Find letter

{name: { $regex: `at`, $options: 'i' }}

{name = { $regex = /at$/ }}

$sum

([{$group: { _id: null, sum: { $sum: `$price` }}}

$avg

([$group: { _id: null, average: { $avg: `$price` }

$count

- db.user.count ({age: 25})
- db.user.countDocuments ({price: { $exists: true }})
- ([{ $group: { _id: null, count: { $sum: 1 } }}

$inc

{$inc: {age: 5}} {$inc: {age: -5}}

$upsert

updateone

({_id: 1}, {$set {name: `Alice`, age: 30}},
{upsert: true}

$unwind

{$unwind: `$age` }

$lookup

{ $lookup: {
from: `name of other collection`,
localField: ` field from current position`,
foreignField: `field from other collection`
as: ` ouput array name`

$ capped collection

db.createCollection (`name`, {capped: true,
size: 1024 * 1024,
max: 1000 })

$rename

- ([{}, { $rename: {`price`: `cost`}])
- db.collectionname.renameCollection (`newname`)

$out

{$out: `newcollectionname`}

$addToSet

if not add
or
if present not add

({name: `Alice`}, {$addToSet: {hobby:

$ bulkwrite

db.product.bulkwrite ([
{insertone: {
`document`: {
`name`: `Laptop`,
`price`: 1200,
`category`: `electronics`
}
}},
{updateone: {
`filter`: {name: `laptop`},
`update`: { $set: { `price`: 1000

},
{updatemany: {
.
.
},
{deleteOne: {
filter: {`name`: `oldLaptop`}
}

Multiple write operation in single batch (insert, update, delete)

$expr

db.product.find ({
$expr: {
$gt: [`$price`, `$goldPrice`]
}})

$elemMatch

keyname
([{gadget
: {
$elemMatch: {
product: `Laptop`,
quantity: { $gte: 2
}}})

ARRAY

create

({name}, {$set: {project: []}})

add

({2}, {$push: {project: {$each: [p2, p3

remove

({3}, {$pull: {project: p3}})
key name

$facet

aggregate

{ $facet: {
totalProducts: [
{$count: `total`}
],
averagePrice: [
{$group: {_id: null, avg:
]
}

output
averagePrice: [
_id: null, avg: 250

Index

- **Single field index**

```
db.collection.createIndex({name:1})
.find({name:joe})
```

- **compound index**

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

- **Multikey index**

```
db.collection.createIndex({tags:1})
```

- **Text index**

```
db.collection.createIndex({description:'text'})
```

- **Hashed index**

```
db.collection.createIndex({user_id:'hashed'})
```

- **Geospatial 2D index**

```
db.collection.createIndex({location:'2d'});
```

- **Geospatial 2Dsphere index**

```
db.collection.createIndex({location:'2dsphere'})
```

- **TTL index**

```
db.collection.createIndex({date:1},{expireAfterSeconds:600});
```

Promise fs

```
const fs=require('fs').promise
fs.readFile('sample.txt','utf8')
.then((data)=>{
  console.log(data)
}).catch((err)=>{
  console.log(err)
})
```

Promise greaty

```
function gt(a,b){
  return new Promise((res,rej)=>{
    if(a>b){
      res(`Promise resolved, ${a} greater than ${b}`)
    }else{
      rej(`Promise rejected, ${a} not greater than ${b}`)
    }
  })
}
gt(9,6).then((message)=>{
  console.log(message)
}).catch((err)=>{
  console.log(err)
})
```

then catch

asyncawait

```
async function handlePro(a,b){
  try{
    const result=await gt(a,b)
    console.log(result)
  }catch(err){
    console.log(err)
  }
}
handlePro(9,12)
```

Map

```
MyMap=new Map()
```

create

```
myMap.set('name','Alice')
```

read

```
myM.get('name')
```

check

```
MyMap.has('name')
```

delete

```
MyMap.delete('name')
```

clear

```
myMap.clear()
```

Pading

```
{$match:{address.city":"Baylore",
          "zip":2015
        }
},
{$group:{_id:null, avgSalary:{$avg:'$salary'}}}
```

Delete

```
db.collection.deleteOne({name:'joe'})
deleteMany
db.collection.drop()
```

$exists

```
{'field':{'$exists':<boolean>}}
```

$limit

```
{$limit:<number>}
```

$sort

- db.collection.find().sort()
- db.collection.aggregate({$sort:{salary:-1}},
  {$limit:1})

rest

```
function sum(...numbers){
  return numbers.reduce((total,num)=>total+num,0)
}
console.log(sum(1,2,3,4))
```