Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

CET4001B Big Data Technologies

School of Computer Engineering and Technology

Big Data Technologies

Teaching Scheme **Theory:** 3 Hrs / Week **Practical:** 2Hrs/Week

- **Course Objectives:**

1) Understand the various aspects of Big Data.
2) Learn the concepts of NoSQL for Big Data.
3) Design an application for distributed systems on Big Data.
4) Explore the various Big Data visualization tools.

- **Course Outcomes:**

1) Apply the insights of Big Data in business applications.
2) Illustrate the application of MongoDB in real world applications.
3) Build hadoop based distributed systems for real world problem.
4) Apply and utilize big data visualization tools for real world applications.

Aggregation and Indexing in MongoDB

LABORATORY ASSIGNMENT NO: 02

Aggregation and Indexing in MongoDB

Index Operations in MongoDB

Creation index

- db.users.ensureIndex({ score: 1 })

Show existing indexes

- db.users.getIndexes()

Drop index

- db.users.dropIndex({score: 1})

Explain—Explain

- db.users.find().explain()
- Returns a document that describes the process and indexes

Hint

- db.users.find().hint({score: 1})
- Overide MongoDB's default index selection

Index Creation

Using CreateIndex

db.CollectionName.createIndex({ KeyName: 1 or -1})

Using ensureIndex

db.CollectionName.ensureIndex({KeyName: 1 or -1})

1 for Ascending Sorting

-1 for Descending Sorting

Index Creation

Using CreateIndex

Single: db.stud.createIndex({ zipcode: 1})

Compound: db.stud.createIndex({ dob: 1, zipcode: -1 })

Unique: db.stud.createIndex({ rollno: 1 }, { unique: true })

Sparse: db.stud.createIndex({ age: 1 }, { sparse: true })

Using ensureIndex

Single: db.stud.ensureIndex({“name":1})

Compound: db.stud.ensureIndex ({“address":1,“name":-1})

Index Display

db.collection.getIndexes()

- Returns an array that holds a list of documents that identify and describe the existing indexes on the collection.

db.collection.getIndexStats()

Displays a human-readable summary of aggregated statistics about an index's B-tree data structure.

db.<collection>.getIndexStats({ index : "<index name>" })

Index Drop

Syntax

- db.collection.dropIndex()
- db.collection.dropIndex(*index*)

Example

- db.stud.dropIndex()
- db.stud.dropIndex({ “name" : 1 })

Demo of indexes in MongoDB

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

```
> db.zips.find().limit(20)
{ "city" : "ACMAR", "loc" : [  -86.51557,  33.584132 ], "pop" : 6055, "state" : "AL", "_id" : "35004" }
{ "city" : "ADAMSVILLE", "loc" : [  -86.959727,  33.588437 ], "pop" : 10616, "state" : "AL", "_id" : "35005" }
{ "city" : "ADGER", "loc" : [  -87.167455,  33.434277 ], "pop" : 3205, "state" : "AL", "_id" : "35006" }
{ "city" : "KEYSTONE", "loc" : [  -86.812861,  33.236868 ], "pop" : 14218, "state" : "AL", "_id" : "35007" }
{ "city" : "NEW SITE", "loc" : [  -85.951086,  32.941445 ], "pop" : 19942, "state" : "AL", "_id" : "35010" }
{ "city" : "ALPINE", "loc" : [  -86.208934,  33.331165 ], "pop" : 3062, "state" : "AL", "_id" : "35014" }
{ "city" : "ARAB", "loc" : [  -86.489638,  34.328339 ], "pop" : 13650, "state" : "AL", "_id" : "35016" }
{ "city" : "BAILEYTON", "loc" : [  -86.621299,  34.268298 ], "pop" : 1781, "state" : "AL", "_id" : "35019" }
{ "city" : "BESSEMER", "loc" : [  -86.947547,  33.409002 ], "pop" : 40549, "state" : "AL", "_id" : "35020" }
{ "city" : "HUEYTOWN", "loc" : [  -86.999607,  33.414625 ], "pop" : 39677, "state" : "AL", "_id" : "35023" }
{ "city" : "BLOUNTSVILLE", "loc" : [  -86.568628,  34.092937 ], "pop" : 9058, "state" : "AL", "_id" : "35031" }
{ "city" : "BREMEN", "loc" : [  -87.004281,  33.973664 ], "pop" : 3448, "state" : "AL", "_id" : "35033" }
{ "city" : "BRENT", "loc" : [  -87.211387,  32.93567 ], "pop" : 3791, "state" : "AL", "_id" : "35034" }
{ "city" : "BRIERFIELD", "loc" : [  -86.951672,  33.042747 ], "pop" : 1282, "state" : "AL", "_id" : "35035" }
{ "city" : "CALERA", "loc" : [  -86.755987,  33.1098 ], "pop" : 4675, "state" : "AL", "_id" : "35040" }
{ "city" : "CENTREVILLE", "loc" : [  -87.11924,  32.950324 ], "pop" : 4902, "state" : "AL", "_id" : "35042" }
{ "city" : "CHELSEA", "loc" : [  -86.614132,  33.371582 ], "pop" : 4781, "state" : "AL", "_id" : "35043" }
{ "city" : "COOSA PINES", "loc" : [  -86.337622,  33.266928 ], "pop" : 7985, "state" : "AL", "_id" : "35044" }
{ "city" : "CLANTON", "loc" : [  -86.642472,  32.835532 ], "pop" : 13990, "state" : "AL", "_id" : "35045" }
{ "city" : "CLEVELAND", "loc" : [  -86.559355,  33.992106 ], "pop" : 2369, "state" : "AL", "_id" : "35049" }
> db.zips.find().count()
29467
```

Demo of indexes in MongoDB

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

```
db.zips.ensureIndex({pop: -1})
db.zips.ensureIndex({state: 1, city: 1})
db.zips.ensureIndex({loc: -1})
```

Demo of indexes in MongoDB

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

```
> db.zips.getIndexes()
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "ns" : "blog.zips",
                "name" : "_id_"
        },
        {
                "v" : 1,
                "key" : {
                        "pop" : 1
                },
                "ns" : "blog.zips",
                "name" : "pop_1"
        },
        {
                "v" : 1,
                "key" : {
                        "state" : 1,
                        "city" : 1
                },
                "ns" : "blog.zips",
                "name" : "state_1_city_1"
        },
        {
                "v" : 1,
                "key" : {
                        "loc" : 1
                },
                "ns" : "blog.zips",
                "name" : "loc_1"
        }
]
```

Demo of indexes in MongoDB

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

```
> db.zips.find().limit(20).hint({pop: -1})
{ "city" : "CHICAGO", "loc" : [  -87.7157,  41.849015 ], "pop" : 112047, "state" : "IL", "_id" : "60623" }
{ "city" : "BROOKLYN", "loc" : [  -73.956985,  40.646694 ], "pop" : 111396, "state" : "NY", "_id" : "11226" }
{ "city" : "NEW YORK", "loc" : [  -73.958805,  40.768476 ], "pop" : 106564, "state" : "NY", "_id" : "10021" }
{ "city" : "NEW YORK", "loc" : [  -73.968312,  40.797466 ], "pop" : 100027, "state" : "NY", "_id" : "10025" }
{ "city" : "BELL GARDENS", "loc" : [  -118.17205,  33.969177 ], "pop" : 99568, "state" : "CA", "_id" : "90201" }
{ "city" : "CHICAGO", "loc" : [  -87.556012,  41.725743 ], "pop" : 98612, "state" : "IL", "_id" : "60617" }
{ "city" : "LOS ANGELES", "loc" : [  -118.258189,  34.007856 ], "pop" : 96074, "state" : "CA", "_id" : "90011" }
{ "city" : "CHICAGO", "loc" : [  -87.704322,  41.920903 ], "pop" : 95971, "state" : "IL", "_id" : "60647" }
{ "city" : "CHICAGO", "loc" : [  -87.624277,  41.693443 ], "pop" : 94317, "state" : "IL", "_id" : "60628" }
{ "city" : "NORWALK", "loc" : [  -118.081767,  33.90564 ], "pop" : 94188, "state" : "CA", "_id" : "90650" }
{ "city" : "CHICAGO", "loc" : [  -87.654251,  41.741119 ], "pop" : 92005, "state" : "IL", "_id" : "60620" }
{ "city" : "CHICAGO", "loc" : [  -87.706936,  41.778149 ], "pop" : 91814, "state" : "IL", "_id" : "60629" }
{ "city" : "CHICAGO", "loc" : [  -87.653279,  41.809721 ], "pop" : 89762, "state" : "IL", "_id" : "60609" }
{ "city" : "CHICAGO", "loc" : [  -87.704214,  41.946401 ], "pop" : 88377, "state" : "IL", "_id" : "60618" }
{ "city" : "JACKSON HEIGHTS", "loc" : [  -73.878551,  40.740388 ], "pop" : 88241, "state" : "NY", "_id" : "11373" }
{ "city" : "ARLETA", "loc" : [  -118.420692,  34.258081 ], "pop" : 88114, "state" : "CA", "_id" : "91331" }
{ "city" : "BROOKLYN", "loc" : [  -73.914483,  40.662474 ], "pop" : 87079, "state" : "NY", "_id" : "11212" }
{ "city" : "SOUTH GATE", "loc" : [  -118.201349,  33.94617 ], "pop" : 87026, "state" : "CA", "_id" : "90280" }
{ "city" : "RIDGEWOOD", "loc" : [  -73.896122,  40.703613 ], "pop" : 85732, "state" : "NY", "_id" : "11385" }
{ "city" : "BRONX", "loc" : [  -73.871242,  40.873671 ], "pop" : 85710, "state" : "NY", "_id" : "10467" }
```

Demo of indexes in MongoDB

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

```
> db.zips.find().limit(20).hint({state: 1, city: 1})
{ "city" : "98791", "loc" : [  -176.310048,  51.938901 ], "pop" : 5345, "state" : "AK", "_id" : "98791" }
{ "city" : "AKHIOK", "loc" : [  -152.500169,  57.781967 ], "pop" : 13309, "state" : "AK", "_id" : "99615" }
{ "city" : "AKIACHAK", "loc" : [  -161.39233,  60.891854 ], "pop" : 481, "state" : "AK", "_id" : "99551" }
{ "city" : "AKIAK", "loc" : [  -161.199325,  60.890632 ], "pop" : 285, "state" : "AK", "_id" : "99552" }
{ "city" : "AKUTAN", "loc" : [  -165.785368,  54.143012 ], "pop" : 589, "state" : "AK", "_id" : "99553" }
{ "city" : "ALAKANUK", "loc" : [  -164.60228,  62.746967 ], "pop" : 1186, "state" : "AK", "_id" : "99554" }
{ "city" : "ALEKNAGIK", "loc" : [  -158.619882,  59.269688 ], "pop" : 185, "state" : "AK", "_id" : "99555" }
{ "city" : "ALLAKAKET", "loc" : [  -152.712155,  66.543197 ], "pop" : 170, "state" : "AK", "_id" : "99720" }
{ "city" : "AMBLER", "loc" : [  -156.455652,  67.46951 ], "pop" : 8, "state" : "AK", "_id" : "99786" }
{ "city" : "ANAKTUVUK PASS", "loc" : [  -151.679005,  68.11878 ], "pop" : 260, "state" : "AK", "_id" : "99721" }
{ "city" : "ANCHORAGE", "loc" : [  -149.876077,  61.211571 ], "pop" : 14436, "state" : "AK", "_id" : "99501" }
{ "city" : "ANCHORAGE", "loc" : [  -150.093943,  61.096163 ], "pop" : 15891, "state" : "AK", "_id" : "99502" }
{ "city" : "ANCHORAGE", "loc" : [  -149.893844,  61.189953 ], "pop" : 12534, "state" : "AK", "_id" : "99503" }
{ "city" : "ANCHORAGE", "loc" : [  -149.74467,  61.203696 ], "pop" : 32383, "state" : "AK", "_id" : "99504" }
{ "city" : "ANCHORAGE", "loc" : [  -149.828912,  61.153543 ], "pop" : 20128, "state" : "AK", "_id" : "99507" }
{ "city" : "ANCHORAGE", "loc" : [  -149.810085,  61.205959 ], "pop" : 29857, "state" : "AK", "_id" : "99508" }
{ "city" : "ANCHORAGE", "loc" : [  -149.897401,  61.119381 ], "pop" : 17094, "state" : "AK", "_id" : "99515" }
{ "city" : "ANCHORAGE", "loc" : [  -149.779998,  61.10541 ], "pop" : 18356, "state" : "AK", "_id" : "99516" }
{ "city" : "ANCHORAGE", "loc" : [  -149.936111,  61.190136 ], "pop" : 15192, "state" : "AK", "_id" : "99517" }
{ "city" : "ANCHORAGE", "loc" : [  -149.886571,  61.154862 ], "pop" : 8116, "state" : "AK", "_id" : "99518" }
```

Demo of indexes in MongoDB

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

```
> db.zips.find().limit(20).hint({loc: -1})
{ "city" : "BARROW", "loc" : [  -156.817409,  71.234637 ], "pop" : 3696, "state" : "AK", "_id" : "99723" }
{ "city" : "WAINWRIGHT", "loc" : [  -160.012532,  70.620064 ], "pop" : 492, "state" : "AK", "_id" : "99782" }
{ "city" : "NUIQSUT", "loc" : [  -150.997119,  70.192737 ], "pop" : 354, "state" : "AK", "_id" : "99789" }
{ "city" : "PRUDHOE BAY", "loc" : [  -148.559636,  70.070057 ], "pop" : 153, "state" : "AK", "_id" : "99734" }
{ "city" : "KAKTOVIK", "loc" : [  -143.631329,  70.042889 ], "pop" : 245, "state" : "AK", "_id" : "99747" }
{ "city" : "POINT LAY", "loc" : [  -162.906148,  69.705626 ], "pop" : 139, "state" : "AK", "_id" : "99759" }
{ "city" : "POINT HOPE", "loc" : [  -166.72618,  68.312058 ], "pop" : 640, "state" : "AK", "_id" : "99766" }
{ "city" : "ANAKTUVUK PASS", "loc" : [  -151.679005,  68.11878 ], "pop" : 260, "state" : "AK", "_id" : "99721" }
{ "city" : "ARCTIC VILLAGE", "loc" : [  -145.423115,  68.077395 ], "pop" : 107, "state" : "AK", "_id" : "99722" }
{ "city" : "KIVALINA", "loc" : [  -163.733617,  67.665859 ], "pop" : 689, "state" : "AK", "_id" : "99750" }
{ "city" : "AMBLER", "loc" : [  -156.455652,  67.46951 ], "pop" : 8, "state" : "AK", "_id" : "99786" }
{ "city" : "KIANA", "loc" : [  -158.152204,  67.18026 ], "pop" : 349, "state" : "AK", "_id" : "99749" }
{ "city" : "BETTLES FIELD", "loc" : [  -151.062414,  67.100495 ], "pop" : 156, "state" : "AK", "_id" : "99726" }
{ "city" : "VENETIE", "loc" : [  -146.413723,  67.010446 ], "pop" : 184, "state" : "AK", "_id" : "99781" }
{ "city" : "NOATAK", "loc" : [  -160.509453,  66.97553 ], "pop" : 395, "state" : "AK", "_id" : "99761" }
{ "city" : "SHUNGNAK", "loc" : [  -157.613496,  66.958141 ], "pop" : 0, "state" : "AK", "_id" : "99773" }
{ "city" : "KOBUK", "loc" : [  -157.066864,  66.912253 ], "pop" : 306, "state" : "AK", "_id" : "99751" }
{ "city" : "KOTZEBUE", "loc" : [  -162.126493,  66.846459 ], "pop" : 3347, "state" : "AK", "_id" : "99752" }
{ "city" : "NOORVIK", "loc" : [  -161.044132,  66.836353 ], "pop" : 534, "state" : "AK", "_id" : "99763" }
{ "city" : "CHALKYITSIK", "loc" : [  -143.638121,  66.719 ], "pop" : 99, "state" : "AK", "_id" : "99788" }
```

Demo of indexes in MongoDB

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

```
> db.zips.find({city: 'NASHVILLE', state: 'TN'}).explain()
{
        "cursor" : "BasicCursor",
        "isMultiKey" : false,
        "n" : 19,
        "nscannedObjects" : 29467,
        "nscanned" : 29467,
        "nscannedObjectsAllPlans" : 29467,
        "nscannedAllPlans" : 29467,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 33,
        "indexBounds" : {

        },
        "server" : "g:27017"
}
```

Demo of indexes in MongoDB

- Import Data
- Create Index
  - Single Field Index
  - Compound Field Indexes
  - Multikey Indexes
- Show Existing Index
- Hint
  - Single Field Index
  - Compound Field Indexes
  - Multikey Indexes
- Explain
- Compare with data without indexes

```
> db.zips.dropIndexes()
{
        "nIndexesWas" : 4,
        "msg" : "non-_id indexes dropped for collection",
        "ok" : 1
}
> db.zips.find({city: 'NASHVILLE', state: 'TN'}).explain()
{
        "cursor" : "BasicCursor",
        "isMultiKey" : false,
        "n" : 19,
        "nscannedObjects" : 29467,
        "nscanned" : 29467,
        "nscannedObjectsAllPlans" : 29467,
        "nscannedAllPlans" : 29467,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 33,
        "indexBounds" : {

        },
        "server" : "g:27017"
}
```

Without Index

With Index

```
> db.zips.find({city: 'NASHVILLE', state: 'TN'}).explain()
{
        "cursor" : "BtreeCursor state_1_city_1",
        "isMultiKey" : false,
        "n" : 19,
        "nscannedObjects" : 19,
        "nscanned" : 19,
        "nscannedObjectsAllPlans" : 19,
        "nscannedAllPlans" : 19,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 0,
        "indexBounds" : {
                "state" : [
                        [
                                "TN",
                                "TN"
                        ]
                ],
                "city" : [
                        [
                                "NASHVILLE",
                                "NASHVILLE"
                        ]
                ]
        },
        "server" : "g:27017"
}
```

Aggregation Framework Operators

- $group
- $project
- $match
- $limit
- $skip
- $sort
- $unwind

Possible stages in aggregation

$project – Used to select some specific fields from a collection.

$match – This is a filtering operation and thus this can reduce the amount of documents that are given as input to the next stage.

$group – This does the actual aggregation as discussed above.

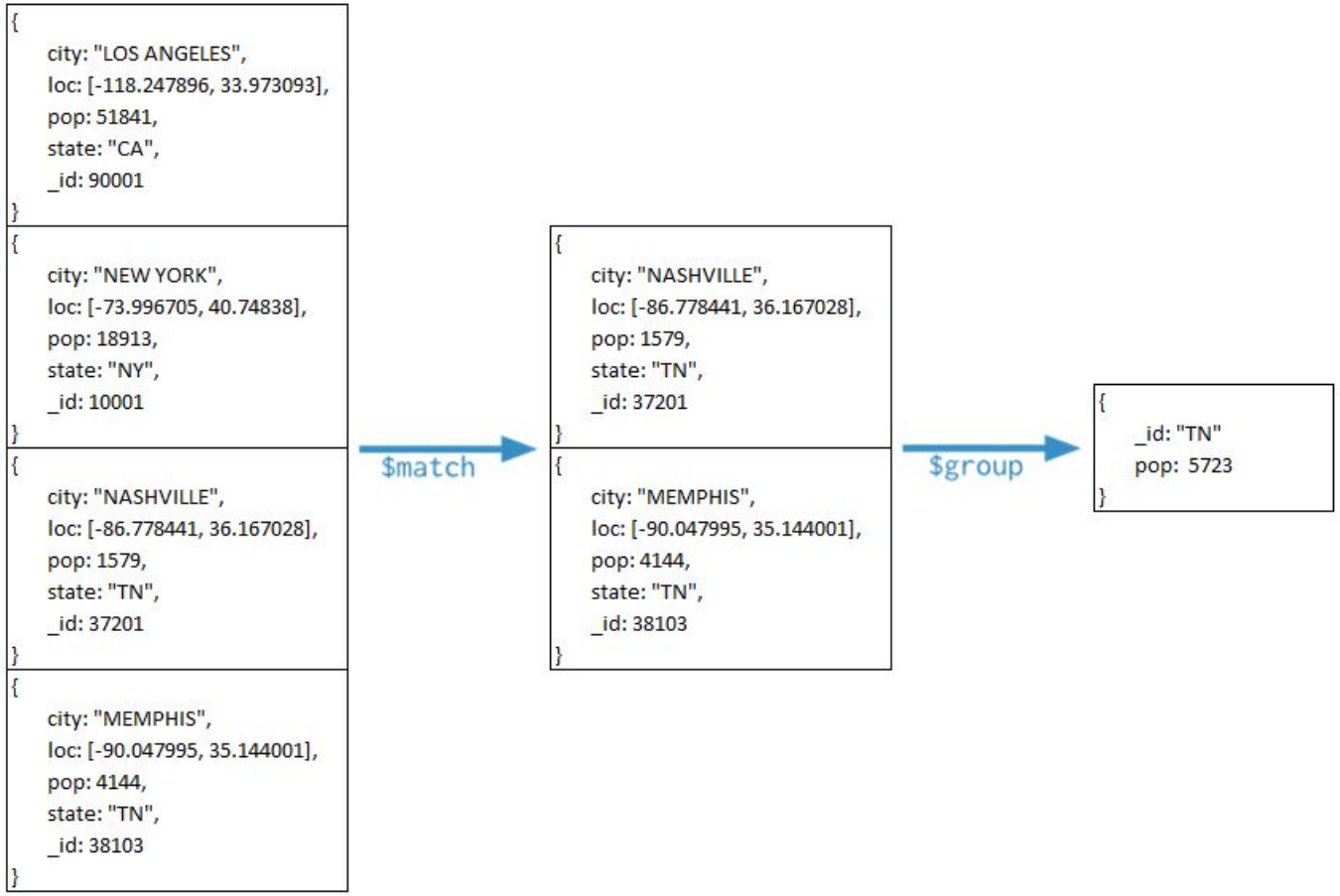
$sort – Sorts the documents.

$skip – With this, it is possible to skip forward in the list of documents for a given amount of documents.

$limit – This limits the amount of documents to look at, by the given number starting from the current positions.

$unwind – This is used to unwind document that are using arrays. When using an array, the data is kind of pre-joined and this operation will be undone with this to have individual documents again. Thus with this stage we will increase the amount of documents for the next stage.

```
db.zips.aggregate(
                    { $match: { state: "TN" } },
                    { $group: {_id: "TN", pop: { $sum: "$pop" } } }
                  );
```



Collection creation to run practical

- db.student.insert({Rollno:1,name:' Navin ',subject:'DMSA',marks:78});
- db.student.insert({Rollno:2,name:' anusha',subject:'OSD',marks:75});
- db.student.insert({Rollno:3,name:'r avi',subject:'TOC',marks:69});
- db.student.insert({Rollno:4,name:' veena',subject:'TOC',marks:70});
- db.student.insert({Rollno:5,name: ‘Pravini',subject: ‘OSD',marks:80});
- db.student.insert({Rollno:6,name: ‘Reena',subject: ‘DMSA',marks:50});
- db.student.insert({Rollno:7,name: ‘Geeta',subject: ‘CN',marks:90});
- db.student.insert({Rollno:8,name: ‘Akash',subject: ‘CN',marks:85});

MIN()

```
db.student.aggregate
([{$group : {_id :
"$subject",marks : {$min
: "$marks"}}}]);
```

SQL Equivalent Query

Select subject, min(marks) from student group by subject

MAX()

```
db.student.aggregate  ([{$group : {_id :
"$subject", marks : {$max :
"$marks"}}}]);
```

SQL Equivalent Query

Select subject, max(marks) from student group by subject

AVG()

```
db.student.aggregate  ([{$group :
{_id : "$subject",marks : {$avg :
"$marks"}}}]);
```

SQL Equivalent Query

Select subject, avg(marks) from student group by subject

FIRST()

- db.student.aggregate([{$group : {_id : "$subject",marks : {$first : "$marks"}}}]);

LAST()

- db.student.aggregate ([{$group : {_id : "$subject",marks : {$last : "$marks"}}}]);

SUM()-Example 1

```
db.student.aggregate  ([{$group : {_id : "$subject",marks
: {$sum :"$marks"}}}]);
```

SQL Equivalent Query

Select subject, sum(marks) from student group by subject

SUM(): Example 2

```
db.student.aggregate  ([{$group : {_id : "$subject",Count: {$sum : 1}}}]);
```

SQL Equivalent Query

Select subject, count(*) from student group by subject

$match

- db.student.aggregate([{ $match: {subject:"OSD"}}])
- db.student.aggregate([{$match:{subject:"OSD"}},{ $group:{_id:null,count:{$sum:1}}}]);

SUM()- Example 3

```
db.student.aggregate([{ $match:
{subject:"OSD"}},{$group:{_id:null,count:{$sum:1}}}]);
```

SQL Equivalent Query

Select subject, count(*) from student group by subject having subject=“OSD”

Limit() & Skip()

```
db.student.aggregate([{
$match:
{subject:"OSD"}},{$limit:1}
]);
```

```
db.student.aggregate([{
$match:
{subject:"OSD"}},{$skip:1}
]);
```

Sort()

```
db.student.aggregate([{ $match:{subject:"OSD"}},{$sort:{marks:-1}}]);
```

```
db.student.aggregate([{ $match:{subject:"OSD"}},{$sort:{marks:1}}]);
```

Unwind()

If following document is their in collection(Array)

- db.student.insert({rollno:9,name:"Anavi",marks:[80,30,50]});

Using Unwind the above document will be unwinded into 3 different document

- db.student.aggregate([{$unwind:"$marks"}])

Batch-01 Exercise

Create Company Database and Create Employee collection with following key

```
{
   "firstName": "John",
   "lastName": "Smith",
   "age": 25,
   "address":
   {
      "streetAddress": "21 2nd Street",
      "city": "New York",
      "state": "NY",
      "postalCode": "10021"
   },
   "phoneNumber":
   [
      {
       "type": "home",
       "number": "212 555-1234"
      },
      {
       "type": "fax",
       "number": "646 555-4567"
      }
   ]
“emailAddress": [ “romin.k.irani@gmail.com” , “tomhanks@gmail.com”]  }
```

Aggregation Operations : Solve Queries

1. Create database Company.
2. Create collection Employee.
3. Insert 10 documents with above mentioned structure.
4. Display all Employee Information.
5. Display state vise total employee
6. Display count of employees of ‘NY’ state
7. Display all the states where more than 10 employees are working
8. Display count of employees having age less than 30
9. Display name of employees having more than two phone nos.
10. How many employees have provided home phone no
11. Display employee vise count of email addresses.

Indexing and querying with MongoDB using suitable example.

- Import books.json
- Perform all these operations.
- Write your analysis

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

Batch-02 Exercise

Create Restaurant Database and Create Hotel collection with following keys

```
{
        Hotel Id:
        Hotel Name:
        Type:
        Ratings
        Address:{
                Area:
                City:
                Pincode:
            }
        Rooms :[
            {
                Roomno:
                Type:
                Price:
            }
            {
                Roomno:
                Type:
                Price:
            }

            ]
        Cuisines : [“Indian”,”Italian”,”Chinese”.....]
        likes:
}
```

Aggregation Operations : Solve Queries

1. Create database Restaurant.
2. Create collection Hotel.
3. Insert 10 documents with above mentioned structure.
4. Display all Hotel information.
5. Display no of rooms in each hotel
6. Compute the top five hotels
7. Return hotels having likes above 1000
8. Return the Five Most Common Cuisines
9. Return all prices of room in different hotel of type 'Deluxe' .
10. Get the total count of hotels having ratings '5 star'
11. Display the count of hotels from 'Pune' city.

Indexing and querying with MongoDB using suitable example.

- Import restaurants.json
- Perform all these operations.
- Write your analysis

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

Practice Assignment

Create Theatre Database and Create Movies collection with following keys

```
{
  "title": "",
  "year": "",
  "directors": [
    {
      "Name": "",
      "DOB": "",
      "Adress": "",
      "TelNo": ""
    }
  ],
  "writers": [
    {
      "Name": "",
      "DOB": "",
      "Adress": "",
      "TelNo": ""
    }
  ],
  "stars": [
    {
      "Name": "",
      "DOB": "",
      "Adress": "",
      "TelNo": ""“          }          ]
 “tags”:[“comedy”,”action”]
 “likes”:                    }
```

Aggregation Operations : Solve Queries

1. Create database Theatre.
2. Create collection Movie.
3. Insert 10 documents with above mentioned structure.
4. Display all Movies information.
5. Display the count of movies having tag ‘comedy’
6. Display three most common tags
7. Display the top five movies
8. Display the count of writers for each movie
9. Display the count of stars for each movie
10. Display the count of producers for each movie
11. Display the count of movies produced by each producers
12. Display the count of movies in which ‘Amir khan’ has worked as star
13. Display the name of writers who have written more than 3 movies
14. Display the name of movies ordered by year of release

Indexing and querying with MongoDB using suitable example.

- Import zips.json
- Perform all these operations.
- Write your analysis

Import Data

Create Index

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Show Existing Index

Hint

- Single Field Index
- Compound Field Indexes
- Multikey Indexes

Explain

Compare with data without indexes

Thank you