VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

SHIVANSHU PANDE(1BM19CS151)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
May-2022 to July-2022

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**BIG DATA ANALYTICS** " carried out by **SHIVANSHU PANDE(1BM19CS151),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **BIG DATA ANALYTICS - (20CS6PEBDA)** work prescribed for the said degree.

Dr.ANTHARA Assistant Professor Professor and Head Department of CSE Department of CSE BMSCE, Bengaluru BMSCE, Bengaluru

`

Index Sheet

Course Outcome

CO1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO3	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

1.MongoDB- CRUD Demonstration

1) Using MongoDB

```
> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
> use myDB;
switched to db myDB
> db;
myDB
> db.createCollection("Student");
2022-06-06T16:47:20.532+0530 E QUERY    [thread1] SyntaxError: illegal character @(shell):1:20
> db.createCollection('Student');
{ "ok" : 1 }
```

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

```
> db.createCollection("Student");
2022-06-06T16:47:20.532+0530 E QUERY    [thread1] SyntaxError: illegal character @(shell):1:20
> db.createCollection('Student');
{ "ok" : 1 }
> db.Student.insert({_id:1,Name:"Ravi", USN:"1BM19CS127",Sem:6,Dept_name:"CSE",CGPA:8.34,Hobbies:["Skating"]});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,Name:"Balaji", USN:"1BM19CS134",Sem:6,Dept_name:"CSE",CGPA:8.5,Hobbies:["Watching Documentaries"]});
WriteResult({ "nInserted" : 1 })
```

ii) Insert required documents to the collection.

```
> db.Student.insert({_id:5,Name:"Sagar", USN:"1BM20CS097",Sem:5,Dept_name:"ME",CGPA:7.95,Hobbies:["Collecting Coins"]});
WriteResult({ "nInserted" : 1 })
> db.Student.find();
{ "_id" : 1, "Name" : "Ravi", "USN" : "1BM19CS127", "Sem" : 6, "Dept_name" : "CSE", "CGPA" : 8.34, "Hobbies" : [ "Skating" ] }
{ "_id" : 2, "Name" : "Balaji", "USN" : "1BM19CS134", "Sem" : 6, "Dept_name" : "CSE", "CGPA" : 8.5, "Hobbies" : [ "Watching Documentaries" ] }
{ "_id" : 3, "Name" : "Skanda", "USN" : "1BM19CS137", "Sem" : 6, "Dept_name" : "CSE", "CGPA" : 8.85, "Hobbies" : [ "Solving Puzzles" ] }
{ "_id" : 4, "Name" : "Nagraj", "USN" : "1BM20CS137", "Sem" : 5, "Dept_name" : "CSE", "CGPA" : 9.25, "Hobbies" : [ "Stamp Collection" ] }
{ "_id" : 5, "Name" : "Sagar", "USN" : "1BM20CS097", "Sem" : 5, "Dept_name" : "ME", "CGPA" : 7.95, "Hobbies" : [ "Collecting Coins" ] }
```

```
> db.St
{
}
{
}
```


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({$match:{Dept_name:"CSE"}},{$group:{_id:"$Sem",Avg_CGPA:{$avg:"$CGPA"}}},{$match:{Avg_CGPA:{$gt:7.5}}}).pretty();

```
> db.Student.aggregate({$match:{Dept_name:"CSE"}}
{ "_id" : 5, "Avg_CGPA" : 9.25 }
{ "_id" : 6, "Avg_CGPA" : 8.563333333333333 }
>
```

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

mongoexport --db myDB --collection Student --type=csv --out C:\Users\skand\Desktop\Output.csv -f "_id,Name,USN,Sem,Dept_name,CGPA"

```
C:\Users\skand>mongoexport --db myDB --collection Student
2022-06-06T17:24:46.101+0530    connected to: localhost
2022-06-06T17:24:46.109+0530    exported 5 records
```

	A	B	C	D	E	F
1	_id	Name	USN	Sem	Dept_nam	CGPA
2	1	Ravi	1BM19CS127	6	CSE	8.34
3	2	Balaji	1BM19CS134	6	CSE	8.5
4	3	Skanda	1BM19CS137	6	CSE	8.85
5	4	Nagraj	1BM20CS137	5	CSE	9.25
6	5	Sagar	1BM20CS097	5	ME	7.95

2. To drop a collection by the name “Student”.

db.Student.drop();

```
> db.Student.drop();
true
```

3. Insert the document for “AryanDavid” 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 from “Skating” to “Chess”.) 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:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true})

```
> db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.Student.update({_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
>
```

;

4. FIND METHOD

A. To search for documents from the “Students” collection based on certain search criteria.

db.Student.find({StudName:"Aryan David"});

```
> db.Student.find({StudName:"AryanDavid"});
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

B. 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({},{StudName:1,Grade:1,_id:0});

```
> db.Student.find({},{StudName:1,Grade:1,_id:0});
{ "StudName" : "MichelleJacintha", "Grade" : "VII" }
{ "Grade" : "VII", "StudName" : "AryanDavid" }
```

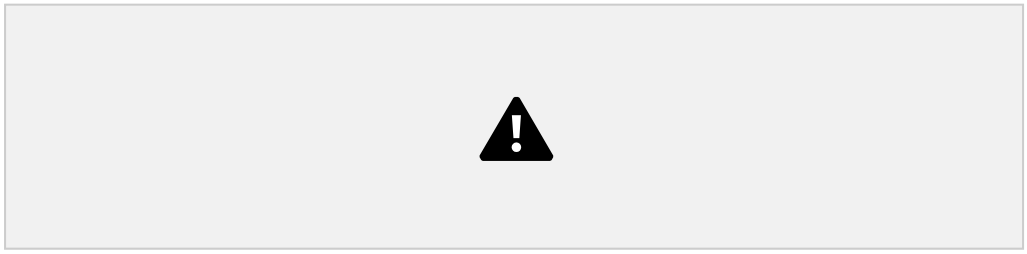
C. To find those documents where the Grade is set to ‘VII’

db.Student.find({Grade:{$eq:'VII'}}).pretty();

```
> db.Student.find({Grade:{$eq:'VII'}}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
>
```

D. 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: ['Chess','Skating']}}).pretty ();



E. To find documents from the Students collection where the StudName begins with "M".

db.Student.find({StudName:/^M/}).pretty();

F. To find documents from the Students collection where the StudName has an "e" in any position.

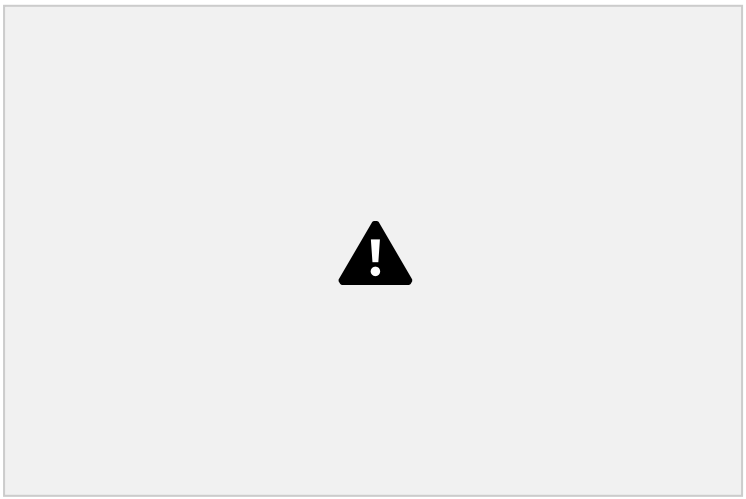
db.Student.find({StudName:/e/}).pretty();

G. To find the number of documents in the Students collection.

db.Student.count();

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

db.Student.find().sort({StudName:-1}).pretty();



I. Save Method :

Save() method will insert a new document, if the document with the _id does not exist. If it exists it will replace the exisiting document.

db.Students.save({StudName:"Vamsi", Grade:"VI"})

II. Add a new field to existing Document:

db.Students.update({_id:3},{$set:{Location:"Network"}})

III. Remove the field in an existing Document

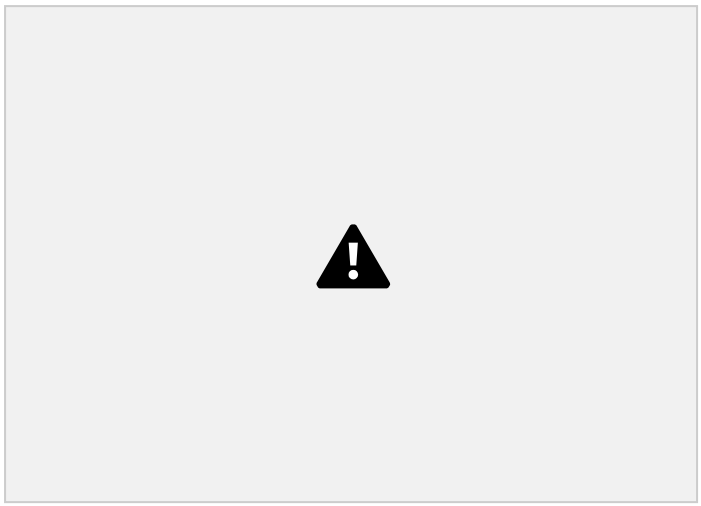
db.Students.update({_id:3},{$unset:{Location:"Network"}})

To set a particular field value to NULL

db.Students.update({_id:3},{$set:{Location:null}})

Sort the document in Ascending order

db.Students.find().sort({StudName:1}).pretty();



Note:
for desending order : db.Students.find().sort({StudName:-1}).pretty();

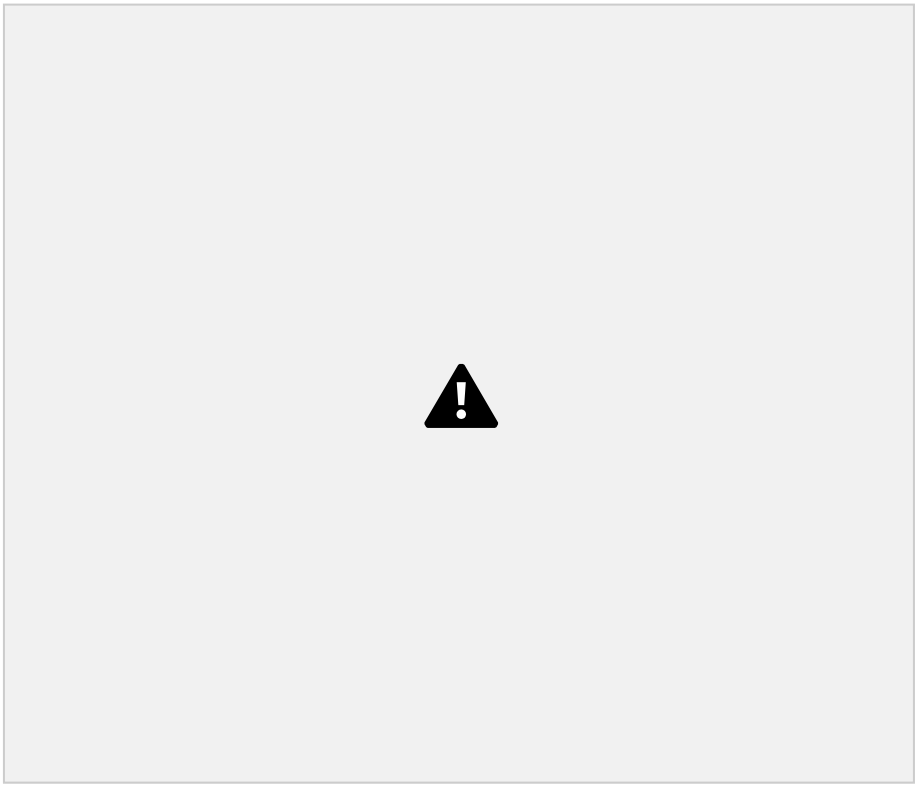
2. Perform the following DB operations using Cassandra. (Employee DB)

1. Create a keyspace by name Employee

create keyspace employee with replication = { 'class':'SimpleStrategy' , 'replication_factor' :1};

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

create table employee_info(emp_id int,emp_name text, designation text, doj timestamp, salary double, dept_name text, primary key(emp_id,salary));

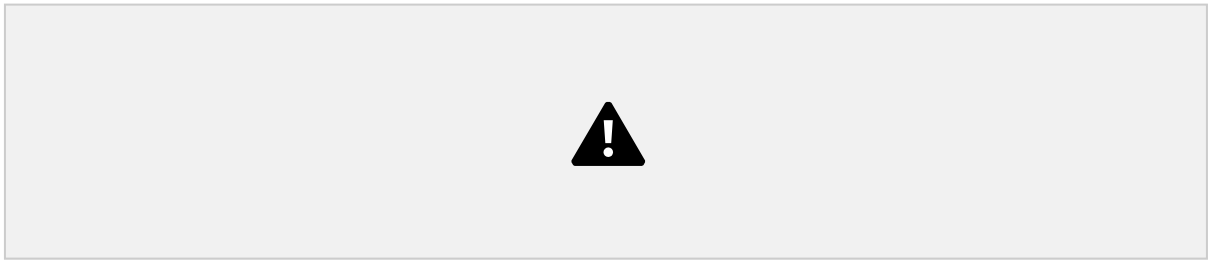


3. Insert the values into the table in batch

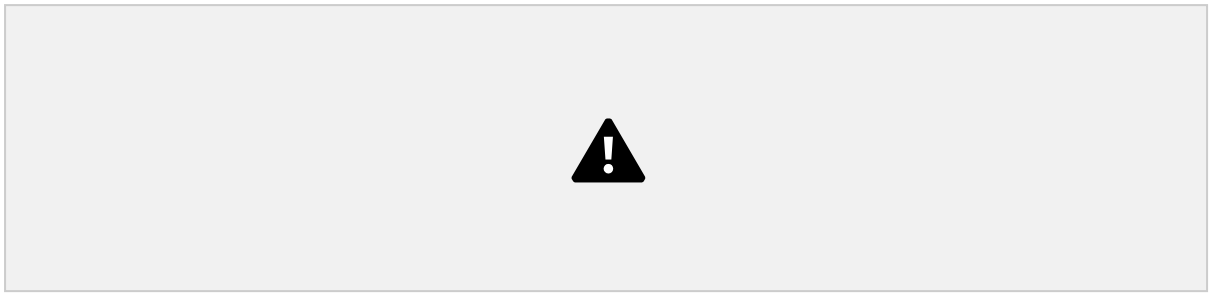
BEGIN BATCH
INSERT INTO employee_info(emp_id, emp_name, designation, doj, salary, dept_name)
VALUES (121, 'Ravi', 'Manager', '2012-03-29', 200000, 'RD')
INSERT INTO employee_info(emp_id, emp_name, designation, doj, salary, dept_name)
VALUES(122, 'David', 'Worker', '2013-02-27', 20000, 'Transport')
APPLY BATCH;

4. Update Employee name and Department of Emp-Id 121

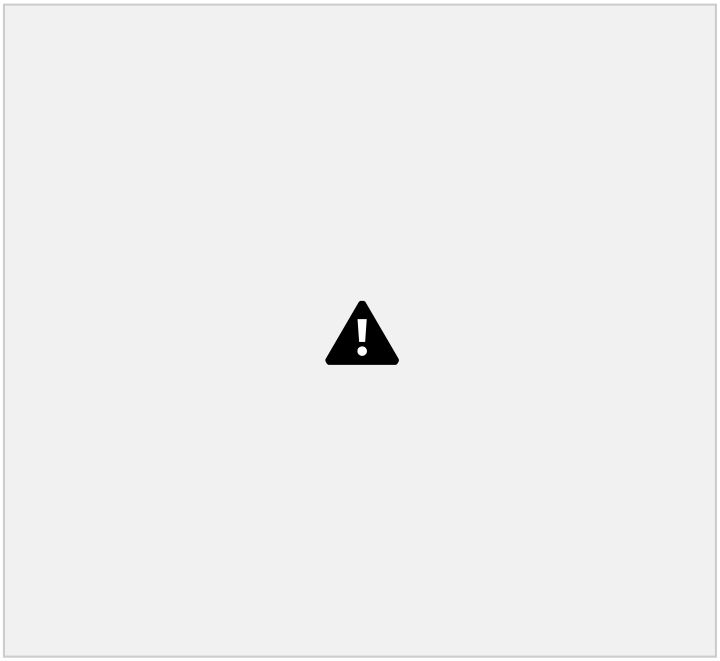
update employee_info set emp_name='Ravi S', dept_name='Research' where emp_id=121 AND salary=200000;



5. Sort the details of Employee records based on salary



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.



7. Update the altered table to add project names.

update employee_info set projects=projects+{'VGST'} where emp_id=121 AND salary=200000;

8. Create a TTL of 15 seconds to display the values of Employee

cqlsh:employee> INSERT INTO employee_info(emp_id, emp_name, designation, doj, salary, dept_name) VALUES(149, 'Saket', 'Developer', '2021-02-20', 100000, 'RD') USING TTL 15;

cqlsh:employee> select ttl(emp_name) from employee_info Where emp_id=149;

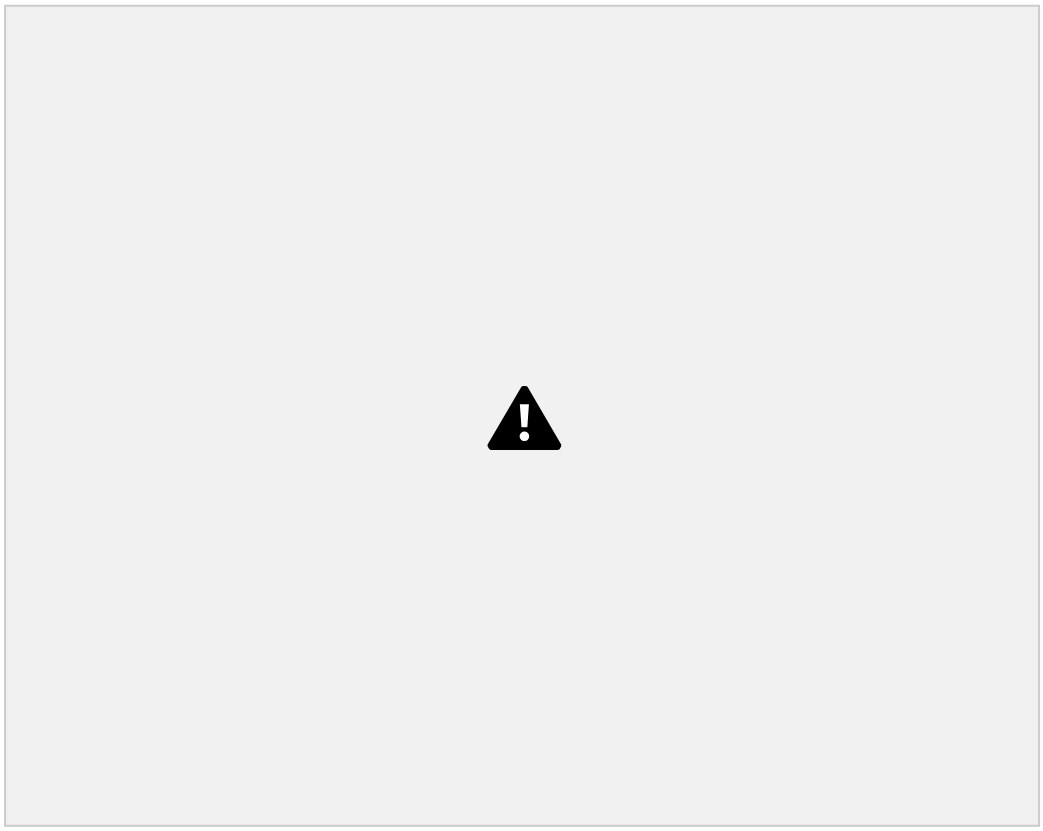
3. Perform the following DB operations using Cassandra. (Library DB)

1. Create a keyspace by name Library

CREATE KEYSPACE Library WITH REPLICATION={'class':'SimpleStrategy','replication_factor':1};

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

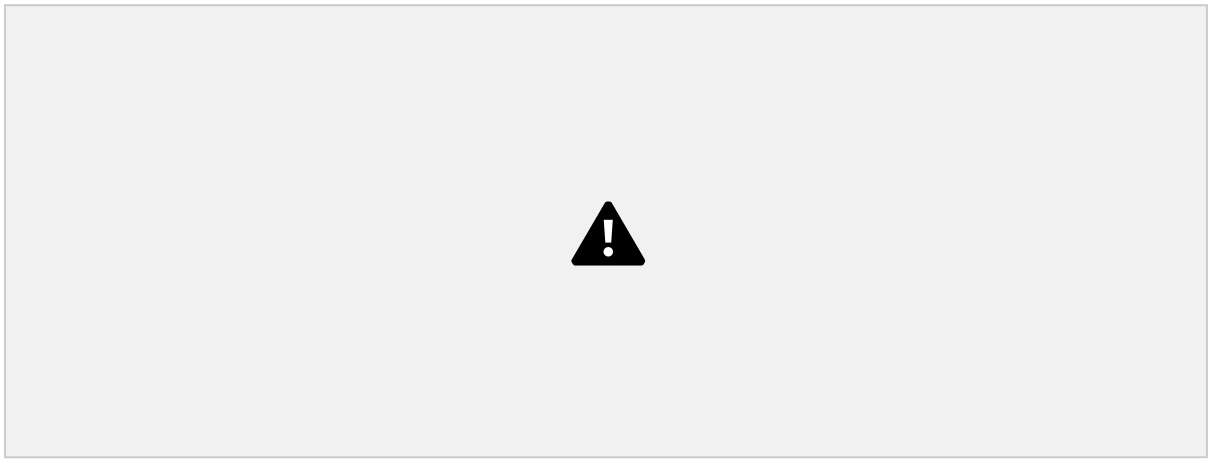
create table library_details(stud_id int,counter_value counter,stud_name text,book_name text,date_of_issue timestamp,book_id int,primary key(stud_id,stud_name,book_name,date_of_issue,book_id));



3. Insert the values into the table in batch

update library_details set counter_value=counter_value+1
where stud_id=111 and stud_name='Ramesh' and book_name='ML' and date_of_issue='2021-11-09' and book_id=200;

update library_details set counter_value=counter_value+1
where stud_id=112 and stud_name='Prabhakar' and book_name='BDA' and date_of_issue='2022-01-01' and book_id=300;
update library_details set counter_value=counter_value+1
where stud_id=113 and stud_name='Gopinath' and book_name='OOMD' and date_of_issue='2021-06-01' and book_id=400;



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

update library_details set counter_value=counter_value+1
where stud_id=112 and stud_name='Prabhakar' and book_name='BDA' and date_of_issue='2021-12-31' and book_id=300;

5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.

select * from library_details where stud_id=112;

6. Export the created column to a csv file

copy library_details(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) to 'library.csv' ;

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

copy library_details(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) from 'library.csv'
;