

PRACTICAL NO – 4

Aim: Indexing using MongoDB

1. Mongo DB indexing
 - a. Create index in Mongo DB
 - b. Finding the indexes in a collection
 - c. Drop indexes in a collection
 - d. Drop all the indexes

```
use students
db.createCollection("studentgrades")
db.studentgrades.insertMany(
[
{name: "Barry", subject: "Maths", score: 92},
{name: "Kent", subject: "Physics", score: 87},
{name: "Harry", subject: "Maths", score: 99, notes: "Exceptional Performance"},
{name: "Alex", subject: "Literature", score: 78},
{name: "Tom", subject: "History", score: 65, notes: "Adequate"}
]
)db
db.studentgrades.find({}, {_id:0})
db.studentgrades.find().pretty()

db.studentgrades.createIndex( {name: 1}, {name: "student name index"} )
```

Code:

```
use students;

db.createCollection("studentsgrades")

db.studentgrades.insertMany(

[

{name:"Barray",subject:"Maths",score:92},

{name:"Kent",subject:"Physics",score:87},

{name:"Harry",subject:"Maths",score:99,notes:"Exceptional
Performance"},

{name:"Alex",subject:"Literature",score:78},
```

```
{name:"Tom",subject:"History",score:65,notes:"Adequate"}}]);
```

Output:

```
test> use students;
switched to db students
students> db.createCollection("studentgrades")
MongoServerError[NamespaceExists]: Collection students.studentgrades already exists.
students> db.createCollection("studentsgrades")
{ ok: 1 }
students> db.studentgrades.insertMany(
... [
... {name:"Barray",subject:"Maths",score:92},
... {name:"Kent",subject:"Physics",score:87},
... {name:"Harry",subject:"Maths",score:99,notes:"Exceptional Performance"},
... {name:"Alex",subject:"Literature",score:78},
... {name:"Tom",subject:"History",score:65,notes:"Adequate"}}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('678a29fd0b1aea3e66bb51cb'),
    '1': ObjectId('678a29fd0b1aea3e66bb51cc'),
    '2': ObjectId('678a29fd0b1aea3e66bb51cd'),
    '3': ObjectId('678a29fd0b1aea3e66bb51ce'),
    '4': ObjectId('678a29fd0b1aea3e66bb51cf')
  }
}
```

Code:

```
db.studentgrades.find({},{_id:0});
```

Output:

```
students> db.studentgrades.find({},{_id:0});
[
  { name: 'Barry', subject: 'Maths', score: 92 },
  { name: 'kent', subject: 'physics', score: 98 },
  {
    name: 'Harry',
    subject: 'Maths',
    score: 99,
    notes: 'Exceptional Performance'
  },
  { name: 'Alex', subject: 'Literature', score: 78 },
  { name: 'Tom', subject: 'History', score: 78 },
  { name: 'Tom', subject: 'History', score: 65, notes: 'Adequate' },
  { name: 'Barray', subject: 'Maths', score: 92 },
  { name: 'Kent', subject: 'Physics', score: 87 },
  {
    name: 'Harry',
    subject: 'Maths',
    score: 99,
    notes: 'Exceptional Performance'
  },
  { name: 'Alex', subject: 'Literature', score: 78 },
]
```

Code:

```
db.studentgrades.find().pretty();
```

Output:

```
students> db.studentgrades.find().pretty();
[
  {
    _id: ObjectId('65f52d56270308fddd06a1ab'),
    name: 'Barry',
    subject: 'Maths',
    score: 92
  },
  {
    _id: ObjectId('65f52d56270308fddd06a1ac'),
    name: 'kent',
    subject: 'physics',
    score: 98
  },
  {
    _id: ObjectId('65f52d56270308fddd06a1ad'),
    name: 'Harry',
    subject: 'Maths',
    score: 99,
    notes: 'Exceptional Performance'
  },
  {
    _id: ObjectId('65f52d56270308fddd06a1ae'),
    name: 'Alex',
    subject: 'Literature',
    score: 78
  },
  {
    _id: ObjectId('65f52d56270308fddd06a1af'),
    name: 'Tom',
    subject: 'History',
    score: 78
  },
  {
    _id: ObjectId('65f52d56270308fddd06a1b0'),
    name: 'Tom',
    subject: 'History',
    score: 65,
    notes: 'Adequate'
  },
]
```

Code:

```
db.studentgrades.createIndex({name: 1},{name:"student name index"});
```

Output:

```
students> db.studentgrades.createIndex({name: 1},{name:"student name index"});
student name index
```

Finding indexes You can find all the available indexes in a MongoDB collection by using the `getIndexes` method. This will return all the indexes in a specific collection. `db.getIndexes()` Let's view all the indexes in the `studentgrades` collection using the following command:

```
db.studentgrades.getIndexes()
```

Code:

```
db.studentgrades.getIndexes();
```

Output:

```
students>
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'student name index' }
]
```

Dropping indexes To delete an index from a collection, use the `dropIndex` method while specifying the index name to be dropped. `db.dropIndex()` Let's remove the user-created index with the index name `student name index`, as shown below. `db.studentgrades.dropIndex("student name index")`

Code:

```
db.studentgrades.dropIndex("student name index");
```

Output:

```
students> db.studentgrades.dropIndex("student name index");
{ nIndexesWas: 2, ok: 1 }
```

You can also use the index field value for removing an index without a defined name: `db.studentgrades.dropIndex({name:1})`

Code:

```
db.studentgrades.dropIndex({name:1});
```

Output:

```
students> db.studentgrades.dropIndex({name:1});
```

The dropIndexes command can also drop all the indexes excluding the default _id index. db.studentgrades.dropIndexes()

Code:

```
db.studentgrades.dropIndexes();
```

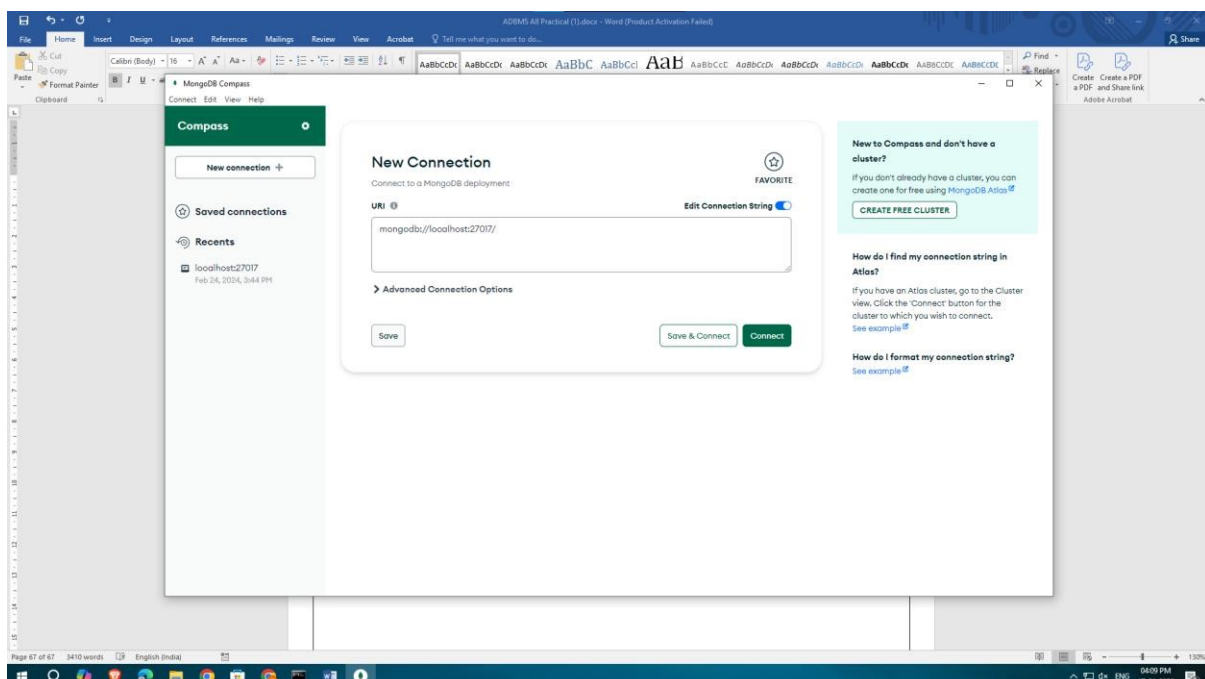
Output:

```
students> db.studentgrades.dropIndexes();
{
  nIndexesWas: 1,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```

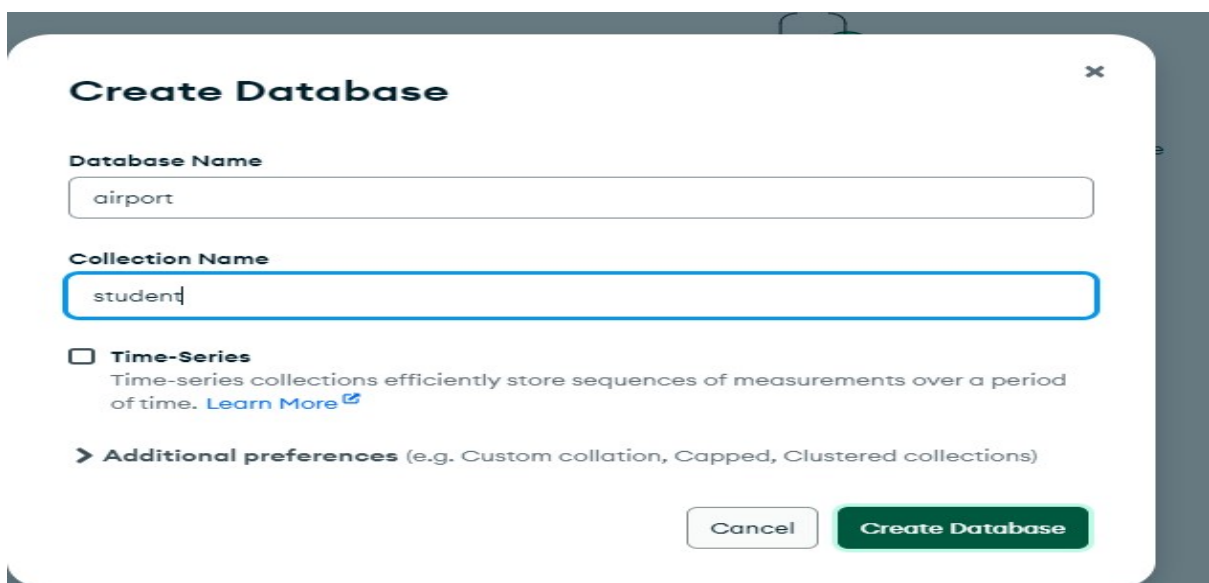
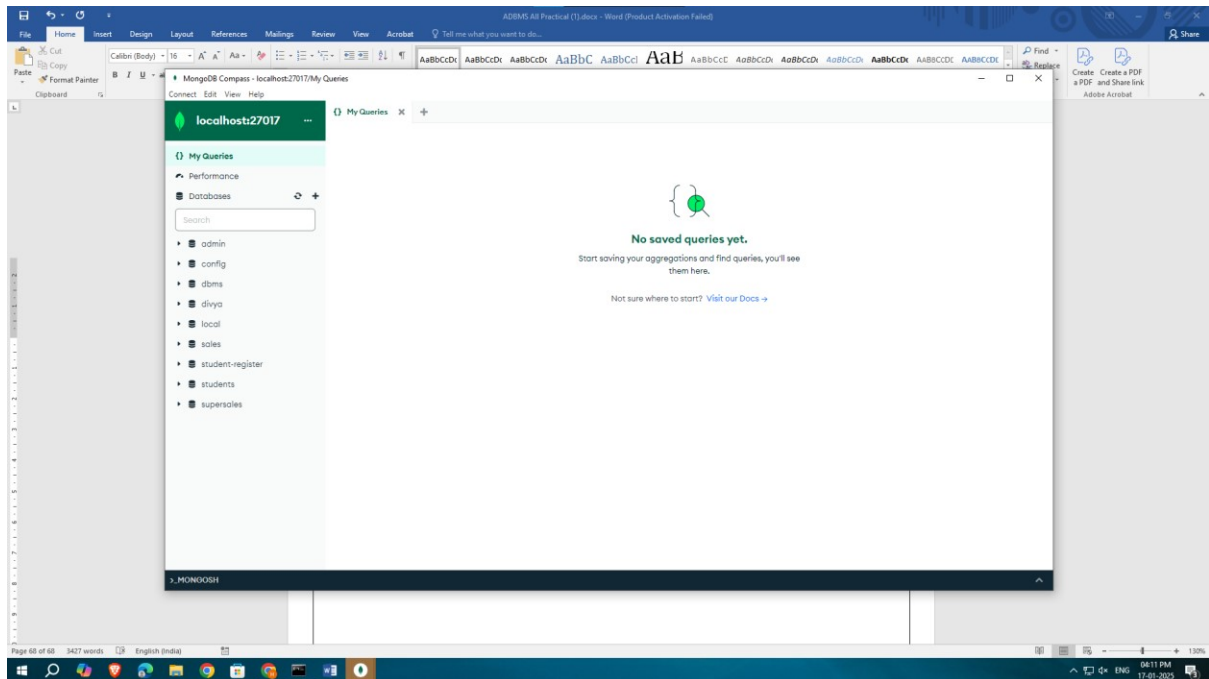
2. Create all the types of indexes (discussed in class) which will help in finding certain words in a document by using AIRPORT (dataset).

Step 1: Go to mongodb compass

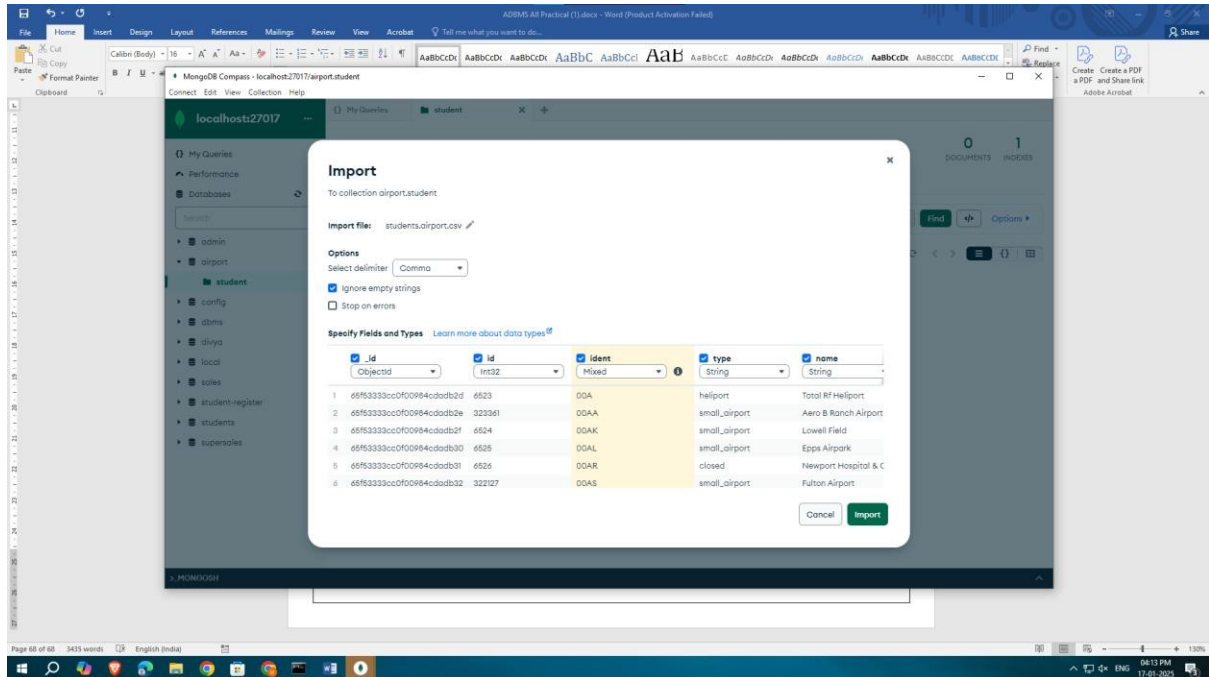
Step 2: Connect to the localhost



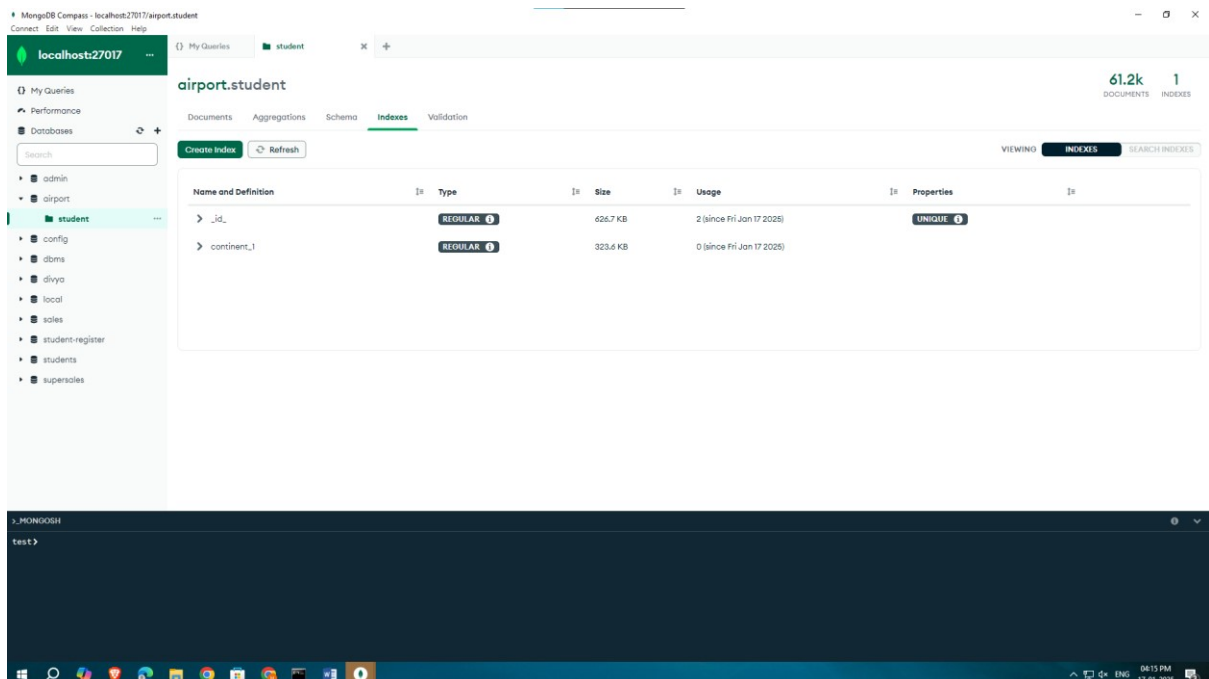
Step 3: Create a databases and upload airport file



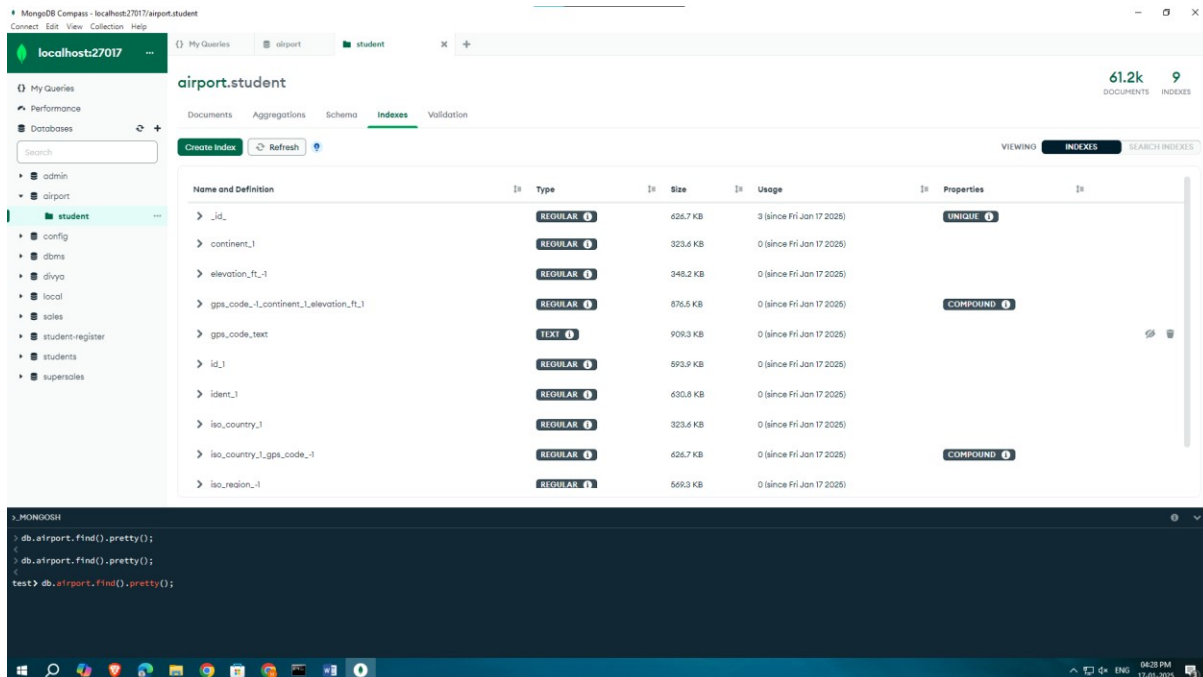
Step 4: Import data and click on import



Step 5: Create Indexes



Step 6: Using different indexes

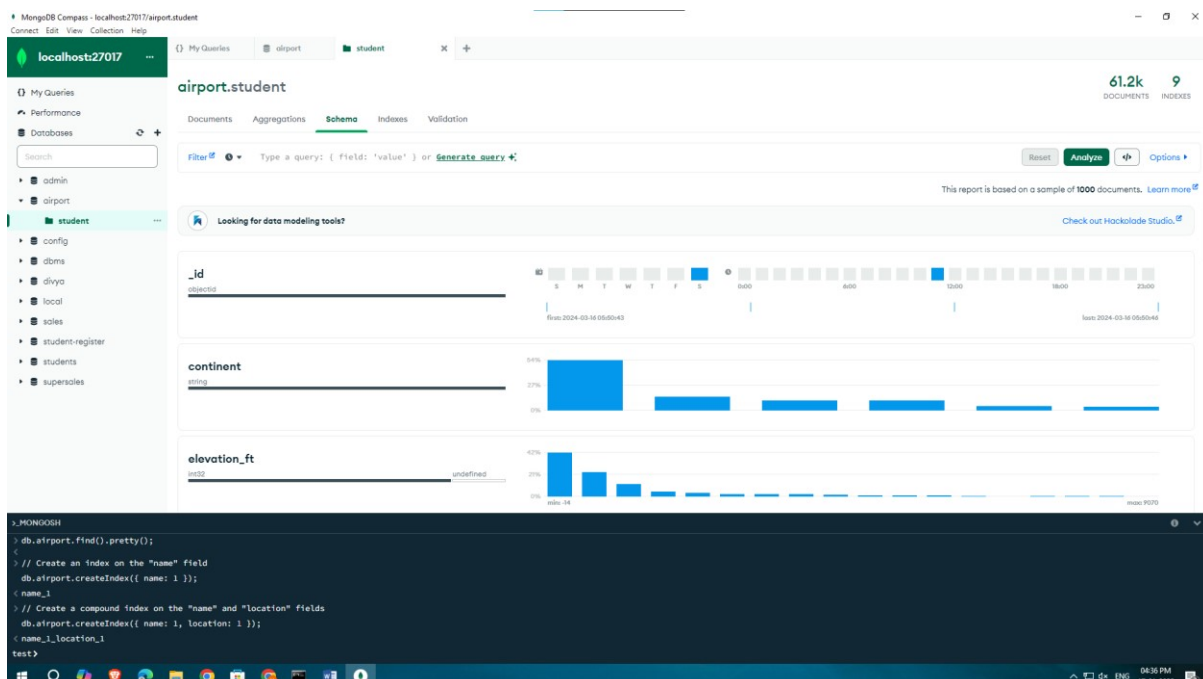


The screenshot shows the MongoDB Compass interface for the 'airport.student' collection. The 'Indexes' tab is selected, displaying a table of indexes. The table has columns for Name and Definition, Type, Size, Usage, and Properties.

Name and Definition	Type	Size	Usage	Properties
> _id	REGULAR	626.7 KB	3 (since Fri Jan 17 2025)	UNIQUE
> continent_1	REGULAR	323.6 KB	0 (since Fri Jan 17 2025)	
> elevation_ft_1	REGULAR	348.2 KB	0 (since Fri Jan 17 2025)	
> gps_code_1,continent_1,elevation_ft_1	REGULAR	676.5 KB	0 (since Fri Jan 17 2025)	COMPOUND
> gps_code_text	TEXT	909.3 KB	0 (since Fri Jan 17 2025)	
> id_1	REGULAR	693.9 KB	0 (since Fri Jan 17 2025)	
> ident_1	REGULAR	630.8 KB	0 (since Fri Jan 17 2025)	
> iso_country_1	REGULAR	323.6 KB	0 (since Fri Jan 17 2025)	
> iso_country_1,gps_code_1	REGULAR	626.7 KB	0 (since Fri Jan 17 2025)	COMPOUND
> iso_region_1	REGULAR	669.3 KB	0 (since Fri Jan 17 2025)	

Below the table, a terminal window shows the following commands:

```
> db.airport.find().pretty();
> db.airport.find().pretty();
test> db.airport.find().pretty();
```



The screenshot shows the MongoDB Compass interface for the 'airport.student' collection, with the 'Schema' tab selected. It displays histograms for the '_id', 'continent', and 'elevation_ft' fields.

_id histogram: The x-axis represents the value of the '_id' field, ranging from 0 to 2500. The y-axis represents the frequency, ranging from 0 to 10. The histogram shows a distribution of values, with a peak around 1000.

continent histogram: The x-axis represents the value of the 'continent' field, ranging from 0 to 2500. The y-axis represents the frequency, ranging from 0 to 10. The histogram shows a distribution of values, with a peak around 1000.

elevation_ft histogram: The x-axis represents the value of the 'elevation_ft' field, ranging from 0 to 2500. The y-axis represents the frequency, ranging from 0 to 10. The histogram shows a distribution of values, with a peak around 1000.

Below the histograms, a terminal window shows the following commands:

```
> db.airport.find().pretty();
// Create an index on the "name" field
db.airport.createIndex({ name: 1 });
// Create a compound index on the "name" and "location" fields
db.airport.createIndex({ name: 1, location: 1 });
test>
```