

Mahatma Education Society's
Pillai College of Arts, Commerce & Science

(Autonomous)
Affiliated to University of Mumbai
NAAC Accredited 'A' grade (3 cycles) Best
College Award by University of Mumbai
ISO 9001:2015 Certified



CERTIFICATE

*This is to certify that Miss. Rohit Malviya of M.Sc. D.A.
Part-I Semester I has completed the practical work in the Subject of
Next Generation Databases during the academic year 2024-25
under the guidance of Prof. Omakar Sherkhane being the
partial requirement for the fulfillment of the curriculum of **Degree of
Master of Science in Data Analytics, University of Mumbai.***

Place:

Date:

Name & Signature of faculty

Name & Signature of external

Index

Practical No.	Details	Signature
1.	<p>MongoDB Basics & Implementing Aggregation</p> <ol style="list-style-type: none"> 1) Write a MongoDB query to create and drop databases. 2) Write a MongoDB query to create, display and drop collection. 3) Write a MongoDB query to insert, query, update and delete a document. 4) Write a MongoDB query to use sum, avg, min and max expressions. 5) Write a MongoDB query to use push and addToSet expressions. 6) Write a MongoDB query to use the first and last expression. 	
2.	<p>Replication, Backup and Restore</p> <ol style="list-style-type: none"> 1) Write a MongoDB query to create Replica of existing database. 2) Write a MongoDB query to create a backup of an existing database. 3) Write a MongoDB query to restore the database from the backup. 	
3.	<p>Java/Python/R and MongoDB</p> <ol style="list-style-type: none"> 1) Connecting Java/Python/R with MongoDB and performing inserting, retrieving, updating and deleting. 	
4.	<p>Create Data Model using Cassandra</p> <ol style="list-style-type: none"> 1) Cassandra Keyspace Operations. 2) Cassandra Table Operations. 3) Cassandra CRUD Operations. 4) Cassandra CQL Types 	
5.	<p>Java/Python/R and Cassandra</p> <ol style="list-style-type: none"> 1) Connecting Java with Cassandra and performing inserting, retrieving, updating and deleting. 	
6.	<p>PostgreSQL</p> <ol style="list-style-type: none"> 1) Table Operations (CRUD) 2) Aggregation Functions 3) Join & Sub query Operations 4) String Functions 	
7.	<p>Java/Python/R and PostgreSQL</p> <p>Connect Java with PostgreSQL and perform inserting, retrieving, updating and deleting.</p>	
8.	<p>CouchDB</p> <ol style="list-style-type: none"> 1) Working with Database 	

	2) Working with Document 3) Attaching file	
9.	Java/Python/R and CoachDB Connect Java with CoachDB and perform inserting, retrieving, updating and deleting.	
10.	Neo4j 1) Neo4j CQL 2) Neo4j CQL Write Clause , Read Clause & General Clause 3) Neo4j String & Aggregate functions 4) CQL Admin (Backup,Restore,index)	

SQL Queries

```
mysql> CREATE DATABASE College;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> Use College  
Database changed  
mysql>
```

Create table cust_24

```
mysql> CREATE TABLE cust_24 (  
->   custid_id VARCHAR(5),  
->   lname VARCHAR(15),  
->   fname VARCHAR(15),  
->   area VARCHAR(10),  
->   phone_no VARCHAR(10),  
->   PRIMARY KEY (custid_id)  
-> );  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> insert into cust_24 values('a101','Agarwal','Rachna','Panvel',9975597675);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> insert into cust_24 values('a102','Hamdule','Anam','Panvel',9978654321);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> insert into cust_24 values('a103','More','Shreya','Panvel',9898654321);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> insert into cust_24 values('a104','Bhattacharjee','Shreya','Panvel',9898654541);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> insert into cust_24 values('a105','Gorde','Priyanka','Panvel',8098654541);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> select * from cust_24;  
+-----+-----+-----+-----+-----+  
| custid_id | lname      | fname  | area  | phone_no |  
+-----+-----+-----+-----+-----+  
| a101      | Agarwal    | Rachna | Panvel | 9975597675 |  
| a102      | Hamdule    | Anam   | Panvel | 9978654321 |  
| a103      | More       | Shreya | Panvel | 9898654321 |  
| a104      | Bhattacharjee | Shreya | Panvel | 9898654541 |  
| a105      | Gorde      | Priyanka | Panvel | 8098654541 |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

creating table movie_24

```
mysql> CREATE TABLE movie_24 (  
->   movie_no INT(5),  
->   title VARCHAR(25),  
->   type VARCHAR(10),  
->   star VARCHAR(25),  
->   price DECIMAL(8, 2), -- Assuming the price is a decimal number  
->   PRIMARY KEY (movie_no)  
-> );  
Query OK, 0 rows affected, 1 warning (0.02 sec)  
  
mysql> select * from movie_24;  
Empty set (0.00 sec)  
  
mysql> INSERT INTO movie_24 (movie_no, title, type, star, price) VALUES (1, 'Krrish', 'Action', 'Hritik', 150);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> INSERT INTO movie_24 (movie_no, title, type, star, price) VALUES (2, 'Mohabbatein', 'Romantic', 'Shahrukh', 100);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> INSERT INTO movie_24 (movie_no, title, type, star, price) VALUES (3, 'Hera Pheri', 'Comedy', 'Akshay', 50);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> INSERT INTO movie_24 (movie_no, title, type, star, price) VALUES (4, 'Baghban', 'Family', 'Amitabh', 80);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> INSERT INTO movie_24 (movie_no, title, type, star, price) VALUES (5, 'Vastushastra', 'Thriller', 'Rajpal', 70);  
Query OK, 1 row affected (0.00 sec)  
  
mysql> select * from movie_24;  
+-----+-----+-----+-----+-----+  
| movie_no | title      | type   | star   | price |  
+-----+-----+-----+-----+-----+  
| 1 | Krrish      | Action | Hritik | 150.00 |  
| 2 | Mohabbatein | Romantic | Shahrukh | 100.00 |  
| 3 | Hera Pheri  | Comedy | Akshay | 50.00  |  
| 4 | Baghban     | Family | Amitabh | 80.00  |  
| 5 | Vastushastra | Thriller | Rajpal | 70.00  |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

creating table invoice_24

```
mysql> CREATE TABLE invoice_24 (  
->   inv_no VARCHAR(5),  
->   movie_no INT(5),  
->   cust_id VARCHAR(5),  
->   issue_date DATE,  
->   return_date DATE,  
->   PRIMARY KEY (inv_no),  
->   FOREIGN KEY (movie_no) REFERENCES movie_24(movie_no),  
->   FOREIGN KEY (cust_id) REFERENCES cust_24(custid_id)  
-> );  
Query OK, 0 rows affected, 1 warning (0.01 sec)  
  
mysql> select * from invoice_24  
-> ;  
Empty set (0.00 sec)
```

```

mysql> INSERT INTO invoice_24 (inv_no, movie_no, cust_id, issue_date, return_date) VALUES ('i01', 1, 'a101', DATE '2007-05-02', DATE '2007-05-04');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO invoice_24 (inv_no, movie_no, cust_id, issue_date, return_date) VALUES ('i02', 2, 'a102', DATE '2007-05-03', DATE '2007-05-05');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO invoice_24 (inv_no, movie_no, cust_id, issue_date, return_date) VALUES ('i03', 3, 'a103', DATE '2007-05-02', DATE '2007-05-05');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO invoice_24 (inv_no, movie_no, cust_id, issue_date, return_date) VALUES ('i04', 3, 'a104', DATE '2007-05-09', DATE '2007-05-11');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO invoice_24 (inv_no, movie_no, cust_id, issue_date, return_date) VALUES ('i05', 4, 'a105', DATE '2007-05-08', DATE '2007-05-10');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO invoice_24 (inv_no, movie_no, cust_id, issue_date, return_date) VALUES ('i06', 4, 'a106', DATE '2007-05-09', DATE '2007-05-12');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO invoice_24 (inv_no, movie_no, cust_id, issue_date, return_date) VALUES ('i07', 5, 'a104', DATE '2007-05-12', DATE '2007-05-15');
Query OK, 1 row affected (0.00 sec)

mysql> select * from invoice_24;
+-----+-----+-----+-----+-----+
| inv_no | movie_no | cust_id | issue_date | return_date |
+-----+-----+-----+-----+-----+
| i01    | 1        | a101    | 2007-05-02 | 2007-05-04 |
| i02    | 2        | a102    | 2007-05-03 | 2007-05-05 |
| i03    | 3        | a103    | 2007-05-02 | 2007-05-05 |
| i04    | 3        | a104    | 2007-05-09 | 2007-05-11 |
| i05    | 4        | a105    | 2007-05-08 | 2007-05-10 |
| i06    | 4        | a106    | 2007-05-09 | 2007-05-12 |
| i07    | 5        | a104    | 2007-05-12 | 2007-05-15 |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

A.Single table retrieval

1.Find out the names of all customers.

```

mysql> select fname, lname from cust_24;
+-----+-----+
| fname | lname |
+-----+-----+
| Rachna | Agarwal |
| Anam   | Hamdule |
| Shreya | More   |
| Shreya | Bhattacharjee |
| Priyanka | Gorde |
+-----+-----+
5 rows in set (0.00 sec)

```

2.Print the entire customer table.

```

mysql> select * from cust_24;
+-----+-----+-----+-----+-----+
| custid_id | lname | fname | area | phone_no |
+-----+-----+-----+-----+-----+
| a101      | Agarwal | Rachna | Panvel | 9975597675 |
| a102      | Hamdule | Anam   | Panvel | 9978654321 |
| a103      | More   | Shreya | Panvel | 9898654321 |
| a104      | Bhattacharjee | Shreya | Panvel | 9898654541 |
| a105      | Gorde  | Priyanka | Panvel | 8098654541 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

3.Retrieve the list of fname and area of all customers.

```
mysql> select fname,area from cust_24;
+-----+-----+
| fname   | area   |
+-----+-----+
| Rachna  | Panvel |
| Anam    | Panvel |
| Shreya  | Panvel |
| Shreya  | Panvel |
| Priyanka | Panvel |
+-----+-----+
5 rows in set (0.00 sec)
```

4. List the various movie types available from the movie table

```
mysql> select type from movie_24;
+-----+
| type   |
+-----+
| Action |
| Romantic |
| Comedy |
| Family |
| Thriller |
+-----+
5 rows in set (0.00 sec)
```

5. Find the names of customers having 'a' as the second letter in their fnames.

```
mysql> select * from cust_24 where fname like '_a%';
+-----+-----+-----+-----+-----+
| custid_id | lname   | fname   | area   | phone_no |
+-----+-----+-----+-----+-----+
| a101      | Agarwal | Rachna  | Panvel | 9975597675 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

6. find the lnames of all customers that begin with 'b' or 'g'.

```
mysql> select lname from cust_24 where lname like 'b%' or lname like '%g';
+-----+
| lname           |
+-----+
| Bhattacharjee   |
+-----+
1 row in set (0.00 sec)
```

7. Find out the customers who stay in the area whose second letter is 'a'.

```
mysql> select * from cust_24 where area like '_a%';
```

custid_id	lname	fname	area	phone_no
a101	Agarwal	Rachna	Panvel	9975597675
a102	Hamdule	Anam	Panvel	9978654321
a103	More	Shreya	Panvel	9898654321
a104	Bhattacharjee	Shreya	Panvel	9898654541
a105	Gorde	Priyanka	Panvel	8098654541

```
5 rows in set (0.00 sec)
```

8. Find the list of all customers who stay in the area 'nashik' or area 'jalgaon' or area 'pune'.

```
mysql> select * from cust_24 where area in ('nashik','jalgaon','pune');
Empty set (0.00 sec)
```

9. Find out the customers whose phone_no start with '99'.

```
mysql> select * from cust_24 where phone_no like '99%';
```

custid_id	lname	fname	area	phone_no
a101	Agarwal	Rachna	Panvel	9975597675
a102	Hamdule	Anam	Panvel	9978654321

10. Print the information from the invoice table of customers who have been issued movies in the month of September.

```
mysql> select * from invoice_24 where month(issue_date)=09;
Empty set (0.00 sec)
```

11. Display the invoice table information for cust_id 'a101','a102'.

```
mysql> select * from invoice_24 where cust_id in ('a101','a102')
-> ;
```

inv_no	movie_no	cust_id	issue_date	return_date
i01	1	a101	2007-05-02	2007-05-04
i02	2	a102	2007-05-03	2007-05-05

12. Find the movies of type 'action','comedy'.

```
mysql> select * from movie_24 where type in ('action','comdedy');
+-----+-----+-----+-----+-----+
| movie_no | title | type | star | price |
+-----+-----+-----+-----+-----+
| 1 | Krrish | Action | Hritik | 150.00 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

13. Find the movies whose price is greater than 150 and less than or equal to 200.

```
mysql> select * from movie_24 where price>100 and price<=200;
+-----+-----+-----+-----+-----+
| movie_no | title | type | star | price |
+-----+-----+-----+-----+-----+
| 1 | Krrish | Action | Hritik | 150.00 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

14. Find the movies that cost more than 150 and also find new cost as original cost*15.

15. Rename the new column in the above query as new_price.

```
mysql> select *,price*15 New_Price from movie_24 where price>150;
Empty set (0.00 sec)

mysql> select * from movie_24
-> ;
+-----+-----+-----+-----+-----+
| movie_no | title | type | star | price |
+-----+-----+-----+-----+-----+
| 1 | Krrish | Action | Hritik | 150.00 |
| 2 | Mohabbatein | Romantic | Shahrukh | 100.00 |
| 3 | Hera Pheri | Comedy | Akshay | 50.00 |
| 4 | Baghban | Family | Amitabh | 80.00 |
| 5 | Vastushastra | Thriller | Rajpal | 70.00 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> INSERT INTO movie_24 (movie_no, title, type, star, price) VALUES (8, 'Titanic', 'Romantic', 'Caprio', 200);
Query OK, 1 row affected (0.00 sec)

mysql> select *,price*15 New_Price from movie_24 where price>150;
+-----+-----+-----+-----+-----+-----+
| movie_no | title | type | star | price | New_Price |
+-----+-----+-----+-----+-----+-----+
| 8 | Titanic | Romantic | Caprio | 200.00 | 3000.00 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

16. List the movies in sorted order of titles.

```
mysql> select * from movie_24 order by title;
```

movie_no	title	type	star	price
4	Baghban	Family	Amitabh	80.00
3	Hera Pheri	Comedy	Akshay	50.00
1	Krrish	Action	Hritik	150.00
2	Mohabbatein	Romantic	Shahrukh	100.00
8	Titanic	Romantic	Caprio	200.00
5	Vastushastra	Thriller	Rajpal	70.00

```
6 rows in set (0.00 sec)
```

17. Print the names and types of all the movies except 'Historical' movies.

```
mysql> select * from movie_24 where type<>'Historical';
```

movie_no	title	type	star	price
1	Krrish	Action	Hritik	150.00
2	Mohabbatein	Romantic	Shahrukh	100.00
3	Hera Pheri	Comedy	Akshay	50.00
4	Baghban	Family	Amitabh	80.00
5	Vastushastra	Thriller	Rajpal	70.00
8	Titanic	Romantic	Caprio	200.00

18. Calculate square root of price of each movie.

```
mysql> select title,sqrt(price) from movie_24;
```

title	sqrt(price)
Krrish	12.24744871391589
Mohabbatein	10
Hera Pheri	7.0710678118654755
Baghban	8.94427190999916
Vastushastra	8.366600265340756
Titanic	14.142135623730951

19. Divide the cost of movie 'Titanic' by difference between its price and 100.

```
mysql> select price, price/(price-100) from movie_24 where title='titanic';
```

price	price/(price-100)
200.00	2.000000

20. List the names, areas and cust_id of customers without phone numbers.

```
mysql> select * from cust_24 where phone_no is null;
Empty set (0.00 sec)
```

21. List the names of customers without lnames.

```
mysql> select * from cust_24 where lname is null;
Empty set (0.00 sec)
```

22. List the movie_no, title, type of movies whose stars begin with letter 'A'.

```
mysql> select movie_no, title, type, star from movie_24 where star like 'a%';
+-----+-----+-----+-----+
| movie_no | title      | type   | star    |
+-----+-----+-----+-----+
|          3 | Hera Pheri | Comedy | Akshay  |
|          4 | Baghban    | Family | Amitabh |
+-----+-----+-----+-----+
```

23. List the movie_no and inv_no of customers having inv_no less than 'i05', from the invoice transaction table.

```
mysql> select movie_no, inv_no from invoice_24 where inv_no < 'i05';
+-----+-----+
| movie_no | inv_no |
+-----+-----+
|          1 | i01    |
|          2 | i02    |
|          3 | i03    |
|          3 | i04    |
+-----+-----+
```

B.Functions and concatenation:-

24. Count the total number of customers.

```
mysql> select count(*) from cust_24;
+-----+
| count(*) |
+-----+
|          5 |
+-----+
```

25. Calculate the total price of all the movies.

```
mysql> select sum(price) from movie_24;
+-----+
| sum(price) |
+-----+
|      650.00 |
+-----+
```

26. Calculate average price of all movies.

```
mysql> select avg(price) from movie_24;
+-----+
| avg(price) |
+-----+
| 108.333333 |
+-----+
```

27. Determine the maximum and minimum movie price. Rename title as max(price) and min(price).

```
mysql> select max(price) max_price,min(price) min_price from movie_24;
+-----+-----+
| max_price | min_price |
+-----+-----+
|      200.00 |      50.00 |
+-----+-----+
```

28. Count the number of movies having price greater than equal to 150.

```
mysql> select * from movie_24 where price>=150;
+-----+-----+-----+-----+-----+
| movie_no | title   | type    | star   | price |
+-----+-----+-----+-----+-----+
|          1 | Krrish  | Action  | Hritik | 150.00 |
|          8 | Titanic | Romantic | Caprio | 200.00 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

29. Print the information of invoice_24 table in the following format for all the records.

A. The invoice number of customer ID {cust_id} is {inv_no} and movie number {movie_no}.

B. {cust_id} has taken the movie number {movie_no} on {issue_date} and will return on {return_date}.

```
mysql> select concat('The invoice no of customer id ',cust_id,' is', inv_no, ' and movie number is ', movie_no) from invoice_24;
+-----+
| concat('The invoice no of customer id ',cust_id,' is', inv_no, ' and movie number is ', movie_no) |
+-----+
| The invoice no of customer id a101 is101 and movie number is 1 |
| The invoice no of customer id a102 is102 and movie number is 2 |
| The invoice no of customer id a103 is103 and movie number is 3 |
| The invoice no of customer id a104 is104 and movie number is 3 |
| The invoice no of customer id a105 is105 and movie number is 4 |
| The invoice no of customer id a109 is106 and movie number is 4 |
| The invoice no of customer id a104 is107 and movie number is 5 |
+-----+
7 rows in set (0.00 sec)

mysql> select concat(cust_id, ' has taken the movie number ', movie_no, ' no ', issue_date) from invoice_24;
+-----+
| concat(cust_id, ' has taken the movie number ', movie_no, ' no ', issue_date) |
+-----+
| a101 has taken the movie number 1 no 2007-05-02 |
| a102 has taken the movie number 2 no 2007-05-03 |
| a103 has taken the movie number 3 no 2007-05-02 |
| a104 has taken the movie number 3 no 2007-05-09 |
| a105 has taken the movie number 4 no 2007-05-08 |
| a109 has taken the movie number 4 no 2007-05-09 |
| a104 has taken the movie number 5 no 2007-05-12 |
+-----+
7 rows in set (0.00 sec)
```

c. Having and group by clause:

30. Find the average price of each movie.

```
mysql> select avg(price) from movie_24;
+-----+
| avg(price) |
+-----+
| 108.333333 |
+-----+
```

31. Find the number of movies in each type.

```
mysql> select type, count(title) from movie_24 group by type;
+-----+-----+
| type      | count(title) |
+-----+-----+
| Action    | 1 |
| Romantic  | 2 |
| Comedy    | 1 |
| Family    | 1 |
| Thriller  | 1 |
+-----+-----+
```

32. Count separately the number of movies in 'Comedy', 'Thriller' types.

```
mysql> select type, count(title) from movie_24 where type in ('Comedy','Thriller') group by type;
+-----+-----+
| type      | count(title) |
+-----+-----+
| Comedy    | 1 |
| Thriller  | 1 |
+-----+-----+
2 rows in set (0.00 sec)
```

33. Calculate the average price for each type that has a maximum price of 100.

```
mysql> select avg(price), type from movie_24 group by type having max(price)=100;
Empty set (0.00 sec)
```

34. Calculate the average price of all movies where type is 'Romantic','Action' and price is greater than 100.

```
mysql> select avg(price), type from movie_24 where type in ('Romantic','Action') and price>100 group by type;
+-----+-----+
| avg(price) | type |
+-----+-----+
| 150.000000 | Action |
| 200.000000 | Romantic |
+-----+-----+
```

D. Joins and correlations:

35. Find out movie_no which has been issued to 'vinaya'.

```
mysql> select movie_no from invoice_24 i
-> inner join cust_24 c on i.cust_id=c.custid_id
-> where fname='vinaya';
Empty set (0.00 sec)
```

```
mysql> select movie_no from invoice_24 i
-> inner join cust_24 c on i.cust_id=c.custid_id
-> where fname='rachna';
```

```
+-----+
| movie_no |
+-----+
| 1 |
+-----+
```

36. Find the names and movie_no of all customers who have been issued a movie.

```
mysql> select * from invoice_24 i
-> inner join cust_24 c on i.cust_id=c.custid_id;
```

inv_no	movie_no	cust_id	issue_date	return_date	custid_id	lname	fname	area	phone_no
i01	1	a101	2007-05-02	2007-05-04	a101	Agarwal	Rachna	Panvel	9975597675
i02	2	a102	2007-05-03	2007-05-05	a102	Hamdule	Anam	Panvel	9978654321
i03	3	a103	2007-05-02	2007-05-05	a103	More	Shreya	Panvel	9898654321
i04	3	a104	2007-05-09	2007-05-11	a104	Bhattacharjee	Shreya	Panvel	9898654541
i05	4	a105	2007-05-08	2007-05-10	a105	Gorde	Priyanka	Panvel	8098654541
i07	5	a104	2007-05-12	2007-05-15	a104	Bhattacharjee	Shreya	Panvel	9898654541

37. Select the title,cust_id,movie_no for all movies that are issued.

```
mysql> select title,cust_id,i.movie_no from invoice_24 i
-> inner join movie_24 m on i.movie_no=m.movie_no;
```

```
+-----+-----+-----+
| title | cust_id | movie_no |
+-----+-----+-----+
| Krrish | a101 | 1 |
| Mohabbatein | a102 | 2 |
| Hera Pheri | a103 | 3 |
| Hera Pheri | a104 | 3 |
| Baghban | a105 | 4 |
| Baghban | a109 | 4 |
| Vastushastra | a104 | 5 |
+-----+-----+-----+
```


38. Find the title and types of movies that have been issued to 'aayesha'.

```
mysql> select title,type,fname from invoice_24 i
-> inner join movie_24 m on i.movie_no=m.movie_no
-> inner join cust_24 c on i.cust_id=c.custid_id
-> where fname='ayesha';
Empty set (0.01 sec)

mysql> select title,type,fname from invoice_24 i
-> inner join movie_24 m on i.movie_no=m.movie_no
-> inner join cust_24 c on i.cust_id=c.custid_id
-> where fname='anam';
+-----+-----+-----+
| title      | type      | fname |
+-----+-----+-----+
| Mohabbatein | Romantic | Anam  |
+-----+-----+-----+
1 row in set (0.00 sec)
```

39. Find the names of all customers who have been issued movie of type 'Romantic'.

```
mysql> select fname,lname from invoice_24 i
-> inner join movie_24 m on i.movie_no=m.movie_no
-> inner join cust_24 c on i.cust_id=c.custid_id
-> where type='Romantic';
+-----+-----+
| fname | lname |
+-----+-----+
| Anam  | Hamdule |
+-----+-----+
1 row in set (0.00 sec)
```

40. Display the title,lname,fname for customers having movie number greater than or equal to three in the following format: The movie taken by {fname} {lname} is {title}.

```
mysql> select concat('The movie taken by ',fname, ' ', lname, ' is ',title) from invoice_24 i
-> inner join movie_24 m on i.movie_no=m.movie_no
-> inner join cust_24 c on i.cust_id=c.custid_id
-> where m.movie_no>=3;
+-----+
| concat('The movie taken by ',fname, ' ', lname, ' is ',title) |
+-----+
| The movie taken by Shreya More is Hera Pheri |
| The movie taken by Shreya Bhattacharjee is Hera Pheri |
| The movie taken by Priyanka Gorde is Baghban |
| The movie taken by Shreya Bhattacharjee is Vastushastra |
+-----+
4 rows in set (0.00 sec)
```

E. Nested Queries:

41. Find out which customers have been issued movie_no 9.

```
mysql> select * from cust_24 where custid_id in (select cust_id from invoice_24 where movie_no=9);
Empty set (0.00 sec)

mysql> select * from cust_24 where custid_id in (select cust_id from invoice_24 where movie_no=5);
+-----+-----+-----+-----+-----+
| custid_id | lname      | fname | area  | phone_no |
+-----+-----+-----+-----+-----+
| a104      | Bhattacharjee | Shreya | Panvel | 9898654541 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

42. Find the customer name and area with invoice number 'i10'.

```
mysql> select fname, lname from cust_24 where custid_id in (select cust_id from invoice_24 where inv_no='i10');
Empty set (0.00 sec)

mysql> select fname, lname from cust_24 where custid_id in (select cust_id from invoice_24 where inv_no='i07');
+-----+-----+
| fname | lname |
+-----+-----+
| Shreya | Bhattacharjee |
+-----+-----+
1 row in set (0.00 sec)
```

43. Find the customer names and phone numbers who have been issued movies before the month of August.

```
mysql> select fname, lname, phone_no from cust_24 where custid_id in (select cust_id from invoice_24 where month(issue_date)=8);
Empty set (0.00 sec)

mysql> select fname, lname, phone_no from cust_24 where custid_id in (select cust_id from invoice_24 where month(issue_date)=5);
+-----+-----+-----+
| fname | lname      | phone_no |
+-----+-----+-----+
| Rachna | Agarwal    | 9975597675 |
| Anam   | Handule    | 9978654321 |
| Shreya | More       | 9898654321 |
| Shreya | Bhattacharjee | 9898654541 |
| Priyanka | Gorde     | 8898654541 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

44. Find the name of the movie issued to 'anita', 'aayesha'.

```
mysql> select title from movie_24 where movie_no in (select movie_no from invoice_24
-> where cust_id in (select custid_id from cust_24 where fname in ('shreya','priyanka')));
+-----+
| title |
+-----+
| Hera Pheri |
| Baghban |
| Vastushastra |
+-----+
3 rows in set (0.00 sec)
```

45. List the movie numbers, movie names issued to all customers.


```
mysql> select movie_no,title from movie_24 where movie_no in (select movie_no from invoice_24
-> where cust_id in (select custid_id from cust_24));
```

movie_no	title
1	Krrish
2	Mohabbatein
3	Hera Pheri
4	Baghban
5	Vastushastra

46. Find the type and movie_no of movie issued to cust_id 'a101', 'a102'.

```
mysql> select movie_no,type from movie_24 where movie_no in (select movie_no from invoice_24
-> where cust_id in (select custid_id from cust_24 where cust_id in ('a101','a102')));
```

movie_no	type
1	Action
2	Romantic

47. Find out if the movie starring 'Hritik' issued to any customer and print the cost_id to whom it issued.

```
mysql> select cust_id from invoice_24 where movie_no in (select movie_no from movie_24 where star='Hritik');
```

cust_id
a101

48. Find the fnames,lname who have been issued movie.

```
mysql> select fname, lname from cust_24 where custid_id in (select cust_id from invoice_24 where movie_no is not null);
```

fname	lname
Rachna	Agarwal
Anam	Hamdole
Shreya	More
Shreya	Bhattacharjee
Priyanka	Gorde

5 rows in set (0.00 sec)

F. Queries using Date:

49. Display the movie number and day on which customers were issued movies.

```
mysql> select movie_no, dayname(issue_date) from invoice_24;
```

movie_no	dayname(issue_date)
1	Wednesday
2	Thursday
3	Wednesday
3	Wednesday
4	Tuesday
4	Wednesday
5	Saturday

7 rows in set (0.00 sec)

50. Display the month (in alphabets) in which customers is supposed to return the movie.

```
mysql> select cust_id, monthname(return_date) from invoice_24;
```

cust_id	monthname(return_date)
a101	May
a102	May
a103	May
a104	May
a105	May
a109	May
a104	May

51. Display the issue_date in the format of 'dd-mm-yy'.

```
mysql> select date_format(issue_date, '%d-%m-%y') from invoice_24;
```

date_format(issue_date, '%d-%m-%y')
02-05-07
03-05-07
02-05-07
09-05-07
08-05-07
09-05-07
12-05-07

52. Find the date 15 after the current date.

```
mysql> SELECT DATE_ADD(CURDATE(), INTERVAL 15 DAY) AS date_15_days_later;
```

date_15_days_later
2023-10-29

53. Find the number of days elapsed between the current date and the return date of the movie for all customers.

```
mysql> SELECT cust_id, movie_no, DATEDIFF(CURDATE(), return_date) AS days_elapsed  
-> FROM invoice_24;
```

cust_id	movie_no	days_elapsed
a101	1	6007
a102	2	6006
a103	3	6006
a104	3	6000
a105	4	6001
a109	4	5999
a104	5	5996

G. Table updating:

54. Change the telephone number of rachna to 2811281.

```
mysql> update cust_24 set phone_no='2811281' where fname='rachna';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from cust_24;
```

custid_id	lname	fname	area	phone_no
a101	Agarwal	Rachna	Panvel	2811281
a102	Hamdule	Anam	Panvel	9978654321
a103	More	Shreya	Panvel	9898654321
a104	Bhattacharjee	Shreya	Panvel	9898654541
a105	Gorde	Priyanka	Panvel	8098654541

55. Change the issue_date of cust_id 'a101' to '05/04/07'.

```
mysql> update invoice_24 set issue_date='2007-04-05' where cust_id='a101';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from invoice_24;
```

inv_no	movie_no	cust_id	issue_date	return_date
i01	1	a101	2007-04-05	2007-05-04
i02	2	a102	2007-05-03	2007-05-05
i03	3	a103	2007-05-02	2007-05-05
i04	3	a104	2007-05-09	2007-05-11
i05	4	a105	2007-05-08	2007-05-10
i06	4	a109	2007-05-09	2007-05-12
i07	5	a104	2007-05-12	2007-05-15

56. Change the price of 'Baghban' to 250.

```
mysql> update movie_24 set price=250 where title='baghban';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from movie_24;
```

movie_no	title	type	star	price
1	Krrish	Action	Hritik	150.00
2	Mohabbatein	Romantic	Shahrukh	100.00
3	Hera Pheri	Comedy	Akshay	50.00
4	Baghban	Family	Amitabh	250.00
5	Vastushastra	Thriller	Rajpal	70.00
6	Titanic	Romantic	Caprio	200.00

57. Delete the record with inv_no 'i07' from invoice2 table.

```
mysql> delete from invoice_24 where inv_no='i07';
Query OK, 1 row affected (0.01 sec)

mysql> select * from invoice_24;
```

inv_no	movie_no	cust_id	issue_date	return_date
i01	1	a101	2007-04-05	2007-05-04
i02	2	a102	2007-05-03	2007-05-05
i03	3	a103	2007-05-02	2007-05-05
i04	3	a104	2007-05-09	2007-05-11
i05	4	a105	2007-05-08	2007-05-10
i06	4	a109	2007-05-09	2007-05-12

58. Delete all records having return_date before 10th may 2007.

```
mysql> delete from invoice_24 where return_date<'2007-05-10';
Query OK, 3 rows affected (0.00 sec)
```

```
mysql> select * from invoice_24;
```

inv_no	movie_no	cust_id	issue_date	return_date
i04	3	a104	2007-05-09	2007-05-11
i05	4	a105	2007-05-08	2007-05-10
i06	4	a109	2007-05-09	2007-05-12

3 rows in set (0.00 sec)

59. Change the area of custid_id 'a103' to 'Kamothe'.

```
mysql> update cust_24 set area='Kamothe' where custid_id='a103';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> select * from cust_24;
```

custid_id	lname	fname	area	phone_no
a101	Agarwal	Rachna	Panvel	2811281
a102	Hamdule	Anam	Panvel	9978654321
a103	More	Shreya	Kamothe	9898654321
a104	Bhattacharjee	Shreya	Panvel	9898654541
a105	Gorde	Priyanka	Panvel	8098654541

5 rows in set (0.00 sec)

60. Change the return_date of invoice number 'i06' to '16-aug-07'.

```
mysql> update invoice_24 set return_date='2007-08-16' where inv_no='i06';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> select * from invoice_24;
```

inv_no	movie_no	cust_id	issue_date	return_date
i04	3	a104	2007-05-09	2007-05-11
i05	4	a105	2007-05-08	2007-05-10
i06	4	a109	2007-05-09	2007-08-16

3 rows in set (0.00 sec)

Practical 1

Aim: MongoDB Basics & Implementing Aggregation

1. Write a MongoDB query to create and drop databases.

```

> use pillaidb
switched to db pillaidb
> db.createCollection('student')
{ "ok" : 1 }
> show dbs
admin      0.000GB
local      0.000GB
mydb       0.000GB
pillaidb   0.000GB
> db.dropDatabase()
{ "dropped" : "pillaidb", "ok" : 1 }
> show dbs
admin      0.000GB
local      0.000GB
mydb       0.000GB

```

2. Write a MongoDB query to create, display and drop collection.

```

> db.createCollection('student')
{ "ok" : 1 }
> db.student.drop()
true
> show collections

```

3. Write a MongoDB query to insert, query, update and delete a document.

- a. Insert

```

> db.student.insertOne({rollno:1,name:"John",address:"New York",marks:84})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("66e2efdc8815597616725540")
}
> db.student.insertOne({rollno:2,name:"Dazai",address:"Yokohama",marks:100})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("66e2effb8815597616725541")
}
> db.student.insertOne({rollno:3,name:"Dimple",address:"Seawoods",marks:48})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("66e2f01b8815597616725542")
}

```

- b. View

```
> db.student.find().pretty()
{
  "_id" : ObjectId("66e2efdc8815597616725540"),
  "rollno" : 1,
  "name" : "John",
  "address" : "New York",
  "marks" : 84
}
{
  "_id" : ObjectId("66e2effb8815597616725541"),
  "rollno" : 2,
  "name" : "Dazai",
  "address" : "Yokohama",
  "marks" : 100
}
{
  "_id" : ObjectId("66e2f01b8815597616725542"),
  "rollno" : 3,
  "name" : "Dimple",
  "address" : "Seawoods",
  "marks" : 48
}
```

c. Update

```
> db.student.updateOne({rollno:1},{set:{address:"California"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
  "_id" : ObjectId("66e2efdc8815597616725540"),
  "rollno" : 1,
  "name" : "John",
  "address" : "California",
  "marks" : 84
}
```

d. Delete

```
> db.student.deleteOne({rollno:3})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find()
{ "_id" : ObjectId("66e2efdc8815597616725540"), "rollno" : 1, "name" : "John", "address" : "California", "marks" : 84 }
{ "_id" : ObjectId("66e2effb8815597616725541"), "rollno" : 2, "name" : "Dazai", "address" : "Yokohama", "marks" : 100 }
>
```

4. Write a MongoDB query to use sum, avg, min and max expressions.

```
> db.student.aggregate([{$group:{_id:null,sum_marks:{$sum:"$marks"}}}])
{ "_id" : null, "sum_marks" : 184 }
> db.student.aggregate([{$group:{_id:null,avg_marks:{$avg:"$marks"}}}])
{ "_id" : null, "avg_marks" : 92 }
> db.student.aggregate([{$group:{_id:null,min_marks:{$min:"$marks"}}}])
{ "_id" : null, "min_marks" : 48 }
> db.student.aggregate([{$group:{_id:null,max_marks:{$max:"$marks"}}}])
{ "_id" : null, "max_marks" : 100 }
```

5. Write a MongoDB query to use push and addToSet expressions.

```

> db.student.updateOne({rollno:1},{ $push:{subject:"maths"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
  "_id" : ObjectId("66e2efdc8815597616725540"),
  "rollno" : 1,
  "name" : "John",
  "address" : "California",
  "marks" : 84,
  "subject" : [
    "maths"
  ]
}

> db.student.updateOne({rollno:1},{ $addToSet:{subject:"chemistry"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
  "_id" : ObjectId("66e2efdc8815597616725540"),
  "rollno" : 1,
  "name" : "John",
  "address" : "California",
  "marks" : 84,
  "subject" : [
    "maths",
    "chemistry"
  ]
}

```

6. Write a MongoDB query to use the first and last expression.

```

> db.student.aggregate([{$group: {_id:'$rollno',firstmarks:{$first:"$marks"},
lastMarks:{$last:"$marks"}}}])
{ "_id" : 2, "firstmarks" : 100, "lastMarks" : 100 }
{ "_id" : 1, "firstmarks" : 84, "lastMarks" : 84 }
>

> db.student.aggregate([ {$sort:{rollno:1}}, {$group:{_id:null, firstmarks:{$first:"$marks"}, lastmarks:{$last:"$marks"}}}])
{ "_id" : null, "firstmarks" : 84, "lastmarks" : 100 }
>

```


Practical 2

Aim: Replication, Backup and Restore

1. Write a MongoDB query to create Replica of existing database.
2. Write a MongoDB query to create a backup of an existing database.
3. Write a MongoDB query to restore the database from the backup.

Server 1:

```
C:\Windows\System32\cmd.exe - mongod --port 27018 --dbpath "D:\data1\db" --replSet rs0
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

D:\MongoDB Setup\bin>mongod --port 27018 --dbpath "D:\data1\db" --replSet rs0
2024-08-05T13:49:42.714+0530 I CONTROL [initandlisten] MongoDB starting : pid=10172 port=27018
2024-08-05T13:49:42.715+0530 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Serv
2024-08-05T13:49:42.716+0530 I CONTROL [initandlisten] db version v3.4.24
2024-08-05T13:49:42.716+0530 I CONTROL [initandlisten] git version: 865b4f6a96d0f5425e39a1
2024-08-05T13:49:42.716+0530 I CONTROL [initandlisten] allocator: tcmalloc
2024-08-05T13:49:42.716+0530 I CONTROL [initandlisten] modules: none
2024-08-05T13:49:42.716+0530 I CONTROL [initandlisten] build environment:
2024-08-05T13:49:42.716+0530 I CONTROL [initandlisten] distmod: 2008plus
2024-08-05T13:49:42.716+0530 I CONTROL [initandlisten] distarch: x86_64
2024-08-05T13:49:42.717+0530 I CONTROL [initandlisten] target_arch: x86_64
2024-08-05T13:49:42.717+0530 I CONTROL [initandlisten] options: { net: { port: 27018 }, re
2024-08-05T13:49:42.718+0530 I STORAGE [initandlisten] wiredtiger_open config: create,cach
base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy)
istics_log=(wait=0),verbose=(recovery_progress),
2024-08-05T13:49:42.756+0530 I CONTROL [initandlisten]
2024-08-05T13:49:42.756+0530 I CONTROL [initandlisten] ** WARNING: Access control is not en
```

Client 1:

```
C:\Windows\System32\cmd.exe - mongo --port 27018
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

D:\MongoDB Setup\bin>mongo --port 27018
MongoDB shell version v3.4.24
connecting to: mongodb://127.0.0.1:27018/
MongoDB server version: 3.4.24
Server has startup warnings:
2024-08-05T13:49:42.756+0530 I CONTROL [initandlisten]
2024-08-05T13:49:42.756+0530 I CONTROL [initandlisten] ** WARNING: Access control is not en
or the database.
2024-08-05T13:49:42.757+0530 I CONTROL [initandlisten] **          Read and write access to
nd configuration is unrestricted.
2024-08-05T13:49:42.758+0530 I CONTROL [initandlisten]
> rs.initiate()
{
  "info2" : "no configuration specified. Using a default configuration for the set",
  "me" : "mes017:27018",
  "ok" : 1
}
```



```
rs0:OTHER> rs.conf()
{
  "_id" : "rs0",
  "version" : 1,
  "protocolVersion" : NumberLong(1),
  "members" : [
    {
      "_id" : 0,
      "host" : "mes017:27018",
      "arbiterOnly" : false,
      "buildIndexes" : true,
      "hidden" : false,
      "priority" : 1,
      "tags" : {

      },
      "slaveDelay" : NumberLong(0),
      "votes" : 1
    }
  ],
  "settings" : {
    "chainingAllowed" : true,
    "heartbeatIntervalMillis" : 2000,
    "heartbeatTimeoutSecs" : 10,
    "electionTimeoutMillis" : 10000,
    "catchUpTimeoutMillis" : 60000,
    "getLastErrorModes" : {

    },
    "getLastErrorDefaults" : {
      "w" : 1,
      "wtimeout" : 0
    },
    "replicaSetId" : ObjectId("66b08b533b1e353cdc4f5115")
  }
}
```

```
rs0:PRIMARY> rs.status()
{
  "set" : "rs0",
  "date" : ISODate("2024-08-05T08:21:18.273Z"),
  "myState" : 1,
  "term" : NumberLong(1),
  "syncingTo" : "",
  "syncSourceHost" : "",
  "syncSourceId" : -1,
  "heartbeatIntervalMillis" : NumberLong(2000),
  "optimes" : {
    "lastCommittedOpTime" : {
      "ts" : Timestamp(1722846077, 1),
      "t" : NumberLong(1)
    },
    "appliedOpTime" : {
      "ts" : Timestamp(1722846077, 1),
      "t" : NumberLong(1)
    },
    "durableOpTime" : {
      "ts" : Timestamp(1722846077, 1),
      "t" : NumberLong(1)
    }
  },
  "members" : [
    {

```

```

rs0:PRIMARY> rs.add("mes017:27019")
{ "ok" : 1 }
rs0:PRIMARY> rs.add("mes017:27020")
{ "ok" : 1 }
rs0:PRIMARY> use college
switched to db college
rs0:PRIMARY> db.posts.insertOne({
...   title: "Post Title 1",
...   body: "Body of post.",
...   category: "News",
...   likes: 1,
...   tags: ["news", "events"],
...   date: Date()
... });
{
  "acknowledged" : true,
  "insertedId" : ObjectId("66b0917b41ce9a35a12e6543")
}

rs0:PRIMARY> db.isMaster()
{
  "hosts" : [
    "mes017:27018",
    "mes017:27019",
    "mes017:27020"
  ],
  "setName" : "rs0",
  "setVersion" : 3,
  "ismaster" : true,
  "secondary" : false,
  "primary" : "mes017:27018",
  "me" : "mes017:27018",
  "electionId" : ObjectId("7fffffff0000000000000001"),
  "lastWrite" : {
    "opTime" : {
      "ts" : Timestamp(1722848947, 1),
      "t" : NumberLong(1)
    },
    "lastWriteDate" : ISODate("2024-08-05T09:09:07Z")
  },
  "maxBsonObjectSize" : 16777216,

```

Server 2:

```

C:\Windows\System32\cmd.exe - mongod --port 27019 --dbpath "D:\data2\db" --replSet rs0
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

D:\MongoDB Setup\bin>mongo --port 27019 --dbpath "D:\data2\db" --replSet rs0
Error parsing command line: unrecognised option '--dbpath'
try 'mongo --help' for more information

D:\MongoDB Setup\bin>mongod --port 27019 --dbpath "D:\data2\db" --replSet rs0
2024-08-05T13:55:22.547+0530 I CONTROL [initandlisten] MongoDB starting : pid=10296 port=
27019 dbpath=D:\data2\db 64-bit host=mes017
2024-08-05T13:55:22.548+0530 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Ser
ver 2008 R2
2024-08-05T13:55:22.548+0530 I CONTROL [initandlisten] db version v3.4.24
2024-08-05T13:55:22.548+0530 I CONTROL [initandlisten] git version: 865b4f6a96d0f5425e39a
18337105f33e8db504d
2024-08-05T13:55:22.548+0530 I CONTROL [initandlisten] allocator: tcmalloc

```

Client 2:

```

C:\Windows\System32\cmd.exe - mongo --port 27019
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

D:\MongoDB Setup\bin>mongo --port 27019
MongoDB shell version v3.4.24
connecting to: mongodb://127.0.0.1:27019/
MongoDB server version: 3.4.24
Server has startup warnings:
2024-08-05T13:55:22.583+0530 I CONTROL [initandlisten]
2024-08-05T13:55:22.583+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled
for the database.
2024-08-05T13:55:22.583+0530 I CONTROL [initandlisten] **           Read and write access to data
and configuration is unrestricted.
2024-08-05T13:55:22.583+0530 I CONTROL [initandlisten]

```

```

C:\Windows\System32\cmd.exe - mongo --port 27019
}
rs0:SECONDARY> rs.slaveOk()
rs0:SECONDARY> show dbs
admin      0.000GB
college    0.000GB
local      0.000GB
rs0:SECONDARY> db.posts.find().pretty()
rs0:SECONDARY> db.posts.find()
rs0:SECONDARY> use college
switched to db college
rs0:SECONDARY> db.posts.find()
{ "_id" : ObjectId("66b0917b41ce9a35a12e6543"), "title" : "Post Title 1", "body" : "Body of post."
, "category" : "News", "likes" : 1, "tags" : [ "news", "events" ], "date" : "Mon Aug 05 2024 14:16
:51 GMT+0530 (India Standard Time)" }
{ "_id" : ObjectId("66b091f041ce9a35a12e6544"), "title" : "Post Title 2", "body" : "Body of post."
, "category" : "Event", "likes" : 2, "tags" : [ "news", "events" ], "date" : "Mon Aug 05 2024 14:1
8:48 GMT+0530 (India Standard Time)" }
{ "_id" : ObjectId("66b091f741ce9a35a12e6545"), "title" : "Post Title 3", "body" : "Body of post."
, "category" : "Technology", "likes" : 3, "tags" : [ "news", "events" ], "date" : "Mon Aug 05 2024
14:18:55 GMT+0530 (India Standard Time)" }
{ "_id" : ObjectId("66b091fc41ce9a35a12e6546"), "title" : "Post Title 4", "body" : "Body of post."
, "category" : "Event", "likes" : 4, "tags" : [ "news", "events" ], "date" : "Mon Aug 05 2024 14:1
9:00 GMT+0530 (India Standard Time)" }
{ "_id" : ObjectId("66b091ff41ce9a35a12e6547"), "title" : "Post Title 4", "body" : "Body of post."

```

Server 3:

```

C:\Windows\System32\cmd.exe - mongod --port 27020 --dbpath "D:\data3\db" --replSet rs0
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

D:\MongoDB Setup\bin>mongod --port 27020 --dbpath "D:\data3\db" --replSet rs0
2024-08-05T14:08:29.605+0530 I CONTROL [initandlisten] MongoDB starting : pid=6524 port
-bit host=mes017
2024-08-05T14:08:29.605+0530 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows S
2024-08-05T14:08:29.605+0530 I CONTROL [initandlisten] db version v3.4.24
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] git version: 865b4f6a96d0f5425e3
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] allocator: tcmalloc
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] modules: none
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] build environment:
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] distmod: 2008plus
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] distarch: x86_64
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] target_arch: x86_64
2024-08-05T14:08:29.606+0530 I CONTROL [initandlisten] options: { port: 27020 }

```

Client 3:

```

C:\Windows\System32\cmd.exe - mongo --port 27020
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

D:\MongoDB Setup\bin>mongo --port 27020
MongoDB shell version v3.4.24
connecting to: mongodb://127.0.0.1:27020/
MongoDB server version: 3.4.24
Server has startup warnings:
2024-08-05T14:08:29.641+0530 I CONTROL [initandlisten]
2024-08-05T14:08:29.641+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-08-05T14:08:29.641+0530 I CONTROL [initandlisten] **          Read-only user configuration is unrestricted.
2024-08-05T14:08:29.641+0530 I CONTROL [initandlisten]

```

```

C:\Windows\System32\cmd.exe - mongo --port 27020
}
rs0:SECONDARY> rs.slaveOk()
rs0:SECONDARY> show dbs
admin      0.000GB
college    0.000GB
local      0.000GB
rs0:SECONDARY> db.posts.find()
rs0:SECONDARY> use college
switched to db college
rs0:SECONDARY> db.posts.find().pretty()
{
  "_id" : ObjectId("66b0917b41ce9a35a12e6543"),
  "title" : "Post Title 1",
  "body" : "Body of post.",
  "category" : "News",
  "likes" : 1,
  "tags" : [
    "news",
    "events"
  ],
  "date" : "Mon Aug 05 2024 14:16:51 GMT+0530 (India Standard Time)"
}

```

Restore

```

C:\Windows\System32\cmd.exe - mongod.exe --dbpath "D:\data\db" --port 27019

D:\MongoDB Setup\bin>mongod.exe --dbpath "D:\data\db" --port 27019
2024-08-12T14:21:37.567+0530 I CONTROL [initandlisten] MongoDB starting : pid=15316 port=27019 dbpath=D:\data\db 64-bit
host=mes017
2024-08-12T14:21:37.567+0530 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2024-08-12T14:21:37.568+0530 I CONTROL [initandlisten] db version v3.4.24
2024-08-12T14:21:37.578+0530 I CONTROL [initandlisten] git version: 865b4f6a96d0f5425e39a18337105f33e8db504d
2024-08-12T14:21:37.579+0530 I CONTROL [initandlisten] allocator: tcmalloc
2024-08-12T14:21:37.579+0530 I CONTROL [initandlisten] modules: none
2024-08-12T14:21:37.579+0530 I CONTROL [initandlisten] build environment:
2024-08-12T14:21:37.580+0530 I CONTROL [initandlisten] distmod: 2008plus
2024-08-12T14:21:37.580+0530 I CONTROL [initandlisten] distarch: x86_64
2024-08-12T14:21:37.580+0530 I CONTROL [initandlisten] target_arch: x86_64
2024-08-12T14:21:37.580+0530 I CONTROL [initandlisten] options: { net: { port: 27019 }, storage: { dbPath: "D:\data\db"
} }
2024-08-12T14:21:37.581+0530 I STORAGE [initandlisten] wiredtiger open config: create,cache_size=6038M,session_max=2000
0,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal
,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),verb
ose=(recovery_progress),
2024-08-12T14:21:37.584+0530 E STORAGE [initandlisten] WiredTiger error (0) [1723452697:584097][15316:140731544983696],
file:WiredTiger.wt, connection: D:\data\db\WiredTiger.wt: handle-open: CreateFileW: The system cannot find the file spe
cified.
2024-08-12T14:21:37.585+0530 I - [initandlisten] Assertion: 28595:2: No such file or directory src\mongo\db\stora

C:\Windows\System32\cmd.exe - mongo --port 27019

D:\MongoDB Setup\bin>mongo --port 27019
MongoDB shell version v3.4.24
connecting to: mongodb://127.0.0.1:27019/
MongoDB server version: 3.4.24
Server has startup warnings:
2024-08-12T14:22:08.925+0530 I CONTROL [initandlisten]
2024-08-12T14:22:08.925+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-08-12T14:22:08.926+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is u
nrestricted.
2024-08-12T14:22:08.927+0530 I CONTROL [initandlisten]

Select C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

D:\MongoDB Setup\bin>mongorestore dump --port 27019
2024-08-12T14:23:17.870+0530 preparing collections to restore from
2024-08-12T14:23:17.874+0530 reading metadata for mydb.posts from dump\mydb\posts.metadata.json
2024-08-12T14:23:17.875+0530 reading metadata for mydb.student from dump\mydb\student.metadata.json
2024-08-12T14:23:17.885+0530 restoring mydb.posts from dump\mydb\posts.bson
2024-08-12T14:23:17.888+0530 no indexes to restore
2024-08-12T14:23:17.888+0530 finished restoring mydb.posts (4 documents)
2024-08-12T14:23:19.893+0530 restoring mydb.student from dump\mydb\student.bson
2024-08-12T14:23:19.895+0530 no indexes to restore
2024-08-12T14:23:19.895+0530 finished restoring mydb.student (2 documents)
2024-08-12T14:23:19.895+0530 done

D:\MongoDB Setup\bin>

```

Practical 3

Aim: Java and MongoDB

1. Connecting Java/Python/R with MongoDB and performing inserting, retrieving, updating and deleting.

1. index.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJISAwIGgFAW/dAiS6JXm"
crossorigin="anonymous">
    </head>
    <body>
        <div class="container">
            <form action="insert.jsp" method="POST">
                <div class="form-group">
                    <label for="Rollno">Roll no. : </label>
                    <input type="text" class="form-control" id="rollno" name="rollno"
placeholder="Enter Roll No. ">
                </div>
                <div class="form-group">
                    <label for="Name">Name : </label>
                    <input type="text" class="form-control" id="name" name="name"
placeholder="Enter Name">
                </div>
                <div class="form-group">
                    <label for="Address">Address : </label>
                    <input type="text" class="form-control" id="addr" name="addr"
placeholder="Enter Address">
                </div>
                <button type="submit" class="btn btn-primary">Submit</button>

            </form>
        </div>
    </body>

```

```

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KChRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI"
crossorigin="anonymous"></script>
</html>

```

2. Insert.jsp

```

<%@page import="java.util.Iterator"%>
<%@page contentType="text/html" pageEncoding="UTF-8" import="com.mongodb.client.*,
    org.bson.Document, com.mongodb.MongoClient" %>
<%

    MongoClient mongo = new MongoClient( "localhost" , 27018);
    MongoDBDatabase database = mongo.getDatabase("mydb");

    int rollNo=Integer.parseInt(request.getParameter("rollno"));
    String name=request.getParameter("name");
    String address=request.getParameter("addr");

    //    database.createCollection("student");
    //    out.println("Collection created successfully<br>");
    MongoCollection<Document> collection =database.getCollection("student");
    out.println("Collection student selected successfully<br>");
    Document document = new Document("Rollno", rollNo)
        .append("studname",name)
        .append("studaddr", address);
    collection.insertOne(document);
    out.println("Document inserted successfully<br>");

    FindIterable<Document> iterDoc = collection.find();
    int i = 1;
    Iterator it = iterDoc.iterator();

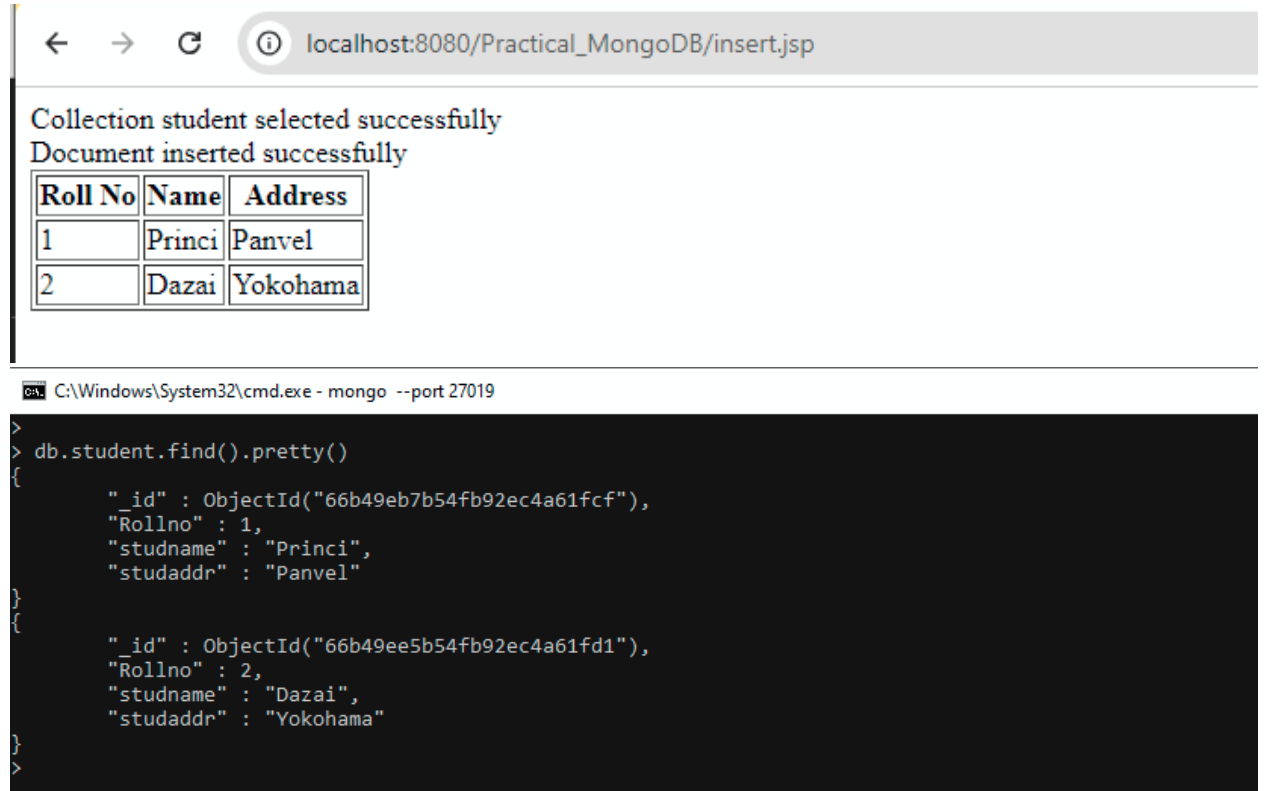
%>
<div class="container">
<table border="1">
    <tr>
    <th>Roll No</th>
    <th>Name</th>

```

```

        <th>Address</th>
    </tr>
<%
    while (it.hasNext()) {
    Document doc=(Document)it.next();
%>
        <tr>
        <td> <%= doc.get("Rollno") %></td>
        <td><%= doc.get("studname") %></td>
        <td><%= doc.get("studaddr") %></td>
        </tr>
<%
        i++;
    }
%>
</table>
</div>

```



Collection student selected successfully
Document inserted successfully

Roll No	Name	Address
1	Princi	Panvel
2	Dazai	Yokohama

```

C:\Windows\System32\cmd.exe - mongo --port 27019
> db.student.find().pretty()
{
  "_id" : ObjectId("66b49eb7b54fb92ec4a61fcf"),
  "Rollno" : 1,
  "studname" : "Princi",
  "studaddr" : "Panvel"
}
{
  "_id" : ObjectId("66b49ee5b54fb92ec4a61fd1"),
  "Rollno" : 2,
  "studname" : "Dazai",
  "studaddr" : "Yokohama"
}
>

```

3. updateform.html

```
<!DOCTYPE html>
```

```
<!--
```

To change this license header, choose License Headers in Project Properties.

To change this template file, choose Tools | Templates

and open the template in the editor.

```
-->
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJISAwIGgFAW/dAiS6JXm"
crossorigin="anonymous">

    </head>
    <body>
      <div class="container">
        <form action="updateData.jsp" method="POST">
          <div class="form-group">
            <label for="Rollno">Roll no. : </label>
            <input type="text" class="form-control" id="rollno" name="rollno"
placeholder="Enter Roll No. ">
          </div>
          <div class="form-group">
            <label for="Name">Name : </label>
            <input type="text" class="form-control" id="name" name="name"
placeholder="Enter Name">
          </div>
          <div class="form-group">
            <label for="Address">Address : </label>
            <input type="text" class="form-control" id="addr" name="addr"
placeholder="Enter Address">
          </div>
          <button type="submit" class="btn btn-primary">Update</button>

        </form>
      </div>
    </body>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KChRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKt3Rn7W3mgPxxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
integrity="sha384-
```

JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI"
crossorigin="anonymous"></script>

</html>

4. updatedata.jsp

```
<%--
    Document : updateData
    Created on : Aug 8, 2024, 3:00:11 PM
    Author : Student
--%>

<%@page import="java.util.Iterator"%>
<%@page import="com.mongodb.client.model.Updates"%>
<%@page import="com.mongodb.client.model.Filters"%>
<%@page contentType="text/html" pageEncoding="UTF-8"
import="com.mongodb.client.*,
    org.bson.Document, com.mongodb.MongoClient" %>

<%
    MongoClient mongo = new MongoClient( "localhost" , 27019 );

    // Accessing the database
    MongoDB database = mongo.getDatabase("mydb");
    int rollno=Integer.parseInt(request.getParameter("rollno"));
    String name=request.getParameter("name");
    String address=request.getParameter("addr");
    // Retrieving a collection
    MongoCollection<Document> collection =database.getCollection("student");
    out.println("Collection student selected successfully");
    collection.updateOne(Filters.eq("Rollno", rollno),
    Updates.set("studaddr", address));
    out.println("Document update successfully...");

    FindIterable<Document> iterDoc = collection.find();
    int i = 1;
    Iterator it = iterDoc.iterator();
%>

<div class="container">
<table border="1">
    <tr>
    <th>Roll No</th>
    <th>Name</th>
    <th>Address</th>
    </tr>

<%
    while (it.hasNext()) {
```

```

        Document doc=(Document)it.next();
    %>
        <tr>
        <td> <%= doc.get("Rollno") %></td>
        <td><%= doc.get("studname") %></td>
        <td><%= doc.get("studaddr") %></td>
        </tr>
    <%
        i++;
    %>
</table>
</div>

```

← → ↻ ⓘ localhost:8080/Practical_MongoDB/updateData.jsp

Collection student selected successfully Document update successfully...

Roll No	Name	Address
1	Princi	New Panvel
2	Dazai	Yokohama

```

    "_id" : ObjectId("66b49eb7b54fb92ec4a61fcf"),
    "Rollno" : 1,
    "studname" : "Princi",
    "studaddr" : "New Panvel"
  },
  {
    "_id" : ObjectId("66b49ee5b54fb92ec4a61fd1"),
    "Rollno" : 2,
    "studname" : "Dazai",
    "studaddr" : "Yokohama"
  }
]
>

```

5. Deleteform.jsp

```

<html>
    <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJISAwIGgFAW/dAiS6JXm"
crossorigin="anonymous">
    </head>

```

```

<body>
<div class="container">
  <form action="deleteData.jsp" method="POST">
    <div class="form-group">
      <label for="Rollno">Roll no. : </label>
      <input type="text" class="form-control" id="rollno" name="rollno"
placeholder="Enter Roll No. ">
    </div>
    <button type="submit" class="btn btn-primary">Delete</button>
  </form>
</div>
</body>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCKRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI"
crossorigin="anonymous"></script>
</html>

```

6. Deletedata.jsp

```

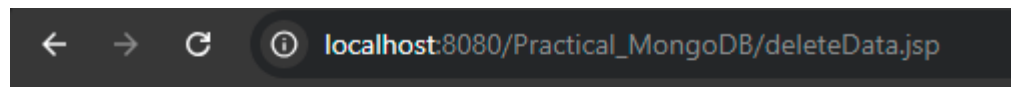
<%@page import="java.util.Iterator"%>
<%@page import="com.mongodb.client.model.Filters"%>
<%@page contentType="text/html" pageEncoding="UTF-8" import="com.mongodb.client.*,
org.bson.Document, com.mongodb.MongoClient" %>
<%
    MongoClient mongo = new MongoClient( "localhost" , 27019 );
    MongoDB database = mongo.getDatabase("mydb");
    int rollNo=Integer.parseInt(request.getParameter("rollno"));
    MongoCollection<Document> collection =database.getCollection("student");
    out.println("Collection student selected successfully");
    collection.deleteOne(Filters.eq("Rollno", rollNo));
    System.out.println("Document deleted successfully...");
    FindIterable<Document> iterDoc = collection.find();
    int i = 1;
    Iterator it = iterDoc.iterator();
%>
<div class="container">
<table border="1">
  <tr>
    <th>Roll No</th>
    <th>Name</th>

```

```

        <th>Address</th>
    </tr>
    <%
        while (it.hasNext()) {
            Document doc=(Document)it.next();
    %>
        <tr>
            <td> <%= doc.get("Rollno") %></td>
            <td><%= doc.get("studname") %></td>
            <td><%= doc.get("studaddr") %></td>
        </tr>
    <%
        i++;
    }
    %>
</table>
</div>

```



Collection student selected successfully

Roll No	Name	Address
2	Dazai	Yokohama

```

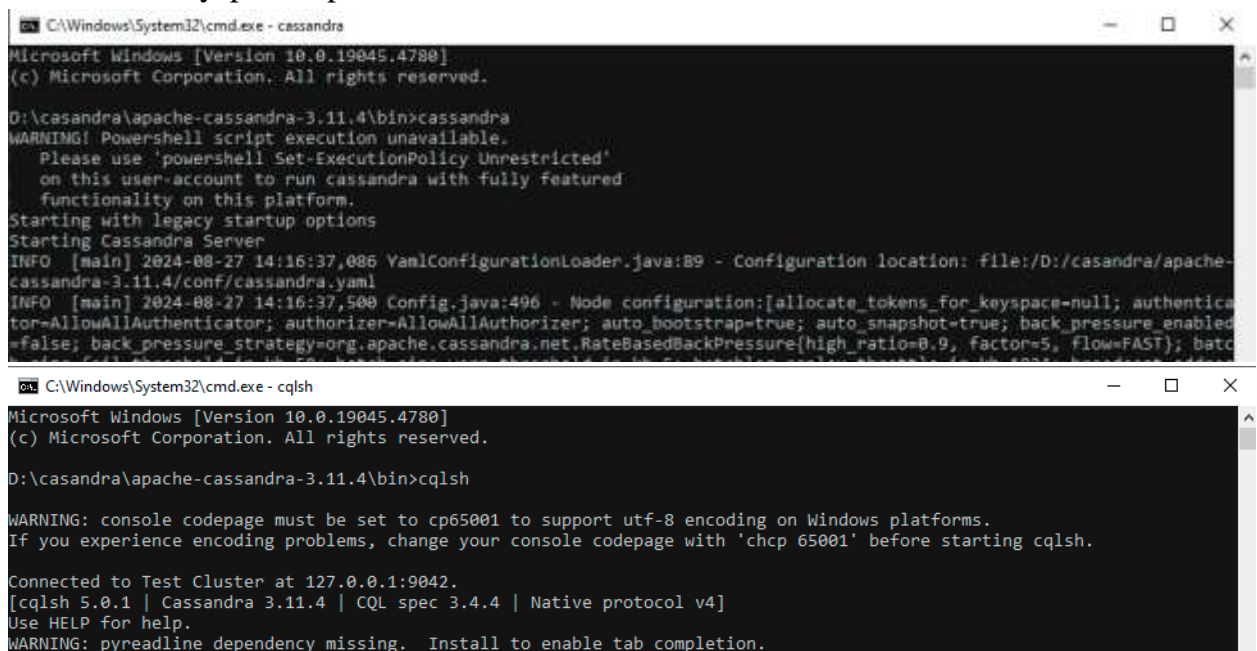
> db.student.find().pretty()
{
  "_id" : ObjectId("66b49ee5b54fb92ec4a61fd1"),
  "Rollno" : 2,
  "studname" : "Dazai",
  "studaddr" : "Yokohama"
}
>

```

Practical 4

Aim: Create Data Model using Cassandra

1. Cassandra Keyspace Operations.



```
C:\Windows\System32\cmd.exe - cassandra

Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

D:\cassandra\apache-cassandra-3.11.4\bin>cassandra
WARNING! Powershell script execution unavailable.
Please use 'powershell Set-ExecutionPolicy Unrestricted'
on this user-account to run cassandra with fully featured
functionality on this platform.
Starting with legacy startup options
Starting Cassandra Server
INFO [main] 2024-08-27 14:16:37,086 YamlConfigurationLoader.java:89 - Configuration location: file:/D:/cassandra/apache-
cassandra-3.11.4/conf/cassandra.yaml
INFO [main] 2024-08-27 14:16:37,500 Config.java:496 - Node configuration:[allocate_tokens_for_keyspace=null; authentica
tor=AllowAllAuthenticator; authorizer=AllowAllAuthorizer; auto_bootstrap=true; auto_snapshot=true; back_pressure_enabled
=false; back_pressure_strategy=org.apache.cassandra.net.RateBasedBackPressure{high_ratio=0.9, factor=5, flow=FAST}; batc

C:\Windows\System32\cmd.exe - cqlsh

Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

D:\cassandra\apache-cassandra-3.11.4\bin>cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting 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.
WARNING: pyreadline dependency missing. Install to enable tab completion.
```

```

cqlsh> create keyspace pillai with replication={'class':'SimpleStrategy','replication_factor':3};
cqlsh> describe keyspaces;

system_schema  system_auth  system  pillai  system_distributed  system_traces

cqlsh> use pillai;
cqlsh:pillai> alter keyspace pillai with replication={'class':'SimpleStrategy','replication_factor':2};
cqlsh:pillai> drop keyspace pillai;
cqlsh:pillai> describe keyspaces;

system_schema  system_auth  system  system_distributed  system_traces

cqlsh:pillai>

```

2. Cassandra Table Operations.

```

cqlsh:pillai> create keyspace msc with replication={'class':'SimpleStrategy','replication_factor':2};
cqlsh:pillai> create table student(rollno int Primary Key,name text,address text);
ConfigurationException: Keyspace pillai doesn't exist
cqlsh:pillai> use pillai;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Keyspace 'pillai' does not exist"
cqlsh:pillai> use msc
... ;
cqlsh:msc> create table student(rollno int Primary Key,name text,address text);
cqlsh:msc> select * from student;

rollno | address | name
-----+-----+-----
(0 rows)

```

3. Cassandra CRUD Operations.

```

cqlsh:msc> insert into student(rollno,name,address) values(1,'Alya','Tokyo');
cqlsh:msc> insert into student(rollno,name,address) values(2,'Rampo','Yokohama');
cqlsh:msc> select * from student;

rollno | address | name
-----+-----+-----
1 | Tokyo | Alya
2 | Yokohama | Rampo
(2 rows)

cqlsh:msc> alter table student add pincode int;
cqlsh:msc> select * from student;

rollno | address | name | pincode
-----+-----+-----+-----
1 | Tokyo | Alya | null
2 | Yokohama | Rampo | null
(2 rows)
cqlsh:msc> drop table student;
cqlsh:msc> select * from student;
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured table student"
cqlsh:msc>

```

4. Cassandra CQL Types

```

cqlsh:msc> create type addr{
... hno int,
... bname text,
... city text};
SyntaxException: line 1:16 mismatched input '(' expecting '(' (create type addr[{}...])
cqlsh:msc> create type addr{
... hno int,
... bname text,
... city text};
cqlsh:msc> create table utt_student(id int Primary Key, name text, address FROZEN<addr>);
cqlsh:msc> insert into utt_student(id,name,address) values(101,'Nakamura',{hno:10,bname:'soul society',city:'Panvel'});
cqlsh:msc> insert into utt_student(id,name,address) values(102,'Yama',{hno:14,bname:'harmony society',city:'Vashi'});
cqlsh:msc> select * from utt_student;

 id | address | name
-----+-----+-----
 102 | {hno: 14, bname: 'harmony society', city: 'Vashi'} | Yama
 101 | {hno: 10, bname: 'soul society', city: 'Panvel'} | Nakamura
(2 rows)

cqlsh:msc> create table coll_student(sid int Primary Key,name text,address map<text,FROZEN<addr>>,email list<text>,contact set<int>);
cqlsh:msc> insert into coll_student(sid,name,address,email,contact) values(1001,'abc',{hno:10,bname:'harmony society',city:'Panvel'},{'Office':{hno:11,bname:'Soul society',city:'Vashi'}},['abc@gmail.com','xyz@gmail.com'],{123456789,4567891,234512786});
cqlsh:msc> insert into coll_student(sid,name,address,email,contact) values(1002,'xyz',{hno:23,bname:'SM society',city:'Belapur'},{'Office':{hno:14,bname:'JK society',city:'Kharghar'}},['sdfkjc@gmail.com','xyzasldfnj@gmail.com'],{147852369,123654789});
cqlsh:msc> select * from coll_student;

 sid | address | contact | email
-----+-----+-----+-----
 1001 | {'Home': {hno: 10, bname: 'harmony society', city: 'Panvel'}, 'Office': {hno: 11, bname: 'Soul society', city: 'Vashi'}} | {4567891, 123456789, 234512786} | ['abc@gmail.com', 'xyz@gmail.com']
 1002 | {'Home': {hno: 23, bname: 'SM society', city: 'Belapur'}, 'Office': {hno: 14, bname: 'JK society', city: 'Kharghar'}} | {123654789, 147852369} | ['sdfkjc@gmail.com', 'xyzasldfnj@gmail.com']
(2 rows)

cqlsh:msc> update coll_student
... set email=email+['abcxyz@gmail.com']
... where sid=1001;
cqlsh:msc> select * from coll_student;

 sid | address | contact | email
-----+-----+-----+-----
 1001 | {'Home': {hno: 10, bname: 'harmony society', city: 'Panvel'}, 'Office': {hno: 11, bname: 'Soul society', city: 'Vashi'}} | {4567891, 123456789, 234512786} | ['abc@gmail.com', 'xyz@gmail.com', 'abcxyz@gmail.com']
 1002 | {'Home': {hno: 23, bname: 'SM society', city: 'Belapur'}, 'Office': {hno: 14, bname: 'JK society', city: 'Kharghar'}} | {123654789, 147852369} | ['sdfkjc@gmail.com', 'xyzasldfnj@gmail.com']
(2 rows)

```



```
cqlsh:msc> update coll_student
... set email[1]='first@gmail.com'
... where sid=1001;
cqlsh:msc> select * from coll_student;
```

sid	address	contact name	email
1001	{'Home': {hno: 10, bname: 'harmony society', city: 'Panvel'}, 'Office': {hno: 11, bname: 'Soul society', city: 'Vashi'}}	{4567891, 123456789, 234512786}	['abc@gmail.com', 'first@gmail.com', 'abcxyz@gmail.com']
1002	{'Home': {hno: 23, bname: 'SM society', city: 'Belapur'}, 'Office': {hno: 14, bname: 'JK society', city: 'Kharghar'}}	{123654789, 147852369}	['sdfkjc@gmail.com', 'xyzasldfnj@gmail.com']

(2 rows)
cqlsh:msc> _

PRACTICAL 5

Aim: Create Data Model using Cassandra

- 1) Cassandra KeySpace Operations.
- 2) Cassandra Table Operations.
- 3) Cassandra CRUD Operations.
- 4) Cassandra CQL Types

Initiate Cassandra

```
C:\Windows\System32\cmd.exe - cassandra
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

D:\apache-cassandra-3.11.4\bin>cassandra
WARNING! Powershell script execution unavailable.
Please use 'powershell Set-ExecutionPolicy Unrestricted'
on this user-account to run cassandra with fully featured
functionality on this platform.
```

```
C:\Windows\System32\cmd.exe - CQLSH
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

D:\apache-cassandra-3.11.4\bin>CQLSH

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.
```

Cassandra KeySpace Operations

```
cqlsh> create keyspace college with replication ={'class':'SimpleStrategy','replication_factor':3};
```

```
cqlsh> DESCRIBE keyspaces;
```

```
C:\Windows\System32\cmd.exe - CQLSH
cqlsh> create keyspace college with replication ={'class':'Simple Strategy', 'replication_factor':3};
ConfigurationException: Unable to find replication strategy class 'org.apache.cassandra.locator.Simple Strategy'
cqlsh> create keyspace college with replication ={'class':'SimpleStrategy', 'replication_factor':3};
cqlsh> DESCRIBE keyspaces;
```

If willing to drop we use the following code:

```
cqlsh> DROP KEYSPACE keyspace_name;
```

To start creating table in a keyspace we write the following:

```
cqlsh> USE college;
```

Cassandra Table Operations

```
cqlsh:college> CREATE TABLE college.student(id int PRIMARY KEY,name text,address text);
```

```
cqlsh:college> select * from student;
```

```
C:\Windows\System32\cmd.exe - CQLSH
Improper use command.
cqlsh> USE college;
cqlsh:college> CREATE TABLE emp(
    ... emp_id int PRIMARY KEY,
    ... emp_name text,
    ... address text);
SyntaxException: line 4:13 mismatched input ',' expecting EOF (...emp_name text,address text)[,]...)
cqlsh:college> CREATE TABLE college.student(id int PRIMARY KEY,name text,address text);
cqlsh:college> select * from student;

 id | address | name
-----+-----+-----

```

Following are codes for altering the table:

```
ALTER TABLE table name
ADD new column datatype;
```

```
ALTER table name
DROP column name;
```

```
To drop table:
DROP TABLE tablename;
```

Cassandra CRUD Operations

Insertion

```
cqlsh:college> insert into student (id,name,address) values (3116,'Anam','Mumbai');
```

```
cqlsh:college> select * from student;
```

```
C:\Windows\System32\cmd.exe - CQLSH

(0 rows)
cqlsh:college> insert into student (id,name,address) values (3116,'Anam','Mumbai');
cqlsh:college> select * from student;

 id | address | name
-----+-----+-----
 3116 | Mumbai | Anam

(1 rows)
cqlsh:college> insert into student (id,name,address) values (3108,'Shreya','Mansarovar');
cqlsh:college> insert into student (id,name,address) values (3126,'Shreya M','Khandeshwar');
cqlsh:college> insert into student (id,name,address) values (3140,'Anamika','Seawoods');
cqlsh:college> select * from student;

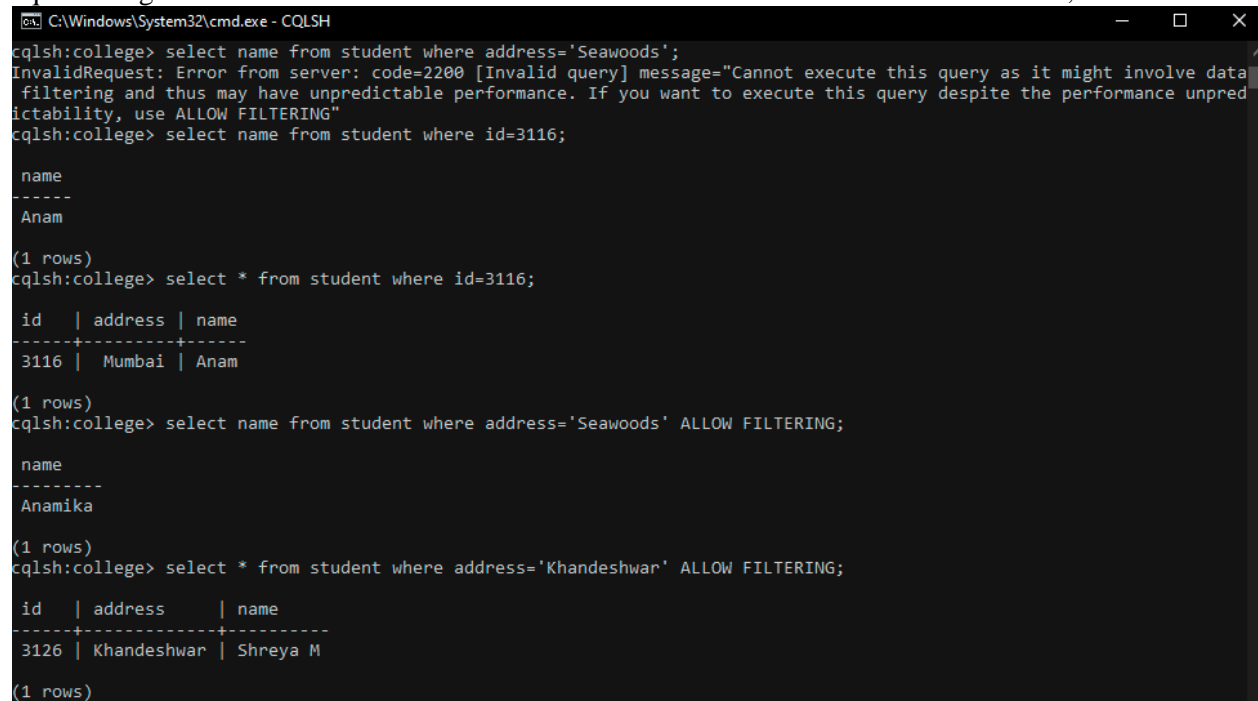
 id | address      | name
-----+-----+-----
 3108 | Mansarovar   | Shreya
 3140 | Seawoods     | Anamika
 3126 | Khandeshwar  | Shreya M
 3116 | Mumbai      | Anam
```

Retrieving

cqlsh:college> select * from student where id=3116;

cqlsh:college> select name from student where address='Seawoods' ALLOW FILTERING;

cqlsh:college> select * from student where address='Khandeshwar' ALLOW FILTERING;



```

C:\Windows\System32\cmd.exe - CQLSH
cqlsh:college> select name from student where address='Seawoods';
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 unpred
ictability, use ALLOW FILTERING"
cqlsh:college> select name from student where id=3116;

name
-----
Anam

(1 rows)
cqlsh:college> select * from student where id=3116;

id | address | name
-----+-----+-----
3116 | Mumbai | Anam

(1 rows)
cqlsh:college> select name from student where address='Seawoods' ALLOW FILTERING;

name
-----
Anamika

(1 rows)
cqlsh:college> select * from student where address='Khandeshwar' ALLOW FILTERING;

id | address | name
-----+-----+-----
3126 | Khandeshwar | Shreya M

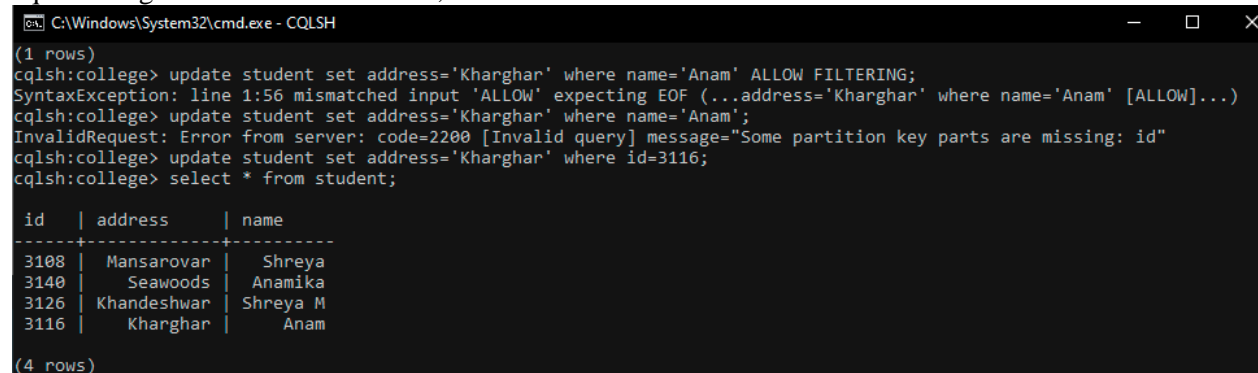
(1 rows)

```

Updation

cqlsh:college> update student set address='Kharghar' where id=3116;

cqlsh:college> select * from student;



```

C:\Windows\System32\cmd.exe - CQLSH
(1 rows)
cqlsh:college> update student set address='Kharghar' where name='Anam' ALLOW FILTERING;
SyntaxException: line 1:56 mismatched input 'ALLOW' expecting EOF (...address='Kharghar' where name='Anam' [ALLOW]...)
cqlsh:college> update student set address='Kharghar' where name='Anam';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Some partition key parts are missing: id"
cqlsh:college> update student set address='Kharghar' where id=3116;
cqlsh:college> select * from student;

id | address | name
-----+-----+-----
3108 | Mansarovar | Shreya
3140 | Seawoods | Anamika
3126 | Khandeshwar | Shreya M
3116 | Kharghar | Anam

(4 rows)

```

Deletion

cqlsh:college> delete from student where id=3108; cqlsh:college>

select * from student;

```
C:\Windows\System32\cmd.exe - CQLSH
(4 rows)
cqlsh:college> delete from student where id=3108;
cqlsh:college> select * from student;

 id | address | name
-----+-----+-----
3140 | Seawoods | Anamika
3126 | Khandeshwar | Shreya M
3116 | Kharghar | Anam
(3 rows)
```

Update And Delete Through Primary Key

cqlsh:college> alter table student add marks int;
 cqlsh:college> select * from student;
 cqlsh:college> update student set marks=90 where id=3116;
 cqlsh:college> update student set marks=92 where id=3126;
 cqlsh:college> update student set marks=91 where id=3140;
 cqlsh:college> select * from student;

```
C:\Windows\System32\cmd.exe - CQLSH
cqlsh:college> alter table student add marks int;
cqlsh:college> select * from student;

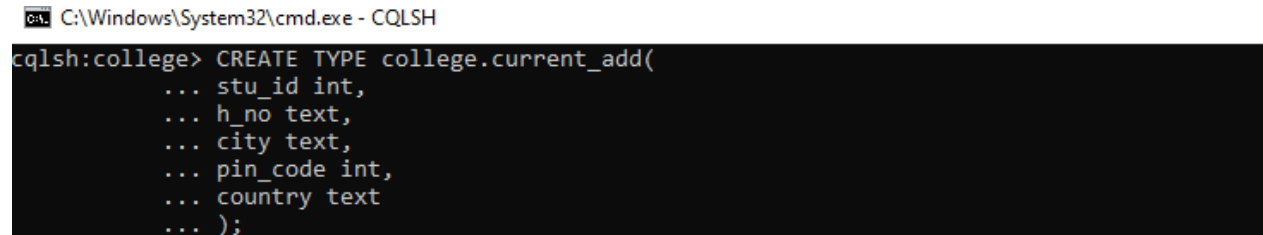
 id | address | marks | name
-----+-----+-----+-----
3140 | Seawoods | null | Anamika
3126 | Khandeshwar | null | Shreya M
3116 | Kharghar | null | Anam
(3 rows)
cqlsh:college> update student set marks=90 where id=3116;
cqlsh:college> update student set marks=92 where id=3126;
cqlsh:college> update student set marks=91 where id=3140;
cqlsh:college> select * from student;

 id | address | marks | name
-----+-----+-----+-----
3140 | Seawoods | 91 | Anamika
3126 | Khandeshwar | 92 | Shreya M
3116 | Kharghar | 90 | Anam
(3 rows)
```

Cassandra CQL Types

User defined type:

```
CREATE TYPE college.current_add(
    ... stu_id int,
    ... h_no text,
    ... city text,
    ... pin_code int,
    ... country text
    ... );
```



```
C:\Windows\System32\cmd.exe - CQLSH
cqlsh:college> CREATE TYPE college.current_add(
    ... stu_id int,
    ... h_no text,
    ... city text,
    ... pin_code int,
    ... country text
    ... );
```

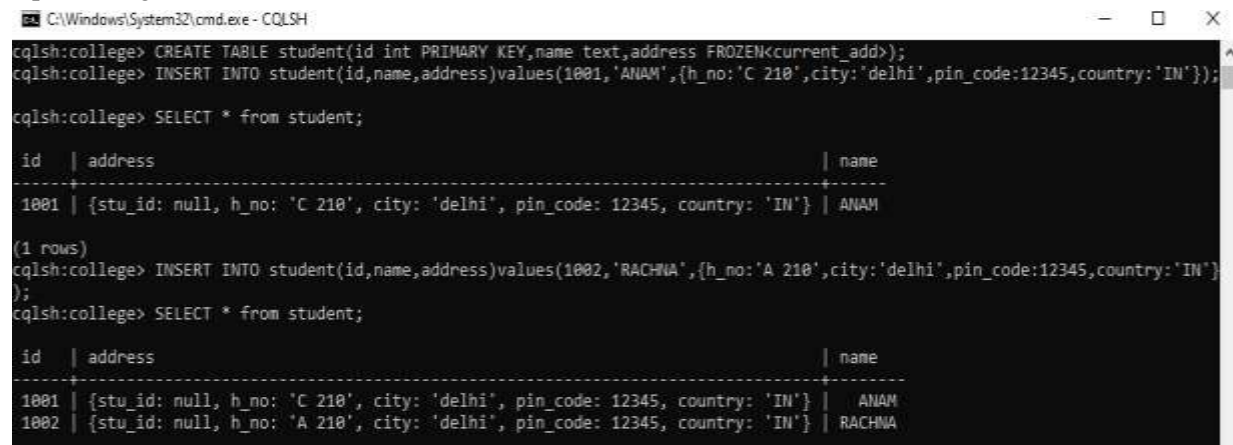
```
cqlsh:college> CREATE TABLE student(id int PRIMARY KEY,name text,address
FROZEN<current_add>);
```

```
cqlsh:college> INSERT INTO student(id,name,address)values(1001,'ANAM',{h_no:'C
210',city:'delhi',pin_code:12345,country:'IN'});
```

```
cqlsh:college> SELECT * from student;
```

```
cqlsh:college> INSERT INTO student(id,name,address)values(1002,'RACHNA',{h_no:'A
210',city:'delhi',pin_code:12345,country:'IN'});
```

```
cqlsh:college> SELECT * from student;
```



```
C:\Windows\System32\cmd.exe - CQLSH
cqlsh:college> CREATE TABLE student(id int PRIMARY KEY,name text,address FROZEN<current_add>);
cqlsh:college> INSERT INTO student(id,name,address)values(1001,'ANAM',{h_no:'C 210',city:'delhi',pin_code:12345,country:'IN'});
cqlsh:college> SELECT * from student;

id | address | name
---+-----+---
1001 | {stu_id: null, h_no: 'C 210', city: 'delhi', pin_code: 12345, country: 'IN'} | ANAM

(1 rows)
cqlsh:college> INSERT INTO student(id,name,address)values(1002,'RACHNA',{h_no:'A 210',city:'delhi',pin_code:12345,country:'IN'});
cqlsh:college> SELECT * from student;

id | address | name
---+-----+---
1001 | {stu_id: null, h_no: 'C 210', city: 'delhi', pin_code: 12345, country: 'IN'} | ANAM
1002 | {stu_id: null, h_no: 'A 210', city: 'delhi', pin_code: 12345, country: 'IN'} | RACHNA
```

To alter:

```
ALTER TYPE keyspace.typename
```

```
ADD field_name field_type;
```

```
cqlsh:college> ALTER TYPE college.current_add ADD building_name text;
```

```
ALTER TYPE typename
```

```
RENAME existing_name TO new_name;
```

To drop Type:

```
cqlsh:college> DROP TYPE college.current_add;
```

With list type:

```
cqlsh:college> CREATE TABLE student1(id int PRIMARY KEY,name text,address
list<FROZEN<current_add>>);
```

```
cqlsh:college>INSERT INTO
```

```
student1(id,name,address)values(1001,'RACHNA',[{stu_id:1001,h_no:'A
210',city:'delhi',pin_code:12345,country:'IN'},{stu_id:1001,h_no:'B110',city:'delhi',pin_code:12245,c
ount ry:'IN'}]);
```

```
cqlsh:college> INSERT INTO student1(id,name,address)values(1002,'ANAM',[{ stu_id:1002,h_no:'A
230',city:'delhi',pin_code:22345,country:'IN'},{stu_id:1002,h_no:'B310',city:'delhi',pin_code:12245,count
ry:'IN'}]);
```

```

C:\Windows\System32\cmd.exe - CQLSH
(2 rows)
cqlsh:college> CREATE TABLE student1(id int PRIMARY KEY,name text,address list<FROZEN<current_add>>);

cqlsh:college> INSERT INTO student1(id,name,address)values(1001,'RACHNA',[{stu_id:1001,h_no:'A 210',ci
ty:'delhi',pin_code:12345,country:'IN'},{stu_id:1001,h_no:'B 110',city:'delhi',pin_code:12245,country:
'IN'}]);

cqlsh:college> INSERT INTO student1(id,name,address)values(1002,'ANAM',[{stu_id:1002,h_no:'A 230',city
:'delhi',pin_code:22345,country:'IN'},{stu_id:1002,h_no:'B 310',city:'delhi',pin_code:12245,country:'I
N'}]);

cqlsh:college> SELECT * from student1;

 id | address
-----+-----
 1001 | [{stu_id: 1001, h_no: 'A 210', city: 'delhi', pin_code: 12345, country: 'IN'}, {stu_id: 1001,
h_no: 'B 110', city: 'delhi', pin_code: 12245, country: 'IN'}] | RACHNA
 1002 | [{stu_id: 1002, h_no: 'A 230', city: 'delhi', pin_code: 22345, country: 'IN'}, {stu_id: 1002,
h_no: 'B 310', city: 'delhi', pin_code: 12245, country: 'IN'}] | ANAM

```

PRACTICAL 6

Aim: Connecting Java with Cassandra and performing inserting, retrieving, updating and deleting

Initiate Cassandra

```
C:\Windows\System32\cmd.exe - cassandra
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

D:\apache-cassandra-3.11.4\bin>cassandra
WARNING! Powershell script execution unavailable.
Please use 'powershell Set-ExecutionPolicy Unrestricted'
on this user-account to run cassandra with fully featured
functionality on this platform.
```

```
C:\Windows\System32\cmd.exe - CQLSH
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

D:\apache-cassandra-3.11.4\bin>CQLSH

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.
```

```
cqlsh> USE college
... ;
cqlsh:college> select * from student;

 id | address | marks | name
-----+-----+-----+-----
 3140 | Seawoods | 91 | Anamika
 3126 | Khandeshwar | 92 | Shreya M
 3116 | Kharghar | 90 | Anam
(3 rows)
cqlsh:college>
```

Creating a java application through Netbeans IDE using Cassandra as database.

CODE:

Inserting through query

```
package cassandra;
```

```
import com.datastax.driver.core.Cluster;import com.datastax.driver.core.Session;public class
Cassandra {
```

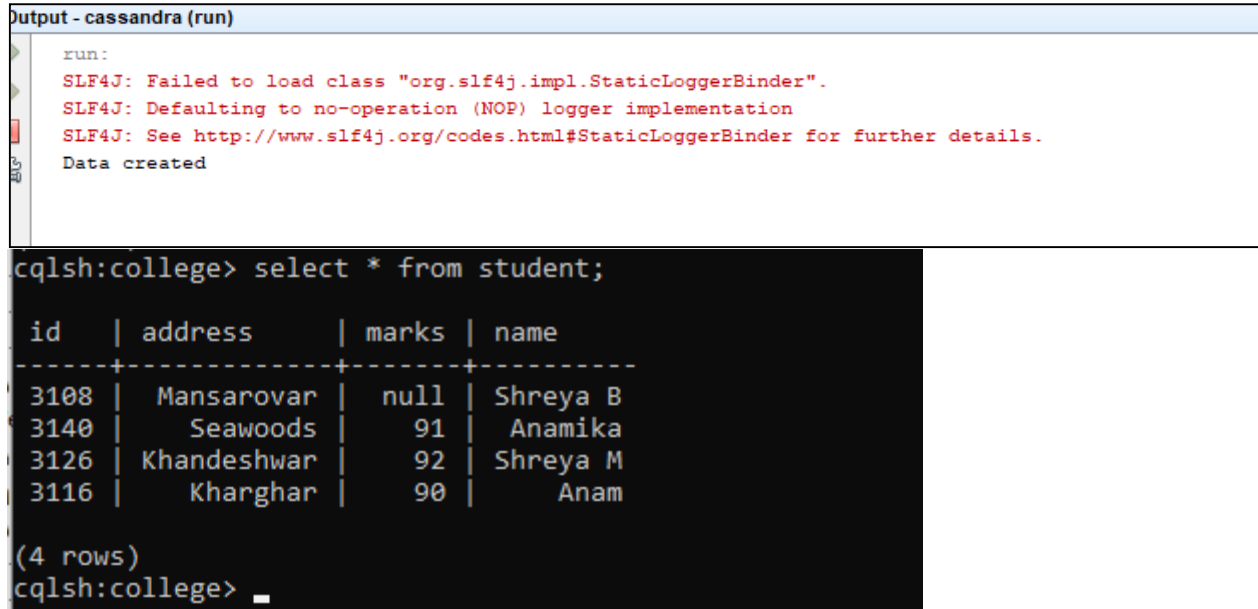
```
public static void main(String[] args) {
```

```
// //queries
```

```
String query1 = "INSERT INTO student (id, name, address)"
+ " VALUES(3108,'Shreya B','Mansarovar');";
```



```
//Creating Cluster object
Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
//Creating Session object
Session session = cluster.connect("college");
//Executing the query session.execute(query1); System.out.println("Data created");
}
}
```



Output - cassandra (run)

```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Data created
```

```
cqlsh:college> select * from student;
```

id	address	marks	name
3108	Mansarovar	null	Shreya B
3140	Seawoods	91	Anamika
3126	Khandeshwar	92	Shreya M
3116	Kharghar	90	Anam

```
(4 rows)
cqlsh:college> _
```

Inserting through user input

```
package cassandra;
import com.datastax.driver.core.Cluster;import com.datastax.driver.core.Session;import java.util.*;
public class Cassandra {
public static void main(String[] args) { Scanner sc=new Scanner(System.in);
System.out.println("Enter roll no"); int roll=sc.nextInt(); System.out.println("Enter Name"); String
name=sc.next(); System.out.println("Enter address"); String add=sc.next();
String query = "INSERT INTO student (id,name,address)"
+ " VALUES("+roll+", '"+name+"', '"+add+"');" ;
Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build(); Session session =
cluster.connect("college");
session.execute(query); System.out.println("Data created");
}}
```

Output - cassandra (run)

```
run:
Enter roll no
3168
Enter Name
Jyoti
Enter address
Mansarovar
```

Output - cassandra (run)

```
Enter Name
Jyoti
Enter address
Mansarovar
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Data created
```

```
cqlsh:college> select * from student;
```

id	address	marks	name
3108	Mansarovar	null	Shreya B
3140	Seawoods	91	Anamika
3126	Khandeshwar	92	Shreya M
3168	Mansarovar	null	Jyoti
3116	Kharghar	90	Anam

```
(5 rows)
```

Updation through user input

```
package cassandra;
```

```
import com.datastax.driver.core.Cluster;import com.datastax.driver.core.Session;import
java.util.Scanner;
```

```
public class cassandra1 {
public static void main(String[] args) {
```

```
//queries
```

```
Scanner sc=new Scanner(System.in);System.out.println("Enter roll no"); int roll=sc.nextInt();
```

```
System.out.println("Enter marks");int mar=sc.nextInt();
```

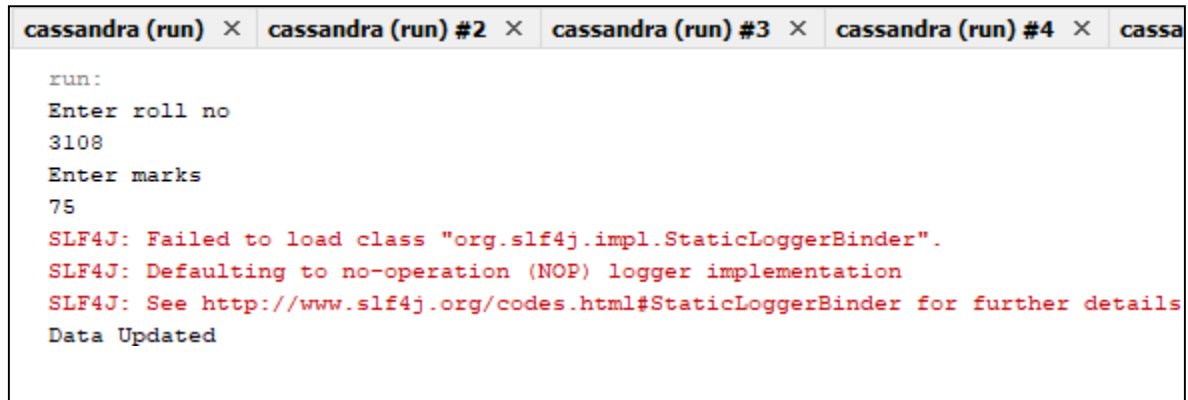
```
String query = "UPDATE student SET marks="+mar+" where id="+roll+";";
```

```
//Creating Cluster object
```

```
Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
```

```
//Creating Session object
Session session = cluster.connect("college");

//Executing the query session.execute(query); System.out.println("Data Updated");
}}
```



The screenshot shows a terminal window with multiple tabs labeled 'cassandra (run)'. The active tab displays the following text:

```
run:
Enter roll no
3108
Enter marks
75
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details
Data Updated
```

Updation through query

```
package cassandra;
```

```
import com.datastax.driver.core.Cluster;import com.datastax.driver.core.Session;import
java.util.Scanner;
public class cassandra1 {
public static void main(String[] args) {
String query = "UPDATE student SET marks=80 where id=3168;"; Cluster cluster =
Cluster.builder().addContactPoint("127.0.0.1").build();Session session = cluster.connect("college");
session.execute(query); System.out.println("Data Updated");
}}
```

```
cqlsh:college> select * from student;
```

id	address	marks	name
3108	Mansarovar	75	Shreya B
3140	Seawoods	91	Anamika
3126	Khandeshwar	92	Shreya M
3168	Mansarovar	80	Jyoti
3116	Kharghar	90	Anam

Retrieval of all data

```
package cassandra;
import com.datastax.driver.core.Cluster;import com.datastax.driver.core.Session;
```

```
import com.datastax.driver.core.ResultSet;import java.util.Scanner;

public class cassandra1 {
public static void main(String[] args) { String query = "Select * from student;";
Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build(); Session session =
cluster.connect("college");
ResultSet result=session.execute(query);System.out.println(result.all());
}}
```

Deletion by user input

```
package cassandra;
import com.datastax.driver.core.Cluster;import com.datastax.driver.core.Session;import
java.util.Scanner;
public class cassandra1 {
public static void main(String[] args) { Scanner sc=new Scanner(System.in);
System.out.println("Enter roll no"); int roll=sc.nextInt();
String query = "DELETE from student where id="+roll+";";
Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build(); Session session =
cluster.connect("college");
session.execute(query); System.out.println("Data deleted");
}}
```

PRACTICAL 7

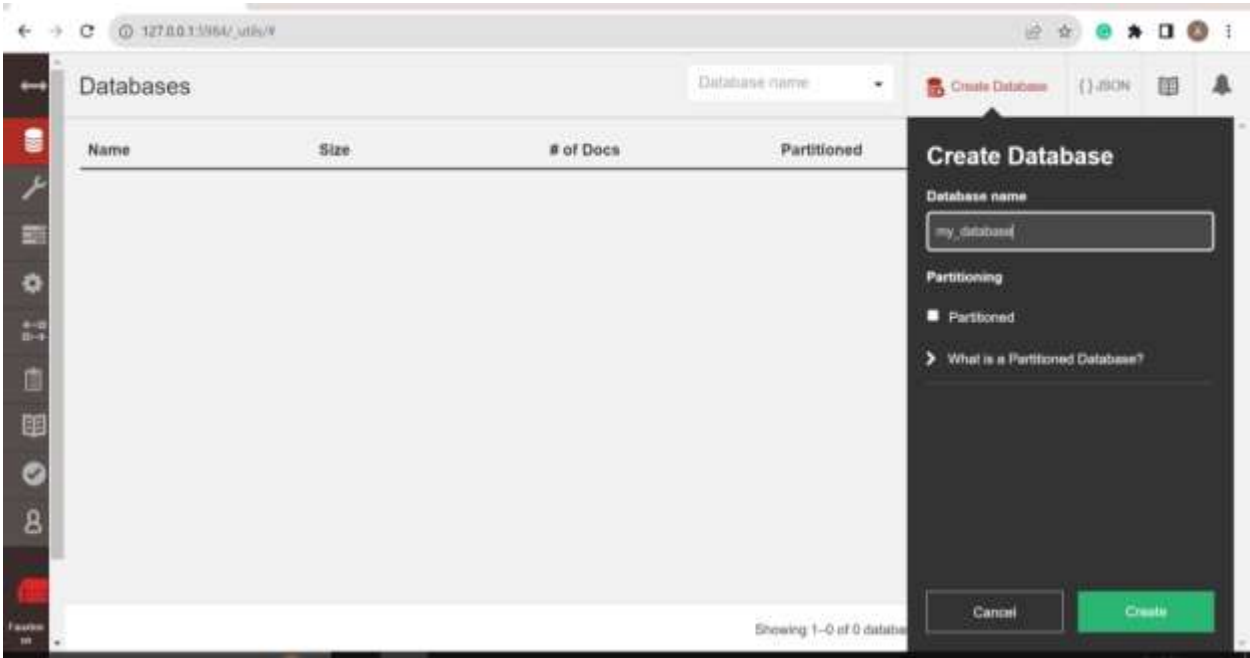
Aim: CouchDB Working with Database

Working with Document
Attaching file

Working with Database-
Using Futon

Futon can be accessed by typing 127.0.0.1:5984/_utils in the browser. Login to Futon using username and password created during installation of couchDB.

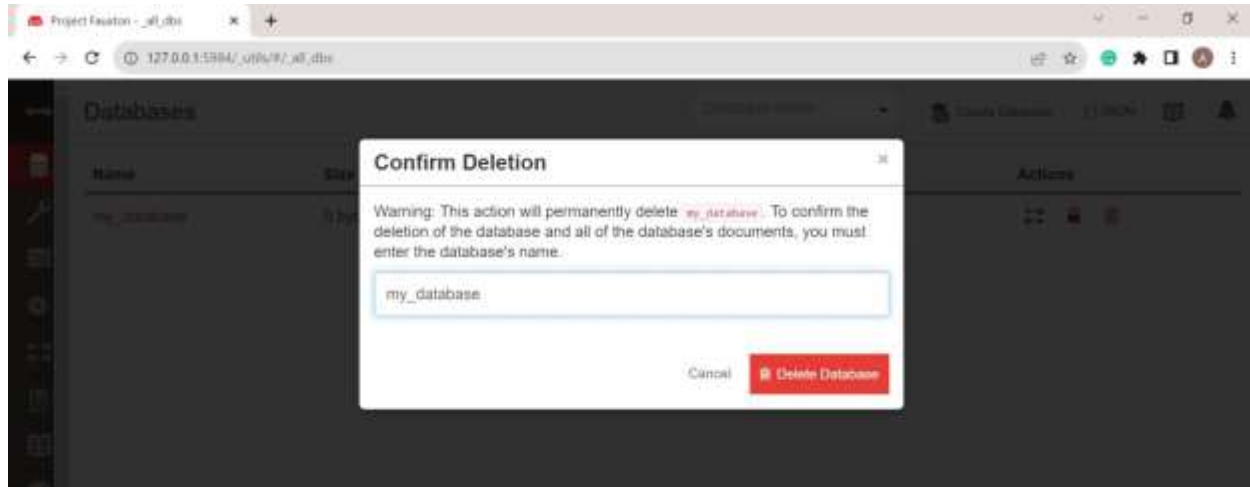
Creating Database



All database are shown created are shown in Futon



Deleting Database



Using command prompt - cURL Utility:

Getting list of Databases

curl -X GET http://admin:123@127.0.0.1:5984/_all_dbs

```
C:\Program Files\Apache CouchDB>curl -X GET http://admin:123@127.0.0.1:5984/_all_dbs
["my_database"]
```

Here admin and 123 is Username and Password created in futon

Creating database

curl -X PUT http://admin:123@127.0.0.1:5984/database name

```
C:\Program Files\Apache CouchDB>curl -X PUT http://admin:123@127.0.0.1:5984/my_db
{"ok":true}
```

Getting info of a particular database

curl -X GET http://admin:123@127.0.0.1:5984/database name

```
C:\Program Files\Apache CouchDB>curl -X GET http://admin:123@127.0.0.1:5984/my_db
```

```
{
  "db_name": "my_db",
  "purge_seq": "0-g1AAAABXeJzLYWBgYMpgTmEQTM4vTc5ISXLIyU90zMnILy7JAUnlsQB8JhgYg9R8IshIZ8KhNZEiqhyjKAgBm",
  "update_seq": "0-g1AAAACbeJzLYWBgYMpgTmEQTM4vTc5ISXLIyU90zMnILy7JAUnlsQB8JhgYg9R8IsjKYExlygQLsyaZpiRaWltj04TEtkSGpHs",
  "i7SUXGQTI2wasgAt3zEL",
  "sizes": {
    "file": 16700,
    "external": 0,
    "active": 0
  },
  "props": {},
  "doc_del_count": 0,
  "doc_count": 0,
  "disk_rmat_version": 8,
  "compact_running": false,
  "cluster": {
    "q": 2,
    "n": 1,
    "w": 1,
    "r": 1
  },
  "instance_start_time": "0"
}
```

Deleting the database

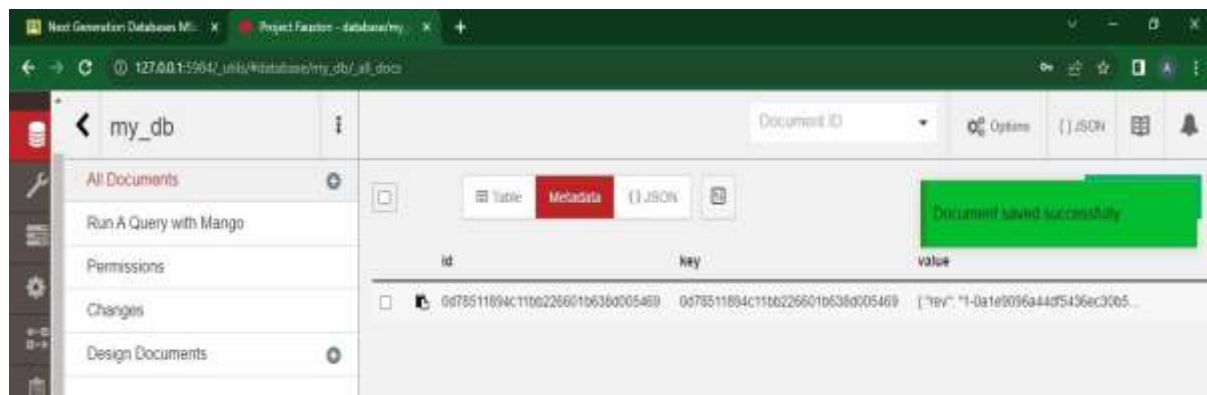
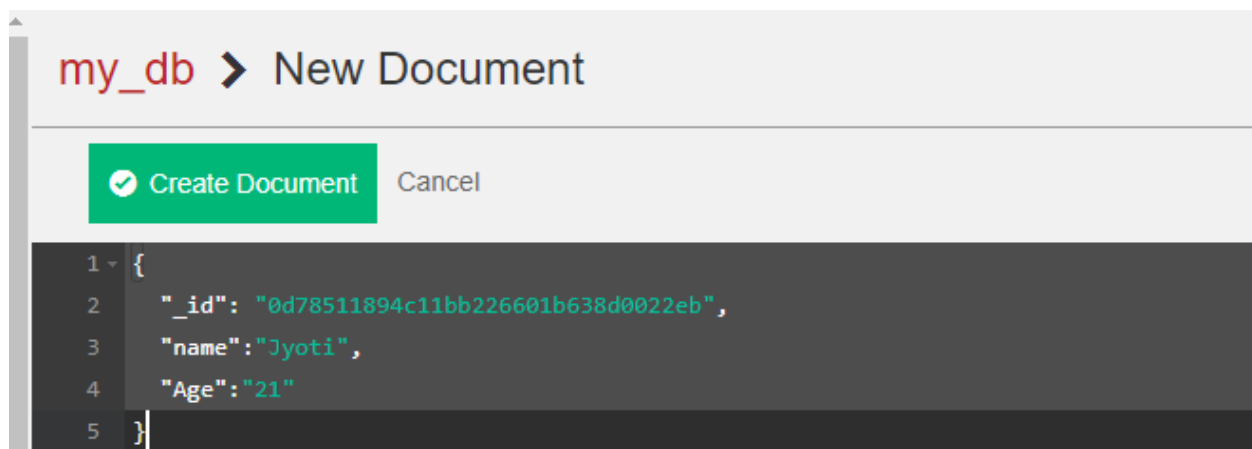
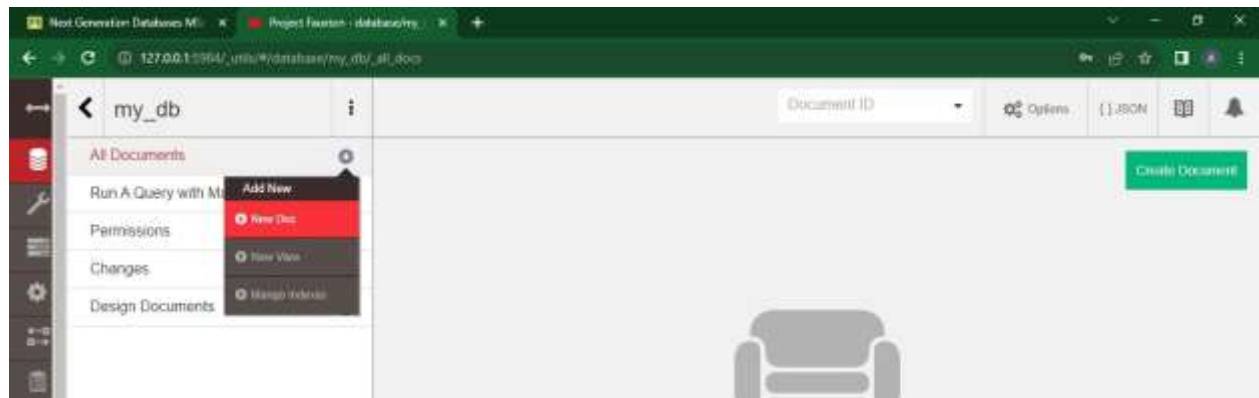
curl -X DELETE http://admin:123@127.0.0.1:5984/database name

```
C:\Program Files\Apache CouchDB>curl -X DELETE http://admin:123@127.0.0.1:5984/college
{"ok":true}
```

Working with Document- Using Futon

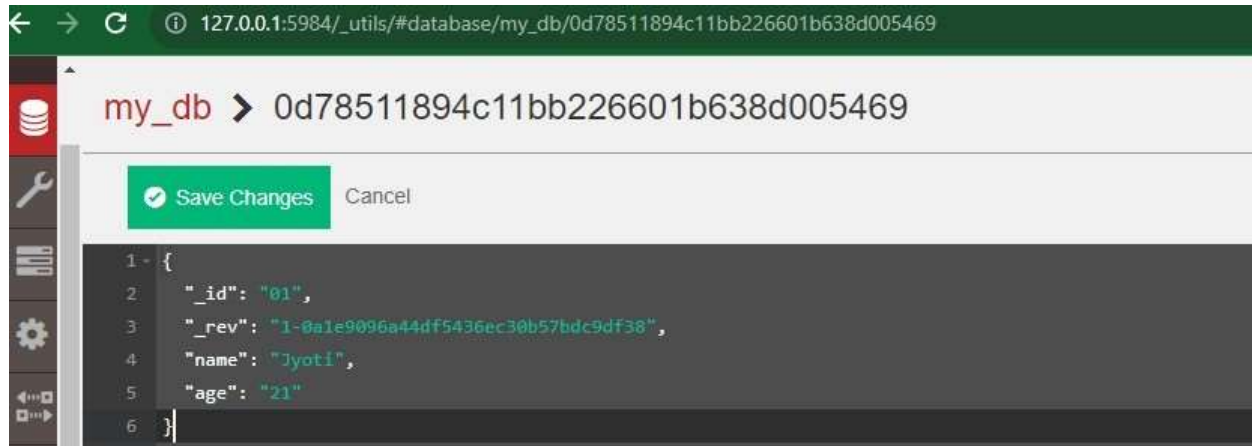
Creating document

Click on the database in which the document is to be created

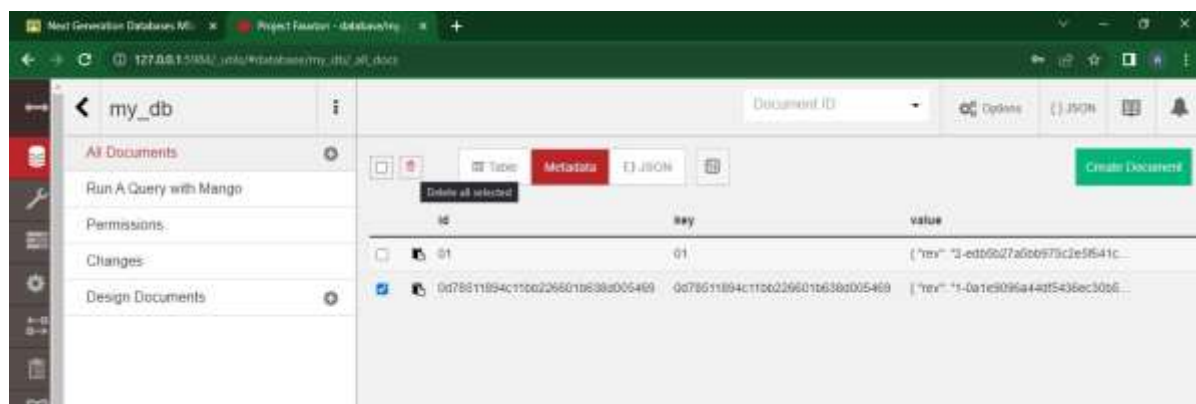


Updating document

Double-click document to update



Deleting Document



Using command prompt - cURL Utility:

Viewing Document

curl -X GET http://admin:123@127.0.0.1:5984/database name/document name

```
C:\Program Files\Apache CouchDB>curl -X GET http://admin:123@127.0.0.1:5984/my_database/001
{"_id":"001","_rev":"5-4f6b69464a528f68753df71876768c24","Name":"Anam","age":21}

C:\Program Files\Apache CouchDB>
```

Creating document

curl -X PUT http://admin:123@127.0.0.1:5984/database name/"document id" -d "{ document}"

```
C:\Program Files\Apache CouchDB>curl -X PUT http://admin:123@127.0.0.1:5984/my_database/"002" -d '{"Age":21}'
{"ok":true,"id":"002","rev":"1-e5271fd2545589db453cfd8895d3a57a"}
```

```
C:\Program Files\Apache CouchDB>curl -X PUT http://admin:123@127.0.0.1:5984/my_database/"003" -d '{"Name":"Shreya","Age":21}'
{"ok":true,"id":"003","rev":"1-ae2afa89df488c2570650911fe7379fa"}
```

Updating Document

First of all, get the revision id of the document that is to be updated. You can find the _rev of the document in the document itself, therefore get the document as shown below.

```
C:\Program Files\Apache CouchDB>curl -X GET http://admin:123@127.0.0.1:5984/my_db/01
{"_id":"01","_rev":"2-edb5b27a5bb975c2e5f641cdaf5fdaef","name":"Jyoti","age":21}
```

curl -X PUT http://admin:123@127.0.0.1:5984/database name/"document id" -d "{ "field:value", "_rev:value"}"

```
C:\Program Files\Apache CouchDB>curl -X PUT http://admin:123@127.0.0.1:5984/my_db/"01" -d '{"age":23,"_rev":"2-edb5b27a5bb975c2e5f641cdaf5fdaef"}'
{"ok":true,"id":"01","rev":"3-af3404530fc097c355331ea8dd0cc3a9"}
```



Deleting Document

Similarly, get revision id for deletion

Then, curl -X DELETE http://admin:123@127.0.0.1:5984/database name/document id?rev= value

```
C:\Program Files\Apache CouchDB>curl -X GET http://admin:123@127.0.0.1:5984/my_db/01
{"_id":"01","_rev":"3-af3404530fc097c355331ea8dd0cc3a9","age":23}

C:\Program Files\Apache CouchDB>curl -X DELETE http://admin:123@127.0.0.1:5984/my_db/01?rev=3-af3404530fc097c355331ea8dd0cc3a9
{"ok":true,"id":"01","rev":"4-088c473f39e291c7f05418f10d62ecae"}
```

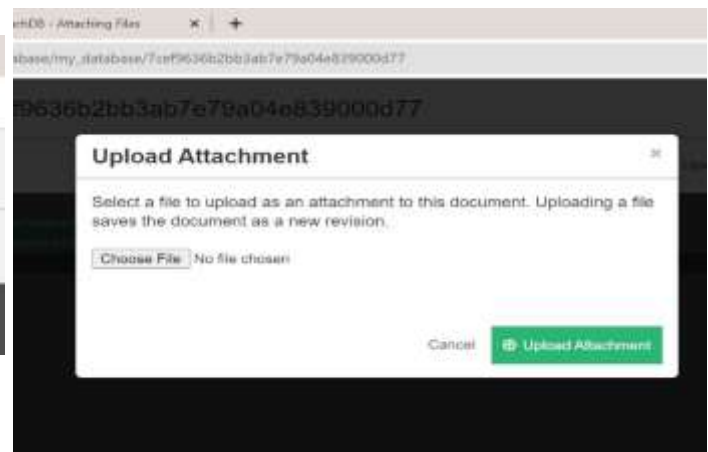
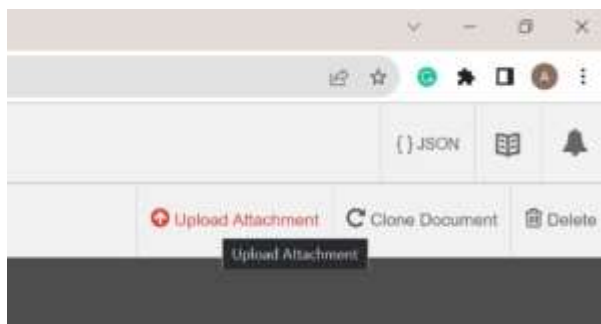
Verify again to check if deleted.

```
C:\Program Files\Apache CouchDB>curl -X GET http://admin:123@127.0.0.1:5984/my_db/01
{"error":"not_found","reason":"deleted"}
```

Attaching file-

Using Futon

Open the document you want to attach file:



PRACTICAL 8

Aim: Connect Java with CouchDB and perform inserting, retrieving, updating and deleting.

Create a Java application on Netbeans IDE and import necessary jar files. Code for creation of a database with a document and viewing the data:

```
package javaapplication3;

import java.io.InputStream;
import java.net.MalformedURLException; import org.ektorp.CouchDbConnector; import
org.ektorp.CouchDbInstance; import org.ektorp.http.HttpClient;
import org.ektorp.http.StdHttpClient;
import org.ektorp.impl.StdCouchDbConnector; import org.ektorp.impl.StdCouchDbInstance; import
org.ektorp.support.DesignDocument; import java.util.Map;
import org.ektorp.Options; public class JavaApplication3 {
public static void main(String[] args) throws MalformedURLException {
//.....Creating Connection...- //          -
HttpClient httpClient = new StdHttpClient.Builder().url("http://admin:123@localhost:5984").build();
CouchDbInstance dbInstance = new StdCouchDbInstance(httpClient);

//.....Creating database.....// CouchDbConnector db = new StdCouchDbConnector("college",
dbInstance);db.createDatabaseIfNotExists();

//.....Creating Document.....-//DesignDocument dd = new DesignDocument("light5");
dd.setAnonymous("address", "Panvel"); dd.setAnonymous("name", "Suraj");
dd.setAnonymous("RollNo", 3144);
db.create(dd);

//print the current attributes Map m=dd.getAnonymous();
System.out.println("Data is "+m);
```

```
// get the list of all documents System.out.println(db.getAllDocIds());

//get name of current document as a idSystem.out.println(dd.getId());

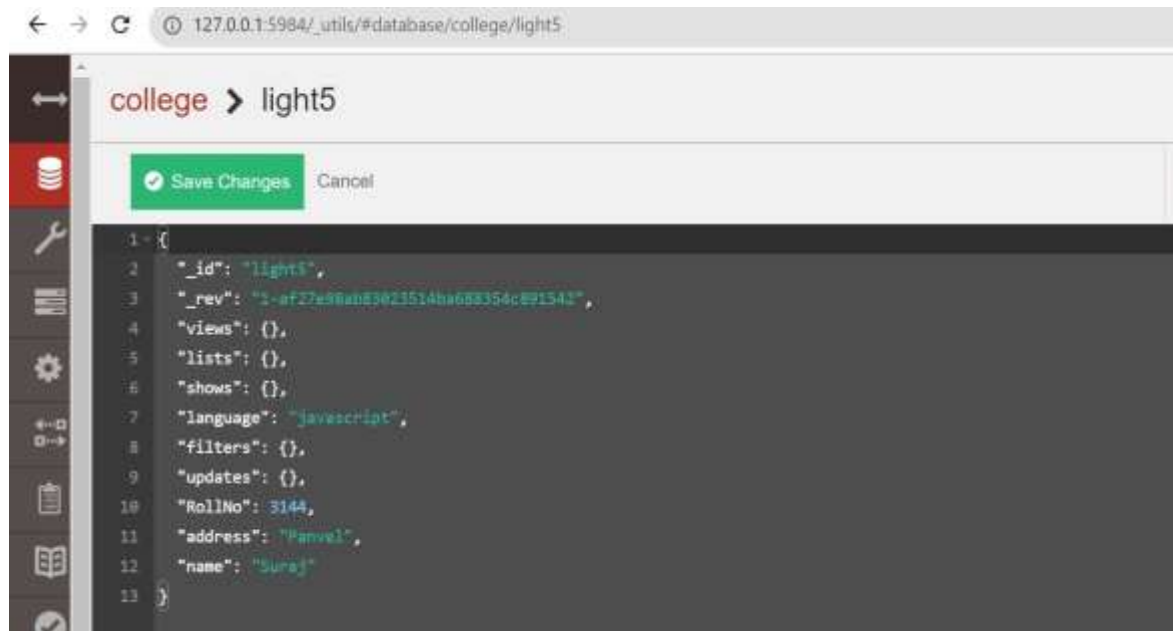
}

}
```

```
20:31:56.041 [main] DEBUG o.e.impl.StreamingJsonSerializer - {"views":
Data is {RollNo=3144, address=Panvel, name=Suraj}
[light5]
light5
BUILD SUCCESSFUL (total time: 1 second)
```

Database college created with a document 'light5'. The data of the document is retrieved.





Code insertion of another document, retrieving it and deleting the previous:

```
package javaapplication3;
```

```
import java.io.InputStream;
import java.net.MalformedURLException;import org.ektorp.CouchDbConnector; import
org.ektorp.CouchDbInstance; import org.ektorp.http.HttpClient;
import org.ektorp.http.StdHttpClient;
import org.ektorp.impl.StdCouchDbConnector;import org.ektorp.impl.StdCouchDbInstance; import
org.ektorp.support.DesignDocument; import java.util.Map;
import org.ektorp.Options; public class JavaApplication3 {
public static void main(String[] args) throws MalformedURLException {
//.....Creating Connection...- //
HttpClient httpClient = new StdHttpClient.Builder().url("http://admin:123@localhost:5984").build();
CouchDbInstance dbInstance = new StdCouchDbInstance(httpClient);

//.....Creating database.....// CouchDbConnector db = new StdCouchDbConnector("college",
dbInstance);db.createDatabaseIfNotExists();

//.....Creating Document.....-//DesignDocument dd = new DesignDocument("light6");
dd.setAnonymous("address", "Kamothe"); dd.setAnonymous("name", "Priya");
```

```

dd.setAnonymous("RollNo", 3168);db.create(dd);

//print the current attributes Map m=dd.getAnonymous();
System.out.println("Data light6 is "+m);

// get the list of all documents System.out.println(db.getAllDocIds());

//get name of current document as a idSystem.out.println(dd.getId());

//delete document
db.delete("light5", "1-af27e98ab83023514ba688354c891542"); //delete the document
System.out.println("Data light5 deleted");

}
}

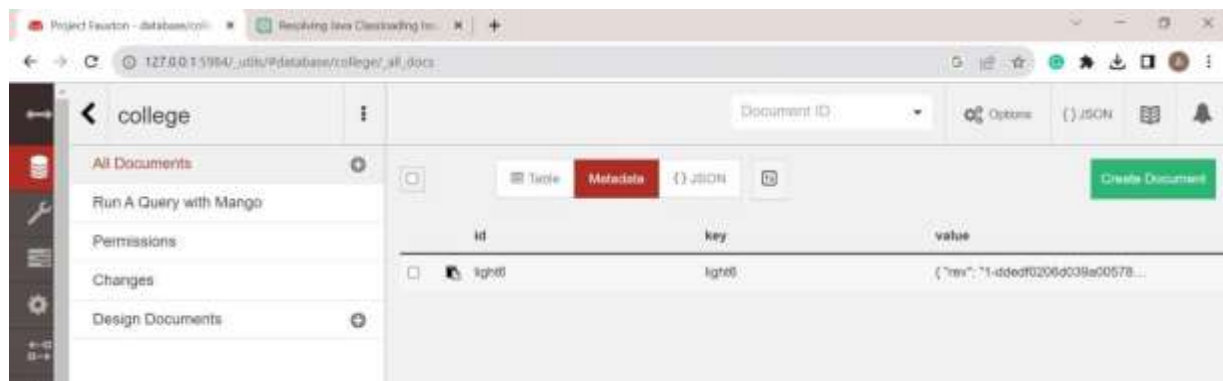
```

```

Data light6 is {RollNo=3168, address=Kamothe, name=Priya}
[light5, light6]
light6
Data light5 deleted
BUILD SUCCESSFUL (total time: 1 second)

```

Document 'ligh6' created and 'light5' deleted.



PRACTICAL 9

Aim: Neo4j

```
neo4j$ create (node)
```

Table

Code

Created 1 node, completed after 13 ms.

```
create(Anam:student{rollno:3116,name:"Anam",add:"Kharghar"}) return Anam
```

Graph

Table

Text

Code

Anam

Overview

Node labels

* (1) student (1)

Displaying 1 nodes, 0 relationships.

```
neo4j$ create(Shreya:student{rollno:3126,name:"Shreya",add:"Kamothe"})...
```

Graph

Table

Text

Code

Shreya

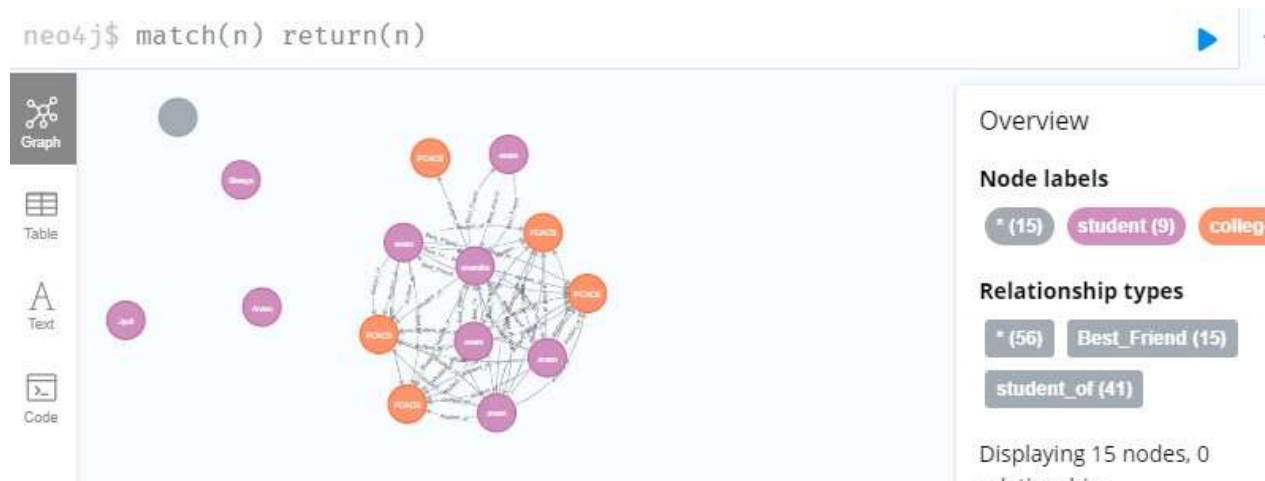
Overview

Node labels

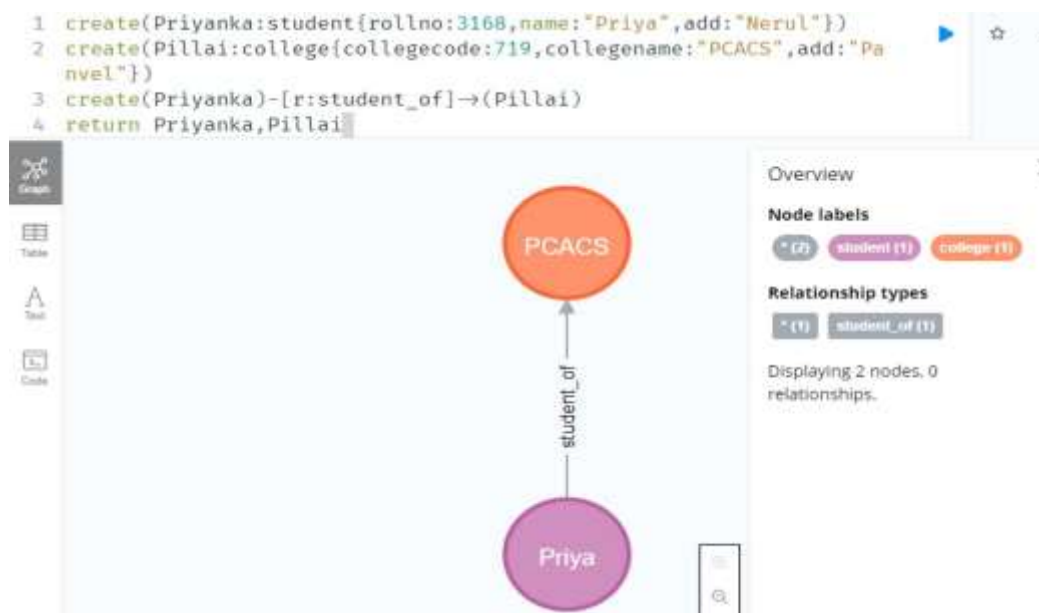
* (1) student (1)

Displaying 1 nodes, 0 relationships.

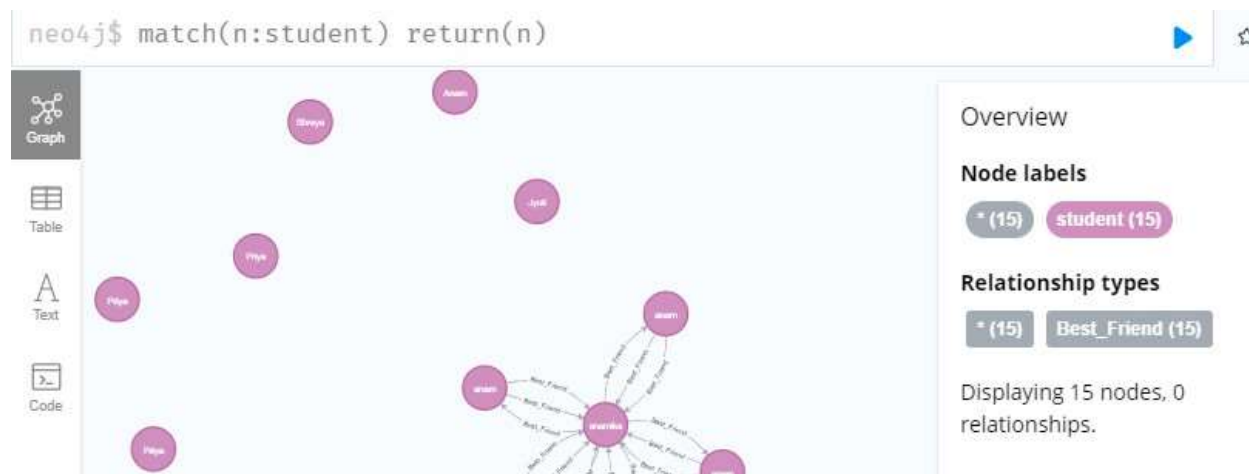
```
match(n) return(n)
```



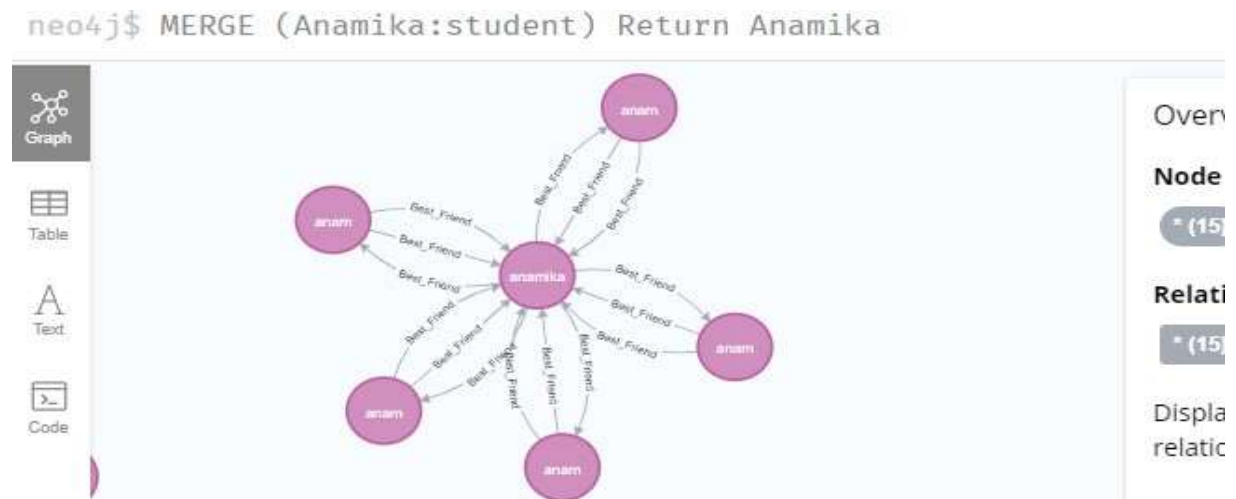
```
create(Priyanka:student{rollno:3168,name:"Priya",add:"Nerul"})
create(Pillai:college{collegecode:719,collegename:"PCACS",add:"Panvel"})
create(Priyanka)-[r:student_of]->(Pillai)
return Priyanka,Pillai
```



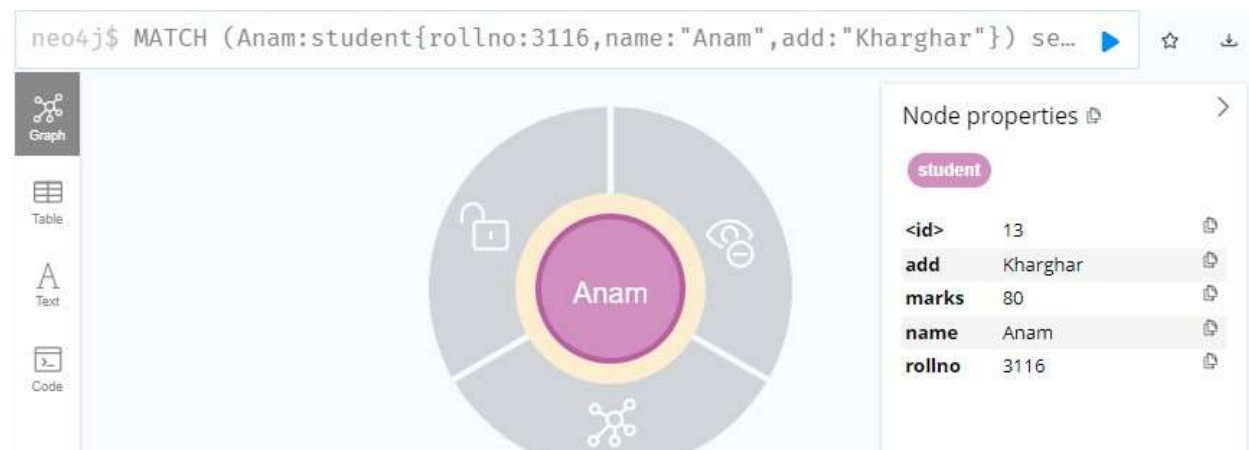
```
match(n:student) return(n)
```

MERGE (Anamika:student) Return Anamika



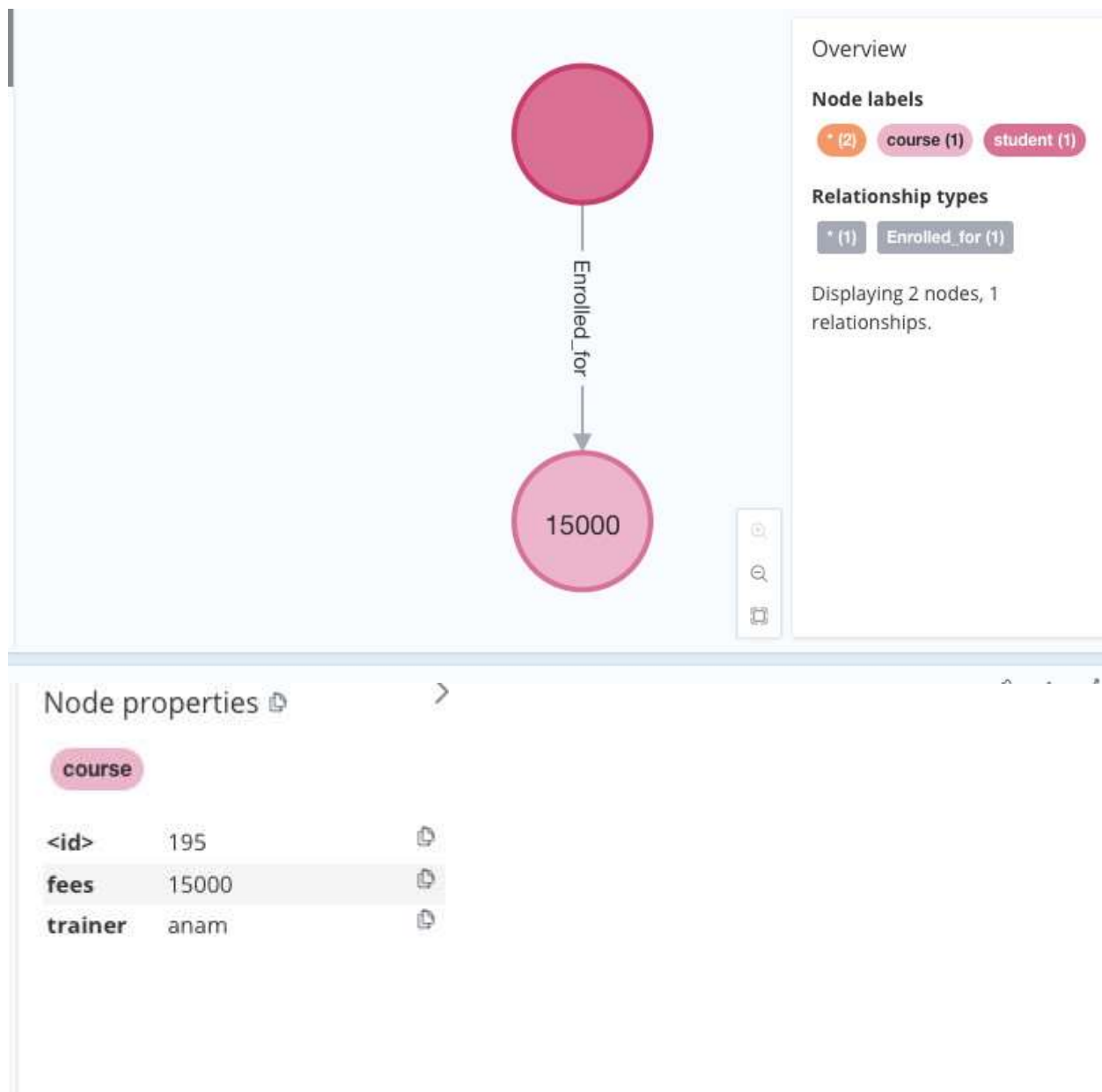
MATCH (Anam:student{rollno:3116,name:"Anam",add:"Kharghar"}) set Anam.marks=80
RETURN Anam



MATCH(shreya:Mscda_student{rollno:101}) delete shreya

Deleted 1 node, completed after 3 ms.

```
CREATE (ml:course {fees: 15000, trainer: 'anam'})
CREATE (rachn:student {rollno: 1, name: 'rachn'})
CREATE (rachn)-[r:Enrolled_for]->(ml)
RETURN ml, rachn, r
```

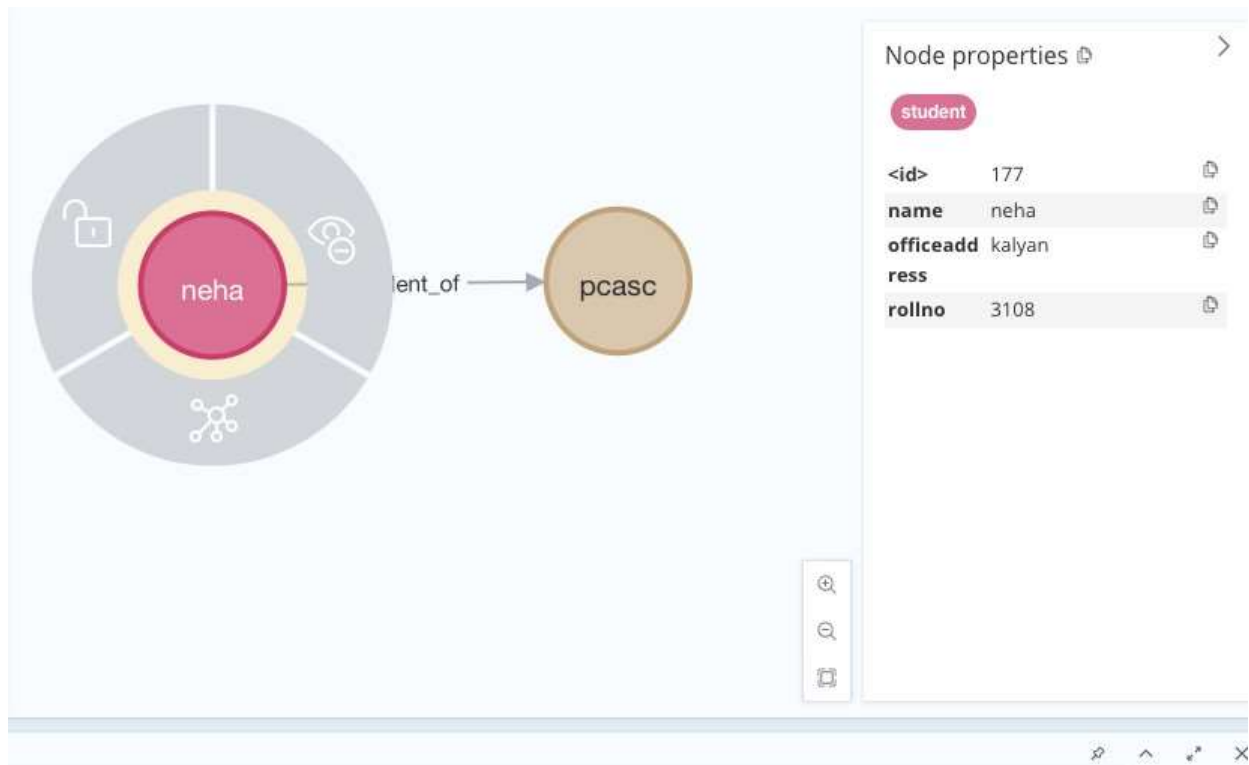


MATCH (n:student) WHERE n.rollno > 1

RETURN n
ORDER BY n.address



MATCH (n:student) WHERE n.rollno > 1
RETURN n
ORDER BY n.address
LIMIT 3



Match (n:student) with n
RETURN collect(n.rollno)

RETURN n
ORDER BY n.address

collect(n.rollno)	
1	[3108, 3108, 3108, 3108, 3108, 3108, 3108, 3108, 1, 1]

MATCH (n:student) Return trim(n.name)

5	"neha"
6	"neha"
7	"neha"
8	"neha"
9	"rachn"
10	"rachn"

MATCH(n:student) RETURN avg (n.rollno)

avg (n.rollno)	
1	2486.6

MATCH(n:student) RETURN max (n.rollno)

RETURN n
ORDER BY n.address

MATCH(n:student) RETURN max (n.rollno)

max (n.rollno)
3108

MATCH(n:student) RETURN count (n.rollno)

count (n.rollno)
10

MATCH(n:student) RETURN sum (n.rollno)

sum (n.rollno)
24866

MATCH (n:student)
RETURN stdev(n.rollno) AS standard_deviation

standard_deviation
1310.0262253524206