

LAB PROGRAM 1: MongoDB- CRUD Demonstration

1) Using MongoDB

i) Create a database for Students and Create a Student Collection (_id,Name, USN, Semester, Dept_Name, CGPA, Hobbies(Set)).

```
use myDB;
```

```
db.createCollection("Student");
```

ii) Insert required documents to the collection.

```
> db.Student.insert({_id:1,Name: "Pranav", sem:"VI",dept: "CSE",CGPA: 8.2,hobbies: ['cycling']});
```

```
WriteResult({ "nInserted" : 1 })
```

```
> db.Student.insert({_id:2,Name: "Anurag", sem:"VII",dept: "ECE",CGPA: 6.8,hobbies: ["Biking"]});
```

```
WriteResult({ "nInserted" : 1 })
```

```
> db.Student.insert({_id:3,Name: "Saurab", sem:"VI",dept:"Architecture",CGPA: 8.8,hobbies: ['Gaming']});
```

```
WriteResult({ "nInserted" : 1 })
```

```
> db.Student.insert({_id:4,Name: "Prateek", sem:"V",dept: "ISE",CGPA: 9.1,hobbies: ["Badminton"]});
```

```
WriteResult({ "nInserted" : 1 })
```

```

> db.Student.insert({_id:1,Name: "Pranav", sem:"VI",dept: "CSE",CGPA: 8.2,hobbies: ['cycling']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,Name: "Anurag", sem:"VII",dept: "ECE",CGPA: 6.8,hobbies: ["Biking"]});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,Name: "Saurab", sem:"VI",dept:"Architecture",CGPA: 8.8,hobbies: ['Gaming']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:4,Name: "Prateek", sem:"V",dept: "ISE",CGPA: 9.1,hobbies: ["Badminton"]});
WriteResult({ "nInserted" : 1 })
> db.Student.find()
{ "_id" : 1, "Name" : "Pranav", "sem" : "VI", "dept" : "CSE", "CGPA" : 8.2, "hobbies" : [ "cycling" ] }
{ "_id" : 2, "Name" : "Anurag", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "Biking" ] }
{ "_id" : 3, "Name" : "Saurab", "sem" : "VI", "dept" : "Architecture", "CGPA" : 8.8, "hobbies" : [ "Gaming" ] }
{ "_id" : 4, "Name" : "Prateek", "sem" : "V", "dept" : "ISE", "CGPA" : 9.1, "hobbies" : [ "Badminton" ] }
>

```

iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and compute the Average CPGA for that semester and filter those documents where the “Avg_CPGA” is greater than 7.5.

```

>db.Student.aggregate({$match:{dept:"CSE"}},{ $group:{_id:"$sem",AverageCGPA:{ $avg:"$CGPA"} }},{ $match:{AverageCGPA:{ $gt:7.5}}});

```

```

> db.Student.aggregate({$match:{dept:"CSE"}},{ $group:{_id:"$sem",AverageCGPA:{ $avg:"$CGPA"} }},{ $match:{AverageCGPA:{ $gt:7.5}}});
{ "_id" : "VI", "AverageCGPA" : 8.2 }
>

```

iv) Insert the document for “Bhuvan” in to the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his Hobbies to “Skating”) Use “Update else insert” (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

```

>db.Student.update({_id:5},{ $set:{ "hobbies": "Cricket" }},{ $upsert:true});

```

```

> db.Student.update({_id:5},{ $set:{ "hobbies": "Cricket" }},{ $upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find({_id:5});
{ "_id" : 5, "Name" : "Bhuvan", "sem" : "VI", "dept" : "CSE", "CGPA" : 9.5, "hobbies" : "Cricket" }
>

```

v)To display only the StudName and Grade from all the documents of the Students collection. The identifier_id should be suppressed and NOT displayed.

```
> db.Student.find({}, {_id:0,"Name":1,"sem":1});
```

```
> db.Student.find({}, {_id:0,"Name":1,"sem":1});
{ "Name" : "Pranav", "sem" : "VI" }
{ "Name" : "Anurag", "sem" : "VII" }
{ "Name" : "Saurab", "sem" : "VI" }
{ "Name" : "Prateek", "sem" : "V" }
{ "Name" : "Bhuvan", "sem" : "VI" }
```

vi) To find those documents where the Grade is set to 'VII'.

```
> db.Student.find({"sem":"VII"});
```

```
> db.Student.find({"sem":"VII"});
{ "_id" : 2, "Name" : "Anurag", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "Biking" ] }
>
```

vii)To find those documents from the Students collection where the Hobbies is set to either 'Chess' or is set to 'Skating'.

```
> db.Student.find({"hobbies":{$in:["Badminton","Gaming"]} });
```

```
> db.Student.find({"hobbies":{$in:["Badminton","Gaming"]} });
{ "_id" : 3, "Name" : "Saurab", "sem" : "VI", "dept" : "Architecture", "CGPA" : 8.8, "hobbies" : [ "Gaming" ] }
{ "_id" : 4, "Name" : "Prateek", "sem" : "V", "dept" : "ISE", "CGPA" : 9.1, "hobbies" : [ "Badminton" ] }
{ "_id" : 5, "Name" : "Bhuvan", "sem" : "VI", "dept" : "CSE", "CGPA" : 9.5, "hobbies" : [ "Cricket", "Badminton" ] }
```

viii)To find documents from the Students collection where the StudName begins with "B" .

```
> db.Student.find({"Name":/^A/});
```

```
> db.Student.find({"Name":/^A/});
{ "_id" : 2, "Name" : "Anurag", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "Biking" ] }
```

ix) To find the number of documents in the Students collection.

```
> db.Student.count();
```

```
> db.Student.count();
5
```

x) To sort the documents from the Students collection in the descending order of StudName.

```
> db.Student.find().sort({"Name":-1});
```

```
> db.Student.find().sort({"Name":-1});
{ "_id" : 3, "Name" : "Saurab", "sem" : "VI", "dept" : "Architecture", "CGPA" : 8.8, "hobbies" : [ "Gaming" ] }
{ "_id" : 4, "Name" : "Prateek", "sem" : "V", "dept" : "ISE", "CGPA" : 9.1, "hobbies" : [ "Badminton" ] }
{ "_id" : 1, "Name" : "Pranav", "sem" : "VI", "dept" : "CSE", "CGPA" : 8.2, "hobbies" : [ "cycling" ] }
{ "_id" : 5, "Name" : "Bhuvan", "sem" : "VI", "dept" : "CSE", "CGPA" : 9.5, "hobbies" : [ "Cricket", "Badminton" ] }
{ "_id" : 2, "Name" : "Anurag", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "Biking" ] }
```

xi) Command used to export MongoDB JSON documents from “Student” Collection into the “Students” database into a CSV file “Output.txt”

```
> mongoexport --host localhost --db Student --collection Student --csv --out /Downloads/student.txt -fields "Name","sem";
```

```
> mongoexport --host localhost --db Student --collection Student --csv --out /Downloads/student.txt -fields "Name","sem";
uncaught exception: SyntaxError: unexpected token: identifier :
@:(shell):1:14
```

LAB PROGRAM 2: Employee database using Cassandra

Program 1. Perform the following DB operations using Cassandra.

1. Create a key space by name Employee

```
cqlsh> create keyspace Employee with REPLICATION ={  
'class':'SimpleStrategy','replication_factor':1 ... };
```

```
bmsce@bmsce-Precision-T1700:~/cassandra/apache-cassandra-3.11.0/bin$ cqlsh  
Connected to Test Cluster at 127.0.0.1:9042.  
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]  
Use HELP for help.
```

```
cqlsh> use Employee; cqlsh:employee> describe keyspaces;
```

```
students system_auth system_distributed system_traces system_schema system  
employee
```

```
cqlsh> describe keyspace employee;  
  
CREATE KEYSPACE employee WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'} AND durable_writes = true;
```

2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name

```
cqlsh:employee> CREATE TABLE Employee_Info( ... emp_id int PRIMARY KEY, ...  
emp_name text, ... designation text, ... date_of_joining timestamp, ... salary  
double, ... dept_name text ... );
```

```
cqlsh:employee> describe tables
```

```
employee_info
```

```
cqlsh:employee> describe table employee_info

CREATE TABLE employee.employee_info (
  emp_id int PRIMARY KEY,
  date_of_joining timestamp,
  dept_name text,
  designation text,
  emp_name text,
  salary double
) WITH additional_write_policy = '99p'
   AND bloom_filter_fp_chance = 0.01
   AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
   AND cdc = false
   AND comment = ''
   AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
   AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
   AND crc_check_chance = 1.0
   AND default_time_to_live = 0
   AND extensions = {}
   AND gc_grace_seconds = 864000
   AND max_index_interval = 2048
   AND memtable_flush_period_in_ms = 0
   AND min_index_interval = 128
   AND read_repair = 'BLOCKING'
   AND speculative_retry = '99p';
```

3. Insert the values into the table in batch cqlsh:employee>BEGIN BATCH

```
cqlsh:employees> BEGIN BATCH
```

```
    ... INSERT INTO
```

```
    ...
```

```
employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name
)
```

```
    ... values(124,'Pranav','Manager','2000-09-24',750000,'Export')
```

```
    ...
```

```
employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name
)
```

```
    ... values(125,'Anurag','AsstManager','2000-01-04',550000,'Export')
```

```
    ...
```

```
employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name
)
```

```
    ... values(126,'Prateek','HR','2000-05-04',650000,'HR')
```

```
    ... APPLY BATCH;
```

```
cqlsh:employee> select * from employee_info;
```

emp_id	date_of_joining	dept_name	designation	emp_name	salary
125	2000-01-03 18:30:00.000000+0000	Export	AsstManager	Anurag	5.5e+05
126	2000-05-03 18:30:00.000000+0000	HR	HR	Prateek	6.5e+05
124	2000-09-23 18:30:00.000000+0000	Export	Manager	Pranav	7.5e+05

4. Update Employee name and Department of Emp-Id 125

```
cqlsh:employees> update employee_info set dept_name='import' where  
emp_id=125;
```

```
cqlsh:employees> SELECT* FROM employee_info;
```

emp_id	date_of_joining	dept_name	designation	emp_name	salary
125	2000-01-03 18:30:00.000000+0000	import	AsstManager	Saurab	5.5e+05
126	2000-05-03 18:30:00.000000+0000	HR	HR	Prateek	6.5e+05
124	2000-09-23 18:30:00.000000+0000	Export	Manager	Pranav	7.5e+05

6. Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

```
cqlsh:employee> alter table employee_info ... add project text; cqlsh:employee>
select * from employee_info;
```

```
cqlsh:employees> ALTER TABLE employee_info add project set<text>;
```

```
cqlsh:employees> SELECT* FROM employee_info;
```

emp_id	date_of_joining	dept_name	designation	emp_name	project	salary
125	2000-01-03 18:30:00.000000+0000	import	AsstManager	Saurab	null	5.5e+05
126	2000-05-03 18:30:00.000000+0000	HR	HR	Prateek	null	6.5e+05
124	2000-09-23 18:30:00.000000+0000	Export	Manager	Pranav	null	7.5e+05

7. Update the altered table to add project names.

```
cqlsh:employees> update employee_info set project={'pro4555','pro2566'} where
emp_id=126;
```

```
cqlsh:employees> update employee_info set project={'pro45','pro25'} where
emp_id=124;
```

```
cqlsh:employees> update employee_info set project={'pro1','pro2'} where
emp_id=125;
```

```
cqlsh:employees> SELECT* FROM employee_info;
```

emp_id	date_of_joining	dept_name	designation	emp_name	project	salary
--------	-----------------	-----------	-------------	----------	---------	--------


```
-----+-----+-----+-----+-----+-----
+-----
125 | 2000-01-03 18:30:00.000000+0000 | import | AsstManager | Saurab |
{'pro1', 'pro2'} | 5.5e+05
126 | 2000-05-03 18:30:00.000000+0000 | HR | HR | Prateek |
{'pro2566', 'pro4555'} | 6.5e+05
124 | 2000-09-23 18:30:00.000000+0000 | Export | Manager | Pranav |
{'pro25', 'pro45'} | 7.5e+05
```

LAB PROGRAM 3: Library database using Cassandra

1 Create a key space by name Library

```
cqlsh> create keyspace libraries with  
replication={'class':'SimpleStrategy','replication_factor':1};
```

```
cqlsh> use libraries;
```

2. Create a column family by name Library-Info with attributes Stud_Id Primary Key, Counter_value of type Counter, Stud_Name, Book-Name, Book-Id, Date_of_issue

```
cqlsh:libraries> CREATE TABLE library_info(Stud_id int, Stud_name text,  
Book_name text, Book_id int, Date_of_issue timestamp, counter_value counter,  
PRIMARY KEY(Stud_id,Stud_name,Book_name,Book_id,Date_of_issue));
```

3. Insert the values into the table in batch cqlsh:library

```
cqlsh:libraries> UPDATE library_info SET counter_value = counter_value + 1  
WHERE Stud_id = 123 AND Stud_name = 'Anurag' AND Book_name = 'BDA' AND  
Book_id = 455 AND Date_of_issue = '2000-09-24';
```

```
cqlsh:libraries> UPDATE library_info SET counter_value = counter_value + 1  
WHERE Stud_id = 123 AND Stud_name = 'Pranav' AND Book_name = 'ADS' AND  
Book_id = 45 AND Date_of_issue = '2003-05-04';
```

```
cqlsh:libraries> UPDATE library_info SET counter_value = counter_value + 1  
WHERE Stud_id = 123 AND Stud_name = 'Saurab' AND Book_name = 'CHY' AND  
Book_id = 245 AND Date_of_issue = '2003-05-07';
```

```
cqlsh:libraries> UPDATE library_info SET counter_value = counter_value + 1  
WHERE Stud_id = 123 AND Stud_name = 'Prateek' AND Book_name = 'CNS' AND  
Book_id = 25 AND Date_of_issue = '2003-05-09';cqlsh:libraries> select* from  
library_info;
```

stud_id	stud_name	book_name	book_id	date_of_issue	counter_value
123	Pranav	ADS	45	2003-05-03 18:30:00.000000+0000	1
123	Anurag	BDA	455	2000-09-23 18:30:00.000000+0000	1
123	Saurab	CHY	245	2003-05-06 18:30:00.000000+0000	1
123	Prateek	CNS	25	2003-05-08 18:30:00.000000+0000	1

(4 rows)

4. Display the details of the table created and increase the value of the counter

```
cqlsh:libraries> UPDATE library_info SET counter_value = counter_value + 1
WHERE Stud_id = 123 AND Stud_name = 'Prateek' AND Book_name = 'CNS' AND
Book_id = 25 AND Date_of_issue = '2003-05-09';
```

```
cqlsh:libraries> select* from library_info;
```

stud_id	stud_name	book_name	book_id	date_of_issue	counter_value
123	Pranav	ADS	45	2003-05-03 18:30:00.000000+0000	1
123	Anurag	BDA	455	2000-09-23 18:30:00.000000+0000	1
123	Saurab	CHY	245	2003-05-06 18:30:00.000000+0000	1

```
123 | Prateek | CNS | 25 | 2003-05-08 18:30:00.000000+0000 |
2
```

(4 rows)

5. Write a query to show that a student with id 1 has taken a book “BDA” 2 times.

```
cqlsh:libraries> UPDATE library_info SET counter_value = counter_value + 1
WHERE Stud_id = 123 AND Stud_name = 'Anurag' AND Book_name = 'BDA' AND
Book_id = 455 AND Date_of_issue = '2000-09-24';
```

```
cqlsh:libraries> select* from library_info;
```

```
stud_id | stud_name | book_name | book_id | date_of_issue          |
counter_value
-----+-----+-----+-----+-----+-----+-----
123 | Pranav | ADS | 45 | 2003-05-03 18:30:00.000000+0000 |      1
123 | Anurag | BDA | 455 | 2000-09-23 18:30:00.000000+0000 |
2
123 | Saurab | CHY | 245 | 2003-05-06 18:30:00.000000+0000 |
1
123 | Prateek | CNS | 25 | 2003-05-08 18:30:00.000000+0000 |
2
```

6. Export the created column to a csv file

```
cqlsh:lab2_library> copy library_info(stud_id,stud_name,book_id,date_of_issue,counter_value)to 'lib.csv';
Using 7 child processes

Starting copy of lab2_library.library_info with columns [stud_id, stud_name, book_id, date_of_issue, counter_v
alue].
Processed: 2 rows; Rate:      9 rows/s; Avg. rate:      9 rows/s
2 rows exported to 1 files in 0.250 seconds.
```

7. Import a given csv dataset from local file system into Cassandra column family

```
cqlsh:library>truncate library_info; cqlsh:library>copy
library_info(stud_id,stud_name,book_id,date_of_issue,counter_value) from
'lib.csv';
```