**ASSIGNMENT – 3**

**Aim:** Create a suitable collection in MongoDB and perform 10 queries to demonstrate various operations on documents.

**Theory**

In mongodb querie help us show different outputs based on the conditions(if any). Queries also help us to show proper outputs.

## **The find() Method**

To query data from mongodb collection, you need to use mongodb's find() method.

### Syntax

The basic syntax of find() method is as follows −

>db.COLLECTION\_NAME.find()

Find() method will display all the documents in a non-structured way.

## **The pretty() Method**

To display the results in a formatted way, you can use pretty() method.

### Syntax

>db.mycol.find().pretty(

O query the document on the basis of some condition, you can use following operations.

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Syntax | Example | RDBMS Equivalent |
| Equality | {<key>:<value>} | Db.mycol.find({"by":"tutorials point"}).pretty() | Where by = 'tutorials point' |
| Less Than | {<key>:{$lt:<value>}} | Db.mycol.find({"likes":{$lt:50}}).pretty() | Where likes < 50 |
| Less Than Equals | {<key>:{$lte:<value>}} | Db.mycol.find({"likes":{$lte:50}}).pretty() | Where likes <= 50 |
| Greater Than | {<key>:{$gt:<value>}} | Db.mycol.find({"likes":{$gt:50}}).pretty() | Where likes > 50 |
| Greater Than Equals | {<key>:{$gte:<value>}} | Db.mycol.find({"likes":{$gte:50}}).pretty() | Where likes >= 50 |
| Not Equals | {<key>:{$ne:<value>}} | Db.mycol.find({"likes":{$ne:50}}).pretty() | Where likes != 50 |

Collections in MongoDB corresponds to the table in SQL. Rows in SQL table is indicated as documents in MongoDB collections. Collection can be created by writing the following command:

**db.createCollection()**

**db.createCollection(name, options)**

Creates a new collection or view. For views, see also db.createView().

Because MongoDB creates a collection implicitly when the collection is first referenced in a command, this method is used primarily for creating new collections that use specific options. For example, you use db.createCollection() to create a capped collection, or to create a new collection that uses document validation.

db.createCollection() is a wrapper around the database command [create](https://docs.mongodb.com/manual/reference/command/create/#mongodb-dbcommand-dbcmd.create).

The db.createCollection() method has the following prototype form:

**Starting in MongoDB 4.2**

MongoDB removes the MMAPv1 storage engine and the MMAPv1 specific options paddingFactor, paddingBytes, preservePadding for db.createCollection().

|  |
| --- |
| db.createCollection( <name>, |
| { |
| capped: <boolean>, |
| timeseries: { *// Added in MongoDB 5.0* |
| timeField: <string>, *// required for time series collections* |
| metaField: <string>, |
| granularity: <string> |
| }, |
| expireAfterSeconds: <number>, |
| autoIndexId: <boolean>, |
| size: <number>, |
| max: <number>, |
| storageEngine: <document>, |
| validator: <document>, |
| validationLevel: <string>, |
| validationAction: <string>, |
| indexOptionDefaults: <document>, |
| viewOn: <string>, *// Added in MongoDB 3.4* |
| pipeline: <pipeline>, *// Added in MongoDB 3.4* |
| collation: <document>, *// Added in MongoDB 3.4* |
| writeConcern: <document> |
| } |
| ) |

The db.createCollection() method has the following parameters:

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| name | string | The name of the collection to create. |
| options | document | Optional. Configuration options for creating a capped collection, for pre-allocating space in a new collection, or for creating a view. |

**Create a Capped Collection**

Capped collections have maximum size or document counts that prevent them from growing beyond maximum thresholds. All capped collections must specify a maximum size and may also specify a maximum document count. MongoDB removes older documents if a collection reaches the maximum size limit before it reaches the maximum document count. Consider the following example:

|  |
| --- |
| db.createCollection(**"log"**, { capped : true, size : 5242880, max : 5000 } ) |

This command creates a collection named log with a maximum size of 5 megabytes and a maximum of 5000 documents.

Query operations can be performed using the db.collection.find() method in mongosh. The examples on this page use the inventory collection. To populate the inventory collection, run the following:

|  |
| --- |
| db.inventory.insertMany([ |
| { item: **"journal"**, qty: 25, size: { h: 14, w: 21, uom: **"cm"** }, status: **"A"** }, |
| { item: **"notebook"**, qty: 50, size: { h: 8.5, w: 11, uom: **"in"** }, status: **"A"** }, |
| { item: **"paper"**, qty: 100, size: { h: 8.5, w: 11, uom: **"in"** }, status: **"D"** }, |
| { item: **"planner"**, qty: 75, size: { h: 22.85, w: 30, uom: **"cm"** }, status: **"D"** }, |
| { item: **"postcard"**, qty: 45, size: { h: 10, w: 15.25, uom: **"cm"** }, status: **"A"** } |
| ]); |

**Select All Documents in a Collection**

To select all documents in the collection, pass an empty document as the query filter parameter to the find method. The query filter parameter determines the select criteria:

|  |
| --- |
| db.inventory.find( {} ) |

**Specify Equality Condition**

To specify equality conditions, use <field>:<value> expressions in the query filter document:

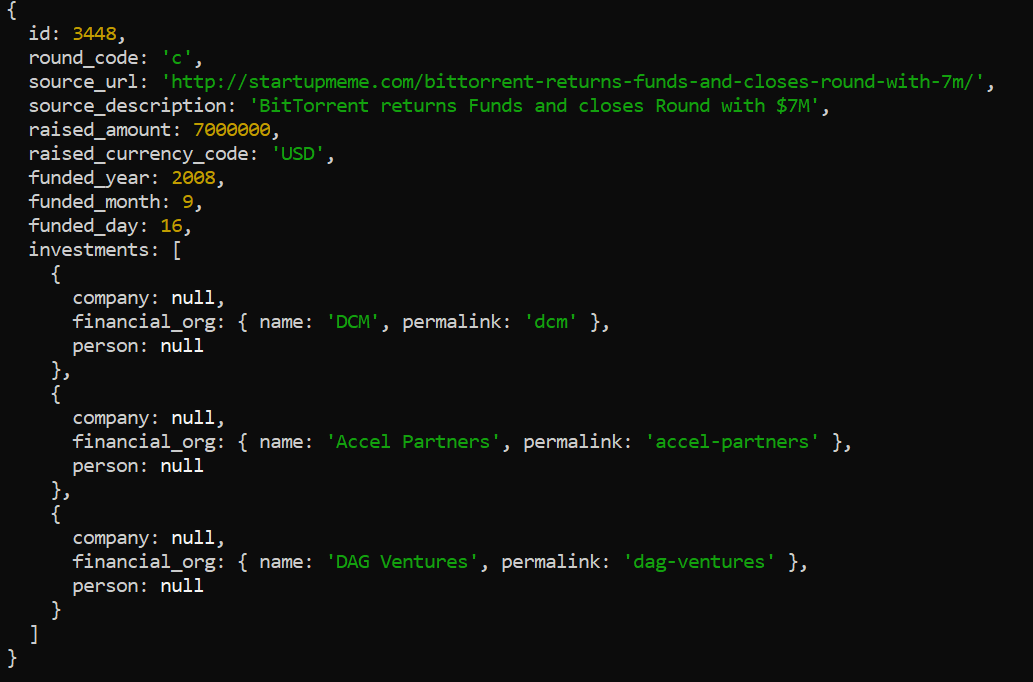
|  |
| --- |
| { <field1>: <value1>, ... } |

### Logical

| **Name** | **Description** |
| --- | --- |
| $and | Joins query clauses with a logical AND returns all documents that match the conditions of both clauses. |
| $not | Inverts the effect of a query expression and returns documents that do not match the query expression. |
| $nor | Joins query clauses with a logical NOR returns all documents that fail to match both clauses. |
| $or | Joins query clauses with a logical OR returns all documents that match the conditions of either clause. |

**Q1. Display all the documents from the companies collection where the founded\_year equals 2004:**

|  |
| --- |
| db.companies.find( {“founded\_year”: 2004 } ).pretty() |



The MongoDB Compass query bar autocompletes the current query based on the keys in your collection's documents, including keys in embedded sub-documents.

**Specify Conditions Using Query Operators**

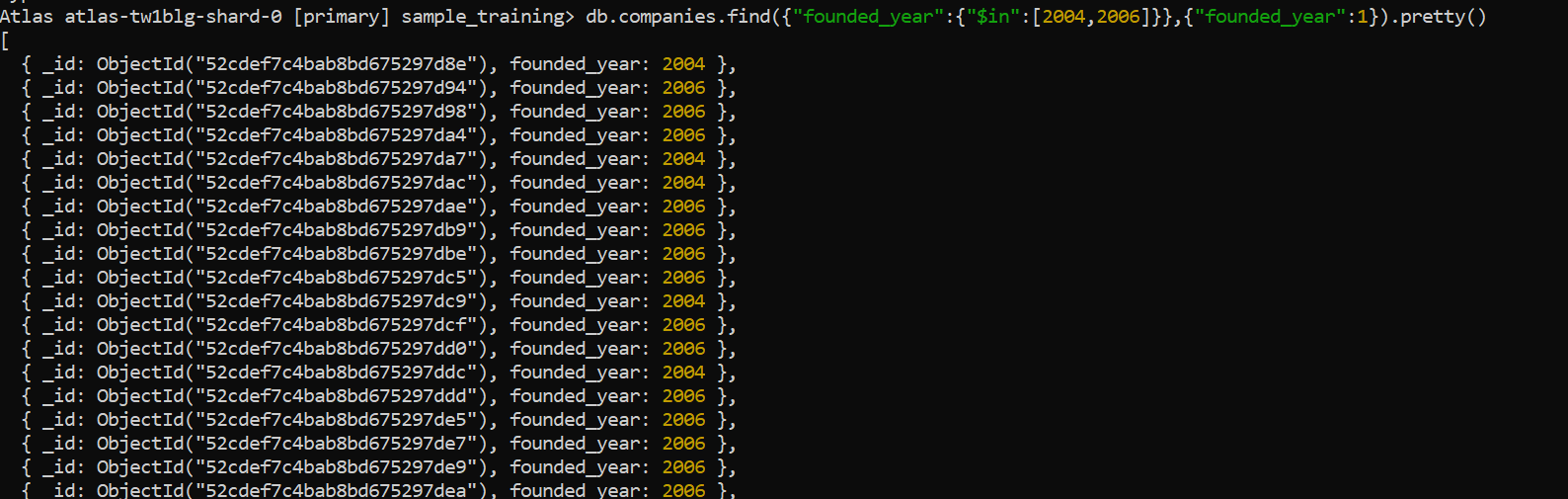
A query filter document can use the query operators to specify conditions in the following form:

|  |
| --- |
| { <field1>: { <operator1>: <value1> }, ... } |

**Q2 Display all documents from the companies collection where founded\_year equals either 2004 or 2008:**

|  |
| --- |
| db.companies.find( {“found\_year”: { $in: [ 2004, 2006 ] } } ) |

Although you can express this query using the [$or](https://docs.mongodb.com/manual/reference/operator/query/or/#mongodb-query-op.-or) operator, use the [$in](https://docs.mongodb.com/manual/reference/operator/query/in/#mongodb-query-op.-in) operator rather than the [$or](https://docs.mongodb.com/manual/reference/operator/query/or/#mongodb-query-op.-or) operator when performing equality checks on the same field.



**Specify AND Conditions**

A compound query can specify conditions for more than one field in the collection's documents. Implicitly, a logical AND conjunction connects the clauses of a compound query so that the query selects the documents in the collection that match all the conditions.

**Q3. Display all documents in the listingsAndReviews collection where the price is less than (**[**$lt**](https://docs.mongodb.com/manual/reference/operator/query/lt/#mongodb-query-op.-lt)**) 200:**

|  |
| --- |
| db.listingsAndReviews.find({ “price”: { “$lt”: 200 } } ) |
|  |

**Specify OR Conditions**

Using the $or operator, you can specify a compound query that joins each clause with a logical OR conjunction so that the query selects the documents in the collection that match at least one condition.

**Q4. Display all documents in the collection where the property\_type equals "Apartment" or price is less than (**[**$lt**](https://docs.mongodb.com/manual/reference/operator/query/lt/#mongodb-query-op.-lt)**) 200:**

|  |
| --- |
| db.listingsAndReviews.find( { $or: [ { “property\_type”: **"Apartment"** }, { “price”: { “$lt”: 200 } } ] } ) |

\

**Specify AND as well as OR Conditions**

**Q5. Display all documents in the collection where the property\_type equals "Apartment" and *either* price is less than ($lt) 200 *or* room\_type starts with the character E:**

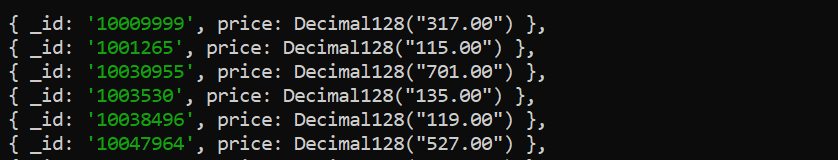
|  |
| --- |
| db.listingsAndReviews.find( { “property\_type”: **"Apartment"**, $or: [ { “price”: { $lt: 200 } }, { “room\_type”: /^E } ] } ) |
|  |
|  |
|  |

# **Comparison Query Operators**

Following are various comparison operators in MondoDB:

| **Name** | **Description** |
| --- | --- |
| $eq | 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. |

**Q6. Display all documents in listingsAndReviews where the price  is greater than or equal to 100.**

db.listingsandReview.find({“price”:{“$gte”:100}})

### Element

| **Name** | **Description** |
| --- | --- |
| $exists | Matches documents that have the specified field. |
| $type | Selects documents if a field is of the specified type. |

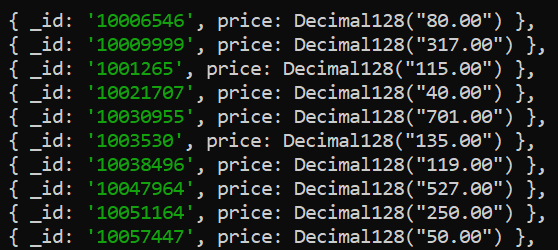
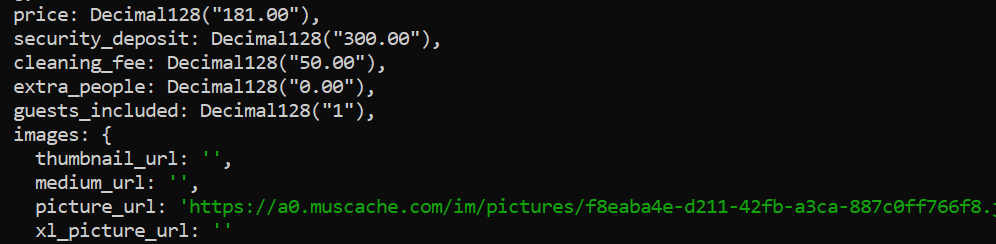
# **$exists**

Syntax: { field: { $exists: <boolean> } }

When <boolean> is true, $exists matches the documents that contain the field, including documents where the field value is null. If <boolean> is false, the query returns only the documents that do not contain the field.

MongoDB $exists does **not** correspond to SQL operator exists. For SQL exists, refer to the [$in](https://docs.mongodb.com/manual/reference/operator/query/in/#mongodb-query-op.-in) operator.

**Q7 Display all documents in the inventory collection where the price field exists *and* its value does not equal 100 or 200.**

db.listingsAndReviews.find( { “price”: { $exists: true, $nin: [ 100, 200 ] } } ) 

### Null Values

The following examples uses a collection named records with the following documents:

|  |
| --- |
| { a: 5, b: 5, c: null } |
| { a: 3, b: null, c: 8 } |
| { a: null, b: 3, c: 9 } |
| { a: 1, b: 2, c: 3 } |
| { a: 2, c: 5 } |
| { a: 3, b: 2 } |
| { a: 4 } |
| { b: 2, c: 4 } |
| { b: 2 } |
| { c: 6 } |

#### $exists: true

**Q8. a) Write the query that specifies the query predicate a: { $exists: true }:**

|  |
| --- |
| db.records.find( { a: { $exists: true } } ) |

The results consist of those documents that contain the field a, including the document whose field a contains a null value:

|  |
| --- |
| { a: 5, b: 5, c: null } |
| { a: 3, b: null, c: 8 } |
| { a: null, b: 3, c: 9 } |
| { a: 1, b: 2, c: 3 } |
| { a: 2, c: 5 } |
| { a: 3, b: 2 } |
| { a: 4 } |

#### $exists: false

**Q8. b) Write the query that specifies the query predicate b: { $exists: false }:**

|  |
| --- |
| db.records.find( { b: { $exists: false } } ) |

The results consist of those documents that do not contain the field b:

|  |
| --- |
| { a: 2, c: 5 } |
| { a: 4 } |
| { c: 6 } |

Consider the following examples use the listingsAndReviews collection that contains the documents:

|  |
| --- |
| \_id: '1459741',  name: 'Loft au centre-ville de Montréal',  summary: 'Superbe loft au coeur de Montréal et de ses activités. Stationnement publique intérieur disponible à quelques mètres pour 20$ par jour ou 10$ apres 18h.',  space: '',  property\_type: 'Loft',  room\_type: 'Entire home/apt',  bed\_type: 'Real Bed',  bedrooms: 0,  beds: 2,  number\_of\_reviews: 27,  bathrooms: Decimal128("1.0"),  amenities: [  'Internet',  'Wifi',  'Air conditioning',  'Kitchen',  'Breakfast',  'Elevator',  'Heating',  'Family/kid friendly',  'Washer',  'Dryer',  'Smoke detector',  'Carbon monoxide detector',  'Fire extinguisher',  'Essentials',  'Shampoo',  '24-hour check-in',  'Hangers',  'Hair dryer',  'Iron',  'Laptop friendly workspace'  ],  price: Decimal128("125.00"),  weekly\_price: Decimal128("500.00"),  cleaning\_fee: Decimal128("20.00"),  extra\_people: Decimal128("15.00"),  guests\_included: Decimal128("1"), |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Use $all to Match Values**

**Q9. Display all documents from listingsAndReviews collection where the value of the amenities field is an array whose elements include Internet, Kitchen, and Heating.**

|  |
| --- |
| db.listingsAndReviews.find( { “amenities” : { $all: [ **"Internet"**, **"Kitchen"**, **"Heating"** ] } } ) |

The above query returns the following documents:

|  |
| --- |
| { |
| \_id: ‘2432138, |
| amenities: [  'TV',  'Cable TV',  'Internet',  'Wifi',  'Kitchen',  'Pets allowed',  'Pets live on this property',  'Cat(s)',  'Heating',  'Family/kid friendly'  ] |
|  |
|  |
|  |
|  |
|  |
|  |
| } |
|  |
| { |
| \_id: '2456142',  amenities: [  'TV',  'Internet',  'Wifi',  'Kitchen',  'Free parking on premises',  'Buzzer/wireless intercom',  'Heating',  'Washer',  'Dryer',  'Smoke detector'  ]  } |

**Use $all with $elemMatch**

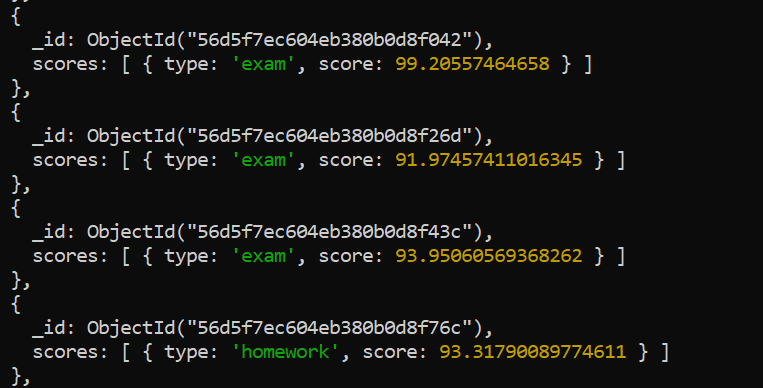
If the field contains an array of documents, we can use the $all with the $elemMatch operator.

**Q10. Display all documents from the inventory collection where the value of the qty field is an array whose elements match the $elemMatch criteria.**

|  |
| --- |
|  |
|  |
| db.grades.find({ "class\_id": 431 },  { "scores": { "$elemMatch": { "score": { "$gt": 85 } } }  }).pretty() |

The query returns the following documents:

|  |
| --- |
| {  \_id: ObjectId("56d5f7ec604eb380b0d8f042"),  scores: [ { type: 'exam', score: 99.20557464658 } ]  },  {  \_id: ObjectId("56d5f7ec604eb380b0d8f26d"),  scores: [ { type: 'exam', score: 91.97457411016345 } ]  },  {  \_id: ObjectId("56d5f7ec604eb380b0d8f43c"),  scores: [ { type: 'exam', score: 93.95060569368262 } ]  },  {  \_id: ObjectId("56d5f7ec604eb380b0d8f76c"),  scores: [ { type: 'homework', score: 93.31790089774611 } ]  }, |

****

**Conclusion:** Thus we have performed the queries in MongoDB by using various operators and keywords.