admin 40.00 KiB
config 72.00 KiB
db 56.00 KiB
Local 40.00 KiB

switched to db

```
[
{
  _id : objectID ('668e9113e9ef ca5a6abb5389');
  name : "student 142'
  age : 24,
  courses : "['History', 'English', 'Physics', 'Computer Science']",
  gpa : 3.41,
  home_city : 'city 5',
  blood_group : 'A+'
  is_hotel_resident : false
},
```

DATE .....
EXP.NO. ...01.....
EXPT TITLE:
PAGE NO. 1

1) a. Illustration of where clause, AND, OR operation in Mongo DB
b. Execute the commands of MongoDB and operations in MongoDB: Insert, Query, update, Delete and Projection
(note: use any collection)

```
test > show dbs
```

```
test> use db
```

a) Illustration of where clause, AND, OR operation in Mongo DB

```
db> db.students.find({$and:[{home_city:"city 5"},
                            {blood_group:"A+"}]});
```

```
{
  _id : objectID ('668e9113e9efca5a6abb54a9'),
  name : 'student 947',
  age : 20,
  Courses : "['Physics', 'History', 'English', 'Computer Science']",
  gpa : 2.86,
  home_city : 'City 5',
  blood_group : 'A+',
  is_hotel_resident : true
},

{
  _id : objectId ['668e9113e9efca5a6abb551b'],
  name : 'student 567',
  Courses : "['Computer Science', 'History', 'English', 'Mathematics']",
  gpa : 2.01,
  home_city : 'City 5',
  blood_group : 'A+',
  is_hotel_resident : true
}
]

→3
```

DATE .....
EXP. NO. .....
EXPT TITLE:
PAGE NO. 2

```
db> db.students.find({ $and : [{ home_city : "City 5" },
    { blood_group : "A+" }]}).count();
```

```
[{ _id : objectId ('668e9113e9efca5a6abb5352'),
  name : 'student 948',
  age : 19,
  courses : "['English', 'Computer Science', 'Physics', 'Mathematics']",
  gpa : 3.44,
  home_city : "city2",
  blood_group : 'B+',
  is_hotel_resident : true
},
...
{ _id : objectId ('668e9113e9efca5a6abb5368'),
  name : 'student 499',
  age : 25
  courses : "['Mathematics', 'English', 'Computer Science', 'Physics']",
  gpa : 2.04,
  home_city : 'city 5',
  blood_group : 'A+'
  is_hotel_resident : false
}
]

→ 374
```

DATE .....
EXP. NO. .....
EXPT TITLE:
PAGE NO. 3

```
db> db.students.find({$or:[{is_hotel_resident : true},
    {gpa: {$lt: 3.0}}]});
```

```
db> db.students.find({$or:[{is_hotel_resident : true},{gpa :
    {$lt: 3.0}}]}).count();
```

```
{
  acknowledged : true,
  insertId : objectId('668ea4527ffe6e6ff2e8e3e8')
}

{ acknowledge : true,
  insertIds : {
    '0' : objectId('668ea4e87ffe6efff2e8e3e9'),
    '1' : objectId('668ea4e87ffe6eff2e8e3ea'),
    '2' : objectId('668ea4e87ffebeff2e8e3eb'),
}

{ acknowledged : true,
  insertedId : null,
  matchedCount : 1,
  modifiedCount : 1,
  upsetedCount : 0
}

{
  acknowledged : true
  insertedId : null,
  matched count : 248,
  modified count : 248,
  upsetted count : 0
}

{ acknowledged : true, deleted count : 1 }

{ acknowledged : true, deleted count : 128 }
```

DATE ..... EXPT TITLE: PAGE NO. 4
EXP. NO. .....

b) Execute the Commands of MongoDB & operation in MongoDB Insert Query, Update, delete & projection.

```
db> db.students.insertOne({ name : "John Doe", age : 20,
    is_hotel_resident : true, gpa : 3.5 });

db> db.students.insertMany([{ name : "Alice", age : 22, is_hotel_
    resident : false, gpa : 2.9 },
{ name : "Bob", age : 21, is_hotel_resident : true, gpa : 3.2 },
{ name : "Charlie", age : 23, is_hotel_resident : false, gpa : 3.8 }]);

db> db.students.updateOne({ name : "John Doe" }, { $set : {
    gpa : 3.7 }});

db> db.students.updateMany({ is_hotel_resident : true },
    { $set : { gpa : 3.0 }});

db> db.students.deleteOne({ name : "Alice" });

db> db.students.deleteMany({ gpa : { $lt : 3.0 }});
```

```
[
  { name: 'student 268', gpa: 3.98 },
  { name: 'student 871', gpa: 3.2 },
  { name: 'student 368', gpa: 3.91 },
  { name: 'student 652', gpa: 3.93 },
  { name: 'student 239', gpa: 3.75 },
  ...
  { name: 'student 508', gpa: 3.48 }
]
```

126

DATE ..... EXPT TITLE:
EXP. NO. 1 PAGE NO. 5

```
db> db.students.find({gpa:{$gt:3.0}},{_id:0, name:1,
                                      gpa:1});
```

```
db> db.students.find({gpa:{$gt:3.0}},
                     {_id:0, name:1, gpa:1}).count();
```

output

```
[
 {
  home : 'student 948'
  gpa : 4
  home_city : 'city 2',
  blood_group : 'O+',
  is_hotel_resident : true
 }
 {
  name : 'student 151'
  gpa : 4,
  home_city : 'city4',
  blood_group : 'O-',
  is_hotel_resident : true
},
.....
{
  name : 'student 504',
  gpa : 4,
  home_city : 'city 2',
  blood_group : 'B+'
  is_hotel_resident : true
}
```


DATE .....
EXP. NO. 02
EXPT TITLE:
PAGE NO. 6

a) Develop a MongoDB query to select certain fields and ignore some fields of the documents

```
db> db.students.find({}, {_id:0, name:1, blood_group:1,
 home_city:1, is_hotel_resident:true, gpa:1});
```

```
db> db.students.find({}, {_id:0, name:1, blood_group:1
 home_city:1, is_hotel_resident:true, gpa:1}).count();
```

output

```
[
 {
  name: 'Student 948',
  gpa : 4
  home_city : 'city2',
  blood_group : 'O+'
 },
 {
  name : 'Student 157'
  gpa : 4
  home_city : 'city 4',
  blood_group : 'O-'
 },
 {
  name : 'Student 316',
  gpa : 4
  home_city : 'city 5',
  blood_group : 'B+'
 },
 {
  name : 'Student 346'
  gpa : 4
  home_city : 'city 8'
  blood_group : 'O-'
 },
 {
  name : 'Student 930'
  gpa : 4
  home_city : 'city 3'
  blood_group : 'A-'
 }
]
```

DATE .....
EXP. NO. 2
EXPT TITLE:
PAGE NO. 7

b) Develop a MongoDB query to display the first 5 document

```
db> db.students.find({},{_id:0, is_hotel_resident:0,
blood_group:1, home_city:1, gpa:1}).limit(5);
```

output :

```
[
 {
  _id : objectId ('668fff265d60ae15ff6644a6'),
  name : 'student 346',
  age : 25,
  courses : "[mathematics, 'History', 'English"],
  gpa : 4
  home_city : 'city8',
  blood_group : 'A+'
  is_hotel_resident : true
 }
 ,
 {
  _id : objectId ('668fff265d60ae15ff6644c7'),
  name : 'student 502',
  age : 25
  courses : " ['computer Science', 'mathematics', 'English', 'Physics']
  gpa : 4
  home_city : 'city8',
  blood_group : 'AB-',
  is_hotel_resident : true
 }
```

234

DATE .....
EXP. NO. 3
EXPT TITLE:
PAGE NO. 8

a) Execute query selectors (comparison selectors, logical selectors):

→ Comparision selectors

```
db> db.student.find({age:{$gt:20}});
```

```
db> db.student.find({age:{$gt:20}}).count();
```

output:

```
[{
  _id : objectId(
  name : 'student 142',
  age : 24
  Courses : "['History', 'English', 'Physics', 'Computer Science']"
  gpa : 3.4
  home_city : 'city 5',
  blood_group : 'A+',
  is_hotel_resident : false
}
{
  _id : objectId(
  name : 'student 947',
  age : 20
  courses : "['physics', 'History', 'English', 'Computer Science']"
  gpa : 4
  home_city : 'city 5',
  blood_group : 'A+'
  is_hotel_resident : true
},
{
  _id : objectId(
  name : 'student 567',
  age : 22
  Courses : "['Computer Science', 'History', 'English', 'mathematics']",
  gpa : 4
  home_city : 'city 5',
  blood_group : 'A+'
  is_hotel_resident : true
}]
```

DATE .....
EXP. NO. .....
EXPT TITLE:
PAGE NO. 9

Logical selectors

```
dbs db.students.find({$and:[{home_city:"city 5"},
    {blood_group:"A+"}]});
```

output

```
[
  {
    _id : 1,
    name : 'Coffee Shop A',
    location : { type : 'Point', Coordinates : [-73.985, 40.748] }
  },
  {
    _id : 2,
    name : 'Restaurant B',
    location : { type : 'point', coordinates : [-74.009, 40.712] }
  },
  {
    _id : 5,
    name : 'Park E',
    location : { type : 'Point', coordinates : [-74.006, 40.705] }
  }
]
```

DATE .....
EXP. NO. 3
EXPT TITLE:
PAGE NO. 10

b) Execute query selectors (Geospatial selectors) and list out the results on any collections

```
db > db.locations.find ( { location : { $geoWithin : { $centerSphere
  : [ [-74.005, 40.712], 0.06621376] } } });
```

output

```
[{
  name : 'Bob Johnson',
  courses : ['Computer Science', 'Mathematics', 'Physics']
},
{
  name : 'Gabriel Miller',
  Courses : ['Computer Science', 'Engineering', 'Robotics']
},
{
  name : 'Kevin Lewis'
  Courses : ['Computer Science', 'Artificial Intelligence', 'Cyber Security']
}
]
```

output

```
[{ _id : objectId ('6696bc3dec1628a4962d9bc69')
  name : 'Alice Smith',
  Courses : ['English', 'Biology']
},---
{ _id : objectId ('6696bc3dec1628a4962d9bc74')
  name : 'Lily Robinsons',
  Courses : ['History', 'Art History']
}
]
```


DATE .....
EXP. NO. 4
EXPT TITLE:
PAGE NO. 11

Create and demonstrate how projection operation loperators ($, $elematch and $slice) would be used in Mongodb.

```
db> db.candidates.find({courses:{$elemMatch:{$eq:"Computer Science"}}},{name:1, courses:1, _id:0});
```

"courses.$":1

```
db> db.candidates.find({}, courses:{$slice:2}});
```

```
db> db.candidates.find({}.courses:{$slice:2}}).count();
```

output

```
[{_id: null, averageGPA: 3.3741035856573705}]
```

```
[{_id: null, minAge: 18}]
```

```
[{_id: null, maxAge: 45}]
```

```
[
  {
    _id: null,
    UniqueCourses: [
      'English',
      'Physics',
      'Mathematics',
      'Political Science',
      ...,
      'Literature'
    ]
  }
]
```

DATE .....
EXP. NO. 5
EXPT TITLE:
PAGE NO. 12

Execute Aggregation operations ($avg, $min, $max, $push, $addToSet etc)
Students encourage to execute several queries to demonstrate various aggregation operators)

Avg :-

```
db> db.students.aggregate([{$group: {_id: null, averageGPA: {$avg: "$gpa"}}}]);
```

Minimum ($min)

```
db> db.students.aggregate([{$group: {_id: null, minAge: {$min: "$age"}}}]);
```

maximum ($max)

```
db> db.students.aggregate([{$group: {_id: null, maxAge: {$min: "$age"}}}]);
```

addToset ($addToSet)

```
db> db.candidates.aggregate([{$unwind: "$courses"}, {$group: {_id: null, UniqueCourses: {$addToSet: "$courses"}}}]);
```

output:

```
[{name: 'David', averageScore : 93.33333333}]
```

output:

```
db > db. st
[ { name: 'David', scores: [98, 95, 87]},
  { name : 'Bob', scores: [90, 88, 95]}
]
```

DATE ..... EXPT TITLE:

EXP. NO. 6 PAGE NO. 13

Execute aggregation pipeline and its operations (~~$avg~~, ~~$min~~, ~~$max~~, ~~$push~~, ~~$addToSet~~ etc) (pipeline must contains $match, $group, $sort, $project, $skip etc. students encourage to execute several queries to demonstrate various aggregation operators

Find students of an avg score above 85 & skip the 1st document

```
db> db. students 6 . aggregate ([{$project : {_id:0, name:1,
                averageScore : {$avg : "$scores"}}},
{$match : {averageScore : {$gt : 85}}}, {$skip : 1}])
```

Find students with age less than 23 sorted by name in ascending order, & only return name & score

```
db > db. students 6. aggregate ([{$match : {age : {$lt : 23}}},
    {$sort : {$sort : {age : 1}}}, {$project : {_id:0, name: 1,
    ~~age : 1~~ scores: 1}}])
```

output :-

```
{
  acknowledged : true, insertedIds: {
  '0' : objectId ('6694e7f55c53d304afcdcdf6'),
  '1' : objectId ('6694e7f55c53d304afcdcdf7'),
  '2' : objectId ('6694e7f55c53d304afcdcdf8'),
  }
}
```

output :

```
[
  { _id : objectId ('6694e7f55c53d304afcdcdf6'),
    listing_url : "http://example.com/listing1",
    name : "Beautiful Apartment",
    address : "123 main st, Anytown, USA",
    host_picture_url : "http://example.com/host1.jpg"
  },
  { _id : objectId ('6694e7f55c53d304afcdcdf7'),
    listing_url : "http://example.com/listing2",
    name : "Cozy Cottage",
    address : "456 Elm St, Anytown, USA",
    host_picture_url : "http://example.com/host2.jpg"
  }
]
```

DATE ..... EXPT TITLE:
EXP. NO. 7 PAGE NO. 14

a. Find all listings with listing_url, name, address, host_picture_url in the listingsandReviews collection that have a host with a picture url

```
db> db.listingsandreviews.insertMany([{
      listing_url: "http://example.com/listings1",
      name: "Beautiful Apartment",
      address: "123 main St, Anytown, USA",
      host_picture_url: "http://example.com/host1.jpg"},
    { listing_url: "http://example.com/listing2",
      name : "Cozy Cottage",
      address : "456 Elm St, Anytown, USA",
      host_picture_url: "http://example.com/host2.jpg"},
    { listing_url : "http://example.com/listing 3",
      name : "Modern Condo",
      address : "789 Oak St, Anytown, USA"}]);

db> db.listingsandreviews.find({host_picture_url:{$exists:true,
    $ne:null}},{listing_url:1, name:1, address:1,
    host_picture_url:1})
```

output :-

```
[ {
  acknowledged : true,
  insertId : {
  '0' : objectId ('6694cfc75c53d304afcdcdfd'),
  '1' : objectId ('6694cfc75c53d304afcdcdfe'),
  '2' : objectId ('6694cfc75c53d304afcdcdff'),
  '3' : objectId ('6694cfc75c53d304afcdce00'),
} }
]
```

output :

```
[
  {
    totalReviews : 4,
    averageRating : 3.25,
    highestRating : 5,
    lowestRating : 1
  }
]
```

DATE ..... EXPT TITLE:
EXP. NO. 7 PAGE NO. 15

b. using E-Commerce collection write a query to display reviews summary

```
db> db.Ecommerce.insertMany([{ product_id : 1,
      review : "Great Product", rating : 5 },
{ product_id : 2, review : "Not bad", rating : 3 },
{ product_id : 3, review : "would not recommend", rating : 1},
{ product_id : 4, review : "Excellent value for money",
      rating : 4 }]);
```

using aggregate function

```
db> db.Ecommerce.aggregate([{ $group : { _id : null,
      totalReviews : { $sum : 1 },
      averageRating : { $avg : "$rating" },
      highestRating : { $max : "$rating" },
      lowestRating : { $min : "$rating" },
    }
  },
  { $project : { _id : 0,
      totalReviews : 1,
      averageRating : 1,
      highestRating : 1,
      lowestRating : 1,
    }
  }
]);
```

output:

```
[
 {
  acknowledged : true,
  insertedIds : {
  '0': objectId('66a1ec9c9c0523dd8790defe')
  '1': objectId('66a1ec9c9c0523dd8790deff')
  '2': objectId('66a1ec9c9c0523dd8790df00')
  '3': objectId('66a1ec9c9c0523dd8790df01')
  '4': objectId('66a1ec9c9c0523dd8790df02'),
 }}
]
```

output:

name_1

age_1

name_1_age_-1

hobbies_1

DATE ..... EXP. NO. 8

EXPT TITLE:

PAGE NO. 16

a Demonstrate creation of different types of indexes on collection (unique, sparse, compound and multikey indexes)

```
db> db.samples.insertMany([{
  name: "Alice", age: 25, hobbies: ["reading", "swimming"],
   address: { city: "NewYork", zip: 10001}},
 { name: "Bob", age: 30, hobbies: ["hiking", "cycling"],
   address: { city: "San Francisco", zip: 94101}},
 { name: "Charlie", age: 35, hobbies: ["gaming", "coding"],
   address: { city: "Los Angeles", zip: 90001}},
 { name: "David", age: 40, hobbies: ["Reading", "hiking"],
   address: { city: "Seattle", zip: 98101}},
 { name: "Eve", age: 28, hobbies: ["swimming", "gaming"],
   address: { city: "Chicago", zip: 60601}}])
```

a. Unique

```
db> db.samplecollection.createIndex({name:1},{unique:true})
```

b. Sparse

```
db> db.samplecollection.createIndex({age:1},{sparse:true})
```

c Compound

```
db> db.samplecollection.createIndex({name:1, age:-1})
```

d multikey

```
db> db.samplecollection.createIndex({hobbies:1})
```

DATE .....
EXP. NO. 8

EXPT TITLE:

b. Demonstrate optimization of queries using indexes

db> db.samplecollection.find({name:"Alice"}).explain("executionStats")

db> db.samplecollection.find({age:{$gt:20}}).explain("executionStats")

DATE
EXP. NO.
EXPT TITLE:
PAGE NO. 18

```
db > db.sampleCollection.find
    ({name: "Alice", age: 25}).explain("executionStats")
```

```
db > db.sampleCollection.find
    ({hobbies: "reading"}).explain("executionStats")
```

output :

```
{
  acknowledged : true,
  insertedIds : {
    '0' : objectId ('669fcfc44e3e8003a3e8e3ed'),
    '1' : objectId ('669fcfc44e3e8003a3e83ee0'),
    '2' : objectId ('669fcfc44e3e8003a3e83ef'),
    '3' : objectId ('669fcfc44e3c8003a3e83f0'),
    '4' : objectId ('669fcfc44e3e8003a3e83f1'),
    '5' : objectId ('669fcfc44e3e8003a3e83f2'),
    '6' : objectId ('669fcfc 44e3e8003a3e83f3'),
  }
}
```

DATE ..... EXPT TITLE: PAGE NO. 19
EXP. NO. ...9.....

a Develop a query to demonstrate text search using catalog data collection for a given word

```
db> db.catalog.insertMany([
  {
    title : "Apple Iphone",
    description : "latest model of Apple Iphone with great
                   features"
  },
  {
    title : "Samsung Galaxy",
    description : "New Samsung Galaxy with an amazing display"
  },
  { title : "Google pixel",
    description : "Google pixel phone with the best camera"
  },
  { title : "Sony Headphones",
    description : "Noise - cancelling headphones from Sony"
  },
  {
    title : "HP Laptop",
    description : "High performance laptop from HP"
  },
  { title : "Dell Monitor",
    description : "Dell 4K monitor with stunning visuals"
  },
  { title : "Canon Camera",
    description : "Professional Canon camera for photographers"
  }
])
```

output:

```
[
  {
    _id: objectId ('669fcfc44e3e8003a3e8e3ed'),
    title: 'Apple iphone',
    description: 'latest model of Apple iphone with great features'
  }
]
```

```
[
  {
    _id: objectId ('669fcfcA4e3e8003a3e8e3ee'),
    title: 'Samsung Galaxy',
    description: 'New Samsung Galaxy with an amazing display'
  }
]
```

DATE
EXP. NO.
EXPT TITLE:
PAGE NO. 30

creating Index

```
db> db.catalog.createIndex({title:"text", description:"text"});
title_text_description_text

db> db.catalog.find({$text:{$search:"Apple"}});
```

b) Develop queries to illustrate excluding documents with certain word & phrases

```
db> db.catalog.find({$text:{$search:"phone"}, text:{$search:
                                    "Samsung"}});
```

output:

```
[
  {
    _id : objectId ('669fcfc44e3e8003a3e8e3ed'),
    title : 'Apple iphone',
    description : 'latest model of Apple iphone with great features',
    score : 1.333333333335
  }
]
```


Develop an aggregation pipeline to illustrate text search on catalog data collection.

```
db = db.catalog.aggregate([
  {
    $match : { $text : { $search : "apple" } }
  },
  {
    $project : {
      title : 1,
      description : 1,
      score : { $meta : "textScore" }
    }
  },
  {
    $sort : { score : { $meta : "textscore" } }
  }
])
```