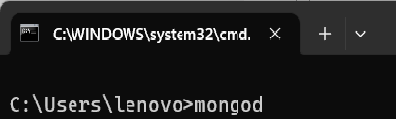
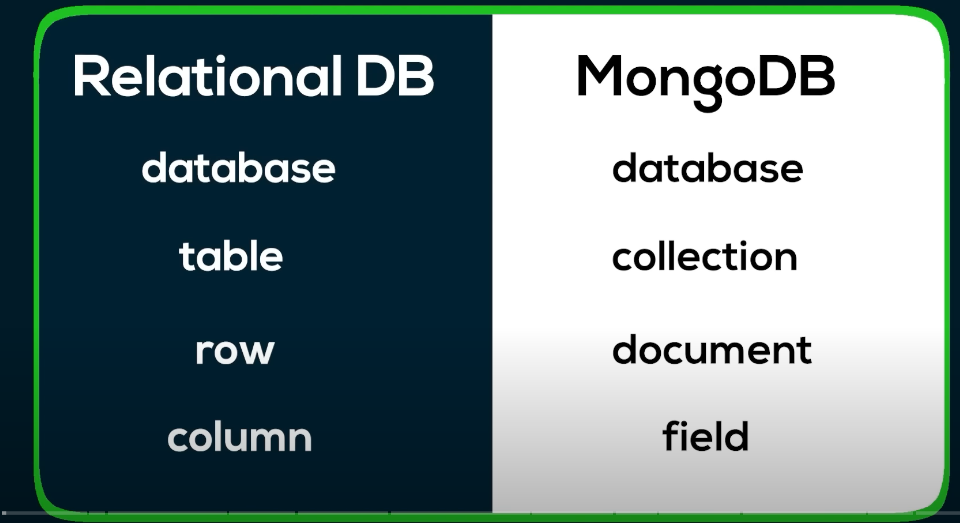
**MongoDB manual documentation:-**

[**https://www.mongodb.com/docs/manual/**](https://www.mongodb.com/docs/manual/)

**How to MongoDb driver start:**

****

* **Go to the command prompt then enter => mongod <=|**

****

|  |  |
| --- | --- |
| **Name** | **Description** |
| **$eq or:** | Matches values that are equal to a specified value |
| **$gt** | Matches values that are greater than a specified value |
| **$gte** | Matches values that are greater than or equal to a  specified value |
| **$in** | Matches any of the values specified in an array |
| **$lt** | Matches values that are less than a specified value |
| **$lte** | Matches values that are less than or equal to a specified value |
| **$ne** | Matches all values that are not equal to a specified value |
| **$nin** | Matches none of the values specified in an array |

**MongoDB: comparison operators :**

|  |  |  |
| --- | --- | --- |
| **MySQL** | **MongoDB** | **Description** |
| > | $gt | greater than |
| >= | $gte | greater equal then |
| **<** | **$lt** | **less than** |
| **=** | **$eq** | **equal to**  **The $eq expression is equivalent to**  **{ field: <value> }.** |
| **>** | **$gt** | **greater than** |
| **>=** | **$gte** | **greater equal then** |
| **!** | **$neq** | **Not equal to** |

**Example:**

|  |  |
| --- | --- |
| SELECT \*  FROM people WHERE **age < 25** | db.people.find(  { **age: { $lt: 25 }** }  ) |

**MongoDB: Aggregation Framework**

|  |  |
| --- | --- |
| **SQL** | **MongoDB** |
| WHERE | $match |
| GROUP BY | $group |
| HAVING | $match |
| SELECT | $project |
| ORDER BY | $sort |
| LIMIT | $limit |
| SUM | $sum |
| COUNT | $sum |

|  |  |
| --- | --- |
| **MySQL clause** | **MongoDB operator** |
| HAVING | aggregate($group, $match) |

|  |
| --- |
| SELECT status,  SUM(age) AS total FROM people  GROUP BY status  HAVING total > 1000 |
| db.orders.aggregate( [  {  $group: {  \_id: "$status",  total: { $sum: "$age" }  }  },  { $match: { total: { $gt: 1000 } } }  ] ) |

**Database Commands:**

**\*\*\*Create a new collection / New Database command:**

**db.createCollection("monthlyBudget");**

**1> View all databases:**

**show dbs;**

**if you want to use another table(document) same database then use:**

**db. db.monthlyBudget.find({}); => you will get/see all the presents under the collection.**

**2>View current Database:**

**Db;**

**3>Switch databases / Use Database:**

**use dbName;**

**4>Delete Database:**

**db.dropDatabase(); // for database delete**

**Or:**

**Db.book.drop(); // for collection delete**

**5> find() || Show Collections :**

**db.calculation.find();**

**Or:**

**db.calculation.find().pretty(); // pretty formate are used for perfectly show data.**

**Or:**

**db.calculation.find({name:'Sumit'});**

**Or:**

**find() with limit(): Limit the number of rows in output:**

**db.calculation.find().pretty().limit(1);**

**Or:**

**Find the first row matching the object**

**db.calculation.findOne({name:'Sumit'}) // Only one data show**

**6>Create Collection:**

**db.createCollection("<collection\_name>");**

**Example:**

**db.createCollection("product");**

**7>MongoDb commands single data, Insert Data :**

**db.students.insertOne({name:"Sumit",age:32,loc:"Kolkata",marks:45,mob:8356462545,sub:"Php"});**

**8> MongoDb multiple commands Insert Data :**

**db.students.insertMany([{name:"Anup",age:31,loc:"Kolkata",marks:56,mob:9336562547,sub:"C++"},{name:"Bappa",age:31,loc:"Delhi",marks:67,mob:8334586563,sub:"React"}]);**

**Or:**

**db.faculty.insertMany([{name:"Rajat",loc:"Kolkata",age:32,skills:["Python","asp.net"]},{name:"Sukanta",loc:"Kolkata",age:33,skills:["ReactJs","Php",”Mysql”]}]);**

**9>$eq Equals a Specified Value find:**

**db.students.find( { loc: { $eq: “Kolkata”} } );**

**OR:**

**db.students.find( { age: { $eq: 31} } );**

**OR:**

**db.students.findOne( { loc: { $eq: “Delhi”} } );**

**\*Note: $eq means ‘Equal’. find & findOne two different works. find is used for all data found which data table value find and findOne is only given to the table to show the first value from the table.**

**10>$ne is Not Equals a Specified Value find:**

**db.students.find( { loc: { $ne: “Kolkata”} } );**

**11>$gt is Greater than search:**

**db.students.find( { age: { $gt: 31} } );**

**12>$gte is Greater than Equal search:**

**db.students.find( { age: { $gte: 31} } );**

**13>$in is In the operator:**

This document describes the $in aggregation operator.

**db.students.find( { loc: { $in: [“Kolkata”,”Goa”,”Delhi”,”Patna”]} } );**

**\*Note: The $in operator selects the documents where the value of a field equals any value in the specified array. To specify an $in expression.**

**12. $and operator:**

**db.students.find( {$and:[{ loc: { $eq: “Kolkata”} },{age:{$gte:30}}]} );**

**OR:**

**db.students.find( {$and:[{ loc: { $ne: “Kolkata”} },{age:{$gte:30}}]} );**

**14> $or operator:**

**db.students.find( {$or:[{ loc: { $ne: “Kolkata”} },{age:{$gte:30}}]} );**

**15> $not operator:**

**db.students.find({sub:{$not:{$eq:"Php"}}});**

**16> $exists Operator: If you check whether the file name exists or not, then you can run this code.**

**db.students.find({sub:{$exists:true,$eq:"Php"}});**

**Or:**

**db.students.find({sub:{$exists:true,$eq:"Php"}});**

**17> $exists Operator:**

**db.monthlyBudget.find({$expr:{$gt:["$budget","$spent"]}});**

**OR:**

**db.monthlyBudget.find({$expr:{$lt:["$budget","$spent"]}});**

**18> $mod operator:**

**db.monthlyBudget.find({budget:{$mod:[200,0]}});**

**19> $all Operator:**

**db.faculty.find({skills:{$all:["Java"]}});**

**Or:**

**db.faculty.find({skills:{$all:["Php",”ReactJs”]}});**

**20>Update a Document:**

**db.<collection\_name>.updateOne({**

**<key>: <value>**

**}, {**

**$set: {**

**<key>: <value>**

**}**

**});**

**This command will update the first document in the collection you are using that matches the query. $set is used to update the document.**

**21>Add new field to a Document:**

**db.<collection\_name>.updateOne({**

**<key>: <value>**

**}, {**

**$set: {**

**<new\_key>: <new\_value>**

**}**

**});**

**22>Drop Collection:**

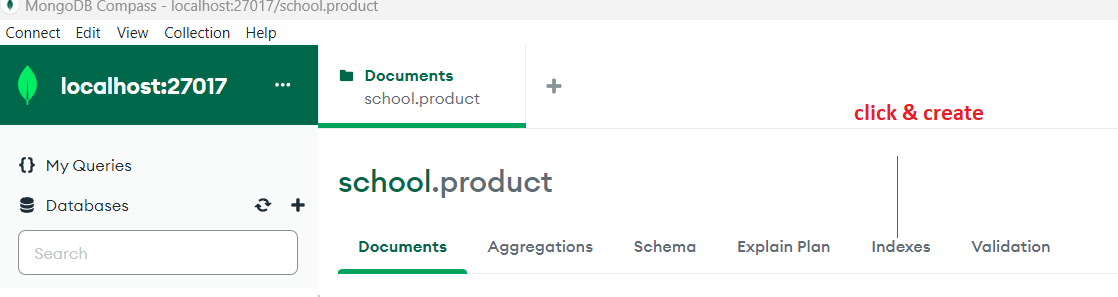
**db.<collection\_name>.drop();**

**23>** **Creating an index**

**$text search:**

**When you text search, first it will create a table ‘index creation’, (go to indexes and chose the option then create.)**

**db.product.find({$text:{$search:"Indian"}});**

****

**Code:**

**db.collection.createIndex(<index keys>, <options>)**

**24>Update table field data:**

**db.books.updateOne({name:'PHP'},{$set:{name:'UpdayeBook',WR:'UpdateAuthor'}});**

**Or:**

**Aggregation Pipeline Operators: (Aggregation function):**

Sum:- $add

**db.exam.aggregate([{$project:{total\_marks:{$add:["$phy","$chem","$bio"]}}}]);**

**Note: $project menace of which filled show.**

**$floor:**

**db.exam.aggregate([{$project:{floor\_val:{$floor:{$round:[{$sqrt:"$chem"},1]}}}}]);**

**Basic Salary Calculation:**

**Create a collection.**

**db.createCollection("salary");**

**Data insert into the collection:**

**db.salary.insertMany([{name:"Anup",basicsal:5000,hr:300,lta:500,pf:1000},{name:"Somnath",basicsal:6000,hr:350,lta:550,pf:1500}]);**

**Sum basic salary calculation command code:**

**db.salary.aggregate([{$project:{basic:{$add:["$basicsal","$hr","$lta"]}}}]);**

**Calculation of basic salary:**

**db.salary.aggregate([{$project:{name:1,gross\_salary:{$subtract:[{$add:["$basicsal","$hr","$lta"]},"$pf"]} }}]);**

**$aggregate:**

**db.faculty.aggregate([{$project:{name:1,loc:1,favoriteskill:{$arrayElemAt:["$skills",1]}}}]);**

\*\*\*It’s only array purpose use.

**Array To ObjectNotation:**

db.arrarToObject.insertMany([

{uid:"u100",info:[["id",10],["city","kolkata"],["age",35]]}

]);

db.arrarToObject.insertMany([

{uid:"ul01",info:[{"k":"id","v":15},{"k":"city","v":"pune"},{"k":"age","v":40}]}

])

**Query:**

**db.arrarToObject.aggregate([{$project:{uid:1,resultobj:{$arrayToObject:"$info"}}}]);**

Concatenate Array

db.arrmerge.insertMany([

{uid:11,sub:["math","physics"],books:["c++ book","math book"]},{uid:12,sub:["english","biology"],books:["english book","biology book"]}

]);

**Query:**

**db.arrmerge.aggregate([**

**{$project:{uid:1,knowledge:{$concatArrays:["$sub","$books"]}}}**

**]);**

**Array True & False**

**db.faculty.aggregate([{$project:{name:1,loc:1,isactive:{$in:["react","$skills"]}}}]);**

**Array Index find:**

**db.createCollection("fruits");**

**db.fruits.insertMany([**

**{find:10,fruitnm:["apple","grapes","banana","apple","guava","apple"]}**

**]);**

**Query:**

**db.fruits.aggregate([{$project:{fid:1,srchresult:{$indexOfArray:["$fruitnm","grapes"]}}}]);**

**$distinct:**

**db.runCommand({distinct:"fruits",key:"fruitnm"});**

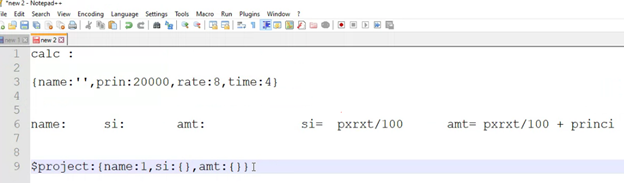
**Calculate simple interest annually:**

**Create a collection:**

**db.createCollection("calculation");**

**Data insert into the collection:**

**db.calculation.insertMany([{name:"Anup",prin:2000,rate:8,time:4},{name:"Sumit",prin:3000,rate:6,time:5}]);**

****

**Si: calculation formula => [ si = (p \* r \* t ) / 100 ]**

**db.calculation.aggregate([{$project:{name:1,si:{$divide:[{$multiply:["$prin","$rate","$time"]},100]}}}]);**

**Si & Amount calculation:**

**db.calculation.aggregate([{$project:{name:1,si:{$divide:[{$multiply:["$prin","$rate","$time"]},100]},**

**amount:{$add:["$prin",{$divide:[{$multiply:["$prin","$rate","$time"]},100]}]}}}]);**

**$count :**

**db.calculation.find().count(); // total collection**

**Or**

**db.calculation.find({name:'Sumit'}).count();**

**$sort:**

**db.calculation.find().sort({name:1}).pretty(); // ASC list wise data show**

**Or:**

**db.calculation.find().sort({name:1}).pretty(); // DESC list wise data show**

**$inc Increment Operator:**

**db.calculation.update({name:'Amit'},**

**{$inc:{**

**time:6**

**}});**

**$rename Rename Operator:**

**db.calculation.update({name:'Amit'},**

**{$rename:{**

**time:'times'**

**}});**

**Or: if you change all field names then use this code:**

**db.calculation.update(**

**{$rename:{**

**time:'times'**

**}});**

**MongoDB atlas login:**

[**https://www.mongodb.com/cloud/atlas/register**](https://www.mongodb.com/cloud/atlas/register)

**A screenshot of a computer

Description automatically generated**

**$slice Operator:**

**db.books.find({page:{$gt:350}},{price:{$slice:2}});**

**Case Sensitivity Considerations in MongoDB:**

**{ordered:false}: when many data insert into the collection then you add ‘ordered false’ because if any data value wrong then last correct value is inserted.**

**Exc.**

**db.students.insertMany([{name:"Mita",age:31},{\_id:objectId(“64d5c75ce8”),name:"Mithu",age:31},{name:"Bappa",age:31},** **{ordered:false}]);**



**Wrong ID created, So it show error but correct data are insert into collection**

**How to Import Collection in MongoDb:**

