

20MCA134 – ADVANCED DBMS LAB

Lab Report Submitted By

JENNY JOHNSON

Reg. No.: AJC22MCA-2053

In Partial fulfilment for the Award of the Degree Of

MASTER OF COMPUTER APPLICATIONS (2 Year) (MCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



**AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,
Accredited by NAAC with ‘A’ grade. Koovapally, Kanjirappally, Kottayam, Kerala – 686518]

2022-2023

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING KANJIRAPPALLY



CERTIFICATE

This is to certify that the lab report, "**20MCA134-ADVANCED DBMS LAB**" is the bonafide work of **JENNY JOHNSON (AJC22MCA-2053)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year **2022-23**.

Mr. G S Ajith

Lab In- Charge

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

Internal Examiner

External Examiner

Course Code	Course Name	Syllabus Year	L-T-P-C
20MCA134	Advanced DBMS Lab	2020	0-1-3-2

VISION

To promote an academic and research environment conducive for innovation centric technical education.

MISSION

- MS1 - Provide foundations and advanced technical education in both theoretical and applied Computer Applications in-line with Industry demands.
- MS2 - Create highly skilled computer professionals capable of designing and innovating real life solutions.
- MS3 - Sustain an academic environment conducive to research and teaching focused to generate up-skilled professionals with ethical values.
- MS4 - Promote entrepreneurial initiatives and innovations capable of bridging and contributing with sustainable, socially relevant technology solutions.

COURSE OUTCOME

CO	Outcome	Target
CO1	Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.	65
CO2	Apply PL/SQL for processing databases	65
CO3	Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.	65
CO4	Apply CRUD operations and retrieve data in a NoSQL environment.	65
CO5	Understand the basic storage architecture of distributed file systems.	65
CO6	Design and deployment of NoSQL databases with real time requirements.	60

COURSE END SURVEY

CO	Survey Question	Answer Format
CO1	To what extend you are able to design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling , designing and implementing a database.	Excellent/Very Good/Good/Fair/Poor
CO2	To what extend you are able to apply PL/SQL for processing datavases.	Excellent/Very Good/Good/Fair/Poor
CO3	To what extend you are able to compare relational and non-relational(NoSQL)databases and configuration of NoSQL Databases.	Excellent/Very Good/Good/Fair/Poor
CO4	To what extend you are able apply CRUD operations and retrieve data in a NoSQL environment.	Excellent/Very Good/Good/Fair/Poor
CO5	To what extent you are able to understand the basic storage architecture of distributed file systems.	Excellent/Very Good/Good/Fair/Poor
CO6	To what extend you are able to design and deployment of NoSQL databases with real time experiments.	Excellent/Very Good/Good/Fair/Poor

Sl. No.	Experiment	Date	CO	Page No.
1	Creation of a database using DDL commands.	15-03-2023	CO1	1
2	Creation of a database using DML commands including integrity constraints.	17-03-2023	CO1	4
3	To familiarize with selecting data from single table	22-03-2023	CO1	6
4	To familiarize with set operations	24-03-2023	CO1	11
5	To familiarize with aggregate functions	29-03-2023	CO1	14
6	To familiarize with Join or Cartesian Product	31-03-2023	CO1	18
7	To familiarize with Group By and Having Clauses	31-03-2023	CO1	21
8	To familiarize with trigger functions	05-04-2023	CO2	24
9	To familiarize with Stored functions and Procedure	12-04-2023	CO2	26
10	Understand the installation and configuration of NoSQL Databases: MongoDB	14-06-2023	CO3	27
11	Create a database and use the insertMany method to insert the colors of the rainbow into the database	16-06-2023	CO4	32
12	Implementing CRUD operations using PHP-MongoDB	22-06-2023	CO4	34
13	Build sample collections/documents to perform the shell queries	12-07-2023	CO5	40
14	To familiarize with indexing in MongoDB	12-07-2023	CO5	44
15	Implementation of Replica Set on MongoDB	19-07-2023	CO5	47
16	Usage of Cloud Storage Management Systems: MongoDB Atlas.	21-07-2023	CO6	51

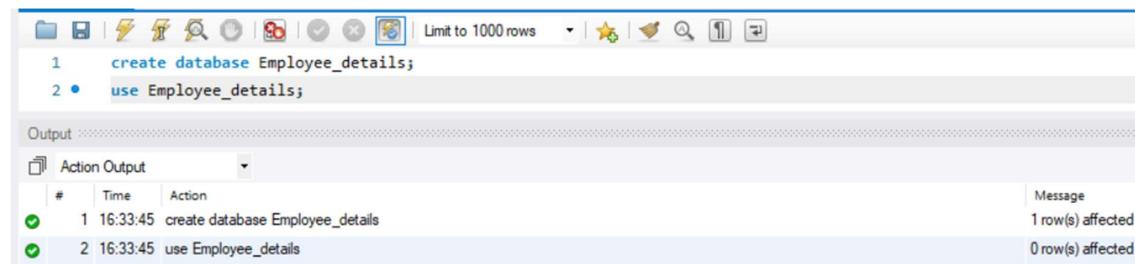
Experiment No.: 1

Aim: Creation of a database using DDL commands.

CO1: Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

Procedure

1.Create database Employee_details



```

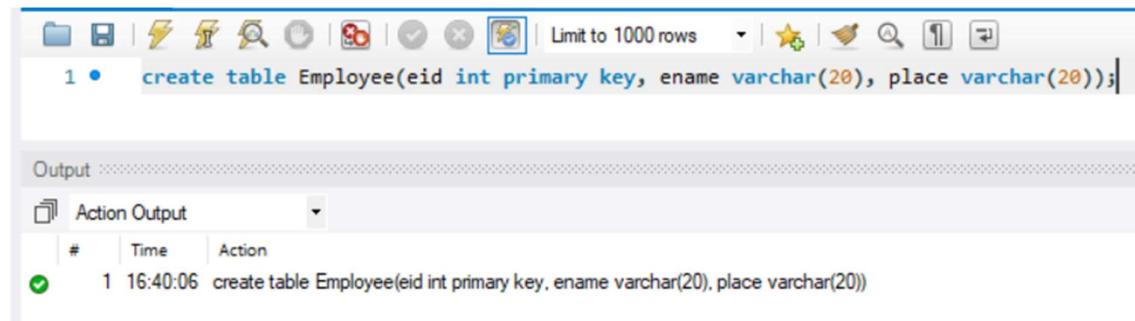
1  create database Employee_details;
2 • use Employee_details;

Output ::

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

```

2.Create table Employee



```

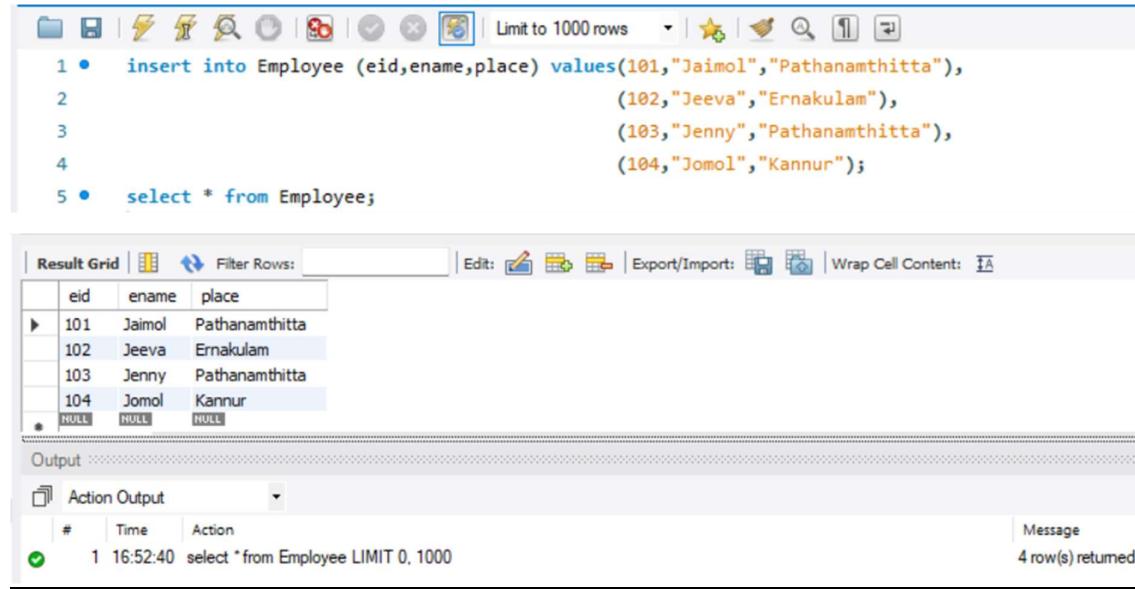
1 • create table Employee(eid int primary key, ename varchar(20), place varchar(20));

Output ::

Action Output
# Time Action
1 16:40:06 create table Employee(eid int primary key, ename varchar(20), place varchar(20))

```

3.Insert values into table Employee



```

1 • insert into Employee (eid,ename,place) values(101,"Jaimol","Pathanamthitta"),
2
3
4
5 • select * from Employee;

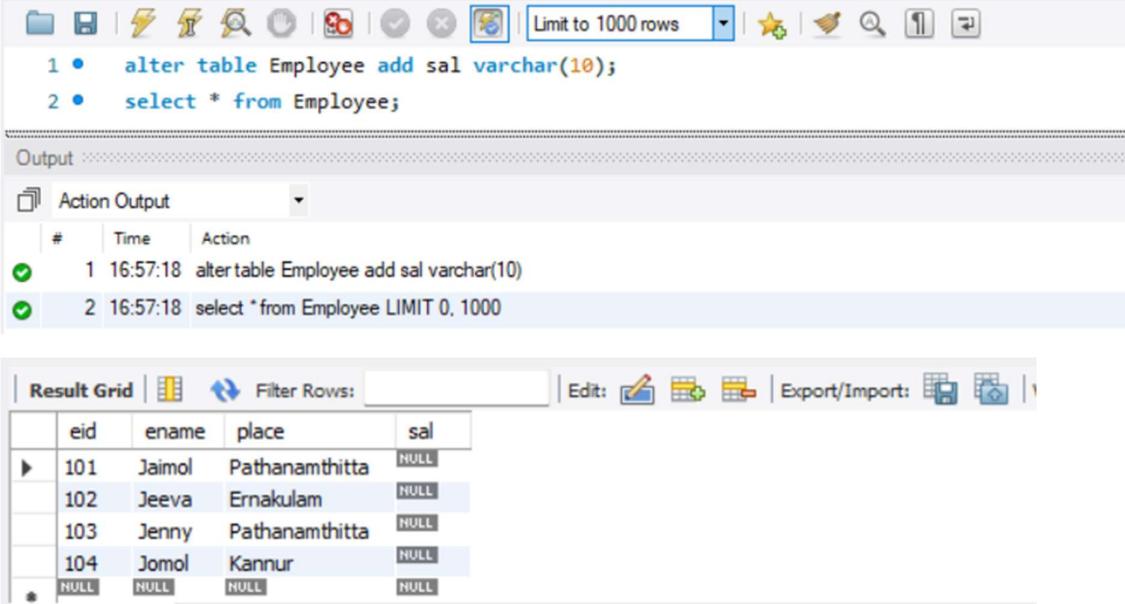
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |
eid ename place
101 Jaimol Pathanamthitta
102 Jeeva Ernakulam
103 Jenny Pathanamthitta
104 Jomol Kannur
* NULL NULL NULL

Output ::

Action Output
# Time Action Message
1 16:52:40 select * from Employee LIMIT 0, 1000 4 row(s) returned

```

4.Alter table- add column sal to table Employee



```

1 • alter table Employee add sal varchar(10);
2 • select * from Employee;

```

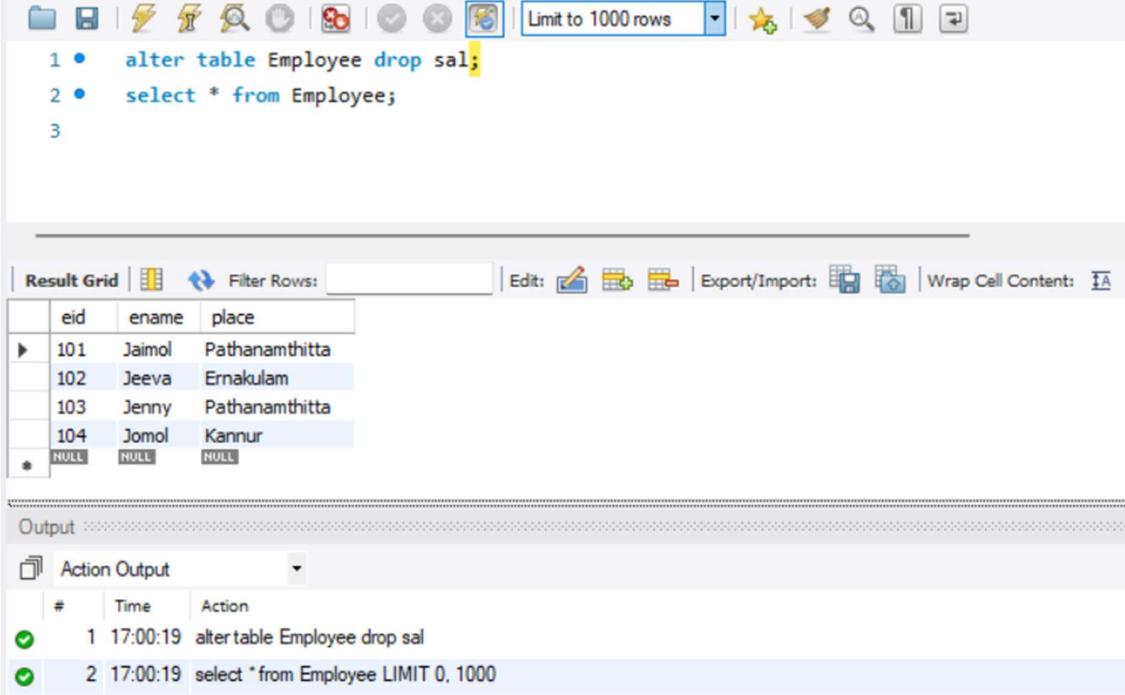
Output :

#	Time	Action
1	16:57:18	altertable Employee add sal varchar(10)
2	16:57:18	select *from Employee LIMIT 0, 1000

Result Grid

	eid	ename	place	sal
▶	101	Jaimol	Pathanamthitta	NULL
	102	Jeeva	Ernakulam	NULL
	103	Jenny	Pathanamthitta	NULL
	104	Jomol	Kannur	NULL
*	NULL	NULL	NULL	NULL

5.Drop column sal



```

1 • alter table Employee drop sal;
2 • select * from Employee;
3

```

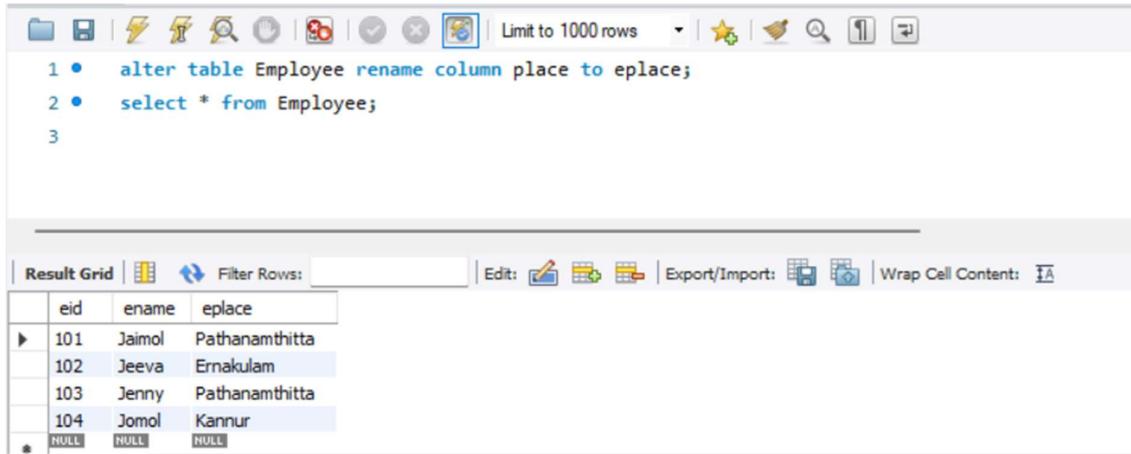
Result Grid

	eid	ename	place
▶	101	Jaimol	Pathanamthitta
	102	Jeeva	Ernakulam
	103	Jenny	Pathanamthitta
	104	Jomol	Kannur
*	NULL	NULL	NULL

Output :

#	Time	Action
1	17:00:19	altertable Employee drop sal
2	17:00:19	select *from Employee LIMIT 0, 1000

6. Rename column place to eplace



```

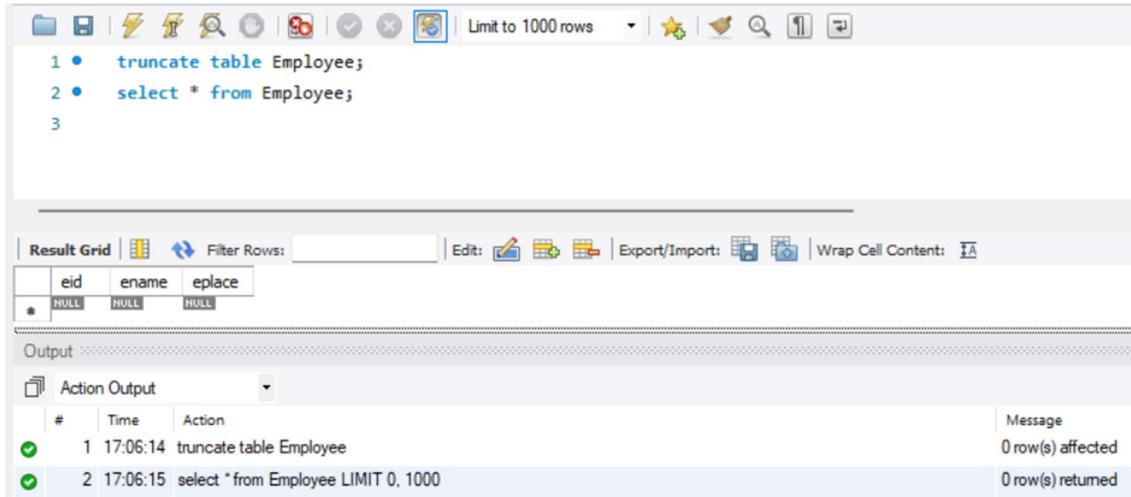
1 • alter table Employee rename column place to eplace;
2 • select * from Employee;
3

```

Result Grid

	eid	ename	eplace
▶	101	Jaimol	Pathanamthitta
	102	Jeeva	Ernakulam
	103	Jenny	Pathanamthitta
	104	Jomol	Kannur
*	NULL	NULL	NULL

7. Truncate table Employee



```

1 • truncate table Employee;
2 • select * from Employee;
3

```

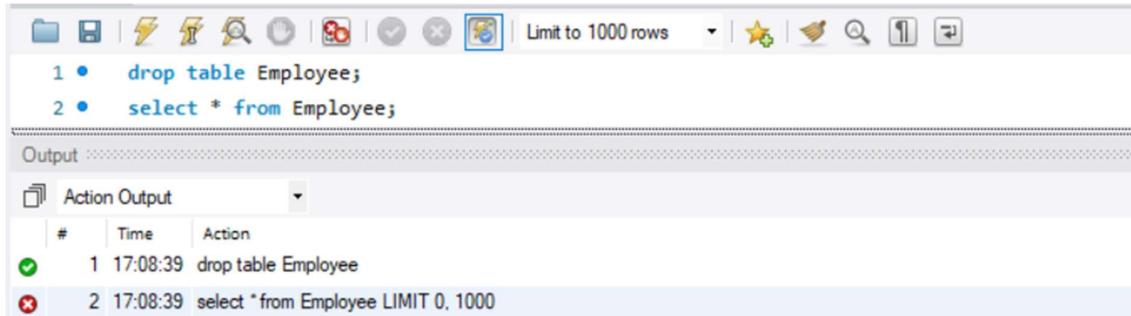
Result Grid

	eid	ename	eplace
*	NULL	NULL	NULL

Output

Action Output		
#	Time	Action
1	17:06:14	truncate table Employee
2	17:06:15	select *from Employee LIMIT 0, 1000

8. Drop table Employee



```

1 • drop table Employee;
2 • select * from Employee;

```

Output

Action Output		
#	Time	Action
1	17:08:39	drop table Employee
2	17:08:39	select *from Employee LIMIT 0, 1000

Result

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

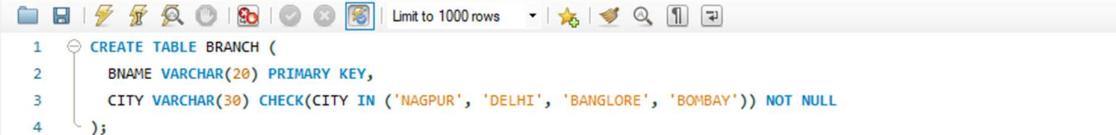
Experiment No.: 2

Aim: Creation of a database using DML commands including integrity constraints.

CO1: Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

Procedure

1. Create table - Branch

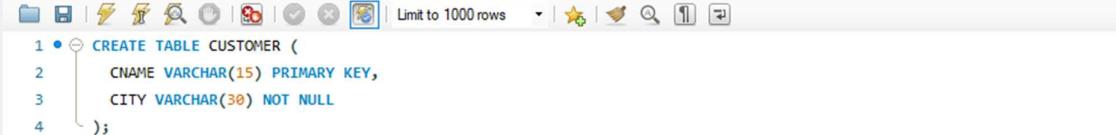


```

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

```

2. Create table - Customer



```

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

```

3. Create table - Deposit

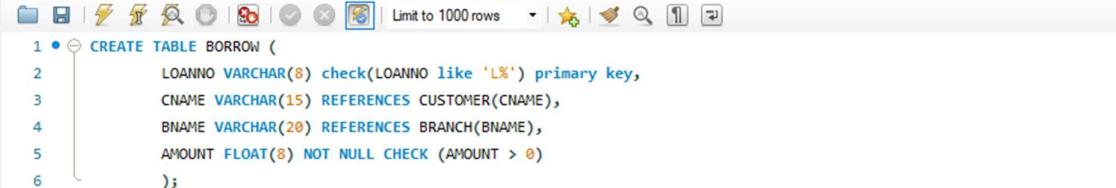


```

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

```

4. Create table - Borrow



```

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

```

5. Insert values in table - Customer

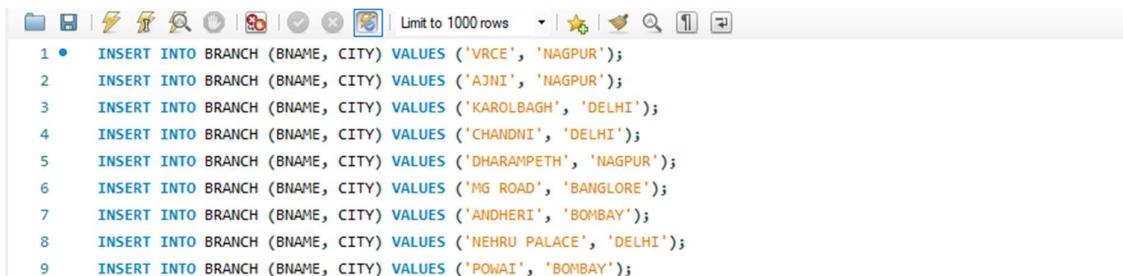


```

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

```

6. Insert values in table - Branch

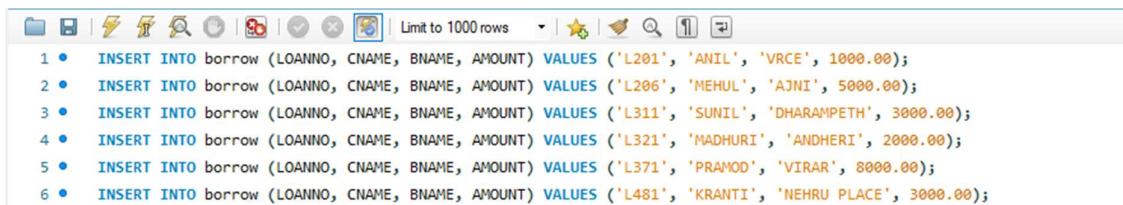


```

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

```

7. Insert values in table - Borrow

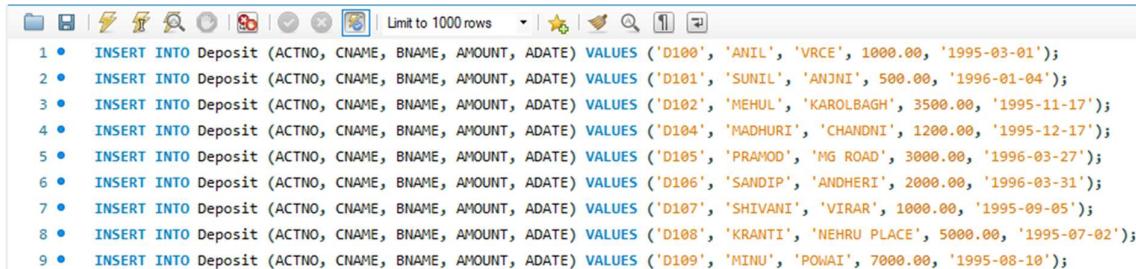


```

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);

```

8.. Insert values into table - Deposit

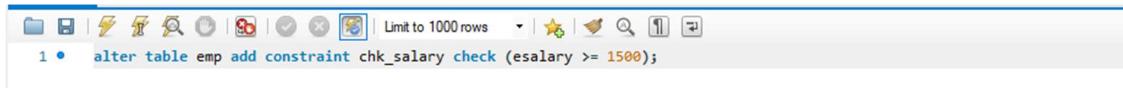


```

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');

```

9. Add constraint - salary <= 5000

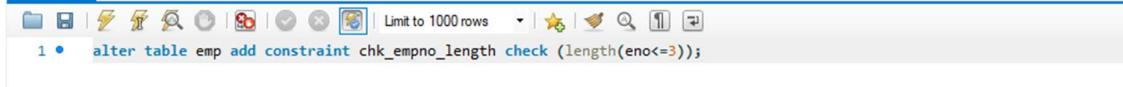


```

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

```

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



```

1 •  alter table emp add constraint chk_empno_length check (length(eno)>=3);

```

Result

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

Experiment No.: 3

Aim: To familiarize with selecting data from single table.

CO1: Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

Procedure

1. list all data from table deposito

select * from deposito;

	ACTNO	CNAME	BNAME	AMOUNT	ADATE
▶	D100	ANIL	VRCE	1	1995-03-01
	D101	SUNIL	AJNI	500	1996-01-04
	D102	MEHUL	KAROLBAGH	3500	1995-11-17
	D104	MADHURI	CHANDNI	1200	1995-12-17
	D105	PRAMOD	MG ROAD	3000	1996-03-27
	D106	SANDIP	ANDHERI	2000	1996-03-31
	D107	SHIVANI	DHARAMPETH	1000	1995-09-05
	D108	KRANTI	NEHRU	5000	1995-07-02
	D109	MANDAR	POWAI	7000	1995-08-10

2. list all data from borrow

select * from borrow;

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

3. list all data from customer

select * from customer;

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

4. list all data from branch

select * from branch;

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
BNAME	CITY				
► AJNI	NAGPUR				
ANDHERI	BOMBAY				
CHANDNI	DELHI				
DHARAMPETH	NAGPUR				
KAROLBAGH	DELHI				
MG ROAD	BANGALORE				
NEHRU	PALACE DELHI				
BORROW 22	DEPOSITE 23	CUSTOMER 24	BRANCH 25	x	

5. give account no and amount of deposite

select actno ,amount from deposite;

Result Grid		Filter Rows:	Edit:	Export/Import:
ACTNO	AMOUNT			
► D100	1			
D101	500			
D102	3500			
D104	1200			
D105	3000			
D106	2000			
D107	1000			
D108	5000			
D109	7000			
* NULL	NULL			

6. give customer name and account no of depositor

select actno ,cname from deposite;

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
ACTNO	CNAME				
► D100	ANIL				
D108	KRANTI				
D104	MADHURI				
D109	MANDAR				
D102	MEHUL				
D105	PRAMOD				
D106	SANDIP				
D107	SHIVANI				
D101	SUNIL				
* NULL	NULL				

7. give name of customers

select cname from customer;

CNAME
ANIL
KRANTI
MADHURI
MANDAR
MEHUL
NAREN
PRAMOD
SANDIP
SHIVANI
SUNIL

8. give name of branches

select bname from branch;

BNAME
AJNI
ANDHERI
CHANDNI
DHARAMPETH
KAROLBAGH
MG ROAD
NEHRU
POWAI
VRCE

9. give name of borrows

select cname from borrow;

CNAME
ANIL
KRANTI
MADHURI
MEHUL
PRAMOD
SUNIL

10. give names of customer living in city nagpur

select cname from customer where city="nagpur";

CNAME
MADHURI
PRAMOD

11. give names of depositors having amount greater than 4000

select cname from deposite where amount>4000;

Result Grid		<input type="button" value="Filter Rows:"/>	<input type="button" value="Export:"/>	<input type="checkbox"/> Wrap Cell Content:
	CNAME			
▶	KRANTI			
	MANDAR			

12. give account date of anil

select adate from deposite where cname="anil";

Result Grid		<input type="button" value="Filter Rows:"/>	<input type="button" value="Edit:"/>	<input type="checkbox"/>
	ADATE			
▶	1995-03-01			

13. give name of all branches located in bombay

select bname from branch where city="bombay";

Result Grid		<input type="button" value="Filter Rows:"/>	<input type="button" value="Edit:"/>	<input type="checkbox"/>
	BNAME			
▶	ANDHERI			
	POWAI			

14. give name of borrower having loan number l205

select cname from borrow where loanno="l205";

Result Grid		<input type="button" value="Filter Rows:"/>	<input type="button" value="Export:"/>	<input type="checkbox"/> Wrap Cell Content:
	CNAME			

15. give names of depositors having account at vrce

select cname from deposite where bname="vrce";

Result Grid		<input type="button" value="Filter Rows:"/>	<input type="button" value="Export:"/>	<input type="checkbox"/> Wrap Cell Content:
	CNAME			
▶	ANIL			

16. give names of all branched located in city delhi

```
select bname from branch where city="delhi";
```

Result Grid		Filter Rows:	Edit:	Export/Import:
	BNAME			
▶	CHANDNI			
	KAROLBAGH			

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

```
select cname from deposite where adate='1996-12-01';
```

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

18. give account no and deposit amount of customers having account opened between dates '1-12-96' and '1-5-96'

```
select actno , cname from deposite where adate between '1996-12-01' and '1996-05-01';
```

Result Grid		Filter Rows:	Edit:
	ACTNO	CNAME	
*	NULL	NULL	

19. give name of the city where branch karolbagh is located

```
select city from branch where bname="karolbagh";
```

Result Grid		Filter Rows:	Export:	Wrap Cell Co
	CITY			
▶	DELHI			

20. give details of customer anil

```
select * from customer where cname="anil";
```

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 modelling, designing and implementing a database.

Procedure

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

select cname from deposite where not exists (select cname from borrow);

Result Grid		Filter Rows:
CNAME		

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

select cname from deposite union (select cname from borrow);

Result Grid		Filter Rows:
CNAME		
▶	ANIL	
	KRANTI	
	MADHURI	
	MANDAR	
	MEHUL	
	PRAMOD	
	SANDIP	
	SHIVANI	
	SUNIL	

3. list all the depositors having deposit in all the branches where sunil is having account.

select cname from deposit where bname in (select bname from deposit where cname="sunil");

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

**4. list all the customers living in city nagpur and having branch city
Bombay or delhi.**

select cname from customer where city="nagpur" and city in(select city from branch where city="bombay" or city="delhi");

Result Grid	
cname	F

5. list all the depositors living in city nagpur

select cname from deposit where cname in(select cname from customer where city="nagpur");

Result Grid	
Filter Rows:	<input type="text"/>
Export:	
Wrap Cell Content: <input checked="" type="checkbox"/>	
cname	
MADHURI	
PRAMOD	

**6. list all the depositors living in the city nagpur and having branch in city
Bombay**

select cname from deposit where cname in(select cname from customer where city="nagpur" and bname in(select bname from branch where city="bombay"));

Result Grid	
cname	

7. list the branch cities of anil and sunil

select city from branch where bname in(select bname from deposit where cname="anil" and cname="sunil");

Result Grid	
city	

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

select cname from deposit where amount>1000 and amount in(select amount<10000 from borrow)

9. list the cities of depositors having branch vrce.

select city from customer where cname in(select cname from deposit where bname='vrce');

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

► CALCUTTA

10. list the depositors having amount less than 1000 and living in the same city as anil

select d1.cname from deposit d1, customer c1 where amount<1000 and c1.city=(c1.cname='anil');

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

► SUNIL

11. list all the cities where branches of anil and sunil are locate

select b1.city from branch b1 where b1.bname in (select d1.bname from deposit d1 where d1.cname in ('anil','sunil'));

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

► NAGPUR

12. list the amount for the depositors living in the city where anil is living

select distinct(d1.cname),d1.amount ,c1.city from deposit d1, customer c1, branch b1 where d1.cname=c1.cname and c1.city in(select c2.city from customer c2 where c2.cname='anil');

Result Grid		Filter Rows:	
	CNAME	AMOUNT	CITY
►	ANIL	1	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 modelling, designing and implementing a database.

Procedure

1.list total loan

select sum(amount) from borrow;

Result Grid		Filter Rows:	Export:
sum(AMOUNT)			
▶ 22000			

2.list total deposit

select sum(amount) from deposite;

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
SUM(AMOUNT)				
▶ 24200.00				

3.list total loan taken from karolbagh branch

select sum(amount) from borrow where bname="karolbagh";

Result Grid		Filter Rows:	Export:
sum(amount)			
▶ NULL			

4.list total deposit of customers having account date later than 1-jan-96

select sum(amount) from deposite where adate>"1996-01-01";

Result Grid		Filter Rows:	Export:
sum(amount)			
▶ 5500			

5.list total deposit of customers living in city nagpur

select sum(amount) from deposite where cname in(select cname from customer where city="nagpur");

Result Grid		Filter Rows:	Export:
sum(amount)			
▶ 4200			

6.list maximum deposit of customer living in bombay

select max(amount) from deposite where cname in(select cname from customer where city="bombay");

Result Grid		Filter Rows:	Export:
	max(amount)		

▶ 5000

7.list total deposit of customer having branch in bombay

select max(amount) from deposite where bname in(select bname from branch where city="bombay");

Result Grid		Filter Rows:	Export:
	max(amount)		

▶ 7000

8.count total number of branch cities

select count(city) from branch;

Result Grid		Filter Rows:	Export:
	count(city)		

▶ 9

9.count total number of customers cities

select count(city) from customer;

Result Grid		Filter Rows:	Export:
	count(city)		

▶ 10

10.give branch names and branch wise deposit

select bname , sum(amount) from deposite group by bname;

Result Grid		Filter Rows:
	BNAME	SUM(AMOUNT)
▶	AJNI	500
	ANDHERI	2000
	CHANDNI	1200
	DHARAMPETH	1000
	KAROLBAGH	3500
	MG ROAD	3000
	NEHRU	5000
	POWAI	7000
	VRCE	1

11.give city wise name and branch wise deposit

```
select c1.city , sum(d1.amount) from customer c1 , deposite d1 where d1.cname = c1.cname
group by c1.city;
```

	CITY	SUM(D1.AMOUNT)
▶	CALCUTTA	1
	DELHI	500
	BARODA	3500
	NAGPUR	4200
	SURAT	2000
	BOMBAY	6000
	PATNA	7000

12.give the branch wise loan of customer living in Nagpur

```
select bname , sum(amount) from borrow,customer where city='nagpur' group by bname;
```

	BNAME	SUM(AMOUNT)
▶	VRCE	2000
	AJNI	10000
	DHARAMPETH	6000
	ANDHERI	4000
	POWAI	16000
	NEHRU	6000

13.count total number of customers

```
select count(cname) from customer;
```

	count(cname)
▶	10

14.count total number of depositors branch wise

```
select bname, count(*) from deposite, customer where deposite.cname=customer.cname
group by bname;
```

	BName	count(*)
▶	VRCE	1
	AJNI	1
	KAROLBAGH	1
	CHANDNI	1
	MG ROAD	1
	ANDHERI	1
	DHARAMPETH	1
	NEHRU	1
	POWAI	1

15.give maximum loan from branch vrce

```
select max(amount) from borrow where bname ='vrce';
```

	MAX(AMOUNT)
▶	1000

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

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

	count(customer.CNAME)
▶	6

Result

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

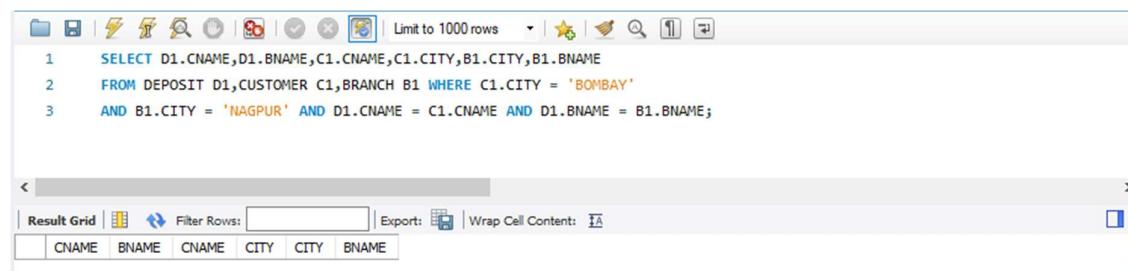
Experiment No.:6

Aim: To familiarize with Join or Cartesian Product.

CO1: Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

Procedure

1. Give name of customers having living city bombay and branch city nagpur



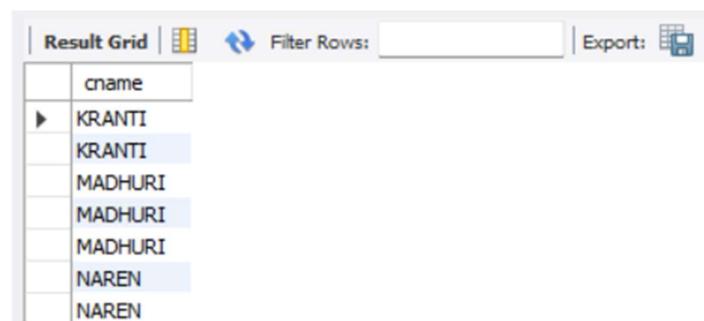
```

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

```

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

select cname from customer,branch where customer.city=branch.city;

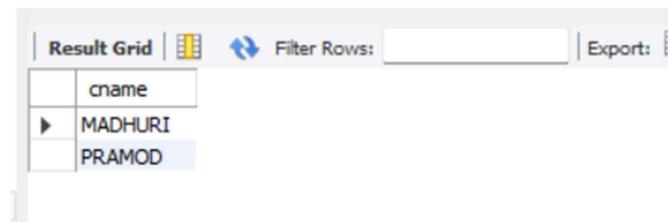


cname
KRANTI
KRANTI
MADHURI
MADHURI
MADHURI
NAREN
NAREN

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

nagpur.

select cname from customer where city="nagpur" and cname in(select cname from borrow) and cname in(select cname from deposite);



cname
MADHURI
PRAMOD

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

select cname,amount from borrow where amount > 2000 and cname in(select cname from deposite where amount > 1000);

Result Grid		Filter Rows:	Export:
	cname	amount	
▶	MEHUL	5000	
	PRAMOD	8000	
	KRANTI	3000	

5. give names of depositors having the same branch as the branch of sunil

select cname from deposit where bname in(select bname from branch where cname="sunil");

Result Grid		Filter Rows:	Export:
	cname		
▶	SUNIL		

6.give names of borrowers having loan amount greater than the loan amount of pramod

select br1.cname,br1.amount from borrow br1 where br1.amount > all (select br2.amount from borrow br2 where br2.cname = 'pramod');

Result Grid		Filter Rows:
	CNAME	AMOUNT

7.give the name of the customer living in the city where branch of depositor sunil is located.

select c.cname from customer c where c.city in (select b.city from branch b where b.bname in (select d.bname from deposite d where d.cname='sunil'));

Result Grid		Filter
	CNAME	
▶	MADHURI	
	PRAMOD	
*	HULL	

8. give branch city and living city of pramod

```
select b.city,c.city from branch b,customer c where cname ="pramod";
```

Result Grid		Filter Rows:
	CITY	CITY
▶	BANGALORE	NAGPUR

9.Give branch city of sunil and branch city of anil

```
select b1.city from deposite d1, branch b1 where d1.bname = b1.bname and d1.cname in ('sunil','anil');
```

Result Grid		Filter
	CITY	
▶	NAGPUR	
	NAGPUR	

10.Give the living city of anil and the living city of sunil

```
select c1.cname, c1.city from customer c1 where c1.cname ='anil' or c1.cname = 'sunil';
```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
CNAME	CITY				
ANIL	CALCUTTA				
SUNIL	DELHI				
NULL	NULL				

Result

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

Experiment No.: 7

Aim: To familiarize with Group by and Having Clauses.

CO1: Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

Procedure**1.list the branches having sum of deposit more than 5000.**

select bname from branch where bname in(select bname from deposite group by bname having sum(amount)>5000);

Result Grid	
<input type="button" value="Filter Rows:"/>	
bname	
▶	POWAI

2.list the branches having sum of deposit more than 500 and located in city bombay

select bname from branch where bname in(select bname from deposite group by bname having sum(amount)>500 and city="bombay");

Result Grid	
<input type="button" value="Filter Rows:"/>	
<input type="button" value="Export:"/>	
bname	
▶	ANDHERI
	POWAI

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

select cname from deposite where amount=(select avg(amount) from deposite group by bname having avg(amount)>5000);

Result Grid	
<input type="button" value="Filter Rows:"/>	
<input type="button" value="Export:"/>	
cname	
▶	MANDAR

4.list the names of customers having maximum deposit

```
select cname,amount from deposite where amount =(select max(amount) from deposite);
```

Result Grid		Filter Rows:	Export:
	cname	AMOUNT	
▶	MANDAR	7000	

5.list the name of the branch having the highest number of depositors

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.

```
select count(cname) from deposite where cname in( select cname from customer where city="nagpur");
```

Result Grid		Filter Rows:
	count(cname)	
▶	2	

7.give names of customers in vrce branch having more deposit than any other customer in same branch.

```
select cname from deposite where bname='vrce' and amount=(select max(amount) from deposite where bname='vrce');
```

Result Grid		Filter Row
	CNAME	
▶	ANIL	

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

select bname from deposit group by bname having count(bname)>5;

		Result Grid		Filter Rows:	<input type="text"/>	Export:		Wrap Cell Content:	
		CNAME							

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

select c.cname ,count(b.bname) from customer c inner join branch b on c.cname=b.bname group by c.cname order by count(b.bname) desc;

		Result Grid		Filter Rows:	<input type="text"/>	Export:		Wrap Cell Content:	
		CNAME	COUNT(B.BNAME)						

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

select count(b1.bname) from deposite d1 , borrow b1 , customer c1 where c1.cname=d1.cname and d1.cname=b1.cname and c1.city in (select city from customer);

		Result Grid		Filter Rows:	<input type="text"/>
		count(b1.bname)			
		6			

Result

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

Experiment No.: 8

Aim: To familiarize with trigger functions.

CO2: Apply PL/SQL for processing databases

Procedure

Step 1: Start

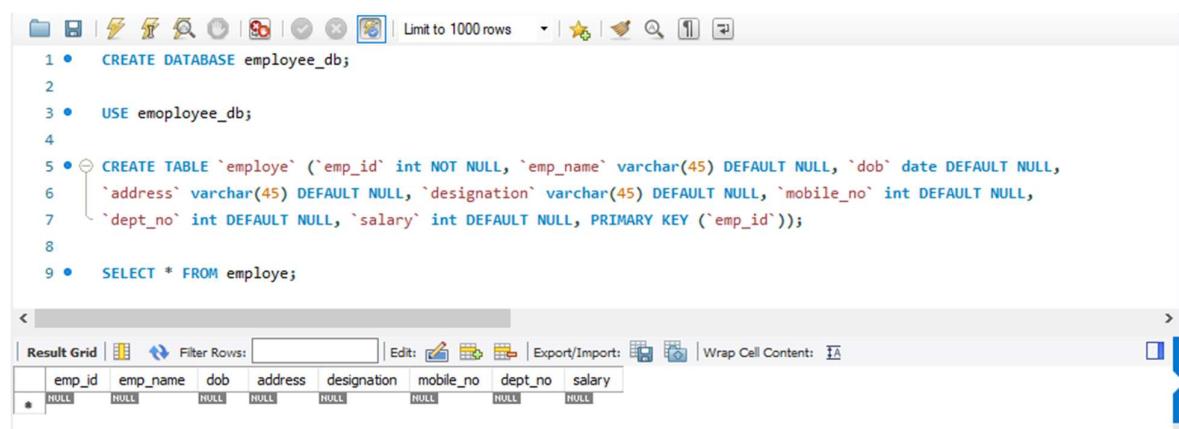
Step 2: Initialize the trigger.

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

Step 4: Execute the trigger procedure after updating

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

Step 6: Stop

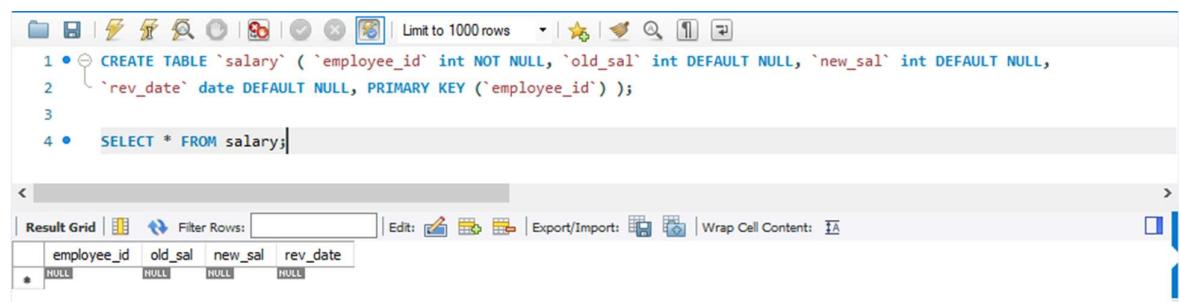


```

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

```

The screenshot shows the MySQL Workbench interface. At the top, there are various toolbar icons. Below the toolbar, a code editor window displays the creation of a database named 'employee_db' and a table named 'employe'. The table has columns: emp_id, emp_name, dob, address, designation, mobile_no, dept_no, and salary. A primary key is defined on the emp_id column. At the bottom of the code editor, there is a 'Result Grid' tab which is currently inactive, indicated by a grey background. The status bar at the bottom of the window shows 'Limit to 1000 rows'.



```

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

```

The screenshot shows the MySQL Workbench interface. It displays the creation of a table named 'salary'. The table has columns: employee_id, old_sal, new_sal, and rev_date. A primary key is defined on the employee_id column. Below the code editor, there is a 'Result Grid' tab which is currently inactive. The status bar at the bottom shows 'Limit to 1000 rows'.

The screenshot shows the MySQL Workbench interface with two main panes. The top pane displays the SQL editor with 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

```

The bottom pane shows the Result Grid with one row of data:

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

The middle section shows the table definition for 'employee' with the following details:

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

Below the table definition, a trigger named 'employee_AFTER_UPDATE' is defined:

```

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 bottom pane shows the SQL editor with the following update statement:

```

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

```

The bottom result grid shows the updated data in 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.

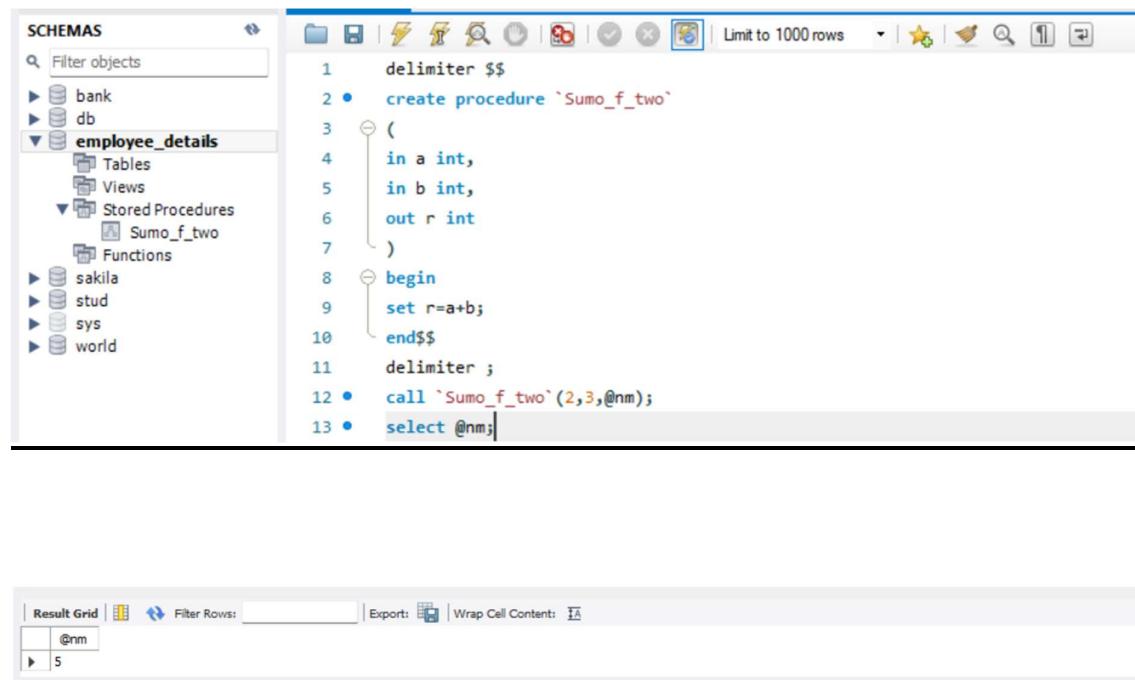
Experiment No.: 9

Aim: To familiarize with Stored functions and Procedure.

CO2: Apply PL/SQL for processing databases

Procedure

Create Stored Procedure Sumo_f_two



```

1 delimiter $$ 
2 • create procedure `Sumo_f_two` 
3 ( 
4     in a int, 
5     in b int, 
6     out r int 
7 ) 
8 begin 
9     set r=a+b; 
10 end$$ 
11 delimiter ; 
12 • call `Sumo_f_two` (2,3,@nm); 
13 • select @nm;

```

The Result Grid shows the output of the stored procedure call, which is 5.

Result

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

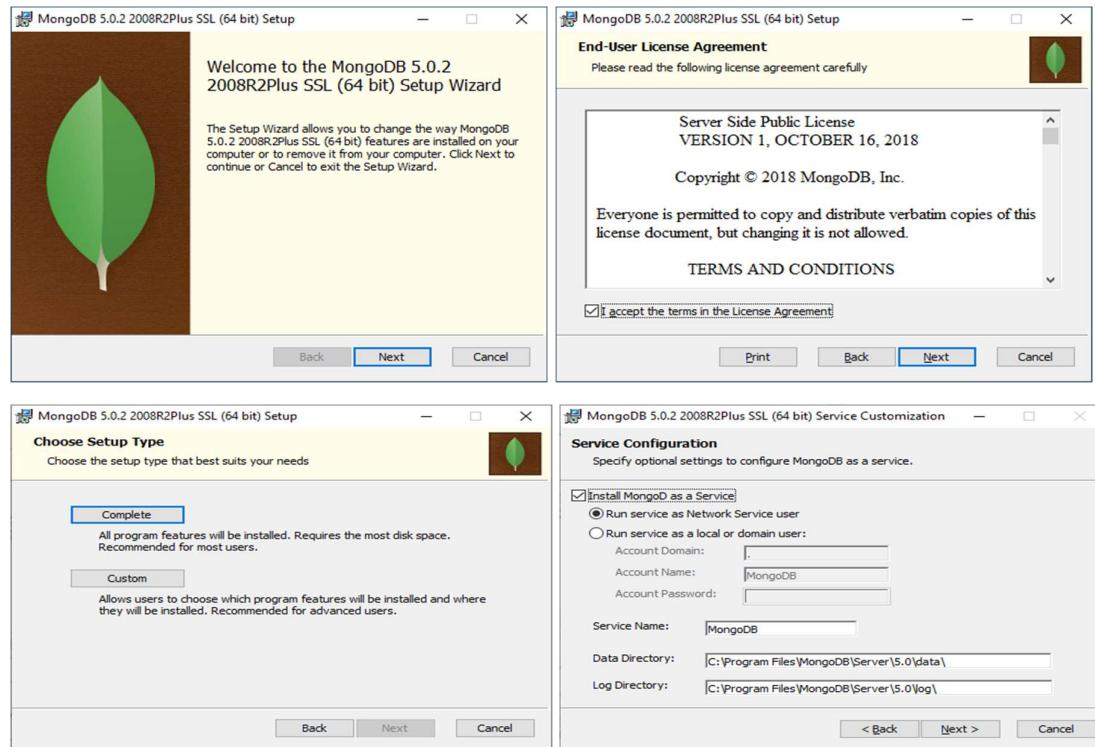
Experiment No.: 10

Aim: Understand the installation and configuration of NoSQL Databases: MongoDB.

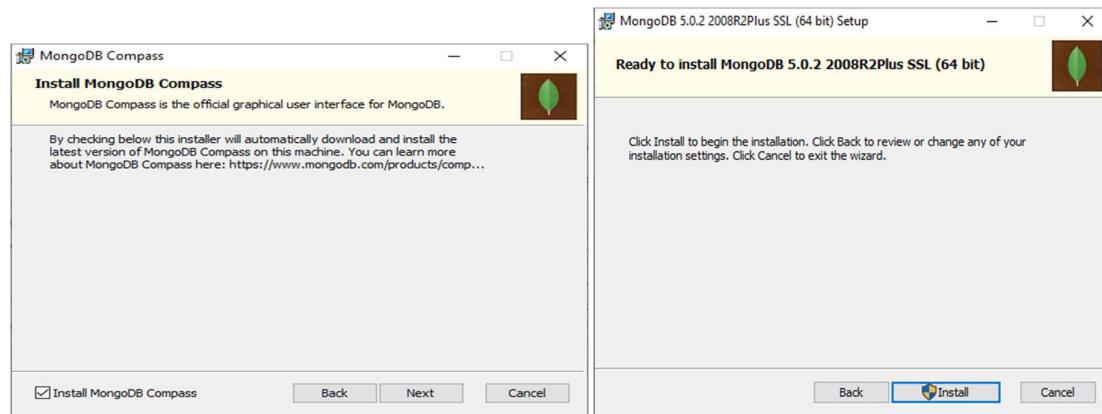
CO3: Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

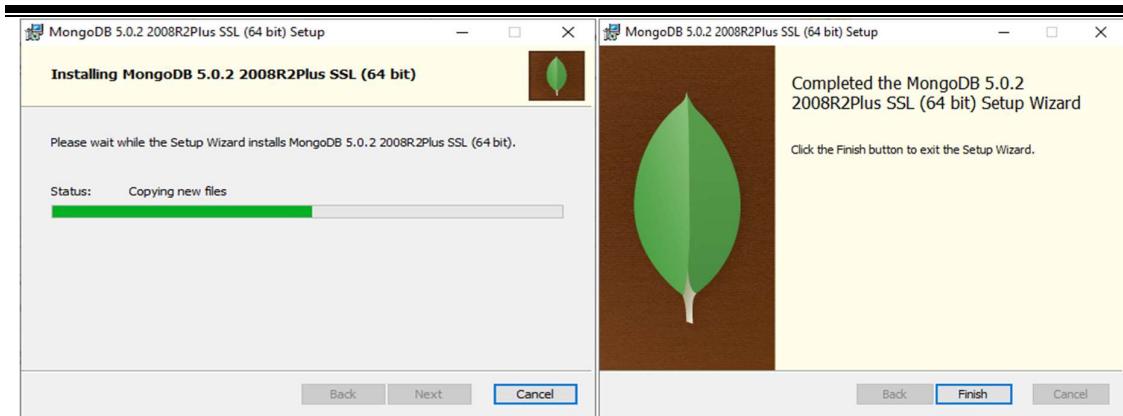
Procedure

1. Setup Wizard



2. Installing MongoDB Compass





3. MongoDB Compass Home Page

The screenshot displays the MongoDB Compass application and a terminal window. The Compass interface shows a connection to 'localhost:27017' with three databases listed: 'admin', 'config', and 'local'. The terminal window shows the output of the command 'C:\xampp\php>php -v', which displays the PHP version information.

```

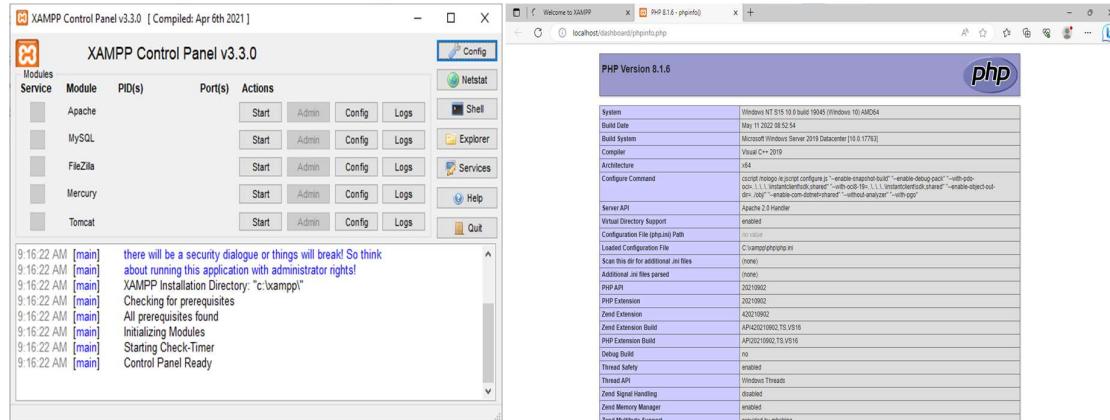
>_MONGOSH
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\xampp\php>php -v
PHP 8.1.6 (cli) (built: May 11 2022 08:55:59) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.1.6, Copyright (c) Zend Technologies

C:\xampp\php>

```

4. Configuring MongoDB to Connect to PHP



5. Installing MongoDB Package for PHP

My Drive - Google Drive x PECI : Package :: mongodb x +

pek.php.net/package/mongodb

Login Packages Support Bug

Search for in the Packages

Home News Documentation: Support

Downloads: Browse Packages Search Packages Download Statistics

Top Level :: Database :: mongodb

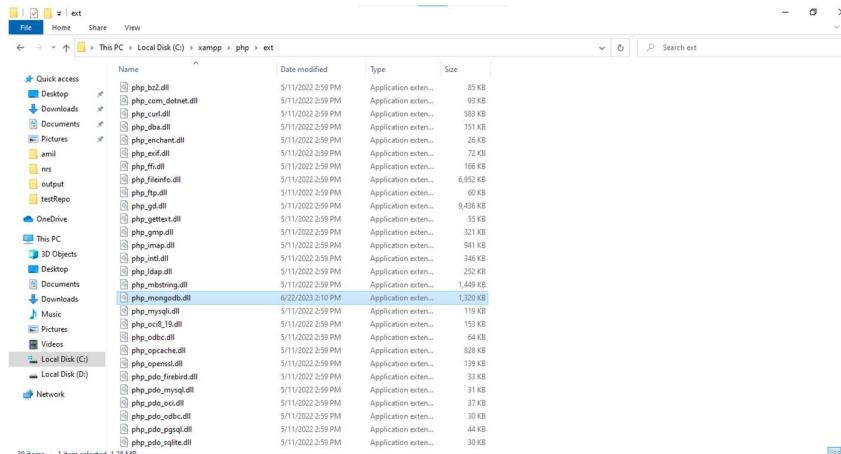
mongodb

Package Information	
Summary	MongoDB driver for PHP
Maintainers	Jeremy Mikola (lead) [details] Katherine Walker (developer) [details] André Braun (lead) [details] Dirk Rethans < dirk@php.net > (lead) [wishlist] [details] Hannes Magnusson < hbo@it.php.net > (lead) [details]
License	Apache License 2.0
Description	The purpose of this driver is to provide exceptionally thin glue between MongoDB and PHP, implementing only fundamental and performance-critical components necessary to build a fully-functional MongoDB driver.
Homepage	http://docs.mongodb.org/ecosystem/drivers/php/

[Latest Tarball] [Browse Source] [ChangeLog] [Package Bugs] [View Statistics] [View Documentation]

Available Releases				
Version	State	Release Date	Downloads	
1.16.1	stable	2023-06-22	mongodb-1.16.1.tgz (1862.9kB)	[ChangeLog]
1.16.0	stable	2023-06-22	mongodb-1.16.0.tgz (1521.9kB)	[ChangeLog]
1.15.3	stable	2023-05-12	mongodb-1.15.3.tgz (1701.8kB)	[ChangeLog]
1.15.2	stable	2023-04-21	mongodb-1.15.2.tgz (1702.1kB)	[ChangeLog]
1.15.1	stable	2023-02-09	mongodb-1.15.1.tgz (1701.4kB)	[ChangeLog]

6. Configuring Apache



```
File Edit Format View Help
;extension=imap
;extension=ldap
extension=mbstring
extension=exif      ; Must be after mbstring as it depends on it
extension=mcrypt
extension=oci8_12c ; Use with Oracle Database 12c Instant Client
extension=oci8_19 ; Use with Oracle Database 19 Instant Client
extension=odbc
extension=openssl
extension=gd
extension=gd_mbstring
extension=pdo
extension=pdo_oci
extension=pdo_odbc
extension=pdo_pgsql
extension=pdo_sqlite
extension=pgsql
extension=hhvm
extension=php_mongodb.dll

; The MongoDB data available in the PHP distribution must be installed.
; See https://www.php.net/manual/en/snmp.installation.php
;extension=snmp

;extension=soap
;extension=sockets
;extension=sodium
;extension=sqlite3
;extension=idn
;extension=xml

zend_extension=opcache

;;;;;;;;;;;;;;;;;;
; Module Settings ;
;;;;;;;;;;;;;;;;;;
asp tags=Off
```

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

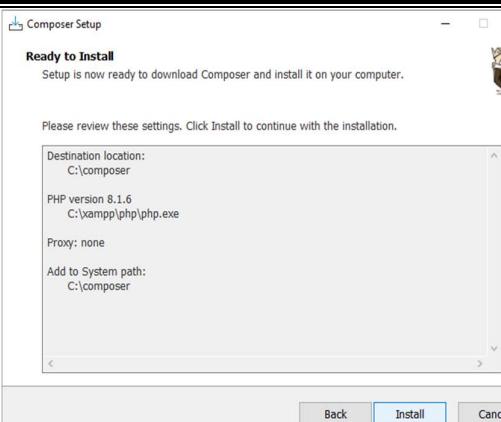
mongodb

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

mysqli

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

 <h3>Installation Options</h3> <p>Choose your installation type.</p> <p>Setup will install Composer to a fixed location for all users. This includes a Control Panel uninstaller and is the recommended option. Click Next to use it.</p> <p><input type="checkbox"/> Developer mode</p> <p>Take control and just install Composer. An uninstaller will not be included.</p>	 <h3>Select Destination Location</h3> <p>Where should Composer be installed?</p> <p> Setup will install Composer into the following folder.</p> <p>To continue, click Next. If you would like to select a different folder, click Browse.</p> <p><input type="text" value="C:\composer"/> <input type="button" value="Browse..."/></p>
<input type="button" value="Next"/> <input type="button" value="Cancel"/>	
<input type="button" value="Back"/> <input type="button" value="Next"/> <input type="button" value="Cancel"/>	



Ready to Install
Setup is now ready to download Composer and install it on your computer.

Please review these settings. Click **Install** to continue with the installation.

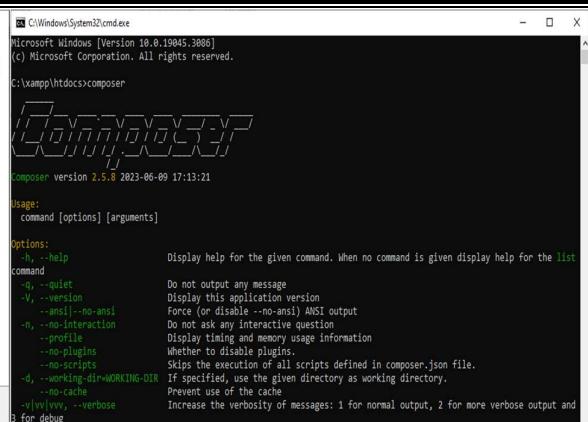
Destination location:
C:\composer

PHP version 8.1.6
C:\xampp\php\php.exe

Proxy: none

Add to System path:
C:\composer

Back **Install** **Cancel**



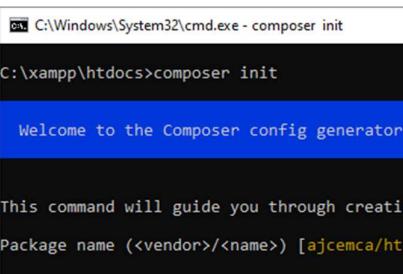
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\xampp\htdocs>composer
[Progress Bar]

Composer version 2.5.0 2023-06-09 17:13:21

Usage: command [options] [arguments]

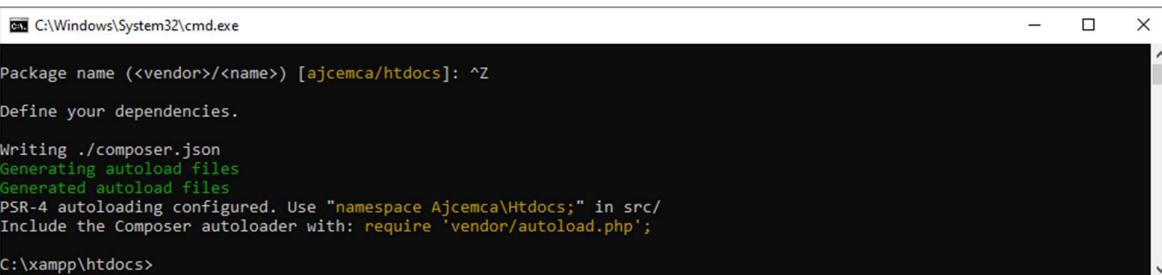
Options:
  -h, --help           Display help for the given command. When no command is given display help for the list command
  -q, --quiet          Do not output any message
  -V, --version         Display this application version
  --ansi|--no-ansi     Force (or disable --no-ansi) ANSI output
  -n, --no-interaction  Do not ask any interactive question
  --profile            Display timing and memory usage information
  --no-plugins          Whether to enable plugins.
  --no-scripts          Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=DIR If specified, use the given directory as working directory.
  --ultra-vvv, --verbose Prevent use of the cache.
  -3 for debug          Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
```

```
C:\Windows\System32\cmd.exe - composer init
C:\xampp\htdocs>composer init
Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

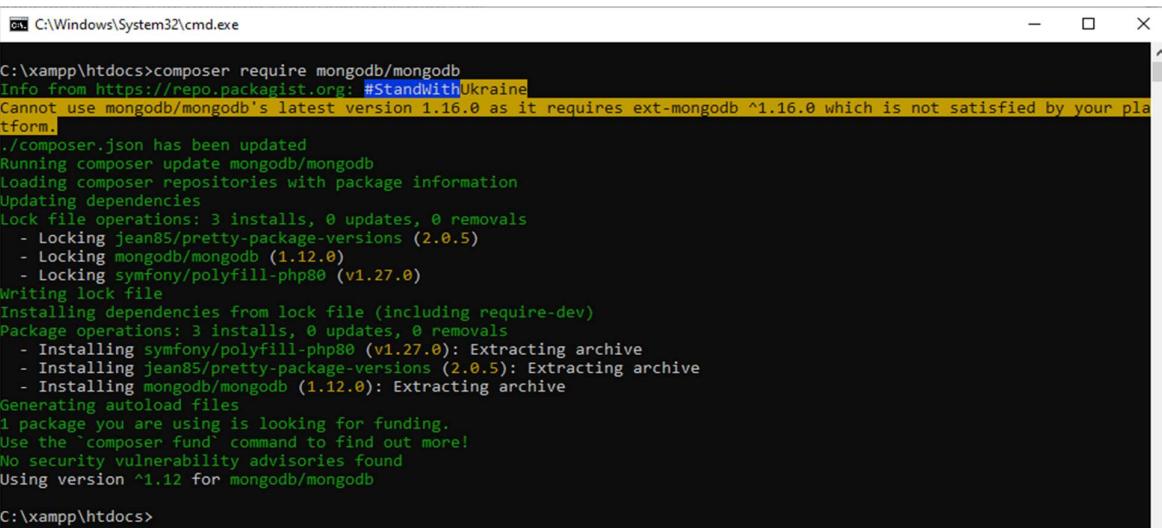
Package name (<vendor>/<name>) [ajcemca/htdocs]:
```

```
C:\Windows\System32\cmd.exe
Package name (<vendor>/<name>) [ajcemca/htdocs]: ^Z
Define your dependencies.

Writing ./composer.json
Generating autoload files
Generated autoload files
PSR-4 autoloading configured. Use "namespace Ajcemca\Htdocs;" in src/
Include the Composer autoloader with: require 'vendor/autoload.php';

C:\xampp\htdocs>
```

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

C:\xampp\htdocs>
```

Result

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

Experiment No.: 11

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

CO4: Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure

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

Output Screenshot

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

MongoDB Compass - localhost:27017/Rainbow.Colors

_id	ObjectId	color	Type
1	ObjectId('64eb2dfb4d739bb68db0')	"Violet"	document
2	ObjectId('64eb2dfb4d739bb68db0')	"Indigo"	document
3	ObjectId('64eb2dfb4d739bb68db0')	"Blue"	document
4	ObjectId('64eb2dfb4d739bb68db0')	"Green"	document
5	ObjectId('64eb2dfb4d739bb68db0')	"Yellow"	document
6	ObjectId('64eb2dfb4d739bb68db0')	"Orange"	document
7	ObjectId('64eb2dfb4d739bb68db0')	"Red"	document

Result

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

Experiment No.: 12

Aim: Implementing CRUD operations using PHP-MongoDB.

CO4: Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure

connection.php

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

register.php

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

edit.php

```

<?php
include_once("connection.php");
if(isset($_POST['update'])){
echo $id = $_GET['id'];
$name = $_POST['name'];
$place = $_POST['place'];
$email = $_POST['email'];
$db->Profile->updateOne(['_id' => new MongoDB\BSON\ObjectId($id)], ['$set' =>
['name'=> $name, 'place' => $place, 'email'=>$email,]]);
header("Location: select.php");
}?
<html><head>
<title>Registration Form</title>
</head>
<?php
$id = $_GET['id'];
$result = $db->Profile->findOne(['_id' => new MongoDB\BSON\ObjectId($id)]);
$name = $result['name'];
$place = $result['place'];
$email = $result['email'];
?>
<body>
<h2>Profile Data</h2>
<form action="#" method="post">
<?php $id = $_GET['id'];?>
Name:<input type="text" name="name" value="<?php echo $name; ?>" required><br>
Place:<input type="place" name="place" value="<?php echo $place; ?>" required><br>
Email:<input type="email" name="email" value="<?php echo $email; ?>" required><br>
<input type="submit" name="update" class="button" value="Update">

```

```

</form></body>
</html>

select.php

<center><br><form action=register.php>
<input type="submit" value=register></form>
<form action=select.php>
<input type="submit" value=View>
</form><br>
<?php
include_once("connection.php");
$result=$collection->find();
?>
<html>
<head><title>select.php</title>
</head><body>
<table border="0" >
<tr>
<th align="center" bgcolor=green><font color=white size=4><b>SlNo.</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Name</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Email</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Place</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Delete</b></th>
<th align="center" bgcolor=green><font color=white size=4><b>Edit</b></th>
</tr><?php
$no=1;
foreach($result as $res){
?> <tr>
<td><font color=DeepSkyBlue size=4><?php echo $no;?>
<td><font color=steelblue size=4><?php echo $res['name'];?></td>
<td><font color=pink size=4><?php echo $res['email'];?></td>
<td><font color=RosyBrown size=4><?php echo $res['place'];?></td>

```

```

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

```

insert.php

```

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

```

delete.php

```

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

```

Output Screenshot

Connection.php



Register.php

A screenshot of a Microsoft Edge browser window. The address bar shows 'localhost/Crud/register.php'. The page title is 'Registration form'. It contains a form titled 'Registrarion Form' with fields for Name (Sara), Email (Sara@gmail.com), and Place (Kanirapally), followed by a 'Register' button. The browser interface includes standard controls.

Select.php

A screenshot of a Microsoft Edge browser window. The address bar shows 'localhost/Crud/select.php'. The page title is 'select.php'. It features a 'register' button and a 'View' button. Below these are two tables: one for profile data (Name: riya, Place: pta, Email: riya@gmail.com) and one for a list of users with columns: SNo, Name, Email, Place, Delete, and Edit. The user list table data is as follows:

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

Edit.php

A screenshot of a Microsoft Edge browser window. The address bar shows 'localhost/Crud/edit.php?id=64a2ad9524e4d59084075374'. The page title is 'Registration Form'. It displays 'Profile Data' with fields for Name (riya), Place (pta), and Email (riya@gmail.com), and a 'Update' button. The browser interface includes standard controls.

A screenshot of a Microsoft Edge browser window. The address bar shows 'localhost/Crud/select.php'. The page title is 'select.php'. It features a 'register' button and a 'View' button. Below these are two tables: one for profile data (Name: riya, Place: pta, Email: riya@gmail.com) and one for a list of users with columns: SNo, Name, Email, Place, Delete, and Edit. The user list table data is as follows:

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

Delete.php



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



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

MongoDB Compass - localhost:27017/admin.Profile

localhost:27017

admin.Profile

_id	ObjectID	name	email	place
1	ObjectID("6442ad99524e4d590040")	"Riya"	"riya@gmail.com"	"Mukkututhara"
2	ObjectID("6409b0ddfb920840e670")	"Jenny "	"jenny@gmail.com"	"Vecchochira"
3	ObjectID("640551d4d739006800")	"Tinu"	"tinu@gmail.com"	"Mundakayam"
4	ObjectID("64059284d739006800")	"Sara"	"Sara@gmail.com"	"Kanjirapally"

Result

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

Experiment No.: 13

Aim: Build sample collections/documents to perform the shell queries.

CO5: Understand the basic storage architecture of distributed file systems.

Procedure

1. Create/Use a database

> use expmongo

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

2. Display current database

>db

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

3. Create a collection

>db.createCollection("actors")

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

4. Insert data into the collection

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

5. Display documents in collection

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

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

6. Display documents in collection

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

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

7. Display

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

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

8. Display

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

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

9. Display

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

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

10. Display

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

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

11. Display

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

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

12. Display

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

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

13. Display

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

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

14. Display

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

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

15. Display

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

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

16. Display

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

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

17. Display

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

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

18. Display

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

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

19. Display

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

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

20. Display

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

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

21. Display

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

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

Result

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

Experiment No.: 14

Aim: To familiarize with indexing in MongoDB.

CO5: Understand the basic storage architecture of distributed file systems.

Procedure

1. Input data –

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

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

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

Example 1 – Create ascending index on a field

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

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

Example 2 – Create descending index on a field

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

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

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

2.Specify the name to the Index

Example – Create index with the index name

```
db.products.createIndex(
{ item: 1, quantity: -1 } ,
{ name: "query for inventory" }
)

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

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

3.dropIndex() Method Example

Example 1 – Drop index by index document

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

Example 2 – Drop index by index name

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

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

Result

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

Experiment No.: 15

Aim: Implementation of Replica Set on MongoDB

CO5: Understand the basic storage architecture of distributed file systems.

Procedure

1. Creating 3 Replica set

Name	Date modified	Type	Size
rs1	7/26/2023 11:21 AM	File folder	
rs2	7/26/2023 11:30 AM	File folder	
rs3	7/26/2023 11:30 AM	File folder	

```
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

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

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("16a80073-b42e-4e43-a85c-64128346f667") }
MongoDB server version: 5.0.2
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved performance 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:
 2023-07-26T11:10:33.628Z-05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
 2023-07-26T11:10:33.628Z-05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <addr> 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. Configuration of Replica set

```
> 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(1690350364, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  },
  "operationTime" : Timestamp(1690350364, 1)
```

3. Status

```
datascience:SECONDARY> rs.status()
{
    "set" : "datascience",
    "date" : ISODate("2023-07-26T05:46:19.076Z"),
    "myState" : 1,
    "term" : NumberLong(1),
    "syncSourceHost" : "",
    "syncSourceId" : -1,
    "heartbeatIntervalMillis" : NumberLong(2000),
    "majorityVoteCount" : 2,
    "writeMajorityCount" : 2,
    "votingMembersCount" : 3,
    "writableVotingMembersCount" : 3,
    "optimes" : {
        "lastCommittedOpTime" : {
            "ts" : Timestamp(1690350377, 1),
            "t" : NumberLong(1)
        },
        "lastCommittedWallTime" : ISODate("2023-07-26T05:46:17.200Z"),
        "readConcernMajorityOpTime" : {
            "ts" : Timestamp(1690350377, 1),
            "t" : NumberLong(1)
        },
        "appliedOpTime" : {
            "ts" : Timestamp(1690350377, 1),
            "t" : NumberLong(1)
        },
        "durableOpTime" : {
            "ts" : Timestamp(1690350377, 1),
            "t" : NumberLong(1)
        },
        "lastAppliedWallTime" : ISODate("2023-07-26T05:46:17.200Z"),
        "lastDurableWallTime" : ISODate("2023-07-26T05:46:17.200Z")
    },
    "lastStableRecoveryTimestamp" : Timestamp(1690350375, 3),
    "electionCandidateMetrics" : {
        "lastElectionReason" : "electionTimeout",
        "lastElectionDate" : ISODate("2023-07-26T05:46:15.010Z"),
        "electionTerm" : NumberLong(1),
        "lastCommittedOpTimeAtElection" : {
            "ts" : Timestamp(0, 0),
            "t" : NumberLong(-1)
        },
        "lastSeenOpTimeAtElection" : {
            "ts" : Timestamp(1690350364, 1),
            "t" : NumberLong(-1)
        },
        "numVotesNeeded" : 2,
        "priorityAtElection" : 1,
        "electionTimeoutMillis" : NumberLong(10000),
        "numCatchUpOps" : NumberLong(0),
        "newTermStartDate" : ISODate("2023-07-26T05:46:15.204Z"),
        "wMajorityWriteAvailabilityDate" : ISODate("2023-07-26T05:46:15.839Z")
    },
    "members" : [
        {
            "_id" : 0,
            "name" : "localhost:27018",
            "health" : 1,
            "state" : 1,
            "stateStr" : "PRIMARY",
            "uptime" : 347,
            "optime" : {
                "ts" : Timestamp(1690350377, 1),
                "t" : NumberLong(1)
            },
            "optimeDate" : ISODate("2023-07-26T05:46:17Z"),
            "syncSourceHost" : "",
            "syncSourceId" : -1,
            "infoMessage" : "",
            "electionTime" : Timestamp(1690350375, 1),
            "electionDate" : ISODate("2023-07-26T05:46:15Z"),
            "configVersion" : 1,
            "configTerm" : 1,
            "self" : true,
            "lastHeartbeatMessage" : ""
        },
        {
            "_id" : 1,
            "name" : "localhost:27019",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 14,
            "optime" : {
                "ts" : Timestamp(1690350375, 6),
                "t" : NumberLong(1)
            }
        }
    ]
}
```

Primary

```
datascience:PRIMARY> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
datascience:PRIMARY> use db1;
switched to db db1
datascience:PRIMARY> db.student.insert({name:"johan"})
WriteResult({ "nInserted" : 1 })
datascience:PRIMARY> use admin
switched to db admin
datascience:PRIMARY> db.shutdownServer()
server should be down...
```

```
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("42943329-b665-4cb5-a195-9342fd7dcde2") }
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:
 2023-07-26T11:11:03.582+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
 2023-07-26T11:11:03.583+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <addr>
 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()
```

Secondary

```
> rs.secondaryOk()
datascience:SECONDARY> show dbs;
admin   0.000GB
config  0.000GB
db1     0.000GB
local   0.000GB
datascience:SECONDARY> use db1;
switched to db db1
datascience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c0b3ae5e25987f931f5dee"), "name" : "johan" }
datascience:SECONDARY> rs.status()
{
  "set" : "datascience",
  "date" : ISODate("2023-07-26T05:51:36.173Z"),
  "myState" : 1,
  "term" : NumberLong(2),
  "syncSourceHost" : "",
  "syncSourceId" : -1,
  "heartbeatIntervalMillis" : NumberLong(2000),
  "majorityVoteCount" : 2,
  "writeMajorityCount" : 2,
  "votingMembersCount" : 3,
  "writableVotingMembersCount" : 3,
  "optimes" : {
    "lastCommittedOpTime" : {
      "ts" : Timestamp(1690350694, 1),
      "t" : NumberLong(2)
    },
    "lastCommittedWallTime" : ISODate("2023-07-26T05:51:34.284Z"),
    "readConcernMajorityOpTime" : {
      "ts" : Timestamp(1690350694, 1),
      "t" : NumberLong(2)
    },
    "appliedOpTime" : {
      "ts" : Timestamp(1690350694, 1),
      "t" : NumberLong(2)
    },
    "durableOpTime" : {
      "ts" : Timestamp(1690350694, 1),
      "t" : NumberLong(2)
    },
    "lastAppliedWallTime" : ISODate("2023-07-26T05:51:34.284Z"),
    "lastDurableWallTime" : ISODate("2023-07-26T05:51:34.284Z")
  }
}
```

```
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("13ae6ea9-9f7f-4310-9b37-7630c0c671b3") }
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:
2023-07-26T11:11:33.309+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2023-07-26T11:11:33.310+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <addr
ess> 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()
...
> rs.secondaryOk()
dataScience:SECONDARY>
```

Result

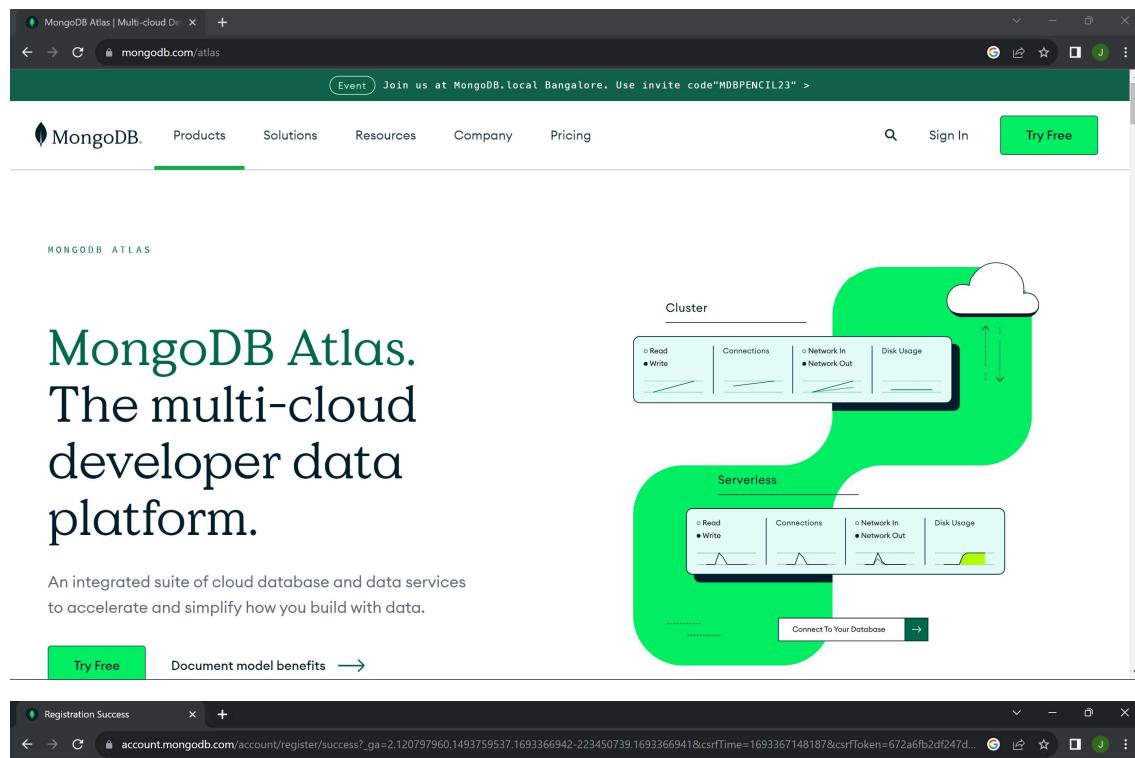
The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

Experiment No.: 16

Aim: Usage of Cloud Storage Management Systems: MongoDB Atlas.

CO6: Design and deployment of NoSQL databases with real time requirements.

Procedure



Welcome!

Use your account to deploy a **cloud database** with **MongoDB Atlas** and contact **Support**.



M10 \$0.08/hour

For production applications with sophisticated workload requirements.

STORAGE 10 GB	RAM 2 GB	vCPU 2 vCPUs
------------------	-------------	-----------------

SERVERLESS \$0.10/1M reads

For application development and testing, or workloads with variable traffic.

STORAGE Up to 1TB	RAM Auto-scale	vCPU Auto-scale
----------------------	-------------------	--------------------

M0 FREE

For learning and exploring MongoDB in a cloud environment.

STORAGE 512 MB	RAM Shared	vCPU Shared
-------------------	---------------	----------------

Provider: **AWS** Google Cloud Azure

\$0.08/hour

Create

Pay-as-you-go! You will be billed hourly and can terminate your cluster anytime. Excludes variable data transfer, backup, and taxes.

I'll deploy my database later

[Access Advanced Configuration](#)

Provider: **AWS** Google Cloud Azure

Region: **N. Virginia (us-east-1) ★**

Name: Cluster0

Tag (optional): Create your first tag to categorize and label your resources; more tags can be added later. [Learn more](#).

Create

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

I'll deploy my database later

[Access Advanced Configuration](#)

The screenshot shows the MongoDB Atlas interface for a project named "JENNY'S ORG - 2023-08-30 > PROJECT 0". The left sidebar is open, showing options like Overview, Deployment, Services (Device Sync, Triggers, Data API, Data Federation, Search, Stream Processing), Security (Quickstart, Backup, Database Access), and Monitoring (Cloud Monitoring, Metrics). The "Data Services" tab is selected.

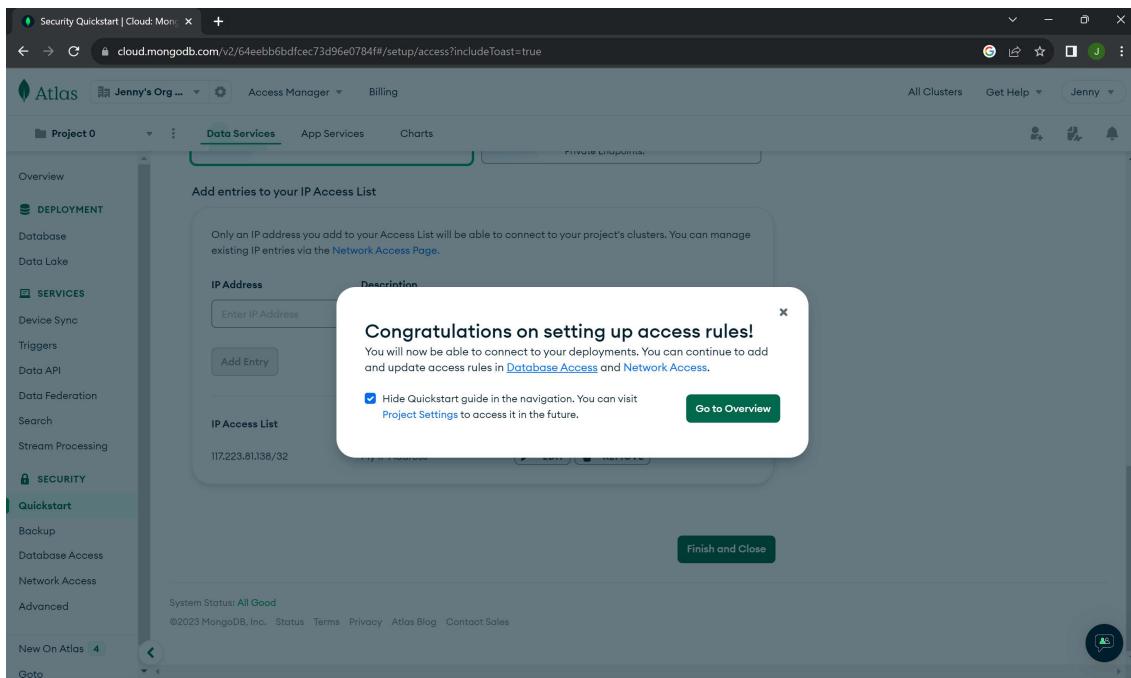
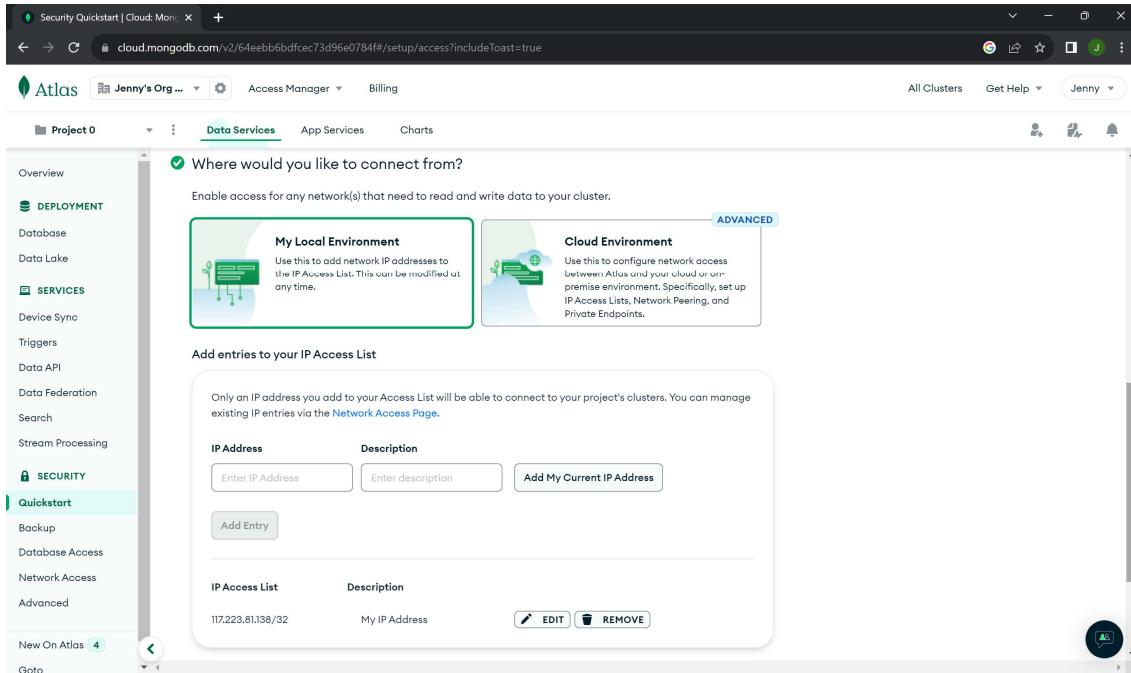
The main content area is titled "Security Quickstart". It asks, "How would you like to authenticate your connection?", with two options: "Username and Password" (selected) and "Certificate". A note says, "We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information." Below this, it says, "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." A "Username" field contains "jennyjohnson" and a "Password" field contains "Rgo5QGeSK9uT3V80". There is also an "Autogenerate Secure Password" button and a "Copy" button.

A progress bar at the bottom indicates "MO Cluster Provisioning..." with the message "We're putting final touches on your cluster." The status shows "New On Atlas 4".

This screenshot shows the continuation of the security setup. The "Create a database user using a username and password" section is still visible, with the "Username" field set to "jennyjohnson" and the "Password" field containing "Rgo5QGeSK9uT3V80". A "Create User" button is present.

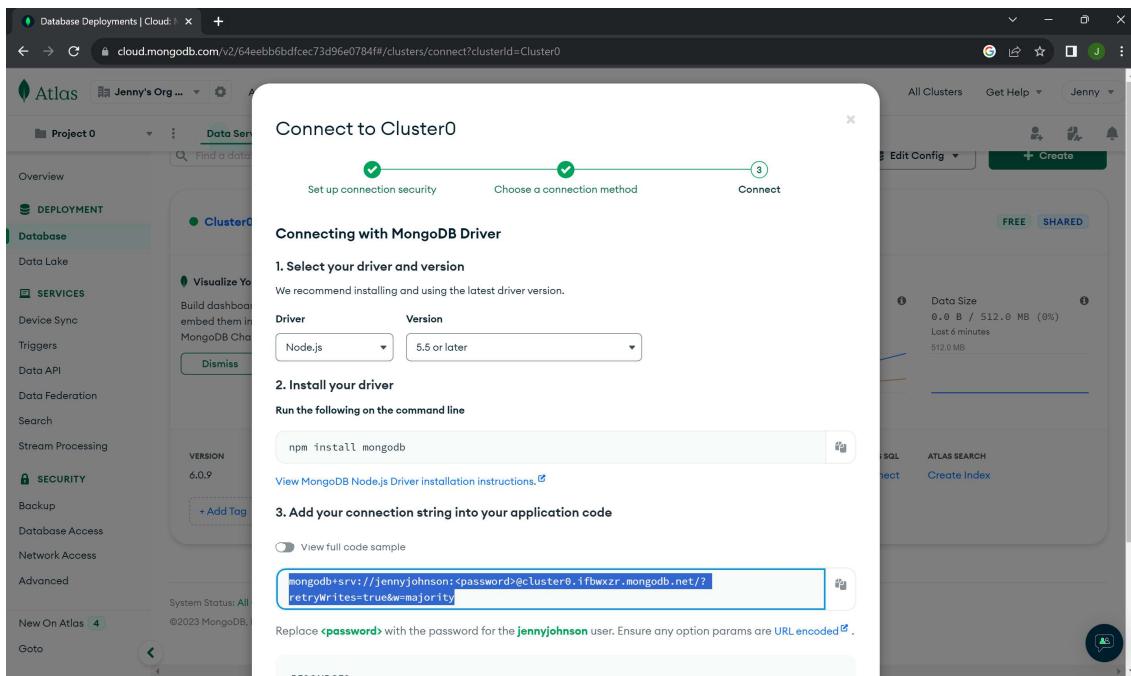
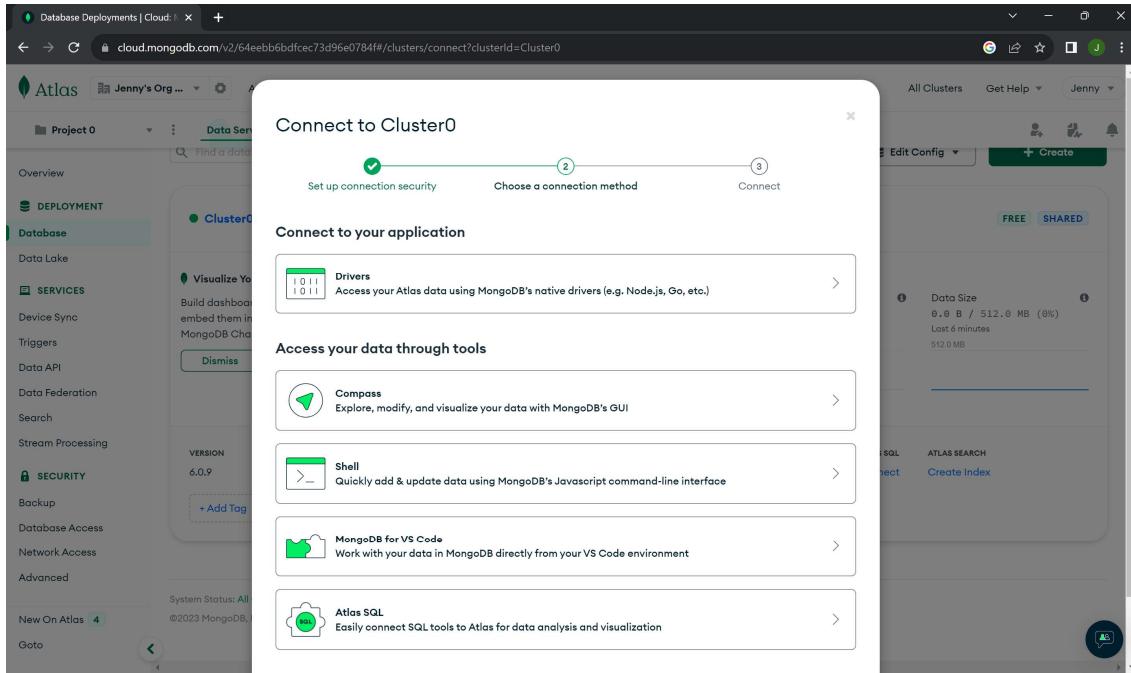
The next section, "Where would you like to connect from?", asks to "Enable access for any network(s) that need to read and write data to your cluster." It shows two options: "My Local Environment" (with a note about adding IP addresses to the IP Access List) and "Cloud Environment" (with a note about configuring network access between Atlas and a cloud or on-premise environment). An "ADVANCED" link is available for the Cloud Environment section.

At the bottom, there is a note: "Add entries to your IP Access List".



The screenshot shows the MongoDB Atlas interface for a project named 'JENNY'S ORG'. The left sidebar includes sections for Deployment, Services (Device Sync, Triggers, Data API, Data Federation, Search, Stream Processing), Security (Backup, Database Access, Network Access, Advanced), and a 'New On Atlas' section. The main 'Overview' tab displays 'Database Deployments' for 'Cluster0', showing options to 'CONNECT' or 'EDIT CONFIGURATION'. It also features 'Add Data', 'Load Sample Data', and 'Data Modeling Templates' buttons. A 'Resources Center' sidebar provides links to PHP, LEARN, and I'M JUST EXPLORING, along with links to Get Started with Atlas, Developer Center, and Ask the MongoDB Community. A 'New On Atlas' callout highlights recent feature enhancements.

This screenshot shows the 'Database Deployments' page for 'Cluster0'. The top navigation bar includes 'Edit Config' and a '+ Create' button. The main area displays monitoring metrics like 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. Below this is a table with columns for VERSION, REGION, CLUSTER TIER, TYPE, BACKUPS, LINKED APP SERVICES, ATLAS SQL, and ATLAS SEARCH. The table shows data for a version 6.0.9 cluster in AWS/N. Virginia (us-east-1) with a M0 Sandbox tier, a Replica Set - 3 nodes type, inactive backups, and linked services for Connect, Atlas SQL, and Atlas Search.



The screenshot shows the MongoDB App Services interface. On the left, a sidebar lists various services and management options under categories like NO ENVIRONMENT, DATA ACCESS, BUILD, and MANAGE. The main area is titled "Build better apps faster with App Services" and features a section titled "Start with an app template". It displays several template cards:

- Build your own App**: Set up your own app services to fit your development needs.
- Real-time Sync**: Mobile app using a Realm SDK to sync data to your backend. Icons include React Native, Expo, Capacitor, and Cordova.
- Manage Database Views**: Event-driven Database Trigger template to update a view in a separate collection.
- React.js + Realm Web Boilerplate**: Todo web app using the Realm Web SDK.
- GraphQL + React App Boilerplate**: Todo web app using the GraphQL API.

This screenshot shows the "Build better apps faster with App Services" interface with a workflow overlay. The steps are:

- Select a Template**: Using App Services reduces the code you need to write. Get started quicker with Templates. Options include React.js + Realm Web Boilerplate.
- Link your Data Source**: Template Starter Applications let you build on top of data in a MongoDB Atlas cluster. Options include Cluster0.
- Name your Application**: This name will be used internally and cannot be changed later. Options include Application-0.
- App Deployment Model**: You can change this later in your Application Settings. View Docs. Options include Single Region • USA (us-east-1) • AWS | Region chosen based on your cluster's region Change.

Application-0

App ID: application-0-wrdmm AWS • Virginia (us-east-1)

Welcome to Your Application

This ready-to-use app has the following services set up for you:

- Authentication Email/Password
- JSON Schema for your collection
- Permissions

</> Pull front-end code

Metrics

Total App Usage Aug 01 - Aug 31, Free Tier resets in 2 days. [Learn more](#) Last Updated: Aug 30, 2023, 9:31:54 AM GMT+5:30

Requests	Data Transfer	Compute Runtime	Sync Runtime
0 / IM FREE TIER USED	0.00 / 10 GB FREE TIER USED	0.00 / 500 Hr FREE TIER USED	0.00 / 10K Hr FREE TIER USED

Application-0

Total Users Requests (1hr) 0
0 Success 0 Fail

This Month Last modified: 0 minutes ago

Requests	0
Data Transfer (GB)	0.00
Compute Runtime (Hr)	0.00
Sync Runtime (Hr)	0.00

Result

The program was executed and the result was successfully obtained. Thus, CO6 was obtained.