Create KeySpace :

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

Describe the existing Keyspaces:

DESCRIBE KEYSPACES;

For More details on existing keyspaces:

SELECT * FROM system.schema_keyspaces;

use the keyspace "Students":

USE Student2;

```
cqlsh> CREATE KEYSPACE Student2 WITH REPLICATION = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES;

employee  students   system_auth         system_traces
student   students1  system_distributed  system_views
student2  system     system_schema       system_virtual_schema

cqlsh> USE Student2;
```

To create table (column family) by name Student_Info:

CREATE TABLE Students_Info (Roll_No int PRIMARY KEY, StudName text,
DateOfJoining timestamp, last_exam_Percent double);

```
cqlsh:student2> CREATE TABLE Students_Info (Roll_No int PRIMARY KEY, StudName text, DateOfJoining timestamp, last_exam_Percent double);
```

Lookup the names of all tables in the current keyspaces
DESCRIBE TABLES;

```
cqlsh:student2> DESCRIBE TABLES;

students_info
```

Describe the table information

DESCRIBE TABLE STUDENTS_INFO;

```
CREATE TABLE student2.students_info (
    roll_no int PRIMARY KEY,
    dateofjoining timestamp,
    last_exam_percent double,
    studname text
) 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 memtable = 'default'
    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';
```

CRUD

Insert :

BEGIN BATCH
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent)
VALUES (1,'Asha','2012-03-12',79.9)
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent)
VALUES (2,'Krian','2012-03-12',89.9)
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent)
VALUES (3,'Tarun','2012-03-12',78.9)
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent)
VALUES (4,'Samrth','2012-03-12',90.9)
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent)
VALUES (5,'Smitha','2012-03-12',67.9)
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent)
VALUES (6,'Rohan','2012-03-12',56.9)
APPLY BATCH;

View data from the table “Students_Info”

SELECT * FROM Students_Info;

```
cqlsh:student2> BEGIN BATCH
           ... INSERT INTO Students_Info(Roll_No,  StudName, DateOfJoining,  last_exam_Percent)
           ... VALUES (1,'Asha','2012-03-12',79.9)
           ... INSERT INTO Students_Info(Roll_No,  StudName, DateOfJoining,  last_exam_Percent)
           ... VALUES (2,'Krian','2012-03-12',89.9)
           ... INSERT INTO Students_Info(Roll_No,  StudName, DateOfJoining,  last_exam_Percent)
           ... VALUES (3,'Tarun','2012-03-12',78.9)
           ... INSERT INTO Students_Info(Roll_No,  StudName, DateOfJoining,  last_exam_Percent)
           ... VALUES (4,'Samrth','2012-03-12',90.9)
           ... INSERT INTO Students_Info(Roll_No,  StudName, DateOfJoining,  last_exam_Percent)
           ... VALUES (5,'Smitha','2012-03-12',67.9)
           ... INSERT INTO Students_Info(Roll_No,  StudName, DateOfJoining,  last_exam_Percent)
           ... VALUES (6,'Rohan','2012-03-12',56.9)
           ... APPLY BATCH;
cqlsh:student2> SELECT * FROM  Students_Info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       5 | 2012-03-11 18:30:00.000000+0000 |              67.9 |   Smitha
       1 | 2012-03-11 18:30:00.000000+0000 |              79.9 |    Asha
       2 | 2012-03-11 18:30:00.000000+0000 |              89.9 |   Krian
       4 | 2012-03-11 18:30:00.000000+0000 |              90.9 |   Samrth
       6 | 2012-03-11 18:30:00.000000+0000 |              56.9 |   Rohan
       3 | 2012-03-11 18:30:00.000000+0000 |              78.9 |   Tarun

(6 rows)
```

View data from the table "Students_Info" where RoolNo column either has a value 1 or 2 or 3

SELECT * FROM Students_Info WHERE Roll_No IN (1,2,3);

```
cqlsh:student2> SELECT * FROM  Students_Info WHERE Roll_No IN (1,2,3);

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2012-03-11 18:30:00.000000+0000 |              79.9 |    Asha
       2 | 2012-03-11 18:30:00.000000+0000 |              89.9 |   Krian
       3 | 2012-03-11 18:30:00.000000+0000 |              78.9 |   Tarun

(3 rows)
```

To execute a non primary key - will throw an error

select * from students_info where Studname= 'Asha';

```
cqlsh:student2> select * from students_info where Studname= 'Asha';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
```

So create an INDEX on the Column as below:
To create an INDEX on StudName Column of the Students_Info column family

CREATE INDEX ON Students_Info (StudName);

Now execute the query based on the INDEXED Column:

select * from students_info where Studname= 'Asha';

```
cqlsh:student2> CREATE INDEX ON  Students_Info ( StudName);
cqlsh:student2> select * from students_info where Studname= 'Asha';

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2012-03-11 18:30:00.000000+0000 |              79.9 |    Asha

(1 rows)
```

To specify the number of rows retured in the output
select Roll_No, StudName from students_info LIMIT 2;

```
cqlsh:student2> select Roll_No, StudName from students_info LIMIT 2;

 roll_no | studname
---------+----------
       5 |   Smitha
       1 |     Asha

(2 rows)
```

Alias for Column:

Select Roll_No as “USN” from Students_info;

```
cqlsh:student2> Select Roll_No as "USN" FROM Students_info;

 USN
-----
   5
   1
   2
   4
   6
   3
```

UPDATE

UPDATE students_info SET StudName='David Sheen' WHERE Roll_No=2;

```
cqlsh:student2> UPDATE students_info SET StudName='David Sheen' WHERE Roll_No=2;
```

Lets try to update the primary key

UPDATE students_info SET roll_no=6 WHERE roll_no=3;

```
cqlsh:student2> UPDATE students_info SET roll_no=6 WHERE roll_no=3;
InvalidRequest: Error from server: code=2200 [Invalid query] message="PRIMARY KEY part roll_no found in SET part"
```

DELETE
DELETE LastExamPercent FROM students_info WHERE Roll_No=2;

```
cqlsh:student2> DELETE LastExamPercent FROM students_info WHERE RollNo=2;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Undefined column name lastexampercent in table student2.students_info"
```

Delete a Row
DELETE FROM student_info WHERE RollNo=2;

```
cqlsh:student2> DELETE FROM students_info WHERE Roll_No=2;
cqlsh:student2> SELECT * FROM  Students_Info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       5 | 2012-03-11 18:30:00.000000+0000 |              67.9 |   Smitha
       1 | 2012-03-11 18:30:00.000000+0000 |              79.9 |     Asha
       4 | 2012-03-11 18:30:00.000000+0000 |              90.9 |   Samrth
       6 | 2012-03-11 18:30:00.000000+0000 |              56.9 |    Rohan
       3 | 2012-03-11 18:30:00.000000+0000 |              78.9 |    Tarun

(5 rows)
```