|  |  |
| --- | --- |
| **EX NO: 10** | **CREATE AN XML DATABASE AND VALIDATE IT USING XML SCHEMA.** |
| **DATE:** |
| **AIM:**  To create a xml database and validate using xml schema.  **PROCEDURE FOR VALIDATING USING XML SCHEMA:**  **Step 1**: Open notepad++.  **Step 2:** Open new file.  **Step 3:** Write your XML file.  **Step 4:** Save file with .xml extension.  **Step 5:** Open another new file and write schema file.  **Step 6:** Save schema file with .xsd extension.  **Step 7:** In the Menu bar click on Plugins. Open Plugins menu open plugins admin.  **Step 8:** In Plugins Admin search and install XML Tools.  **Step 9:** Click Plugins and under XML Tools Click Validate now.  **Step 10:** Enter the location of the schema file.  **Step 11:** Click ‘OK’.  **CREATING XML DATABASE:**  <?xml version="1.0" encoding="UTF-8"?>  <student>  <std1>  <name>ajil</name>  <regno>1</regno>  <address>ngl</address>  </std1>  <std2>  <name>allen</name>  <regno>2</regno>  <address>TVL</address>  </std2>  <std3>  <name>abu</name>  <regno>3</regno>  <address>Tvm</address>  </std3>  </student> | |

**XML SCHEMA:**

<xs:schemaattributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs=["http://www](http://www.w3.org/2001/XMLSchema).[w3.org/2001/XMLSchema">](http://www.w3.org/2001/XMLSchema)

<xs:element name="student">

<xs:complexType>

<xs:sequence>

<xs:element name="std1">

<xs:complexType>

<xs:sequence>

<xs:element name="name" type="xs:string" />

<xs:element name="regno" type="xs:unsignedByte" />

<xs:element name="address" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="std2">

<xs:complexType>

<xs:sequence>

<xs:element name="name" type="xs:string" />

<xs:element name="regno" type="xs:unsignedByte" />

<xs:element name="address" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="std3">

<xs:complexType>

<xs:sequence>

<xs:element name="name" type="xs:string" />

<xs:element name="regno" type="xs:unsignedByte" />

<xs:element name="address" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

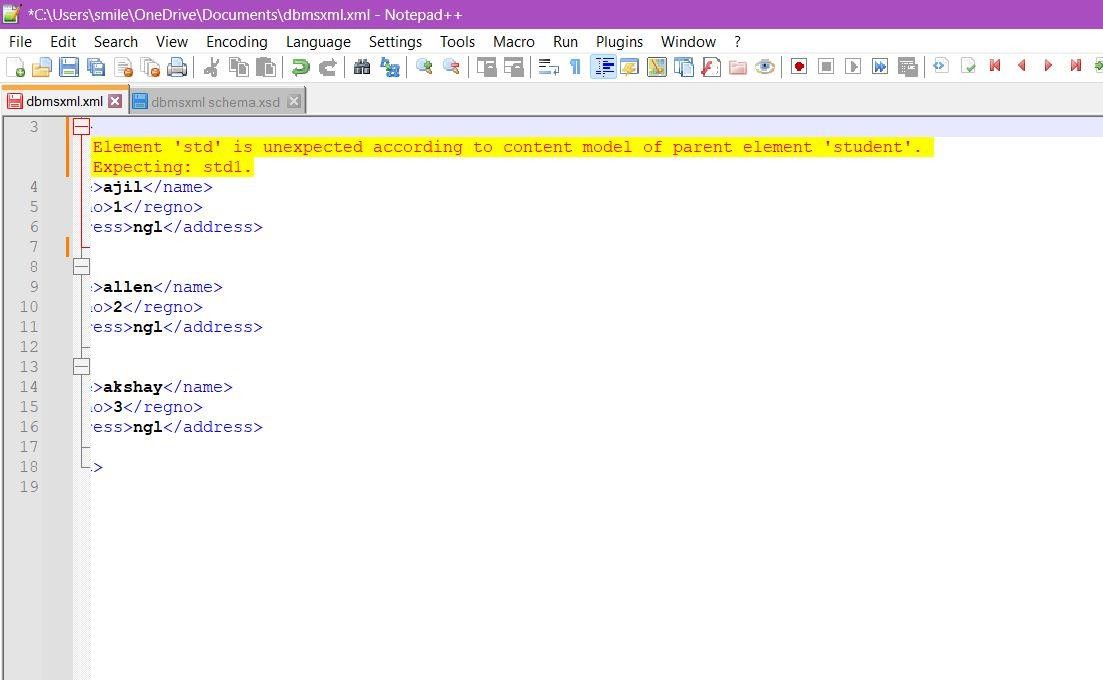
</xs:sequence>

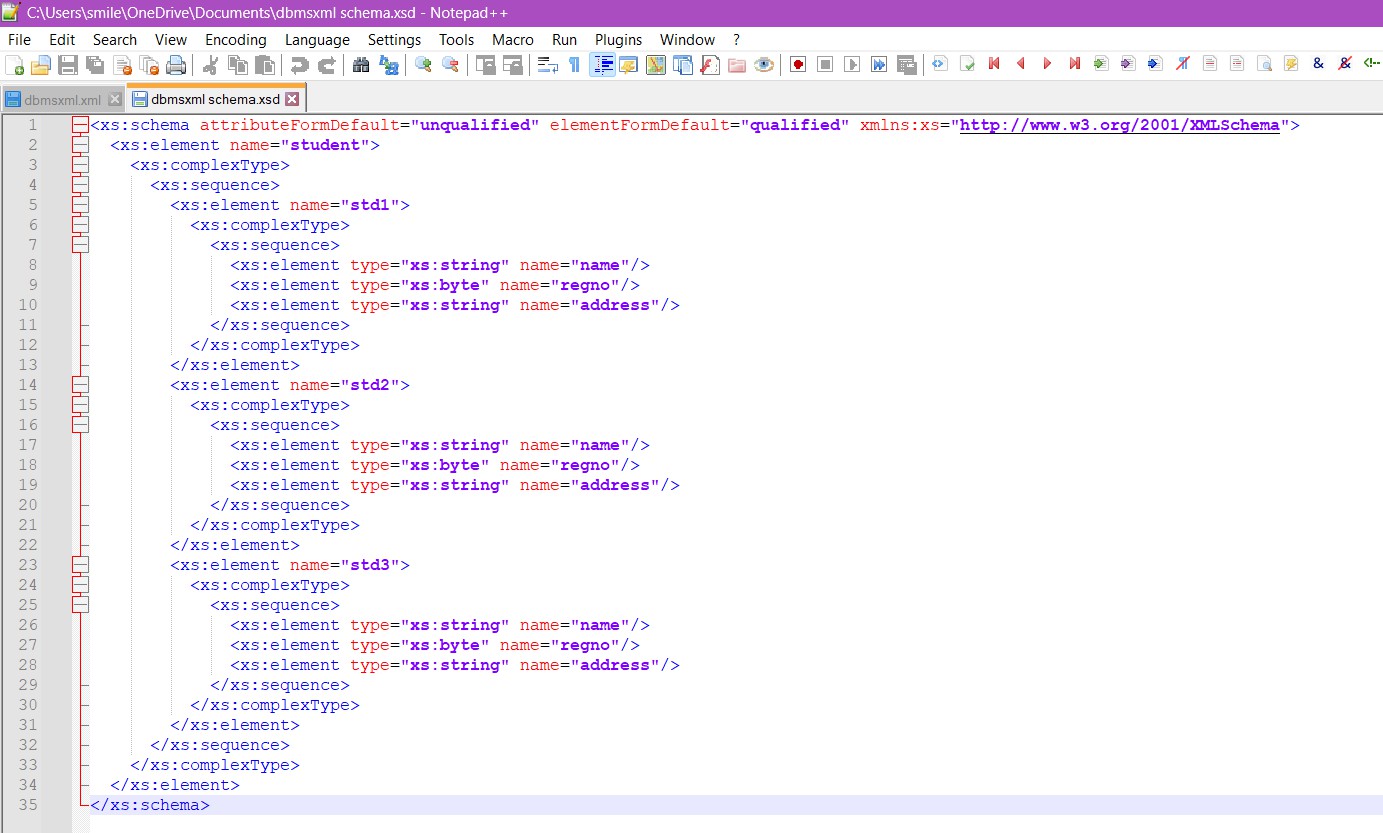
</xs:complexType>

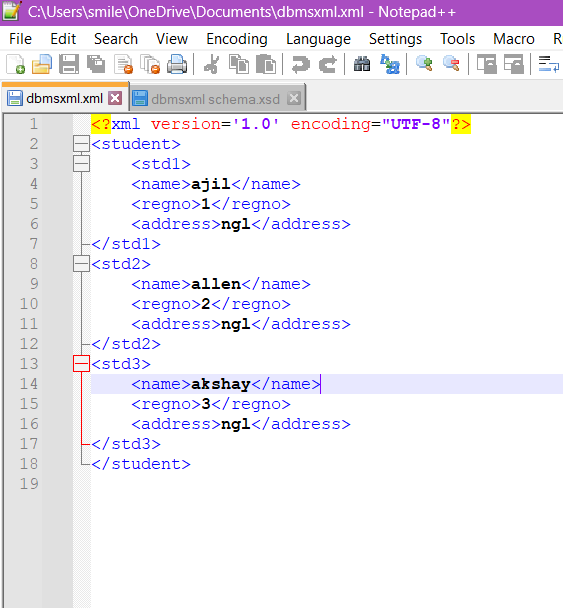
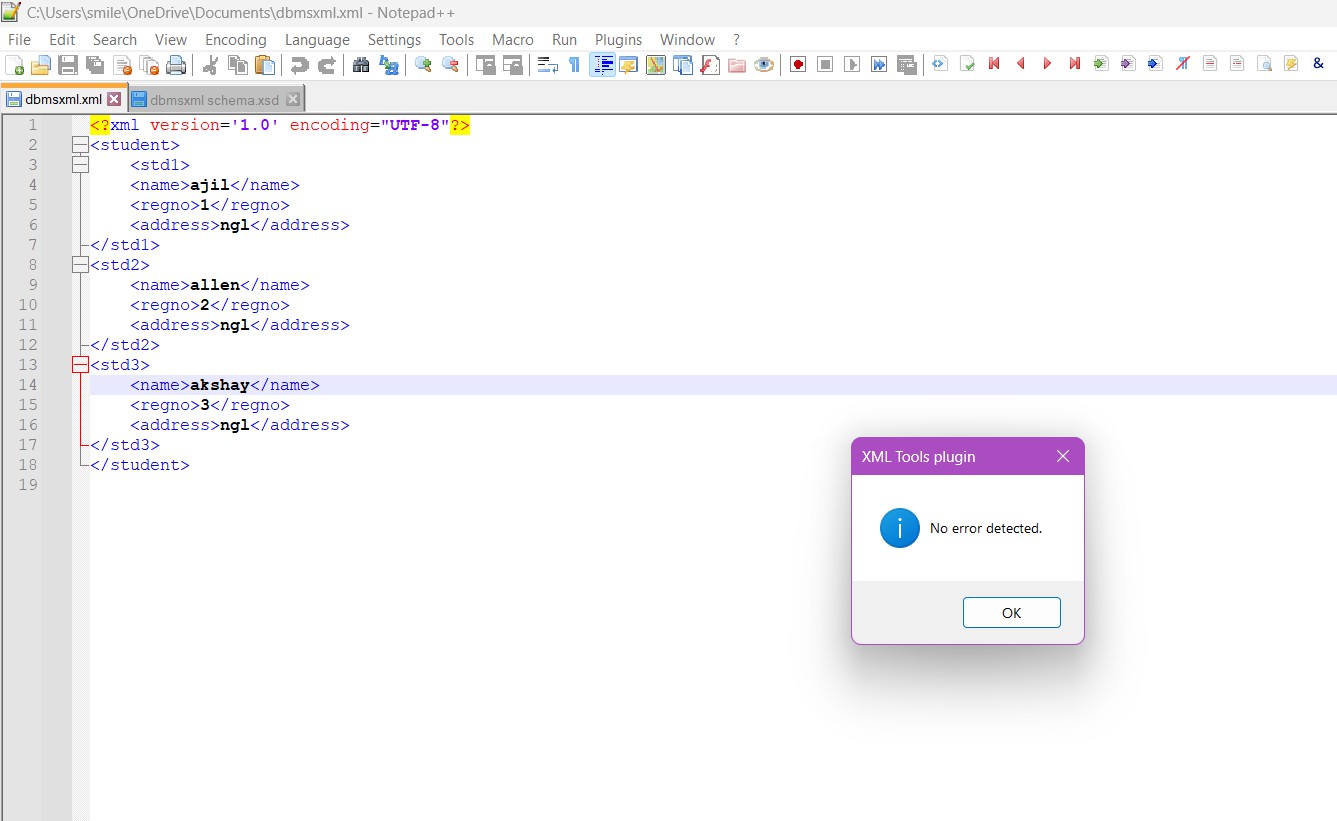
</xs:element>

</xs:schema>

**QUERY AND OUTPUT:**







**RESULT:**

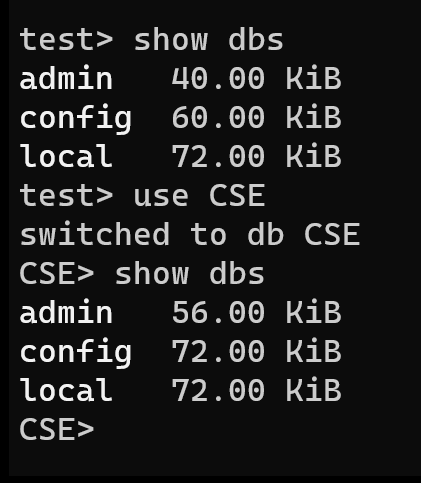
Thus, the SQL query to create a set of tables, add foreign key constraints and incorporate referential integrity is verified and executed successfully.

|  |  |
| --- | --- |
| **EX NO:11** | **CREATE DOCUMENT, COLUMN AND GRAPH BASED DATA USING NOSQL DATABASE TOOLS** |
| **DATE:** |
| **AIM:**  To create a document-based NoSQL database using MongoDB, with columns and graphs, and populate it with sample data.  **NOSQL:**  NOSQL (Not Only SQL) is a type of database that does not use the traditional relational model that has been used in traditional SQL databases. Instead, NoSQL databases use a non- relational approach for data storage and retrieval. They are designed to handle large amounts of unstructured, semi-structured, and structured data, including text, images, videos, and social media data.  NoSQL databases are often used in big data and real-time web applications, where high performance and scalability are essential. Some popular examples of NoSQL databases include MongoDB, Cassandra, Couchbase, and Apache HBase.  **VIEW DATABASE:**  In MongoDB, a view is a virtual collection that presents the results of a pre-defined aggregation pipeline as if it were a collection of documents. Views do not store data themselves but rather provide a read-only representation of the data in one or more underlying collections.  **QUERY AND OUTPUT:**  mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000  **CREATING DATABASE:**  In the mongo shell, you can create a database with the help of the following command.  This command actually switches you to the new database if the given name does not exist and if the given name exists, then it will switch you to the existing database. Now at this stage, if you use the show command to see the database list where you will find that your new database is not present in that database list because, in MongoDB, the database is actually created when you start entering data in that database. | |

**SYNTAX:**

use database\_name

**QUERY AND OUTPUT:**



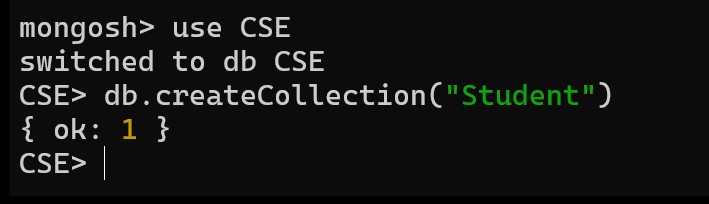
**CREATING COLLECTION:**

In MongoDB, a collection is a grouping of MongoDB documents, similar to a table in a relational database. To create a collection in MongoDB, you can use the db.createCollection() method.

**SYNTAX:**

db.createCollection(name, options)

**QUERY AND OUTPUT:**



**INSERTMANY COMMAND:**

insertMany is a method in MongoDB that allows you to insert multiple documents into a collection at once. The method takes an array of documents as its argument, and inserts each document as a separate document in the collection.

**SYNTAX:**

db.collection.insertMany(

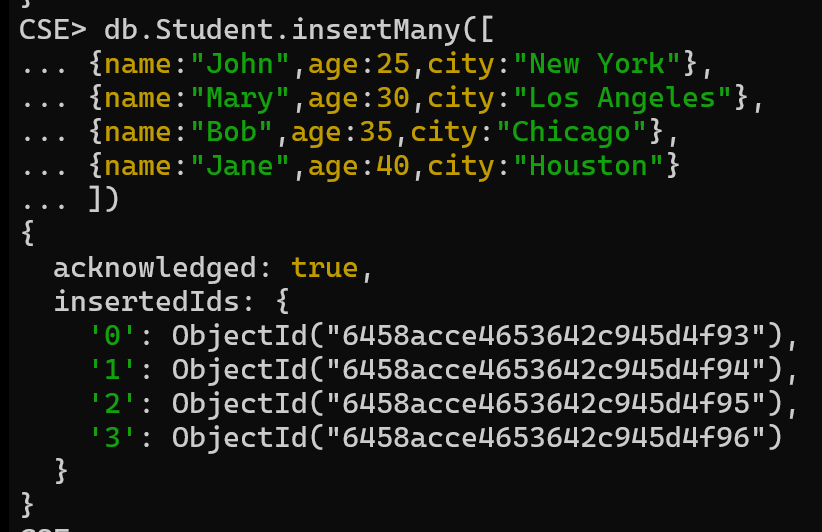
[ <document 1> , <document 2>, ... ],

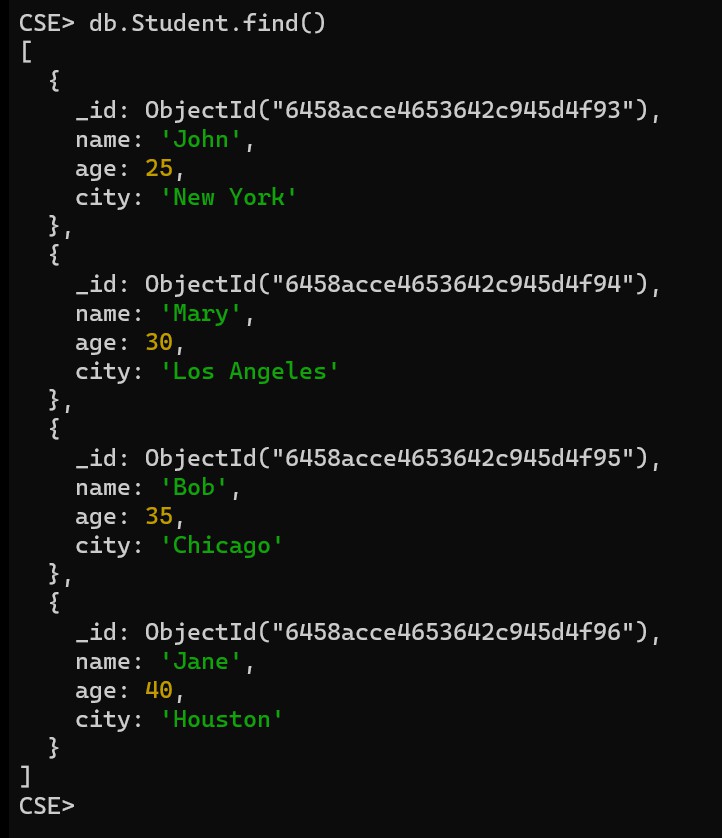
{

writeConcern: <document>, ordered: <boolean>

}

)

**QUERY AND OUTPUT: DOCUMENT VIEW:**

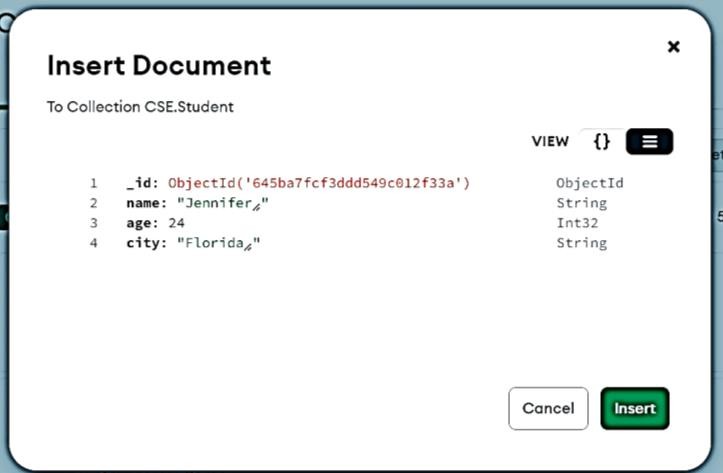


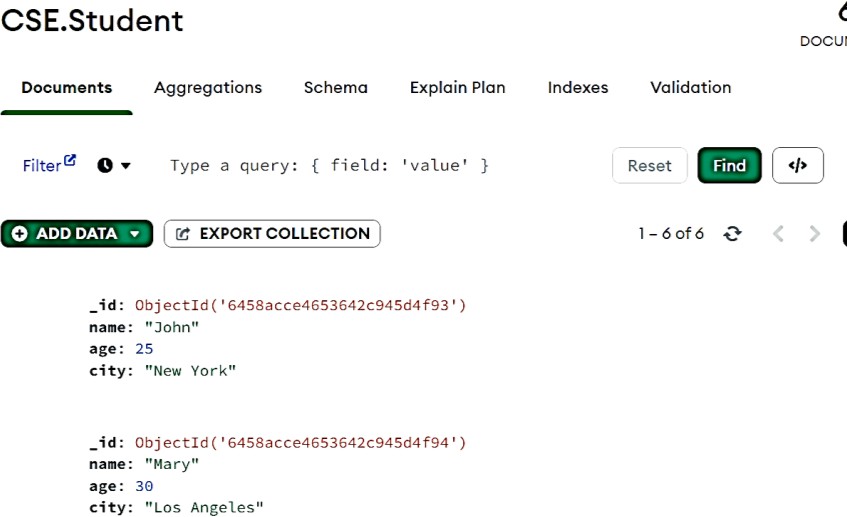
**GRAPHICAL VIEW:**

Step 1: Open MongoDB Compass Application. Step 2: Click Connect on the displayed window.

# Step 3: From the Databases select the database which has the collection you want to insert document.

Step 4: Click the collection -> ADD DATA -> Insert document. Add fields to the document and click Insert.





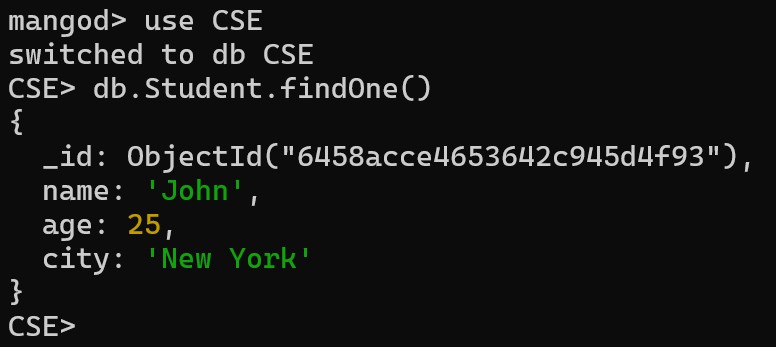
**FINDING A SINGLE DOCUMENT:**

In MongoDB, we can find a single document using the findOne() method, This method returns the first document that matches the given filter query expression.

**SYNTAX:**

db.collection\_name.findOne ()

**QUERY AND OUTPUT:**



**DISPLAYING DOCUMENTS IN A FORMATTED WAY:**

In MongoDB, we can display documents of the specified collection in a well-formatted way using the pretty() method.

**SYNTAX:**

db.collection\_name.find().pretty()

**QUERY AND OUTPUT:**



**GREATER THAN FILTER QUERY:**

To get the specific numeric data using conditions like greater than equal or less than equal use the $gte or $lte operator in the find() method.

**SYNTAX:**

db.collection\_name.find({< key > : {$gte : < value >}}) or

db.collection\_name.find({< key > : {$lte : < value >}})

**QUERY AND OUTPUT:**



**DELETING THE FIRST DOCUMENT:**

In this example, we are deleting the first document from the Student collection by passing an empty document in the db.collection.deleteOne() method.

**SYNTAX:**

db.collection\_name.deleteOne({})

**QUERY AND OUTPUT:**



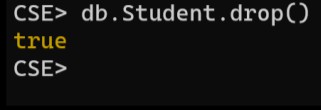
**DROPPING A COLLECTION:**

In MongoDB, a collection is a group of MongoDB documents that are stored together. Dropping a collection in MongoDB means permanently deleting the entire collection and all of its contents from the database.

**SYNTAX:**

db.collection\_name.drop()

**QUERY AND OUTPUT:**



**DROP A DATABASE:**

In MongoDB, databases hold collections of documents. On a single MongoDB server, we can run multiple databases. when you install MongoDB some databases are automatically generated to use. many times, you need to delete some database when the database is no longer used.

db.dropDatabase() the command is used to drop an existing database. This command will delete the currently selected database. If you have not selected any database, then it will delete the default ‘test’ database.

**SYNTAX:**

db.dropDatabase()

**QUERY AND OUTPUT:**



**RESULT:**

Thus create a document-based NoSQL database using MongoDB, with columns and graphs, and populate it with sample data.