MONGO DB

1. MongoDB has the same concept of a database with which you are likely already familiar (or a schema for you

Oracle folks). Within a MongoDB instance you can have zero or more databases, each acting as high-level

containers for everything else.

2. A database can have zero or more collections. A collection shares enough in common with a traditional table

that you can safely think of the two as the same thing.

3. Collections are made up of zero or more documents. Again, a document can safely be thought of as a row.

4. A document is made up of one or more fields, which you can probably guess are a lot like columns.

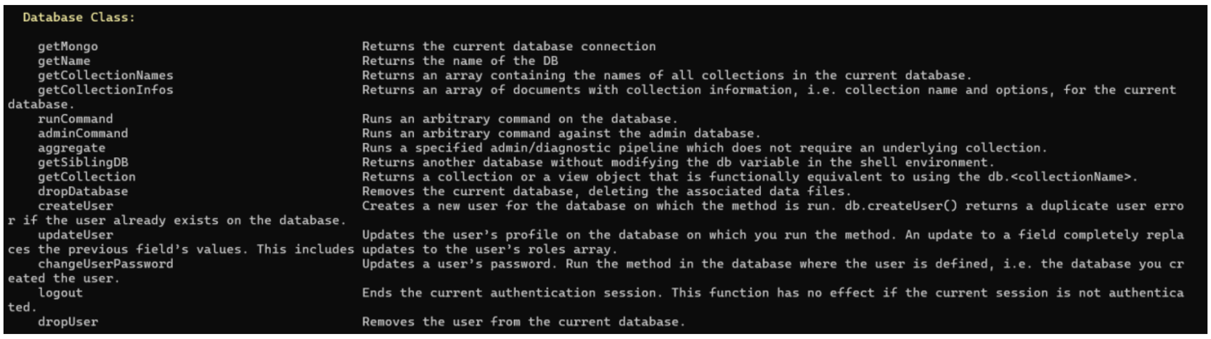
5. Indexes in MongoDB function mostly like their RDBMS counterparts.

6. Cursors are different than the other five concepts but they are important enough, and often overlooked, that

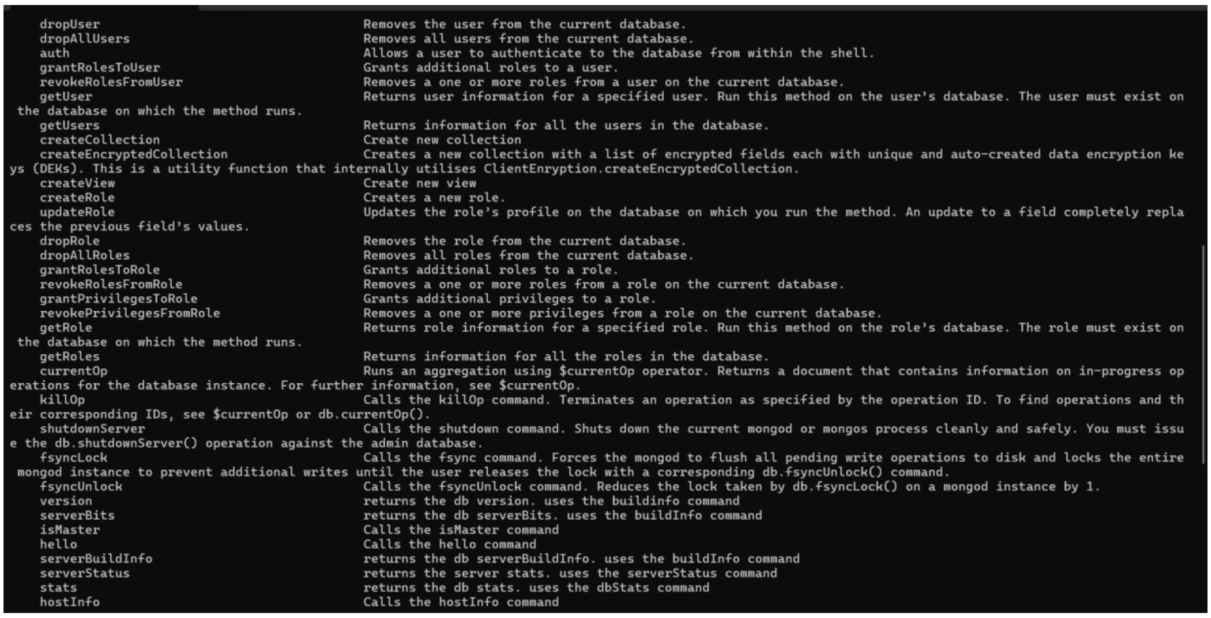
I think they are worthy of their own discussion. The important thing to understand about cursors is that when

you ask MongoDB for data, it returns a pointer to the result set called a cursor, which we can do things to, such

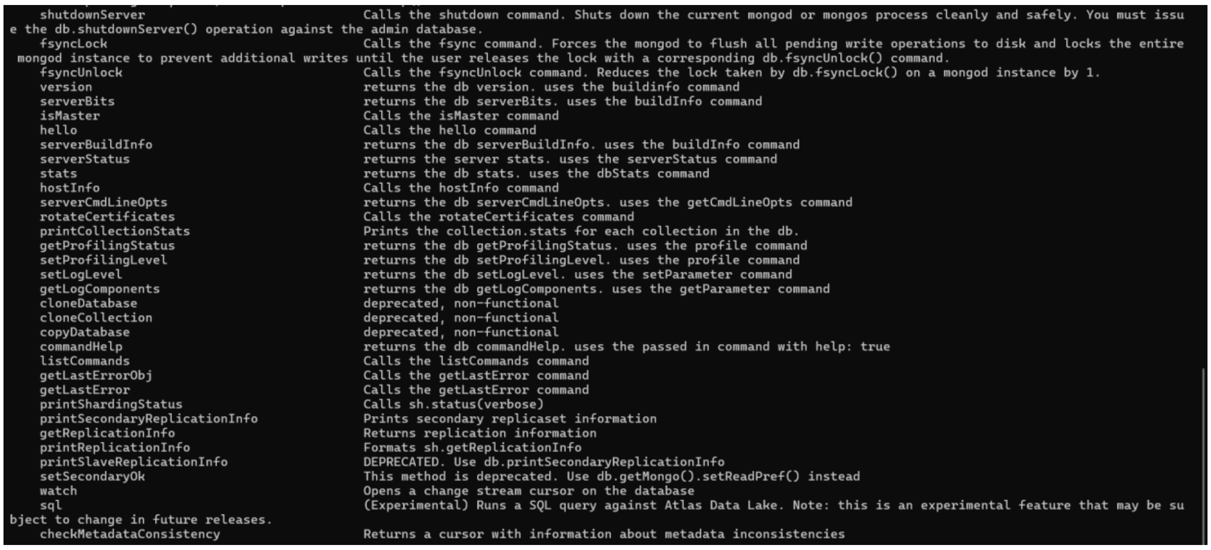
as counting or skipping ahead, before actually pulling down data.



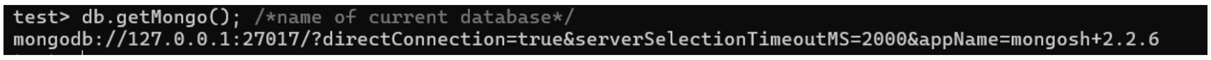
```
  Database Class:

    getMongo                                  Returns the current database connection
    getName                                   Returns the name of the DB
    getCollectionNames                        Returns an array containing the names of all collections in the current database.
    getCollectionInfos                        Returns an array of documents with collection information, i.e. collection name and options, for the current
database.
    runCommand                                Runs an arbitrary command on the database.
    adminCommand                              Runs an arbitrary command against the admin database.
    aggregate                                 Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
    getSiblingDB                              Returns another database without modifying the db variable in the shell environment.
    getCollection                             Returns a collection or a view object that is functionally equivalent to using the db.<collectionName>.
    dropDatabase                              Removes the current database, deleting the associated data files.
    createUser                                Creates a new user for the database on which the method is run. db.createUser() returns a duplicate user erro
r if the user already exists on the database.
    updateUser                                Updates the user's profile on the database on which you run the method. An update to a field completely repla
ces the previous field's values. This includes updates to the user's roles array.
    changeUserPassword                        Updates a user's password. Run the method in the database where the user is defined, i.e. the database you cr
eated the user.
    logout                                    Ends the current authentication session. This function has no effect if the current session is not authentica
ted.
    dropUser                                  Removes the user from the current database.
```



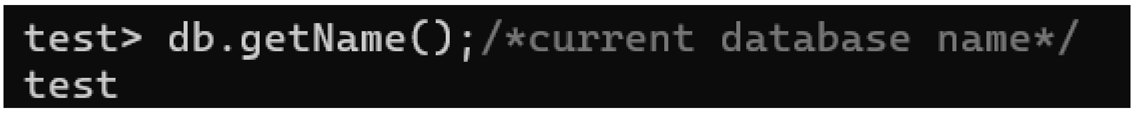
```
    dropUser                                   Removes the user from the current database.
    dropAllUsers                               Removes all users from the current database.
    auth                                       Allows a user to authenticate to the database from within the shell.
    grantRolesToUser                           Grants additional roles to a user.
    revokeRolesFromUser                        Removes a one or more roles from a user on the current database.
    getUser                                    Returns user information for a specified user. Run this method on the user's database. The user must exist on
 the database on which the method runs.
    getUsers                                   Returns information for all the users in the database.
    createCollection                           Create new collection
    createEncryptedCollection                  Creates a new collection with a list of encrypted fields each with unique and auto-created data encryption ke
ys (DEKs). This is a utility function that internally utilises ClientEnryption.createEncryptedCollection.
    createView                                 Create new view
    createRole                                 Creates a new role.
    updateRole                                 Updates the role's profile on the database on which you run the method. An update to a field completely repla
ces the previous field's values.
    dropRole                                   Removes the role from the current database.
    dropAllRoles                               Removes all roles from the current database.
    grantRolesToRole                           Grants additional roles to a role.
    revokeRolesFromRole                        Removes a one or more roles from a role on the current database.
    grantPrivilegesToRole                      Grants additional privileges to a role.
    revokePrivilegesFromRole                   Removes a one or more privileges from a role on the current database.
    getRole                                    Returns role information for a specified role. Run this method on the role's database. The role must exist on
 the database on which the method runs.
    getRoles                                   Returns information for all the roles in the database.
    currentOp                                  Runs an aggregation using $currentOp operator. Returns a document that contains information on in-progress op
erations for the database instance. For further information, see $currentOp.
    killOp                                     Calls the killOp command. Terminates an operation as specified by the operation ID. To find operations and th
eir corresponding IDs, see $currentOp or db.currentOp().
    shutdownServer                             Calls the shutdown command. Shuts down the current mongod or mongos process cleanly and safely. You must issu
e the db.shutdownServer() operation against the admin database.
    fsyncLock                                  Calls the fsync command. Forces the mongod to flush all pending write operations to disk and locks the entire
 mongod instance to prevent additional writes until the user releases the lock with a corresponding db.fsyncUnlock() command.
    fsyncUnlock                                Calls the fsyncUnlock command. Reduces the lock taken by db.fsyncLock() on a mongod instance by 1.
    version                                    returns the db version. uses the buildinfo command
    serverBits                                 returns the db serverBits. uses the buildInfo command
    isMaster                                   Calls the isMaster command
    hello                                      Calls the hello command
    serverBuildInfo                            returns the db serverBuildInfo. uses the buildInfo command
    serverStatus                               returns the server stats. uses the serverStatus command
    stats                                      returns the db stats. uses the dbStats command
    hostInfo                                   Calls the hostInfo command
```



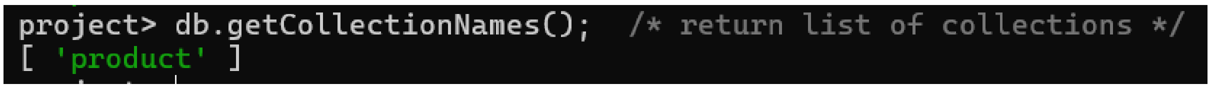
```
    shutdownServer                             Calls the shutdown command. Shuts down the current mongod or mongos process cleanly and safely. You must issu
e the db.shutdownServer() operation against the admin database.
    fsyncLock                                  Calls the fsync command. Forces the mongod to flush all pending write operations to disk and locks the entire
 mongod instance to prevent additional writes until the user releases the lock with a corresponding db.fsyncUnlock() command.
    fsyncUnlock                                Calls the fsyncUnlock command. Reduces the lock taken by db.fsyncLock() on a mongod instance by 1.
    version                                    returns the db version. uses the buildinfo command
    serverBits                                 returns the db serverBits. uses the buildInfo command
    isMaster                                   Calls the isMaster command
    hello                                      Calls the hello command
    serverBuildInfo                            returns the db serverBuildInfo. uses the buildInfo command
    serverStatus                               returns the server stats. uses the serverStatus command
    stats                                      returns the db stats. uses the dbStats command
    hostInfo                                   Calls the hostInfo command
    serverCmdLineOpts                          returns the db serverCmdLineOpts. uses the getCmdLineOpts command
    rotateCertificates                         Calls the rotateCertificates command
    printCollectionStats                       Prints the collection.stats for each collection in the db.
    getProfilingStatus                         returns the db getProfilingStatus. uses the profile command
    setProfilingLevel                          returns the db setProfilingLevel. uses the profile command
    setLogLevel                                returns the db setLogLevel. uses the setParameter command
    getLogComponents                           returns the db getLogComponents. uses the getParameter command
    cloneDatabase                              deprecated, non-functional
    cloneCollection                            deprecated, non-functional
    copyDatabase                               deprecated, non-functional
    commandHelp                                returns the db commandHelp. uses the passed in command with help: true
    listCommands                               Calls the listCommands command
    getLastErrorObj                            Calls the getLastError command
    getLastError                               Calls the getLastError command
    printShardingStatus                        Calls sh.status(verbose)
    printSecondaryReplicationInfo              Prints secondary replicaset information
    getReplicationInfo                         Returns replication information
    printReplicationInfo                       Formats sh.getReplicationInfo
    printSlaveReplicationInfo                  DEPRECATED. Use db.printSecondaryReplicationInfo
    setSecondaryOk                             This method is deprecated. Use db.getMongo().setReadPref() instead
    watch                                      Opens a change stream cursor on the database
    sql                                        (Experimental) Runs a SQL query against Atlas Data Lake. Note: this is an experimental feature that may be su
bject to change in future releases.
    checkMetadataConsistency                   Returns a cursor with information about metadata inconsistencies
```



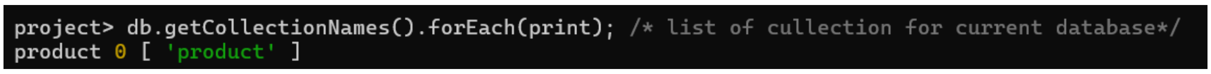
```
test> db.getMongo(); /*name of current database*/
mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.6
```



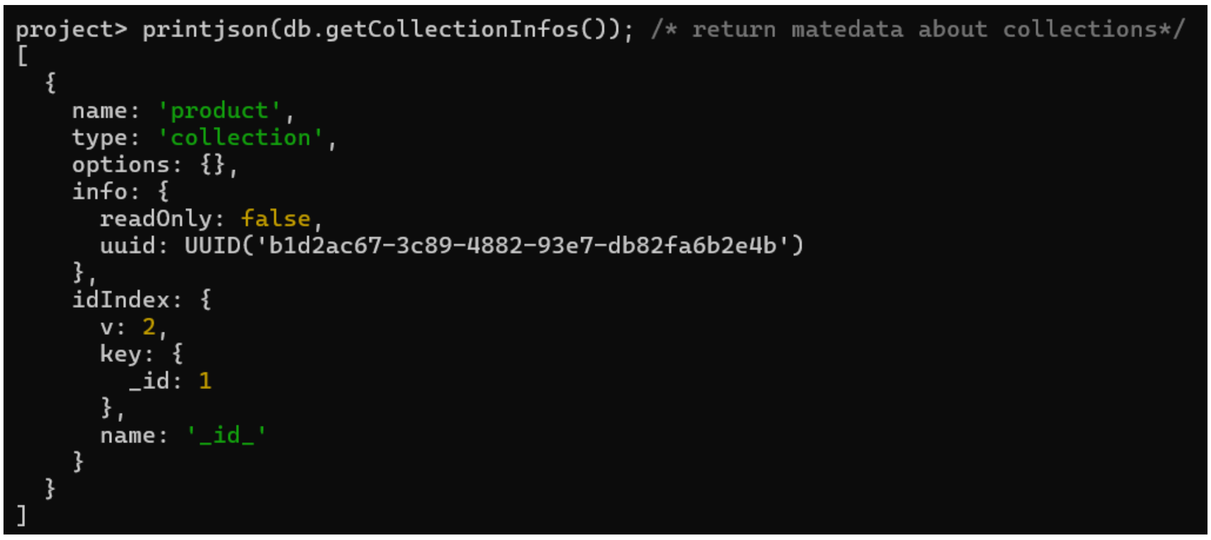
```
test> db.getName();/*current database name*/
test
```



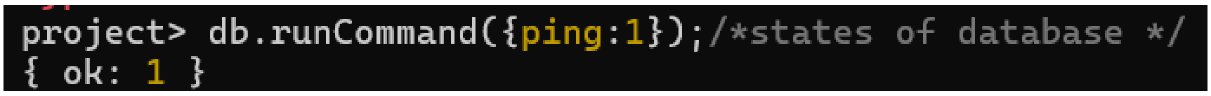
```
project> db.getCollectionNames();  /* return list of collections */
[ 'product' ]
```



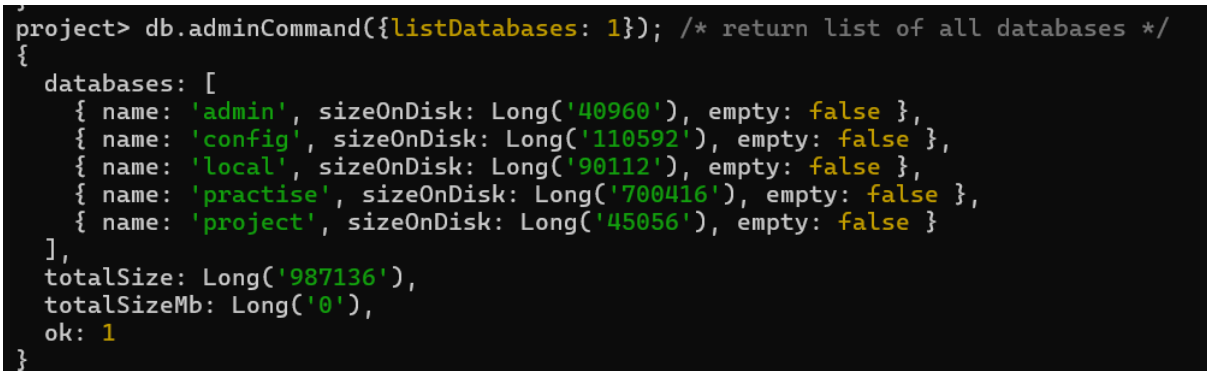
```
project> db.getCollectionNames().forEach(print); /* list of cullection for current database*/
product 0 [ 'product' ]
```



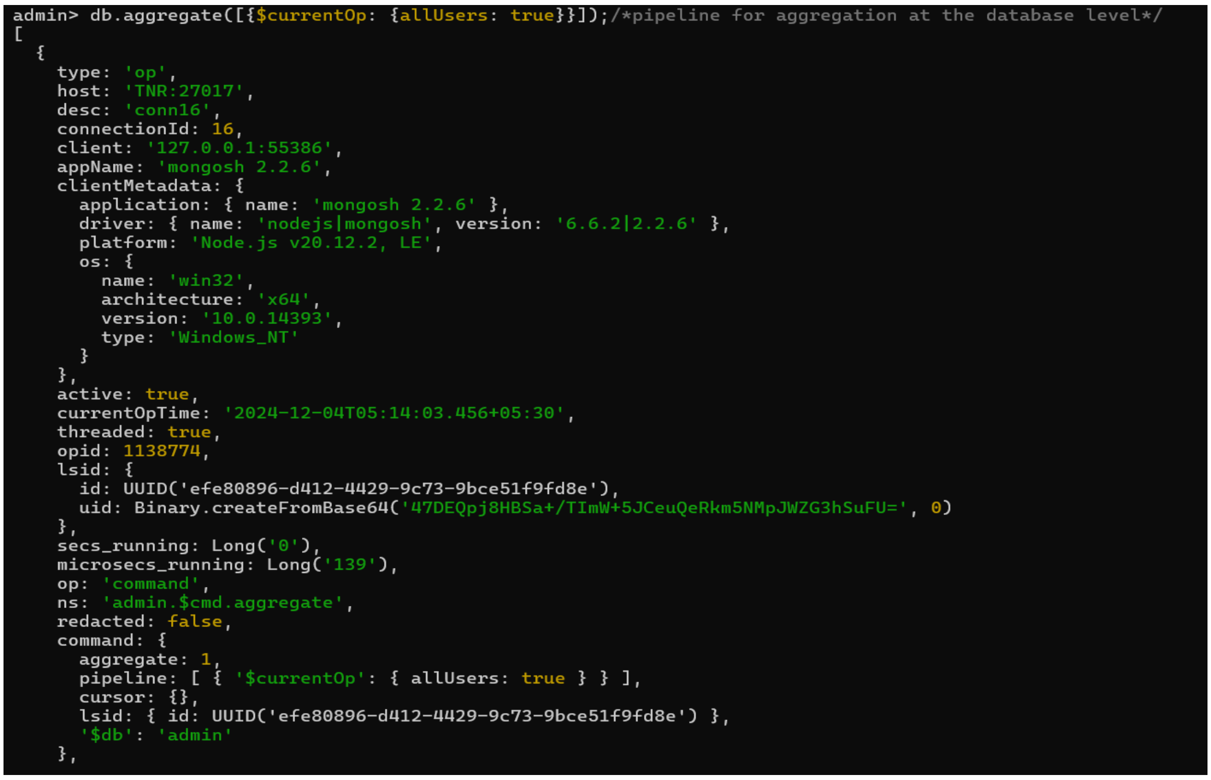
```
project> printjson(db.getCollectionInfos()); /* return matedata about collections*/
[
  {
    name: 'product',
    type: 'collection',
    options: {},
    info: {
      readOnly: false,
      uuid: UUID('b1d2ac67-3c89-4882-93e7-db82fa6b2e4b')
    },
    idIndex: {
      v: 2,
      key: {
        _id: 1
      },
      name: '_id_'
    }
  }
]
```



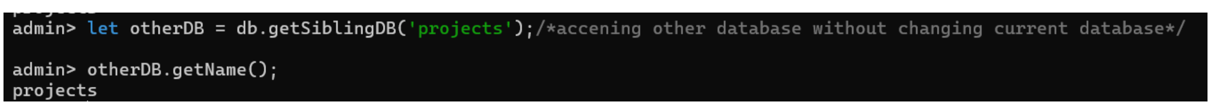
```
project> db.runCommand({ping:1});/*states of database */
{ ok: 1 }
```



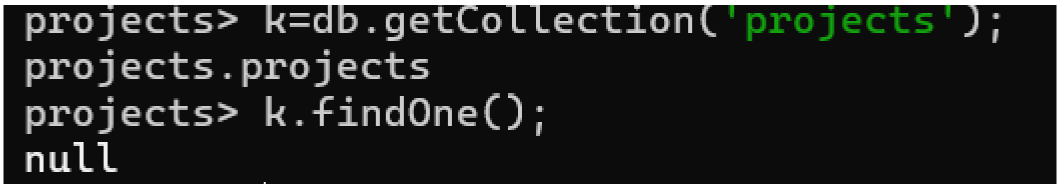
```
project> db.adminCommand({listDatabases: 1}); /* return list of all databases */
{
  databases: [
    { name: 'admin', sizeOnDisk: Long('40960'), empty: false },
    { name: 'config', sizeOnDisk: Long('110592'), empty: false },
    { name: 'local', sizeOnDisk: Long('90112'), empty: false },
    { name: 'practise', sizeOnDisk: Long('700416'), empty: false },
    { name: 'project', sizeOnDisk: Long('45056'), empty: false }
  ],
  totalSize: Long('987136'),
  totalSizeMb: Long('0'),
  ok: 1
}
```



```
admin> db.aggregate([{$currentOp: {allUsers: true}}]);/*pipeline for aggregation at the database level*/
[
  {
    type: 'op',
    host: 'TNR:27017',
    desc: 'conn16',
    connectionId: 16,
    client: '127.0.0.1:55386',
    appName: 'mongosh 2.2.6',
    clientMetadata: {
      application: { name: 'mongosh 2.2.6' },
      driver: { name: 'nodejs|mongosh', version: '6.6.2|2.2.6' },
      platform: 'Node.js v20.12.2, LE',
      os: {
        name: 'win32',
        architecture: 'x64',
        version: '10.0.14393',
        type: 'Windows_NT'
      }
    },
    active: true,
    currentOpTime: '2024-12-04T05:14:03.456+05:30',
    threaded: true,
    opid: 1138774,
    lsid: {
      id: UUID('efe80896-d412-4429-9c73-9bce51f9fd8e'),
      uid: Binary.createFromBase64('47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=', 0)
    },
    secs_running: Long('0'),
    microsecs_running: Long('139'),
    op: 'command',
    ns: 'admin.$cmd.aggregate',
    redacted: false,
    command: {
      aggregate: 1,
      pipeline: [ { '$currentOp': { allUsers: true } } ],
      cursor: {},
      lsid: { id: UUID('efe80896-d412-4429-9c73-9bce51f9fd8e') },
      '$db': 'admin'
    },
```

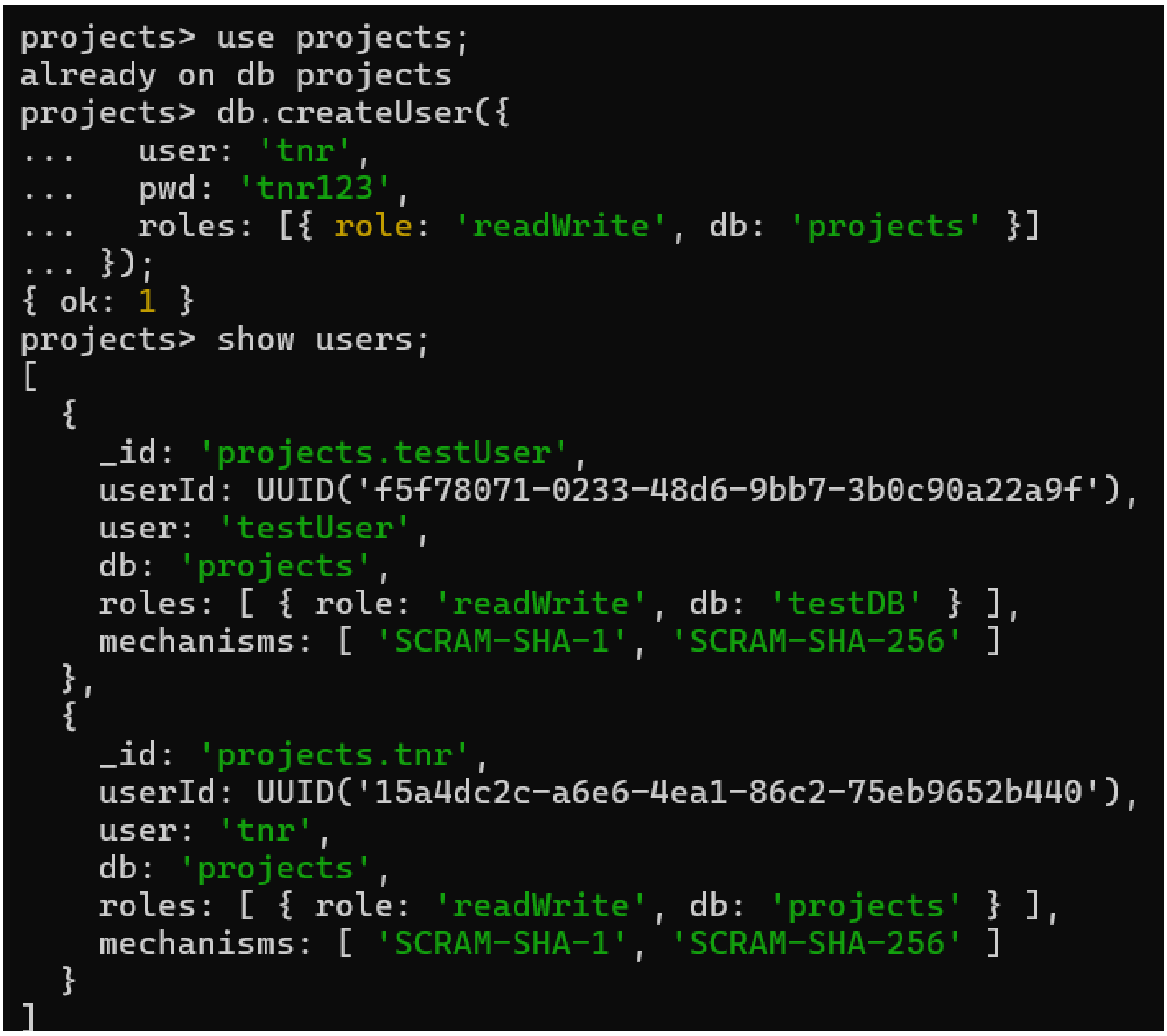


```
admin> let otherDB = db.getSiblingDB('projects');/*accening other database without changing current database*/

admin> otherDB.getName();
projects
```



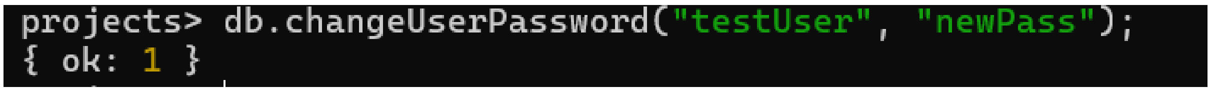
```
projects> k=db.getCollection('projects');
projects.projects
projects> k.findOne();
null
```

db.dropDatabase();



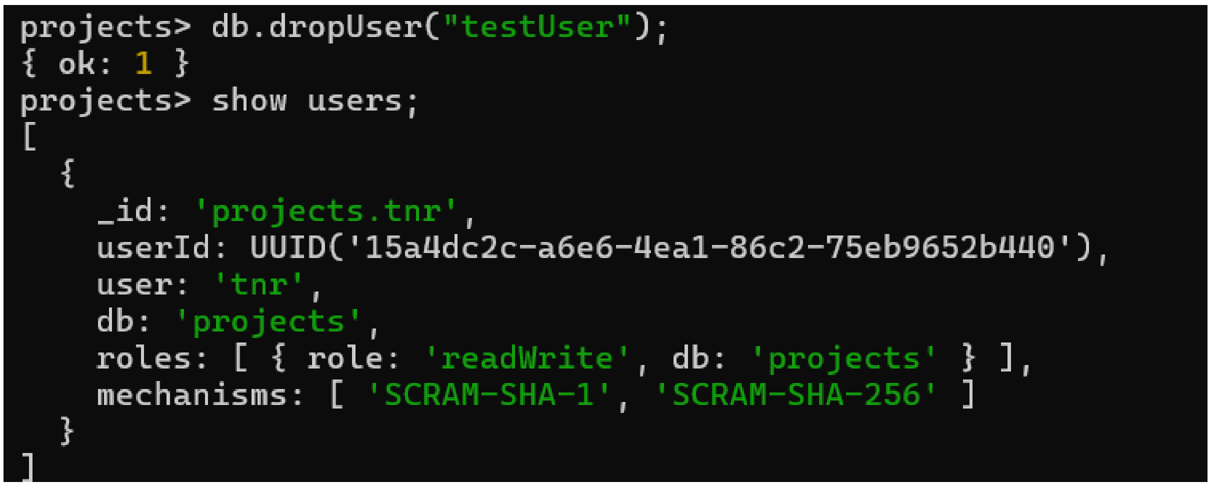
```
projects> use projects;
already on db projects
projects> db.createUser({
...   user: 'tnr',
...   pwd: 'tnr123',
...   roles: [{ role: 'readWrite', db: 'projects' }]
... });
{ ok: 1 }
projects> show users;
[
  {
    _id: 'projects.testUser',
    userId: UUID('f5f78071-0233-48d6-9bb7-3b0c90a22a9f'),
    user: 'testUser',
    db: 'projects',
    roles: [ { role: 'readWrite', db: 'testDB' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  },
  {
    _id: 'projects.tnr',
    userId: UUID('15a4dc2c-a6e6-4ea1-86c2-75eb9652b440'),
    user: 'tnr',
    db: 'projects',
    roles: [ { role: 'readWrite', db: 'projects' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
]
```

db.updateUser("testUser", {roles: [{role: "read", db: "testDB"}]});

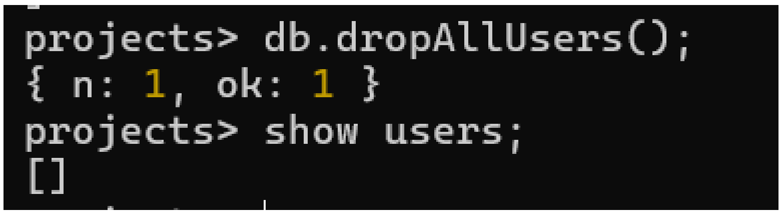


```
projects> db.changeUserPassword("testUser", "newPass");
{ ok: 1 }
```

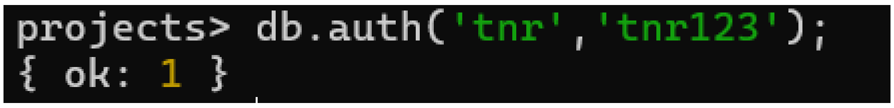
db.logout();



```
projects> db.dropUser("testUser");
{ ok: 1 }
projects> show users;
[
  {
    _id: 'projects.tnr',
    userId: UUID('15a4dc2c-a6e6-4ea1-86c2-75eb9652b440'),
    user: 'tnr',
    db: 'projects',
    roles: [ { role: 'readWrite', db: 'projects' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
]
```



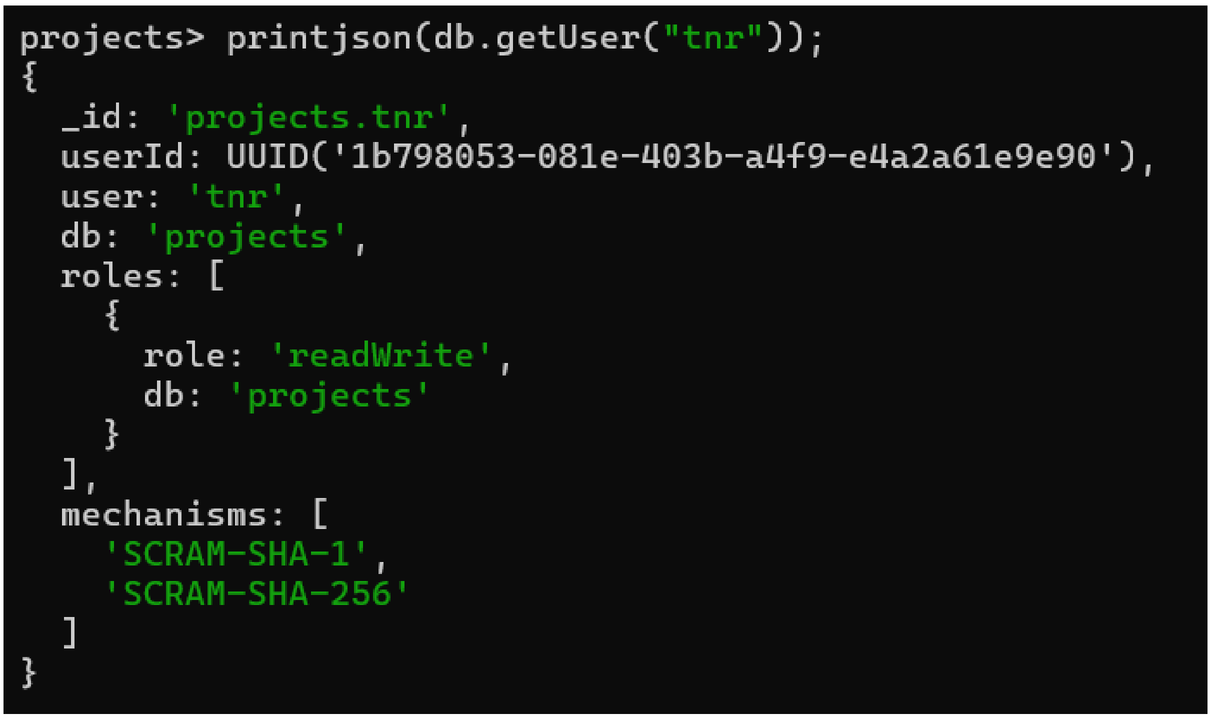
```
projects> db.dropAllUsers();
{ n: 1, ok: 1 }
projects> show users;
[]
```



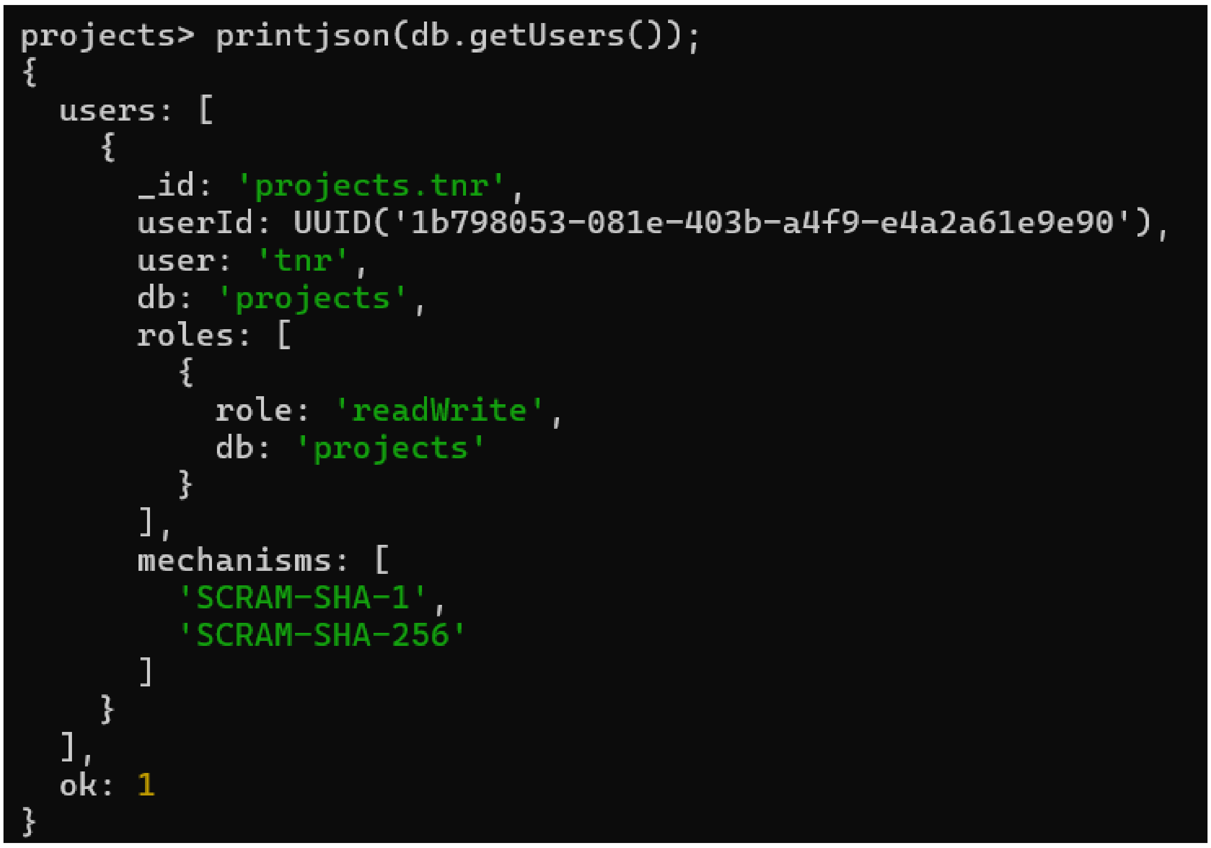
```
projects> db.auth('tnr','tnr123');
{ ok: 1 }
```

db.grantRolesToUser("testUser", [{role: "dbAdmin", db: "testDB"}]);

db.revokeRolesFromUser("testUser", [{role: "readWrite", db: "testDB"}]);



```
projects> printjson(db.getUser("tnr"));
{
  _id: 'projects.tnr',
  userId: UUID('1b798053-081e-403b-a4f9-e4a2a61e9e90'),
  user: 'tnr',
  db: 'projects',
  roles: [
    {
      role: 'readWrite',
      db: 'projects'
    }
  ],
  mechanisms: [
    'SCRAM-SHA-1',
    'SCRAM-SHA-256'
  ]
}
```



```
projects> printjson(db.getUsers());
{
  users: [
    {
      _id: 'projects.tnr',
      userId: UUID('1b798053-081e-403b-a4f9-e4a2a61e9e90'),
      user: 'tnr',
      db: 'projects',
      roles: [
        {
          role: 'readWrite',
          db: 'projects'
        }
      ],
      mechanisms: [
        'SCRAM-SHA-1',
        'SCRAM-SHA-256'
      ]
    }
  ],
  ok: 1
}
```

db.createCollection("newCollection");

db.createEncryptedCollection("encryptedCollection", {encryptedFields: ["field1"]});