

Task 1

Operation 1:

The screenshot shows the MongoDB Shell interface. The top bar indicates the current database is 'testMongodB'. The command 'use testMongodB' has been entered and executed. The output shows the execution time as 0.013 s and the message 'switched to db testMongodB'.

```
* test:restaurants@localhost x
localhost testMongodB
1 use testMongodB

0.013 s Show Timestamps
1 switched to db testMongodB
```

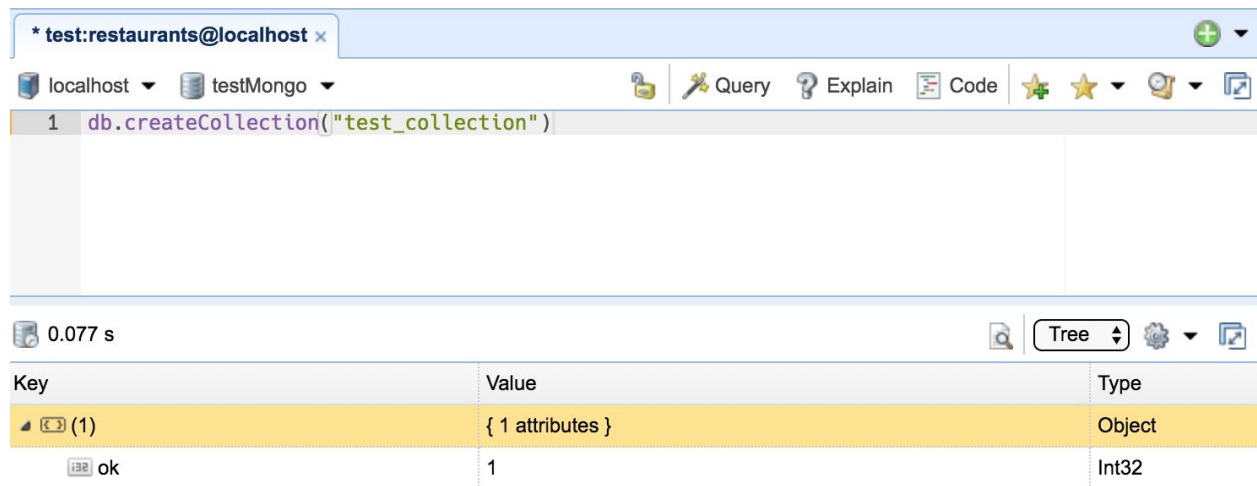
Operation 2:

The screenshot shows the MongoDB Shell interface. The command 'db.dropDatabase()' has been entered and executed. The output shows the execution time as 0.005 s and the message 'ok'. The output is displayed in a table format.

```
* test:restaurants@localhost x
localhost testMongodB
1 db.dropDatabase()

0.005 s
Tree
Key Value Type
(1) { 1 attributes } Object
ok 1 Int32
```

Operation 3:



The screenshot shows the MongoDB Shell interface. The command `db.createCollection("test_collection")` has been executed. The output is displayed in a table with three columns: Key, Value, and Type.

Key	Value	Type
<code>(1)</code>	<code>{ 1 attributes }</code>	Object
<code>ok</code>	<code>1</code>	Int32

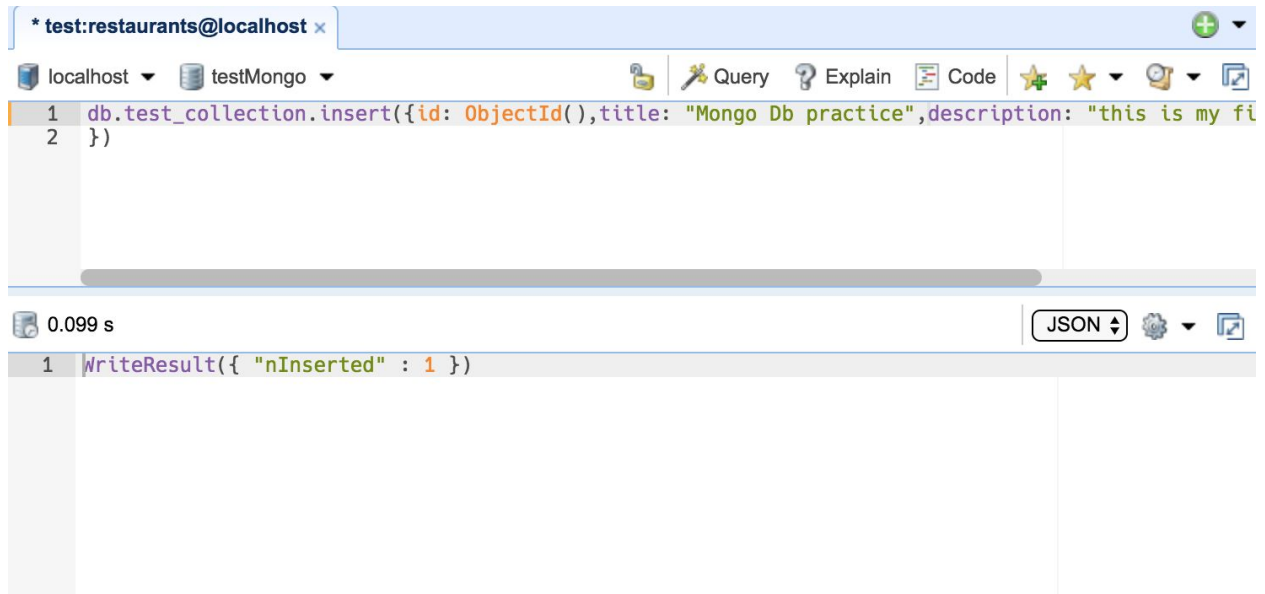
Operation 4:



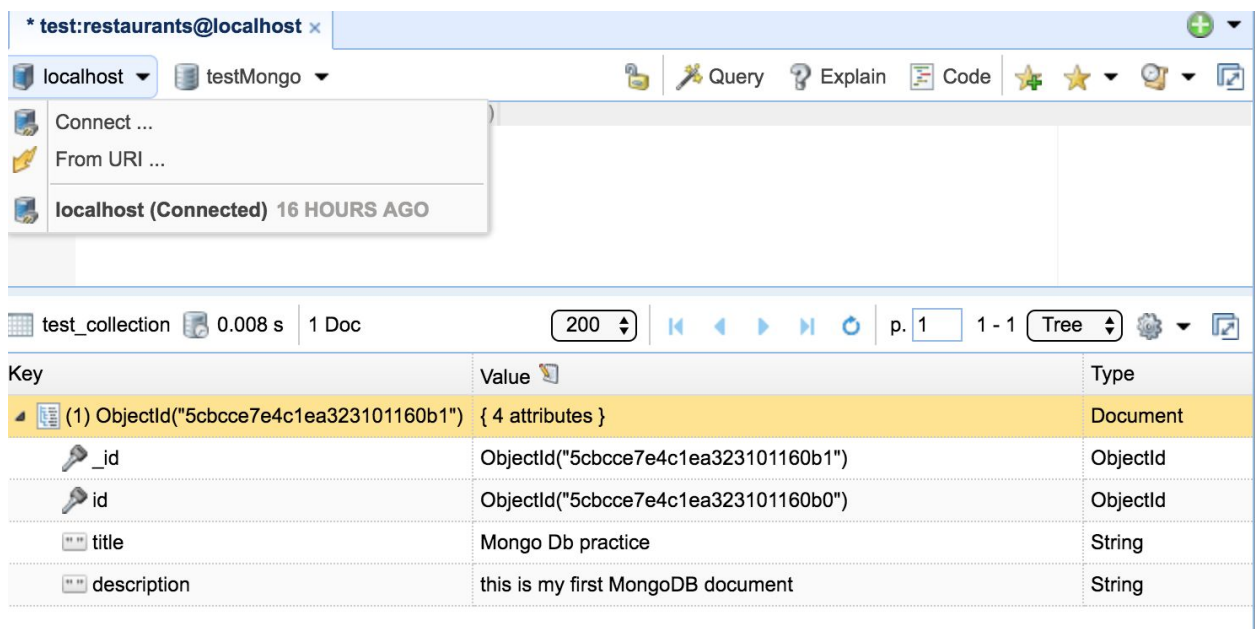
The screenshot shows the MongoDB Shell interface. The command `db.test_collection.drop()` has been executed. The output is displayed in a table with three columns: Key, Value, and Type.

Key	Value	Type
<code>1</code>	<code>true</code>	Boolean

Operation 5:



Operation 6:



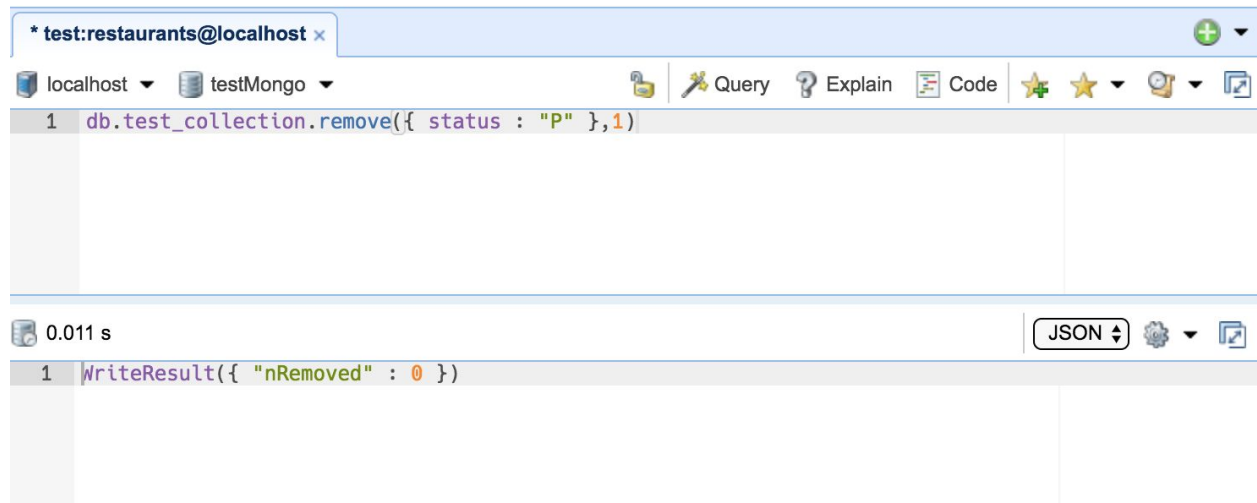
Operation 7:



The screenshot shows the MongoDB Shell interface. The top bar indicates the current database is `test:restaurants@localhost`. The command entered is `db.test_collection.update({'title':'MongoDB practice'},{$set:{ 'title':'New MongoDB practice'}})`. The execution time is 0.015 s. The result is `WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })`.

```
* test:restaurants@localhost x
localhost testMongo
1 db.test_collection.update({'title':'MongoDB practice'},{$set:{ 'title':'New MongoDB practice'}})
0.015 s
1 WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
```

Operation 8:



The screenshot shows the MongoDB Shell interface. The top bar indicates the current database is `test:restaurants@localhost`. The command entered is `db.test_collection.remove({ status : "P" },1)`. The execution time is 0.011 s. The result is `WriteResult({ "nRemoved" : 0 })`.

```
* test:restaurants@localhost x
localhost testMongo
1 db.test_collection.remove({ status : "P" },1)
0.011 s
1 WriteResult({ "nRemoved" : 0 })
```

Task 2

Query 1:

```
db.restaurants.find( { name: /Ice Cream/ }, { name: 1 })
```

* test:restaurants@localhost

localhost test

Query Explain Code

```
1 db.restaurants.find( { name: /Ice Cream/ }, { name: 1 } )
```

restaurants 0.024 s 85 Docs

200 1 - 85 Tree

Key	Value	Type
(1) ObjectId("5cbbd1a2f2f273eff1dbcdab")	{ 2 attributes }	Document
_id	ObjectId("5cbbd1a2f2f273eff1dbcdab")	ObjectId
name	Carvel Ice Cream	String
(2) ObjectId("5cbbd1a2f2f273eff1dbcdac")	{ 2 attributes }	Document
(3) ObjectId("5cbbd1a2f2f273eff1dbcd0")	{ 2 attributes }	Document
(4) ObjectId("5cbbd1a2f2f273eff1dbcdbd")	{ 2 attributes }	Document
(5) ObjectId("5cbbd1a2f2f273eff1dbcdcb")	{ 2 attributes }	Document
(6) ObjectId("5cbbd1a2f2f273eff1dbd139")	{ 2 attributes }	Document
(7) ObjectId("5cbbd1a2f2f273eff1dbd482")	{ 2 attributes }	Document
(8) ObjectId("5cbbd1a2f2f273eff1dbd6bc")	{ 2 attributes }	Document
(9) ObjectId("5cbbd1a2f2f273eff1dbd9f5")	{ 2 attributes }	Document
(10) ObjectId("5cbbd1a2f2f273eff1dbda63")	{ 2 attributes }	Document
(11) ObjectId("5cbbd1a2f2f273eff1dbdaaa")	{ 2 attributes }	Document
(12) ObjectId("5cbbd1a2f2f273eff1dbdb33")	{ 2 attributes }	Document
(13) ObjectId("5cbbd1a2f2f273eff1dbdb79")	{ 2 attributes }	Document
(14) ObjectId("5cbbd1a2f2f273eff1dbde1e")	{ 2 attributes }	Document
(15) ObjectId("5cbbd1a2f2f273eff1dbde4a")	{ 2 attributes }	Document
(16) ObjectId("5cbbd1a2f2f273eff1dbde75")	{ 2 attributes }	Document

Free Edition Show Log Feedback 02:16:26 pm

Query 2:

```
db.restaurants.find(
  { $and: [{ cuisine : { $in : ["Italian","American"] } }, { borough: "Brooklyn" }
]
}
)
```

*test:restaurants@localhost x

localhost test

Query Explain Code

```
1 db.restaurants.find({$and: [{cuisine :{$in :["Italian","American"]}}, {borough: "Brooklyn"}]})
```

restaurants 0.017 s 1,465 Docs

200 p. 1 1 - 200 Tree

Key	Value	Type
(1) ObjectId("5cbbd1a2f2f273eff1dbcd9e")	{ 7 attributes }	Document
_id	ObjectId("5cbbd1a2f2f273eff1dbcd9e")	ObjectId
address	{ 4 attributes }	Object
borough	Brooklyn	String
cuisine	American	String
grades	Array[4]	Array
name	Riviera Caterer	String
restaurant_id	40356018	String
(2) ObjectId("5cbbd1a2f2f273eff1dbcd4")	{ 7 attributes }	Document
(3) ObjectId("5cbbd1a2f2f273eff1dbcd6")	{ 7 attributes }	Document
(4) ObjectId("5cbbd1a2f2f273eff1dbcdad")	{ 7 attributes }	Document
(5) ObjectId("5cbbd1a2f2f273eff1dbcd4")	{ 7 attributes }	Document
(6) ObjectId("5cbbd1a2f2f273eff1dbcd1")	{ 7 attributes }	Document
(7) ObjectId("5cbbd1a2f2f273eff1dbcd7")	{ 7 attributes }	Document
(8) ObjectId("5cbbd1a2f2f273eff1dbcd6")	{ 7 attributes }	Document
(9) ObjectId("5cbbd1a2f2f273eff1dbcd6")	{ 7 attributes }	Document
(10) ObjectId("5cbbd1a2f2f273eff1dbcd6")	{ 7 attributes }	Document
(11) ObjectId("5cbbd1a2f2f273eff1dbcd5")	{ 7 attributes }	Document

Free Edition

Show Log Feedback 02:16:58 pm

Query 3:

db.restaurants.aggregate(

```
{
  $match : {cuisine : "American"}
},
```

```
{
  $group : {
    _id : "$borough",
    total : {$sum : 1}
  }
}
```

```
{
  $sort : { total : -1}
}
```

)

The screenshot shows the MongoDB Compass interface. The query editor contains the following aggregation pipeline:

```
1 db.restaurants.aggregate(
2
3
4 {
5   $match : {cuisine : "American"}
6 },
7
8
9 {
10  $group : {
11    _id : "$borough",
12    total : {$sum : 1}
13  }
```

The results pane shows 6 documents. The first five are grouped by borough, and the sixth is a 'Missing' document. The 'total' field is highlighted in red for each document.

Key	Value	Type
▲ (1) Manhattan	{ 2 attributes }	Document
_id	Manhattan	String
total	3,205 (3.2K)	Double
▲ (2) Brooklyn	{ 2 attributes }	Document
_id	Brooklyn	String
total	1,273 (1.3K)	Double
▲ (3) Queens	{ 2 attributes }	Document
_id	Queens	String
total	1,040 (1.0K)	Double
▲ (4) Bronx	{ 2 attributes }	Document
_id	Bronx	String
total	411	Double
▲ (5) Staten Island	{ 2 attributes }	Document
_id	Staten Island	String
total	244	Double
▲ (6) Missing	{ 2 attributes }	Document
_id	Missing	String
total	10	Double

Query 4:

```
db.restaurants.aggregate(
```

```
{
  $match : {cuisine : "American", borough: "Manhattan"}
},
```

```
{
  $unwind : "$grades"
```

```

}

{
  $group : {
    _id : {
      total : {$sum : "$grades.score"}
    }
  }

{
  $sort : { _id : -1}
}

).limit(5)

```

The screenshot shows the MongoDB Compass interface. The top bar indicates the connection to 'localhost' and the 'test' database. The query editor on the left contains the following aggregate query:

```

1 db.restaurants.aggregate(
2
3
4 {
5   $match : {cuisine : "American", borough: "Manhattan"}
6 },
7
8 {
9   $unwind : "$grades"
10 }
11
12 {
13   $group : {

```

The results pane on the right shows 5 documents. The first document is expanded, showing its structure:

Key	Value	Type
(1) { total : 131 }	{ 1 attributes }	Document
_id	{ 1 attributes }	Object
(2) { total : 90 }	{ 1 attributes }	Document
(3) { total : 89 }	{ 1 attributes }	Document
(4) { total : 84 }	{ 1 attributes }	Document
(5) { total : 82 }	{ 1 attributes }	Document

The bottom of the interface shows the 'Free Edition' logo, 'Show Log' and 'Feedback' buttons, and the time '11:48:28 pm'.

Query 5: