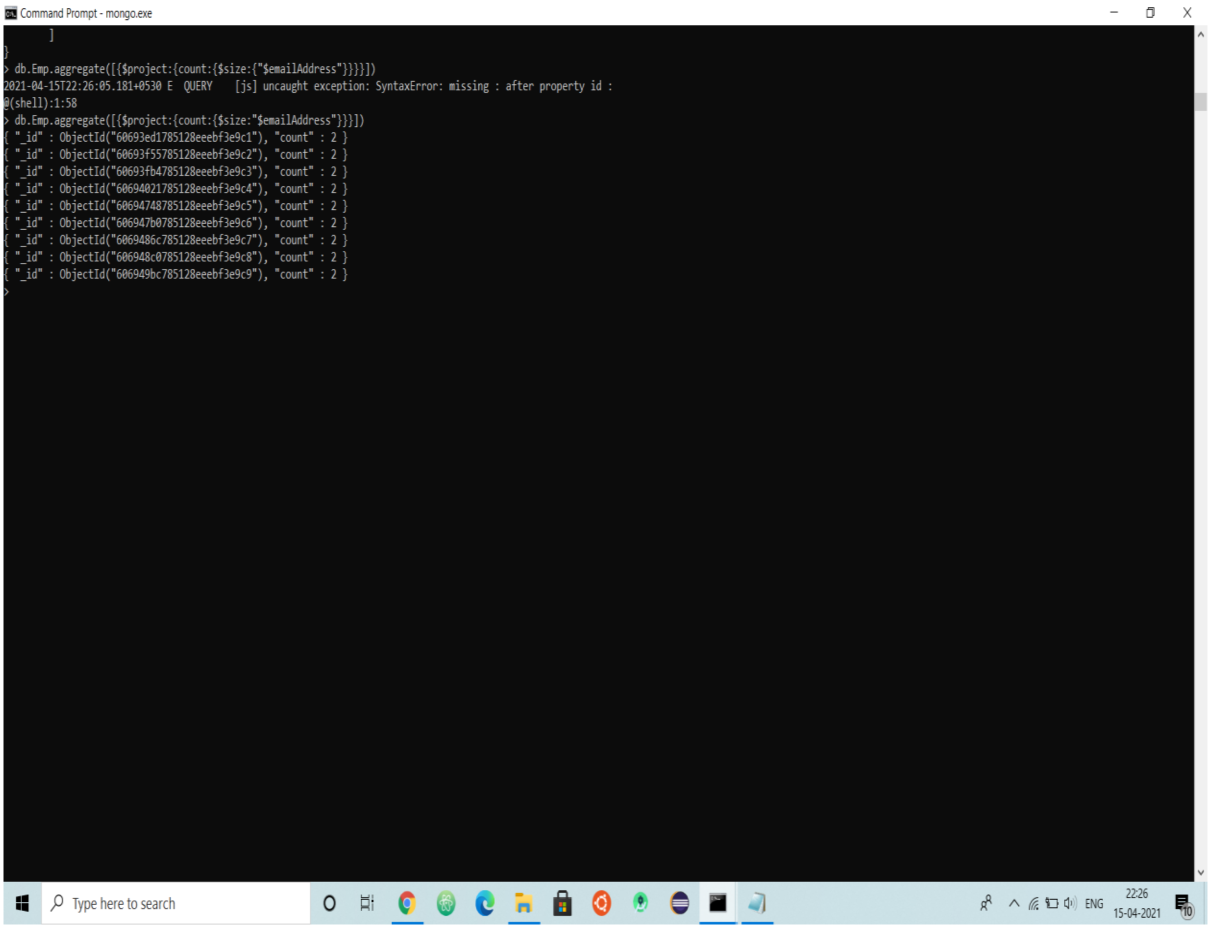
Command Prompt - mongo.exe

```
@(shell):1:1
> db.Emp.aggregate( [ { $unwind: "$emailAddress" },  { $sortByCount: "$firstName" } ] )
{ "_id" : "John", "count" : 2 }
{ "_id" : "Jackie", "count" : 2 }
{ "_id" : "Bulbul", "count" : 2 }
{ "_id" : "Albert", "count" : 2 }
{ "_id" : "Scarlett", "count" : 2 }
{ "_id" : "Gal", "count" : 2 }
{ "_id" : "Iron", "count" : 2 }
{ "_id" : "Issac", "count" : 2 }
{ "_id" : "Abhi", "count" : 2 }
>
```



Command Prompt - mongo.exe

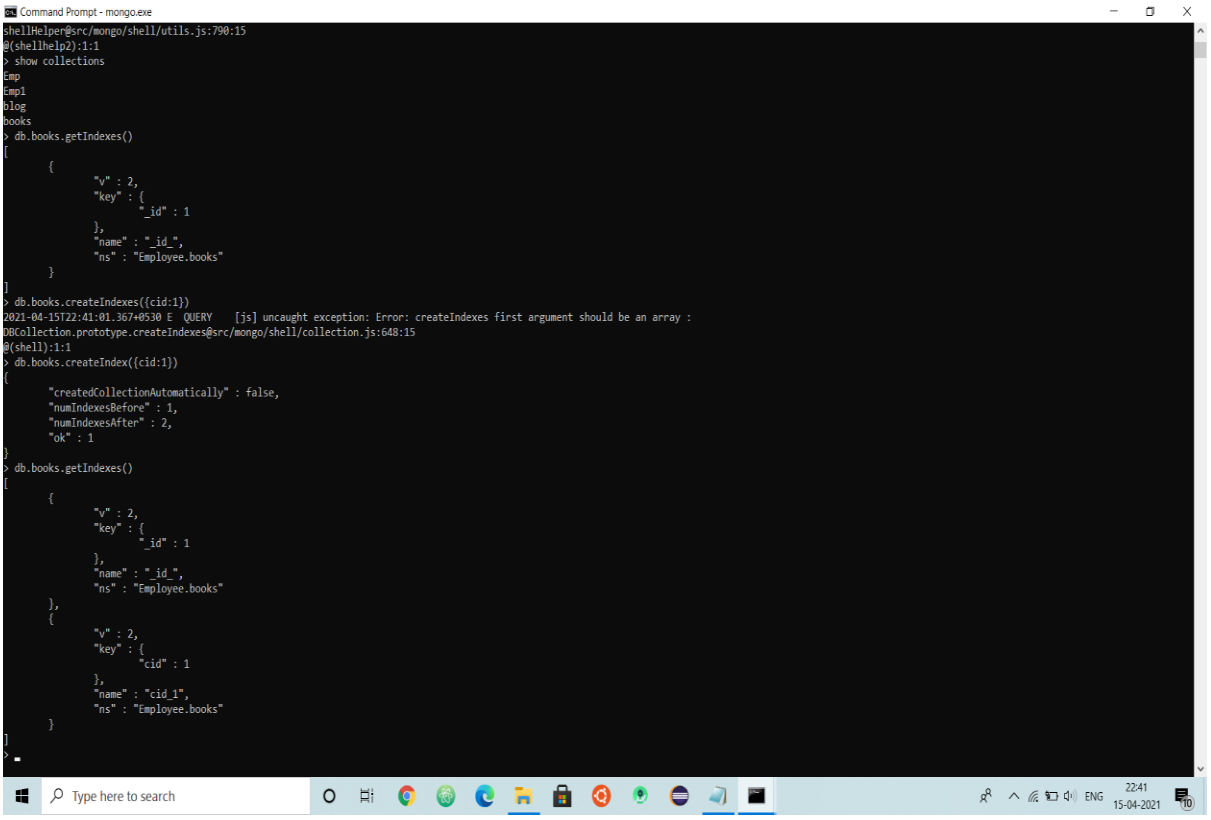
```
DBCollection.prototype.aggregate@src/mongo/shell/collection.js:1024:12
@(shell):1:1
> db.Emp.aggregate([{$project:{phoneNumber:1, count:1, _id:0}}, {$unwind:"$phoneNumber"}, {$group:{_id:"$phoneNumber", count:{$sum:1}}}, {$sort:{count:1}}])
{ "_id" : { "type" : "fax", "number" : "6348-7235" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "7267-656" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "6778-7123" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "6778-71355" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "646555-4567" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "7267-6788" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "2134-6767" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "632-715" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "64455-4567" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "2112-1234" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "7288-6788" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "6758-723" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "6778-723" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "212555-1234" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "7237-6236" }, "count" : 1 }
{ "_id" : { "type" : "fax", "number" : "6778-7255" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "3437-1236" }, "count" : 1 }
{ "_id" : { "type" : "home", "number" : "337-122" }, "count" : 1 }
>
```



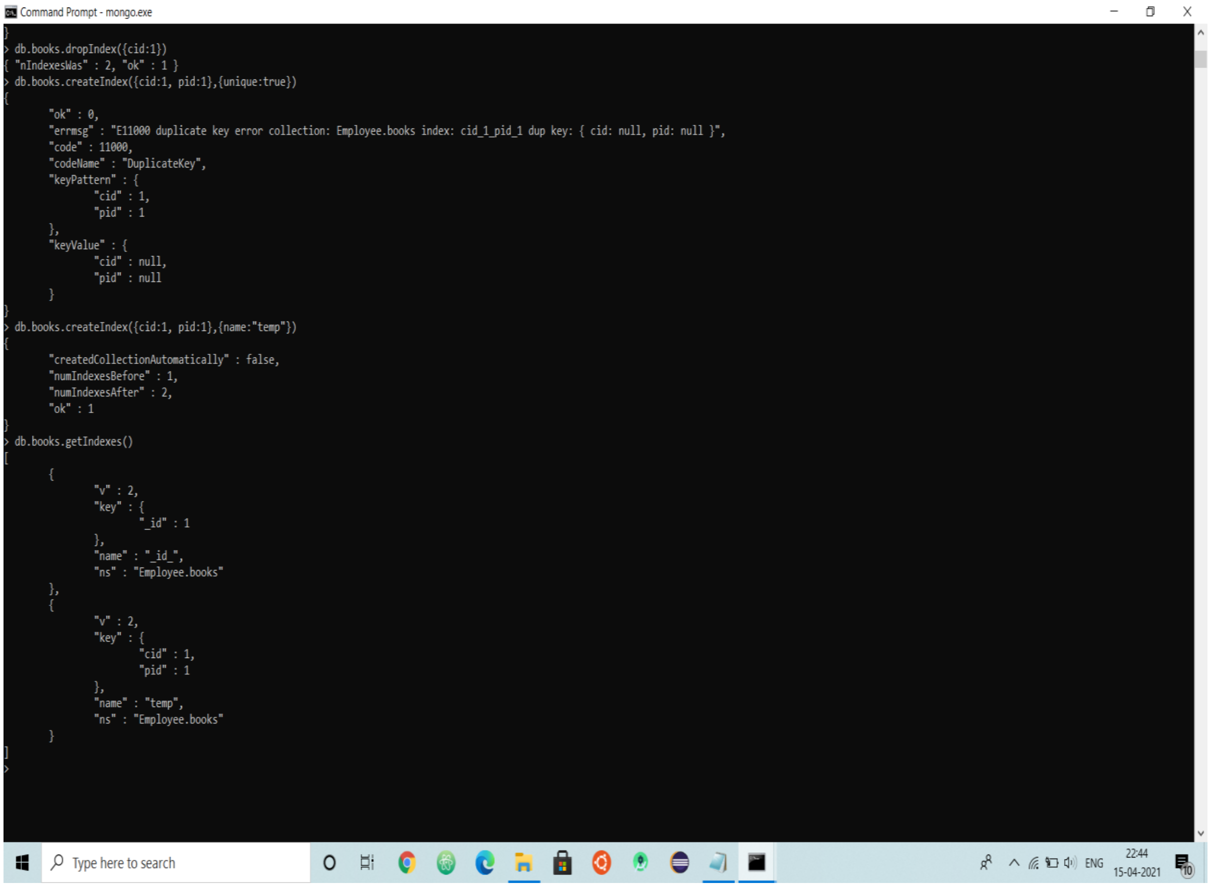
Command Prompt - mongo.exe

```
        ]
}
> db.Emp.aggregate([{$project:{count:{$size:{"$emailAddress"}}}}])
2021-04-15T22:26:05.181+0530 E  QUERY    [js] uncaught exception: SyntaxError: missing : after property id :
@(shell):1:58
> db.Emp.aggregate([{$project:{count:{$size:"$emailAddress"}}}])
{ "_id" : ObjectId("60693ed1785128eeebf3e9c1"), "count" : 2 }
{ "_id" : ObjectId("60693f55785128eeebf3e9c2"), "count" : 2 }
{ "_id" : ObjectId("60693fb4785128eeebf3e9c3"), "count" : 2 }
{ "_id" : ObjectId("60694021785128eeebf3e9c4"), "count" : 2 }
{ "_id" : ObjectId("60694748785128eeebf3e9c5"), "count" : 2 }
{ "_id" : ObjectId("606947b0785128eeebf3e9c6"), "count" : 2 }
{ "_id" : ObjectId("6069486c785128eeebf3e9c7"), "count" : 2 }
{ "_id" : ObjectId("606948c0785128eeebf3e9c8"), "count" : 2 }
{ "_id" : ObjectId("606949bc785128eeebf3e9c9"), "count" : 2 }
>
```

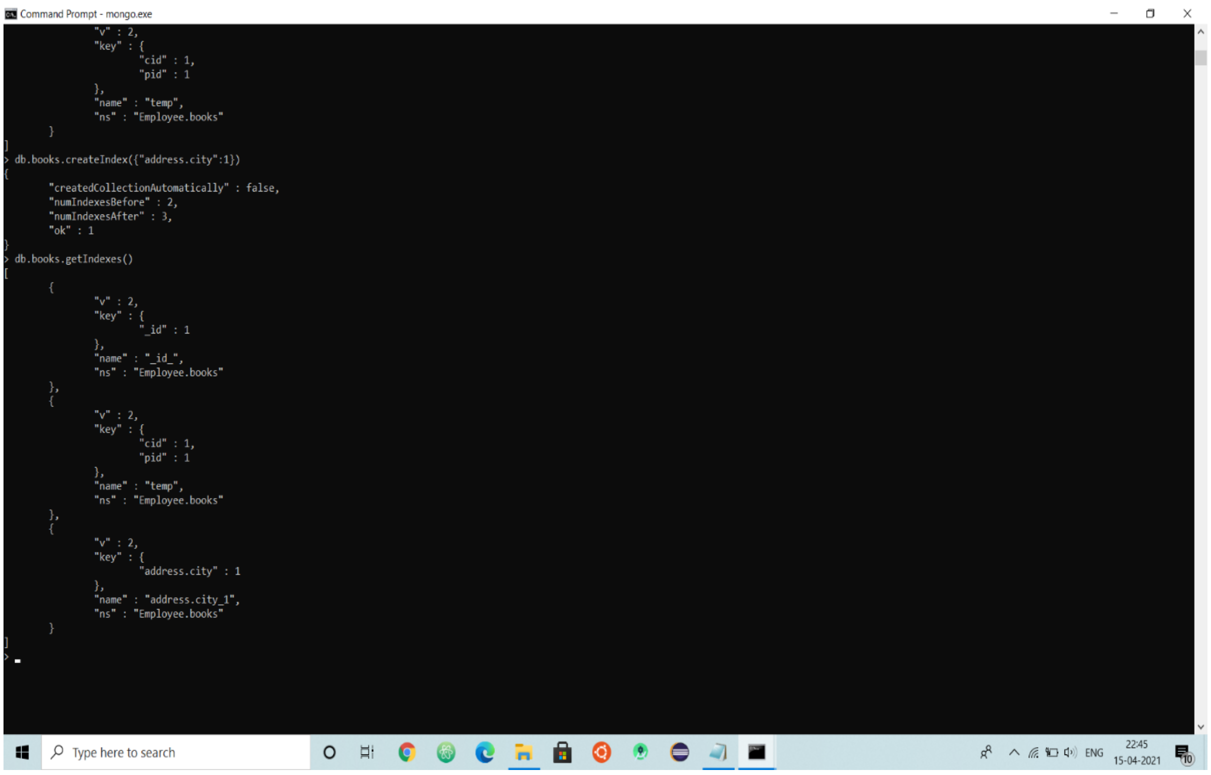
INDEXING:



```
shellHelper@src/mongo/shell/utils.js:790:15
@(shellhelp2):1:1
> show collections
Emp
Emp1
blog
books
> db.books.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "Employee.books"
        }
]
> db.books.createIndexes({cid:1})
2021-04-15T22:41:01.367+0530 E  QUERY    [js] uncaught exception: Error: createIndexes first argument should be an array :
DBCollection.prototype.createIndexes@src/mongo/shell/collection.js:648:15
@(shell):1:1
> db.books.createIndex({cid:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.books.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "Employee.books"
        },
        {
                "v" : 2,
                "key" : {
                        "cid" : 1
                },
                "name" : "cid_1",
                "ns" : "Employee.books"
        }
]
> _
```

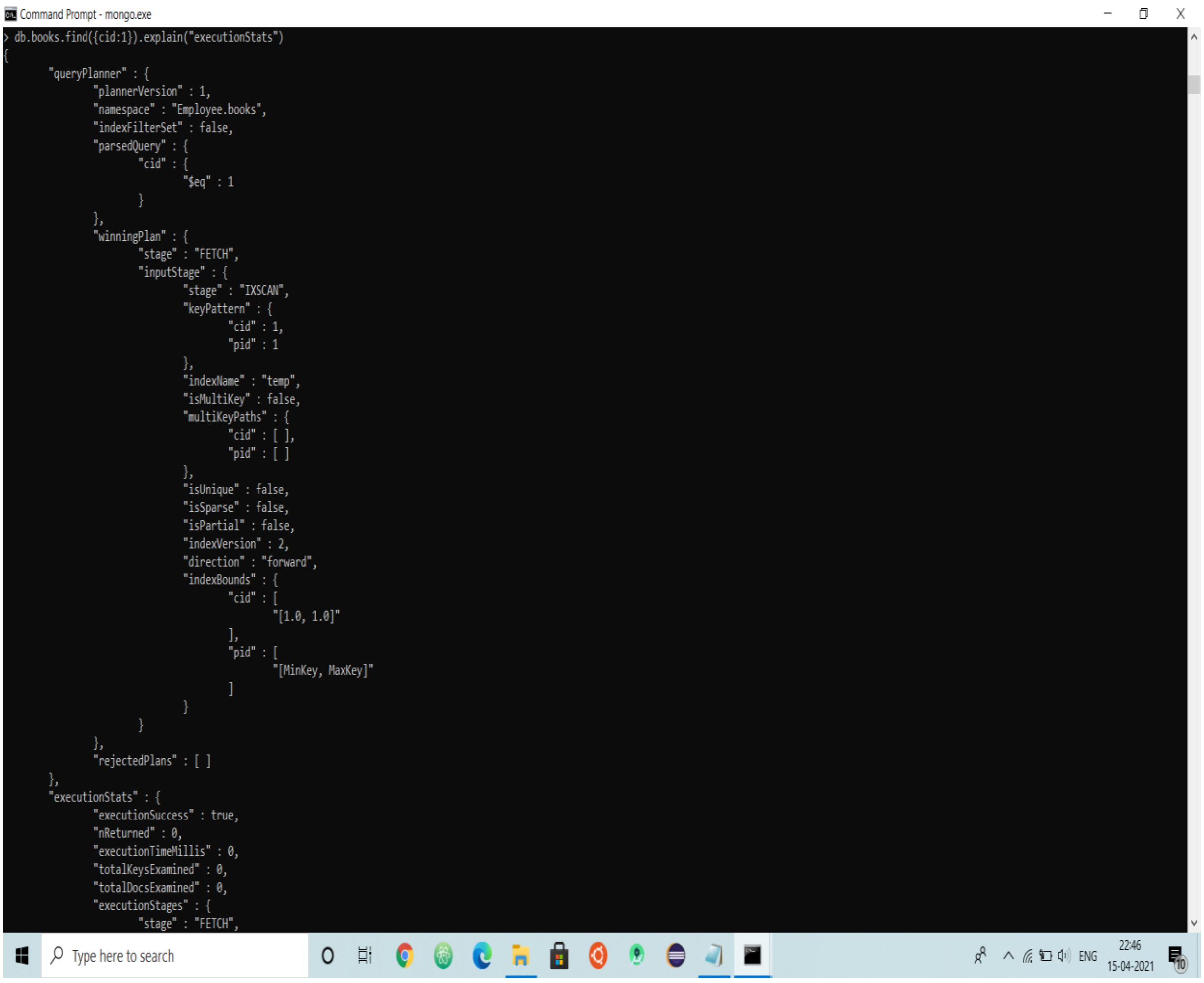


```
}
> db.books.dropIndex({cid:1})
{ "nIndexesWas" : 2, "ok" : 1 }
> db.books.createIndex({cid:1, pid:1},{unique:true})
{
        "ok" : 0,
        "errmsg" : "E11000 duplicate key error collection: Employee.books index: cid_1_pid_1 dup key: { cid: null, pid: null }",
        "code" : 11000,
        "codeName" : "DuplicateKey",
        "keyPattern" : {
                "cid" : 1,
                "pid" : 1
        },
        "keyValue" : {
                "cid" : null,
                "pid" : null
        }
}
> db.books.createIndex({cid:1, pid:1},{name:"temp"})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.books.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "Employee.books"
        },
        {
                "v" : 2,
                "key" : {
                        "cid" : 1,
                        "pid" : 1
                },
                "name" : "temp",
                "ns" : "Employee.books"
        }
]
>
```



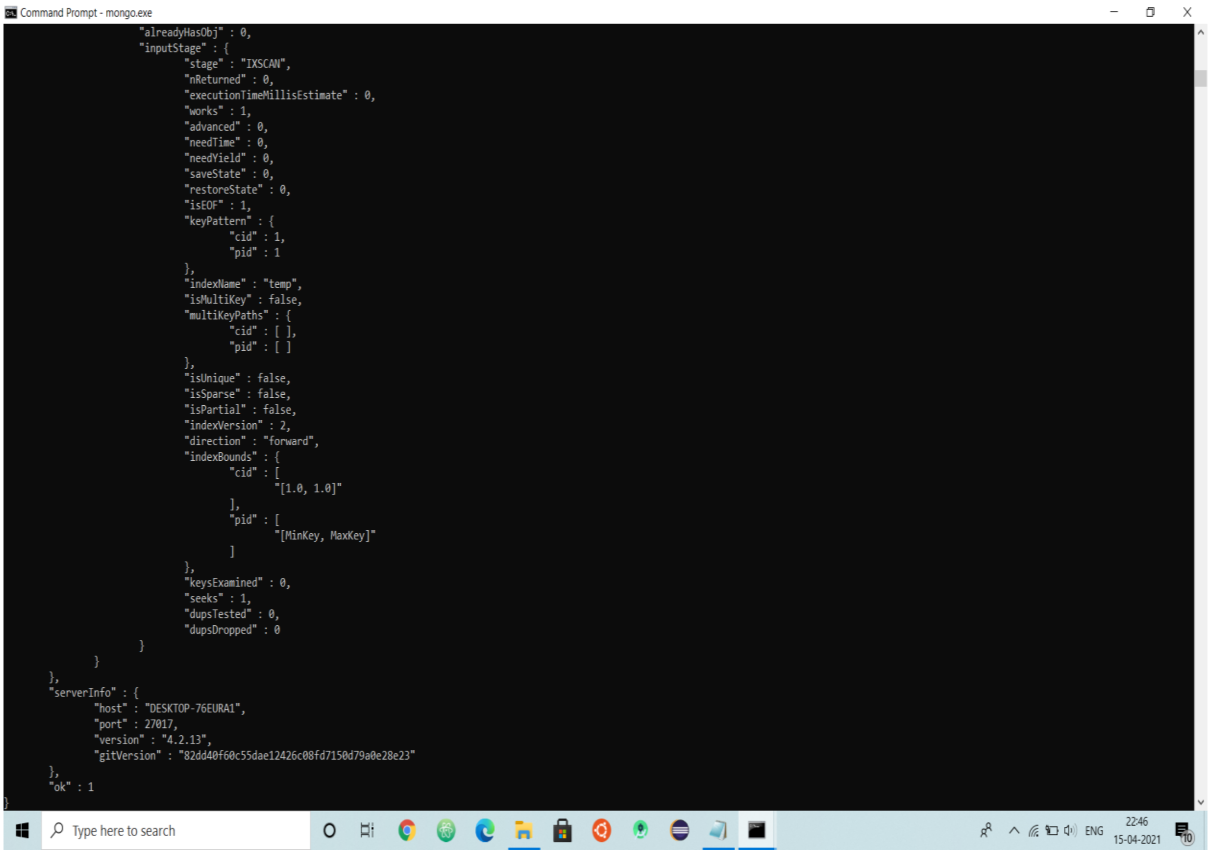
Command Prompt - mongo.exe

```
                "v" : 2,
                "key" : {
                        "cid" : 1,
                        "pid" : 1
                },
                "name" : "temp",
                "ns" : "Employee.books"
        }
]
> db.books.createIndex({"address.city":1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
> db.books.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "Employee.books"
        },
        {
                "v" : 2,
                "key" : {
                        "cid" : 1,
                        "pid" : 1
                },
                "name" : "temp",
                "ns" : "Employee.books"
        },
        {
                "v" : 2,
                "key" : {
                        "address.city" : 1
                },
                "name" : "address.city_1",
                "ns" : "Employee.books"
        }
]
> 
```

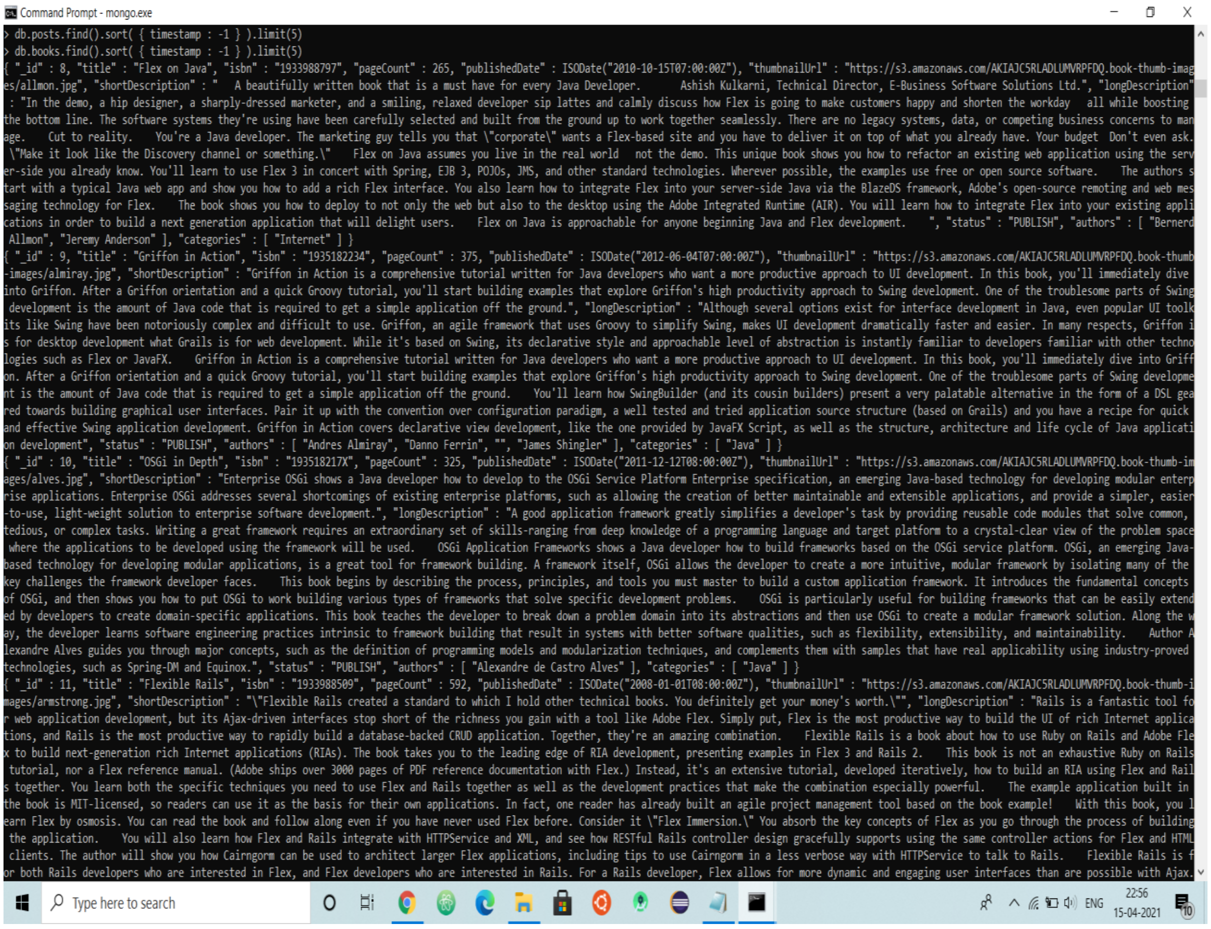


Command Prompt - mongo.exe

```
> db.books.find({cid:1}).explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "Employee.books",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "cid" : {
                                "$eq" : 1
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "cid" : 1,
                                        "pid" : 1
                                },
                                "indexName" : "temp",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "cid" : [ ],
                                        "pid" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "cid" : [
                                                "[1.0, 1.0]"
                                        ],
                                        "pid" : [
                                                "[MinKey, MaxKey]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 0,
                "executionTimeMillis" : 0,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 0,
                "executionStages" : {
                        "stage" : "FETCH",
```



```
                        "alreadyHasObj" : 0,
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "nReturned" : 0,
                                "executionTimeMillisEstimate" : 0,
                                "works" : 1,
                                "advanced" : 0,
                                "needTime" : 0,
                                "needYield" : 0,
                                "saveState" : 0,
                                "restoreState" : 0,
                                "isEOF" : 1,
                                "keyPattern" : {
                                        "cid" : 1,
                                        "pid" : 1
                                },
                                "indexName" : "temp",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "cid" : [ ],
                                        "pid" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "cid" : [
                                                "[1.0, 1.0]"
                                        ],
                                        "pid" : [
                                                "[MinKey, MaxKey]"
                                        ]
                                },
                                "keysExamined" : 0,
                                "seeks" : 1,
                                "dupsTested" : 0,
                                "dupsDropped" : 0
                        }
                }
        },
        "serverInfo" : {
                "host" : "DESKTOP-76EURA1",
                "port" : 27017,
                "version" : "4.2.13",
                "gitVersion" : "82dd40f60c55dae12426c08fd7150d79a0e28e23"
        },
        "ok" : 1
}
```



```
> db.posts.find().sort( { timestamp : -1 } ).limit(5)
> db.books.find().sort( { timestamp : -1 } ).limit(5)
{ "_id" : 8, "title" : "Flex on Java", "isbn" : "1933988797", "pageCount" : 265, "publishedDate" : ISODate("2010-10-15T07:00:00Z"), "thumbnailUrl" : "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-imag
es/allmon.jpg", "shortDescription" : "   A beautifully written book that is a must have for every Java Developer.       Ashish Kulkarni, Technical Director, E-Business Software Solutions Ltd.", "longDescription"
 : "In the demo, a hip designer, a sharply-dressed marketer, and a smiling, relaxed developer sip lattes and calmly discuss how Flex is going to make customers happy and shorten the workday   all while boosting
the bottom line. The software systems they're using have been carefully selected and built from the ground up to work together seamlessly. There are no legacy systems, data, or competing business concerns to man
age.    Cut to reality.    You're a Java developer. The marketing guy tells you that \"corporate\" wants a Flex-based site and you have to deliver it on top of what you already have. Your budget  Don't even ask.
 \"Make it look like the Discovery channel or something.\"    Flex on Java assumes you live in the real world   not the demo. This unique book shows you how to refactor an existing web application using the serv
er-side you already know. You'll learn to use Flex 3 in concert with Spring, EJB 3, POJOs, JMS, and other standard technologies. Wherever possible, the examples use free or open source software.    The authors s
tart with a typical Java web app and show you how to add a rich Flex interface. You also learn how to integrate Flex into your server-side Java via the BlazeDS framework, Adobe's open-source remoting and web mes
saging technology for Flex.    The book shows you how to deploy to not only the web but also to the desktop using the Adobe Integrated Runtime (AIR). You will learn how to integrate Flex into your existing appli
cations in order to build a next generation application that will delight users.    Flex on Java is approachable for anyone beginning Java and Flex development.    ", "status" : "PUBLISH", "authors" : [ "Bernerd
 Allmon", "Jeremy Anderson" ], "categories" : [ "Internet" ] }
{ "_id" : 9, "title" : "Griffon in Action", "isbn" : "1935182234", "pageCount" : 375, "publishedDate" : ISODate("2012-06-04T07:00:00Z"), "thumbnailUrl" : "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb
-images/almiray.jpg", "shortDescription" : "Griffon in Action is a comprehensive tutorial written for Java developers who want a more productive approach to UI development. In this book, you'll immediately dive
into Griffon. After a Griffon orientation and a quick Groovy tutorial, you'll start building examples that explore Griffon's high productivity approach to Swing development. One of the troublesome parts of Swing
 development is the amount of Java code that is required to get a simple application off the ground.", "longDescription" : "Although several options exist for interface development in Java, even popular UI toolk
its like Swing have been notoriously complex and difficult to use. Griffon, an agile framework that uses Groovy to simplify Swing, makes UI development dramatically faster and easier. In many respects, Griffon i
s for desktop development what Grails is for web development. While it's based on Swing, its declarative style and approachable level of abstraction is instantly familiar to developers familiar with other techno
logies such as Flex or JavaFX.    Griffon in Action is a comprehensive tutorial written for Java developers who want a more productive approach to UI development. In this book, you'll immediately dive into Griff
on. After a Griffon orientation and a quick Groovy tutorial, you'll start building examples that explore Griffon's high productivity approach to Swing development. One of the troublesome parts of Swing developme
nt is the amount of Java code that is required to get a simple application off the ground.    You'll learn how SwingBuilder (and its cousin builders) present a very palatable alternative in the form of a DSL gea
red towards building graphical user interfaces. Pair it up with the convention over configuration paradigm, a well tested and tried application source structure (based on Grails) and you have a recipe for quick
and effective Swing application development. Griffon in Action covers declarative view development, like the one provided by JavaFX Script, as well as the structure, architecture and life cycle of Java applicati
on development", "status" : "PUBLISH", "authors" : [ "Andres Almiray", "Danno Ferrin", "", "James Shingler" ], "categories" : [ "Java" ] }
{ "_id" : 10, "title" : "OSGi in Depth", "isbn" : "193518217X", "pageCount" : 325, "publishedDate" : ISODate("2011-12-12T08:00:00Z"), "thumbnailUrl" : "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-im
ages/alves.jpg", "shortDescription" : "Enterprise OSGi shows a Java developer how to develop to the OSGi Service Platform Enterprise specification, an emerging Java-based technology for developing modular enterp
rise applications. Enterprise OSGi addresses several shortcomings of existing enterprise platforms, such as allowing the creation of better maintainable and extensible applications, and provide a simpler, easier
-to-use, light-weight solution to enterprise software development.", "longDescription" : "A good application framework greatly simplifies a developer's task by providing reusable code modules that solve common,
tedious, or complex tasks. Writing a great framework requires an extraordinary set of skills-ranging from deep knowledge of a programming language and target platform to a crystal-clear view of the problem space
 where the applications to be developed using the framework will be used.    OSGi Application Frameworks shows a Java developer how to build frameworks based on the OSGi service platform. OSGi, an emerging Java-
based technology for developing modular applications, is a great tool for framework building. A framework itself, OSGi allows the developer to create a more intuitive, modular framework by isolating many of the
key challenges the framework developer faces.    This book begins by describing the process, principles, and tools you must master to build a custom application framework. It introduces the fundamental concepts
of OSGi, and then shows you how to put OSGi to work building various types of frameworks that solve specific development problems.    OSGi is particularly useful for building frameworks that can be easily extend
ed by developers to create domain-specific applications. This book teaches the developer to break down a problem domain into its abstractions and then use OSGi to create a modular framework solution. Along the w
ay, the developer learns software engineering practices intrinsic to framework building that result in systems with better software qualities, such as flexibility, extensibility, and maintainability.    Author A
lexandre Alves guides you through major concepts, such as the definition of programming models and modularization techniques, and complements them with samples that have real applicability using industry-proved
technologies, such as Spring-DM and Equinox.", "status" : "PUBLISH", "authors" : [ "Alexandre de Castro Alves" ], "categories" : [ "Java" ] }
{ "_id" : 11, "title" : "Flexible Rails", "isbn" : "1933988509", "pageCount" : 592, "publishedDate" : ISODate("2008-01-01T08:00:00Z"), "thumbnailUrl" : "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-i
mages/armstrong.jpg", "shortDescription" : "\"Flexible Rails created a standard to which I hold other technical books. You definitely get your money's worth.\"", "longDescription" : "Rails is a fantastic tool fo
r web application development, but its Ajax-driven interfaces stop short of the richness you gain with a tool like Adobe Flex. Simply put, Flex is the most productive way to build the UI of rich Internet applica
tions, and Rails is the most productive way to rapidly build a database-backed CRUD application. Together, they're an amazing combination.    Flexible Rails is a book about how to use Ruby on Rails and Adobe Fle
x to build next-generation rich Internet applications (RIAs). The book takes you to the leading edge of RIA development, presenting examples in Flex 3 and Rails 2.    This book is not an exhaustive Ruby on Rails
 tutorial, nor a Flex reference manual. (Adobe ships over 3000 pages of PDF reference documentation with Flex.) Instead, it's an extensive tutorial, developed iteratively, how to build an RIA using Flex and Rail
s together. You learn both the specific techniques you need to use Flex and Rails together as well as the development practices that make the combination especially powerful.    The example application built in
the book is MIT-licensed, so readers can use it as the basis for their own applications. In fact, one reader has already built an agile project management tool based on the book example!    With this book, you l
earn Flex by osmosis. You can read the book and follow along even if you have never used Flex before. Consider it \"Flex Immersion.\" You absorb the key concepts of Flex as you go through the process of building
 the application.    You will also learn how Flex and Rails integrate with HTTPService and XML, and see how RESTful Rails controller design gracefully supports using the same controller actions for Flex and HTML
 clients. The author will show you how Cairngorm can be used to architect larger Flex applications, including tips to use Cairngorm in a less verbose way with HTTPService to talk to Rails.    Flexible Rails is f
or both Rails developers who are interested in Flex, and Flex developers who are interested in Rails. For a Rails developer, Flex allows for more dynamic and engaging user interfaces than are possible with Ajax.
```