

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 fundamental tasks involved with modeling, designing and implementing a database.

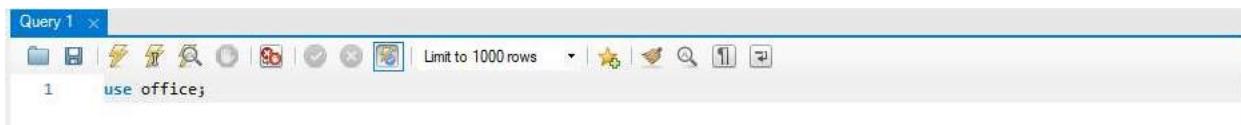
Procedure:

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

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

Query 1

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

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

Tables_in_office
emp_details

9. TRUNCATE TABLE

Query 1

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

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

eno	name	ehiredate	esalary	ecomission
*	NULL	NULL	NULL	NULL

10. DROP TABLE

Query 1

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

8 12:16:37 drop table emp_details 0 row(s) affected

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



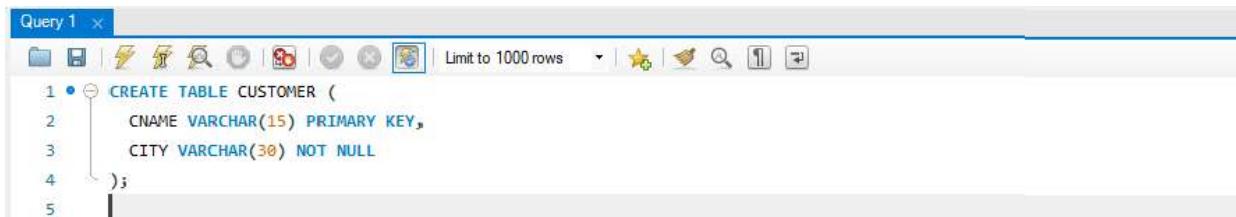
```
Query 1
use bank;
```

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 ×
INSERT INTO BRANCH (BNAME, CITY) VALUES ('VRCE', 'NAGPUR');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('AJNI', 'NAGPUR');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('KAROLBAGH', 'DELHI');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('CHANDNI', 'DELHI');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('DHARAMPETH', 'NAGPUR');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('MG ROAD', 'BANGLORE');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('ANDHERI', 'BOMBAY');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('NEHRU PALACE', 'DELHI');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('POWAI', 'BOMBAY');
  
```

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 compete ce with the fundamentals 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 "Result Grid" displays the following data:

ACTNO	CNAME	BNAME	AMOUNT	ADATE
D100	ANIL	VRCE	1000.00	1995-03-01
D101	SUNIL	ANJANI	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

2. List all data from borrow

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is: "SELECT * FROM borrow;". Below the query, the "Result Grid" displays the following data:

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

3. List all data from customer

Query 1

```
1   SELECT * FROM customers;
```

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

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

4. List all data from branch

Query 1

```
1   SELECT * FROM branch;
```

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

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

5. Give account no and amount of deposit

Query 1

```
1   SELECT actno, amount FROM deposit;
```

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

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	D 100
SUNIL	D 101
MEHUL	D 102
MADHURI	D 104
PRAMOD	D 105
SANDIP	D 106
SHIVANI	D 107
KRANTI	D 108
MINU	D 109
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;
2  
```

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	
*	HULL

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 = 'l205';
```

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

CNAME

15. Give names of depositors having account at VRCE

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

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

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

CNAME
ANIL

16. Give names of all branched located in city Delhi

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

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

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

BNAME
CHANDNI
KAROLBAGH
NEHRU PALACE
NULL

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

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

```
1  SELECT CNAME FROM DEPOSIT WHERE ADATE = '1996-12-01';  
2
```

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

CNAME

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';
```

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

ACTNO	AMOUNT
HULL	HULL

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

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

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

CITY
DELHI

20. Give details of customer ANIL

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

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

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

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 and a results grid. The query is:

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

The results 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 and a results grid. The query is:

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

The results 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

Query 1

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

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

CNAME
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 | Filter Rows: Export: Wrap Cell Content:

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 | Filter Rows: Export: Wrap Cell Content:

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

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query selects the city from the DEPOSIT and BRANCH tables where the branch name matches and the customer name is either 'SUNIL' or 'ANIL'. The result grid shows two rows: 'CITY' and 'NAGPUR'.

CITY
NAGPUR

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

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

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query selects distinct customer names from the deposit and borrow tables where the deposit amount is greater than 1000 and the loan amount is less than 10000. The result grid shows six customer names: MEHUL, MADHURI, PRAMOD, SANDIP, KRANTI, and MINU.

CNAME
MEHUL
MADHURI
PRAMOD
SANDIP
KRANTI
MINU

9. List the cities of depositors having branch VRCE.

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

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query selects the city from the deposit and branch tables where the branch name is 'VRCE'. The result grid shows one row: 'CITY' and 'NAGPUR'.

CITY
NAGPUR

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

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

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query selects the name of customers from the "deposit" table (alias D1) who have an amount less than 1000 and live in the same city as "ANIL" (from the "customer" table, alias C1). The result grid shows one row: "SUNIL".

CNAME
SUNIL

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

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

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query selects the city from the "branch" table (alias B1) where the branch name is in the list of branch names corresponding to "ANIL" and "SUNIL" from the "deposit" table (alias D1). The result grid shows one row: "NAGPUR".

CITY
NAGPUR

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

```
Query 1 ×
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 screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query uses a subquery to find the city where "ANIL" lives and then lists the distinct name, amount, and city for depositors in that city. The result grid shows one row: "ANIL" with amount "1000.00" and city "CALCUTTA".

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

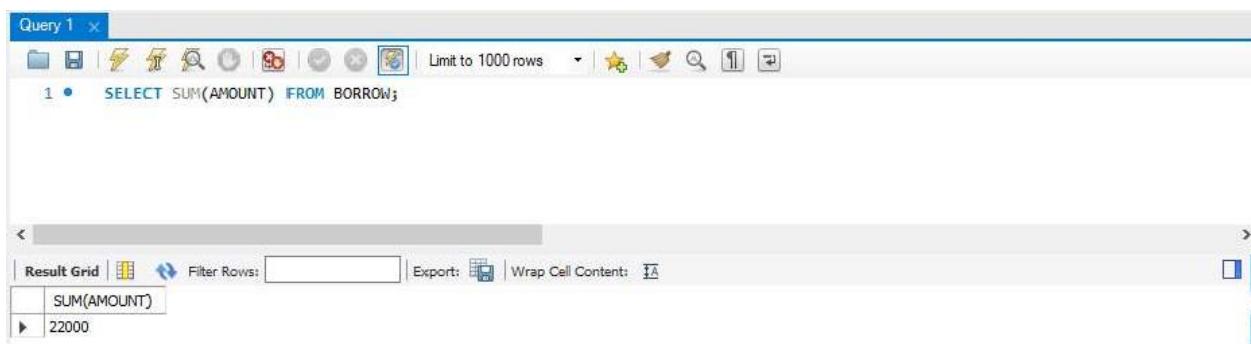
To familiarize with Aggregate Functions.

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

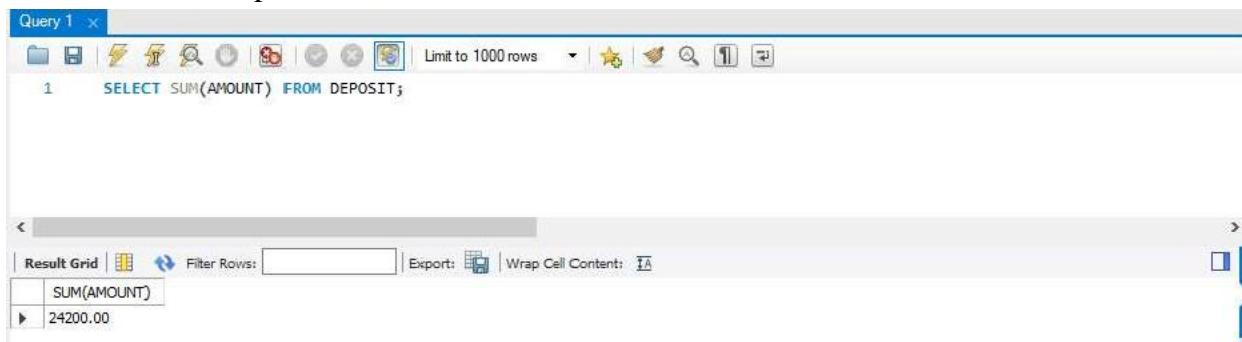


```
Query 1 ×
SELECT SUM(AMOUNT) FROM BORROW;
```

The screenshot shows the MySQL Workbench interface with a single query window titled "Query 1". The query is "SELECT SUM(AMOUNT) FROM BORROW;". The results are displayed in a grid with one row, showing the sum of amounts as 22000.

SUM(AMOUNT)
22000

2. List total deposit

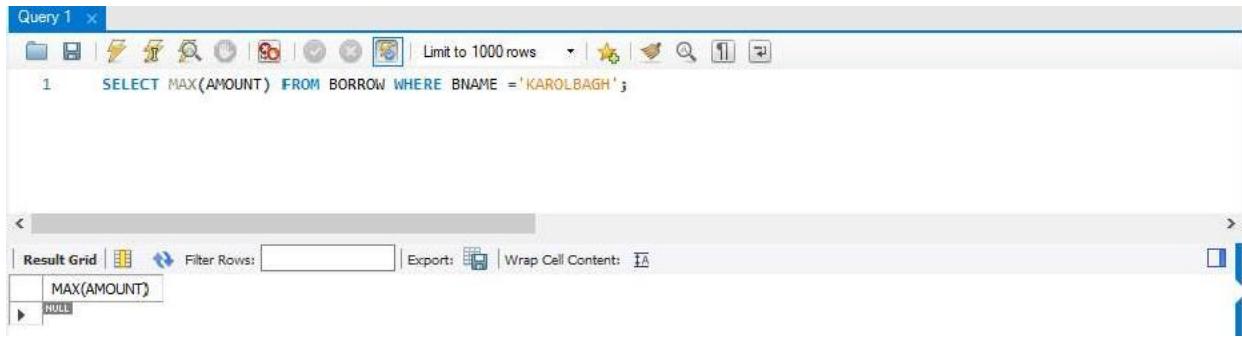


```
Query 1 ×
SELECT SUM(AMOUNT) FROM DEPOSIT;
```

The screenshot shows the MySQL Workbench interface with a single query window titled "Query 1". The query is "SELECT SUM(AMOUNT) FROM DEPOSIT;". The results are displayed in a grid with one row, showing the sum of amounts as 24200.00.

SUM(AMOUNT)
24200.00

3. List total loan taken from KAROLBAGH branch



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

The screenshot shows the MySQL Workbench interface with a single query window titled "Query 1". The query is "SELECT MAX(AMOUNT) FROM BORROW WHERE BNAME = 'KAROLBAGH';". The results are displayed in a grid with one row, showing the maximum amount as NULL.

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

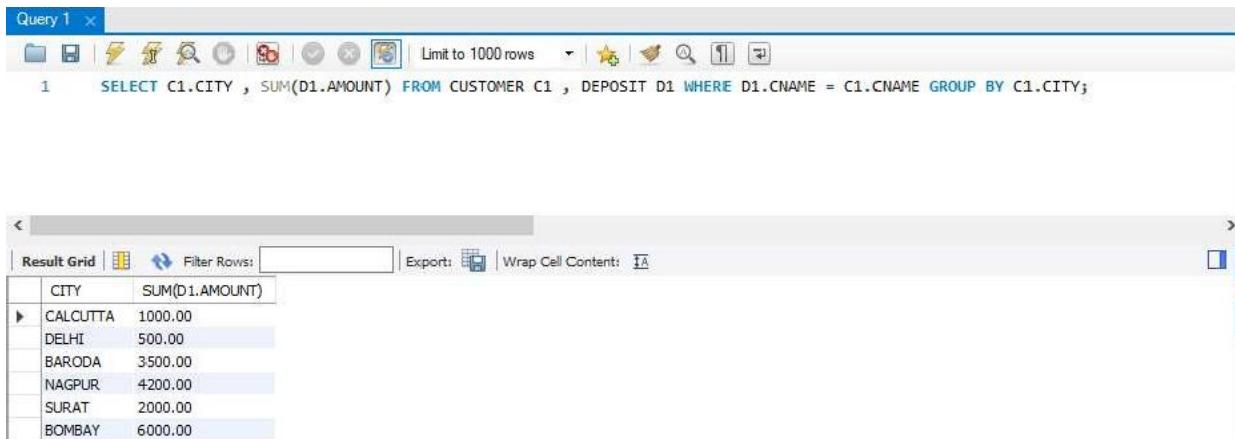
Query 1 ×

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

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

BNAME	SUM(AMOUNT)
VRCE	1000.00
ANJNİ	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

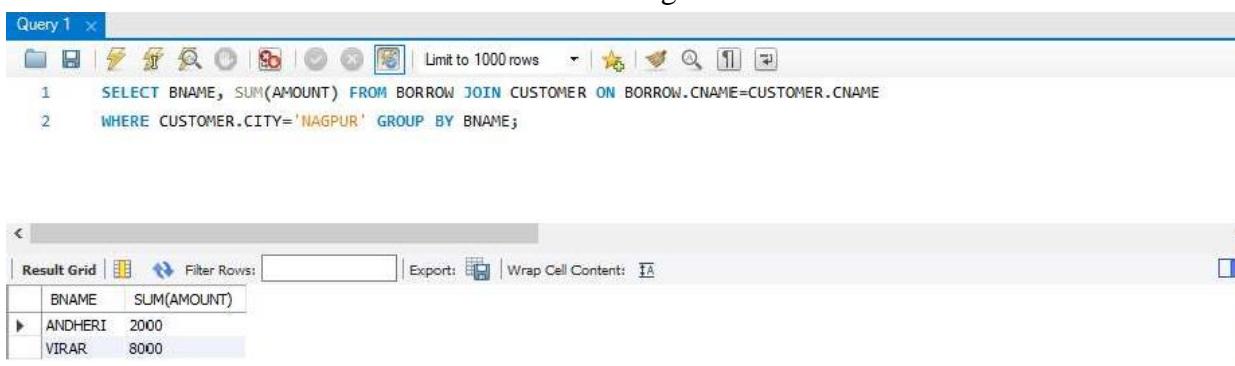


```
Query 1
1  SELECT C1.CITY , SUM(D1.AMOUNT) FROM CUSTOMER C1 , DEPOSIT D1 WHERE D1.CNAME = C1.CNAME GROUP BY C1.CITY;
```

Result Grid

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

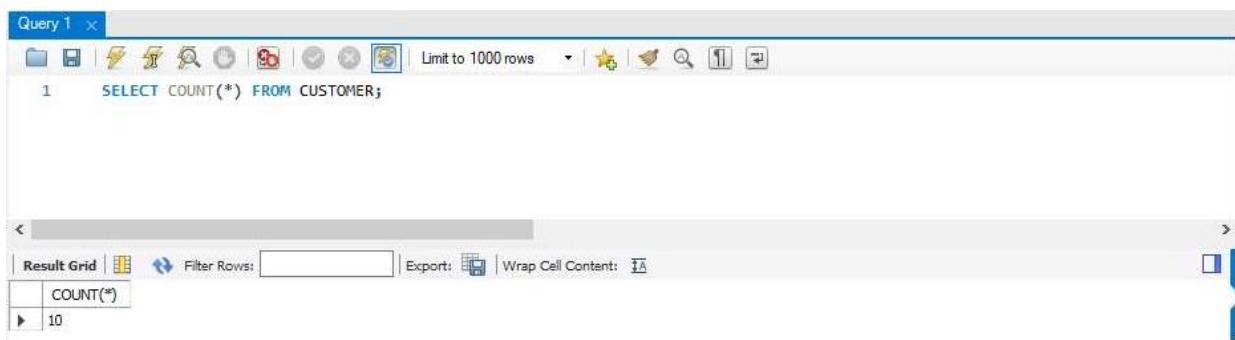


```
Query 1
1  SELECT BNAME, SUM(AMOUNT) FROM BORROW JOIN CUSTOMER ON BORROW.CNAME=CUSTOMER.CNAME
2  WHERE CUSTOMER.CITY='NAGPUR' GROUP BY BNAME;
```

Result Grid

BNAME	SUM(AMOUNT)
ANDHERI	2000
VIRAR	8000

13. Count total number of customers



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

Result Grid

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 | Filter Rows: Export: Wrap Cell Content:

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 | Filter Rows: Export: Wrap Cell Content:

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 | Filter Rows: Export: Wrap Cell Content:

count(customer.CNAME)
6

Result

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

Experiment no: 6

Aim

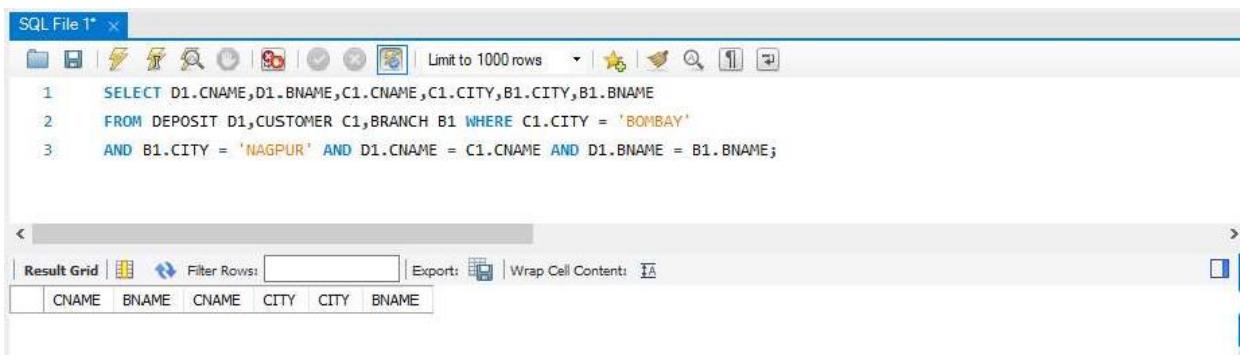
To familiarize with Join or Cartesian Product

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

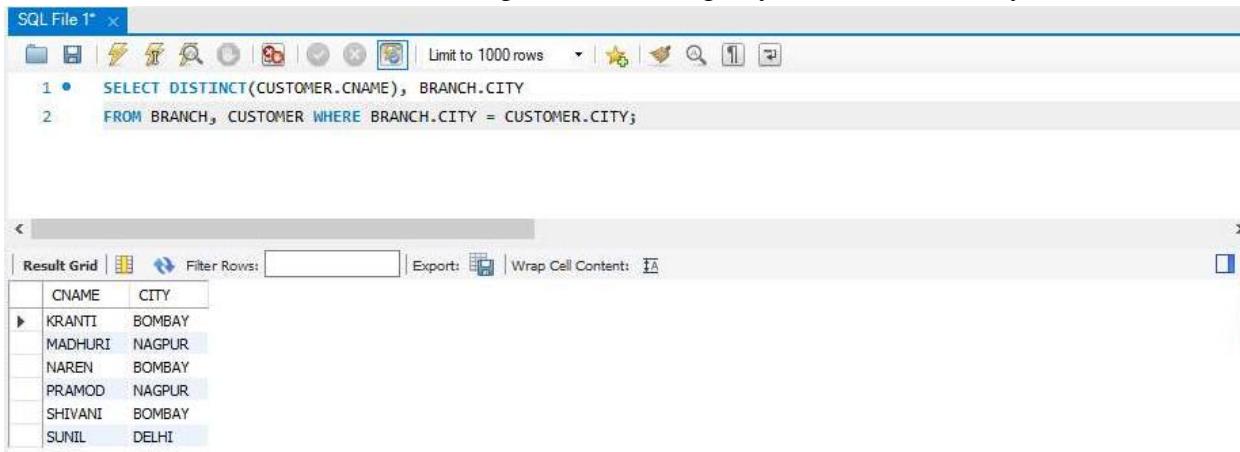


```
SQL File 1* ×
SELECT D1.CNAME,D1.BNAME,C1.CNAME,C1.CITY,B1.CITY,B1.BNAME
FROM DEPOSIT D1,CUSTOMER C1,BRANCH B1 WHERE C1.CITY = 'BOMBAY'
AND B1.CITY = 'NAGPUR' AND D1.CNAME = C1.CNAME AND D1.BNAME = B1.BNAME;
```

The screenshot shows the SQL query above in the query editor. Below it, the results are displayed in a grid:

CNAME	BNAME	CNAME	CITY	CITY	BNAME

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



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

The screenshot shows the SQL query above in the query editor. Below it, the results are displayed in a grid:

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* x
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* x
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* x
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

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

```
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 result grid shows two columns: CNAME and AMOUNT. There are no results displayed.

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

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

```
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 result grid shows two columns: CNAME and AMOUNT. There are no results displayed.

8. Give branch city and living city of Pramod

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

```
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 result grid shows two columns: CNAME and AMOUNT. There are no results displayed.

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

The screenshot shows the SQL File 1 window in SSMS. 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 shows one row:

CITY
NAGPUR

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

The screenshot shows the SQL File 1 window in SSMS. The query is:

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

The result grid shows three rows:

CNAME	CITY
ANIL	CALCUTTA
SUNIL	DELHI
*	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 fundamentals 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 the SQL Editor window with the following query:

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

The results pane shows a single row:

BNAME
POWAI

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 the SQL Editor window with the following query:

```
SELECT D.BNAME FROM DEPOSIT D, BRANCH B WHERE
D.BNAME=B.BNAME GROUP BY D.BNAME HAVING SUM(D.AMOUNT)>5000;
```

The results pane shows a single 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 the SQL Editor window with the following query:

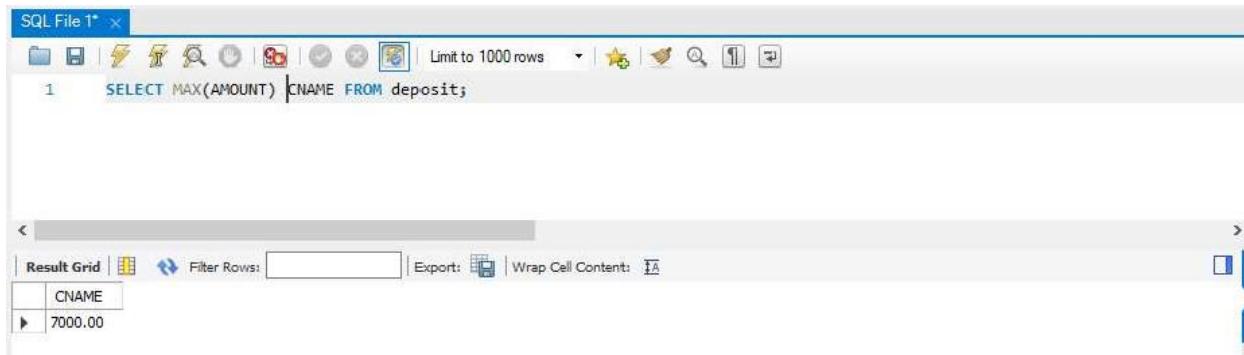
```
SELECT CNAME from deposit where AMOUNT=(select AVG(Amount) from DEPOSIT)
```

A tooltip "GR Execute the selected portion of the script or everything, if there is no selection" is visible over the second line of the query.

The results pane shows a single row:

CNAME
MINU

4. List the names of customers having maximum deposit



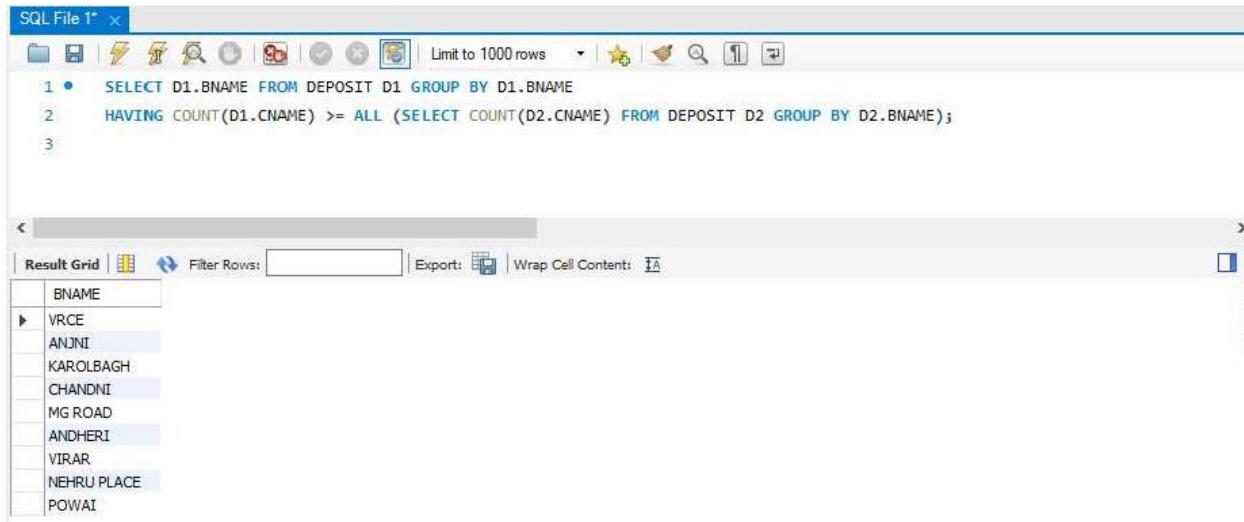
SQL File 1* x

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

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

CNAME
7000.00

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



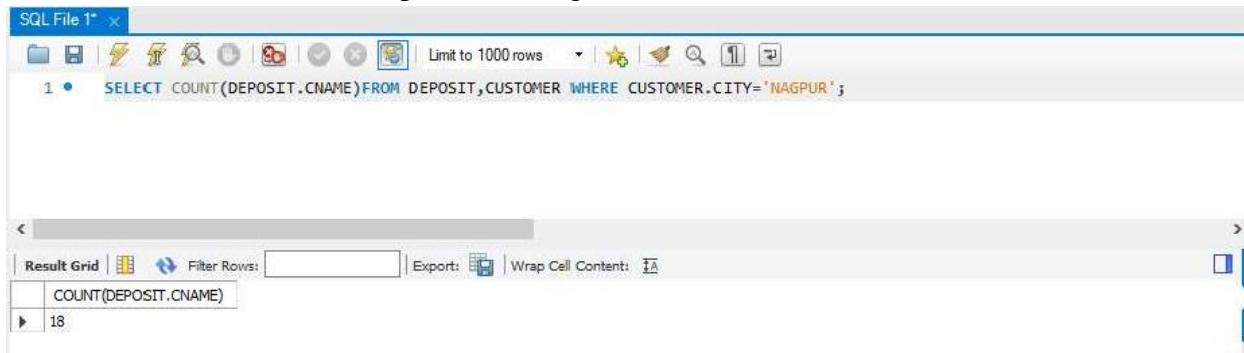
SQL File 1* x

```
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
```

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

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

6. Count the number of depositors living in NAGPUR.



SQL File 1* x

```
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
```

The result grid shows one row:

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
```

The result grid shows one row:

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.CNAME GROUP BY C.CNAME ORDER BY COUNT(B.BNAME) DESC;
3
```

The result grid shows one row:

CNAME	COUNT(B.BNAME)

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

```
SQL File 1* 
1 • SELECT COUNT(B1.BNAME) FROM DEPOSIT D1 JOIN BORROW_B1_CUSTOMER C1 WHERE
2 C1.CITY=D1.CITY AND D1.BNAME=B1.BNAME AND C1.CITY IN (SELECT CITY FROM CUSTOMER);
3
```

The result grid shows one row:

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

```

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;

```

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The code listed creates a database named "employee_db", selects it, creates a table "employee" with columns for emp_id, emp_name, dob, address, designation, mobile_no, dept_no, and salary, and finally selects all rows from the employee table. Below the code is a result grid showing one row with all columns set to NULL.

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

```

Query 1 x
CREATE TABLE `salary` (`employee_id` int NOT NULL, `old_sal` int DEFAULT NULL, `new_sal` int DEFAULT NULL,
`rev_date` date DEFAULT NULL, PRIMARY KEY (`employee_id`));
SELECT * FROM salary;

```

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The code listed creates a table "salary" with columns for employee_id, old_sal, new_sal, and rev_date, and a primary key on employee_id. Below the code is a result grid showing one row with all columns set to NULL.

employee_id	old_sal	new_sal	rev_date
*	HULL	HULL	HULL

The screenshot shows the MySQL Workbench interface. At the top, there is a SQL editor window titled "employee" containing the following code:

```

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

```

Below the SQL editor is a "Result Grid" showing the data inserted into the "employee" table:

	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
*								

On the left side of the interface, there is a tree view showing table definitions. The "employee" table is selected, and its details are shown in a panel on the right:

- Table Name: employee
- Schema: employee_db
- Charset/Collation: utf8mb4
- Engine: InnoDB
- Comments: (empty)

Below the table definition, the trigger definition for the "employee" table is displayed:

```

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

```

The screenshot shows the MySQL Workbench interface. At the top, there is a SQL editor window titled "salary" containing the following code:

```

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

```

Below the SQL editor is a "Result Grid" showing the data inserted into the "salary" table:

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

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. The top window is titled "SQL File 5*". It contains the following PL/SQL code:

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

The bottom window is titled "Result Grid". It displays the output of the stored procedure, which is a table with four columns: LOANNO, CNAME, BNAME, and AMOUNT. The data is as follows:

LOANNO	CNAME	BNAME	AMOUNT
L201	ANI	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

Result

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

Experiment No: 10

Aim

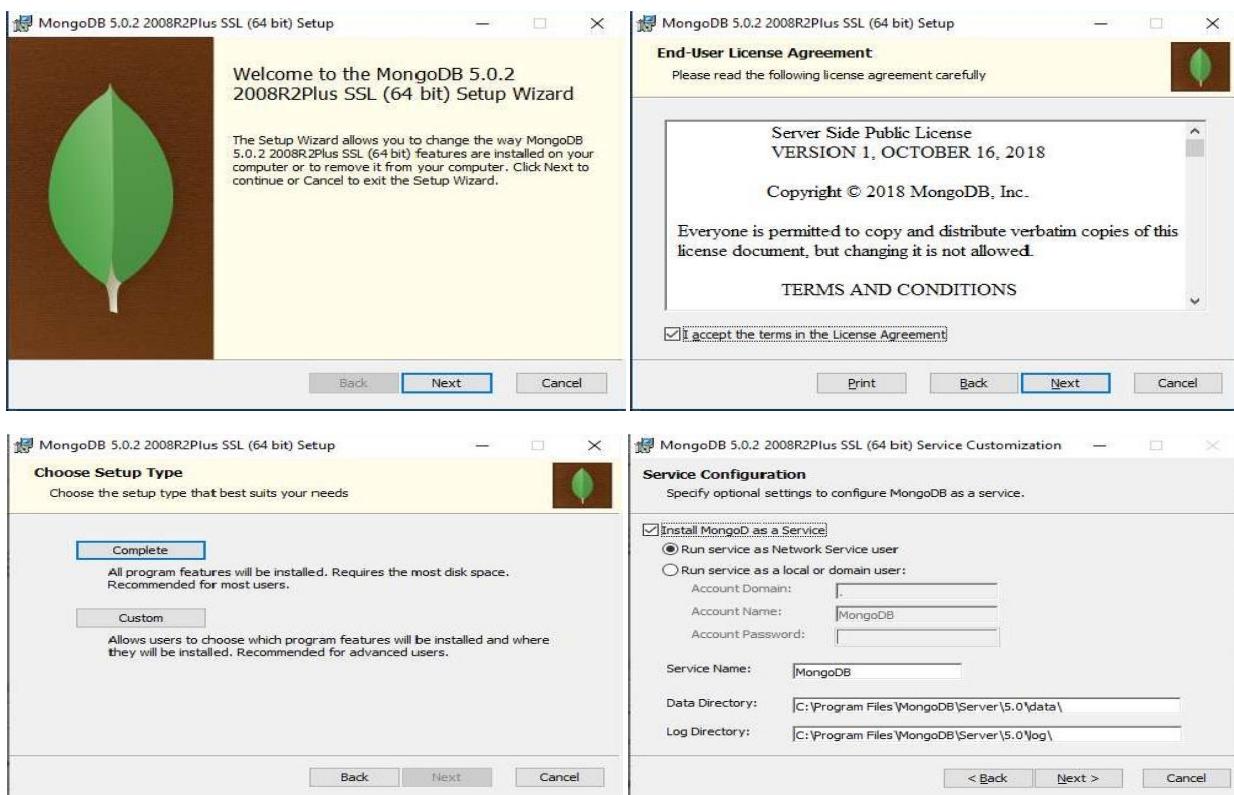
Understand the installation and configuration of NoSQL Databases: MongoDB

CO3

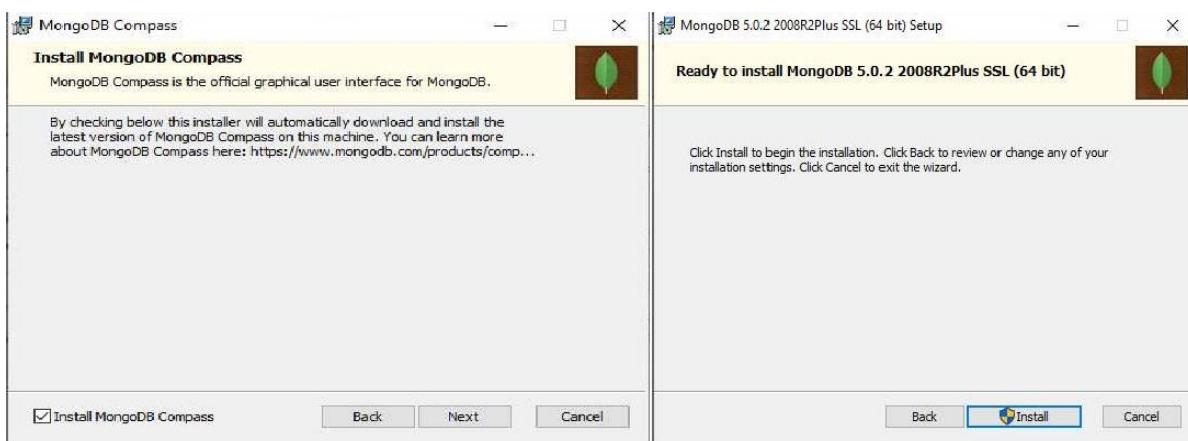
Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

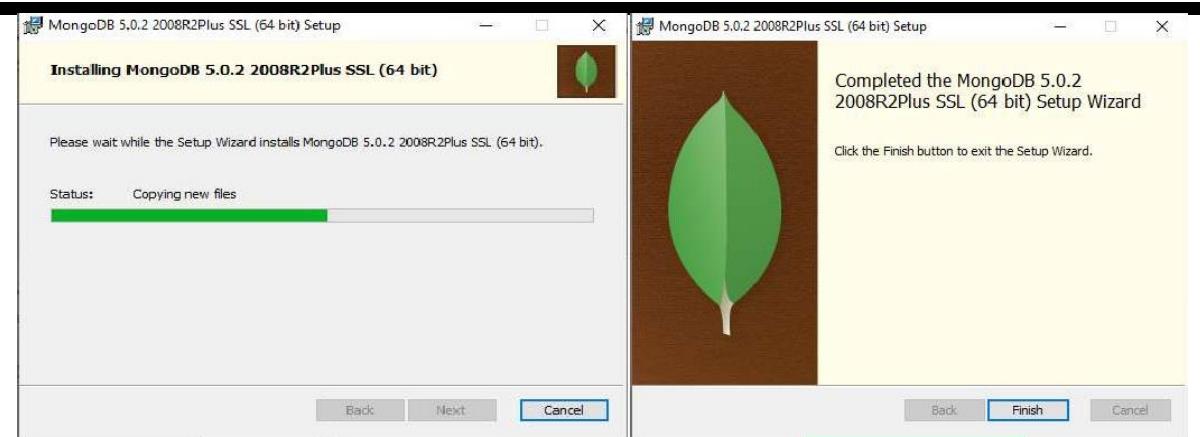
Procedure

1. Setup Wizard



2. Installing MongoDB Compass





3. MongoDB Compass Home Page

Welcome to Compass

Build aggregation pipelines, optimize queries, analyze schemas, and more. All with the GUI built by the team for MongoDB.

To help improve our products, anonymous usage data is collected and sent to MongoDB in accordance with MongoDB's privacy policy. Manage the behavior on the Compass Settings page.

New Connection

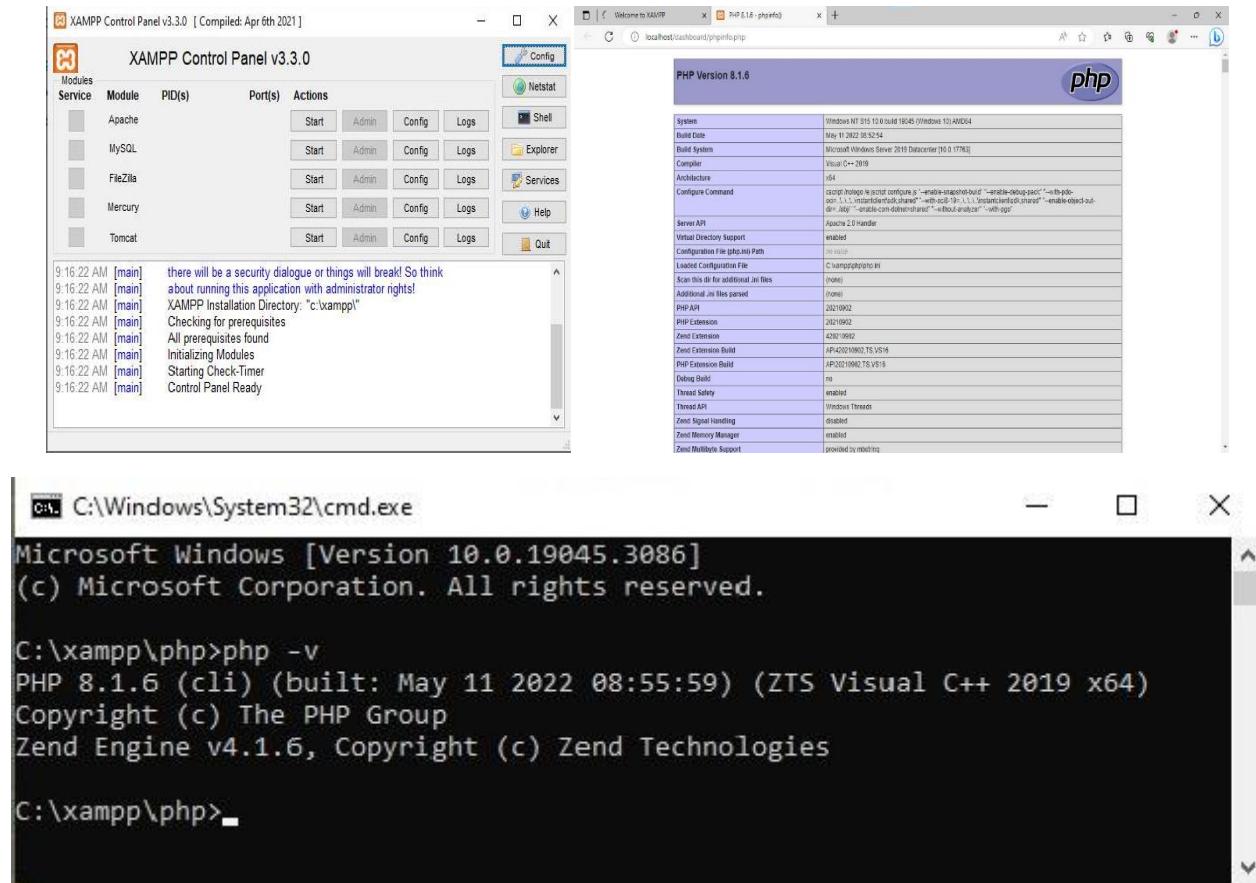
Connection MongoDB deployment

URL: Edit Connection String

Databases

- admin** Storage size: 20.48 kB Collections: 1 Indexes: 1
- config** Storage size: 24.56 kB Collections: 1 Indexes: 2
- local** Storage size: 36.86 kB Collections: 1 Indexes: 1

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 for Home, News, Support, and API documentation.
- Downloads:** Includes links for Browse Packages, Search Packages, and Download Statistics.
- Package Information:**
 - Maintainers:** Jeremy Mikola (lead), Katherine Walker (developer), Andreas Brauer (lead), Derick Rethans (lead), and Hannes Magnusson (btori) (lead).
 - License:** Apache License.
 - Description:** A brief description of the MongoDB driver's purpose and dependencies.
 - Homepage:** <http://docs.mongodb.org/ecosystem/drivers/php/>
- Available Releases:** A table listing the available MongoDB PHP driver versions, their states, release dates, file sizes, and download links.

Version	State	Release Date	Downloads
1.16.1	stable	2023-06-22	mongodbc-1.16.1.tgz (1862.9kB)
1.16.0	stable	2023-06-22	mongodbc-1.16.0.tgz (1521.9kB)
1.15.3	stable	2023-05-12	mongodbc-1.15.3.tgz (1701.9kB)
1.15.2	stable	2023-04-21	mongodbc-1.15.2.tgz (1702.1kB)
1.15.1	stable	2023-02-09	mongodbc-1.15.1.tar (1701.4kB)

6. Configuring Apache

Apache (httpd.conf)

Apache (httpd-ssl.conf)

Apache (httpd-xampp.conf)

PHP (php.ini)

phpMyAdmin (config.inc.php)

<Browse> [Apache]

<Browse> [PHP]

<Browse> [phpMyAdmin]

php - Notepad

```

File Edit Format View Help
;extension=mbstring
extension=zip
extension=xml
extension=xmlwriter
extension=xif ; Must be after mbstring as it depends on it
extension=sqlite
;extension=oci12c ; Use with Oracle Database 12c Instant Client
;extension=oci19 ; Use with Oracle Database 19 Instant Client
extension=odbc
extension=openssl
extension=firebird
extension=pgsql
extension=oci11
extension=odbc
extension=pdo_pgsql
extension=pdo_sqlite
extension=pdo_sql
extension=soap
extension=sockets
extension=sodium
extension=sqlite3
extension=tidy
extension=xsl
zend_extension=opcache

;-----[Module Settings]-----;
;-----[Module Settings]-----;
asp_tags=Off

```

Ln 94, Col 1 100% Windows (CR/LF) UTF-8

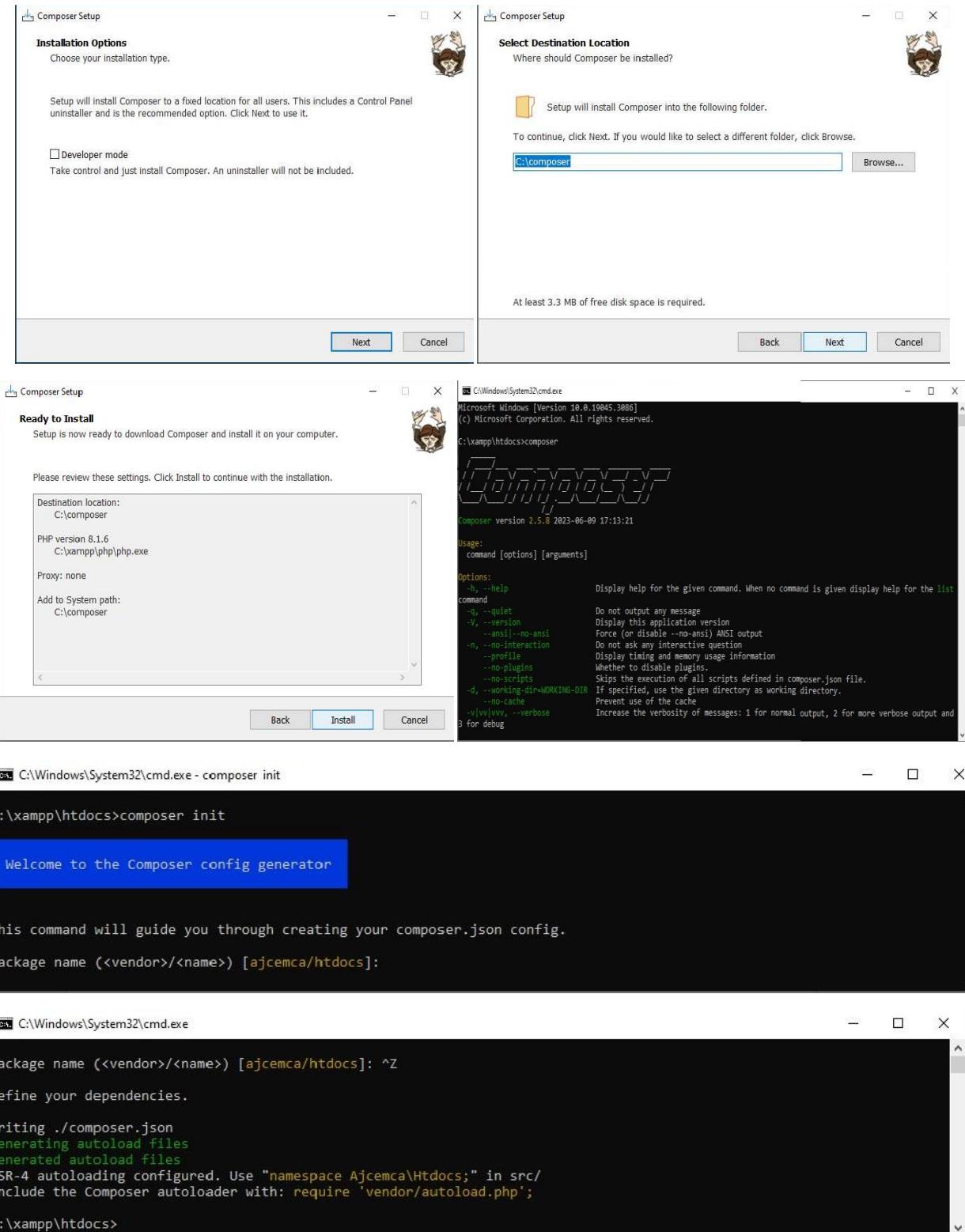
Directive	Local Value	Master Value
mbstring.regex_stack_limit	1000000	1000000
mbstring.strict_detection	Off	Off
mbstring.substitute_character	no value	no value

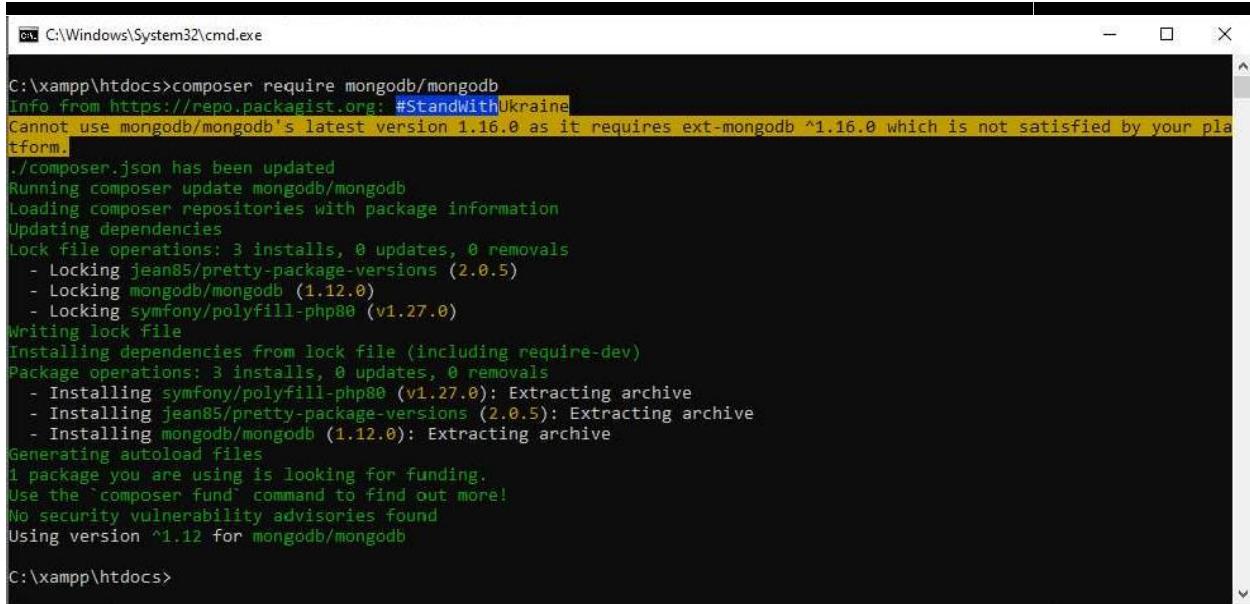
mongodb

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	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
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 -> 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 the MongoDB Compass interface connected to 'localhost:27017'. The left sidebar lists databases: My Queries, Databases, colors (selected), colors.VIBGYOR (highlighted in blue), config, and local. The main area displays the 'colors.VIBGYOR' collection with 7 documents and 1 index. A search bar at the top says 'Type a query: { field: 'value' }'. Below it are 'ADD DATA' and 'EXPORT DATA' buttons. The documents list is as follows:

- `_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
include 'connection.php';
$result=$conn->find();
?>
<html>
<body>
<head>
<style>

</style>
</head>
<center>
<table border = 1>
<tr>
<th>NAME</th>
<th>EMAIL</th>
<th>PLACE</th>
<th>DELETE</th>
<th>EDIT</th>
</tr>
<?php
    foreach($result as $res){
?>
<tr>
    <td><?php echo $res ['name'];?></td>
    <td><?php echo $res ['email'];?></td>
```

```

<td><?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>
```

</tr>

```

<?php
}
?>
```

</table>

</body>

</html>

INSERT.PHP

```

<?php
include 'connection.php';
if(isset($_POST['submit'])){
    $name=$_POST['name'];
    $email=$_POST['email'];
    $place=$_POST['place'];
    $doc=array("name"=>$name,"email"=>$email,"place"=>$place);
    $conn->insertOne($doc);
    echo "inserted";
    header( "Location: select.php" );
}
```

?>

Registration.php

<!DOCTYPE html>

```

<html lang="en" dir="ltr">
<head>
    <style>
url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700&display=swap');
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
```

```
}

body{
    min-height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    background: #0e1229;
}

.wrapper{
    position: relative;
    max-width: 430px;
    width: 100%;
    background: #fff;
    padding: 34px;
    border-radius: 6px;
    box-shadow: 0 5px 10px rgba(0,0,0,0.2);
}

.wrapper h2{
    position: relative;
    font-size: 22px;
    font-weight: 600;
    color: #333;
}

.wrapper h2::before{
    content: "";
    position: absolute;
    left: 0;
    bottom: 0;
    height: 3px;
    width: 28px;
    border-radius: 12px;
    background: #4070f4;
}

.wrapper form{
    margin-top: 30px;
}

.wrapper form .input-box{
```

```
height: 52px;  
margin: 18px 0;  
}  
  
form .input-box input{  
    height: 100%;  
    width: 100%;  
    outline: none;  
    padding: 0 15px;  
    font-size: 17px;  
    font-weight: 400;  
    color: #333;  
    border: 1.5px solid #C7BEBE;  
    border-bottom-width: 2.5px;  
    border-radius: 6px;  
    transition: all 0.3s ease;  
}  
  
.input-box input:focus,  
.input-box input:valid{  
    border-color: #4070f4;  
}  
  
form .policy{  
    display: flex;  
    align-items: center;  
}  
  
form h3{  
    color: #707070;  
    font-size: 14px;  
    font-weight: 500;  
    margin-left: 10px;  
}  
  
.input-box.button input{  
    color: #fff;  
    letter-spacing: 1px;  
    border: none;  
    background: #5540f4;  
    cursor: pointer;  
}
```

```
.input-box.button input:hover{
    background: #0e4bf1;
}

form .text h3{
    color: #333;
    width: 100%;
    text-align: center;
}

form .text h3 a{
    color: #4070f4;
    text-decoration: none;
}

form .text h3 a:hover{
    text-decoration: underline;
}


```

</style>

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title> Registration or Sign Up form in HTML CSS | CodingLab </title>
<link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="wrapper">
        <h2>Registration</h2>
        <form method="post" action="insert.php">
            <div class="input-box">
                <input type="text" name="name" placeholder="Enter your name" required>
            </div>
            <div class="input-box">
                <input type="Email" name="email" placeholder="Enter your email" required>
            </div>
            <div class="input-box">
                <input type="text" name="place" placeholder="Place" required>
            </div>
            <div class="input-box button">
                <input type="submit" name="submit">
            </div>
        
```

```
<div class="text">
  <h3>Already have an account? <a href="insert.php">Login now</a></h3>
</div>
</form>
</div>
</body>
</html>
```

Select.php

```
<?php
include 'connection.php';
$result=$conn->find();
?>
<html>
<body>
<head>
<style>

</style>
</head>
<center>
<table border = 1>
<tr>
<th>NAME</th>
<th>EMAIL</th>
<th>PLACE</th>
<th>DELETE</th>
<th>EDIT</th>
</tr>
<?php
  foreach($result as $res){
?>
<tr>
  <td><?php echo $res ['name'];?></td>
  <td><?php echo $res ['email'];?></td>
  <td><?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}>
</table>
```

```
</body>
</html>
```

Delete.php

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

Edit.php

```
<?php
include "connection.php";

$id = $_GET['id'];
$objectId = new MongoDB\BSON\ObjectId($id);
$item = $conn->findOne(['_id' => $objectId]);

if (isset($_POST['submit']))
{
    $name = $_POST['name'];
    $email = $_POST['email'];
    $place = $_POST['place'];
    $updateData = ['$set' => ['name' => $name, 'email' => $email, 'place' => $place]];
}
```

```
$myid = ['_id' => $objectId];
```

```
$result = $conn->updateOne($myid, $updateData);
header("Location: select.php");
```

```
?>
```

```
<html>
```

```
<head>
```

```
    <title>Form </title>
```

```
    <link rel="stylesheet" type="text/css" href="styles.css">
```

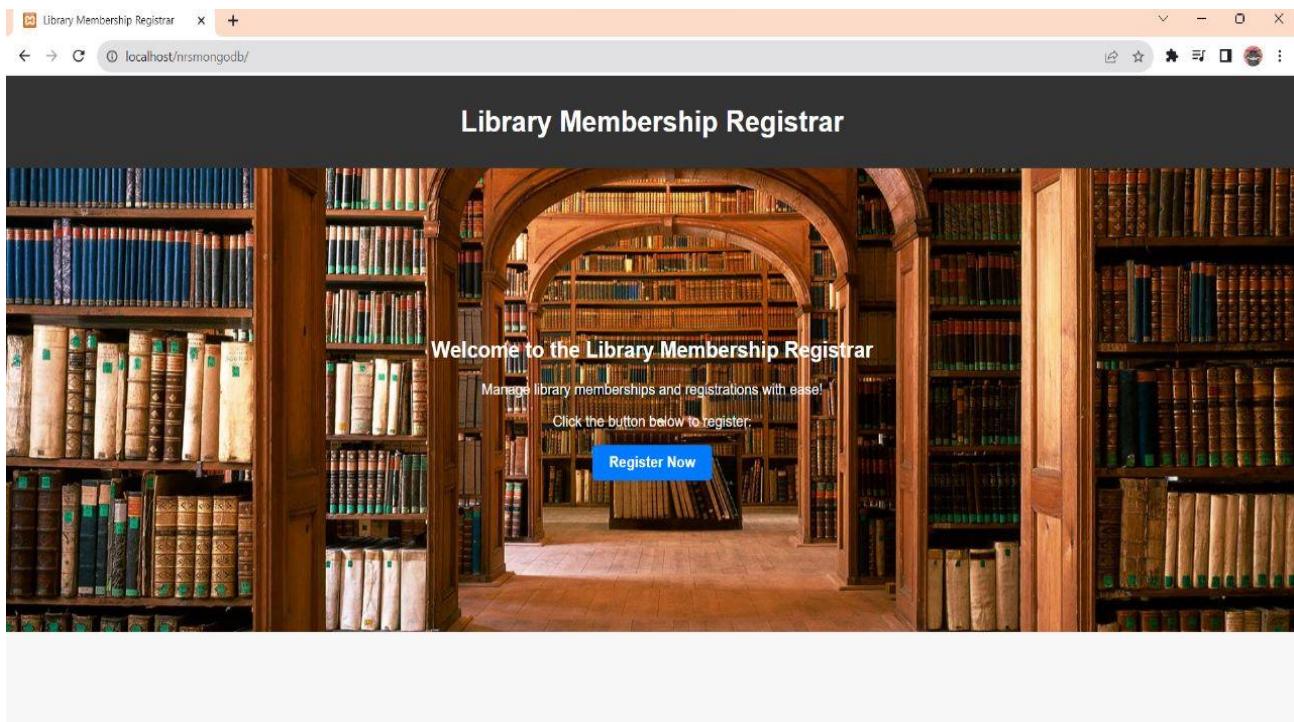
```
</head>
```

```
<body>
```

<center>

```
<h1>Registration Form</h1>
<form method="POST" action="#">
    <label for="name">Full Name:</label>
    <input type="text" value=<?php echo $item["name"]; ?>" name="name" id="name" required><br><br>
    <label for="email">Email:</label>
    <input type="email" value=<?php echo $item["email"]; ?>" name="email" id="email" required><br><br>
    <label for="name">Place:</label>
    <input type="text" value=<?php echo $item["place"]; ?>" name="place" id="place" required><br><br>
    <input type="submit" name="submit" value="Submit">
</form>
<a href="viewdata.php">Go Back</a>
</center>
</body>
</html>
```

OUTPUT



The screenshot shows two browser windows. The top window is titled 'Registration or Sign Up form in ...' and displays a registration form with fields for name, email, place, and a submit button. It also includes a link to log in if already registered. The bottom window is titled 'localhost/nrsmongodb/select.php' and displays a table of user data with columns for Name, Email, Place, Delete, and Edit.

NAME	EMAIL	PLACE	DELETE	EDIT
Amil Mary Joseph	amil@gmail.com	Idukki	Delete	Edit
Anina Elizabeth	anina@gmail.com	Thrissur	Delete	Edit
Anitta Rose	anitta@gmail.com	Pathamthitta	Delete	Edit
Clithira	cb@gmail.com	Wayanad	Delete	Edit
Gayathri	gayu@gmail.com	Kottayam	Delete	Edit
Nandana	nandana@gmail.com	Trivandrum	Delete	Edit
Sanila	san@gmail.com	Thrissur	Delete	Edit
SnehaElsa	elsa@gmail.com	Calicut	Delete	Edit
Neha	neha@gmail.com	Kannur	Delete	Edit
Nandana R.S	nandana@gmail.com	Trivandrum	Delete	Edit
Shruthy	srw@gmail.com	Trivandrum	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

C05

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": "schniederova", "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": "schniederova", "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

Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure

1. Create and Use a Database



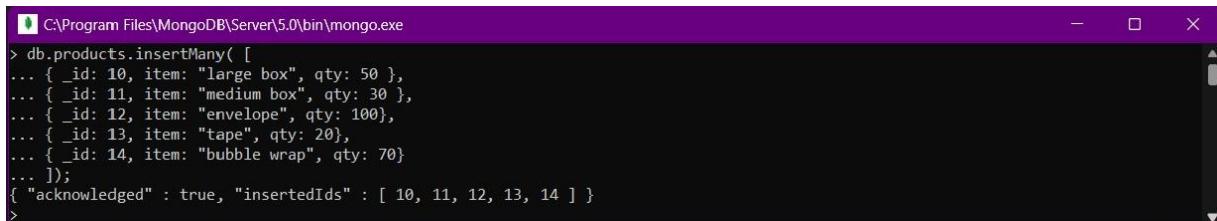
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> use indexdemo;
switched to db indexdemo
> -
```

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: 15

Aim

Usage of Cloud Storage Management Systems: MongoDB Atlas

CO5

Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure

The screenshot shows the MongoDB Atlas website at mongodb.com/atlas. The top navigation bar includes links for Products, Solutions, Resources, Company, Pricing, Sign In, and Try Free. A banner at the top right announces "MongoDB to release new Vector Search and Stream Processing capabilities. Learn more >". The main content features a large green callout graphic comparing "Cluster" and "Serverless" environments. Both sections show four-line charts for Read, Write, Network In, and Network Out metrics. The "Cluster" section has a smaller chart for Disk Usage, while the "Serverless" section has a larger one.

Creating Account :

The screenshot shows the MongoDB deployment interface. It features three main cluster options:

- M10**: \$0.08/hour. For production applications with sophisticated workload requirements. Includes 10 GB storage, 2 GB RAM, and 2 vCPUs.
- SERVERLESS**: \$0.10/1M reads. For application development and testing, or workloads with variable traffic. Includes Up to 1TB storage, Auto-scale RAM, and Auto-scale vCPU.
- M0**: FREE. For learning and exploring MongoDB in a cloud environment. Includes 512 MB storage, Shared RAM, and Shared vCPU.

A green button labeled "Create" is visible at the bottom left. Below it, a note says "Pay-as-you-go! You will be billed hourly and can terminate your cluster anytime. Excludes variable data transfer, backup, and taxes." To the right, a link says "I'll deploy my database later". A small circular icon with a person icon is in the bottom right corner.

Cluster name : cluster0

The screenshot shows the AWS Lambda deployment interface for creating a new cluster. The cluster is named "cluster0".

Region settings show "Mumbai (ap-south-1)" selected. The "Name" field contains "Cluster0". Under "Tag (optional)", there are two fields: "Select or enter key" and "Select or enter value".

Setting password and username for accessing the DB :

The screenshot shows the MongoDB Atlas interface for creating a database user. On the left sidebar, under the 'Data Services' tab, the 'Quickstart' section is selected. In the main area, there are two options: 'Username and Password' (selected) and 'Certificate'. A message box states: 'We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information.' Below this, instructions say: '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.' The 'Username' field contains 'sanilasunny' and the 'Password' field contains 'SAN828136sunny'. There are buttons for 'Autogenerate Secure Password' and 'Copy'. A large green 'Create User' button is at the bottom.

The screenshot shows the MongoDB Atlas interface for configuring IP access. Under the 'Data Services' tab, the 'Quickstart' section is selected. It displays a table with one row: 'Username' (sanilasunny) and 'Authentication Type' (Password). Buttons for 'EDIT' and 'REMOVE' are shown. A message box asks: 'Where would you like to connect from?' It provides two options: 'My Local Environment' (using a local IP icon) and 'Cloud Environment' (using a cloud icon). An 'ADVANCED' link is above the 'Cloud Environment' section. A message box says: 'We added your current IP address. You can connect to your cluster locally from this device.' Below this, a section for 'Add entries to your IP Access List' is shown with a note: 'Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage'.

Connecting Database :

The screenshot shows the MongoDB Atlas interface for a project named "Project 0". The "Data Services" tab is selected. A prominent section titled "Database Deployments" displays a message about loading sample datasets into "Cluster0". It includes a "Load sample dataset" button and a "Dismiss" button. Below this, there's a summary of cluster metrics: R: 0, W: 0, Connections: 0, In: 0.0 B/s, Out: 0.0 B/s, and Data Size: 0.0 B / 512.0 MB (0%). There are also "Connect", "View Monitoring", and "Browse Collections" buttons. On the left sidebar, under the "Database" section, the "Data Lake" option is selected.

Connection Services provided by :

A modal window titled "Connect to Cluster0" is displayed over the main interface. It shows a three-step process: "Set up connection security" (step 1), "Choose a connection method" (step 2), and "Connect" (step 3). Step 1 is completed with a green checkmark. Step 2 is currently active. Step 3 is partially visible. The "Choose a connection method" step contains a section titled "Connect to your application" with a "Drivers" option, which describes using MongoDB's native drivers (e.g., Node.js, Go, etc.). Below this, there are sections for "Access your data through tools" featuring "Compass" (MongoDB's GUI), "Shell" (MongoDB's command-line interface), and "MongoDB for VS Code" (MongoDB support for the Visual Studio Code environment).

Choose connecting option Driver :

Set up connection security

Choose a connection method

Connect

Connecting with MongoDB Driver

1. Select your driver and version
We recommend installing and using the latest driver version.

Driver: Node.js **Version**: 5.5 or later

2. Install your driver
Run the following on the command line

```
npm install mongodb
```

[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

View full code sample

```
mongodb+srv://nandanars:<password>@cluster0.4qf0c3u.mongodb.net/?retryWrites=true&w=majority
```

Replace <password> with the password for the **nandanars** user. Ensure any option params are [URL encoded](#).

Connection code

```

1 <?php
2 require '../vendor/autoload.php';
3
4 //connect to mongodb
5 $con = new MongoDB\Client("mongodb+srv://nandanars:Cz2iYNZFEz6PJ5Ld@cluster0.4qf0c3u.mongodb.net/?retryWrites=true&w=majority");
6
7 //select a databasae
8 $db= $con->AtlasUser;
9
10 echo "Database admin selected";
11
12 $collection = $db->sample;
13
14 echo "Collection sample created succesfully";
15
16 ?>
```

Insert one code

```

1 <?php
2 include_once ("conection.php");
3
4 require '../vendor/autoload.php';
5
6 $atlasuser = array('name' => 'Shruthy' , 'place' =>'kerala');
7
8 $collection -> insertOne($atlasuser);
9
10 echo "value inserted succesfully";
11
12 ?>
```

Output of InsertOne

The screenshot shows the MongoDB Compass interface. In the left sidebar, under the 'AtlasUser' database, there is a 'sample' collection. On the right, the 'Find' tab is selected. A search bar at the top says 'Type a query: { field: 'value' }'. Below it, a 'Filter' button and an 'Apply' button are present. The results section shows 'QUERY RESULTS: 1-4 OF 4' and displays a single document:

```
_id: ObjectId('64b7695a1278134cb2077605')
name: "Shruthy"
place: "kerala"
```

InsertMany code

```

1 <?php
2 include_once ("conection.php");
3
4 require '../vendor/autoload.php';
5
6 $atlasuser1 = array('name' => 'Hashim' , 'place' =>'xyz');
7 $atlasuser2 = array('name' => 'Amal' , 'place' =>'abc');
8 $atlasuser3 = array('name' => 'Nandana' , 'place' =>'lmn');
9
10
11 $collection -> insertMany([ $atlasuser1, $atlasuser2 , $atlasuser3 ]);
12
13 echo "values inserted succesfully";
14 ?>
15
16
17

```

Output of insertMany:

The screenshot shows the MongoDB Compass interface. The left sidebar shows the 'sample' collection under 'AtlasUser'. The 'Find' tab is selected. The results section shows three documents inserted successfully:

```

_id: ObjectId('64b769b11278134cb2077606')
name: "Hashim"
place: "xyz"

_id: ObjectId('64b769b11278134cb2077607')
name: "Amal"
place: "abc"

_id: ObjectId('64b769b11278134cb2077608')

```

At the bottom left, it says 'System Status: All Good'.

MONGO ATALS APP SERVICE

The screenshot shows the MongoDB Atlas App Service interface. On the left, a sidebar lists 'Project 0' with sections for Apps, Application-0, NO ENVIRONMENT, DATA ACCESS (Rules, Schema, App Users), and BUILD (Realm SDKs, Device Sync, GraphQL, Functions, Triggers, HTTPS Endpoints). A central modal window titled 'Welcome to your Application Guides' displays five guided walkthroughs: 'Device Sync' (10 minutes), 'GraphQL' (10 minutes), 'Triggers' (10 minutes), 'Web SDK' (10 minutes), and 'Endpoints' (10 minutes). Below this, a large form for creating a new app service is shown. Step 1, 'Name', has a field containing 'Application-1'. Step 2, 'Link your Database', shows options for connecting to an existing MongoDB Atlas cluster ('Cluster0') or creating a new one ('We'll automatically create a free-tier 6.0 Atlas cluster for you with 512 MB of space'). Other optional steps like 'CONNECT TO GITHUB' and 'ADVANCED CONFIGURATION' are also listed. At the bottom right of the form is a green 'Create App Service' button.

The screenshot shows the OCI App Services interface for a project named 'Project 0'. The main view is for 'Application-1' with the App ID 'application-1-lhacw' and location 'AWS - Virginia (us-east-1)'. A 'Get Started With App Services' section provides links to 'REALM CLI', 'SDKS', 'DOCUMENTATION', and 'DEPLOYMENT'. Below this is a 'Metrics' section showing a green circular progress bar. The left sidebar contains sections for 'DATA ACCESS' (Rules, Schema, App Users, Authentication), 'BUILD' (Realm SDKs, Device Sync, GraphQL, Functions, Triggers, HTTPS Endpoints, Values), and 'MANAGE' (Atlas, Access Manager). The bottom section shows 'Total App Usage' for the period Jul 01 - Jul 31, with metrics for Requests (0/1M FREE TIER USED), Data Transfer (0.00/10 GB FREE TIER USED), Compute Runtime (0.00/500 Hr FREE TIER USED), and Sync Runtime (0.00/10K Hr FREE TIER USED).

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

C05:

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 -repSet datascience -logpath \data\rs1\1.log -dbpath \data\rs1 -port 27018
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -repSet datascience -logpath \data\rs2\2.log -dbpath \data\rs2 -port 27019
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -repSet 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
datastage:SECONDARY> rs.status()
{
  "set" : "datastage",
  "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),
      "t" : NumberLong(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" : ""
  }
],
```

```
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
}

Select C:\Windows\System32\cmd.exe - mongo --port 27018
{
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1690344920, 2),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
            "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 dataascidb
switched to db dataascidb
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

```

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 <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,"AAAAAAAAAAAAAAAAAAAAAA="),
        "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
datacidb 0.000GB
local 0.000GB
datascience:SECONDARY> use datacidb
switched to db datacidb
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,"AAAAAAAAAAAAAAAAAAAAAA="),
        "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("64c0a1023151b8630c0e8860"), "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
datacidb 0.000GB
local 0.000GB
datascience:SECONDARY> use datacidb
```



```
on C:\Windows\System32\cmd.exe - mongo -port 27020
dataScience:SECONDARY> use dataScience
switched to db dataScience
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("64c0a1023151b8630c0e88e0"), "name" : "akhil" }
dataScience:SECONDARY>
```

Result:

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