

# DS108-04-08 - NoSQL - Lesson 4

## Hands-On

For your Lesson 4 Hands-On, you will be working with your new knowledge on NoSQL. This Hands-On will be graded, so be sure you complete all requirements.

[!Caution] Caution! Do not submit your project until you have completed all requirements, as you will not be able to resubmit.

[!info] To Submit Be sure to zip and submit your NoSQL-HandsOn4 text document when finished! You will not be able to re-submit, so be sure the screenshots to each part are located within this document.

## Requirements

This Hands-On is structured into *two* parts, and each part may ask you to run multiple queries. After each query, please take a screenshot and add it to a text document (or an equivalent) and name this file NoSQL-HandsOn4. This way, you will be able to submit your answers to each part all at once. Good luck! \_\_\_\_

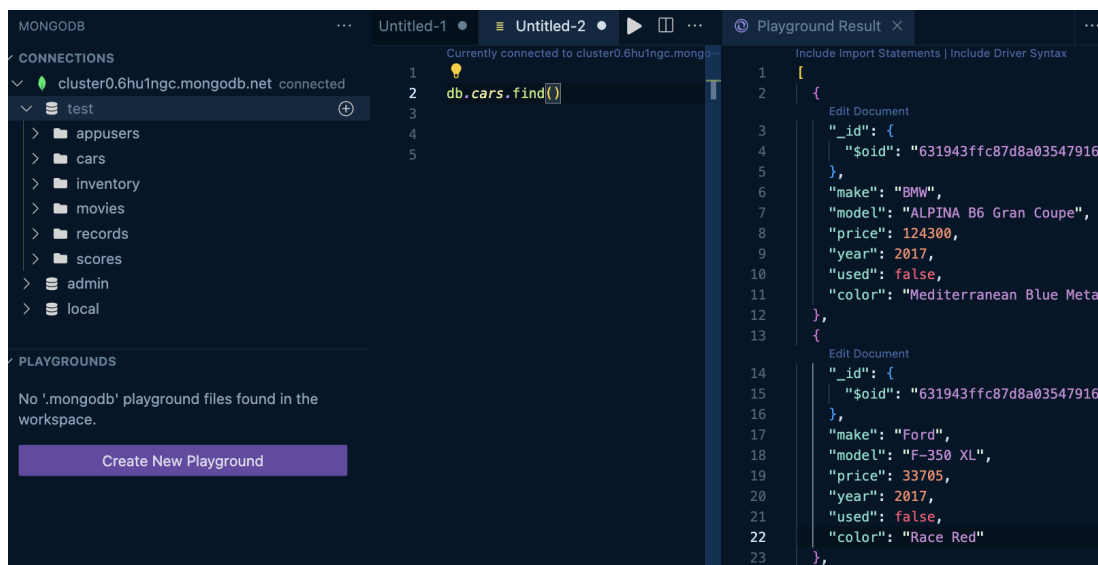
# Part 1

Follow the below steps:

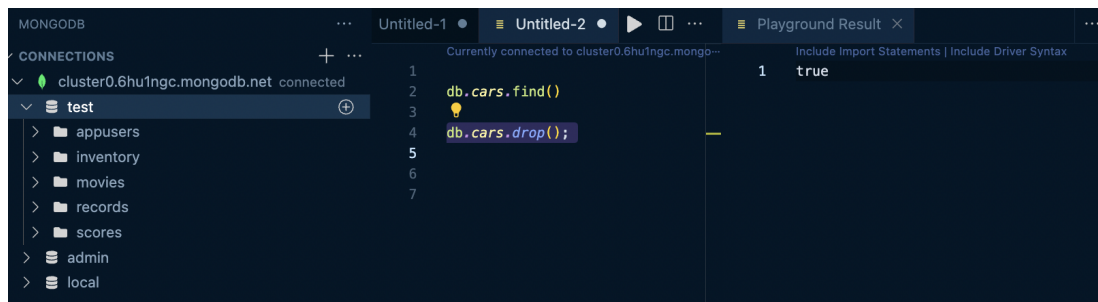
## 1.1 Delete the entire collection cars

- Start off by deleting the entire collection cars.
- Take a screenshot of the query *as well* as the list of your collections in Atlas to be sure this collection has been deleted.

`db.cars.find();`



`db.cars.drop();`



## 1.2 Run the query to recreate the cars collection

- Next, run the following query to recreate the cars collection.
- The following includes more cars than before.

```
db.cars.insertMany([
  {
    make: "Hyundai",
    model: "Santa Fe",
    price: 8000,
    year: 2003,
    used: true,
    color: "Black"
  },
  {
    make: "BMW",
    model: "ALPINA B6 Gran Coupe",
    price: 124300,
    year: 2017,
    used: false,
    color: "Mediterranean Blue Metallic"
  },
  {
    make: "Subaru",
    model: "Crosstrek 2.0i Premium",
    price: 22595,
    year: 2014,
    used: true,
    color: "Sunshine Orange"
  },
  {
    make: "Ford",
    model: "F-350 XL",
    price: 33705,
    year: 2017,
    used: false,
    color: "Race Red"
```

```
},
{
  make: "Toyota",
  model: "Acura MDX",
  price: 28800,
  year: 2014,
  used: true,
  color: "Graphite Luster Metallic"
},
{
  make: "BMW",
  model: "5 Series 535i Sedan",
  price: 18995,
  year: 2013,
  used: true,
  color: "Space Gray Metallic"
},
{
  make: "Ford",
  model: "Escape",
  price: 7480,
  year: 2011,
  used: true,
  color: "Sterling Grey Metallic"
},
{
  make: "Subaru",
  model: "Impreza",
  price: 18495,
  year: 2018,
  used: false,
  color: "Crimson Red Pearl"
},
{
  make: "Toyota",
  model: "Yaris",
  price: 15635,
  year: 2018,
  used: false,
  color: "Super White"
```

```

    },
    {
      make: "Honda",
      model: "Civic LX",
      price: 14999,
      year: 2016,
      used: true,
      color: "Crystal Black Pearl"
    },
    {
      make: "Volkswagen",
      model: "Jetta 1.4T S",
      price: 19495,
      year: 2018,
      used: false,
      color: "Silk Blue Metallic"
    }
  ]
);

```

```

MONGODB
CONNECTIONS
cluster0.6hu1ngc.mongodb.net...
test
  appusers
  cars
  inventory
  movies
  records
  scores
  admin
  local

PLAYGROUNDS
No 'mongodb' playground files found in the workspace.
Create New Playground

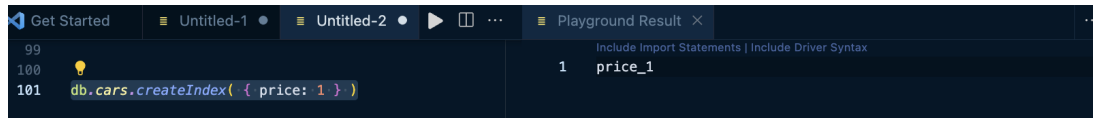
db.cars.insertMany([
  {
    make: "Hyundai",
    model: "Santa Fe",
    price: 8000,
    year: 2003,
    used: true,
    color: "Black"
  },
  {
    make: "BMW",
    model: "ALPINA B6 Gran Coupe",
    price: 124300,
    year: 2017,
    used: false,
    color: "Mediterranean Blue Metallic"
  },
  {
    make: "Subaru",
    model: "Crosstrek 2.0i Premium",
    price: 22595,
    year: 2014,
    used: true,
    color: "Sunshine Orange"
  },
  {
    make: "Ford",
    model: "F-350 XL",
    price: 33705,
    year: 2017,
    used: false,
    color: "Race Red"
  },
  {
    make: "Toyota",
    model: "Acura MDX",
    price: 28800,
    year: 2014,
    used: true,
    color: "Graphite Luster Metallic"
  }
]);

{
  "acknowledged": true,
  "insertedIds": {
    "0": {
      "_id": "63194eaa5a89427097b8f362"
    },
    "1": {
      "_id": "63194eaa5a89427097b8f363"
    },
    "2": {
      "_id": "63194eaa5a89427097b8f364"
    },
    "3": {
      "_id": "63194eaa5a89427097b8f365"
    },
    "4": {
      "_id": "63194eaa5a89427097b8f366"
    },
    "5": {
      "_id": "63194eaa5a89427097b8f367"
    },
    "6": {
      "_id": "63194eaa5a89427097b8f368"
    },
    "7": {
      "_id": "63194eaa5a89427097b8f369"
    },
    "8": {
      "_id": "63194eaa5a89427097b8f36a"
    },
    "9": {
      "_id": "63194eaa5a89427097b8f36b"
    },
    "10": {
      "_id": "63194eaa5a89427097b8f36c"
    }
  }
}

```

## 1.3 Create an index on the price field.

```
db.cars.createIndex( { price: 1 } )
```



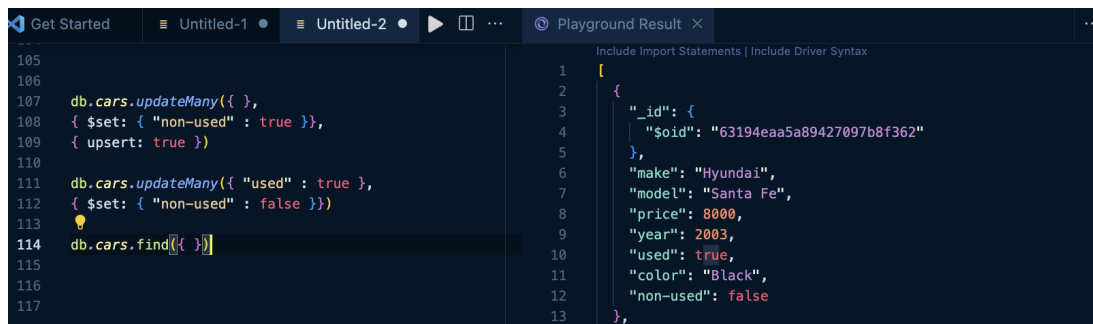
The screenshot shows the MongoDB Playground interface. On the left, the code editor contains the command `db.cars.createIndex({ price: 1 })` at line 101. On the right, the 'Playground Result' tab shows the output: a single index named `price_1`.

## 1.4 Create an index on the non-used field for the cars collection.

```
db.cars.updateMany({ },  
  { $set: { "non-used" : true }},  
  { upsert: true })
```

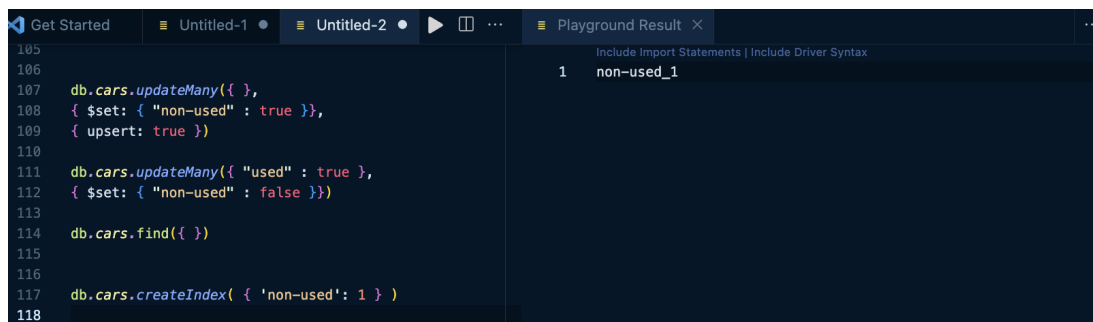
```
db.cars.updateMany({ "used" : true },  
  { $set: { "non-used" : false }})
```

```
db.cars.find({ })
```



The screenshot shows the MongoDB Playground interface. The code editor on the left contains three commands: `db.cars.updateMany({ }, { $set: { "non-used" : true }, { upsert: true })` at line 108, `db.cars.updateMany({ "used" : true }, { $set: { "non-used" : false }})` at line 112, and `db.cars.find({ })` at line 114. The 'Playground Result' tab on the right shows a single JSON document with fields: `_id`, `$oid`, `make`, `model`, `price`, `year`, `used`, `color`, and `non-used`.

```
db.cars.createIndex( { 'non-used': 1 } )
```



The screenshot shows the MongoDB Playground interface. The code editor on the left contains four commands: `db.cars.updateMany({ }, { $set: { "non-used" : true }, { upsert: true })` at line 108, `db.cars.updateMany({ "used" : true }, { $set: { "non-used" : false }})` at line 112, `db.cars.find({ })` at line 114, and `db.cars.createIndex( { 'non-used': 1 } )` at line 117. The 'Playground Result' tab on the right shows the output: a single index named `non-used_1`.

## 1.5 Find and delete all documents with a year before 2012.

- Be sure to do a find with your filtering criteria first to be sure you're about to delete the correct documents.

```
db.cars.find({ year: { $lt : 2012 } })
```



The screenshot shows the MongoDB Playground interface. On the left, the code editor contains the query `db.cars.find({ year: { $lt : 2012 } })` at line 121. On the right, the 'Playground Result' pane shows two documents returned by the query. The first document is a Hyundai Santa Fe from 2003, and the second is a Ford Escape from 2011. Both documents have a 'year' field less than 2012.

```
1 {
2   {
3     "_id": {
4       "$oid": "63194eaa5a89427097b8f362"
5     },
6     "make": "Hyundai",
7     "model": "Santa Fe",
8     "price": 8000,
9     "year": 2003,
10    "used": true,
11    "color": "Black",
12    "non-used": false
13  },
14 }
15 {
16   {
17     "_id": {
18       "$oid": "63194eaa5a89427097b8f368"
19     },
20     "make": "Ford",
21     "model": "Escape",
22     "price": 7480,
23     "year": 2011,
24     "used": true,
25     "color": "Sterling Grey Metallic",
26     "non-used": false
27   }
28 }
```

```
db.cars.deleteMany({ year: { $lt : 2012 } })
```

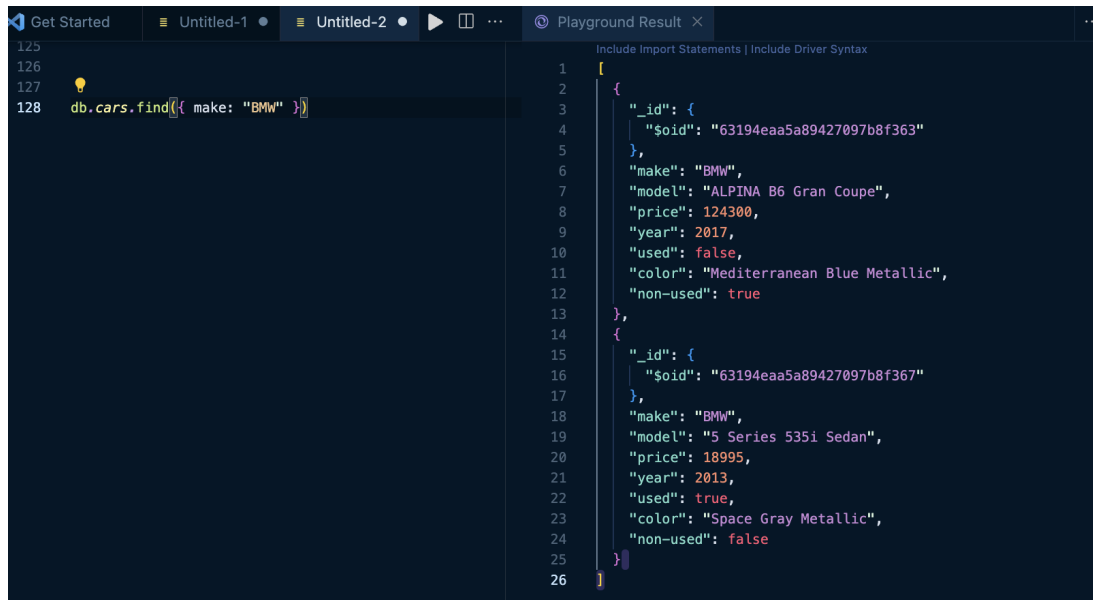


The screenshot shows the MongoDB Playground interface after executing the `deleteMany` query. The code editor on the left shows the query `db.cars.deleteMany({ year: { $lt : 2012 } })` at line 123. The 'Playground Result' pane on the right shows a single document with the fields `"acknowledged": true` and `"deletedCount": 2`, indicating that two documents were successfully deleted.

```
1 {
2   "acknowledged": true,
3   "deletedCount": 2
4 }
```

## 1.6 Delete the first document that is a BMW.

```
db.cars.find({ make: "BMW" })
```



The screenshot shows the MongoDB Playground interface. On the left, the query `db.cars.find({ make: "BMW" })` is entered. On the right, the results are displayed as a JSON array containing two documents. The first document is an Alpina B6 Gran Coupe from 2017, and the second is a BMW 5 Series 535i Sedan from 2013.

```
1 {
2   {
3     "_id": {
4       "$oid": "63194eaa5a89427097b8f363"
5     },
6     "make": "BMW",
7     "model": "ALPINA B6 Gran Coupe",
8     "price": 124300,
9     "year": 2017,
10    "used": false,
11    "color": "Mediterranean Blue Metallic",
12    "non-used": true
13  },
14  {
15    "_id": {
16      "$oid": "63194eaa5a89427097b8f367"
17    },
18    "make": "BMW",
19    "model": "5 Series 535i Sedan",
20    "price": 18995,
21    "year": 2013,
22    "used": true,
23    "color": "Space Gray Metallic",
24    "non-used": false
25  }
26 }
```

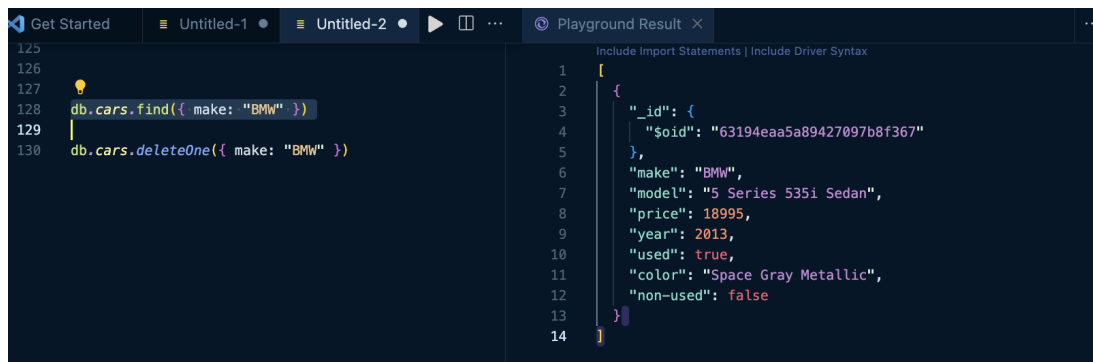
```
db.cars.deleteOne({ make: "BMW" })
```



The screenshot shows the MongoDB Playground interface. On the left, the query `db.cars.deleteOne({ make: "BMW" })` is entered. On the right, the results are displayed as a JSON object indicating that one document was deleted successfully.

```
1 {
2   "acknowledged": true,
3   "deletedCount": 1
4 }
```

```
db.cars.find({ make: "BMW" })
```



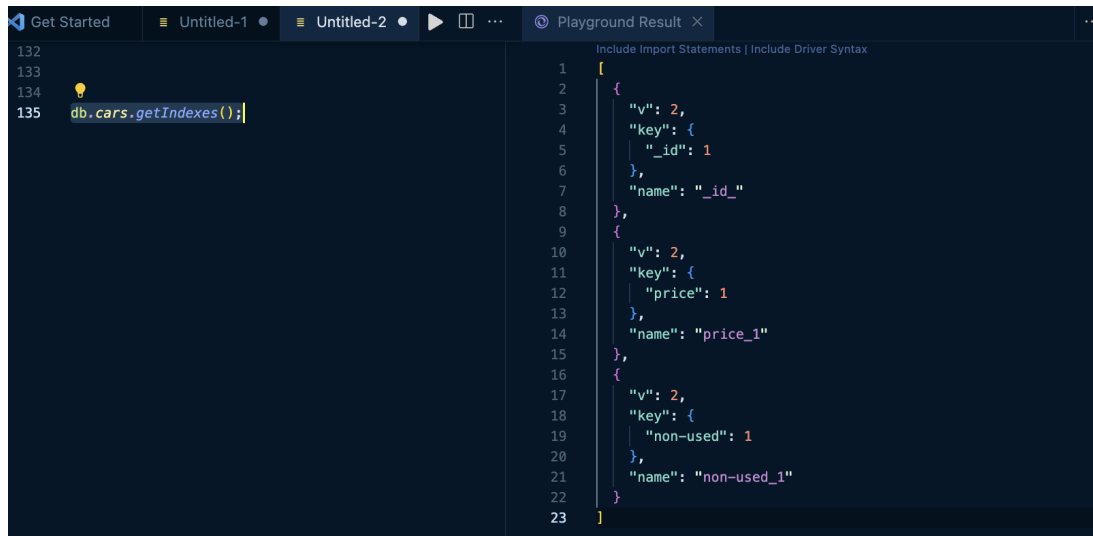
The screenshot shows the MongoDB Playground interface. On the left, the query `db.cars.find({ make: "BMW" })` is entered. On the right, the results are displayed as a JSON array containing only one document, which is the BMW 5 Series 535i Sedan from 2013. The first document (the Alpina) has been successfully deleted.

```
1 {
2   {
3     "_id": {
4       "$oid": "63194eaa5a89427097b8f367"
5     },
6     "make": "BMW",
7     "model": "5 Series 535i Sedan",
8     "price": 18995,
9     "year": 2013,
10    "used": true,
11    "color": "Space Gray Metallic",
12    "non-used": false
13  }
14 }
```



## 1.7 Drop the index created on the non-used cars created above.

```
db.cars.getIndexes();
```



The screenshot shows the MongoDB Playground interface. On the left, the command `db.cars.getIndexes();` is entered in the console. On the right, the result is displayed as a JSON array. The array contains three index objects, each with a `"v": 2` field and a `"key"` field. The first index has a `"key"` of `{ "_id": 1 }` and a `"name"` of `"_id_"`. The second index has a `"key"` of `{ "price": 1 }` and a `"name"` of `"price_1"`. The third index has a `"key"` of `{ "non-used": 1 }` and a `"name"` of `"non-used_1"`.

```
1 {
2   {
3     "v": 2,
4     "key": {
5       "_id": 1
6     },
7     "name": "_id_"
8   },
9   {
10    "v": 2,
11    "key": {
12      "price": 1
13    },
14    "name": "price_1"
15  },
16  {
17    "v": 2,
18    "key": {
19      "non-used": 1
20    },
21    "name": "non-used_1"
22  }
23 }
```

```
db.cars.dropIndex( { 'non-used': 1 } )
```

```
db.cars.getIndexes();
```



The screenshot shows the MongoDB Playground interface. On the left, the command `db.cars.getIndexes();` is entered in the console. On the right, the result is displayed as a JSON array. The array contains two index objects, each with a `"v": 2` field and a `"key"` field. The first index has a `"key"` of `{ "_id": 1 }` and a `"name"` of `"_id_"`. The second index has a `"key"` of `{ "price": 1 }` and a `"name"` of `"price_1"`. The third index, which was `{ "non-used": 1 }` and `"name": "non-used_1"`, has been removed.

```
1 {
2   {
3     "v": 2,
4     "key": {
5       "_id": 1
6     },
7     "name": "_id_"
8   },
9   {
10    "v": 2,
11    "key": {
12      "price": 1
13    },
14    "name": "price_1"
15  }
16 }
```


## Part 2

Below is a real-life scenario. Please read this scenario and run the appropriate queries needed.

You are currently working for a car dealership. They sell both used and new cars. The company would like to easily and efficiently search through their cars using the “make” of the car. Recently, they made the searching efficient using the price of the car, but that is no longer needed since they will now be using the make of the vehicles. Please reflect that in the database. Also, the company has decided to no longer sell Volkswagens and has already sold the last Volkswagen on the lot so they would like you to reflect that in the database as well.

### 2.1 Delete the price index

```
db.cars.getIndexes();
```

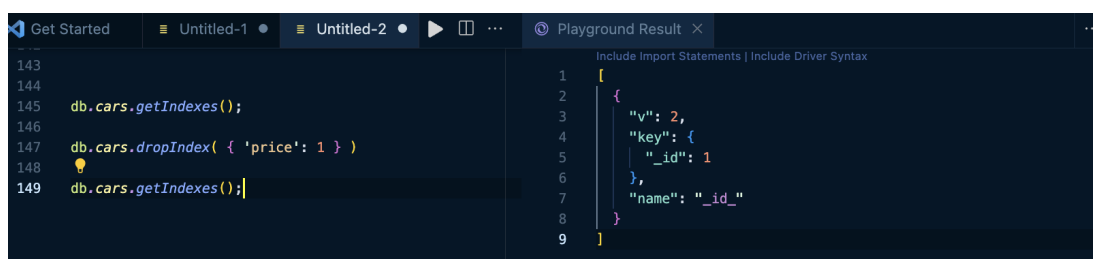


The screenshot shows a MongoDB Playground window with two tabs: 'Untitled-2' and 'Playground Result'. The 'Untitled-2' tab contains the command `db.cars.getIndexes();` at line 145. The 'Playground Result' tab shows the output of the command, which is a JSON array of index information. The output is as follows:

```
1  [
2    {
3      "v": 2,
4      "key": {
5        "_id": 1
6      },
7      "name": "_id_"
8    },
9    {
10     "v": 2,
11     "key": {
12       "price": 1
13     },
14     "name": "price_1"
15   }
16 ]
```

```
db.cars.dropIndex( { 'non-used': 1 } )
```

```
db.cars.getIndexes();
```



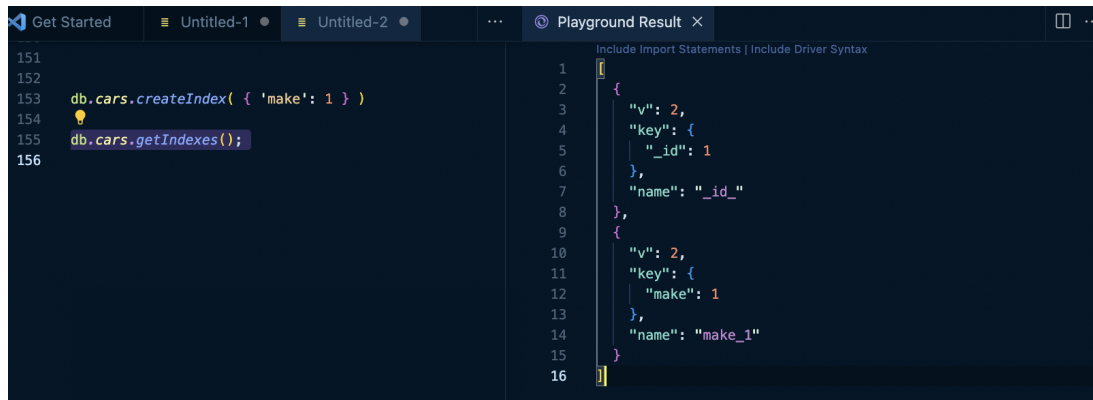
The screenshot shows a MongoDB Playground window with two tabs: 'Untitled-2' and 'Playground Result'. The 'Untitled-2' tab contains the following commands: `db.cars.getIndexes();` at line 145, `db.cars.dropIndex( { 'price': 1 } )` at line 147, and `db.cars.getIndexes();` at line 149. The 'Playground Result' tab shows the output of the first command, which is a JSON array of index information. The output is as follows:

```
1  [
2    {
3      "v": 2,
4      "key": {
5        "_id": 1
6      },
7      "name": "_id_"
8    }
9  ]
```

## 2.2 Create a make index

```
db.cars.createIndex( { 'make': 1 } )
```

```
db.cars.getIndexes();
```



The screenshot shows a MongoDB Playground interface with two panels. The left panel contains the following code:

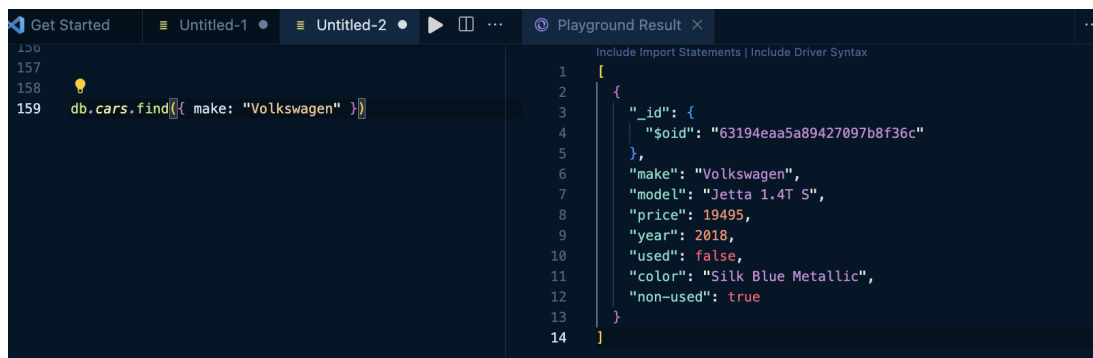
```
151  
152  
153 db.cars.createIndex( { 'make': 1 } )  
154  
155 db.cars.getIndexes();  
156
```

The right panel, titled "Playground Result", shows the output of the `getIndexes()` command as a JSON array:

```
1  
2 [  
3   {  
4     "v": 2,  
5     "key": {  
6       "_id": 1  
7     },  
8     "name": "_id_"  
9   },  
10  {  
11    "v": 2,  
12    "key": {  
13      "make": 1  
14    },  
15    "name": "make_1"  
16  }  
17 ]
```

## 2.3 Delete documents with make: "Volkswagen"

```
db.cars.find({ make: "Volkswagen" })
```



The screenshot shows a MongoDB Playground interface with two panels. The left panel contains the following code:

```
156  
157  
158  
159 db.cars.find({ make: "Volkswagen" })
```

The right panel, titled "Playground Result", shows the output of the `find` command as a JSON array with one document:

```
1  
2 [  
3   {  
4     "_id": {  
5       "$oid": "63194eaa5a89427097b8f36c"  
6     },  
7     "make": "Volkswagen",  
8     "model": "Jetta 1.4T S",  
9     "price": 19495,  
10    "year": 2018,  
11    "used": false,  
12    "color": "Silk Blue Metallic",  
13    "non-used": true  
14  }  
15 ]
```

```
db.cars.deleteMany({ make: "Volkswagen" })
```



The screenshot shows a MongoDB Playground interface with two panels. The left panel contains the following code:

```
156  
157  
158  
159 db.cars.find({ make: "Volkswagen" })  
160  
161 db.cars.deleteMany({ make: "Volkswagen" })
```

The right panel, titled "Playground Result", shows the output of the `deleteMany` command as a JSON object:

```
1  
2 {  
3   "acknowledged": true,  
4   "deletedCount": 1  
5 }
```