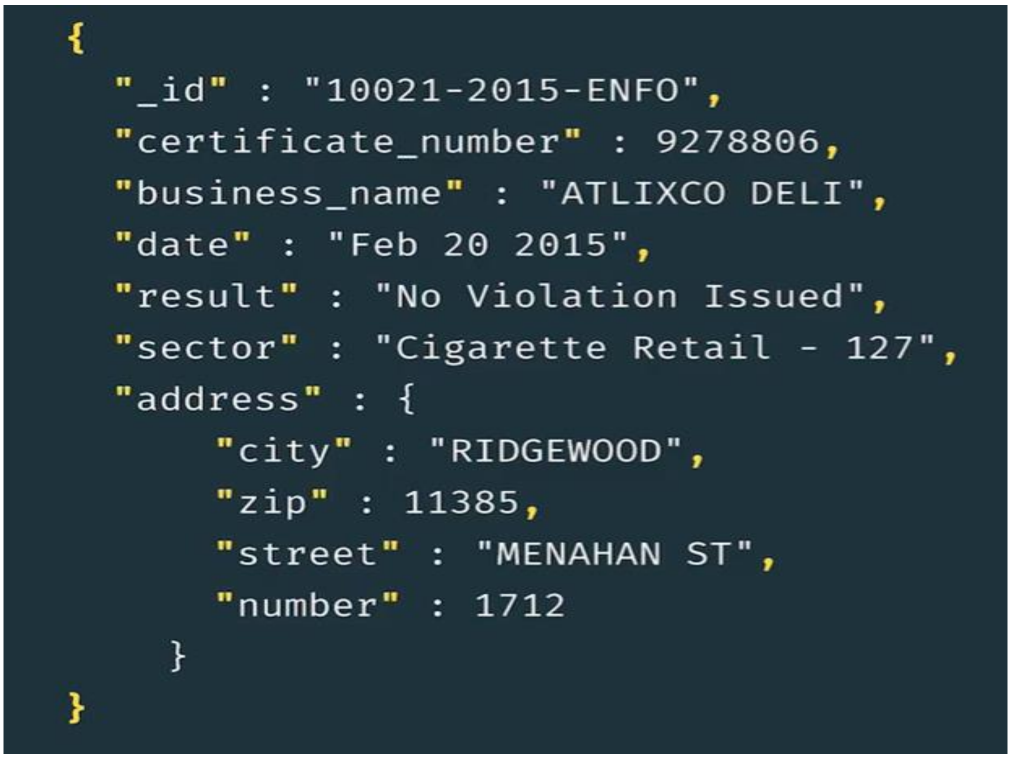
1. How does MongoDB Store Data

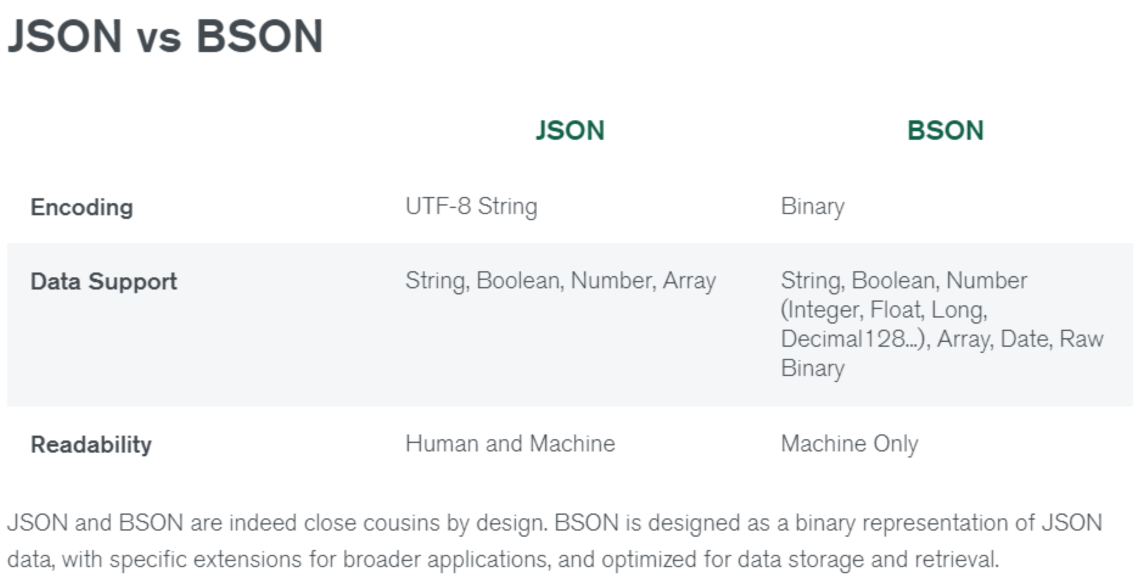
- Uses documents
- Represented in JSON(JavaScript Standard Object Notation) → User friendly, Readable, Familiar

  JSON Format: Start and end with curly braces {}, Separate each key and value with colon : , Separate each key:value pair with a comma , "keys" must be surrounded by quotation marks "" , In MongoDB "keys" are called fields



```
{
  "_id" : "10021-2015-ENFO",
  "certificate_number" : 9278806,
  "business_name" : "ATLIXCO DELI",
  "date" : "Feb 20 2015",
  "result" : "No Violation Issued",
  "sector" : "Cigarette Retail - 127",
  "address" : {
      "city" : "RIDGEWOOD",
      "zip" : 11385,
      "street" : "MENAHAN ST",
      "number" : 1712
    }
}
```

- Disadvantages: Text-based format parsing text is difficult, space-consuming, supports limited datatypes
- We also have another format called BSON (Binary JSON): binary representation to store data in JSON format. Bridges the gap between binary representation and JSON format. Optimized for speed, space, flexibility, High performance, general-purpose focus.
- MongoDB stores data in BSON internally and over the network.
- JSON can be natively stored and retrieved in MongoDB



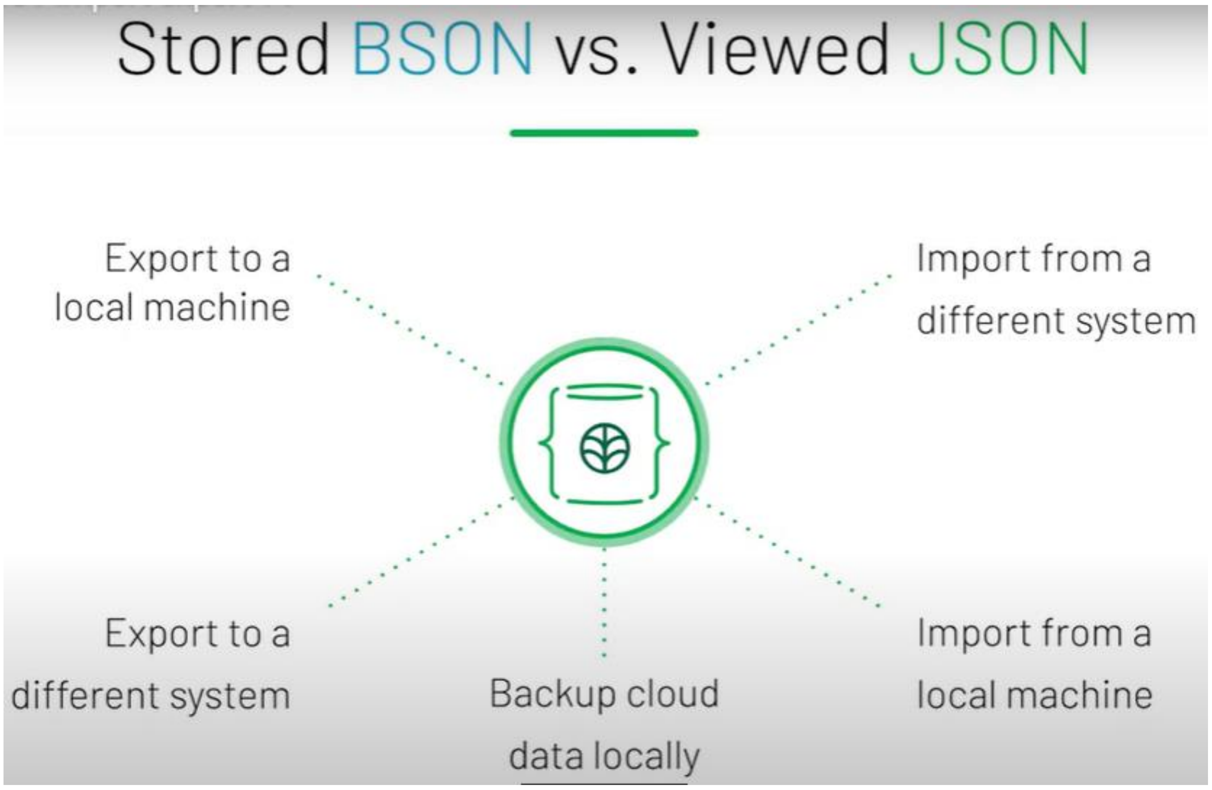
JSON vs BSON

	JSON	BSON
Encoding	UTF-8 String	Binary
Data Support	String, Boolean, Number, Array	String, Boolean, Number (Integer, Float, Long, Decimal128...), Array, Date, Raw Binary
Readability	Human and Machine	Machine Only

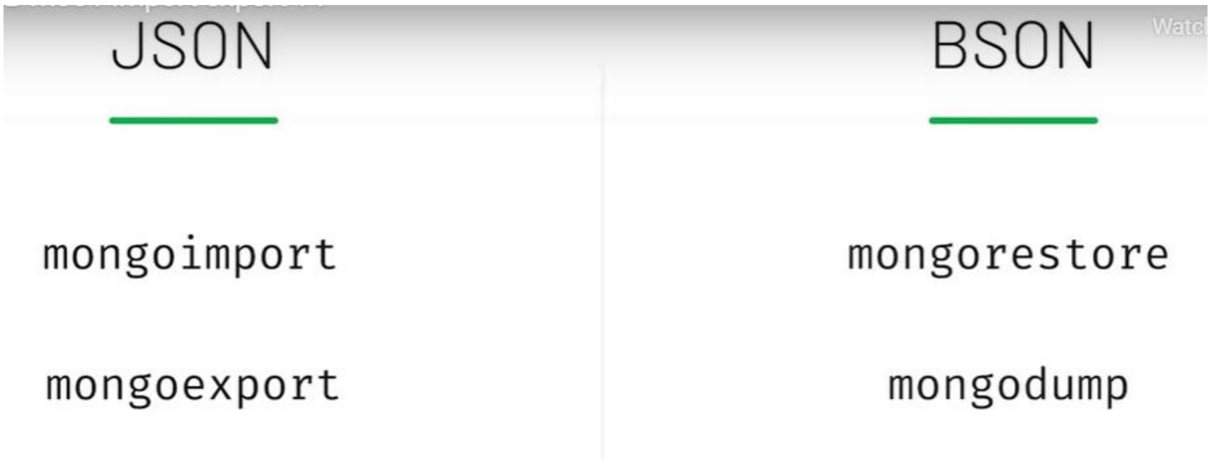
JSON and BSON are indeed close cousins by design. BSON is designed as a binary representation of JSON data, with specific extensions for broader applications, and optimized for data storage and retrieval.

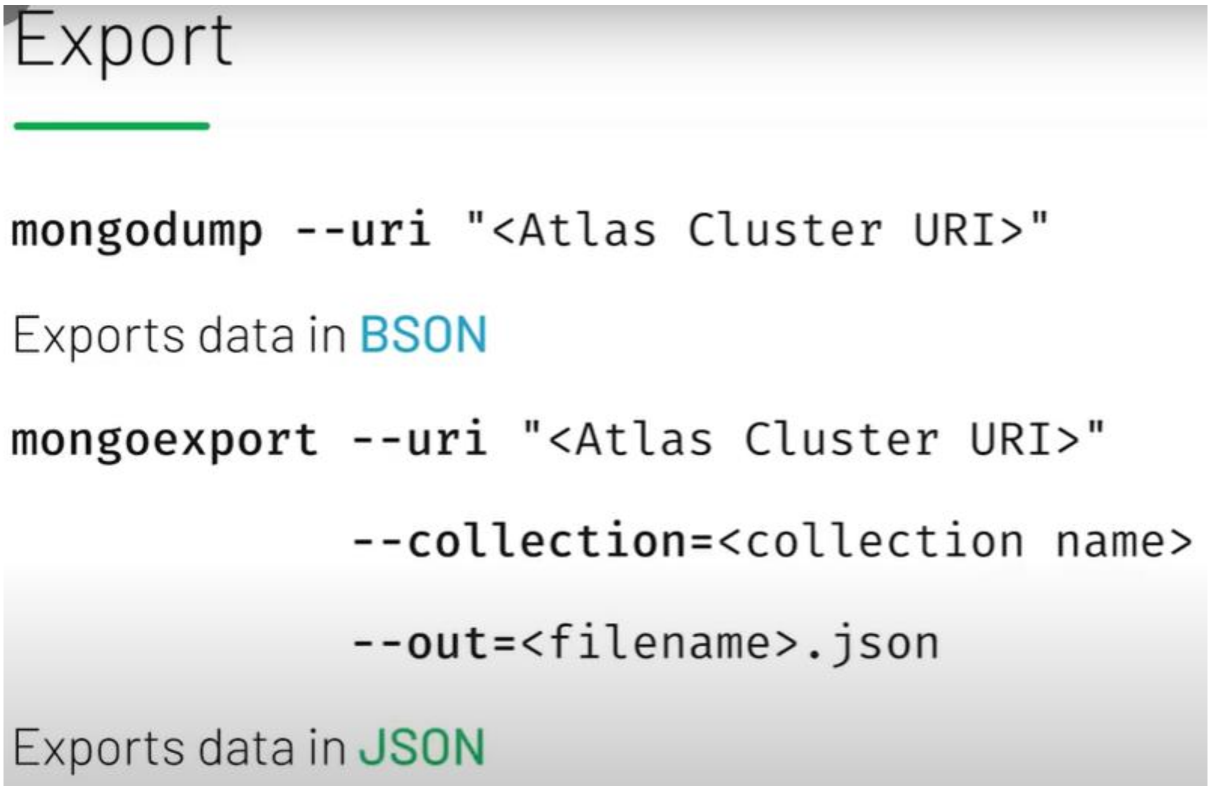
-

2. Importing and Exporting Data:


Stored BSON vs. Viewed JSON
Export to a local machine
Import from a different system
Export to a different system
Backup cloud data locally
Import from a local machine

-

- 
JSON	BSON
mongoimport	mongorestore
mongoexport	mongodump

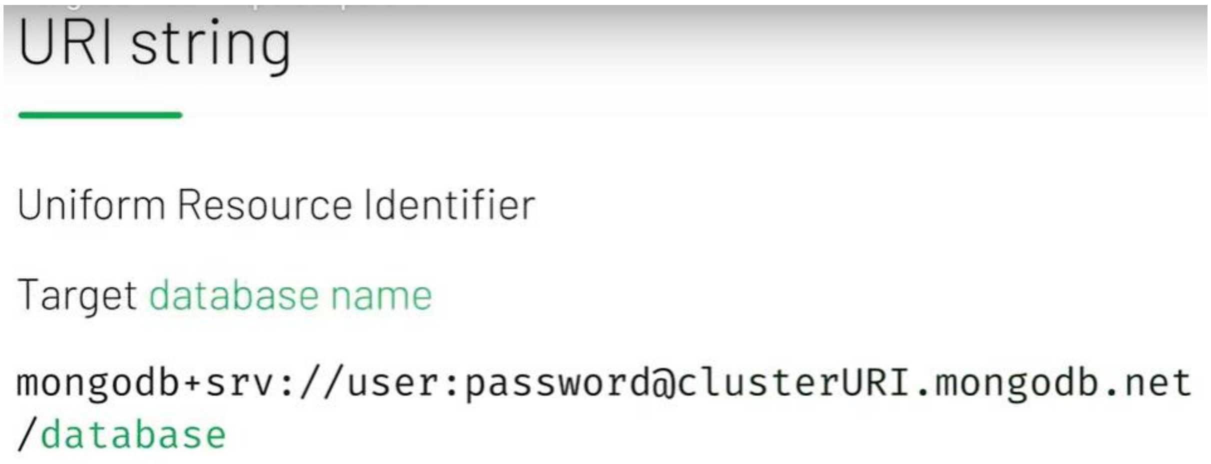
- 
  Export

  ```
  mongodump --uri "<Atlas Cluster URI>"
  ```

  Exports data in BSON

  ```
  mongoexport --uri "<Atlas Cluster URI>"
                --collection=<collection name>
                --out=<filename>.json
  ```

  Exports data in JSON

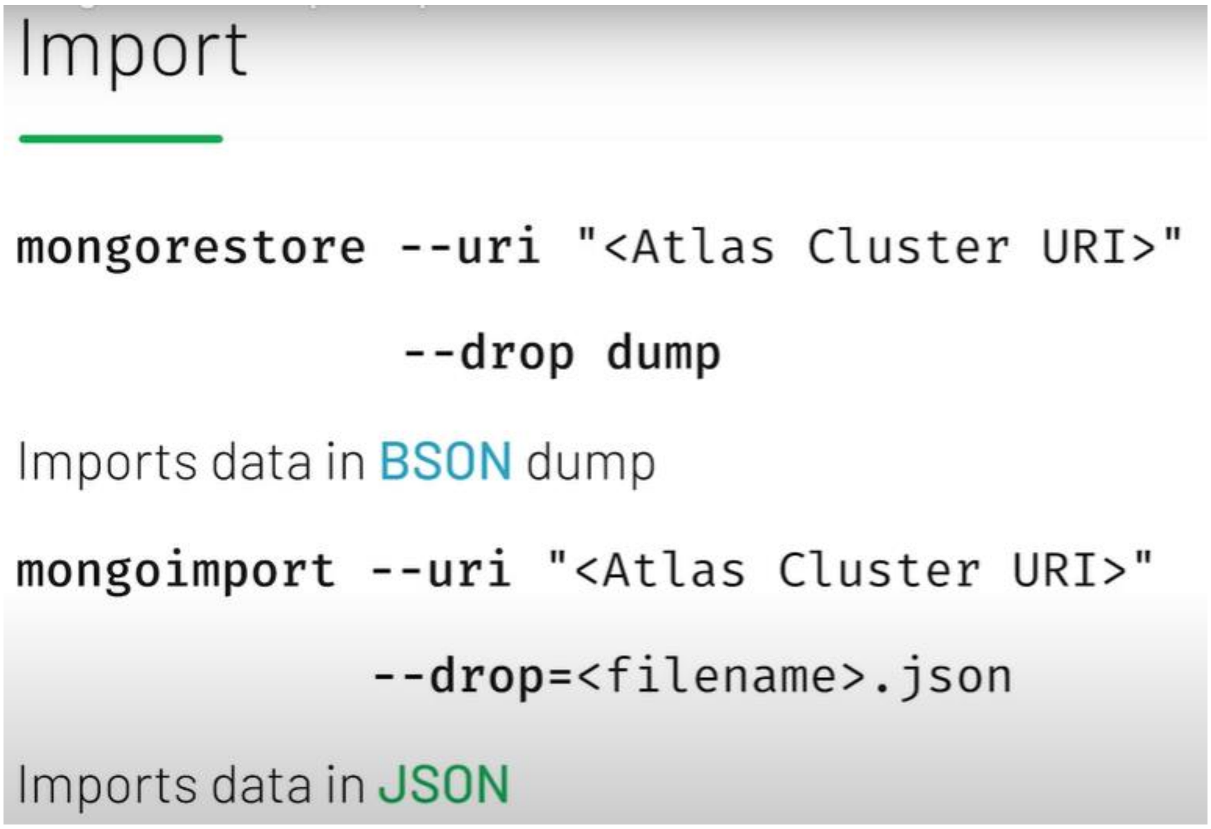
- 
  URI string

  Uniform Resource Identifier

  Target database name

  ```
  mongodb+srv://user:password@clusterURI.mongodb.net
  /database
  ```

- Mongoexport –uri = “mongodb+srv://m001-mongodb-basics@sandbox.pass.mongodb/sample_supplies(filename)”

- Collection = sales –out=sales.json
- less sales.json → shows the data
- 

Import

```
mongorestore --uri "<Atlas Cluster URI>"
             --drop dump
```

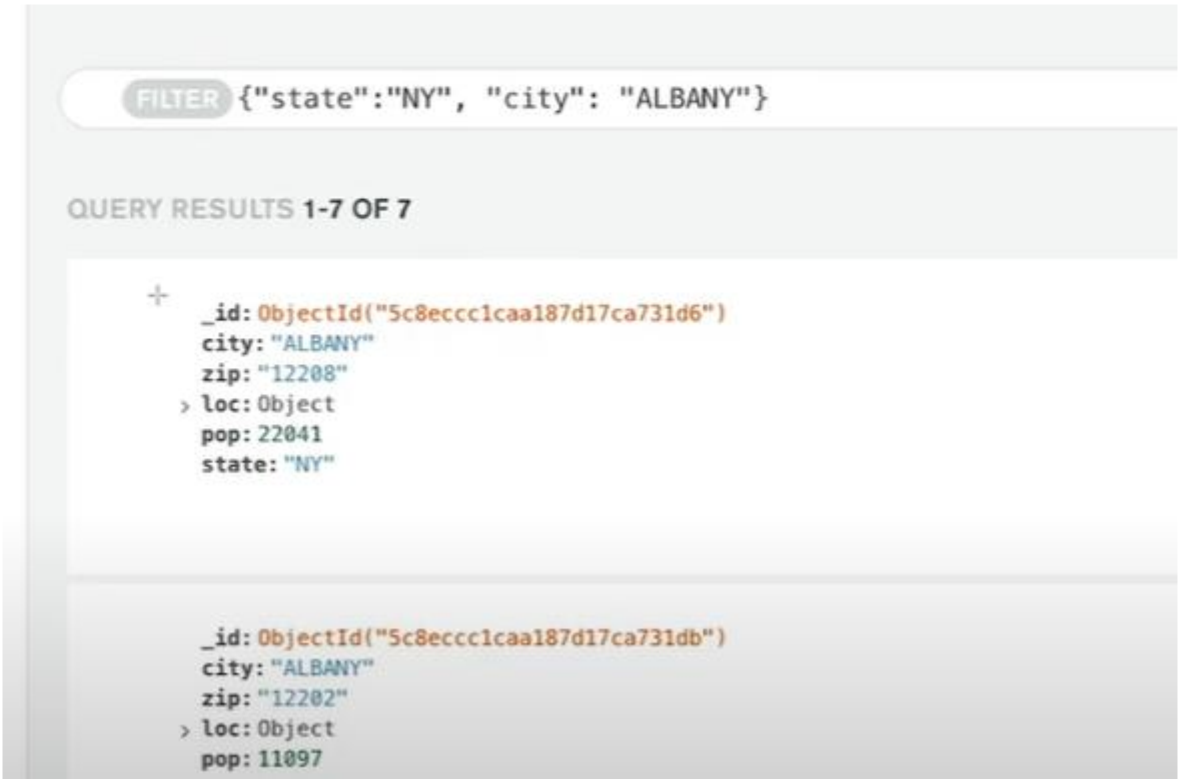
Imports data in BSON dump

```
mongoimport --uri "<Atlas Cluster URI>"
            --drop=<filename>.json
```

Imports data in JSON

- Mongo restore –uri"Atlas uri" –drop dump(filename) → this removes the stored data

3. Querying

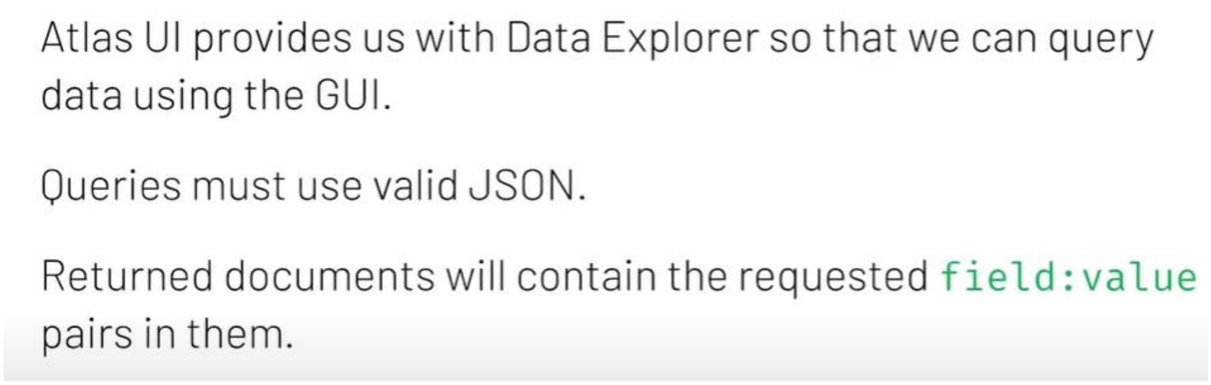
- 

```
FILTER {"state":"NY", "city": "ALBANY"}

QUERY RESULTS 1-7 OF 7

_id: ObjectId("5c8eccc1caa187d17ca731d6")
city: "ALBANY"
zip: "12208"
> loc: Object
pop: 22041
state: "NY"

_id: ObjectId("5c8eccc1caa187d17ca731db")
city: "ALBANY"
zip: "12202"
> loc: Object
pop: 11097
```

- 

  Atlas UI provides us with Data Explorer so that we can query data using the GUI.

  Queries must use valid JSON.

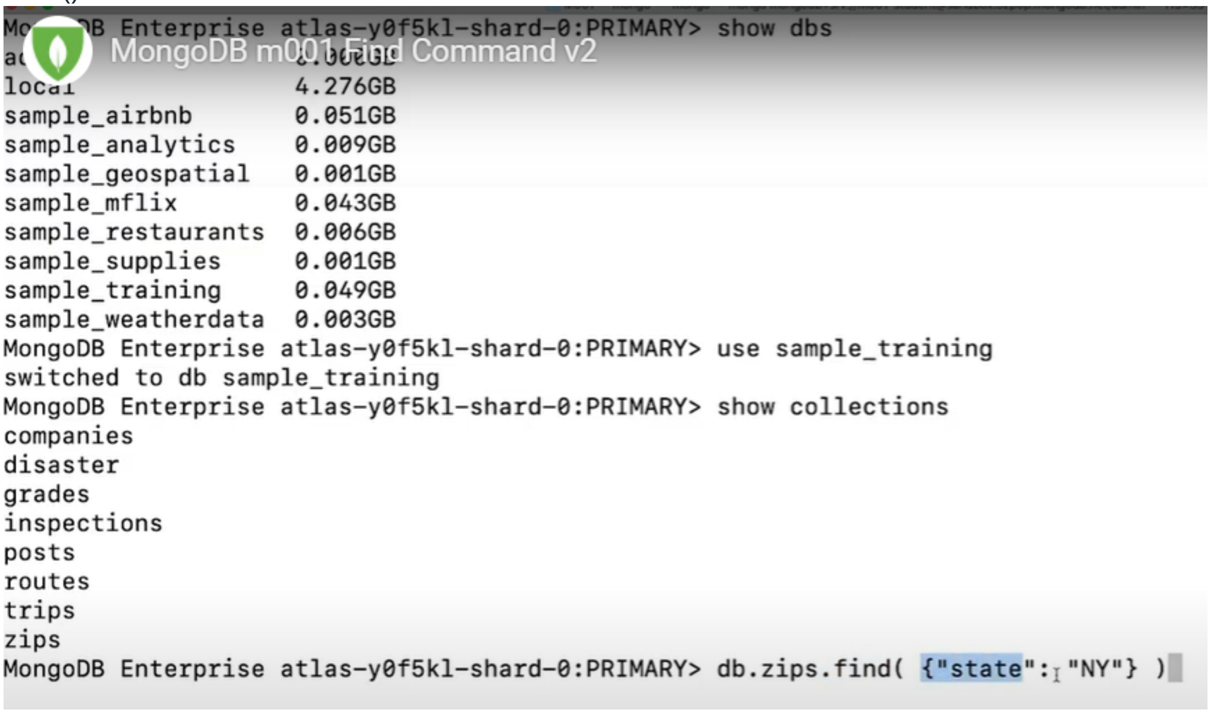
  Returned documents will contain the requested `field:value` pairs in them.

  Namespace - The concatenation of the database name and collection name is called a namespace.

  We looked at the `sample_training.zips` collection and issued the following queries:

  - `{"state": "NY"}`
  - `{"state": "NY", "city": "ALBANY"`

4. find() command:

- 

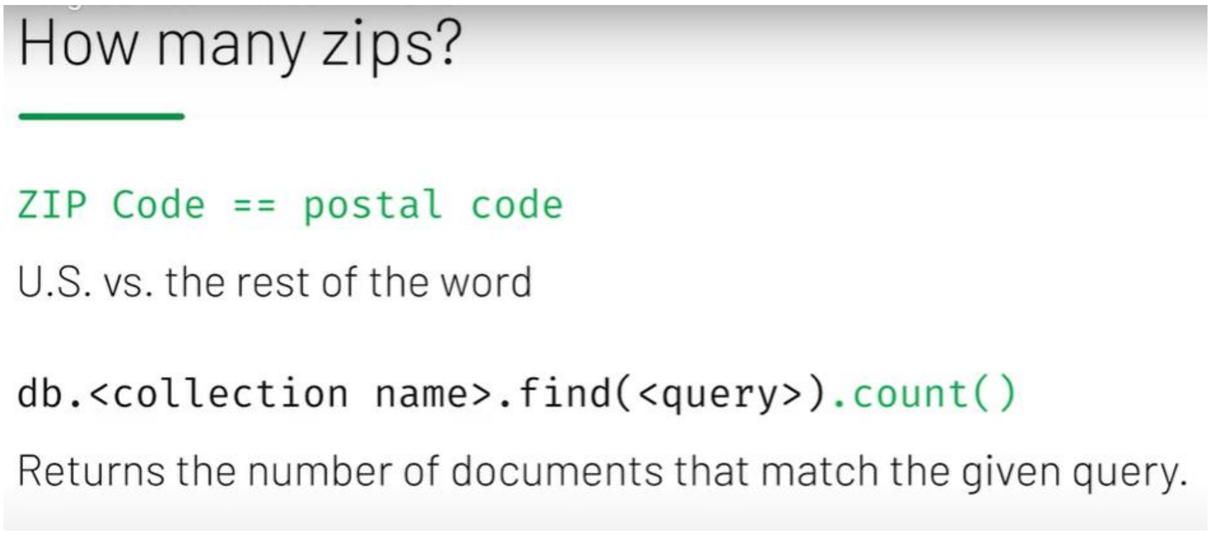
```
Mo   DB Enterprise atlas-y0f5kl-shard-0:PRIMARY> show dbs
a[illegible]
local               4.276GB
sample_airbnb       0.051GB
sample_analytics    0.009GB
sample_geospatial   0.001GB
sample_mflix        0.043GB
sample_restaurants  0.006GB
sample_supplies     0.001GB
sample_training     0.049GB
sample_weatherdata  0.003GB
MongoDB Enterprise atlas-y0f5kl-shard-0:PRIMARY> use sample_training
switched to db sample_training
MongoDB Enterprise atlas-y0f5kl-shard-0:PRIMARY> show collections
companies
disaster
grades
inspections
posts
routes
trips
zips
MongoDB Enterprise atlas-y0f5kl-shard-0:PRIMARY> db.zips.find( {"state": "NY"} )
```

- We can query the data by connecting shell
- Connect to the Atlas cluster:

- `mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"`
- `show dbs`
- `use sample_training`

- `show collections`
- `db.zips.find({"state": "NY"})`
- `it` iterates through the cursor.
- `db.zips.find({"state": "NY"}).count()`
- `db.zips.find({"state": "NY", "city": "ALBANY"})`
- `db.zips.find({"state": "NY", "city": "ALBANY"}).pretty()`

Cursor is a pointer to a result set of query, A pointer is a direct address of the memory location.



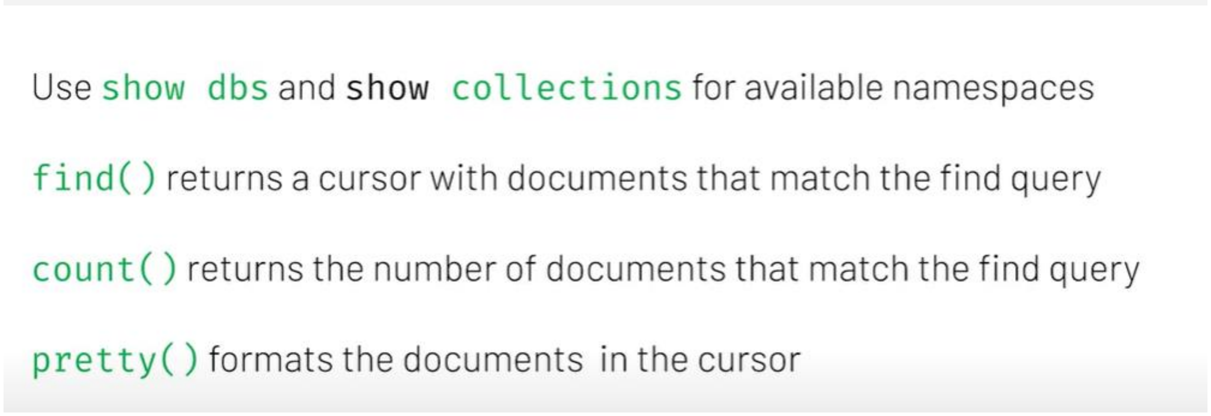
How many zips?

`ZIP Code == postal code`

U.S. vs. the rest of the word

`db.<collection name>.find(<query>).count()`

Returns the number of documents that match the given query.



Use `show dbs` and `show collections` for available namespaces

`find()` returns a cursor with documents that match the find query

`count()` returns the number of documents that match the find query

`pretty()` formats the documents in the cursor