

Experiment No: 1**Aim**

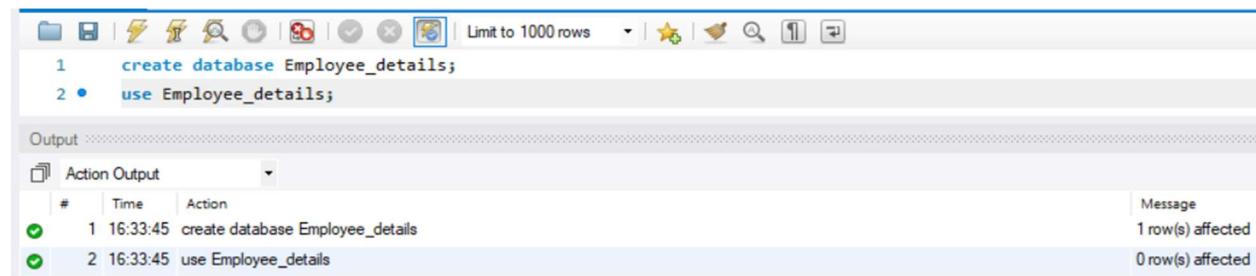
Creation of a database using DDL commands.

CO1

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

1. Create database
2. Use database



The screenshot shows the MySQL Workbench interface. In the top query editor, two statements are run:

```

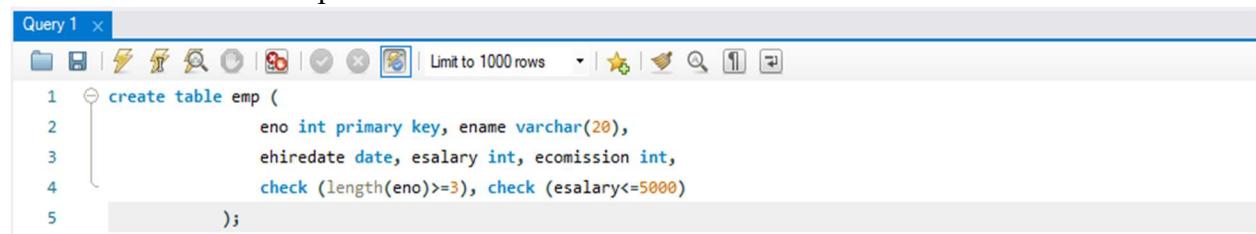
1  create database Employee_details;
2 • use Employee_details;

```

In the bottom 'Action Output' pane, the results of these statements are shown:

#	Time	Action	Message
1	16:33:45	create database Employee_details	1 row(s) affected
2	16:33:45	use Employee_details	0 row(s) affected

3. Create table -emp



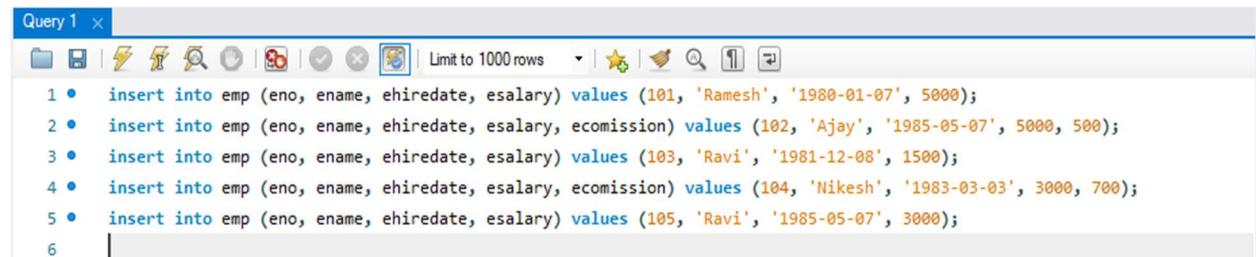
The screenshot shows the MySQL Workbench interface. A single statement creates a table 'emp' with specific constraints:

```

1 • create table emp (
    eno int primary key, ename varchar(20),
    ehiredate date, esalary int, ecommission int,
    check (length(eno)>=3), check (esalary<=5000)
);

```

4. Insert values into table



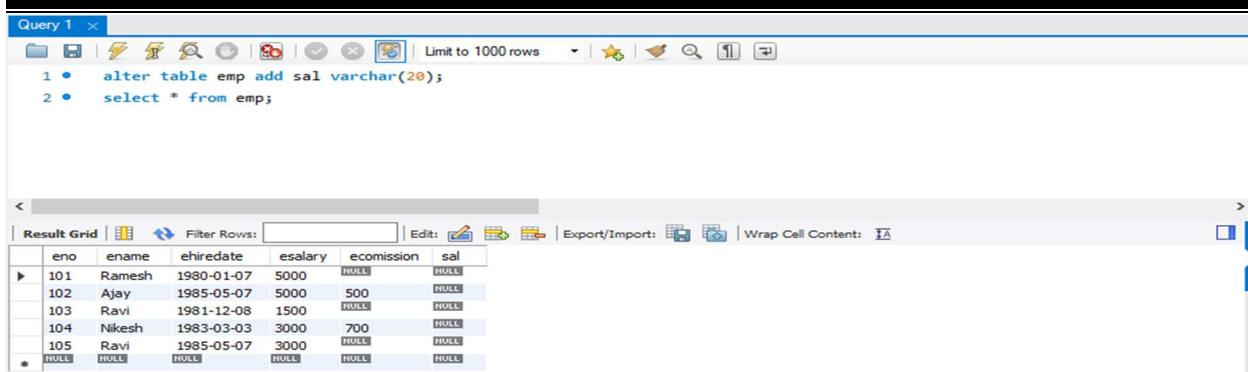
The screenshot shows the MySQL Workbench interface. Five insert statements are run into the 'emp' table:

```

1 • insert into emp (eno, ename, ehiredate, esalary) values (101, 'Ramesh', '1980-01-07', 5000);
2 • insert into emp (eno, ename, ehiredate, esalary, ecommission) values (102, 'Ajay', '1985-05-07', 5000, 500);
3 • insert into emp (eno, ename, ehiredate, esalary) values (103, 'Ravi', '1981-12-08', 1500);
4 • insert into emp (eno, ename, ehiredate, esalary, ecommission) values (104, 'Nikesh', '1983-03-03', 3000, 700);
5 • insert into emp (eno, ename, ehiredate, esalary) values (105, 'Ravi', '1985-05-07', 3000);
6 |

```

5. Add column-SAL



Query 1

```

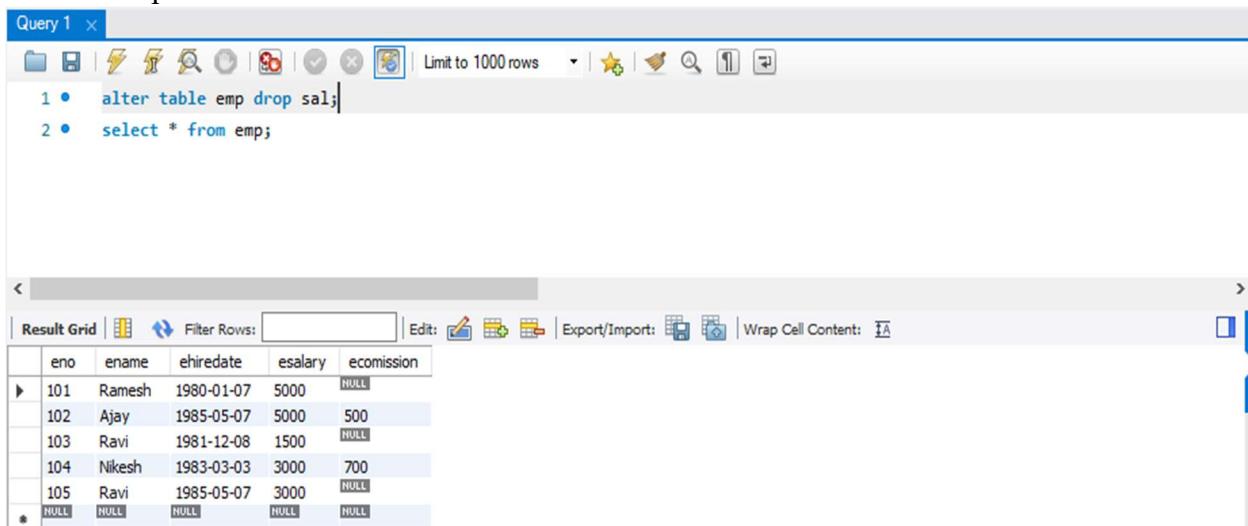
1 • alter table emp add sal varchar(20);
2 • select * from emp;

```

Result Grid

eno	ename	ehiredate	esalary	ecommission	sal
101	Ramesh	1980-01-07	5000	NULL	NULL
102	Ajay	1985-05-07	5000	500	NULL
103	Ravi	1981-12-08	1500	NULL	NULL
104	Nikesh	1983-03-03	3000	700	NULL
105	Ravi	1985-05-07	3000	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL

6. Drop column-SAL



Query 1

```

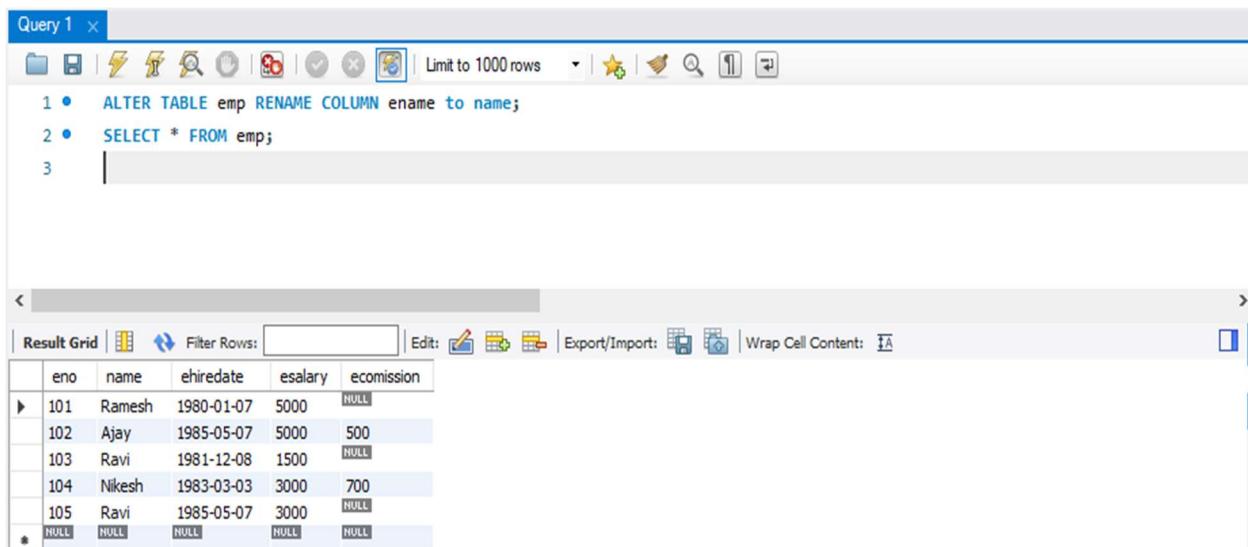
1 • alter table emp drop sal;
2 • select * from emp;

```

Result Grid

eno	ename	ehiredate	esalary	ecommission
101	Ramesh	1980-01-07	5000	NULL
102	Ajay	1985-05-07	5000	500
103	Ravi	1981-12-08	1500	NULL
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-05-07	3000	NULL
*	NULL	NULL	NULL	NULL

7. Rename column- ename to name



Query 1

```

1 • ALTER TABLE emp RENAME COLUMN ename to name;
2 • SELECT * FROM emp;
3 |

```

Result Grid

eno	name	ehiredate	esalary	ecommission
101	Ramesh	1980-01-07	5000	NULL
102	Ajay	1985-05-07	5000	500
103	Ravi	1981-12-08	1500	NULL
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-05-07	3000	NULL
*	NULL	NULL	NULL	NULL

8. Rename table

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The code entered is:

```
1 • Alter table emp rename to emp_details;
2 • show tables;
```

The results grid shows the output of the "show tables" command, listing "Tables_in_office" and "emp_details".

9. Truncate table

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The code entered is:

```
1 • truncate emp_details;
2 • SELECT * FROM emp_details;
3
```

The results grid shows the output of the "SELECT * FROM emp_details" command, displaying columns eno, name, ehiredate, esalary, and ecommission, all of which are NULL.

10. Drop table

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The code entered is:

```
1 • drop table emp_details;
2 • SELECT * FROM emp_details;
```

The results grid shows the output of the "drop table emp_details" command, indicating 0 row(s) affected. The next line shows the result of the "SELECT * FROM emp_details" command, with an error message: "Error Code: 1146. Table 'office.emp_details' doesn't exist".

Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

Experiment No: 2

Aim

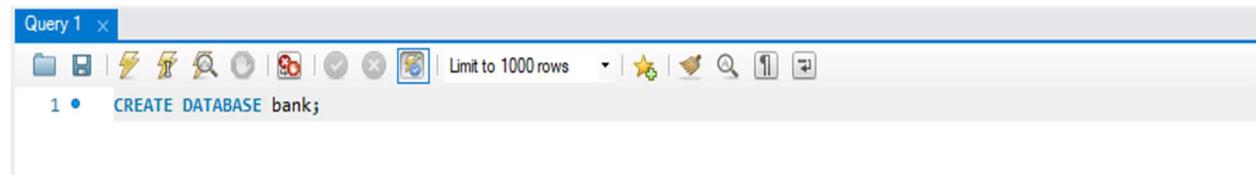
Creation of a database using DML commands including integrity constraints.

CO2

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

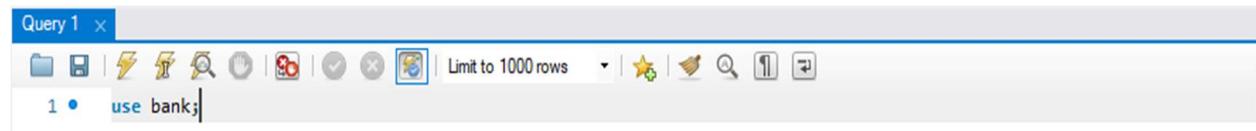
Procedure

1. Create database



```
Query 1 ×
CREATE DATABASE bank;
```

2. Use database



```
Query 1 ×
use bank;
```

3.Create table - Customer

Query 1 x

```

1 • CREATE TABLE BRANCH (
2     BNAME VARCHAR(20) PRIMARY KEY,
3     CITY VARCHAR(30) CHECK(CITY IN ('NAGPUR', 'DELHI', 'BANGLORE', 'BOMBAY')) NOT NULL
4 );
5

```

4. Create table - Customer

Query 1 x

```

1 • CREATE TABLE CUSTOMER (
2     CNAME VARCHAR(15) PRIMARY KEY,
3     CITY VARCHAR(30) NOT NULL
4 );
5

```

5.Create table - Deposit

Query 1 x

```

1 • CREATE TABLE DEPOSIT (
2     ACTNO VARCHAR(5) PRIMARY KEY CHECK (ACTNO LIKE 'D%'),
3     CNAME VARCHAR(15) REFERENCES CUSTOMER(CNAME),
4     BNAME VARCHAR(20) REFERENCES BRANCH(BNAME),
5     AMOUNT DECIMAL(8,2) NOT NULL CHECK (AMOUNT > 0),
6     ADATE DATE
7 );
8

```

6. Create table - Borrow

Query 1 x

```

1 • CREATE TABLE BORROW (
2     LOANNO VARCHAR(8) check(LOANNO like 'L%') primary key,
3     CNAME VARCHAR(15) REFERENCES CUSTOMER(CNAME),
4     BNAME VARCHAR(20) REFERENCES BRANCH(BNAME),
5     AMOUNT FLOAT(8) NOT NULL CHECK (AMOUNT > 0)
6 );
7

```

7. Insert values in table - Customer

Query 1

```

1 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('ANIL', 'CALCUTTA');
2 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SUNIL', 'DELHI');
3 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MEHUL', 'BARODA');
4 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MANDAR', 'PATNA');
5 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MADHURI', 'NAGPUR');
6 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('PRAMOD', 'NAGPUR');
7 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SANDIP', 'SURAT');
8 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SHIVANI', 'BOMBAY');
9 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('KRANTI', 'BOMBAY');
10 • INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('NAREN', 'BOMBAY');
11

```

8. Insert values in table - Branch

Query 1

```

1 • INSERT INTO BRANCH (BNAME, CITY) VALUES ('VRCE', 'NAGPUR');
2 INSERT INTO BRANCH (BNAME, CITY) VALUES ('AJNI', 'NAGPUR');
3 INSERT INTO BRANCH (BNAME, CITY) VALUES ('KAROLBAGH', 'DELHI');
4 INSERT INTO BRANCH (BNAME, CITY) VALUES ('CHANDNI', 'DELHI');
5 INSERT INTO BRANCH (BNAME, CITY) VALUES ('DHARAMPETH', 'NAGPUR');
6 INSERT INTO BRANCH (BNAME, CITY) VALUES ('MG ROAD', 'BANGLORE');
7 INSERT INTO BRANCH (BNAME, CITY) VALUES ('ANDHERI', 'BOMBAY');
8 INSERT INTO BRANCH (BNAME, CITY) VALUES ('NEHRU PALACE', 'DELHI');
9 INSERT INTO BRANCH (BNAME, CITY) VALUES ('POWAI', 'BOMBAY');
10

```

9. Insert values in table - Borrow

Query 1

```

1 • INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L201', 'ANIL', 'VRCE', 1000.00);
2 • INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L206', 'MEHUL', 'AJNI', 5000.00);
3 • INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L311', 'SUNIL', 'DHARAMPETH', 3000.00);
4 • INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L321', 'MADHURI', 'ANDHERI', 2000.00);
5 • INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L371', 'PRAMOD', 'VIRAR', 8000.00);
6 • INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L481', 'KRANTI', 'NEHRU PLACE', 3000.00);
7

```

10. Insert values into table - Deposit

Query 1

```

1 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D100', 'ANIL', 'VRCE', 1000.00, '1995-03-01');
2 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D101', 'SUNIL', 'AJNI', 500.00, '1996-01-04');
3 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D102', 'MEHUL', 'KAROLBAGH', 3500.00, '1995-11-17');
4 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D104', 'MADHURI', 'CHANDNI', 1200.00, '1995-12-17');
5 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D105', 'PRAMOD', 'MG ROAD', 3000.00, '1996-03-27');
6 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D106', 'SANDIP', 'ANDHERI', 2000.00, '1996-03-31');
7 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D107', 'SHIVANI', 'VIRAR', 1000.00, '1995-09-05');
8 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D108', 'KRANTI', 'NEHRU PLACE', 5000.00, '1995-07-02');
9 • INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D109', 'MINU', 'POWAI', 7000.00, '1995-08-10');
10

```

11. Add constraint - salary <= 5000

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:alter table emp add constraint chk_salary check (esalary >= 1500);

```
This query adds a check constraint named "chk_salary" to the "emp" table, which ensures that the value in the "esalary" column is greater than or equal to 1500.
```

12. Add constraint - length(eno) >= 3

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:alter table emp add constraint chk_empno_length check (length(eno)>=3);

```
This query adds a check constraint named "chk_empno_length" to the "emp" table, which ensures that the length of the value in the "eno" column is greater than or equal to 3.
```

Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

Experiment No: 3

Aim

To familiarize with selecting data from single table

CO1

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

1. List all data from table deposit.

```
select * from deposit;
```

	ACTNO	CNAME	BNAME	AMOUNT	ADATE
▶	D100	ANIL	VRCE	1000.00	1995-03-01
	D101	SUNIL	ANJINI	500.00	1996-01-04
	D102	MEHUL	KAROLBAGH	3500.00	1995-11-17
	D104	MADHURI	CHANDNI	1200.00	1995-12-17
	D105	PRAMOD	MG ROAD	3000.00	1996-03-27
	D106	SANDIP	ANDHERI	2000.00	1996-03-31
	D107	SHIVANI	VIRAR	1000.00	1995-09-05
	D108	KRANTI	NEHRU PLACE	5000.00	1995-07-02
	D109	MINU	POWAI	7000.00	1995-08-10
*	NULL	NULL	NULL	NULL	NULL

2. List all data from borrow

```
select * from borrow;
```

	LOANNO	CNAME	BNAME	AMOUNT
▶	L201	ANIL	VRCE	1000
	L206	MEHUL	AJINI	5000
	L311	SUNIL	DHARAMPETH	3000
	L321	MADHURI	ANDHERI	2000
	L371	PRAMOD	VIRAR	8000
*	L481	KRANTI	NEHRU PLACE	3000
*	NULL	NULL	NULL	NULL

3. List all data from customer

```
select * from customer;
```

	CNAME	CITY
▶	ANIL	CALCUTTA
	KRANTI	BOMBAY
	MADHURI	NAGPUR
	MANDAR	PATNA
	MEHUL	BARODA
	NAREN	BOMBAY
	PRAMOD	NAGPUR
	SANDIP	SURAT
	SHIVANI	BOMBAY
*	SUNIL	DELHI
*	NULL	NULL

4. List all data from branch

```
select * from branch;
```

BNAME	CITY
AJNI	NAGPUR
ANDHERI	BOMBAY
CHANDNI	DELHI
DHARAMPETH	NAGPUR
KAROLBAGH	DELHI
MG ROAD	BANGLORE
NEHRU PALACE	DELHI
POWAI	BOMBAY
VRCE	NAGPUR
NULL	NULL

5. Give account no and amount of deposit
 select actno ,amount from deposit;

actno	amount
D100	1000.00
D101	500.00
D102	3500.00
D104	1200.00
D105	3000.00
D106	2000.00
D107	1000.00
D108	5000.00
D109	7000.00

6. Give customer name and account no of depositors
 select actno ,cname from deposit;

CNAME	ACTNO
ANIL	D100
SUNIL	D101
MEHUL	D102
MADHURI	D104
PRAMOD	D105
SANDIP	D106
SHIVANI	D107
KRANTI	D108
MINU	D109
NULL	NULL

7. Give name of customers
 select cname from customer;

CNAME
ANIL
KRANTI
MADHURI
MANDAR
MEHUL
NAREN
PRAMOD
SANDIP
SHIVANI
SUNIL
NULL

8. Give name of branches
 select bname from branch;

Result Grid	
Filter Rows:	
Edit:	
Export/Import:	
Wrap Cell Content:	
BNAMES	
▶ AJNI	
ANDHERI	
CHANDNI	
DHARAMPETH	
KAROLBAGH	
MG ROAD	
NEHRU PALACE	
POWAI	
VRCE	
* NULL	

9. Give name of borrows

select cname from borrow;

Result Grid	
Filter Rows:	
Export:	
Wrap Cell Content:	
CNAME	
▶ ANIL	
MEHUL	
SUNIL	
MADHURI	
PRAMOD	
KRANTI	

10. Give names of customer living in city Nagpur

select cname from customer where city="nagpur";

Result Grid	
Filter Rows:	
Edit:	
Export/Import:	
Wrap Cell Content:	
CNAME	
▶ MADHURI	
PRAMOD	
* NULL	

11. Give names of depositors having amount greater than 4000

select cname from deposit where amount>4000;

Result Grid	
Filter Rows:	
Export:	
Wrap Cell Content:	
CNAME	
▶ KRANTI	
MINU	

12. Give account date of Anil

select adate from deposite where cname="anil";

Result Grid	
Filter Rows:	
ADATE	1995-03-01

13. Give name of all branches located in Bombay

select bname from branch where city="bombay";

Result Grid	
Filter Rows:	
Edit:	
Export/Import:	
Wrap Cell Content:	
BNAME	
▶ ANDHERI	
POWAI	
* NULL	

14. Give name of borrower having loan number l205
 select cname from borrow where loanno="l205";

CNAME
ANIL

15. Give names of depositors having account at VRCE
 select cname from deposite where bname="vrce";

CNAME
ANIL

16. Give names of all branched located in city Delhi
 select bname from branch where city="delhi";

BNAME
CHANDNI
KAROLBAGH
NEHRU PALACE
HULL

17. Give name of the customers who opened account date '1-12-96'
 select cname from deposite where adate='1996-12-01';

CNAME
ANIL

18. Give account no and deposit amount of customers having account opened between dates '1- 12-96' and '1-5-96'
 select actno , cname from deposite where adate between '1996-12-01' and '1996-05-01';

ACTNO	AMOUNT
HULL	HULL

19. Give name of the city where branch KAROL BAGH is located
 select city from branch where bname="karolbagh";

CITY
DELHI

20. Give details of customer ANIL
 select * from customer where cname="anil";

ACTNO	CNAME	BNAME	AMOUNT	ADATE
D100	ANIL	VRCE	1000.00	1995-03-01
HULL	HULL	HULL	HULL	HULL

Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

Experiment No: 4

Aim

To familiarize with Set Operations.

CO1

Design and build a simple relational database system and demonstrate competence with the fundamental tasks involved with modeling, designing and implementing a database.

Procedure

1. List all the customers who are depositors but not borrowers.

```
select cname from deposit where cname not in (select cname from borrow);
```

Result Grid	
<input type="checkbox"/>	CNAME
<input checked="" type="checkbox"/>	SANDIP
<input checked="" type="checkbox"/>	SHIVANI
<input checked="" type="checkbox"/>	MINU

2. List all the customers who are both depositors and borrowers.

```
select cname from deposit union (select cname from borrow);
```

Result Grid	
<input type="checkbox"/>	CNAME
<input checked="" type="checkbox"/>	ANIL
<input checked="" type="checkbox"/>	SUNIL
<input checked="" type="checkbox"/>	MEHUL
<input checked="" type="checkbox"/>	MADHURI
<input checked="" type="checkbox"/>	PRAMOD
<input checked="" type="checkbox"/>	SANDIP
<input checked="" type="checkbox"/>	SHIVANI
<input checked="" type="checkbox"/>	KRANTI
<input checked="" type="checkbox"/>	MINU

3. List all the depositors having deposit in all the branches where Sunil is having Account

```
select D1.cname from deposit D1 where D1.bname in (select D2.bname from deposit
where D2.cname="sunil");
```

Result Grid	
<input type="checkbox"/>	CNAME
<input checked="" type="checkbox"/>	SUNIL

4. List all the customers living in city NAGPUR and having branch city BOMBAY or DELHI

Query 1

```
1 •  SELECT C1.CNAME FROM CUSTOMER C1,DEPOSIT D1, BRANCH B1 WHERE C1.CITY = 'NAGPUR' AND
2     C1.CNAME = D1.CNAME AND D1.BNAME = B1.BNAME AND B1.CITY IN ('BOMBAY','DELHI');
3
```

Result Grid

CNAME
MADHURI

5. List all the depositors living in city NAGPUR

Query 1

```
1 •  SELECT DISTINCT(CUSTOMER.CNAME) from CUSTOMER,DEPOSIT WHERE City='NAGPUR';
2
```

Result Grid

CNAME
PRAMOD
MADHURI

6. List all the depositors living in the city NAGPUR and having branch in city BOMBAY

Query 1

```
1 •  SELECT C1.CNAME FROM CUSTOMER C1,DEPOSIT D1, BRANCH B1 WHERE C1.CITY = 'NAGPUR' AND C1.CNAME = D1.CNAME AND
2     D1.BNAME = B1.BNAME AND B1.CITY IN ('BOMBAY');
```

Result Grid

CNAME

7. List the branch cities of Anil and Sunil

Query 1

```
1 •  SELECT B1.CITY FROM DEPOSIT D1, BRANCH B1 WHERE D1.BNAME = B1.BNAME AND D1.CNAME IN ('SUNIL','ANIL');
2
```

Result Grid

CITY
NAGPUR

8. List the customers having deposit greater than 1000 and loan less than 10000.

Query 1

```
1 •   SELECT DISTINCT D1.CNAME FROM deposit D1, borrow B1 WHERE D1.AMOUNT>1000 AND B1.AMOUNT<10000;
2
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CNAME
MEHUL
MADHURI
PRAMOD
SANDIP
KRANTI
MINU

9. List the cities of depositors having branch VRCE.

Query 1

```
1 •   SELECT B1.CITY FROM deposit D1, branch B1 WHERE D1.BNAME=B1.BNAME AND B1.BNAME='VRCE';
2
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CITY
NAGPUR

10. List the depositors having amount less than 1000 and living in the same city as Anil

Query 1

```
1     SELECT D1.CNAME FROM deposit D1, customer C1 WHERE AMOUNT<1000 AND C1.CITY=(C1.CNAME='ANIL');
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CNAME
SUNIL

11. List all the cities where branches of Anil and Sunil are located

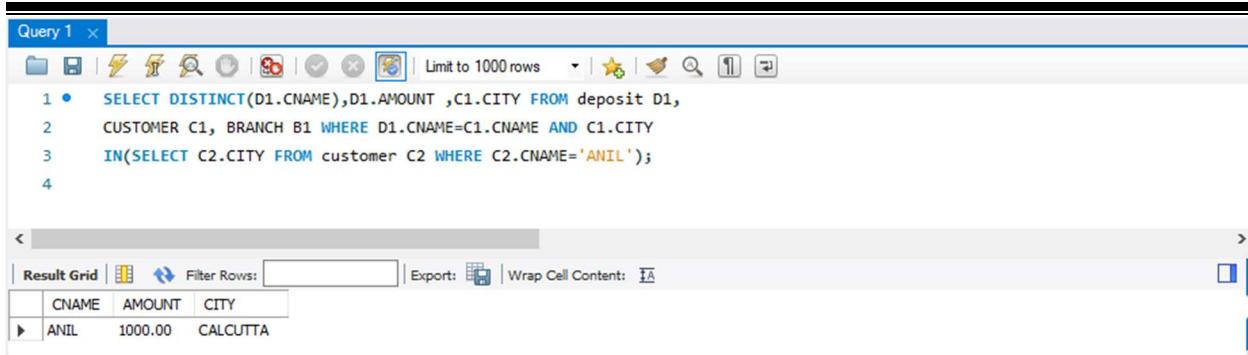
Query 1

```
1     SELECT B1.CITY FROM BRANCH B1 WHERE B1.BNAME IN (SELECT D1.BNAME FROM DEPOSIT D1 WHERE D1.CNAME IN ('ANIL','SUNIL'));
2
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CITY
NAGPUR

12. List the amount for the depositors living in the city where Anil is living



The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```

1 •  SELECT DISTINCT(D1.CNAME),D1.AMOUNT ,C1.CITY FROM deposit D1,
2     CUSTOMER C1, BRANCH B1 WHERE D1.CNAME=C1.CNAME AND C1.CITY
3     IN(SELECT C2.CITY FROM customer C2 WHERE C2.CNAME='ANIL');
4

```

The result grid displays one row:

CNAME	AMOUNT	CITY
ANIL	1000.00	CALCUTTA

Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

Experiment No: 5

Aim

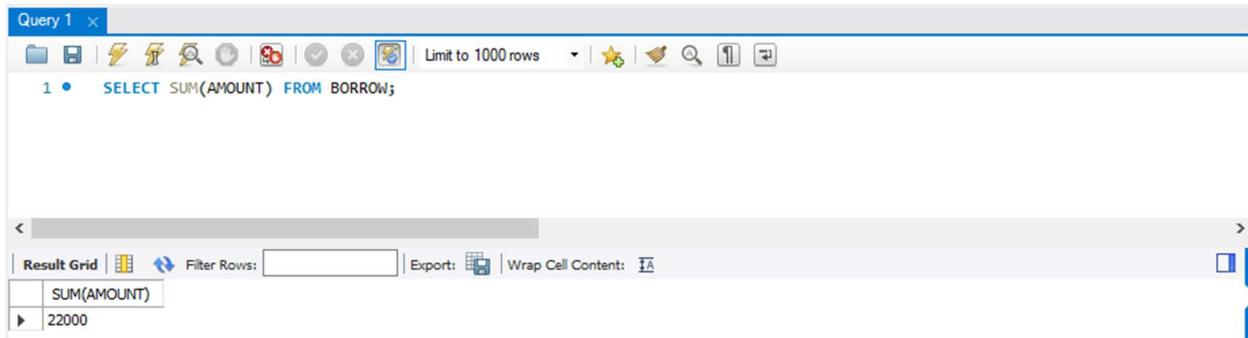
To familiarize with Aggregate Functions.

CO1

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

1. List total loan



The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```

1 •  SELECT SUM(AMOUNT) FROM BORROW;

```

The result grid displays one row:

SUM(AMOUNT)
22000

2. List total deposit

Query 1

```
1   SELECT SUM(AMOUNT) FROM DEPOSIT;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

SUM(AMOUNT)
24200.00

3. List total loan taken from KAROLBAGH branch

Query 1

```
1   SELECT MAX(AMOUNT) FROM BORROW WHERE BNAME = 'KAROLBAGH';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

MAX(AMOUNT)
NULL

4. List total deposit of customers having account date later than 1-Jan-96

Query 1

```
1   SELECT SUM(AMOUNT) from deposit where adate>'1995-03-01';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

SUM(AMOUNT)
23200.00

5. List total deposit of customers living in city NAGPUR

Query 1

```
1   SELECT SUM(D1.AMOUNT) FROM DEPOSIT D1 , CUSTOMER C1 WHERE C1.CITY = 'NAGPUR' AND C1.CNAME = D1.CNAME;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

SUM(D1.AMOUNT)
4200.00

6. List maximum deposit of customer living in Bombay

Query 1

```

1   SELECT MAX(D1.AMOUNT) FROM DEPOSIT D1 , CUSTOMER C1 WHERE C1.CITY
2     = 'Bombay' AND C1.CNAME = D1.CNAME;
3

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MAX(D1.AMOUNT)
5000.00

7. List total deposit of customer having branch in BOMBAY

Query 1

```

1 •  SELECT SUM(AMOUNT) from deposit,BRANCH where city='BOMBAY';

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SUM(AMOUNT)
48400.00

8. Count total number of branch cities

Query 1

```

1   SELECT COUNT(DISTINCT(CITY)) FROM BRANCH ;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(DISTINCT(CITY))
4

9. Count total number of customers cities

Query 1

```

1   SELECT count(city) from CUSTOMER;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(city)
10

10. Give branch names and branch wise deposit

Query 1

```
1   SELECT BNAME , SUM(AMOUNT) FROM DEPOSIT GROUP BY BNAME;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

BNAME	SUM(AMOUNT)
VRCE	1000.00
ANJNI	500.00
KAROLBAGH	3500.00
CHANDNI	1200.00
MG ROAD	3000.00
ANDHERI	2000.00
VIRAR	1000.00
NEHRU PLACE	5000.00
POWAI	7000.00

11. Give city wise name and branch wise deposit

Query 1

```
1   SELECT C1.CITY , SUM(D1.AMOUNT) FROM CUSTOMER C1 , DEPOSIT D1 WHERE D1.CNAME = C1.CNAME GROUP BY C1.CITY;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CITY	SUM(D1.AMOUNT)
CALCUTTA	1000.00
DELHI	500.00
BARODA	3500.00
NAGPUR	4200.00
SURAT	2000.00
BOMBAY	6000.00

12. Give the branch wise loan of customer living in NAGPUR

Query 1

```
1   SELECT BNAME, SUM(AMOUNT) FROM BORROW JOIN CUSTOMER ON BORROW.CNAME=CUSTOMER.CNAME
2   WHERE CUSTOMER.CITY='NAGPUR' GROUP BY BNAME;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

BNAME	SUM(AMOUNT)
ANDHERI	2000
VIRAR	8000

13. Count total number of customers

Query 1

```
1   SELECT COUNT(*) FROM CUSTOMER;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

COUNT(*)
10

14. Count total number of depositors branch wise

Query 1

```
1   SELECT BNAME, COUNT(DISTINCT CNAME) FROM DEPOSIT GROUP BY BNAME;
```

Result Grid

BNAME	COUNT(DISTINCT CNAME)
ANDHERI	1
ANJNI	1
CHANDNI	1
KAROLBAGH	1
MG ROAD	1
NEHRU PLACE	1
POWAI	1
VIRAR	1
VRCE	1

15. Give maximum loan from branch VRCE

Query 1

```
1 •  SELECT BNAME, count(*) FROM DEPOSIT, CUSTOMER WHERE deposit.CNAME = CUSTOMER.CNAME GROUP BY BNAME;
```

Result Grid

BNAME	count(*)
VRCE	1
ANJNI	1
KAROLBAGH	1
CHANDNI	1
MG ROAD	1
ANDHERI	1
VIRAR	1
NEHRU PLACE	1

16. Give the number of customers who are depositors as well as borrowers

Query 1

```
1   select count(customer.CNAME) from customer where customer.CNAME IN (select deposit.cname from deposit)
2   and customer.CNAME IN (select borrow cname from borrow);
```

Result Grid

count(customer.CNAME)
6

Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

Experiment no: 6**Aim**

To familiarize with Join or Cartesian Product

CO1

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

1. Give name of customers having living city BOMBAY and branch city NAGPUR

```
SQL File 1* 
1   SELECT D1.CNAME,D1.BNAME,C1.CNAME,C1.CITY,B1.CITY,B1.BNAME
2   FROM DEPOSIT D1,CUSTOMER C1,BRANCH B1 WHERE C1.CITY = 'BOMBAY'
3   AND B1.CITY = 'NAGPUR' AND D1.CNAME = C1.CNAME AND D1.BNAME = B1.BNAME;
```

The screenshot shows the SQL query above in the query editor. Below it is the result grid with columns: CNAME, BNAME, CNAME, CITY, CITY, BNAME. The data returned is:

CNAME	BNAME	CNAME	CITY	CITY	BNAME

2. Give names of customers having the same living city as their branch city

```
SQL File 1* 
1 •  SELECT DISTINCT(CUSTOMER.CNAME), BRANCH.CITY
2   FROM BRANCH, CUSTOMER WHERE BRANCH.CITY = CUSTOMER.CITY;
```

The screenshot shows the SQL query above in the query editor. Below it is the result grid with columns: CNAME, CITY. The data returned is:

CNAME	CITY
KRANTI	BOMBAY
MADHURI	NAGPUR
NAREN	BOMBAY
PRAMOD	NAGPUR
SHIVANI	BOMBAY
SUNIL	DELHI

3. Give names of customers who are borrowers as well as depositors and having city NAGPUR.

```
SQL File 1* 
1 •  SELECT C1.CNAME FROM CUSTOMER C1,DEPOSIT D1,BORROW B1
2   WHERE C1.CITY='NAGPUR' AND C1.CNAME=D1.CNAME AND D1.CNAME = B1.CNAME;
```

The screenshot shows the SQL query above in the query editor. Below it is the result grid with columns: CNAME. The data returned is:

CNAME
MADHURI
PRAMOD

4. Give names of borrowers having deposit amount greater than 1000 and loan amount greater than 2000.

```
SQL File 1* 
1 •  SELECT BR1.CNAME, BR1.AMOUNT, D1.CNAME, D1.AMOUNT
2   FROM BORROW BR1,DEPOSIT D1 WHERE D1.CNAME = BR1.CNAME AND D1.AMOUNT > 1000 AND BR1.AMOUNT > 2000;
3
```

The screenshot shows the SQL query above in the query editor. Below it is the result grid with columns: CNAME, AMOUNT, CNAME, AMOUNT. The data returned is:

CNAME	AMOUNT	CNAME	AMOUNT
MEHUL	5000	MEHUL	3500.00
PRAMOD	8000	PRAMOD	3000.00
KRANTI	3000	KRANTI	5000.00

5. Give names of depositors having the same branch as the branch of Sunil

```
SQL File 1* 
1 •  SELECT D1.CNAME FROM DEPOSIT D1 WHERE D1.BNAME
2   IN (SELECT D2.BNAME FROM DEPOSIT D2 WHERE D2.CNAME = 'SUNIL');
```

The screenshot shows the SQL query above. Below it, the result grid displays a single row with the column headers 'CNAME' and 'AMOUNT'. The row contains the value 'SUNIL' under 'CNAME'.

CNAME	AMOUNT
SUNIL	

6. Give names of borrowers having loan amount greater than the loan amount of Pramod

```
SQL File 1* 
1 •  SELECT BR1.CNAME, BR1.AMOUNT FROM BORROW BR1
2   WHERE BR1.AMOUNT > ALL (SELECT BR2.AMOUNT FROM BORROW BR2 WHERE BR2.CNAME = 'PRAMOD');
```

The screenshot shows the SQL query above. Below it, the result grid displays a single row with the column headers 'CNAME' and 'AMOUNT'. The row contains the value 'PRAMOD' under 'CNAME' and '0' under 'AMOUNT'.

CNAME	AMOUNT
PRAMOD	0

7. Give the name of the customer living in the city where branch of depositor Sunil is located.

```
SQL File 1* 
1 •  SELECT C.CNAME FROM CUSTOMER C WHERE C.CITY IN (SELECT B.CITY FROM BRANCH B
2   WHERE B.BNAME IN (SELECT D.BNAME FROM DEPOSIT D WHERE D.CNAME='SUNIL'));
```

The screenshot shows the SQL query above. Below it, the result grid displays a single row with the column headers 'CNAME' and 'AMOUNT'. The row contains the value 'PRAMOD' under 'CNAME' and '0' under 'AMOUNT'.

CNAME	AMOUNT
PRAMOD	0

8. Give branch city and living city of Pramod

```
SQL File 1* 
1 •  SELECT B1.CITY , C1.CITY FROM BRANCH B1,CUSTOMER C1, DEPOSIT D1
2   WHERE C1.CNAME = 'PRAMOD' AND C1.CNAME = D1.CNAME AND D1.BNAME = B1.BNAME;
```

The screenshot shows the SQL query above. Below it, the result grid displays a single row with the column headers 'CNAME' and 'AMOUNT'. The row contains the value 'PRAMOD' under 'CNAME' and '0' under 'AMOUNT'.

CNAME	AMOUNT
PRAMOD	0

9. Give branch city of Sunil and branch city of Anil

```
SQL File 1* 
1 •   SELECT B1.CITY FROM DEPOSIT D1, BRANCH B1 WHERE D1.BNAME = B1.BNAME AND D1.CNAME IN ('SUNIL', 'ANIL');
```

The screenshot shows the results of the executed query:

CITY
NAGPUR

10. Give the living city of Anil and the living city of Sunil

```
SQL File 1* 
1     SELECT C1.CNAME, C1.CITY FROM CUSTOMER C1 WHERE C1.CNAME = 'ANIL' OR C1.CNAME = 'SUNIL';
```

The screenshot shows the results of the executed query:

CNAME	CITY
ANIL	CALCUTTA
SUNIL	DELHI
*	*
NULL	NULL

Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

Experiment no: 7

Aim

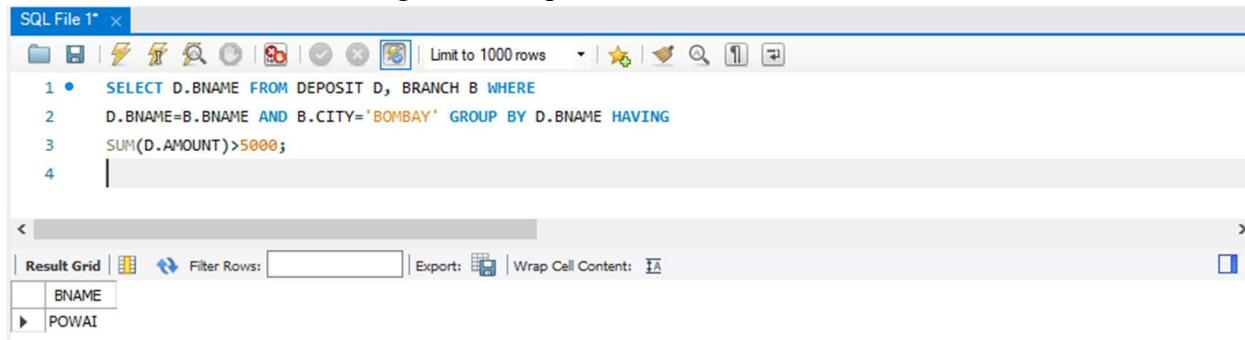
To familiarize with Group By and Having Clauses

CO1

Design and build a simple relational database system and demonstrate competence with the fundamental tasks involved with modeling, designing and implementing a database.

Procedure

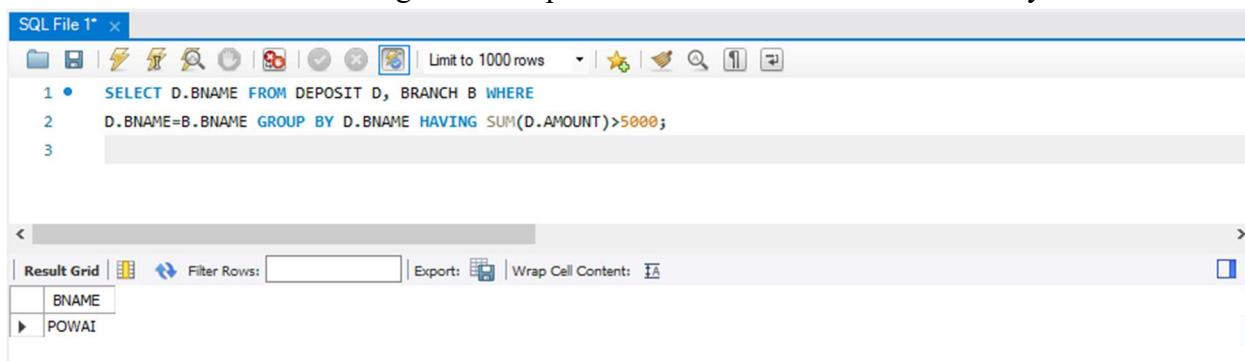
1. List the branches having sum of deposit more than 5000.



```
SQL File 1* 
1 •  SELECT D.BNAME FROM DEPOSIT D, BRANCH B WHERE
2     D.BNAME=B.BNAME AND B.CITY='BOMBAY' GROUP BY D.BNAME HAVING
3     SUM(D.AMOUNT)>5000;
4
```

The screenshot shows a SQL query in the query editor of SQL Server Management Studio. The query selects branch names from the DEPOSIT and BRANCH tables where the branch names match and the city is 'BOMBAY'. It groups by branch name and filters to show only those with a total deposit amount greater than 5000. The results are displayed in a grid with columns for BNAME, showing 'POWAI' as the result.

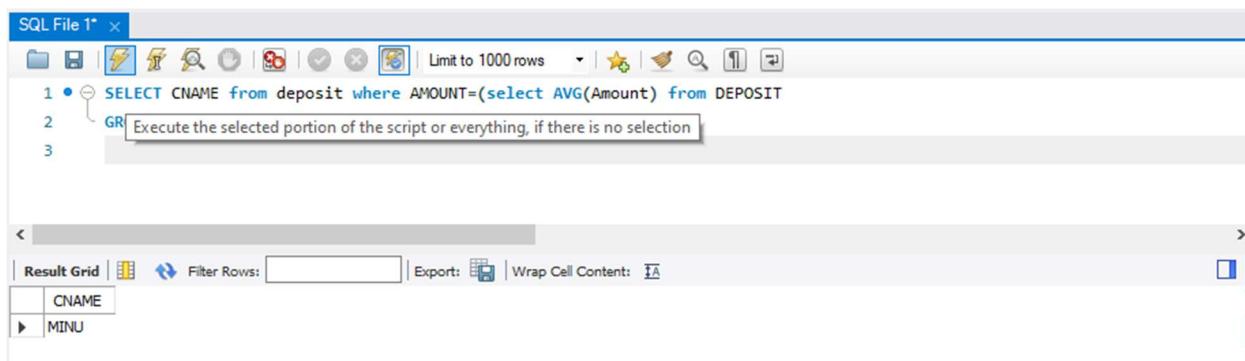
2. List the branches having sum of deposit more than 500 and located in city BOMBAY



```
SQL File 1* 
1 •  SELECT D.BNAME FROM DEPOSIT D, BRANCH B WHERE
2     D.BNAME=B.BNAME GROUP BY D.BNAME HAVING SUM(D.AMOUNT)>5000;
3
```

The screenshot shows a similar SQL query to the previous one, but it uses a GROUP BY clause instead of a HAVING clause. It selects branch names from the DEPOSIT and BRANCH tables where the branch names match, groups by branch name, and filters to show only those with a total deposit amount greater than 500. The results are displayed in a grid with columns for BNAME, showing 'POWAI' as the result.

3. List the names of customers having deposited in the branches where the average deposit is more than 5000.



```
SQL File 1* 
1 •  SELECT CNAME from deposit where AMOUNT=(select AVG(Amount) from DEPOSIT
2   GR Execute the selected portion of the script or everything, if there is no selection
3
```

The screenshot shows a complex SQL query. It selects customer names from the DEPOSIT table where the amount deposited is equal to the average amount from the same table. A comment 'GR Execute the selected portion of the script or everything, if there is no selection' is present in the query. The results are displayed in a grid with columns for CNAME, showing 'MINU' as the result.

4. List the names of customers having maximum deposit

```
SQL File 1* 
1   SELECT MAX(AMOUNT) | CNAME FROM deposit;
```

The screenshot shows the SQL Editor window with the query above. Below it is the Result Grid showing one row of data:

CNAME
7000.00

5. List the name of branch having highest number of depositors?

```
SQL File 1* 
1 •  SELECT D1.BNAME FROM DEPOSIT D1 GROUP BY D1.BNAME
2     HAVING COUNT(D1.CNAME) >= ALL (SELECT COUNT(D2.CNAME) FROM DEPOSIT D2 GROUP BY D2.BNAME);
3
```

The screenshot shows the SQL Editor window with the query above. Below it is the Result Grid showing a list of branches:

BNAME
VRCE
ANJANI
KAROLBAGH
CHANDNI
MG ROAD
ANDHERI
VIRAR
NEHRU PLACE
POWAI

6. Count the number of depositors living in NAGPUR.

```
SQL File 1* 
1 •  SELECT COUNT(DEPOSIT.CNAME)FROM DEPOSIT,CUSTOMER WHERE CUSTOMER.CITY='NAGPUR';
```

The screenshot shows the SQL Editor window with the query above. Below it is the Result Grid showing one row of data:

COUNT(DEPOSIT.CNAME)
18

7. Give names of customers in VRCE branch having more deposite than any other customer in same branch

SQL File 1*

```

1 SELECT CNAME FROM DEPOSIT WHERE BNAME='VRCE' AND AMOUNT=(SELECT
2 MAX(AMOUNT) FROM DEPOSIT WHERE BNAME='VRCE');
3

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: □

CNAME
ANIL

8. Give the names of branch where number of depositors is more than 5

SQL File 1*

```

1 • SELECT BNAME from deposit GROUP BY BNAME HAVING
2 COUNT(BNAME)>5;
3

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: □

BNAME

9. Give the names of cities in which the maximum number of branches are located

SQL File 1*

```

1 • SELECT C.CNAME ,COUNT(B.BNAME) FROM CUSTOMER C JOIN BRANCH B ON
2 C.CNAME=B.DNAME GROUP BY C.CNAME ORDER BY COUNT(B.BNAME) DESC;
3

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: □

CNAME	COUNT(B.BNAME)
-------	----------------

10. Count the number of customers living in the city where branch is located

SQL File 1*

```

1 • SELECT COUNT(R1.BNAME) FROM DEPOSIT D1 JOIN BORROW_B1 C1 WHERE
2 C1.CITY=D1.CITY AND D1.CITY=C1.CITY IN (SELECT CITY FROM CUSTOMER);
3

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: □

COUNT(B1.BNAME)
6

Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

Experiment No: 8

Aim

To familiarize with trigger functions

CO2

Apply PL/SQL for processing databases

Procedure

Step 1: Start

Step 2: Initialize the trigger.

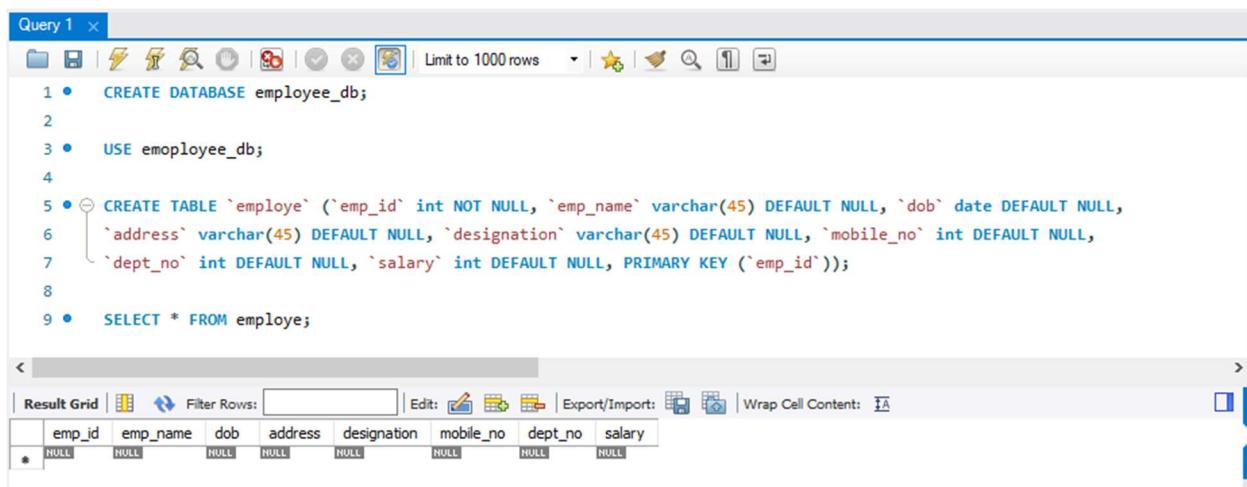
Step 3: On update the trigger has to be executed.

Step 4: Execute the trigger procedure after updation

Step 5: Carry out the operation on the table to check for trigger execution.

Step 6: Stop

Create a Trigger for employee table it will update another table salary while updating values



```

Query 1 < X
CREATE DATABASE employee_db;
USE employee_db;
CREATE TABLE `employee` (`emp_id` int NOT NULL, `emp_name` varchar(45) DEFAULT NULL, `dob` date DEFAULT NULL,
`address` varchar(45) DEFAULT NULL, `designation` varchar(45) DEFAULT NULL, `mobile_no` int DEFAULT NULL,
`dept_no` int DEFAULT NULL, `salary` int DEFAULT NULL, PRIMARY KEY (`emp_id`));
SELECT * FROM employee;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: 

```

emp_id	emp_name	dob	address	designation	mobile_no	dept_no	salary
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Query 1

```

1 • CREATE TABLE `salary` ( `employee_id` int NOT NULL, `old_sal` int DEFAULT NULL, `new_sal` int DEFAULT NULL,
2   `rev_date` date DEFAULT NULL, PRIMARY KEY (`employee_id`) );
3
4 •  SELECT * FROM salary;

```

Result Grid

employee_id	old_sal	new_sal	rev_date
*	NULL	NULL	NULL

employe

```

1   INSERT INTO `employee_db`.`employe`
2     (`emp_id`, `emp_name`, `dob`, `address`, `designation`, `mobile_no`, `dept_no`, `salary`)
3     VALUES ('1', 'amal', '1995-04-29', 'wayanad', 'developer', '9469664422', '7', '40000');
4
5 •  SELECT * FROM employe;
6

```

Result Grid

emp_id	emp_name	dob	address	designation	mobile_no	dept_no	salary
1	amal	199... NULL	wayanad NULL	developer NULL	9469664... NULL	7 NULL	40000 NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table Definition

Table Name: **employe** Schema: **employee_db**

Charset/Collation: **utf8mb4** Engine: **InnoDB**

Comments:

BEFORE INSERT
AFTER INSERT
BEFORE UPDATE
▼ AFTER UPDATE
 employe_AFTER_UPDATE
BEFORE DELETE
AFTER DELETE

```

1 •  CREATE DEFINER=`root`@`localhost`
2   TRIGGER `employe_AFTER_UPDATE` AFTER UPDATE ON `employe` FOR EACH ROW BEGIN
3     if(new.salary != old.salary)
4       then
5         INSERT INTO salary (employee_id,old_sal,new_sal,rev_date) values
6         (new.emp_id,old.salary,new.salary,sysdate());
7       END if;
8     END

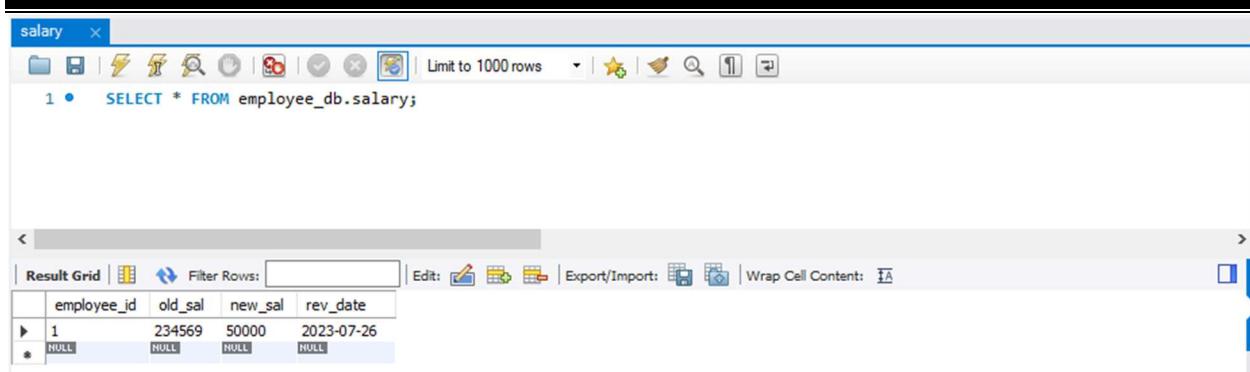
```

salary

```

1   UPDATE employe SET salary=50000 WHERE emp_id=1;

```



The screenshot shows a MySQL Workbench interface titled "salary". In the query editor, the following SQL command is entered:

```
1 •  SELECT * FROM employee_db.salary;
```

The result grid displays the following data:

	employee_id	old_sal	new_sal	rev_date
▶	1	234569	50000	2023-07-26
*	HULL	NULL	NULL	NULL

Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained

Experiment No: 9

Aim

To familiarize with Stored functions and Procedure

CO2

Apply PL/SQL for processing databases

Procedure

```
DELIMITER //
CREATE PROCEDURE GET_BRANCH()
BEGIN
    SELECT * FROM BORROW;
END//
```



```
DELIMITER ;
CALL GET_BRANCH ()
```

```

1  DELIMITER //
2
3 • CREATE PROCEDURE GET_BRANCH()
4 BEGIN
5     SELECT * FROM BRANCH;
6 END //
7
8  DELIMITER ;
9 • CALL GET_DEPOSIT();

```

ACTNO	CNAME	BNAME	AMOUNT	ADATE
D100	ANIL	VRCE	1000	1995-03-01
D101	SUNIL	AJNI	500	1996-01-04
D102	MEHUL	KAROLBAGH	3500	1995-11-07
D103	MANDAR	CHANDNI	4500	1995-12-17
D104	MADHURI	DHARAMPETH	1200	1995-12-17

Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained

Experiment No: 10

Aim

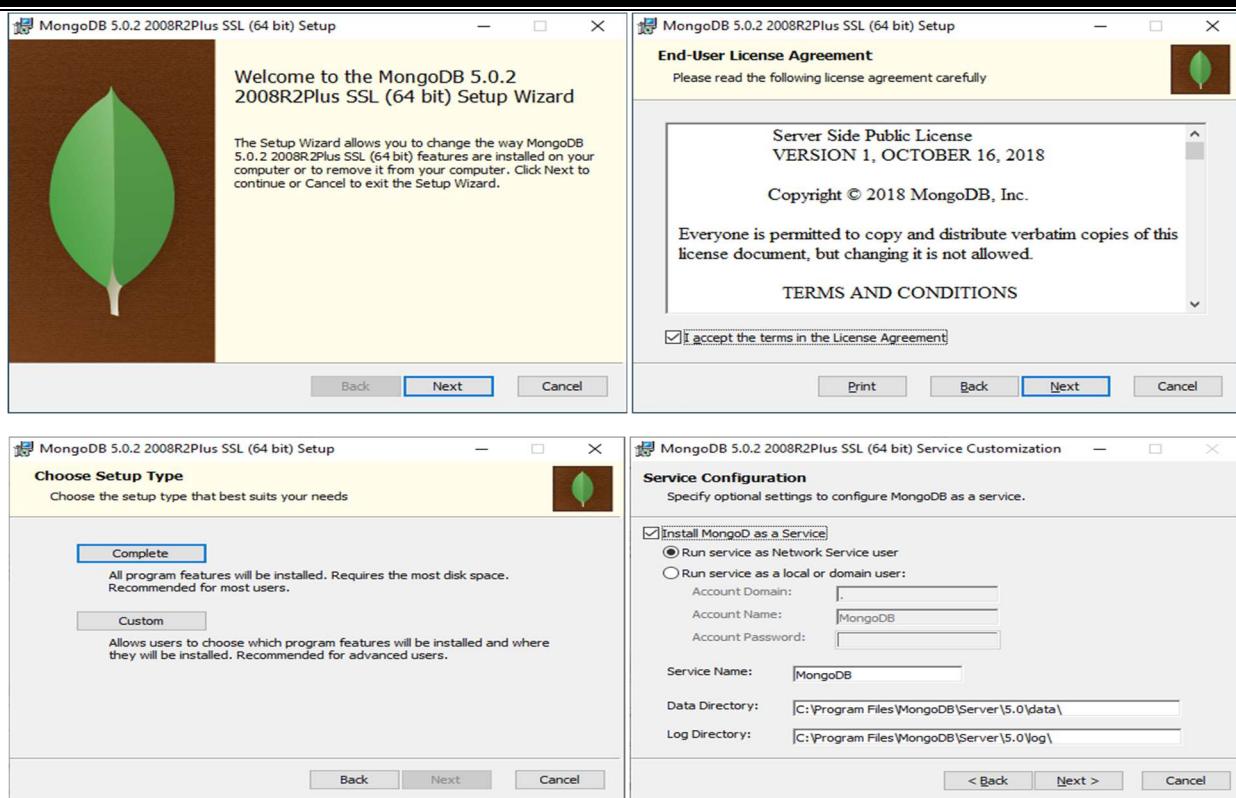
Understand the installation and configuration of NoSQL Databases: MongoDB

CO3

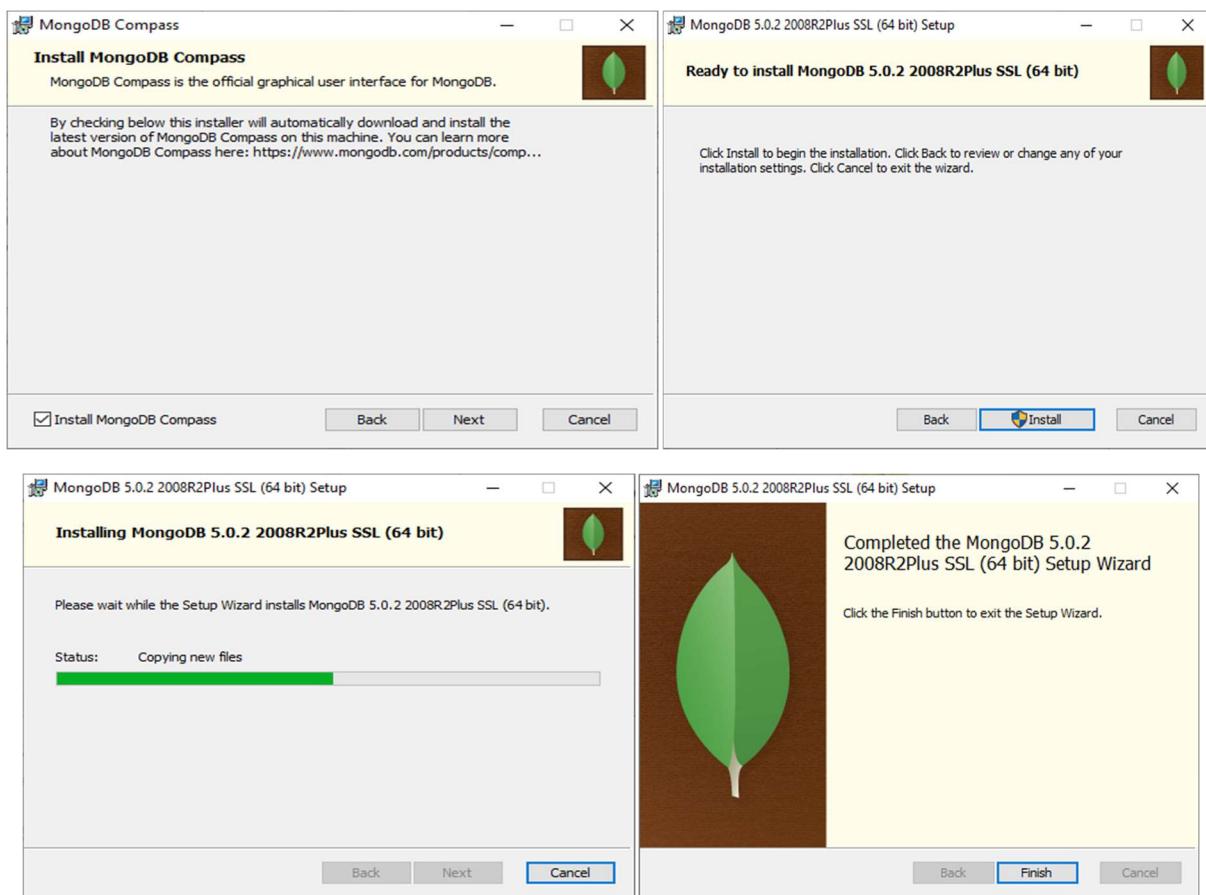
Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

Procedure

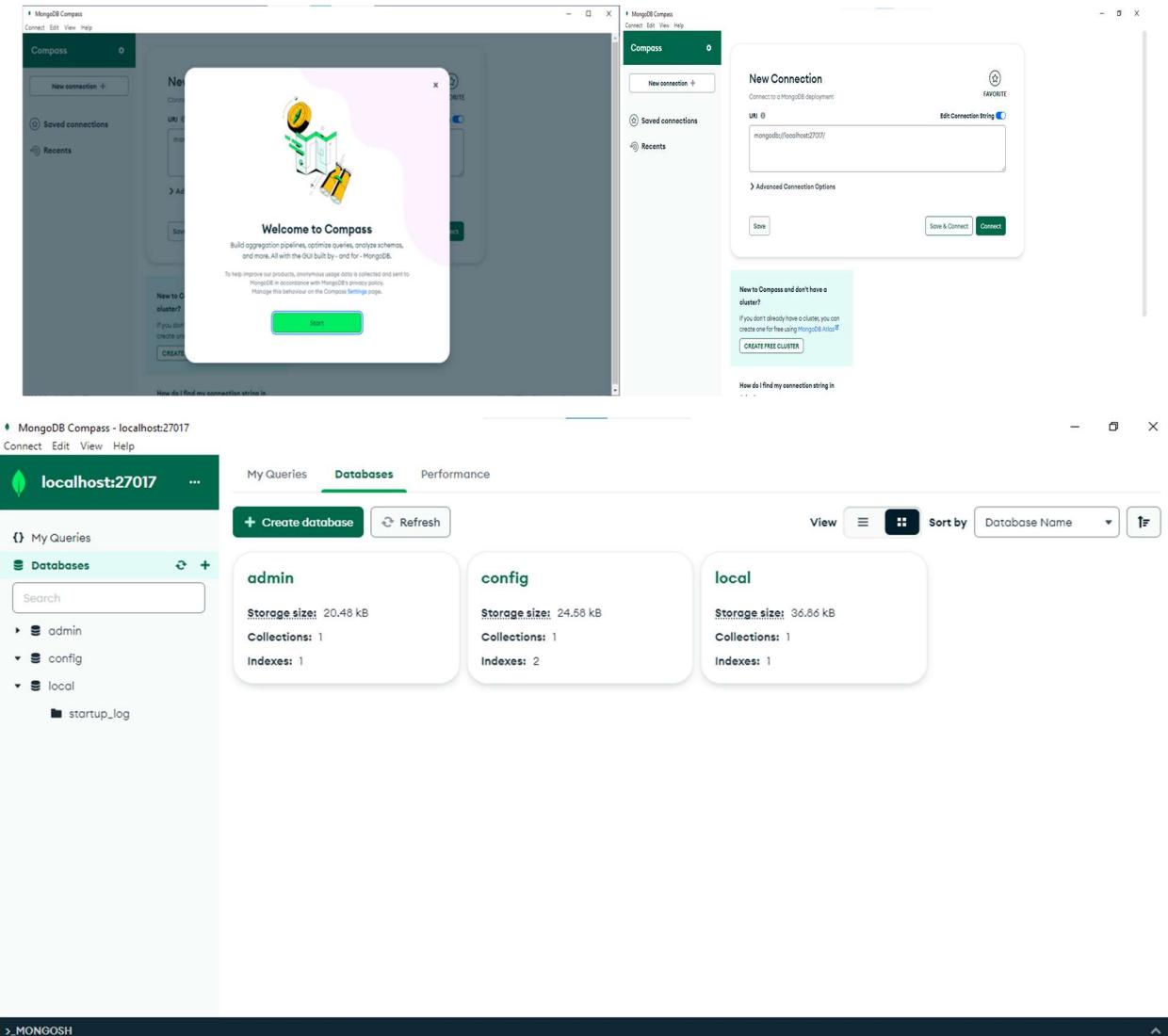
1. Setup Wizard



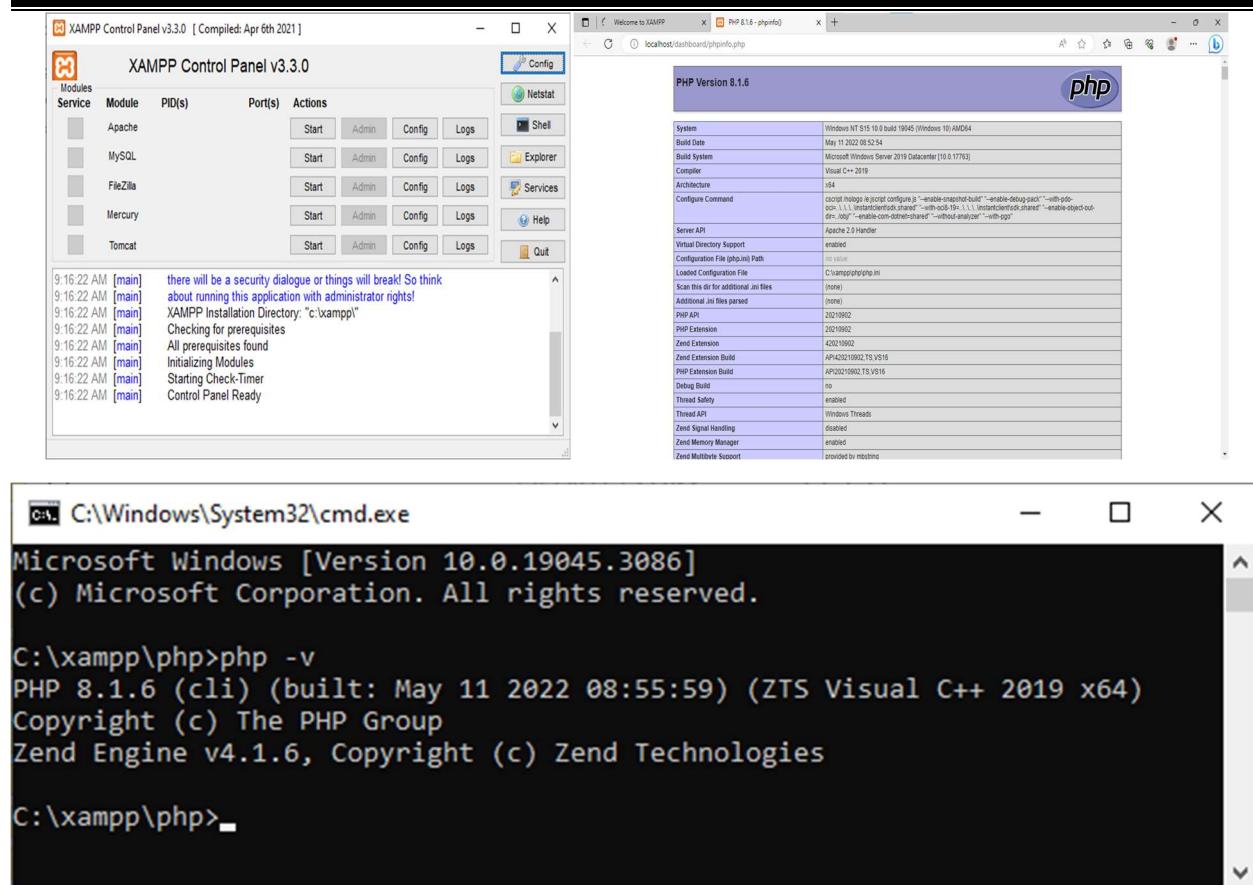
2. Installing MongoDB Compass



3. MongoDB Compass Home Page



4. Configuring MongoDB to Connect to PHP



5. Installing MongoDB Package for PHP

The screenshot shows the PECL package manager interface for the MongoDB driver:

- Documentation:** Includes links to Home, News, and Support.
- Downloads:** Includes links to Browse Packages, Search Packages, and Download Statistics.
- Package Information:**
 - Maintainers:** Jeremy Mikola (lead), Katherine Walker (developer), Andreas Braun (lead), Derick Rethans (lead at phpb dot net), Hannes Magnusson (biori at phpb dot net) (lead).
 - License:** Apache License.
 - Description:** The purpose of this driver is to provide exceptionally thin glue between MongoDB and PHP, implementing only fundamental and performance-critical components necessary to build a fully-functional MongoDB driver.
 - Homepage:** <http://docs.mongodb.org/ecosystem/drivers/php/>
- Available Releases:** A table showing releases for MongoDB PHP driver:

Version	State	Release Date	Downloads
1.16.1	stable	2023-06-22	mongodb-1.16.1.tgz (1862.3KB)
1.16.0	stable	2023-06-22	mongodb-1.16.0.tgz (1521.9KB)
1.15.3	stable	2023-05-12	mongodb-1.15.3.tgz (1701.8KB)
1.15.2	stable	2023-04-21	mongodb-1.15.2.tgz (1702.1KB)
1.15.1	stable	2023-07-09	mongodb-1.15.1.tgz (1701.4KB)

6. Configuring Apache

File Explorer

Apache (httpd.conf)

Apache (httpd-ssl.conf)

Apache (httpd-xampp.conf)

PHP (php.ini)

```

; PHP - NotePad
File Edit Format View Help
;extension=imap
;extension=ldap
extension=mbstring
extension=exif ; Must be after mbstring as it depends on it
extension=mysql
;extension=oci8_12c ; Use with Oracle Database 12c Instant Client
;extension=oci8_19 ; Use with Oracle Database 19 Instant Client
extension=odbc
extension=pdo
extension=pdo_sqlite
extension=pdo_pgsql
extension=pdo_mysql
extension=pdo_firebird
extension=pdo_mysql
;extension=pdo_oci
;extension=pdo_odb
;extension=pdo_pgsql
;extension=pdo_sqlite
;extension=pdo_pgsql
;extension=sybase
;extension=openssl
;extension=pdo_mssql
;extension=soap
;extension=sockets
;extension=sodium
;extension=sqlite3
;extension=tidy
;extension=xsl
;zend_extension=opcache
;*****;
; Module Settings ;
;*****;
;so taes=Off

```

phpMyAdmin (config.inc.php)

<Browse> [Apache]

<Browse> [PHP]

<Browse> [phpMyAdmin]

PHP 8.1.6 - phpinfo()

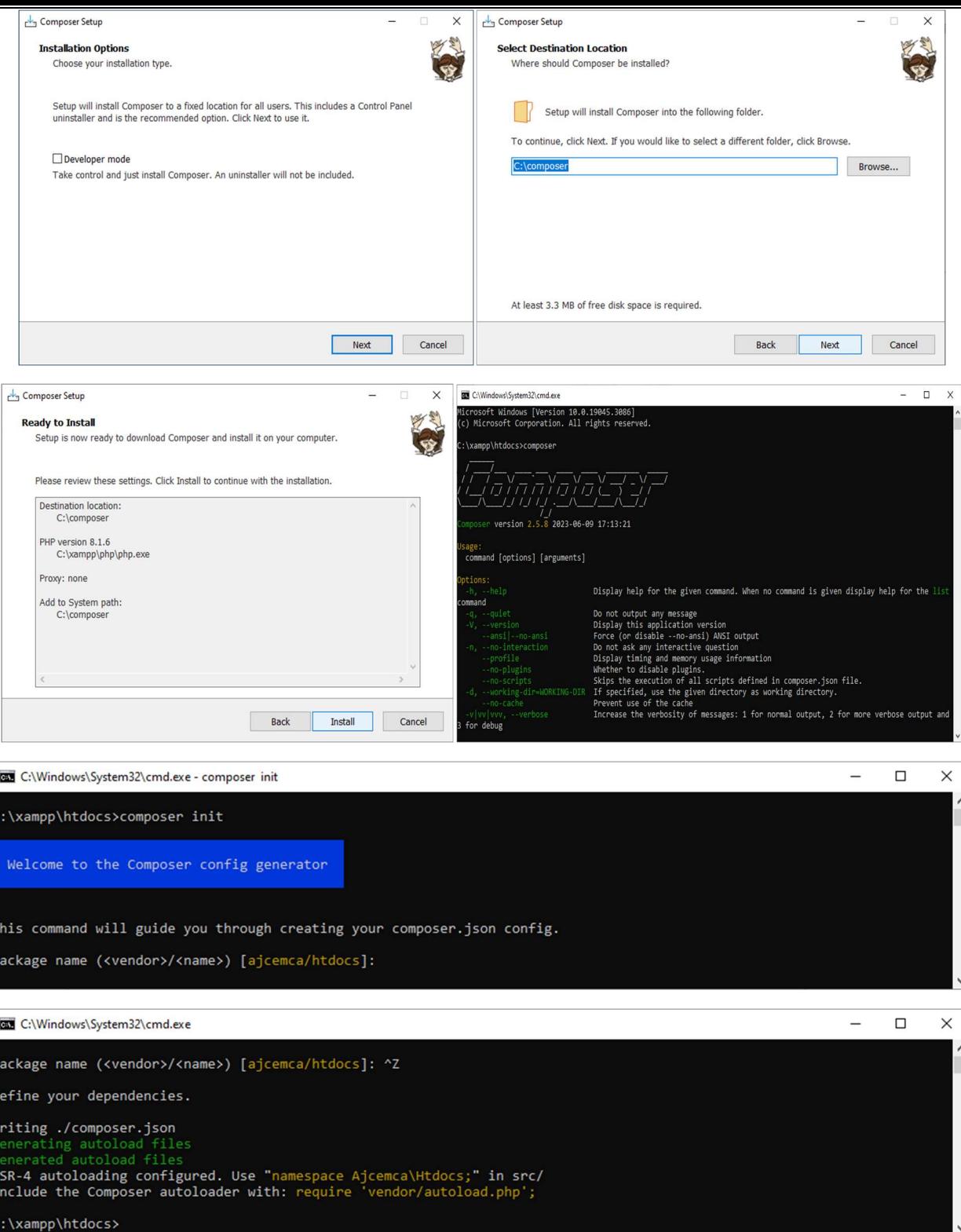
Directive	Local Value	Master Value
max_execution_time	700000	700000
mbstring.regex_stack_limit	100000	100000
mbstring.strict_detection	Off	Off
mbstring.substitute_character	no value	no value

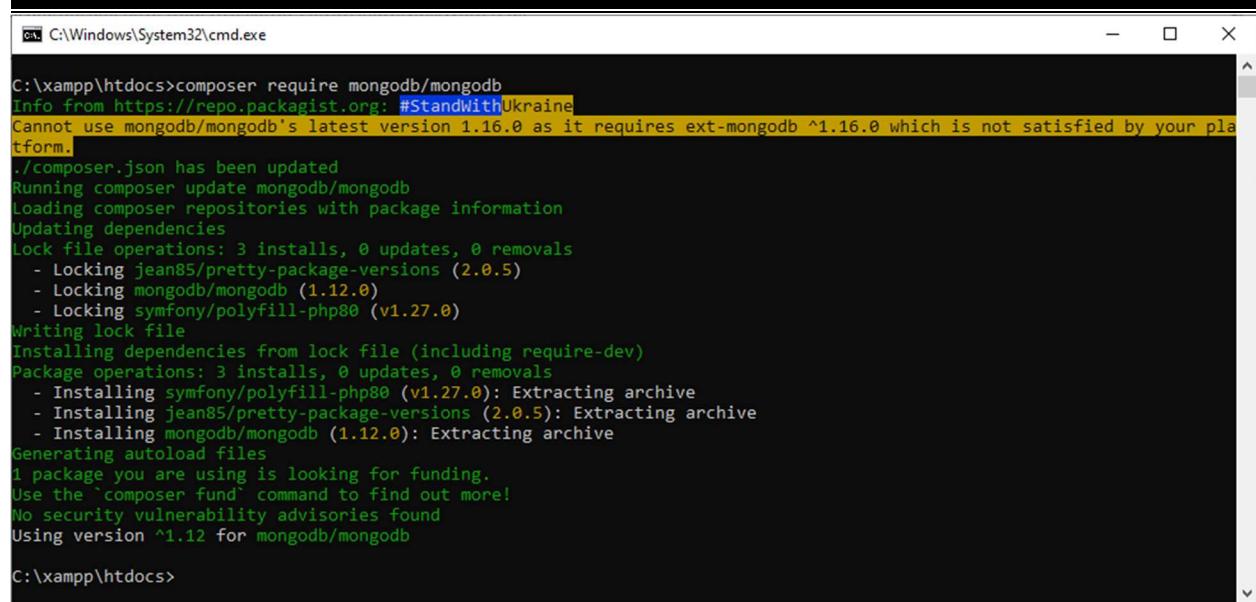
mongodb

MongoDB support	enabled
MongoDB extension version	1.13.0
MongoDB extension stability	stable
libbson bundled version	1.21.1
libmongoc bundled version	1.21.1
libmongoc SSL	enabled
libmongoc SSL library	OpenSSL
libmongoc crypto	enabled
libmongoc crypto library	libcrypto
libmongoc crypto system profile	disabled
libmongoc SASL	enabled
libmongoc ICU	disabled
libmongoc compression	disabled
libmongocrypt bundled version	1.3.2
libmongocrypt crypt	enabled
libmongocrypt crypto library	libcrypto

Directive	Local Value	Master Value
mongodb.debug	no value	no value
mongodb.mock_service_id	Off	Off

mysqli





```
C:\xampp\htdocs>composer require mongodb/mongodb
Info from https://repo.packagist.org: #StandWithUkraine
Cannot use mongodb/mongodb's latest version 1.16.0 as it requires ext-mongodb ^1.16.0 which is not satisfied by your platform.
./composer.json has been updated
Running composer update mongodb/mongodb
Loading composer repositories with package information
Updating dependencies
Lock file operations: 3 installs, 0 updates, 0 removals
- Locking jean85/pretty-package-versions (2.0.5)
- Locking mongodb/mongodb (1.12.0)
- Locking symfony/polyfill-php80 (v1.27.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
- Installing symfony/polyfill-php80 (v1.27.0): Extracting archive
- Installing jean85/pretty-package-versions (2.0.5): Extracting archive
- Installing mongodb/mongodb (1.12.0): Extracting archive
Generating autoload files
1 package you are using is looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found
Using version ^1.12 for mongodb/mongodb

C:\xampp\htdocs>
```

Result

The program was executed and the result was successfully obtained. Thus CO3 was obtained

Experiment No: 11**Aim**

Create a database and use the insertMany method to insert the colors of the rainbow into the database

CO4

Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure

```
<?php  
require '../vendor/autoload.php';  
$conn = new MongoDB\Client("mongodb://localhost:27017");  
echo "Connection Successful.";  
echo "<br>";  
$db = $conn -> Rainbow;  
echo " Rainbow database created.";  
echo "<br>";  
$collection = $db -> createCollection("Colors");  
echo "Collection Colors created";  
echo "<br>";  
$collection = $db -> Colors;  
echo "<br>";  
$c1 = array('color' => 'Violet');  
$c2 = array('color' => 'Indigo');  
$c3 = array('color' => 'Blue');  
$c4 = array('color' => 'Green');  
$c5 = array('color' => 'Yellow');  
$c6 = array('color' => 'Orange');  
$c7 = array('color' => 'Red');
```

```
$collection -> insertMany([\$c1, \$c2, \$c3, \$c4, \$c5, \$c6, \$c7]);
echo "Colors Inserted ";
echo "<br>";
?>
```

Output

```
Connection Successful.
Rainbow database created.
Collection Colors created
Colors Inserted
```

#	Color
1	Violet
2	Indigo
3	Blue
4	Green
5	Yellow
6	Orange
7	Red

Result

The program was executed and the result was successfully obtained. Thus CO4 was obtained

Experiment No: 12**Aim**

Implementing CRUD operations using PHP-MongoDB

CO4

Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure**connection.php**

```
<?php  
require '../vendor/autoload.php';  
  
$con=new MongoDB\Client("mongodb://localhost:27017");  
//echo $con;  
  
$db=$con->admin;  
$collection=$db->Profile;  
?>
```

register.php

```
<html>  
<head><title>Registration form</title>  
</head>
```

```
<body><table border="1" align="center">
<tr><td>
<h1>Registration Form</h1>
<form action="insert.php" method="post">
Name:<input type="text" name="name" required><br><br>
Email:<input type="text" name="email" required><br><br>
Place :<input type="text" name="place" required><br><br>
<input type="submit" name="submit" class="button" value="Register" >
</form>
</td></tr>
</table>
</body>
</html>
```

edit.php

```
<?php
include_once("connection.php");
if(isset($_POST['update'])){
echo $id = $_GET['id'];
$name = $_POST['name'];
$place = $_POST['place'];
$email = $_POST['email'];
$db->Profile->updateOne(['_id' => new MongoDB\BSON\ObjectId($id)], ['$set' => ['name'=>
$name, 'place' => $place, 'email'=>$email,]]);
header("Location: select.php");
}?>
```

```
<html><head>
<title>Registration Form</title>
</head>
```

```
<?php
$id = $_GET['id'];
```

```

$result = $db->Profile->findOne(['_id' => new MongoDB\BSON\ObjectID($id)]);

$name = $result['name'];
$place = $result['place'];
$email = $result['email'];

?>

<body>

<h2>Profile Data</h2>

<form action="#" method="post">

<?php $id = $_GET['id'];?>

Name:<input type="text" name="name" value="<?php echo $name; ?>" required><br>
Place:<input type="place" name="place" value="<?php echo $place; ?>" required><br>
Email:<input type="email" name="email" value="<?php echo $email; ?>" required><br>
<input type="submit" name="update" class="button" value="Update">

</form></body>

</html>

```

select.php

```

<center><br><form action=register.php>
<input type="submit" value=register></form>
<form action=select.php>
<input type="submit" value=View>
</form><br>
<?php
include_once("connection.php");
$result=$collection->find();
?>
<html>
<head><title>select.php</title>
</head><body>
<table border="0" >

```

```

<tr>
<th align="center" bgcolor=green><font color=white size=4><b>SlNo.</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Name</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Email</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Place</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Delete</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Edit</b></th>
</tr><?php
$no=1;
foreach($result as $res){
?> <tr>
<td><font color=DeepSkyBlue size=4><?php echo $no;?>
<td><font color=steelblue size=4><?php echo $res['name'];?></td>
<td><font color=pink size=4><?php echo $res['email'];?></td>
<td><font color=RosyBrown size=4><?php echo $res['place'];?></td>
<td><a href=<?php echo "delete.php?id=$res[_id]";?>" onClick="return confirm('Are you sure
you want to delete?')">Delete</a></td>
<td><a href=<?php echo "edit.php?id=$res[_id]";?>">Edit</a></td>
<?php
$no++;
} ?>
</tr></table></body></html>

```

insert.php

```

<?php
include_once("connection.php");
if(isset($_POST["submit"])){
    $user=array(
        'name'=>$_POST['name'],
        'email'=>$_POST['email'],
        'place'=>$_POST['place']

```

```

    );
$collection->insertOne($user);
echo "Inserted";
}

header ("location:select.php");
?>

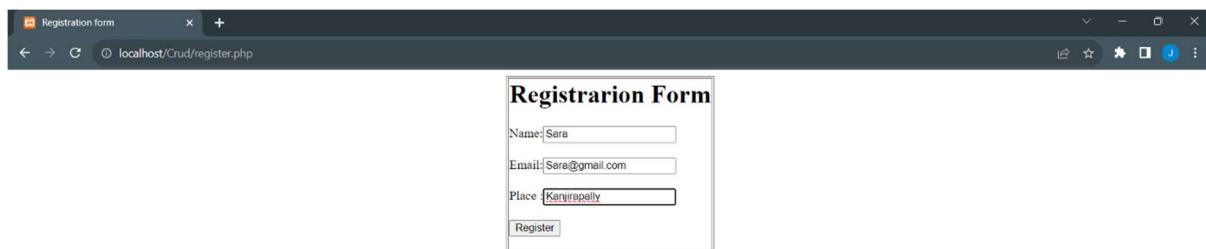
```

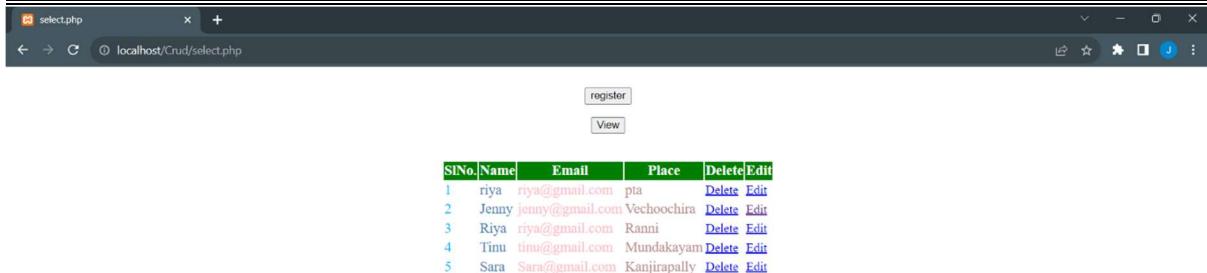
delete.php

```

<?php
include _once("connection.php");
$id = $_GET['id'];
$collection->deleteOne(['_id' => new MongoDB\BSON\ObjectId($id)]);
header("location:select.php");
?>

```

Output Screenshot**Connection.php****Register.php****Select.php**



A screenshot of a web browser window titled "select.php". The page displays a table with columns: SlNo, Name, Email, Place, Delete, and Edit. The data rows are:

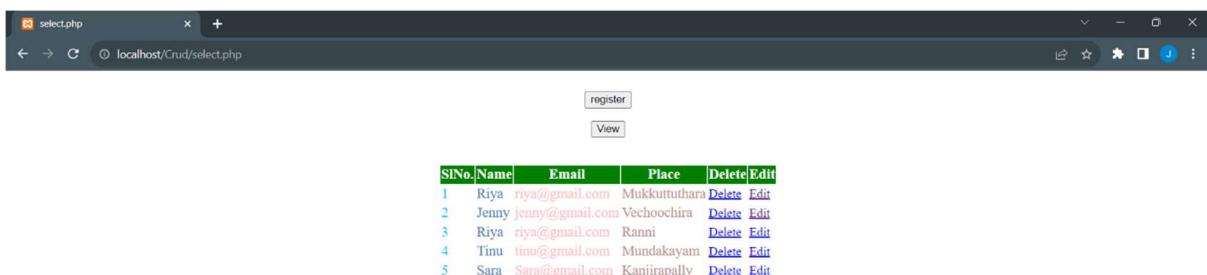
SlNo	Name	Email	Place	Delete	Edit
1	Riya	riya@gmail.com	Pita	Delete	Edit
2	Jenny	jenny@gmail.com	Vechoochira	Delete	Edit
3	Riya	riya@gmail.com	Ranni	Delete	Edit
4	Tinu	tinu@gmail.com	Mundakayam	Delete	Edit
5	Sara	Sara@gmail.com	Kanjirapally	Delete	Edit

Below the table are two buttons: "register" and "View".

Edit.php



A screenshot of a web browser window titled "Registration Form". The URL is "localhost/Crud/edit.php?id=64a2ad9524e4d59084075374". The page contains a section titled "Profile Data" with three input fields: "Name: riya", "Place: pta", and "Email: riya@gmail.com". Below the fields is a "Update" button.

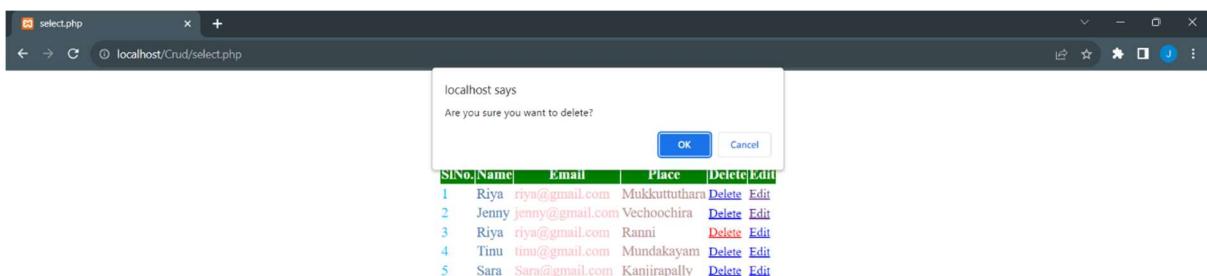


A screenshot of a web browser window titled "select.php". The page displays a table with columns: SlNo, Name, Email, Place, Delete, and Edit. The data rows are identical to the first table:

SlNo	Name	Email	Place	Delete	Edit
1	Riya	riya@gmail.com	Mukkututhari	Delete	Edit
2	Jenny	jenny@gmail.com	Vechoochira	Delete	Edit
3	Riya	riya@gmail.com	Ranni	Delete	Edit
4	Tinu	tinu@gmail.com	Mundakayam	Delete	Edit
5	Sara	Sara@gmail.com	Kanjirapally	Delete	Edit

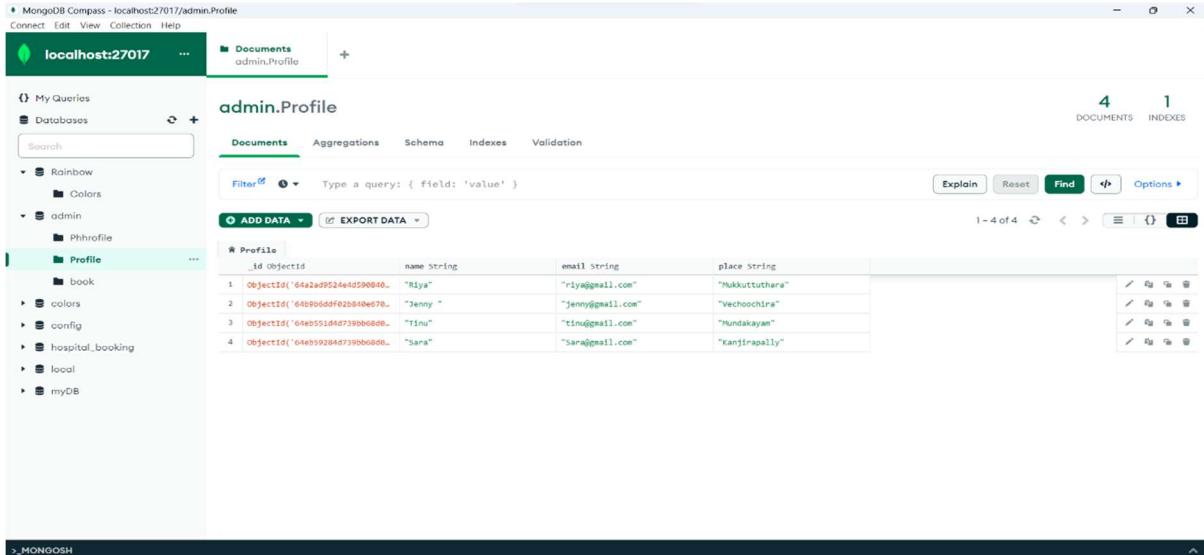
Below the table are two buttons: "register" and "View".

Delete.php



A screenshot of a web browser window titled "select.php". The URL is "localhost/Crud/select.php". A confirmation dialog box is displayed, asking "Are you sure you want to delete?". The "OK" button is highlighted. Below the dialog is a table with the same data as the previous tables:

SlNo	Name	Email	Place	Delete	Edit
1	Riya	riya@gmail.com	Mukkututhari	Delete	Edit
2	Jenny	jenny@gmail.com	Vechoochira	Delete	Edit
3	Riya	riya@gmail.com	Ranni	Delete	Edit
4	Tinu	tinu@gmail.com	Mundakayam	Delete	Edit
5	Sara	Sara@gmail.com	Kanjirapally	Delete	Edit



The screenshot shows the MongoDB Compass interface connected to localhost:27017/admin.Profile. The left sidebar lists databases and collections, with 'Profile' selected. The main area displays the 'admin.Profile' collection, which contains 4 documents and 1 index. The table shows the following data:

SINo.	Name	Email	Place	Delete	Edit
1	Riya	riya@gmail.com	Mukkututhara	Delete	Edit
2	Jenny	jenny@gmail.com	Vechoochira	Delete	Edit
3	Tinu	tinu@gmail.com	Mundakayam	Delete	Edit
4	Sara	Sara@gmail.com	Kanjirappally	Delete	Edit

Result

The program was executed and the result was successfully obtained. Thus CO4 was obtained

Experiment No: 13

Aim

Build sample collections/documents to perform the shell queries

CO5

Understand the basic storage architecture of distributed file systems.

Procedure

1. Create/Use a database

```
> use expmongo
```

```
◆ C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> use expmongo
switched to db expmongo
>
```

2. Display current database

```
>db
```

```
◆ C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db
expmongo
> -
```

3. Create a collection

```
>db.createCollection("actors")
```

```
◆ C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.createCollection("actors")
{ "ok" : 1 }
>
```

4. Insert data into the collection

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.insertMany([
... { _id: "trojan", name: "Ivan Trojan", year: 1964, movies: [ "samotari", "medvidek" ] },
... { _id: "machacek", name: "Jiri Machacek", year: 1966, movies: [ "medvidek", "vratnelahve", "samotari" ] },
... { _id: "schneiderova", name: "Jitka Schneiderova", year: 1973, movies: [ "samotari" ] },
... { _id: "sverak", name: "Zdenek Sverak", year: 1936, movies: [ "vratnelahve" ] },
... { _id: "geislerova", name: "Anna Geislerova", year: 1976 }
... ])
{
    "acknowledged" : true,
    "insertedIds" : [
        "trojan",
        "machacek",
        "schneiderova",
        "sverak",
        "geislerova"
    ]
}
>
```

5. Display documents in collection

```
>db.actors.find()
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find()
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
> -
```

6. Display documents in collection

```
>db.actors.find({ })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
> -
```

7. Display

```
>db.actors.find({ _id: "trojan" })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ _id: "trojan" })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
> -
```

8. Display

```
>db.actors.find({ name: "Ivan Trojan", year: 1964 })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ name: "Ivan Trojan", year: 1964 })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
> -
```

9. Display

```
>db.actors.find({ year: { $gte: 1960, $lte: 1980 } })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ year: { $gte: 1960, $lte: 1980 } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schniederova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

10. Display

```
>db.actors.find({ movies: { $exists: true } })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: { $exists: true } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schniederova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
>
```

11. Display

```
>db.actors.find({ movies: "medvidek" })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: "medvidek" })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
>
```

12. Display

```
>db.actors.find({ movies: { $in: [ "medvidek", "pelisky" ] } })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: { $in: [ "medvidek", "pelisky" ] } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
>
```

13. Display

```
>db.actors.find({ movies: { $all: [ "medvidek", "pelisky" ] } })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: { $all: [ "medvidek", "pelisky" ] } })
>
```

14. Display

```
>db.actors.find({ $or: [ { year: 1964 }, { rating: { $gte: 3 } } ] })
```

```
Select C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ $or: [ { year: 1964 }, { rating: { $gte: 3 } } ] })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
>
```

15. Display

```
>db.actors.find({ rating: { $not: { $gte: 3 } } })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ rating: { $not: { $gte: 3 } } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
> -
```

16. Display

```
>db.actors.find( {}, { name: 1, year: 1 })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find( {}, { name: 1, year: 1 })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964 }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966 }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973 }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936 }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

17. Display

```
>db.actors.find( {}, { movies: 0, _id: 0 })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find( {}, { movies: 0, _id: 0 })
{ "name" : "Ivan Trojan", "year" : 1964 }
{ "name" : "Jiri Machacek", "year" : 1966 }
{ "name" : "Jitka Schneiderova", "year" : 1973 }
{ "name" : "Zdenek Sverak", "year" : 1936 }
{ "name" : "Anna Geislerova", "year" : 1976 }
>
```

18. Display

```
>db.actors.find( {}, { name: 1, movies: { $slice: 2 }, _id: 0 })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find( {}, { name: 1, movies: { $slice: 2 }, _id: 0 })
{ "name" : "Ivan Trojan", "movies" : [ "samotari", "medvidek" ] }
{ "name" : "Jiri Machacek", "movies" : [ "medvidek", "vratnelahve" ] }
{ "name" : "Jitka Schneiderova", "movies" : [ "samotari" ] }
{ "name" : "Zdenek Sverak", "movies" : [ "vratnelahve" ] }
{ "name" : "Anna Geislerova" }
>
```

19. Display

```
>db.actors.find().sort({ year: 1, name: -1 })
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find().sort({ year: 1, name: -1 })
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

20. Display

```
>db.actors.find().sort({ name: 1 }).skip(1).limit(2)
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find().sort({ name: 1 }).skip(1).limit(2)
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
>
```

21. Display

```
>db.actors.find().sort({ name: 1 }).limit(2).skip(1)
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find().sort({ name: 1 }).limit(2).skip(1)
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
> -
```

Result

The program was executed and the result was successfully obtained. Thus CO5 was obtained

Experiment No: 14

Aim

To familiarize with indexing in MongoDB

CO5

Understand the basic storage architecture of distributed file systems.

Procedure

1.Input data –

```
db.products.insertMany([
  { _id: 10, item: "large box", qty: 50 },
  { _id: 11, item: "medium box", qty: 30 },
  { _id: 12, item: "envelope", qty: 100},
  { _id: 13, item: "tape", qty: 20},
  { _id: 14, item: "bubble wrap", qty: 70}
])
```

```
> db.products.insertMany( [
... { _id: 10, item: "large box", qty: 50 },
... { _id: 11, item: "medium box", qty: 30 },
... { _id: 12, item: "envelope", qty: 100},
... { _id: 13, item: "tape", qty: 20},
... { _id: 14, item: "bubble wrap", qty: 70}
... ])
{ "acknowledged" : true, "insertedIds" : [ 10, 11, 12, 13, 14 ] }
```



```
> db.products.find()
{ "_id" : 10, "item" : "large box", "qty" : 50 }
{ "_id" : 11, "item" : "medium box", "qty" : 30 }
{ "_id" : 12, "item" : "envelope", "qty" : 100 }
{ "_id" : 13, "item" : "tape", "qty" : 20 }
{ "_id" : 14, "item" : "bubble wrap", "qty" : 70 }
```

Example 1 – Create ascending index on a field

```
db.collection.createIndex( { item: 1 } )
```

```
> db.collection.createIndex( { item: 1 } )
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : true,
  "ok" : 1
}
```

Example 2 – Create descending index on a field

```
db.collection.createIndex( { qty: -1 } )
```

```
> db.collection.createIndex( { qty: -1 } )
{
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
```

```
> db.products.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

2.Specify the name to the Index

Example – Create index with the index name
 db.products.createIndex(

```
{ item: 1, quantity: -1 } ,
{ name: "query for inventory" }
)

> db.products.createIndex( { item: 1, quantity: -1 } , { name: "query for inventory" } )
{
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "createdCollectionAutomatically" : true,
    "ok" : 1
}

> db.products.getIndexes()
[
    {
        "v" : 2,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_"
    },
    {
        "v" : 2,
        "key" : {
            "item" : 1,
            "quantity" : -1
        },
        "name" : "query for inventory"
    }
]
```

3.dropIndex() Method Example

Example 1 – Drop index by index document

```
db.products.dropIndex(
{ item: 1, quantity: -1 }

)
> db.products.dropIndex(
... { item: 1, quantity: -1 }
...
{ "nIndexesWas" : 2, "ok" : 1 }
```

Example 2 – Drop index by index name

```
db.products.dropIndex( "query for inventory" )
```

```
> db.products.dropIndex( "query for inventory" )
{ "nIndexesWas" : 2, "ok" : 1 }
```

Result

The program was executed and the result was successfully obtained. Thus CO5 was obtained

Experiment No: 15

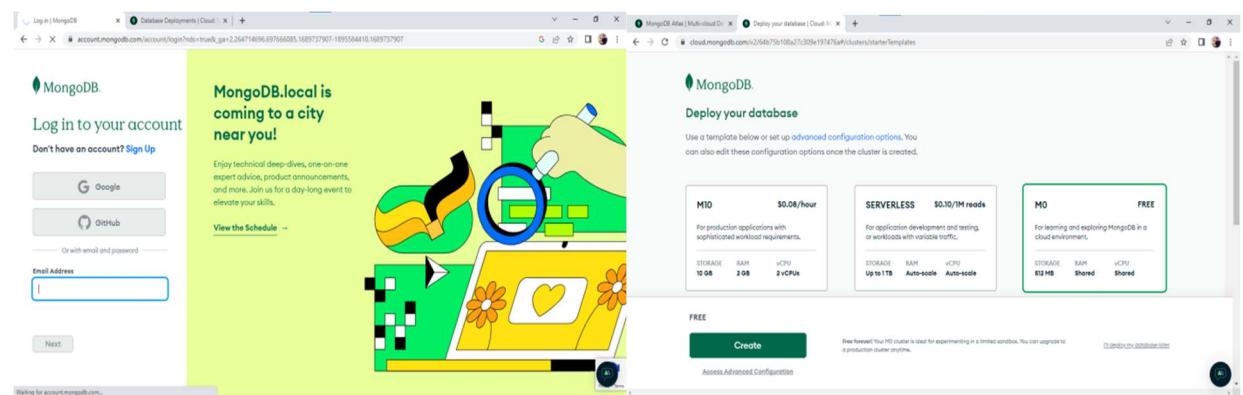
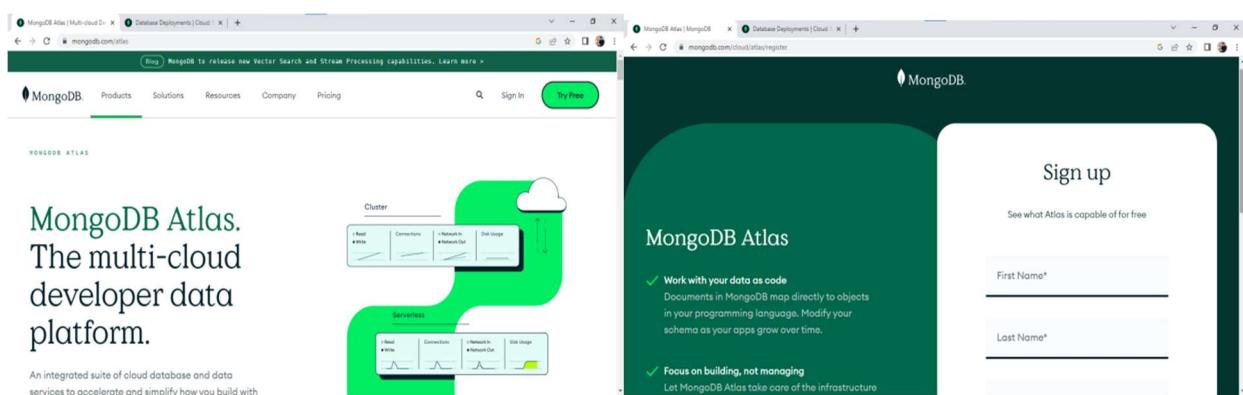
Aim

Usage of Cloud Storage Management Systems: MongoDB Atlas

CO6

Design and deployment of NoSQL databases with real time requirements.

Procedure



Provider





Region ★ Recommended region ⓘ

 Mumbai (ap-south-1) ★

Name
You cannot change the name once the cluster is created.

Tag (optional)

FREE

Create

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

I'll deploy my database later

[Access Advanced Configuration](#)

Name
You cannot change the name once the cluster is created.

Security Quickstart | Cloud Mon: +

cloud.mongodb.com/v2/6deeb6b6bfcc73d96e0784ff/upgrade/access?includetost=true

Atlas Jenny's Org ... Access Manager Billing All Clusters Get Help Jenny

Project 0 Data Services App Services Charts

JENNY'S ORG - 2023-08-30 - PROJECT 0

Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. [Learn more about security setup](#)

1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password **Certificate**

We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information.

Create a database user using a username and password. Users will be given the read and write to any database privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username:
Password: Rgo5QGeSK9uT3V80

MO Cluster Provisioning... We're putting final touches on your cluster.

Data Services App Services Charts

Info We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information. **X**

Create a database user using a username and password. Users will be given the *read and write to any database privilege* by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username
tinuclaraemmanuel

Password 
WDN09849tGOpwyCu  

Create User

Atlas TINU CLARA... Access Manager Billing All Clusters Get Help TINU CLARA EMMANUEL

Project 0 Data Services App Services Charts

DEPLOYMENT

- Database**
 - Data Lake
- Services**
 - Device Sync
 - Triggers
 - Data API
 - Data Federation
 - Search
 - Stream Processing
- Security**
 - Backup
 - Database Access
 - Network Access

Visualize Your Data

Build dashboards and charts, and embed them in your apps with MongoDB Charts.

Metrics

Category	Value	Last 7 minutes
Read Operations (R)	0	100.0 B/s
Write Operations (W)	0	100.0 B/s
Connections	0	Lost 7 minutes
Inbound Bandwidth (In)	0.0 B/s	100.0 Bi/s
Outbound Bandwidth (Out)	0.0 B/s	100.0 Bi/s
Data Size	0.0 B / 512.0 MB (0%)	512.0 MB

Cluster Details

Setting	Value
VERSION	6.0.8
REGION	AWS / Mumbai (ap-south-1)
CLUSTER TIER	M0 Sandbox (General)
TYPE	Replica Set - 3 nodes
BACKUPS	Inactive
LINKED APP SERVICES	None Linked
ATLAS SQL	Connect
ATLAS SEARCH	Create Index

System Status: All Good

©2023 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

The screenshot shows the MongoDB Atlas interface. At the top, a 'Connect to Cluster0' wizard is open, divided into three steps: 'Set up connection security' (completed), 'Choose a connection method' (in progress), and 'Connect' (not yet reached). Below the wizard, there are sections for 'Connect to your application' (Drivers, Shell, MongoDB for VS Code, Atlas SQL) and 'Access your data through tools' (Compass, Shell, MongoDB for VS Code, Atlas SQL). The main dashboard below shows the 'atlasuser.atlas1' database with storage details (STORAGE SIZE: 4KB, LOGICAL DATA SIZE: 85B, TOTAL DOCUMENTS: 1, INDEXES TOTAL SIZE: 4KB). It includes tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A query result is displayed: `_id: ObjectId('64b766e79ff32a5fcc03eb90')
name: "tinu"
email: "tinu@gmail.com"
place: "mundakayam"`. On the left sidebar, there are sections for Deployment, Database, Services (Device Sync, Triggers, Data API, Data Federation, Search, Stream Processing), Security (Backup, Database Access), and App Services.

APP services

The screenshot shows the Realm App Services interface. The left sidebar includes sections for App Services, NO ENVIRONMENT, DATA ACCESS (Rules, Schema, App Users, Authentication), BUILD (Realm SDKs, Device Sync, GraphQL, Functions, Triggers, HTTPS Endpoints, Values), and MANAGE (Linked Data Sources, Deployment, Hosting, Logs). The main content area features a section titled 'Build better apps faster with App Services' with a sub-section 'Start with an app template'. It lists four templates: 'Build your own App' (Set up your own app services to fit your development needs), 'Real-time Sync' (Mobile app using a Realm SDK to sync data to your backend), 'Manage Database Views' (Event-driven Database Trigger template to update a view in a separate collection), and 'React.JS + Realm Web Boilerplate' (Todo web app using the Realm Web SDK). A 'Deployment' section on the right shows a GitHub integration.

The screenshot shows the MongoDB App Services interface. On the left, a sidebar lists various services: NO ENVIRONMENT, DATA ACCESS (Rules, Schema, App Users, Authentication), BUILD (Realm SDKs, Device Sync, GraphQL, Functions, Triggers, HTTPS Endpoints, Values), and MANAGE (Linked Data Sources, Deployment, Hosting, Logs, App Settings). The main area is titled "Build better apps faster with App Services" and guides the user through four steps: 1. Select a Template (React.JS + Realm Web Boilerplate), 2. Link your Data Source (Cluster0), 3. Name your Application (Application-0), and 4. App Deployment Model (Single Region - Virginia (us-east-1) • AWS). Below this, the "App Services" tab is selected in the navigation bar, and the "Applications" section shows "Total App Usage" for Jul 01 - Jul 31. At the bottom, a "Create an App Service" form is displayed, asking for a name ("Application-1") and linking it to an existing MongoDB Atlas Data Source ("Cluster0").

The screenshot shows the OCI App Services interface for a project named 'Project 0'. On the left sidebar, under the 'DATA ACCESS' section, the 'App Users' option is selected. The main content area displays 'Application-1' with an App ID of 'application-1-dhjm' and an environment of 'AWS • Mumbai (ap-south-1)'. A 'Get Started With App Services' panel is open, featuring four sections: 'REALM CLI' (with a sub-link to 'Enable Local Development'), 'SDKS' (with a sub-link to 'Install SDKs'), 'DOCUMENTATION' (with a sub-link to 'Visit App Services documentation'), and 'DEPLOYMENT' (with a sub-link to 'Connect to a Github account and repo to use Git').

Result

The program was executed and the result was successfully obtained. Thus CO5 was obtained

Experiment No: 16

Aim

Implementation of Replica Set on MongoDB

CO5

To what extent you are able to understand the basic storage architecture of distributed file systems.

Procedure

1. Create Replica Set

```
C:\Windows\System32\cmd.exe - mongo --port 27018
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

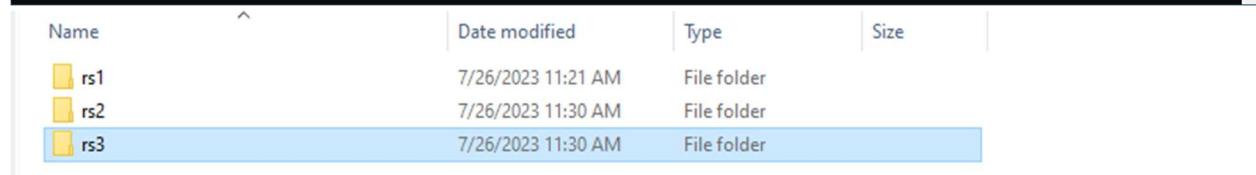
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replSet datascience -logpath \data\rs1\1.log -dbpath \data\rs1 -port 27018
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replSet datascience -logpath \data\rs2\2.log -dbpath \data\rs2 -port 27019
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replSet datascience -logpath \data\rs3\3.log -dbpath \data\rs3 -port 27020

C:\Program Files\MongoDB\Server\5.0\bin>mongo --port 27018
MongoDB shell version v5.0.2
connecting to: mongodb://127.0.0.1:27018/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6999d461-e370-4fc5-a255-3e9d7dde2467") }
MongoDB server version: 5.0.2
-----
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
-----
The server generated these startup warnings when booting:
 2023-07-26T09:32:53.076+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
 2023-07-26T09:32:53.077+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

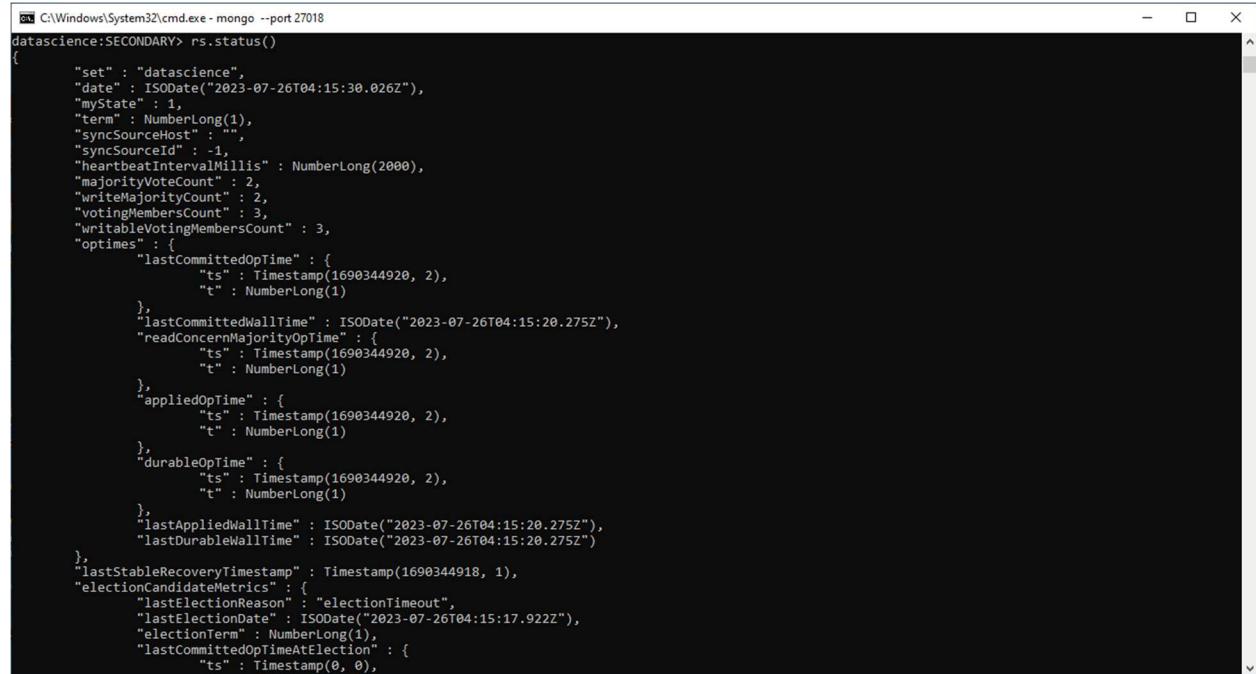
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
```

2. Configure Replica Set



```
C:\Windows\System32\cmd.exe - mongo --port 27018
> config={_id:"datascience",members:[{_id:0,host:"localhost:27018"},{_id:1,host:"localhost:27019"},{_id:2,host:"localhost:27020"}]}
{
    "_id" : "datascience",
    "members" : [
        {
            "_id" : 0,
            "host" : "localhost:27018"
        },
        {
            "_id" : 1,
            "host" : "localhost:27019"
        },
        {
            "_id" : 2,
            "host" : "localhost:27020"
        }
    ]
}
> rs.initiate(config)
{
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1690344907, 1),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1690344907, 1)
}
```

3. Replica Set Status



```
C:\Windows\System32\cmd.exe - mongo --port 27018
datascience:SECONDARY> rs.status()
{
    "set" : "datascience",
    "date" : ISODate("2023-07-26T04:15:30.026Z"),
    "myState" : 1,
    "term" : NumberLong(1),
    "syncSourceHost" : "",
    "syncSourceID" : -1,
    "heartbeatIntervalMillis" : NumberLong(2000),
    "majorityVoteCount" : 2,
    "writeMajorityCount" : 2,
    "votingMembersCount" : 3,
    "unavailableVotingMembersCount" : 0,
    "optimes" : {
        "lastCommittedOpTime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "lastCommittedWallTime" : ISODate("2023-07-26T04:15:20.275Z"),
        "readConcernMajorityOptime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "appliedOptime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "durableOptime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "lastAppliedWallTime" : ISODate("2023-07-26T04:15:20.275Z"),
        "lastDurableWallTime" : ISODate("2023-07-26T04:15:20.275Z")
    },
    "lastStableRecoveryTimestamp" : Timestamp(1690344918, 1),
    "electionCandidateMetrics" : {
        "lastElectionReason" : "electionTimeout",
        "lastElectionDate" : ISODate("2023-07-26T04:15:17.922Z"),
        "electionTerm" : NumberLong(1),
        "lastCommittedOptimeAtElection" : {
            "ts" : Timestamp(0, 0)
        }
    }
}
```

4. List Replica Set

```
C:\Windows\System32\cmd.exe - mongo --port 27018
{
  "members": [
    {
      "_id": 0,
      "name": "localhost:27018",
      "health": 1,
      "state": 1,
      "stateStr": "PRIMARY",
      "uptime": 759,
      "optime": {
        "ts": Timestamp(1690344920, 2),
        "t": NumberLong(1)
      },
      "optimeDate": ISODate("2023-07-26T04:15:20Z"),
      "syncSourceHost": "",
      "syncSourceId": -1,
      "infoMessage": "Could not find member to sync from",
      "electionTime": Timestamp(1690344917, 1),
      "electionDate": ISODate("2023-07-26T04:15:17Z"),
      "configVersion": 1,
      "configTerm": 1,
      "self": true,
      "lastHeartbeatMessage": ""
    }
  ],
  "ok": 1,
  "$clusterTime": {
    "clusterTime": Timestamp(1690344920, 2),
    "signature": {
      "hash": BinData(0, "AAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId": NumberLong(0)
    }
  },
  "operationTime": Timestamp(1690344920, 2)
}
```

```
Select C:\Windows\System32\cmd.exe - mongo --port 27018
{
  "_id": 1,
  "name": "localhost:27019",
  "health": 1,
  "state": 2,
  "stateStr": "SECONDARY",
  "uptime": 22,
  "optime": {
    "ts": Timestamp(1690344920, 2),
    "t": NumberLong(1)
  },
  "optimeDurable": {
    "ts": Timestamp(1690344920, 2),
    "t": NumberLong(1)
  },
  "optimeDate": ISODate("2023-07-26T04:15:20Z"),
  "optimeDurableDate": ISODate("2023-07-26T04:15:20Z"),
  "lastHeartbeat": ISODate("2023-07-26T04:15:30.008Z"),
  "lastHeartbeatRecv": ISODate("2023-07-26T04:15:29.078Z"),
  "pingMs": NumberLong(0),
  "lastHeartbeatMessage": "",
  "syncSourceHost": "localhost:27018",
  "syncSourceId": 0,
  "infoMessage": "",
  "configVersion": 1,
  "configTerm": 1
},
```

```
Select C:\Windows\System32\cmd.exe - mongo --port 27018
{
  "_id": 2,
  "name": "localhost:27020",
  "health": 1,
  "state": 2,
  "stateStr": "SECONDARY",
  "uptime": 22,
  "optime": {
    "ts": Timestamp(1690344920, 2),
    "t": NumberLong(1)
  },
  "optimeDurable": {
    "ts": Timestamp(1690344920, 2),
    "t": NumberLong(1)
  },
  "optimeDate": ISODate("2023-07-26T04:15:20Z"),
  "optimeDurableDate": ISODate("2023-07-26T04:15:20Z"),
  "lastHeartbeat": ISODate("2023-07-26T04:15:30.008Z"),
  "lastHeartbeatRecv": ISODate("2023-07-26T04:15:29.050Z"),
  "pingMs": NumberLong(0),
  "lastHeartbeatMessage": "",
  "syncSourceHost": "localhost:27018",
  "syncSourceId": 0,
  "infoMessage": "",
  "configVersion": 1,
  "configTerm": 1
}
```

5. Performing operations at Primary

```
cmd Select C:\Windows\System32\cmd.exe - mongo --port 27018
dataScience:PRIMARY> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
dataScience:PRIMARY> use dataSciDB
switched to db dataSciDB
dataScience:PRIMARY> db.student.insert({name:"amal"})
WriteResult({ "nInserted" : 1 })
dataScience:PRIMARY> db.student.find()
{ "_id" : ObjectId("64c09e557696f07607f0cff"), "name" : "amal" }
dataScience:PRIMARY> use admin
switched to db admin
> db.shutdownServer()
server should be down...
> db.startServer()
uncaught exception: TypeError: db.startServer is not a function :
@(shell):1:1
> use admin
switched to db admin
```

6. Performing operations at Secondary

```
cmd C:\Windows\System32\cmd.exe - mongo --port 27019
C:\Program Files\MongoDB\Server\5.0\bin>mongo --port 27019
MongoDB shell version v5.0.2
connecting to: mongodb://127.0.0.1:27019/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5da5f244-9f73-435c-bf1e-5b838f2f77a1") }
MongoDB server version: 5.0.2
-----
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
-----
The server generated these startup warnings when booting:
 2023-07-26T09:33:56.375+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
 2023-07-26T09:33:56.376+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip
d_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, sta
rt the server with --bind_ip 127.0.0.1 to disable this warning
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
dataScience:SECONDARY> show dbs
uncaught exception: Error: listDatabases failed:
{
  "topologyVersion" : {
    "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
    "counter" : NumberLong(4)
  },
  ...
}
{
  "ok" : 0,
  "errmsg" : "not master and slaveOk=false",
  "code" : 13435,
  "codeName" : "NotPrimaryNoSecondaryOk",
  "$clusterTime" : {
    "clusterTime" : Timestamp(1690345198, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  },
  "operationTime" : Timestamp(1690345198, 1)
}:
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:145:19
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:97:12
shellHelper.show@src/mongo/shell/utils.js:956:13
shellHelper@src/mongo/shell/utils.js:838:15
@shellHelp2:i:1
dataScience:SECONDARY> rs.slaveOk()
WARNING: slaveOk() is deprecated and may be removed in the next major release. Please use secondaryOk() instead.
dataScience:SECONDARY> rs.secondaryOk()
dataScience:SECONDARY> show dbs
admin 0.000GB
config 0.000GB
dataSciDB 0.000GB
local 0.000GB
dataScience:SECONDARY> use dataSciDB
switched to db dataSciDB
dataScience:SECONDARY> db.student.insert({name:"vimal"})
WriteCommandError({
  "topologyVersion" : {
    "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
    "counter" : NumberLong(4)
  },
  "ok" : 0,
```

```
cmd C:\Windows\System32\cmd.exe - mongo --port 27019
{
  "ok" : 0,
  "errmsg" : "not master and slaveOk=false",
  "code" : 13435,
  "codeName" : "NotPrimaryNoSecondaryOk",
  "$clusterTime" : {
    "clusterTime" : Timestamp(1690345198, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  },
  "operationTime" : Timestamp(1690345198, 1)
}:
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:145:19
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:97:12
shellHelper.show@src/mongo/shell/utils.js:956:13
shellHelper@src/mongo/shell/utils.js:838:15
@shellHelp2:i:1
dataScience:SECONDARY> rs.slaveOk()
WARNING: slaveOk() is deprecated and may be removed in the next major release. Please use secondaryOk() instead.
dataScience:SECONDARY> rs.secondaryOk()
dataScience:SECONDARY> show dbs
admin 0.000GB
config 0.000GB
dataSciDB 0.000GB
local 0.000GB
dataScience:SECONDARY> use dataSciDB
switched to db dataSciDB
dataScience:SECONDARY> db.student.insert({name:"vimal"})
WriteCommandError({
  "topologyVersion" : {
    "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
    "counter" : NumberLong(4)
  },
  "ok" : 0,
```

```
C:\Windows\System32\cmd.exe - mongo --port 27019
{
    "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
    "counter" : NumberLong(4),
},
"ok" : 0,
"errmsg" : "not master",
"code" : 10107,
"codeName" : "NotWritablePrimary",
"$clusterTime" : {
    "clusterTime" : Timestamp(1690345328, 1),
    "signature" : {
        "hash" : BinData(0, "AAAAAAAAAAAAAAAAAAAAAAA="),
        "keyId" : NumberLong(0)
    }
},
"operationTime" : Timestamp(1690345328, 1)
})
dataScience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
dataScience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
dataScience:PRIMARY> db.student.insert({name:"akhil"})
WriteResult({ "nInserted" : 1 })
dataScience:PRIMARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
{ "_id" : ObjectId("64c0a01023151b8630c0e8860"), "name" : "akhil" }
dataScience:PRIMARY> use admin
switched to db admin
dataScience:PRIMARY> db.shutdownServer()
server should be down...
> -
```

```
C:\Windows\System32\cmd.exe - mongo --port 27020
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\MongoDB\Server\5.0\bin>mongo --port 27020
MongoDB shell version v5.0.2
connecting to: mongod://127.0.0.1:27020/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6b712f84-8359-4df2-bd62-d5663e2cc7a7") }
MongoDB server version: 5.0.2
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====

The server generated these startup warnings when booting:
2023-07-26T09:34:04.417+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2023-07-26T09:34:04.417+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --
bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desire
d, start the server with --bind_ip 127.0.0.1 to disable this warning
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

dataScience:SECONDARY> rs.secondaryOk()
dataScience:SECONDARY> show dbs
admin      0.000GB
config     0.000GB
dataSciDb  0.000GB
local      0.000GB
dataScience:SECONDARY> use dataSciDb
```

```
C:\Windows\System32\cmd.exe - mongo --port 27020
dataScience:SECONDARY> use dataSciDb
switched to db dataSciDb
dataScience:SECONDARY> db.student.insert({name:"vikas"})
WriteCommandError({
    "topologyVersion" : {
        "processId" : ObjectId("64c09b33cf106257bbb46f97"),
        "counter" : NumberLong(5)
    },
    "ok" : 0,
    "errmsg" : "not master",
    "code" : 10107,
    "codeName" : "NotWritablePrimary",
    "$clusterTime" : {
        "clusterTime" : Timestamp(1690345478, 1),
        "signature" : {
            "hash" : BinData(0, "AAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1690345478, 1)
})
dataScience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
dataScience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
{ "_id" : ObjectId("64c0a01023151b8630c0e8860"), "name" : "akhil" }
dataScience:SECONDARY> -
```

Result

The program was executed and the result was successfully obtained. Thus CO5 was obtained