

Experiment No: 1Aim

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

```
Query 1
1 create database office;
```

2. USE DATABASE

```
Query 1
1 use office;
```

3. CREATE TABLE - EMP

```
Query 1
1 create table emp (
2     eno int primary key, ename varchar(20),
3     ehiredate date, esalary int, ecommission int,
4     check (length(eno)>=3), check (esalary<=5000)
5 );
```

4. INSERT VALUES INTO TABLE - EMP

```
Query 1
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 | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	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
*	HULL	HULL	HULL	HULL	HULL	HULL

6. DROP COLUMN - SAL

Query 1

```

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

```

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

	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
*	HULL	HULL	HULL	HULL	HULL

7. RENAME COLUMN - ename to name

Query 1

```

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

```

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

	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
*	HULL	HULL	HULL	HULL	HULL

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 pane shows the table "Tables_in_office" with one entry: "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 pane shows the table "emp_details" with columns eno, name, ehiredate, esalary, and ecommission, all containing NULL values.

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 pane shows the following output:

- Line 8: 12:16:37 drop table emp_details 0 row(s) affected
- Line 9: 12:16:37 SELECT * FROM emp_details LIMIT 0, 1000 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



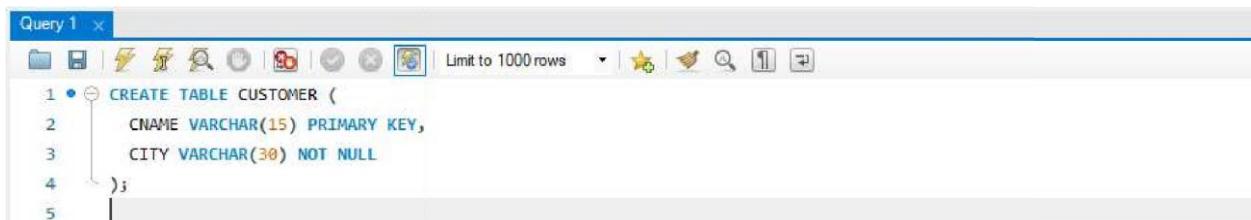
```
Query 1
use bank;
```

3. CREATE TABLE - BRANCH



```
Query 1
CREATE TABLE BRANCH (
    BNAME VARCHAR(20) PRIMARY KEY,
    CITY VARCHAR(30) CHECK(CITY IN ('NAGPUR', 'DELHI', 'BANGLORE', 'BOMBAY')) NOT NULL
);
```

4. CREATE TABLE - CUSTOMER



```
Query 1
CREATE TABLE CUSTOMER (
    CNAME VARCHAR(15) PRIMARY KEY,
    CITY VARCHAR(30) NOT NULL
);
```

5. CREATE TABLE - DEPOSIT

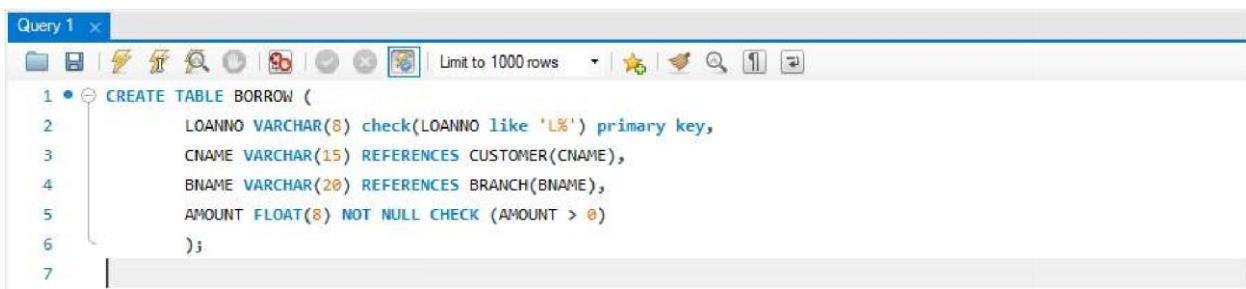


```

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

```

6. CREATE TABLE - BORROW

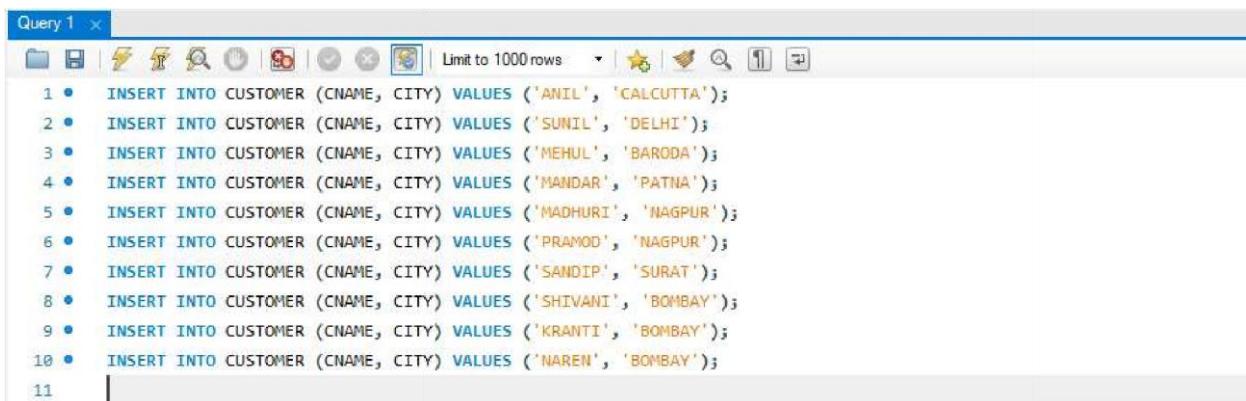


```

Query 1 ×
CREATE TABLE BORROW (
    LOANNO VARCHAR(8) check(LOANNO like 'L%') primary key,
    CNAME VARCHAR(15) REFERENCES CUSTOMER(CNAME),
    BNAME VARCHAR(20) REFERENCES BRANCH(BNAME),
    AMOUNT FLOAT(8) NOT NULL CHECK (AMOUNT > 0)
);

```

7. INSERT VALUES IN TABLE - CUSTOMER



```

Query 1 ×
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('ANIL', 'CALCUTTA');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SUNIL', 'DELHI');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MEHUL', 'BARODA');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MANDAR', 'PATNA');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MADHURI', 'NAGPUR');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('PRAMOD', 'NAGPUR');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SANDIP', 'SURAT');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SHIVANI', 'BOMBAY');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('KRANTI', 'BOMBAY');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('NAREN', 'BOMBAY');

```

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

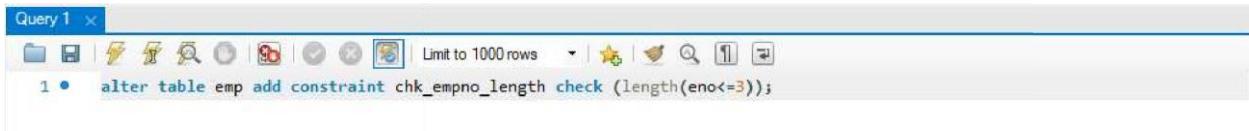
Query 1

```

1 • alter table emp add constraint chk_salary check (esalary >= 1500);

```

12. ADD CONSTRAINT - length(eno) >= 3



```
Query 1
1 • alter table emp add constraint chk_empno_length check (length(eno)<=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 fundamental tasks involved with modeling, designing and implementing a database.

Procedure

1. List all data from table deposit.

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is "SELECT * FROM Deposit;". Below the query, the results are displayed in a "Result Grid" table. The table has columns: ACTNO, CNAME, BNAME, AMOUNT, and ADATE. The data consists of 10 rows, each representing a deposit entry with values corresponding to the columns.

	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

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

```
1  SELECT * FROM borrows;
```

The results are displayed in a "Result Grid" table:

	LOANNO	CNAME	BNAME	AMOUNT
▶	L201	ANIL	VRCE	1000
	L206	MEHUL	AJNI	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

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

```
1  SELECT * FROM customer;
```

The results are displayed in a "Result Grid" table:

	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

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1" containing the SQL command: "SELECT * FROM branch;". Below the editor is a "Result Grid" displaying the data from the branch table. The table has two columns: BNAME and CITY. The data rows are:

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

5. Give account no and amount of deposit

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1" containing the SQL command: "SELECT actno, amount FROM deposit;". Below the editor is a "Result Grid" displaying the data from the deposit table. The table has two columns: actno and amount. The data rows are:

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

Query 1

```
1 •  SELECT CNAME, ACTNO FROM DEPOSIT;
```

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

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

Query 1

```
1 •  SELECT CNAME FROM CUSTOMER;
```

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

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

8. Give name of branches

Query 1

```
1  SELECT BNAME FROM BRANCH;
```

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

BNAME
AJNI
ANDHERI
CHANDNI
DHARAMPETH
KAROLBAGH
MG ROAD
NEHRU PALACE
POWAI
VRCE
NULL

9. Give name of borrows

Query 1

```
1  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

Query 1

```
1 • SELECT CNAME FROM CUSTOMER WHERE CITY = 'NAGPUR';  
Execute the selected portion of the script or everything, if there is no selection
```

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

CNAME
MADHURI
PRAMOD
NULL

11. Give names of depositors having amount greater than 4000

Query 1

```
1 • SELECT CNAME FROM DEPOSIT WHERE AMOUNT > 4000;
```

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

CNAME
KRANTI
MINU

12. Give account date of Anil

Query 1

```
1 • SELECT ADATE FROM DEPOSIT WHERE CNAME = 'Anil';
```

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

ADATE
1995-03-01

13. Give name of all branches located in Bombay

Query 1

```
1  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

Query 1

```
1  SELECT CNAME FROM BORROW WHERE LOANNO = '1205';
```

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

CNAME

15. Give names of depositors having account at VRCE

Query 1

```
1  SELECT CNAME FROM DEPOSIT WHERE BNAME = 'VRCE';
```

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

CNAME
ANIL

16. Give names of all branched located in city Delhi

Query 1

```
1   SELECT BNAME FROM BRANCH WHERE CITY = 'DELHI';
```

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

BNAME
CHANDNI
KAROLBAGH
NEHRU PALACE
*
NONE

17. Give name of the customers who opened account date ‘1-12-96’

Query 1

```
1   SELECT CNAME FROM DEPOSIT WHERE ADATE = '1996-12-01';
```

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

CNAME
*
NONE

18. Give account no and deposit amount of customers having account opened between dates ‘1- 12-96’ and ‘1-5-96’

Query 1

```
1 •   SELECT ACTNO, AMOUNT FROM DEPOSIT WHERE ADATE BETWEEN '01-DEC-96' AND '01-MAY-96';
```

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

ACTNO	AMOUNT
*	NONE

19. Give name of the city where branch KAROL BAGH is located

Query 1

```
1   SELECT CITY FROM BRANCH WHERE BNAME = 'KAROLBAGH';
2
```

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

CITY
DELHI

20. Give details of customer ANIL

Query 1

```
1 •  SELECT * FROM DEPOSIT WHERE CNAME = 'Anil';
```

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

ACTNO	CNAME	BNAME	AMOUNT	ADATE
D100	ANIL	VRCE	1000.00	1995-03-01
*	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 fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

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

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

```
1  SELECT CNAME FROM deposit WHERE CNAME NOT IN (SELECT CNAME FROM borrow);
```

The result grid displays the following data:

CNAME
SANDIP
SHIVANI
MINU

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

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

```
1  SELECT CNAME FROM DEPOSIT UNION (SELECT CNAME FROM BORROW);
```

The result grid displays the following data:

CNAME
ANIL
SUNIL
MEHUL
MADHURI
PRAMOD
SANDIP
SHIVANI
KRANTI
MINU

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

The screenshot shows a SQL query window titled "Query 1". The query is:

```

1  SELECT D1.CNAME FROM DEPOSIT D1 WHERE D1.BNAME IN (SELECT D2.BNAME FROM DEPOSIT D2 WHERE D2.CNAME = 'SUNIL' );
2

```

The result grid shows one row:

CNAME
SUNIL

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

The screenshot shows a SQL query window titled "Query 1". The query is:

```

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

```

The result grid shows one row:

CNAME
MADHURI

5. List all the depositors living in city NAGPUR

The screenshot shows a SQL query window titled "Query 1". The query is:

```

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

```

The result grid shows two rows:

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 | Filter Rows: [] | Export: [] | Wrap Cell Content: []

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 | Filter Rows: [] | Export: [] | Wrap Cell Content: []

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 locate

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 database query interface titled "Query 1". The query itself 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
```

Below the query, there is a "Result Grid" table with three columns: CNAME, AMOUNT, and CITY. The result is a single row:

CNAME	AMOUNT	CITY
ANIL	1000.00	CALCUTTA

At the bottom of the interface, there are buttons for "Result Grid", "Filter Rows", "Export", and "Wrap Cell Content".

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 single query window titled "Query 1". The query is:

```
1 •   SELECT SUM(AMOUNT) FROM BORROW;
```

The results are displayed in a grid:

SUM(AMOUNT)
22000

2. List total deposit

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

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

The results are displayed in a grid:

SUM(AMOUNT)
24200.00

3. List total loan taken from KAROLBAGH branch

Query 1

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

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';
```

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;
```

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
```

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;
```

BNAME	SUM(AMOUNT)
ANDHERI	2000
VIRAR	8000

13. Count total number of customers

Query 1

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

COUNT(*)
10

14. Count total number of depositors branch wise

Query 1

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

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;
```

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);
```

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 fundamental 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 Editor window with the above query. Below the editor is the Result Grid pane, which displays the following data:

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 Editor window with the above query. Below the editor is the Result Grid pane, which displays the following data:

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;
```

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

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
```

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

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

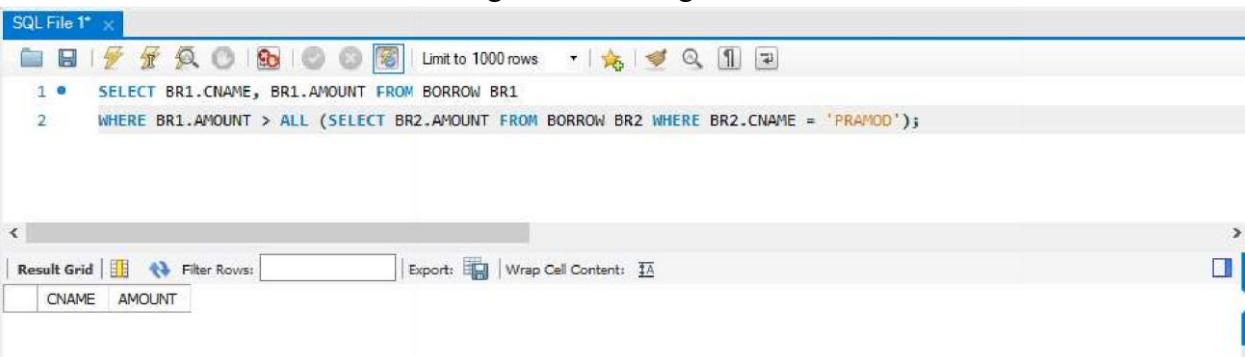
5. Give names of depositors having the same branch as the branch of Sunil

```
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');
```

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

CNAME
SUNIL

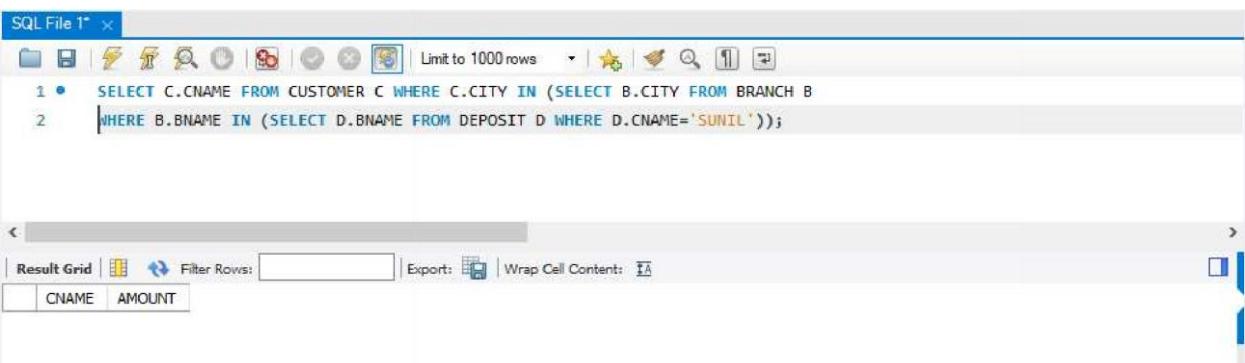
6. Give names of borrowers having loan amount greater than the loan amount of Pramod



```
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 for question 6. The interface includes a toolbar with various icons, a status bar at the bottom, and a results grid below the query window.

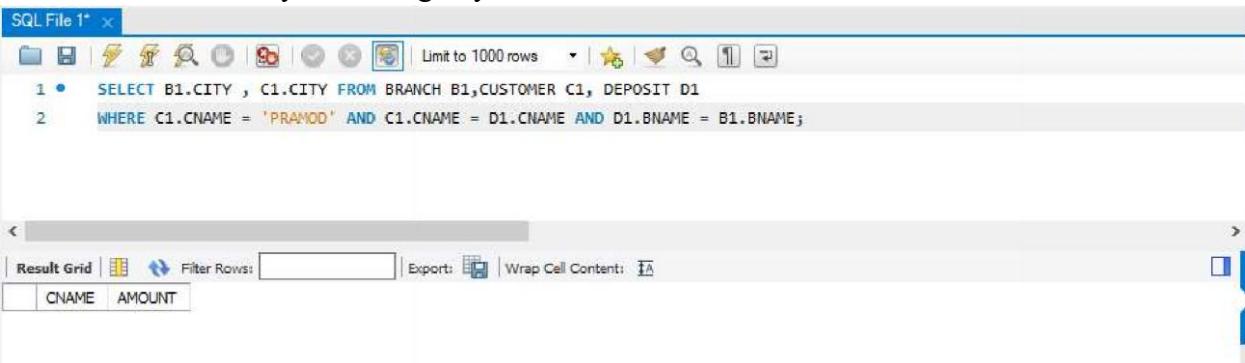
7. Give the name of the customer living in the city where branch of depositor Sunil is located.



```
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 for question 7. The interface includes a toolbar with various icons, a status bar at the bottom, and a results grid below the query window.

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 for question 8. The interface includes a toolbar with various icons, a status bar at the bottom, and a results grid below the query window.

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

The screenshot shows the SQL Developer interface with a single query window titled "SQL File 1". The query is:

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

The result grid displays one row:

CITY
NAGPUR

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

The screenshot shows the SQL Developer interface with a single query window titled "SQL File 1". The query is:

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

The result grid displays three rows:

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

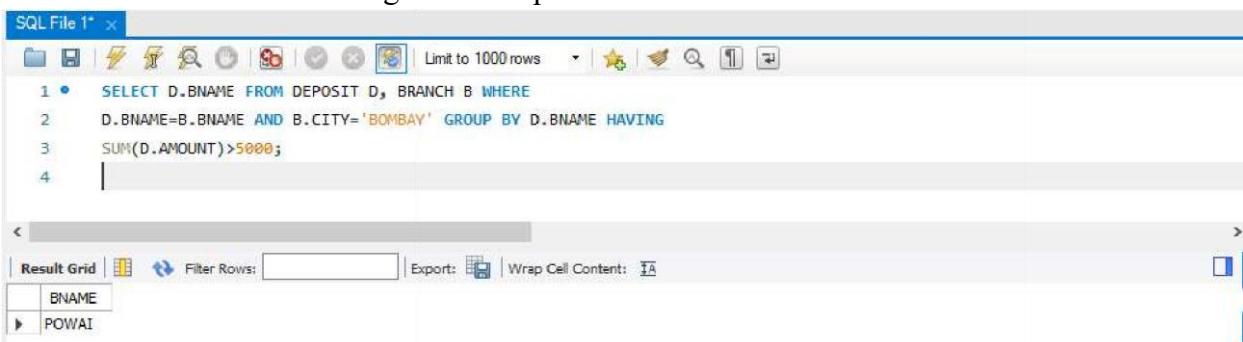
To familiarize with Group By and Having Clauses

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.



The screenshot shows the SQL Developer interface with a SQL file named 'SQL File 1'. The code is:

```

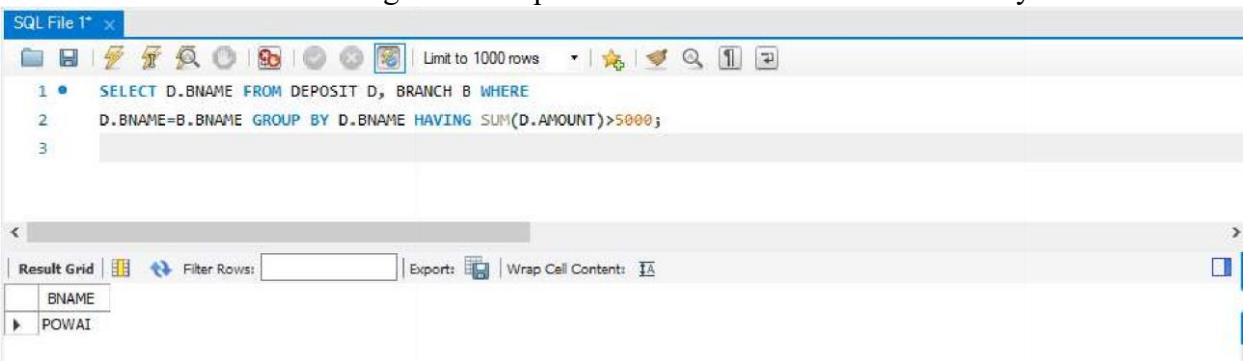
1 • SELECT D.BNAME FROM DEPOSIT D, BRANCH B WHERE
2   D.BNAME=B.BNAME AND B.CITY='BOMBAY' GROUP BY D.BNAME HAVING
3   SUM(D.AMOUNT)>5000;
4

```

The result grid shows one row:

BNAME
POWAI

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



The screenshot shows the SQL Developer interface with a SQL file named 'SQL File 1'. The code is:

```

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 result grid shows one row:

BNAME
POWAI

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 SQL query window titled "SQL File 1*". The query is:

- 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

4. List the names of customers having maximum deposit

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

The screenshot shows a SQL query window titled "SQL File 1*". The query is:

- 1 `SELECT MAX(AMOUNT) CNAME FROM deposit;`

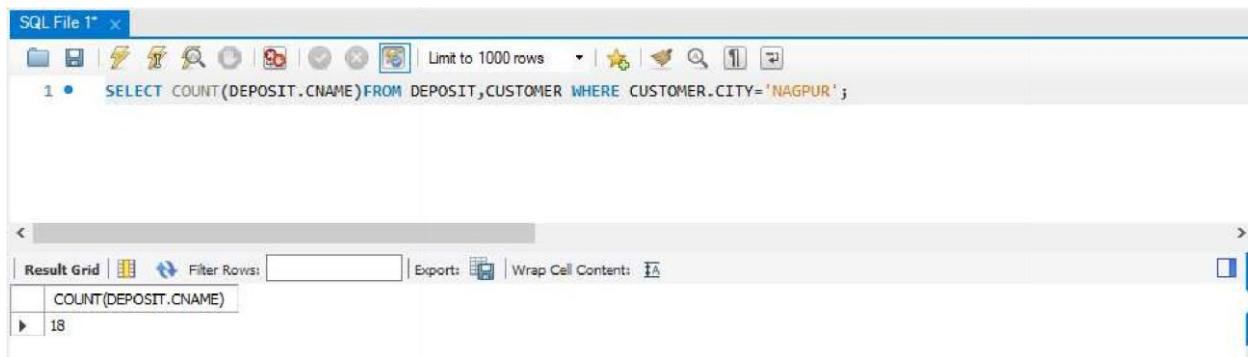
5. List the name of branch having highest number of depositors?

```
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 a SQL query window titled "SQL File 1*". The query is:

- 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

6. Count the number of depositors living in NAGPUR.



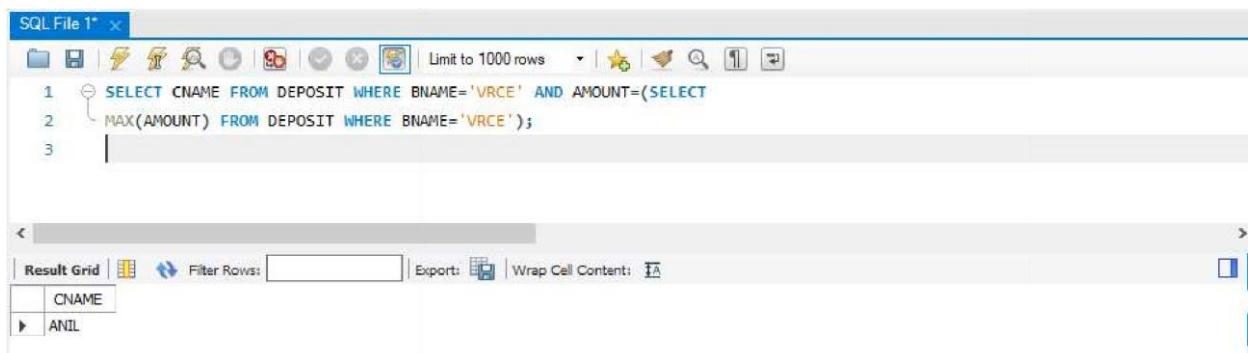
SQL File 1*

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

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

COUNT(DEPOSIT.CNAME)
18

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



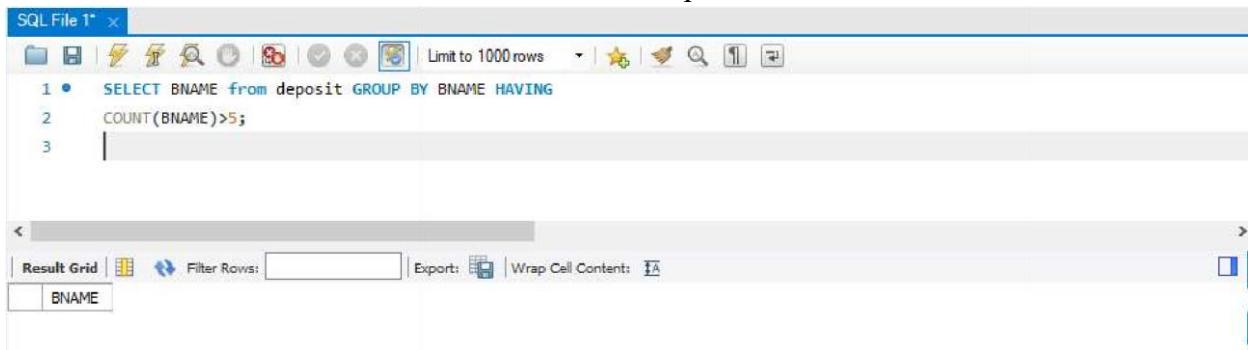
SQL File 1*

```
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

The screenshot shows the SQL Server Management Studio interface with a query window titled "SQL File 1". The query is:

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

The result grid shows the following data:

CNAME	COUNT(B.BNAME)
Chennai	6

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

The screenshot shows the SQL Server Management Studio interface with a query window titled "SQL File 1". The query is:

```
1 • SELECT COUNT(B1.BNAME) FROM DEPOSIT D1 JOIN BORROW B1 ON D1.CID=B1.CID JOIN CUSTOMER C1 WHERE C1.CITY=D1.CITY AND D1.CITY=C1.CITY IN (SELECT CITY FROM CUSTOMER);
```

The result grid shows the following data:

COUNT(B1.BNAME)
6

Result

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

Experiment No: 8

Aim

To have familiarize with trigger functions

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

CO2

Apply PL/SQL for processing databases

Procedure

Step 1: Start

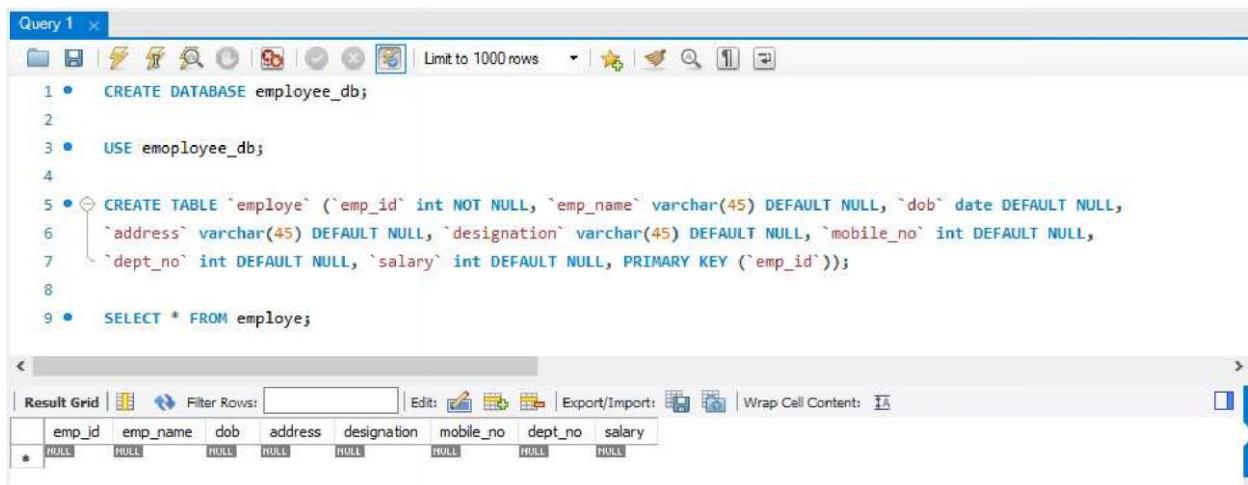
Step 2: Initialize the trigger.

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



The screenshot shows the Oracle SQL Developer interface. The top part displays a query window with the following SQL code:

```

1 • CREATE DATABASE employee_db;
2
3 • USE employee_db;
4
5 • CREATE TABLE `employee`(`emp_id` int NOT NULL, `emp_name` varchar(45) DEFAULT NULL, `dob` date DEFAULT NULL,
6   `address` varchar(45) DEFAULT NULL, `designation` varchar(45) DEFAULT NULL, `mobile_no` int DEFAULT NULL,
7   `dept_no` int DEFAULT NULL, `salary` int DEFAULT NULL, PRIMARY KEY (`emp_id`));
8
9 • SELECT * FROM employee;

```

The bottom part shows a result grid for the query in step 9, which is currently empty, displaying only the column headers:

emp_id	emp_name	dob	address	designation	mobile_no	dept_no	salary
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

employee

```

1 INSERT INTO `employee_db`.`employee`
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 employee;
6

```

Result Grid

emp_id	emp_name	dob	address	designation	mobile_no	dept_no	salary
1	amal	1995-04-29	wayanad	developer	9469664422	7	40000
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table Name: employee Schema: employee_db

Charset/Collation: utf8mb4 Engine: InnoDB

Comments:

BEFORE INSERT
AFTER INSERT
BEFORE UPDATE
▼ AFTER UPDATE
 employee_AFTER_UPDATE
BEFORE DELETE
AFTER DELETE

```

1 • CREATE DEFINER='root'@'localhost'
2 • TRIGGER `employee_AFTER_UPDATE` AFTER UPDATE ON `employee` 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 employee SET salary=50000 WHERE emp_id=1;

```

The screenshot shows a MySQL Workbench interface with a query editor window titled "salary". The query is:

```
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	HULL	HULL	HULL

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 WhoBorrow()
BEGIN
SELECT * FROM BORROW;
END//
```

```
DELIMITER ; call
WhoBorrow()
```

The screenshot shows the Oracle SQL Developer interface. At the top, there's a toolbar with various icons. Below it is a code editor window titled "SQL File 5*". The code in the editor is:

```
1  DELIMITER ;
2  •  call WhoBorrow()
3  |
```

Below the code editor is a result grid window titled "Result Grid". It displays the output of the executed query:

LOANNO	CNAME	BNAME	AMOUNT
L201	ANIL	VRCE	1000.00
L206	MEHUL	AJNI	5000.00
L311	SUNIL	DHARAMPETH	3000.00
L321	MADHURI	ANDHERI	2000.00
L371	PRAMOD	VIRAR	8000.00
L481	KRANTI	NEHRU PLACE	3000.00

On the right side of the interface, there's a sidebar with icons for "Result Grid" (which is selected), "Form Editor", and "Read Only".

Result

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

Experiment No: 10

Aim

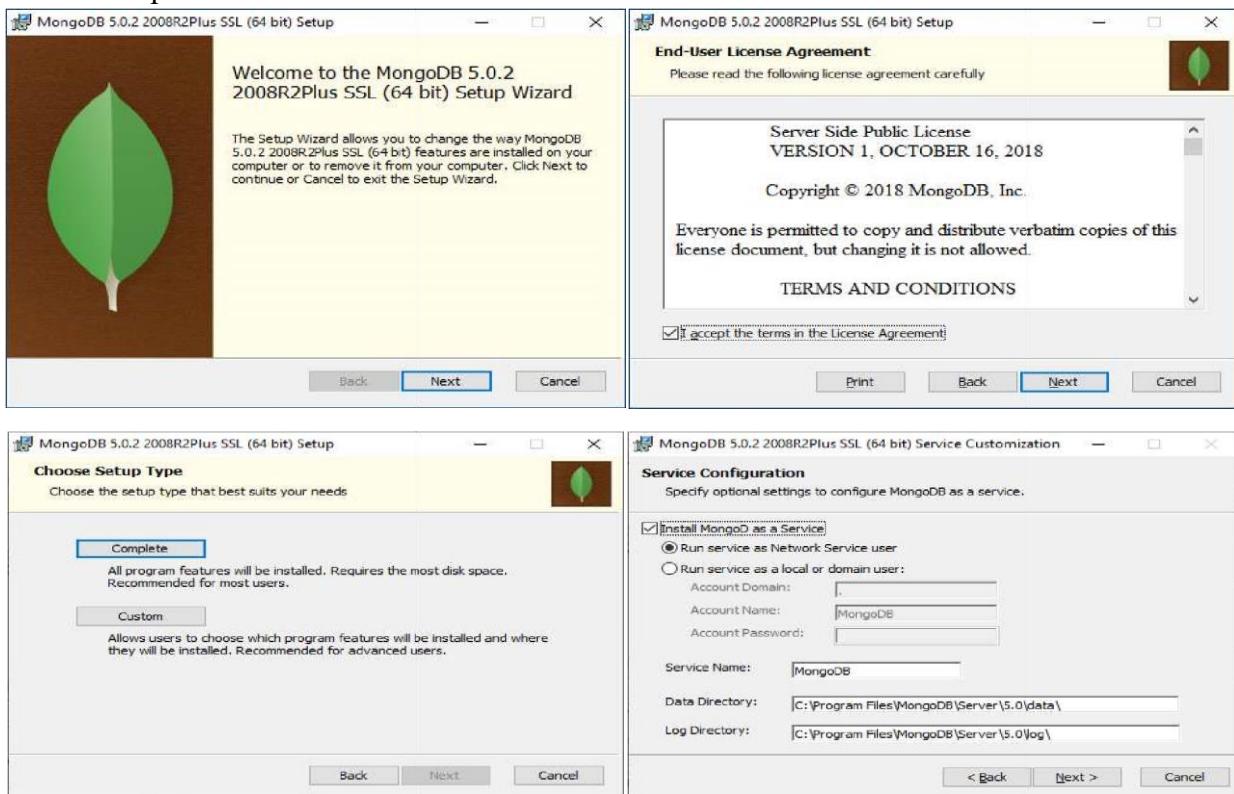
Understand the installation and configuration of NoSQL Databases: MongoDB

CO3

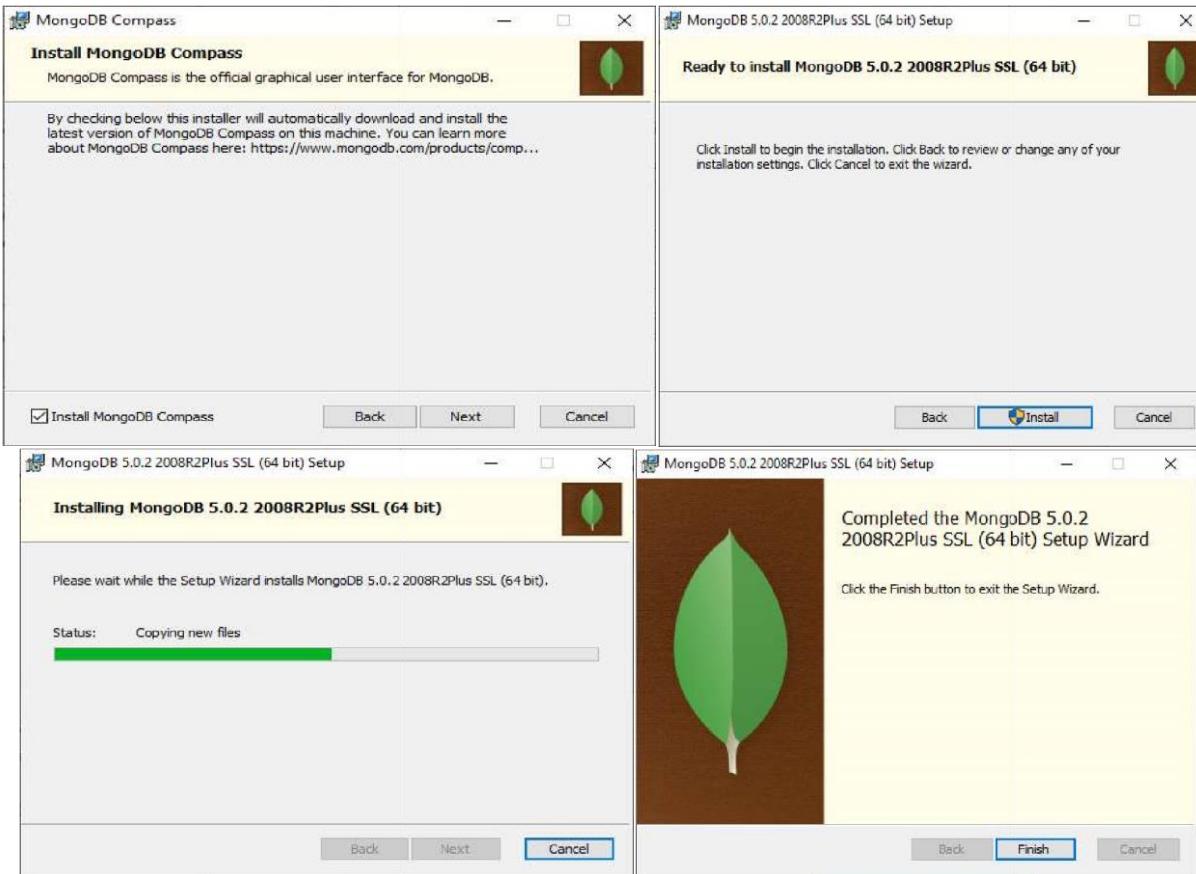
Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

Procedure

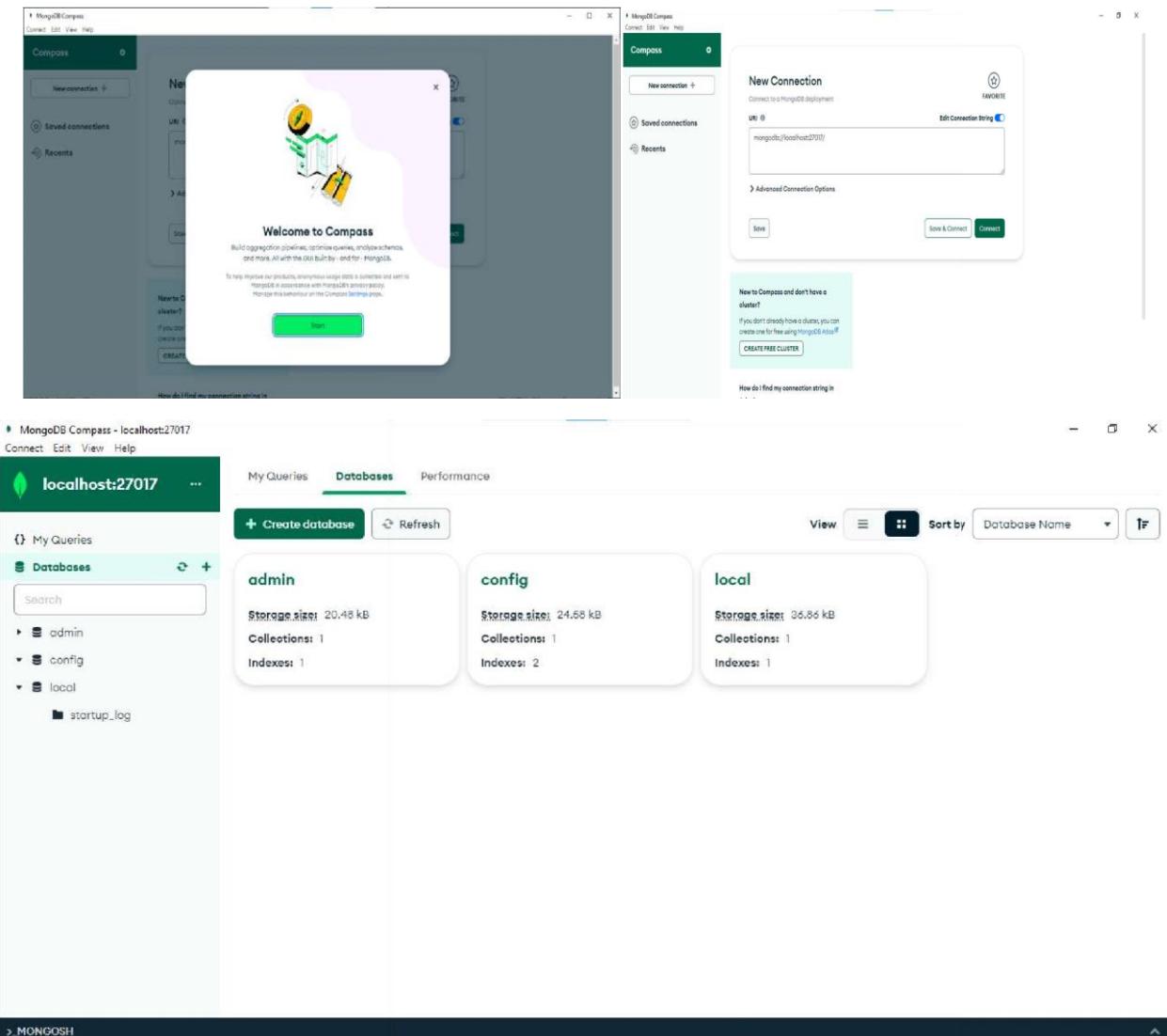
1. Setup Wizard



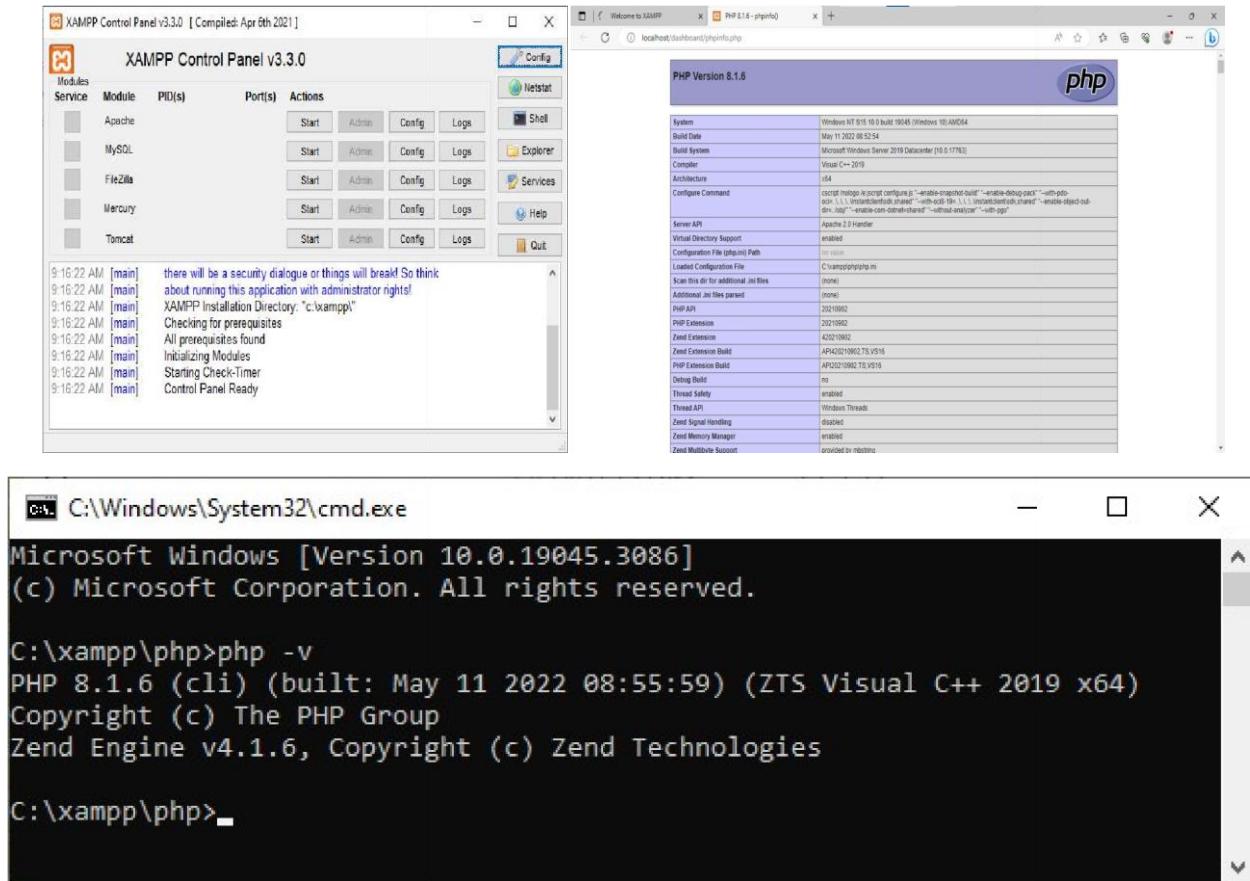
2. Installing MongoDB Compass



3. MongoDB Compass Home Page



4. Configuring MongoDB to Connect to PHP



5. Installing MongoDB Package for PHP

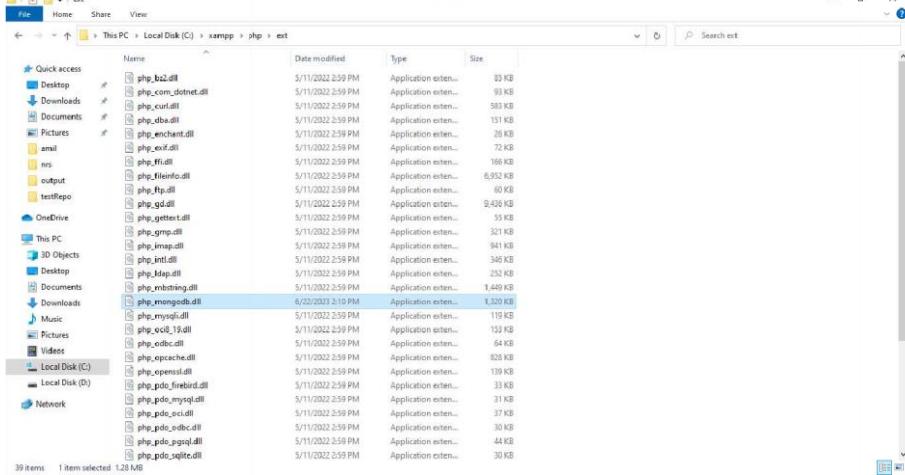
The screenshot shows the PECL package manager interface with the following details:

- Top Level :: Database :: mongodb**
- mongodb** (selected)
- Package Information** table:

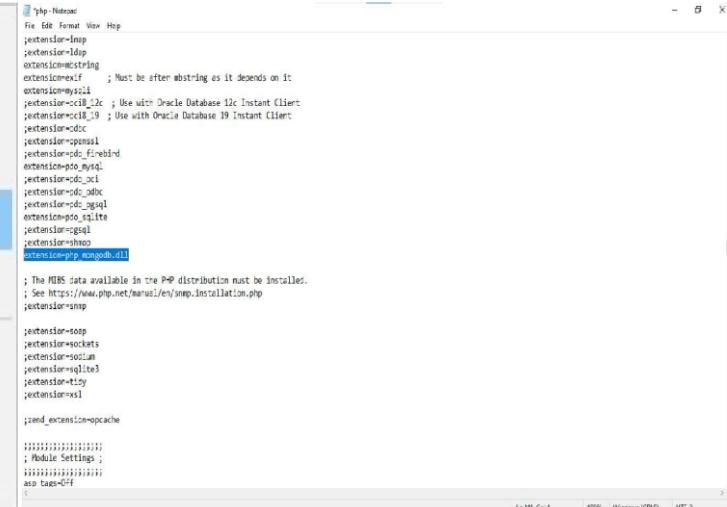
Summary	MongoDB driver for PHP
Maintainers	Jeremy Mikola (lead) [details], Katheme Walker (developer) [details], Andreas Brauer (lead) [details], Derick Rethans < derick at php dot net > (lead) [wishlist] [details], Hannes Magnusson < bjorni at php dot net > (lead) [details]
License	Apache License
Description	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** table:

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 (1701.9kB)
1.15.2	stable	2023-04-21	mongodb-1.15.2.tgz (1702.1kB)
1.15.1	stable	2023-02-09	mongodb-1.15.1.tgz (1701.4kB)

6. Configuring Apache



The screenshot shows a Windows File Explorer window with the path: This PC > Local Disk (C) >xampp > php > ext. The 'Quick access' sidebar lists various folders and drives. The main pane displays a list of files, with 'php_mongodb.dll' highlighted.

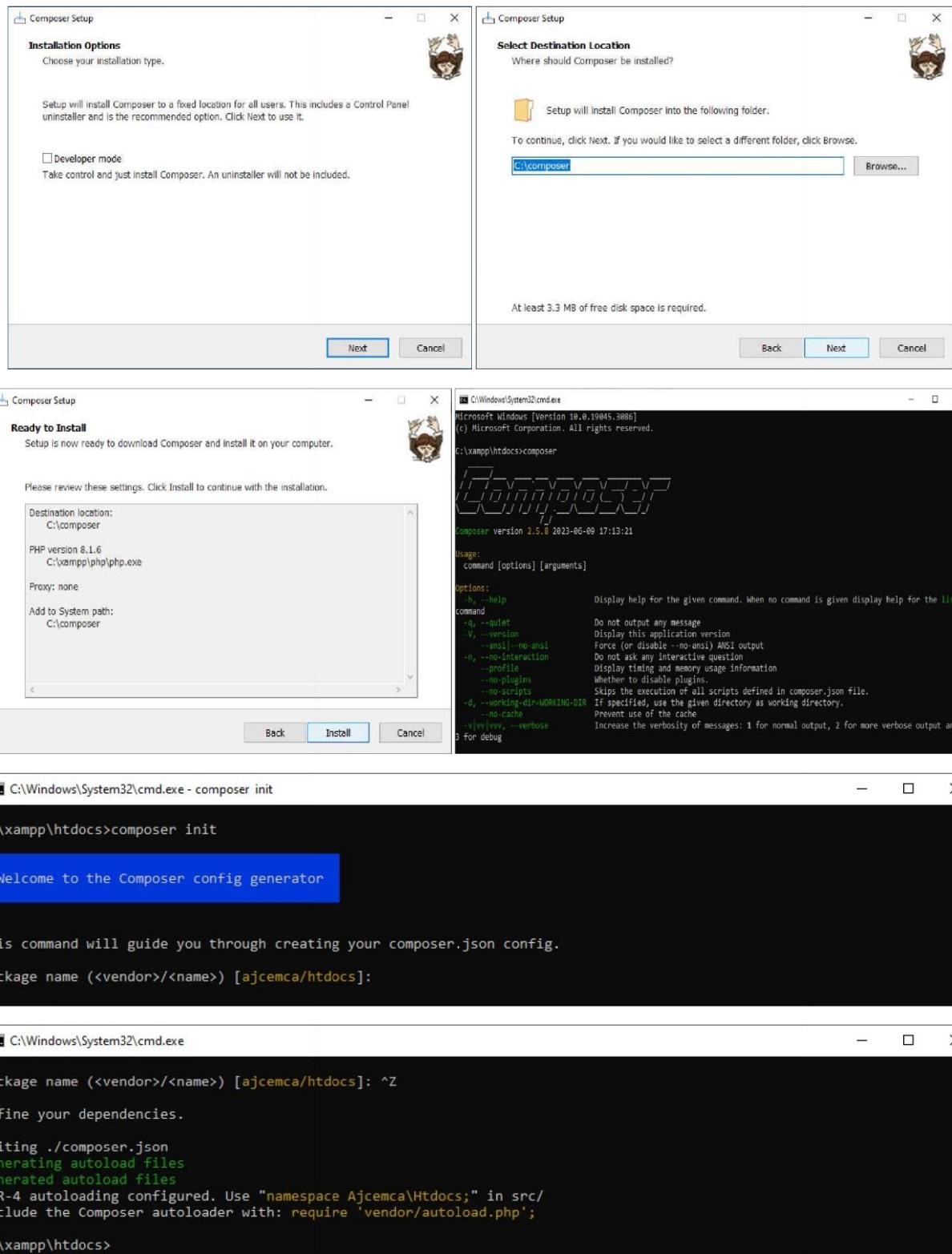


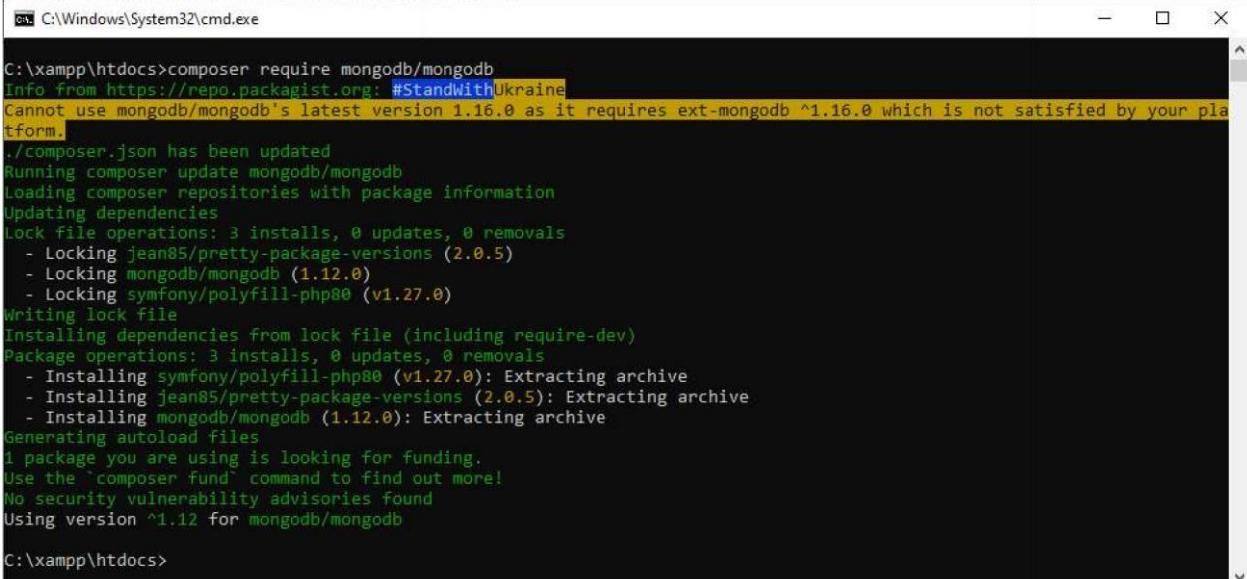
The screenshot shows a Notepad window with the file 'php - Notepad' open. The content is the PHP configuration file 'php.ini'. It includes sections for MySQL, Firebird, PostgreSQL, SQLite, and Oracle Database 12c Instant Client, 19 Instant Client, and 10g Instant Client. It also includes sections for OpenSSL, PDO, and various extensions like mbstring, gd, and curl.

Directive	Local Value	Master Value
mbstring.regex_max_size	100000	100000
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 crypto	enabled	
libmongocrypt crypto library	libcrypto	

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





```
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 -> colors; echo "Database Creation Successful !!!"; echo  
"<br>";  
$collection = $db -> createCollection("VIBGYOR");  
echo "Collection Creation Successful !!!"; echo  
"<br>";  
$collection = $db -> VIBGYOR; echo  
"Collection Selected Successful !!!";  
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 "Data Inserted !!!"; echo "<br>";  
?>
```

Output

The screenshot shows two windows. The top window is a web browser displaying the URL `localhost/amal/color.php`. The page content is:

```
Connection Successful !!!  
Database Creation Successful !!!  
Collection Creation Successful !!!  
Collection Selected Successful !!!  
Data Inserted !!!
```

The bottom window is MongoDB Compass, connected to the database `localhost:27017/colors.VIBGYOR`. It shows the `colors.VIBGYOR` collection with 7 documents and 1 index. The documents are:

- `_id: ObjectId('64e4bda69ab84b6aaa0a5129')`
`color: "Violet"`
- `_id: ObjectId('64e4bda69ab84b6aaa0a512a')`
`color: "Indigo"`
- `_id: ObjectId('64e4bda69ab84b6aaa0a512b')`
`color: "Blue"`
- `_id: ObjectId('64e4bda69ab84b6aaa0a512c')`
`color: "Green"`
- `_id: ObjectId('64e4bda69ab84b6aaa0a512d')`
`color: "Yellow"`
- `_id: ObjectId('64e4bda69ab84b6aaa0a512e')`
`color: "Orange"`

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';  
$connection = new MongoDB\Client("mongodb://localhost:27017");  
echo "Connection Successful !!!"; echo "<br>";  
$db = $connection -> CONTACTS; echo  
"Database Creation Successful !!!"; echo  
"<br>";  
$collection = $db -> createCollection("PROFILE");  
echo "Collection Creation Successful !!!"; echo  
"<br>";  
$collection = $db -> PROFILE; echo  
"Collection Selected Successful !!!";  
?>
```

register.php

```
<title>Registration Form</title>  
<center>  
<br>  
<h1><u>New User Registration</u></h1>  
</center>  
<form action=view.php>  
<input type="submit" value=DISPLAY_USERS>  
</form>  
<center>  
<table class="table2">  
<tr>  
<td>  
<head>  
<h2 style="color:red;" align="center">REGISTRATION FORM</h2>  
<style>
```

```
.table1 {  
    border-style:solid;  
    border-width:3px;  
    font-size: 20px;  
    border-color:black;  
    margin: 10px;  
    padding-top: 15px;  
    padding-bottom: 15px;  
    padding-left: 15px;  
    padding-right: 15px;  
}  
.table2 {  
    padding-top: 150px;  
}  
</style> </head>  
<form action="insert.php" method="POST">  
<table style="background-color:lightskyblue;" class="table1">  
<tr>  
<td> NAME : </td>  
<td> <input type="text" name="name" required> </td>  
</tr>  
<tr>  
<td> EMAIL : </td>  
<td> <input type="text" name="email" required> </td>  
</tr>  
<tr>  
<td> ADDRESS : </td>  
<td> <textarea name="address" rows="5" cols="21" required></textarea></td>  
</tr>  
<tr>  
<td align="right" colspan="2"> <input type="submit" name="submit" value="Register"> </td>  
</tr>  
</table> </form>  
</td>  
</tr>  
</table>  
</center>  
</body>
```

```
insert.php <?php
include_once("connection.php")
; if(isset($_POST["submit"])) {
    $user=array(
        'name'=>$_POST['name'],
        'email'=>$_POST['email'],
        'address'=>$_POST['address']
    );
    $collection->insertOne($user);
echo "Inserted";
} header
("location:view.php");
?>
```

view.php

```
<form action=register.php>
<input type="submit" value=REGISTER_NEW>
</form>
<center> <?php
include_once("connection.php")
;
$result=$collection->find();
?>
<br>
<h2><u> USER DETAILS </u></h2>
<table border="2" style="width:60%">
<tr style="height:50px">
<th align="center"><b>SL NO.</b></th>
<th align="center"><b>NAME</b></th>
<th align="center"><b>EMAIL</b></th>
<th align="center"><b>ADDRESS</b></th>
<th align="center"><b>DELETE</b></th>
<th align="center"><b>EDIT</b></th>
</tr>
<?php $no=1;
foreach($result as $res)
{
?>
<tr style="height:30px">
<td align="center"><?php echo $no;?>
<td><?php echo $res['name'];?></td>
```

```

<td><?php echo $res['email'];?></td>
<td><?php echo $res['address'];?></td>
<td align="center"><a href=<?php echo "delete.php?id=$res[_id]";?>" onClick="return
confirm('Are you sure you want to delete?')">DELETE</a></td>
<td align="center"><a href=<?php echo "edit.php?id=$res[_id]";?>">EDIT</a></td>
<?php
$no++;
}
?>
</tr>
</table>
</html>

```

```

edit.php <?php include
'connection.php'; if
(isset($_GET['id'])) {
    $res = $collection->findOne(['_id' => new MongoDB\BSON\ObjectID($_GET['id'])]);
} if(isset($_POST['update'])){
    $collection->updateOne(
        ['_id' => new MongoDB\BSON\ObjectID($_GET['id'])],
        ['$set' => ['name' => $_POST['name'], 'email' => $_POST['email'], 'address'=>
$_POST['address']]]
    );
    echo "Data Updated !!!";
    header("Location: view.php");
}
?>
<title>Registration Form</title>
<center>
<h1>Registration Form</h1>
<h3><u>Update User Details</u></h3>
<br>
</center>
<center>
<table class="table2">
<tr>
<td>
<head>
<h2 style="color:red;" align="center">REGISTRATION FORM</h2>
<style>
.table1 {
    border-style:solid;

```

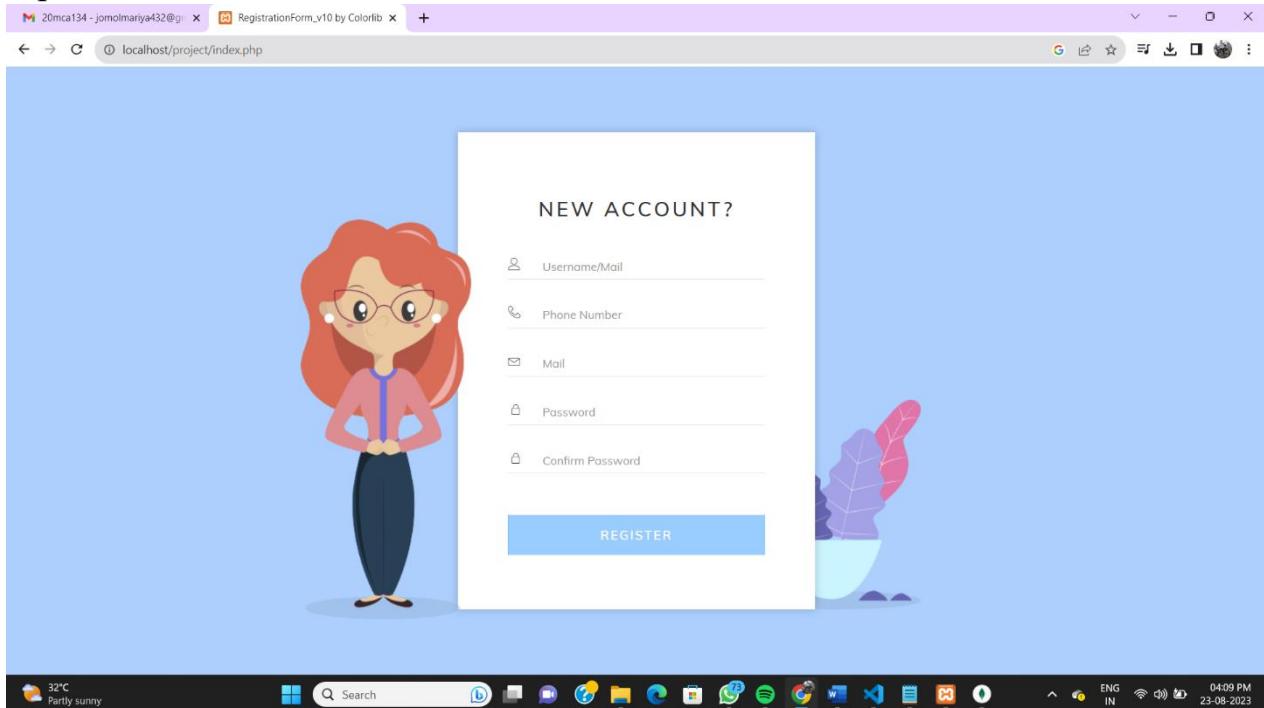
```
border-width:3px;
font-size: 20px;
border-color:black;
margin: 10px;
padding-top: 15px;
padding-bottom: 15px;
padding-left: 15px;
padding-right: 15px;
}

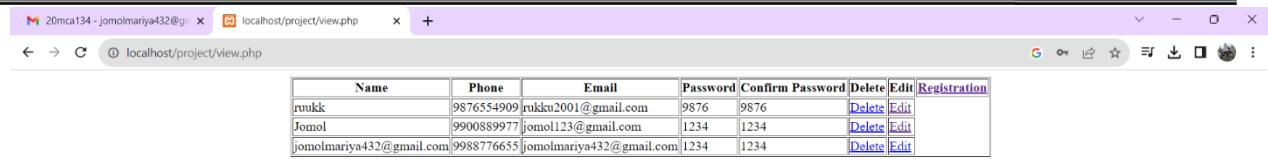
.table2 {
    padding-top: 150px;
}
</style> </head>
<form method="POST">
<table style="background-color:lightskyblue;" class="table1">
<tr>
<td> NAME : </td>
<td> <input type="text" name="name" value="<?php echo $res->name;?>"> </td>
</tr>
<tr>
<td> EMAIL : </td>
<td> <input type="text" name="email" value="<?php echo $res->email;?>"> </td>
</tr>
<tr>
<td> ADDRESS : </td>
<td> <textarea name="address" rows="5" cols="21"><?php echo $res-
>address;?></textarea></td>
</tr>
<tr>
<td align="right" colspan="2"> <input type="submit" name="update" value="Update"> </td>
</tr>
</table> </form>
</td>
</tr>
</table>
```

```
</center>
</body>
```

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

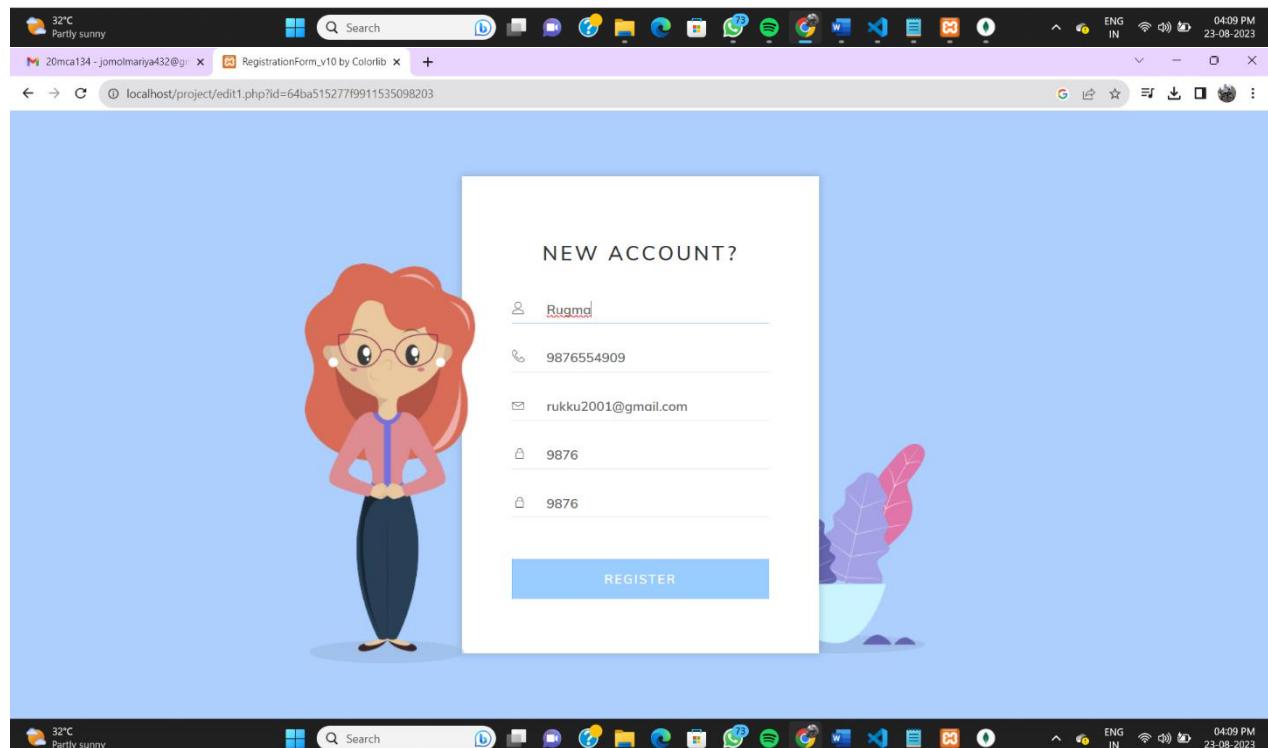
Output





A screenshot of a web browser window titled "localhost/project/view.php". The page displays a table with columns: Name, Phone, Email, Password, Confirm Password, Delete, Edit, and Registration. There are four rows of data:

Name	Phone	Email	Password	Confirm Password	Delete	Edit	Registration
rukku	987654909	rukku2001@gmail.com	9876	9876	Delete	Edit	
Jomol	9900889977	jomol123@gmail.com	1234	1234	Delete	Edit	
jomolmariya432@gmail.com	9988776655	jomolmariya432@gmail.com	1234	1234	Delete	Edit	



A screenshot of a web browser window titled "RegistrationForm_v10 by Colorlib". The page features a cartoon character on the left and a registration form on the right. The form has fields for Name, Phone, Email, and Password, each with a placeholder value. A "REGISTER" button is at the bottom.

NEW ACCOUNT?

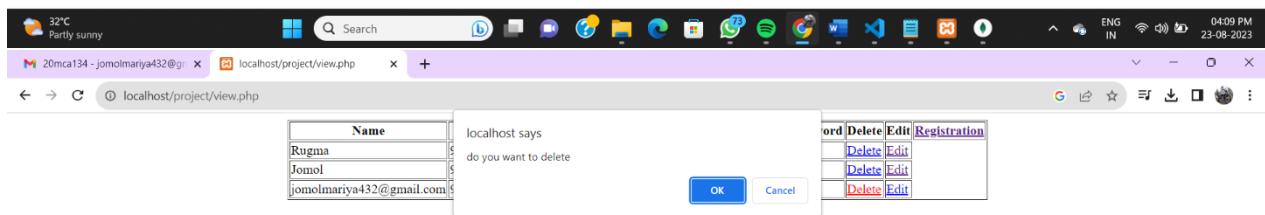
User Input Fields (Placeholder Values):

- Name: Rugmd
- Phone: 9876554909
- Email: rukku2001@gmail.com
- Password: 9876
- Confirm Password: 9876

REGISTER

A screenshot of a web browser window titled "localhost/project/view.php". The page displays a table with columns: Name, Phone, Email, Password, Confirm Password, Delete, Edit, and Registration. There are four rows of data:

Name	Phone	Email	Password	Confirm Password	Delete	Edit	Registration
Rugma	9876554909	rukku2001@gmail.com	9876	9876	Delete	Edit	
Jomol	9900889977	jomol123@gmail.com	1234	1234	Delete	Edit	
jomolmariya432@gmail.com	9988776655	jomolmariya432@gmail.com	1234	1234	Delete	Edit	



Result

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

Experiment No: 13Aim

Build sample collections/documents to perform the shell queries

CO5

Apply CRUD operations and retrieve data in a NoSQL environment.

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": "schneiderova", "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": "schneiderova", "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: 14Aim

To familiarize with indexing in MongoDB

CO5

Apply CRUD operations and retrieve data in a NoSQL environment.

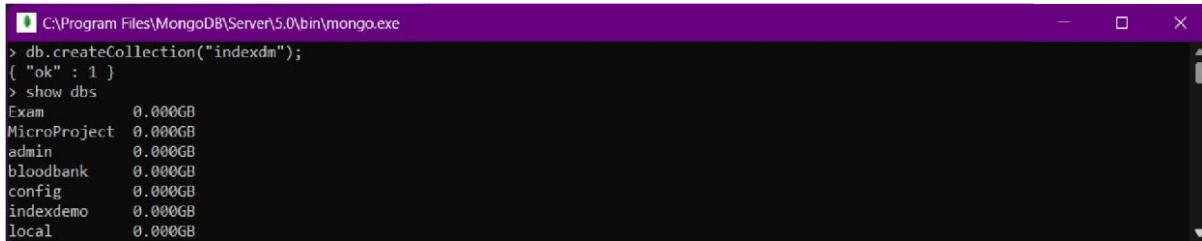
Procedure

1. Create and Use a Database



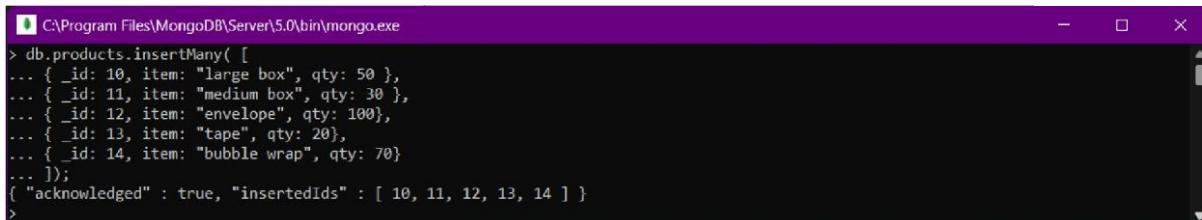
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> use indexdemo;
switched to db indexdemo
> -
```

2. Show Databases



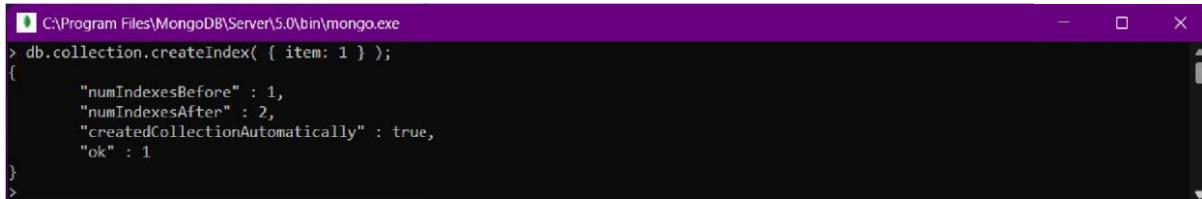
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.createCollection("indexdm");
{ "ok" : 1 }
> show dbs
Exam          0.000GB
MicroProject  0.000GB
admin         0.000GB
bloodbank     0.000GB
config        0.000GB
indexdemo     0.000GB
local         0.000GB
```

3. Input data



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> 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 ] }
```

4. Create ascending index on a field



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.collection.createIndex( { item: 1 } );
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : true,
  "ok" : 1
}
```

5. Create descending index on a field

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.collection.createIndex( { qty: -1 } );
{
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
>
```

6. Create index with the index name

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.products.createIndex(
... { item: 1, quantity: -1 } ,
... { name: "query for inventory" }
... );
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
>
```

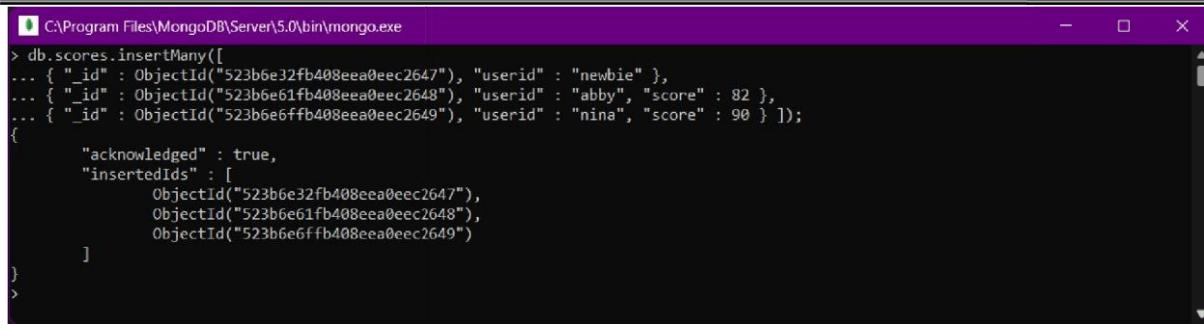
7. To list out indexes on the <collection>,

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.products.getIndexes();
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "item" : 1,
      "quantity" : -1
    },
    "name" : "query for inventory"
  }
]
>
```

8. Drop index by index document

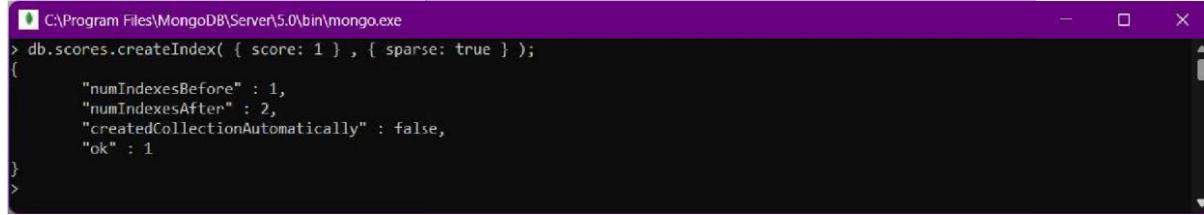
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.products.dropIndex(
... { item: 1, quantity: -1 }
... );
{
  "nIndexesWas" : 2, "ok" : 1
}
>
```

9. Insert data



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.scores.insertMany([
... { "_id" : ObjectId("523b6e32fb408eea0eec2647"), "userid" : "newbie" },
... { "_id" : ObjectId("523b6e61fb408eea0eec2648"), "userid" : "abby", "score" : 82 },
... { "_id" : ObjectId("523b6e6ffb408eea0eec2649"), "userid" : "nina", "score" : 90 } ]);
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("523b6e32fb408eea0eec2647"),
        ObjectId("523b6e61fb408eea0eec2648"),
        ObjectId("523b6e6ffb408eea0eec2649")
    ]
}
>
```

10. Create a sparse index



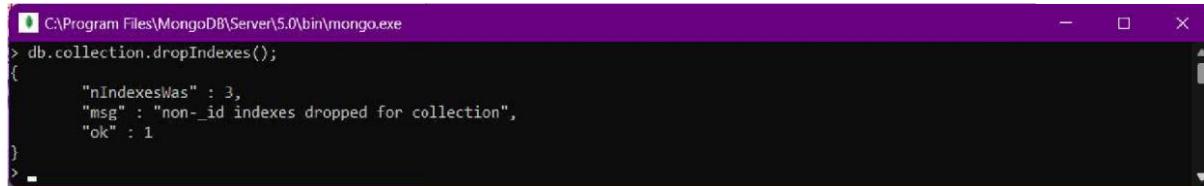
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.scores.createIndex( { score: 1 } , { sparse: true } );
{
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "createdCollectionAutomatically" : false,
    "ok" : 1
}
>
```

11. Use Sparse index on the score field



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.scores.find( { score: { $lt: 90 } } );
{ "_id" : ObjectId("523b6e61fb408eea0eec2648"), "userid" : "abby", "score" : 82 }
> -
```

12. Drop all indexes but _id index



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.collection.dropIndexes();
{
    "nIndexesWas" : 3,
    "msg" : "non-_id indexes dropped for collection",
    "ok" : 1
}
> -
```

Result

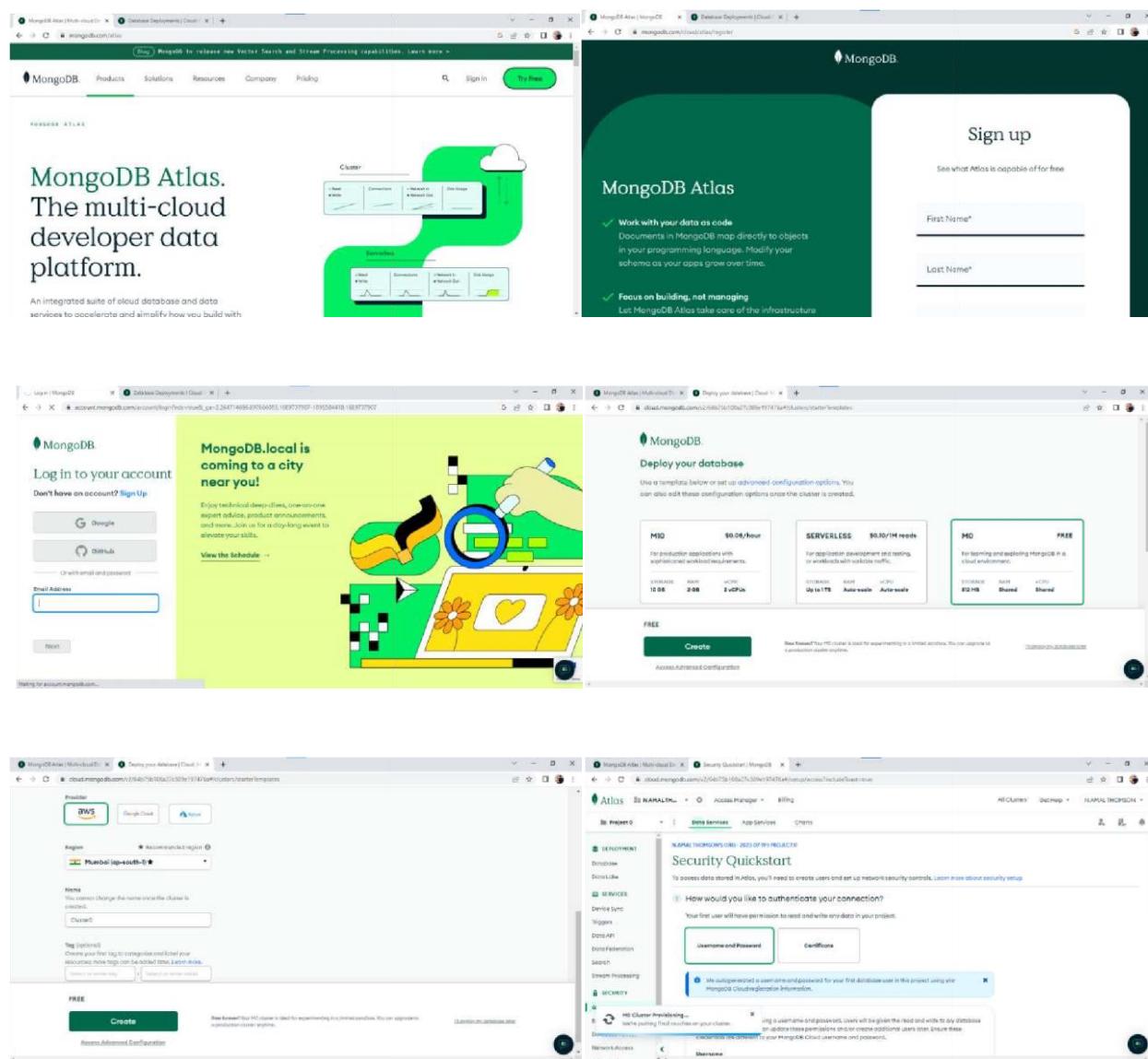
The program was executed and the result was successfully obtained. Thus CO5 was obtained

Experiment No: 15Aim

Usage of Cloud Storage Management Systems: MongoDB Atlas

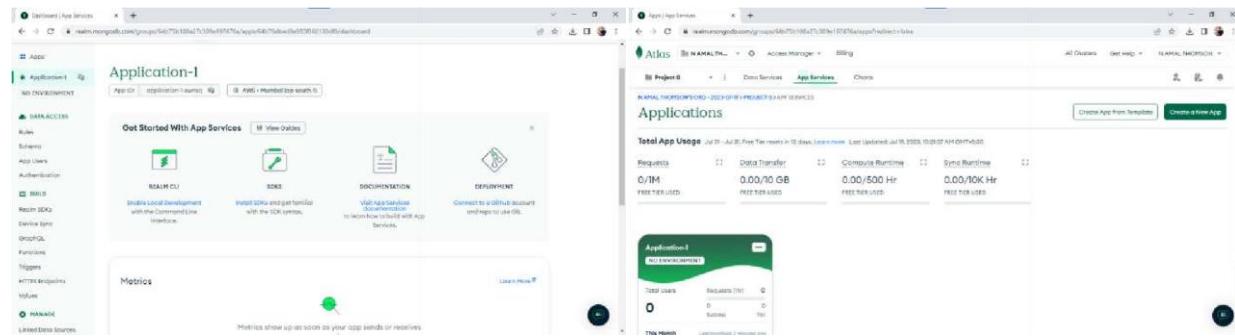
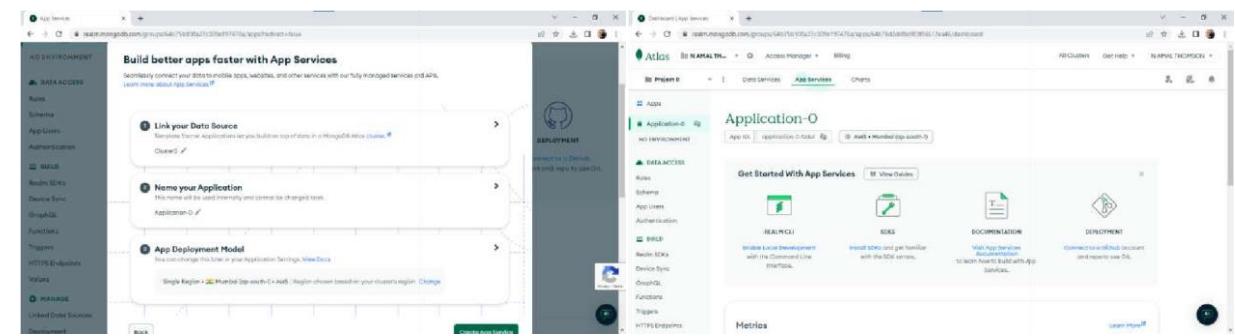
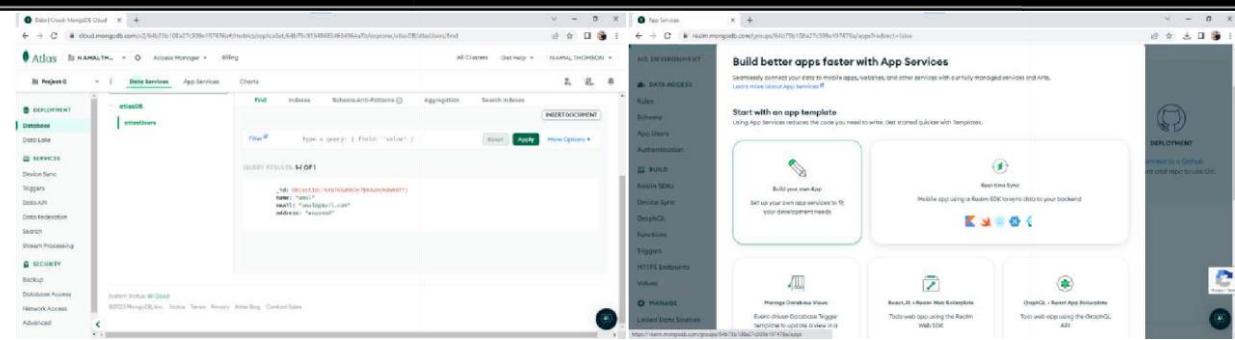
CO5

Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure

The screenshots illustrate the setup of a MongoDB cluster in the 'Project 0' environment:

- Creating a Database User:** Shows the creation of a user 'nammal' with password '123456'.
- Connecting from Local Environment:** Shows connection details for the local environment.
- Connecting from Cloud Environment:** Shows connection details for the cloud environment.
- Setting IP Access Rules:** Shows adding an IP address '192.168.1.12' to the IP Access List.
- Congratulations on setting up access rules!**: Confirmation message after adding the IP rule.
- Database Deployments:** Overview of database deployments across regions (US West, EU West, Asia Pacific) and clusters (Cluster0).
- Connect to Cluster0:** Step-by-step guide to connect to Cluster0, including setting security and choosing a connection method.
- Connect to your application:** Tools for connecting to the application, including Grails, Node.js, Python, Java, and .NET.
- Access your data through tools:** Options for Compass (MongoDB UI), Shell (MongoDB shell), and MongoDB ODBC Driver.



Result

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

Experiment No: 16Aim

Implementation of Replica Set on MongoDB

CO5

Apply CRUD operations and retrieve data in a NoSQL environment.

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 -replicaSet datascience -logpath \data\rs1\1.log -dbpath \data\rs1 -port 27018
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replicaSet datascience -logpath \data\rs2\2.log -dbpath \data\rs2 -port 27019
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replicaSet 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,
  "writableVotingMembersCount" : 3,
  "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
datascience:PRIMARY> rs.list()
{
  "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" : ""
    }
  ]
}

Select C:\Windows\System32\cmd.exe - mongo --port 27018
datascience:SECONDARY> rs.list()
{
  {
    "_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
  }
}
```

```

C:\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
}

```



```

C:\Select C:\Windows\System32\cmd.exe - mongo --port 27018
],
"ok" : 1,
"$clusterTime" : {
    "clusterTime" : Timestamp(1690344920, 2),
    "signature" : {
        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
        "keyId" : NumberLong(0)
    }
},
"operationTime" : Timestamp(1690344920, 2)
}

```

5. Performing operations at Primary

```

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("64c09e5576967f07607f0cff"), "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):i:1
> use admin
switched to db admin

```

6. Performing operations at Secondary

```

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-bf838f2f77a1") }
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 <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()
...
datascience:SECONDARY> show dbs
uncaught exception: Error: listDatabases failed:
  "topologyVersion" : {
    "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
    "counter" : NumberLong(4)
  },

```

```
C:\Windows\System32\cmd.exe - mongo --port 27019
},
"ok" : 0,
"errmsg" : "not master and slaveOk=false",
"code" : 13435,
"codeName" : "NotPrimaryNoSecondaryOk",
"$clusterTime" : {
    "clusterTime" : Timestamp(1690345198, 1),
    "signature" : {
        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
        "keyId" : NumberLong(0)
    }
},
"operationTime" : Timestamp(1690345198, 1)
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs<@src/mongo/shell/mongo.js:145:19
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:97:12
shellHelper.show@src/mongo/shell/utils.js:956:13
shellHelper@src/mongo/shell/utils.js:838:15
@shellhelp2):1:
dataScience:SECONDARY> rs.slaveOk()
WARNING: slaveOk() is deprecated and may be removed in the next major release. Please use secondaryOk() instead.
dataScience:SECONDARY> rs.secondaryOk()
dataScience:SECONDARY> show dbs
admin 0.000GB
config 0.000GB
dataScience 0.000GB
local 0.000GB
dataScience:SECONDARY> use dataScience
switched to db dataScience
dataScience:SECONDARY> db.student.insert({name:"Vimal"})
WriteCommandError({
    "topologyVersion" : {
        "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
        "counter" : NumberLong(4)
    },
    "ok" : 0,
},
"ok" : 0,
"errmsg" : "not master",
"code" : 10107,
"codeName" : "NotWritablePrimary",
"$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("64c0a1023151b8030c0e8860"), "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: mongodb://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 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()
=====
dataScience:SECONDARY> rs.secondaryOk()
dataScience:SECONDARY> show dbs
admin 0.000GB
config 0.000GB
dataScience 0.000GB
local 0.000GB
dataScience:SECONDARY> use dataScience
```



```
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" : 1010,
  "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("64c09e1023151b8630c0e8860"), "name" : "skhil" }
datascience:SECONDARY> -
```

Result

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