

20MCA134 – ADVANCED DBMS LAB

Lab Report Submitted By

ABDUL RAOOF V A

Reg. No.: AJC22MCA-2001

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. Kananthoor, Kananthoor, 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 **ABDUL RAOOF V A (AJC22MCA-2001)** 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. Amal K Jose
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 extent 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 extent you are able to apply PL/SQL for processing databases	Excellent/Very Good/Good/Fair/Poor
CO3	To what extent you are able to compare relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases	Excellent/Very Good/Good/Fair/Poor
CO4	To what extent you are able to 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 extent you are able to design and deployment of NoSQL Good/Good/Fair/Poor databases with real time requirements.	Excellent/Very

CONTENT

Sl. No.	Experiment	Date	CO	Page No.
1	Familiarization DDL Commands	7-03-2023	CO1	1
2	Familiarization of DML Commands	13-03-2023	CO1	4
3	Familiarization DCL & TCL Commands	14-03-2023	CO1	13
4	Accessing and Optimizing databases using SELECT, Filtering using WHERE and DATE, NUMBER and CHARACTER FUNCTIONS	27-03-2023	CO1	15
5	Optimizing database using Aggregate Functions	28-03-2023	CO1	18
6	Optimizing database using Set Operations	03-04-2023	CO1	22
7	Accessing database using HAVING, GROUP BY, ORDER BY Clauses	04-04-2023	CO1	25
8	Optimizing database using Join Query	11-04-2023	CO1	28
9	Accessing database with Subquery	18-04-2023	CO1	31
10	Accessing database with View Table	29-04-2023	CO1	33
11	Familiarization of PL/SQL Program-Functions	12-06-2023	CO2	36
12	Familiarization of PL/SQL Program- Stored Procedures	20-06-2023	CO2	38
13	Familiarization of PL/SQL Program- Trigger	20-06-2023	CO2	41
14	Installation and configuration of NoSQL Database - MongoDB	27-06-2023	CO3	44
15	NoSQL Operations - Build sample collections/documents to perform query operations	10-07-2023	CO4	50
16	Build sample collections / documents to perform the shell commands	11-07-2023	CO5	54
17	Develop sample applications using any of the front-end tools and NoSQL	24-07-2023	CO6	56

DDL COMMANDS

Experiment No.: 1

07-03-2023

Aim

To Familiarization DDL Commands

CO

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

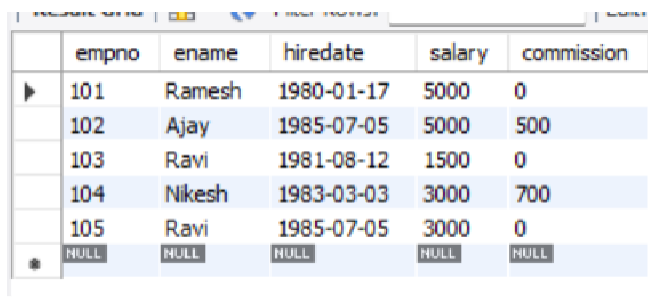
Procedure

1. Create a table emp with attributes empno number (4) as primary key, ename char (10), hiredate, salary, commission
insert 5 rows of data

```
101  Ramesh 17-Jan-1980 5000;
102  Ajay   05-Jul-1985   5000  500;
103  Ravi   12-Aug-1981   1500;
104  Nikesh 03-Mar-1983   3000  700;
105  Ravi   05-Jul-1985   3000;
```

Query and Output:

- INSERT INTO emp VALUES (101,'Ramesh','1980-01-17',5000,0),
(102,'Ajay','1985-07-05',5000,500),
(103,'Ravi','1981-08-12',1500,0),
(104,'Nikesh','1983-03-03',3000,700),
(105,'Ravi','1985-07-05',3000,0);



	empno	ename	hiredate	salary	commission
▶	101	Ramesh	1980-01-17	5000	0
	102	Ajay	1985-07-05	5000	500
	103	Ravi	1981-08-12	1500	0
	104	Nikesh	1983-03-03	3000	700
	105	Ravi	1985-07-05	3000	0
✱	NULL	NULL	NULL	NULL	NULL

2. Modifying the structure of tables

a. Add new columns: sal number(7,2)

Query and Output:

- ALTER TABLE emp ADD sal INT

	empno	ename	hiredate	salary	commission	sal
▶	101	Ramesh	1980-01-17	5000	0	NULL
	102	Ajay	1985-07-05	5000	500	NULL
	103	Ravi	1981-08-12	1500	0	NULL
	104	Nikesh	1983-03-03	3000	700	NULL
	105	Ravi	1985-07-05	3000	0	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

b. Dropping a column from a table: sal

Query and Output:

- ALTER TABLE emp DROP COLUMN sal

	empno	ename	hiredate	salary	commission
▶	101	Ramesh	1980-01-17	5000	0
	102	Ajay	1985-07-05	5000	500
	103	Ravi	1981-08-12	1500	0
	104	Nikesh	1983-03-03	3000	700
	105	Ravi	1985-07-05	3000	0
*	NULL	NULL	NULL	NULL	NULL

c. Modifying existing column: ename varchar2(15)

Query and Output:

- ALTER TABLE emp MODIFY ename VARCHAR(15)

	empno	ename	hiredate	salary	commission
▶	101	Ramesh	1980-01-17	5000	0
	102	Ajay	1985-07-05	5000	500
	103	Ravi	1981-08-12	1500	0
	104	Nikesh	1983-03-03	3000	700
	105	Ravi	1985-07-05	3000	0
*	NULL	NULL	NULL	NULL	NULL

d. Renaming the tables: emp to emp1

Query and Output:

- RENAME TABLE emp TO emp1
SELECT * FROM dbms.emp1;

	empno	ename	hiredate	salary	commission
▶	101	Ramesh	1980-01-17	5000	0
	102	Ajay	1985-07-05	5000	500
	103	Ravi	1981-08-12	1500	0
	104	Nikesh	1983-03-03	3000	700
	105	Ravi	1985-07-05	3000	0
*	NULL	NULL	NULL	NULL	NULL

e. truncating the tables: emp1

Query and Output:

- TRUNCATE TABLE emp1

	empno	ename	hiredate	salary	commission
*	NULL	NULL	NULL	NULL	NULL

f. Destroying tables: emp

Query and Output:

- DROP TABLE EMP1

3. Create a table stud with sname varchar2(20) primary key, rollno number(10) not null, dob date not null

Query and Output:

- CREATE TABLE stud (sname VARCHAR(20) PRIMARY KEY, rollno INT NOT NULL, dob DATE NOT NULL);



	sname	rollno	dob
	NULL	NULL	NULL

4. Create a table student as regno number (6), mark number (3) check constraint (mark >=0 and mark <=100));

In table student add check constraint(length(regno)<=4))

Query and Output:

- CREATE TABLE student (regno INT, mark INT CHECK (mark >= 0 AND mark <= 100));



regno	mark
-------	------

ALTER TABLE student ADD CONSTRAINT regno_length CHECK (LENGTH(regno) <= 4);

5. Create a table cust with (custid number (6) constraint unique, name char (10))

Query and Output:

- CREATE TABLE cust (custid int unique, name CHAR(10));




custid	name
--------	------

6. Refer the table “stud” in table “student”

Query and Output:

- ALTER TABLE student ADD COLUMN sname VARCHAR(20) REFERENCES stud(sname);



regno	mark	sname
-------	------	-------

Result

The query was executed and the output was successfully obtained.

DML COMMANDS

Experiment No.: 2

13-03-2023

Aim

Familiarization of DML Commands

CO

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

Create the following Tables

DEPOSIT

ACTNO VARCHAR2(5) PRIMARY KEY, FIRST LETTER MUST START WITH 'D',
 CNAME VARCHAR2(15) FOREIGN KEY REFERENCES CUSTOMER,
 BNAME VARCHAR2(20) FOREIGN KEY REFERENCES BRANCH,
 AMOUNT NUMBER (8,2) NOT NULL, CANNOT BE 0, ADATE DATE

Query and Output:

- CREATE TABLE DEPOSIT(ACTNO VARCHAR(5) PRIMARY KEY Check (ACTNO LIKE 'D%'),CNAME VARCHAR(15) ,FOREIGN KEY (CNAME) REFERENCES CUSTOMER(CNAME),BNAME VARCHAR(20) ,FOREIGN KEY(BNAME) REFERENCES BRANCH (BNAME),AMOUNT decimal(8,2) NOT NULL Check(AMOUNT !=0),ADATE DATE);



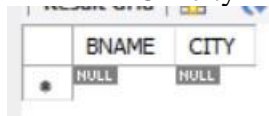
	ACTNO	CNAME	BNAME	AMOUNT	ADATE
*	NULL	NULL	NULL	NULL	NULL

BRANCH

BNAME VARCHAR2(20) PRIMARY KEY
 CITY VARCHAR2(30) NOT NULL , any one of NAGPUR, DELHI,
 BANGALORE, BOMBAY

Query and Output:

- CREATE TABLE BRANCH (BNAME VARCHAR(20) PRIMARY KEY,CITY VARCHAR(30) NOT NULL CHECK(city LIKE "NAGPUR" OR city LIKE "DELHI" OR city LIKE "BANGALORE" OR city LIKE "BOMBAY"));



	BNAME	CITY
*	NULL	NULL

CUSTOMER

CNAME VARCHAR2(15) PRIMARY KEY,CITY VARCHAR2(20) NOT NULL

Query and Output:

- CREATE TABLE CUSTOMER(CNAME VARCHAR(15) PRIMARY KEY,CITY VARCHAR(20) NOT NULL);

	CNAME	CITY
	NULL	NULL

BORROW

LOANNO VARCHAR2(8) PRIMARY KEY / FIRST LETTER MUST START WITH 'L'

CNAME VARCHAR2(15) FOREIGN KEY REFERENCES CUSTOMER

BNAME VARCHAR2(20) FOREIGN KEY REFERENCES BRANCH

AMOUNT NUMBER(8,2) NOT NULL, CANNOT BE 0

Query and Output:

- CREATE TABLE BORROW(LOANNO VARCHAR(8) PRIMARY KEY CHECK (loanno LIKE "L%"),CNAME VARCHAR(15) , FOREIGN KEY(CNAME) REFERENCES CUSTOMER(CNAME),BNAME VARCHAR(20) , FOREIGN KEY(BNAME) REFERENCES BRANCH(BNAME),AMOUNT DECIMAL(8,2)NOT NULL Check(AMOUNT !=0));

	LOANNO	CNAME	BNAME	AMOUNT
	NULL	NULL	NULL	NULL

INSERTION OF VALUES

1. Inserting values to Branch

VRCE NAGPUR

AJNI NAGPUR

KAROLBAGH DELHI

CHANDNI DELHI

DHARAMPETH NAGPUR

MG ROAD BANGALORE

ANDHERI BOMBAY

NEHRU PALACE DELHI

POWAI BOMBAY

Query and Output:

- INSERT INTO BRANCH VALUES ('VRCE','NAGPUR'),
('AJNI','NAGPUR'),
('KAROLBAGH','DELHI'),
('CHANDNI','DELHI'),
('DHARAMPETH','NAGPUR'),
('MG ROAD','BANGALORE'),
('ANDHERI','BOMBAY'),
('NEHRU PALACE','DELHI'),
('POWAI','BOMBAY');

Result Grid		Filter Rows:
	BNAME	CITY
▶	AJNI	NAGPUR
	ANDHERI	BOMBAY
	CHANDNI	DELHI
	DHARAMPETH	NAGPUR
	KAROLBAGH	DELHI
	MG ROAD	BANGALORE
	NEHRU PALACE	DELHI
	POWAI	BOMBAY
	VRCE	NAGPUR
*	NULL	NULL

2. Inserting values into Customer table

ANIL CALCUTTA

SUNILDELHI

MEHUL BARODA

MANDAR PATNA

MADHURI NAGPUR

PRAMOD NAGPUR

SANDIP SURAT

SHIVANI BOMBAY

KRANTI BOMBAY

NAREN BOMBAY

Query and Output:

- INSERT INTO Customer VALUES
 ('ANIL','CALCUTTA'),
 ('MEHUL','BARODA'),
 ('MANDAR','PATNA'),
 ('MADHURI','NAGPUR'),
 ('PRAMOD','NAGPUR'),
 ('SANDIP','SURAT'),
 ('SHIVANI','BOMBAY'),
 ('KRANTI','BOMBAY'),
 ('NAREN','BOMBAY');

Result Grid		Filter Rows
	CNAME	CITY
▶	ANIL	CALCUTTA
	KRANTI	BOMBAY
	MADHURI	NAGPUR
	MANDAR	PATNA
	MEHUL	BARODA
	NAREN	BOMBAY
	PRAMOD	NAGPUR
	SANDIP	SURAT
	SHIVANI	BOMBAY
	NULL	NULL

3. Inserting values into Deposit table

DEPOSITE

Actno	Cname	Bname	Amount	Adate
D100	ANIL	VRCE	1000.00	1-MAR-95
D101	SUNIL	AJNI	500.00	4-JAN-96
D102	MEHUL	KAROLBAGH	3500.00	17-NOV-95
D104	MADHURI	CHANDNI	1200.00	17-DEC-95
D105	PRAMOD	MG ROAD	3000.00	27-MAR-96
D106	SANDIP	ANDHERI	2000.00	31-MAR-96
D107	SHIVANI	VIRAR	1000.00	5-SEP-95
D108	KRANTI	NEHRU PLACE	5000.00	2-JUL-95
D109	MINU	POWAI	7000.00	10-AUG-95

Query and Output:

- INSERT INTO DEPOSIT VALUES
('D100','ANIL','VRCE',1000.00,'1995-03-01'),
('D101','SUNIL','AJNI',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'),
('D108','KRANTI','NEHRU PALACE',5000.00,'1995-07-02');

	ACTNO	CNAME	BNAME	AMOUNT	ADATE
▶	D100	ANIL	VRCE	1000.00	1995-03-01
	D101	SUNIL	AJNI	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
	D108	KRANTI	NEHRU PALACE	5000.00	1995-07-02
*	NULL	NULL	NULL	NULL	NULL

4. Inserting values into borrow table

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

Query and Output:

- INSERT INTO borrow VALUES
('L201','ANIL','VRCE',1000.00),
('L206','MEHUL','AJNI',5000.00),
('L311','SUNIL','DHARAMPETH',3000.00),
('L321','MADHURI','ANDHERI',2000.00),
('L371','PRAMOD','VRCE',8000.00),
('L481','KRANTI','NEHRU PALACE',3000.00);

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

SELECTING DATA FROM SINGLE TABLE

1. List all data from table deposit

Query and Output:

- SELECT * FROM dbms.deposit;

	ACTNO	CNAME	BNAME	AMOUNT	ADATE
▶	D100	ANIL	VRCE	1000.00	1995-03-01
	D101	SUNIL	AJNI	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
	D108	KRANTI	NEHRU PALACE	5000.00	1995-07-02
•	NULL	NULL	NULL	NULL	NULL

2. List all data from borrow

Query and Output:

- SELECT * FROM dbms.borrow;

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

3. List all data from customer

Query and Output:

- SELECT * FROM dbms.customer;

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

4. List all data from branch

Query and Output:

- SELECT * FROM dbms.branch;

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

5. Give account no and amount of deposit

Query and Output:

- SELECT ACTNO,AMOUNT FROM deposit;

	ACTNO	AMOUNT
▶	D100	1000.00
	D101	500.00
	D102	3500.00
	D104	1200.00
	D105	3000.00
	D106	2000.00
	D108	5000.00
•	NULL	NULL

6. Give customer name and account no of depositors

Query and Output:

- SELECT CNAME,ACTNO FROM deposit;

	CNAME	ACTNO
▶	ANIL	D100
	KRANTI	D108
	MADHURI	D104
	MEHUL	D102
	PRAMOD	D105
	SANDIP	D106
	SUNIL	D101
•	NULL	NULL

7. Give name of customers

Query and Output:

- SELECT CNAME FROM customer;

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

8. Give name of branches

Query and Output:

- SELECT BNAME FROM branch;

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

9. Give name of borrows

Query and Output:

- SELECT CNAME FROM borrow;

CNAME
ANIL
KRANTI
MADHURI
MEHUL
PRAMOD
SUNIL

10. Give names of customer living in city Nagpur

Query and Output:

- SELECT CNAME,CITY FROM CUSTOMER WHERE CITY="NAGPUR";

CNAME	CITY
MADHURI	NAGPUR
PRAMOD	NAGPUR
NULL	NULL

11. Give names of depositors having amount greater than 4000

Query and Output:

- SELECT CNAME,AMOUNT FROM deposit WHERE AMOUNT>4000;

	CNAME	AMOUNT
▶	KRANTI	5000.00

12. Give account date of Anil

Query and Output:

- SELECT CNAME,ADATE FROM DEPOSIT WHERE CNAME="ANIL";

	CNAME	ADATE
▶	ANIL	1995-03-01

13. Give name of all branches located in Bombay

Query and Output:

- SELECT BNAME,CITY FROM branch WHERE CITY="BOMBAY";

	BNAME	CITY
▶	ANDHERI	BOMBAY
	POWAI	BOMBAY
*	NULL	NULL

14. Give name of borrower having loan number l206

Query and Output:

- SELECT CNAME,LOANNO FROM borrow where LOANNO='L206';

	CNAME	LOANNO
▶	MEHUL	L206
*	NULL	NULL

15. Give names of depositors having account at VRCE

Query and Output:

- SELECT CNAME,BNAME FROM deposit WHERE BNAME="VRCE";

	CNAME	BNAME
▶	ANIL	VRCE

16. Give names of all branched located in city Delhi

Query and Output:

- SELECT BNAME,CITY FROM branch WHERE CITY="DELHI";

	BNAME	CITY
▶	CHANDNI	DELHI
	KAROLBAGH	DELHI
	NEHRU PALACE	DELHI
*	NULL	NULL

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

Query and Output:

- SELECT CNAME,ADATE FROM deposit WHERE ADATE='1996-12-01';

CNAME	ADATE

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

Query and Output:

- SELECT ACTNO,AMOUNT FROM deposit WHERE ADATE BETWEEN '1996-05-01' AND '1996-12-01';

ACTNO	AMOUNT
NULL	NULL

19. Give name of the city where branch KAROLBAGH is located

Query and Output:

- SELECT CITY,BNAME FROM branch WHERE BNAME="KAROLBAGH";

CITY	BNAME
DELHI	KAROLBAGH
NULL	NULL

20. Give details of customer ANIL

Query and Output:

- SELECT * FROM deposit WHERE CNAME="ANIL";

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

Result

The query was executed and the output was successfully obtained.

DCL & TCL COMMANDS

Experiment No.: 3

7-03-2023

Aim

Familiarization DCL & TCL Commands

CO

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. Commit

Query and Output:

- DELETE FROM emp WHERE empno = 106;
COMMIT;



empno	ename	hiredate	salary	commission
101	Ramesh	1980-01-17	5000	0
102	Ajay	1985-07-05	5000	500
103	Ravi	1981-08-12	1500	0
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-07-05	3000	0
NULL	NULL	NULL	NULL	NULL

2. Rollback

Query and Output:

- DELETE FROM emp WHERE empno = 106;
ROLLBACK;



empno	ename	hiredate	salary	commission
101	Ramesh	1980-01-17	5000	0
102	Ajay	1985-07-05	5000	500
103	Ravi	1981-08-12	1500	0
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-07-05	3000	0
NULL	NULL	NULL	NULL	NULL

3. Savepoint

Query and Output:

- INSERT INTO emp VALUES (106, 'Anil', '1985-07-05', 5000, 0);
SAVEPOINT a;



The screenshot shows a database query result window titled 'Result Grid'. It contains a table with 6 columns: empno, ename, hiredate, salary, and commission. The table has 7 rows of data. The first six rows represent employees with their respective details, and the seventh row shows NULL values for all columns.

empno	ename	hiredate	salary	commission
101	Ramesh	1980-01-17	5000	0
102	Ajay	1985-07-05	5000	500
103	Ravi	1981-08-12	1500	0
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-07-05	3000	0
106	Anil	1985-07-05	5000	0
NULL	NULL	NULL	NULL	NULL

4. Grant

Query and Output:

- GRANT SELECT,UPDATE ON emp TO bank;

5. Revoke

Query and Output:

- REVOKE SELECT, UPDATE ON emp FROM bank;

Result

The query was executed and the output was successfully obtained.

SELECT, WHERE AND DATE, NUMBER AND CHARACTER FUNCTIONS

Experiment No.: 4

27-03-2023

Aim

Accessing and Optimizing databases using SELECT, Filtering using WHERE and DATE, NUMBER and CHARACTER FUNCTIONS.

CO

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. Display the names of all customer who have “a” and “i” in their names

Query and Output:

- SELECT CNAME FROM customer WHERE CNAME LIKE '%A%' AND CNAME LIKE '%I%';



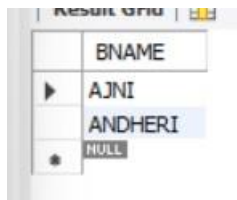
A screenshot of a database result grid titled 'Result Grid'. It displays a table with one column labeled 'CNAME'. The rows contain the names: ANIL, KRANTI, MADHURI, SANDIP, SHIVANI, and NULL. The row with 'NULL' is highlighted with a blue background.

CNAME
ANIL
KRANTI
MADHURI
SANDIP
SHIVANI
NULL

2. Using the branch table, write a query to display all branch names whose names start with “A” and have “a” and “i” anywhere in their name

Query and Output:

- SELECT BNAME FROM branch WHERE BNAME LIKE 'A%' AND BNAME LIKE '%A%' AND BNAME LIKE '%I%';



A screenshot of a database result grid titled 'Result Grid'. It displays a table with one column labeled 'BNAME'. The rows contain the names: AJNI, ANDHERI, and NULL. The row with 'NULL' is highlighted with a blue background.

BNAME
AJNI
ANDHERI
NULL

3. What is the total salary of all employees?

Query and Output:

- SELECT SUM(salary) FROM emp;

	SUM(salary)
▶	17500

4. What is the Average salary of employees?

Query and Output:

- SELECT AVG(salary) FROM emp;

	AVG(salary)
▶	3500.0000

5. Which employee has the highest salary ?

Query and Output:

- SELECT ename, salary FROM EMP WHERE salary IN (SELECT MAX(salary) FROM EMP);

	ename	salary
▶	Ramesh	5000
	Ajay	5000

6. Who is having the highest commission and the commission?

Query and Output:

- SELECT ename,commission FROM EMP WHERE commission IN (SELECT MAX(commission) FROM EMP);

	ename	commission
▶	Nikesh	700

7. Which employee has the lowest salary ?

Query and Output:

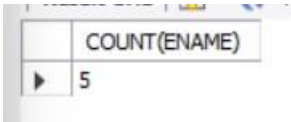
- SELECT ename, salary FROM EMP WHERE salary IN (SELECT min(salary) FROM EMP);

	ename	salary
▶	Ravi	1500

8. How many employees are there?

Query and Output:

- SELECT COUNT(ENAME) FROM EMP;

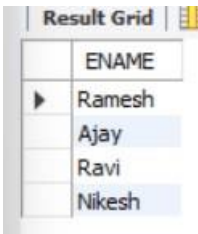


	COUNT(ENAME)
▶	5

9. Find the non repeating names of employees

Query and Output:

- SELECT distinct(ENAME) FROM emp;

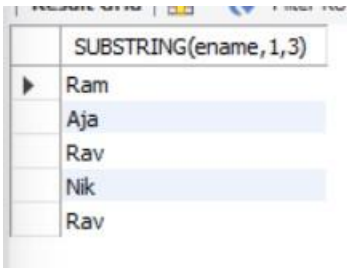


	ENAME
▶	Ramesh
	Ajay
	Ravi
	Nikesh

10. List the first 3 characters of employee names

Query and Output:

- SELECT SUBSTRING(ename,1,3) FROM emp;



	SUBSTRING(ename,1,3)
▶	Ram
	Aja
	Rav
	Nik
	Rav

Result

The query was executed and the output was successfully obtained.

AGGREGATE FUNCTIONS

Experiment No.:5

28-03-2023

Aim

Optimizing database using Aggregate Functions.

CO

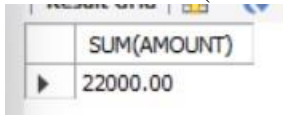
CO1: Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

Procedure

1. List total loan

Query and Output:

- SELECT SUM(AMOUNT) FROM borrow;

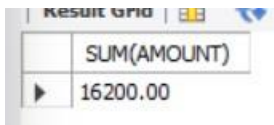


SUM(AMOUNT)
22000.00

2. List total deposit

Query and Output:

- SELECT SUM(AMOUNT) FROM deposit;

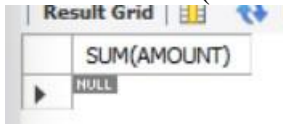


SUM(AMOUNT)
16200.00

3. List total loan taken from KAROLBAGH branch

Query and Output:

- SELECT SUM(AMOUNT) FROM borrow WHERE BNAME="KAROLBAGH";

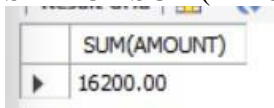


SUM(AMOUNT)
NULL

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

Query and Output:

- SELECT SUM(AMOUNT) FROM deposit WHERE ADATE >1996-01-01;



SUM(AMOUNT)
16200.00

5. List total deposit of customers living in city NAGPUR

Query and Output:

- SELECT SUM(AMOUNT) FROM deposit D JOIN customer C ON C.cname = D.cname
WHERE C.city LIKE 'NAGPUR';

SUM(AMOUNT)
4200.00

6. List total deposit of customer having branch in BOMBAY

Query and Output:

- SELECT SUM(AMOUNT) FROM deposit D JOIN branch B ON D.BNAME=B.BNAME WHERE B.CITY='BOMBAY';

SUM(AMOUNT)
2000.00

7. Count total number of branch cities

Query and Output:

- SELECT CITY,COUNT(CITY) FROM branch GROUP BY CITY;

CITY	COUNT(CITY)
NAGPUR	3
BOMBAY	2
DELHI	3
BANGALORE	1

8. Give branch names and branch wise deposit

Query and Output:

- SELECT BNAME,SUM(AMOUNT) FROM deposit group by BNAME;

BNAME	SUM(AMOUNT)
AJNI	500.00
ANDHERI	2000.00
CHANDNI	1200.00
KAROLBAGH	3500.00
MG ROAD	3000.00
NEHRU PALACE	5000.00
VRCE	1000.00

9. Give city wise name and branch wise deposit

Query and Output:

- SELECT B.CITY,B.BNAME,SUM(D.AMOUNT) FROM deposit D JOIN branch B ON D.BNAME=B.BNAME group by B.BNAME;

	CITY	BNAME	SUM(D.AMOUNT)
▶	NAGPUR	VRCE	1000.00
	NAGPUR	AJNI	500.00
	DELHI	KAROLBAGH	3500.00
	DELHI	CHANDNI	1200.00
	BANGALORE	MG ROAD	3000.00
	BOMBAY	ANDHERI	2000.00
	DELHI	NEHRU PALACE	5000.00

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

Query and Output:

- SELECT BNAME,SUM(AMOUNT) FROM borrow B JOIN customer C ON B.CNAME=C.CNAME WHERE C.CITY="NAGPUR" GROUP BY BNAME;

	BNAME	SUM(AMOUNT)
▶	ANDHERI	2000.00
	VRCE	8000.00

11. Count total number of customers

Query and Output:

- SELECT * FROM dbms.branch;

	COUNT(CNAME)
▶	10

12. Count total number of depositors branch wise

Query and Output:

- SELECT BNAME,count(CNAME) FROM deposit group by BNAME;

	BNAME	count(CNAME)
▶	AJNI	1
	ANDHERI	1
	CHANDNI	1
	KAROLBAGH	1
	MG ROAD	1
	NEHRU PALACE	1
	VRCE	1

13. Give maximum loan from branch VRCE

Query and Output:

- SELECT BNAME,MAX(AMOUNT) FROM borrow WHERE BNAME="VRCE";

	BNAME	MAX(AMOUNT)
▶	VRCE	8000.00

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

Query and Output:

- SELECT COUNT(D.CNAME) FROM deposit D JOIN borrow B ON D.CNAME=B.CNAME;



COUNT(D.CNAME)
6

Result

The query was executed and the output was successfully obtained.

SET OPERATIONS

Experiment No.: 6

03-04-2023

Aim

Optimizing database using Set Operations

CO

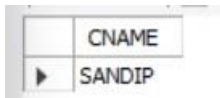
CO1: Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

Procedure

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

Query and Output:

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

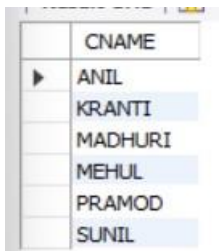


CNAME
SANDIP

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

Query and Output:

- SELECT CNAME FROM deposit WHERE CNAME IN (SELECT CNAME FROM borrow);



CNAME
ANIL
KRANTI
MADHURI
MEHUL
PRAMOD
SUNIL

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

Query and Output:

- SELECT D.CNAME FROM deposit D WHERE D.BNAME IN (SELECT BNAME FROM deposit WHERE CNAME='SUNIL');

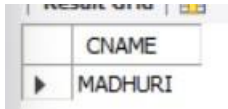


CNAME
SUNIL

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

Query and Output:

- SELECT C.CNAME FROM CUSTOMER C,DEPOSIT D, BRANCH B WHERE C.CITY = 'NAGPUR' AND C.CNAME = D.CNAME AND D.BNAME = B.BNAME AND B.CITY IN ('BOMBAY','DELHI');



	CNAME	BNAME
▶	MADHURI	

- List all the depositors living in city NAGPUR

Query and Output:

- SELECT CNAME FROM deposit WHERE CNAME IN(SELECT CNAME FROM customer WHERE CITY='NAGPUR');



	CNAME	BNAME
▶	MADHURI	
	PRAMOD	

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

Query and Output:

- SELECT D.CNAME FROM DEPOSIT D,CUSTOMER C,BRANCH B WHERE C.CITY = 'NAGPUR' AND C.CNAME = D.CNAME AND D.BNAME = B.BNAME AND B.CITY='BOMBAY';



	CNAME	BNAME
	CNAME	

- List the branch cities of Anil and Sunil

Query and Output:

- SELECT BNAME,CITY FROM BRANCH WHERE BNAME IN(SELECT BNAME FROM deposit WHERE CNAME IN('ANIL','SUNIL'));



	BNAME	CITY
▶	VRCE	NAGPUR
	AJNI	NAGPUR
*	NULL	NULL

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

Query and Output:

- SELECT CNAME FROM deposit WHERE AMOUNT>1000 UNION(SELECT CNAME FROM borrow WHERE AMOUNT<1000);

CNAME
MEHUL
MADHURI
PRAMOD
SANDIP
KRANTI

9. List the cities of depositors having branch VRCE.

Query and Output:

- SELECT CITY FROM branch WHERE BNAME IN (SELECT BNAME FROM deposit WHERE BNAME='VRCE');

CITY
NAGPUR

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

Query and Output:

- SELECT D.cname FROM deposit D, customer C WHERE D.amount < 1000 AND C.CITY = (SELECT city FROM customer WHERE cname = "ANIL");

cname
SUNIL

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

Query and Output:

- SELECT BNAME,CITY FROM BRANCH WHERE BNAME IN (SELECT BNAME FROM deposit WHERE CNAME IN ('ANIL','SUNIL'));

BNAME	CITY
VRCE	NAGPUR
AJNI	NAGPUR
NULL	NULL

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

Query and Output:

- SELECT D.CNAME,D.AMOUNT,C.CITY FROM deposit D, customer C WHERE D.CNAME=C.CNAME AND C.CITY = (SELECT city FROM customer WHERE cname = "ANIL");

CNAME	AMOUNT	CITY
ANIL	1000.00	CALCUTTA

Result

The query was executed and the output was successfully obtained.

HAVING, GROUP BY, ORDER BY CLAUSES

Experiment No.: 7

04-04-2023

Aim

Accessing database using HAVING, GROUP BY, ORDER BY Clauses

CO

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.

Query and Output:

- SELECT BNAME FROM deposit GROUP BY BNAME HAVING sum(AMOUNT)>5000;

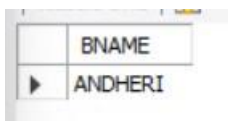


BNAME

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

Query and Output:

- SELECT BNAME FROM deposit WHERE BNAME IN (SELECT BNAME FROM BRANCH WHERE CITY = 'BOMBAY') GROUP BY BNAME HAVING SUM(AMOUNT) > 500;

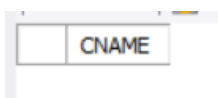


BNAME
ANDHERI

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

Query and Output:

- SELECT CNAME from deposit where AMOUNT=(select AVG(Amount) from DEPOSIT GROUP BY BNAME having AVG(Amount)>5000);



CNAME

- List the names of customers having maximum deposit

Query and Output:

- SELECT CNAME FROM deposit WHERE AMOUNT = (SELECT MAX(AMOUNT) FROM deposit);



CNAME
KRANTI

- List the name of branch having highest number of depositors?

Query and Output:

- SELECT BNAME FROM DEPOSIT GROUP BY BNAME HAVING COUNT(CNAME) >= ALL (SELECT COUNT(D2.CNAME) FROM DEPOSIT D2 GROUP BY D2.BNAME);

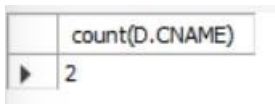


BNAME
AJNI
ANDHERI
CHANDNI
KAROLBAGH
MG ROAD
NEHRU PALACE
VRCE

- Count the number of depositors living in NAGPUR.

Query and Output:

- SELECT count(D.CNAME) FROM deposit D JOIN customer C ON D.CNAME=C.CNAME WHERE C.CITY='NAGPUR';



count(D.CNAME)
2

- Give names of customers in VRCE branch having more deposit than any other customer in same branch

Query and Output:

- SELECT CNAME FROM deposit WHERE BNAME='VRCE' and amount=(select max(AMOUNT) from deposit where BNAME='VRCE');

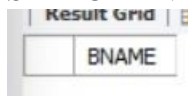


CNAME
ANIL

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

Query and Output:

- SELECT BNAME from deposit GROUP BY BNAME HAVING COUNT(BNAME)>5;

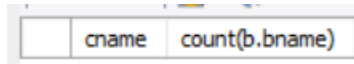


BNAME

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

Query and Output:

- 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)

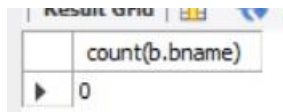


cname	count(b.bname)
-------	----------------

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

Query and Output:

- select count(b.bname) from deposit d,borrow b,customer c where c.cname and d.cname = b.cname and c.city in(select city from customer)



count(b.bname)
0

11. Display the name, hire date, number of months employed and day of the week on which the employee has started. Order the results by the day of the week starting with Monday.

Query and Output:

- SELECT eNAME,HIREDATE,ROUND((YEAR(CURDATE()) - YEAR(HIREDATE)) * 12 + (MONTH(CURDATE()) - MONTH(HIREDATE)), 0) AS MONTHS_EMPLOYED,DATE_FORMAT(HIREDATE,'%W') AS START_DAY_OF_WEEK FROM EMP ORDER BY DAYOFWEEK(HIREDATE);



	eNAME	HIREDATE	MONTHS_EMPLOYED	START_DAY_OF_WEEK
▶	Ravi	1981-08-12	504	Wednesday
	Ramesh	1980-01-17	523	Thursday
	Nikesh	1983-03-03	485	Thursday
	Ajay	1985-07-05	457	Friday
	Ravi	1985-07-05	457	Friday
	Anil	1985-07-05	457	Friday
	AMIL	1985-07-05	457	Friday

Result

The query was executed and the output was successfully obtained.

JOIN QUERY

Experiment No.: 8

11-04-2023

Aim

Optimizing database using Join Query

CO

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

Query and Output:

- SELECT D.cname AS CUSTOMER, B.city AS BRANCH_CITY, C.city AS CITY
FROM deposit D JOIN customer AS C ON C.cname=D.cname JOIN branch AS B ON
B.bname=D.bname WHERE C.city LIKE "BOMBAY" AND D.bname IN (SELECT
bname FROM branch WHERE city LIKE "NAGPUR");



CUSTOMER	BRANCH_CITY	CITY
----------	-------------	------

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

Query and Output:

- SELECT D.cname AS CUSTOMER, B.city AS BRANCH_CITY, C.city AS CITY
FROM deposit D JOIN customer AS C ON C.cname=D.cname JOIN branch AS B ON
B.bname=D.bname WHERE C.city=B.CITY;



CUSTOMER	BRANCH_CITY	CITY
----------	-------------	------

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

Query and Output:

- SELECT D.CNAME FROM deposit AS D JOIN borrow AS B ON
D.CNAME=B.CNAME JOIN customer C ON C.CNAME=D.CNAME WHERE
C.CITY="NAGPUR";



CNAME
MADHURI
PRAMOD

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

Query and Output:

- SELECT B.cname AS CUSTOMER, B.amount AS LOAN, D.amount AS DEPOSIT FROM borrow AS B JOIN deposit D ON D.cname=B.cname WHERE D.amount > 1000 AND B.amount > 2000;

	CUSTOMER	LOAN	DEPOSIT
▶	MEHUL	5000.00	3500.00
	PRAMOD	8000.00	3000.00
	KRANTI	3000.00	5000.00

- Give names of depositors having the same branch as the branch of Sunil

Query and Output:

- SELECT cname AS CUSTOMER, bname AS BRANCH FROM deposit WHERE bname LIKE (SELECT bname FROM deposit WHERE cname LIKE "Sunil");

	CUSTOMER	BRANCH
▶	SUNIL	AJNI

- Give names of borrowers having loan amount greater than the loan amount of Pramod

Query and Output:

- SELECT cname AS CUSTOMER, amount AS LOAN FROM borrow WHERE amount > (SELECT amount FROM borrow WHERE cname LIKE "PRAMOD");

	CUSTOMER	LOAN
--	----------	------

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

Query and Output:

- SELECT DISTINCT C.cname AS CUSTOMER, C.city AS CITY FROM customer AS C JOIN deposit AS D ON D.cname=C.cname JOIN branch AS B ON B.city=C.city WHERE C.city LIKE (SELECT city FROM branch WHERE bname LIKE (SELECT bname FROM deposit WHERE cname LIKE "Sunil"));

	CUSTOMER	CITY
▶	PRAMOD	NAGPUR
	MADHURI	NAGPUR

- Give branch city and living city of Pramod

Query and Output:

- SELECT D.cname AS CUSTOMER, B.city AS BRANCH_CITY, C.city AS LIVING_CITY FROM deposit D JOIN branch AS B ON B.bname = D.bname JOIN customer AS C ON C.cname = D.cname WHERE D.cname LIKE "PRAMOD";

	CUSTOMER	BRANCH_CITY	LIVING_CITY
▶	PRAMOD	BANGALORE	NAGPUR

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

Query and Output:

- SELECT B.city AS BRANCH_CITY, D.cname AS CUSTOMER FROM branch AS B JOIN deposit AS D ON D.bname=B.bname WHERE D.cname IN ("SUNIL","Anil");

	BRANCH_CITY	CUSTOMER
▶	NAGPUR	ANIL
	NAGPUR	SUNIL

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

Query and Output:

- SELECT city AS LIVING_CITY,cname AS CUSTOMER FROM customer WHERE cname IN ("Anil", "Sunil");

	LIVING_CITY	CUSTOMER
▶	CALCUTTA	ANIL
	DELHI	SUNIL

Result

The query was executed and the output was successfully obtained.

SUBQUERY

Experiment No.: 9

18-04-2023

Aim

Accessing database with Subquery

CO

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

- a) Create table of **salesorder** with following fields: Order Number(Unique key begins with 0),Clientnumber(FK),Order Date.

Query and Output:

- create table salesorder(ordnno int primary key, clientno int, Orddate date, FOREIGN KEY (clientno) REFERENCES clientmaster(clientno));

✓ 48 20:28:57 create table salesorder(ordnno int primary key, clie... 0 row(s) affected 0.031 sec

- b) Create table **clientmaster** with following fields: clientno(PK), cname,city.

Query and Output:

- create table clientmaster(clientno int primary key, cname varchar(20),city varchar(20))

✓ 47 20:28:25 create table clientmaster(clientno int primary key, ... 0 row(s) affected 0.062 sec

- c) Create table **salesmaster** with following fields: salesmanno(PK),sname,city.

Query and Output:

- create table salesmaster(salesmanno int primary key, sname varchar(20),city varchar(20))

✓ 49 20:29:41 create table salesmaster(salesmanno int primary k... 0 row(s) affected 0.016 sec

- d) Add 5 rows of data in 3 tables

- clientmaster**

	clientno	cname	city
▶	101	senchu	kannur
	102	sobin	kannur
	103	sajan	mumbai
	104	siju	gujarath
	105	george	kottayam
•	NULL	NULL	NULL

- salesmaster**

	salesmanno	sname	city
▶	1	biju	mumbai
	2	amith	mumbai
	3	sreelal	kannur
	4	anoop	thrissur
	5	goutham	pala
•	NULL	NULL	NULL

- **salesorder**

	ordnno	clientno	Orddate
▶	0	101	2013-07-01
	1	103	2013-02-01
	2	103	2013-03-21
	3	102	2013-03-02
	4	105	2013-04-02
•	NULL	NULL	NULL

1. Retrieve all orders placed by a client named sajan.

Query and Output:

- select ordnno from salesorder where clientno= (select clientno from clientmaster where cname="sajan");

	ordnno
▶	1
	2
•	NULL

2. Retrieve the names of all clients and salesman in the city of mumbai.

Query and Output:

- SELECT cname,sname FROM clientmaster,salesmaster WHERE cname IN(SELECT cname FROM clientmaster WHERE CITY='MUMBAI') AND sname IN(SELECT sname FROM salesmaster WHERE CITY='MUMBAI');

	cname	sname
▶	sajan	biju
	sajan	amith

3. Retrieve the order number, cname, order date from client master and sales order table.

Query and Output:

- SELECT s.ordnno,s.Orddate,c.cname FROM clientmaster c,salesorder s where c.cname in(SELECT cname FROM clientmaster WHERE c.clientno = s.clientno);

	ordnno	Orddate	cname
▶	0	2013-07-01	senchu
	1	2013-02-01	sajan
	2	2013-03-21	sajan
	3	2013-03-02	sobin
	4	2013-04-02	george

Result

The query was executed and the output was successfully obtained.

VIEW TABLE

Experiment No.: 10

29-04-2023

Aim

Accessing database with View Table

CO

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 a view view1 from salesorder with (order number,order date)

Query and Output:

- create view view1 as (select ordnno,Orddate from salesorder)

60 20:47:21 create view view1 as (select ordnno,Orddate from... 0 row(s) affected 0.000 sec

2. Describe the view view1

Query and Output:

- desc view1

	Field	Type	Null	Key	Default	Extra
▶	ordnno	int	NO		NULL	
	Orddate	date	YES		NULL	

3. Display Values of view1

Query and Output:

- select * from view1

	ordnno	Orddate
▶	0	2013-07-01
	1	2013-02-01
	2	2013-03-21
	3	2013-03-02
	4	2013-04-02

4. Insert one row of data in view1 and observe the output/error message

Query and Output:

- insert into view1(ordnno,Orddate) values (5,"2013-07-22")

64 20:56:59 insert into view1(ordnno,Orddate) values (5,"2013... 1 row(s) affected 0.015 sec

- Create a view view2 from clientmaster (clientno,cname)

Query and Output:

- create view view2 as (select clientno,cname from clientmaster)

65 20:58:27 create view view2 as (select clientno,cname from ... 0 row(s) affected 0.000 sec

- Insert one row of data in view2 and observe the output/error message

Query and Output:

- insert into view2(clientno,cname) values (106,"chiru")

66 20:59:30 insert into view2(clientno,cname) values (106,"chi... 1 row(s) affected 0.016 sec

- Create a view view3 from salesmaster with all columns with condition 'city=mumbai'

Query and Output:

- create view view3 as (select * from salesmaster where city="mumbai");

67 21:00:20 create view view3 as (select * from salesmaster w... 0 row(s) affected 0.000 sec

- Create a view view4 from clientmaster and salesmaster with details (cname,city,sname,city)

Query and Output:

- create view view4 as (select clientmaster.cname,clientmaster.city as ccity,salesmaster.sname,salesmaster.city as scity from clientmaster,salesmaster);

68 21:01:27 create view view4 as (select clientmaster.cname,... 0 row(s) affected 0.016 sec

- Describe view4

Query and Output:

- describe view4

	Field	Type	Null	Key	Default	Extra
▶	cname	varchar(20)	YES		NULL	
	ccity	varchar(20)	YES		NULL	
	sname	varchar(20)	YES		NULL	
	scity	varchar(20)	YES		NULL	

- Insert one row of data in view4 and observe the output/error message

Query and Output:

- insert into view4(cname,ccity,sname,scity) values ("ani","mumbai","Luku","bobmay")

70 21:04:11 insert into view4(cname,ccity,sname,scity) values ... Error Code: 1393. Can not modify more than one b... 0.000 sec

11. Create a view view5 from view4(cname,sname)

Query and Output:

- create view view5 as (select cname,sname from view4)

71 21:05:16 create view view5 as (select cname,sname from v... 0 row(s) affected 0.000 sec

12. Describe view5

Query and Output:

- describe view5

	Field	Type	Null	Key	Default	Extra
	cname	varchar(20)	YES		NULL	
	sname	varchar(20)	YES		NULL	

13. Insert one row of data in view5 and observe the output/error message

Query and Output:

- insert into view5(cname,sname) values ("Achu","Akki")

73 21:07:17 insert into view5(cname,sname) values ("Achu","... Error Code: 1393. Can not modify more than one b... 0.000 sec

14. Delete the view4 table

Query and Output:

- drop view view4

74 21:08:13 drop view view4 0 row(s) affected 0.016 sec

15. Delete the view5 table.

Query and Output:

- drop view view5

75 21:08:55 drop view view5 0 row(s) affected 0.016 sec

Result

The query was executed and the output was successfully obtained.

FUNCTIONS

Experiment No.: 11

12-06-2023

Aim

Familiarization of PL/SQL Program-Functions.

CO

CO2: Apply PL/SQL for processing databases.

Procedure

1. Create 'Student' table with fields (Regno, Name, English_mark, Malayalam_mark, Physics_mark, Chemistry_mark, Maths_mark).

Query and Output:

- CREATE TABLE Student (Regno INT PRIMARY KEY, Name VARCHAR(50), English_mark INT, Malayalam_mark INT, Physics_mark INT, Chemistry_mark INT, Maths_mark INT);

Insert 5 rows into the table

Result Grid							
		Filter Rows:		Edit:		Export/Import:	
	Regno	Name	English_mark	Malayalam_mark	Physics_mark	Chemistry_mark	Maths_mark
▶	101	arun	90	80	79	67	90
	102	amit	90	60	79	87	90
	103	amit	90	60	79	87	89
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Write a function which accepts the Regno, calculate the Total Marks and return the Total Marks.

Query and Output:

- CREATE DEFINER=`root`@`localhost` PROCEDURE `get_result`(INOUT var1 INT)
 BEGIN
 select sum(English_mark + Malayalam_mark + Physics_mark + Chemistry_mark + Maths_mark) into var1 from `aninadb1`.`student` where reg_no =var1 ;

 END

- SET @M = '2';

```
CALL get_result(@M);  
SELECT @M as total_mark;
```

	total_mark
▶	497

Result

The query was executed and the output was successfully obtained.

STORED PROCEDURES

Experiment No.: 12

20-06-2023

Aim

Familiarization of PL/SQL Program- Stored Procedures

CO

CO2: Apply PL/SQL for processing databases.

Procedure

1. Create a table named 'Employee' with the following fields.

Empid(int),empname(varchar(50), empdept(varchar(20),empage(int),emptytype(varchar(45))

Query and Output:

- Create table Employee (Empid int,empname varchar(50), empdept varchar(20),empage int,emptytype varchar(45));

97 21:34:23 Create table Employee (Empid int,empname varchar(50), empdept varchar(20),empage int,emptytype varchar(45)) 0 row(s) affected

0.015 sec

2. Insert at least 15 rows of data.

Emp type should be contract or permanent and all department should contain employees in both categories. Employee department should be in the format Dpt1,Dpt2..etc.

empid	empname	empage	emptytype	empdept
101	akhsa	25	permanent	dept1
102	arun	20	temporary	dept2
103	ashwathy	27	permanent	dept3
104	anitta	26	permanent	dept2
105	akash	28	temporary	dept4
106	ashwin	32	permanent	dept4
107	akhil	34	temporary	dept4
108	aswin	30	temporary	dept3
109	roof	31	permanent	dept3
110	akku	34	permanent	dept2
111	althaf	33	permanent	dept1
112	dilna	36	permanent	dept3
113	ardra	38	permanent	dept
NULL	NULL	NULL	NULL	NULL

3. Create stored procedure to demonstrate all four types of parameter passing

1. Procedure without Parameter

Query and Output:

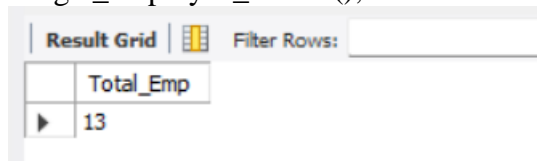
- CREATE DEFINER=`root`@`localhost` PROCEDURE `get_employee_details`()

BEGIN

SELECT * FROM `dbms`.`employee` WHERE EmpDept = 'Dpt3' AND EmpType=
'Contract';

SELECT COUNT(EmpID) AS Total_Emp FROM `dbms`.`employee`;

END
- CALL get_employee_details();



Total_Emp
13

2. Procedure with IN Parameter

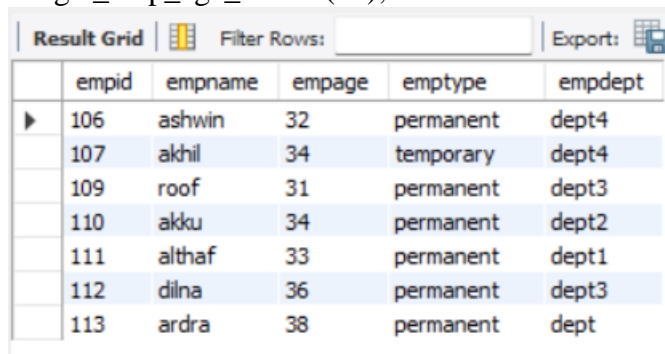
Query and Output:

- CREATE DEFINER=`root`@`localhost` PROCEDURE `get_emp_age_details`(IN
in_age INT)

BEGIN

SELECT * FROM `dbms`.`employee` WHERE EmpAge > in_age;

END
- CALL get_emp_age_details(30);



empid	empname	empage	emptype	empdept
106	ashwin	32	permanent	dept4
107	akhil	34	temporary	dept4
109	roof	31	permanent	dept3
110	akku	34	permanent	dept2
111	althaf	33	permanent	dept1
112	dilna	36	permanent	dept3
113	ardra	38	permanent	dept

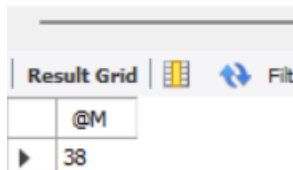
3. Procedure with OUT Parameter

Query and Output:

- CREATE DEFINER=`root`@`localhost` PROCEDURE `get_emp_max_age`(OUT MaxAge INT)

BEGIN

SELECT MAX(EmpAge) INTO MaxAge FROM `dbms`.`employee`;
END
- CALL get_emp_max_age(@M);
SELECT @M;



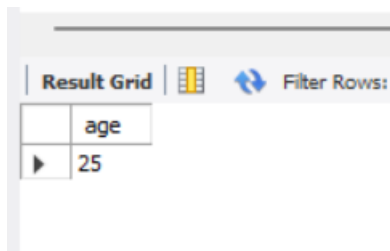
	@M
	38

4. Procedures with INOUT Parameter

Query and Output:

- CREATE DEFINER=`root`@`localhost` PROCEDURE `get_details`(INOUT var1 int)
BEGIN
SELECT * FROM `dbms`.`employee` WHERE EmpID < var1;

SELECT MAX(EmpAge) INTO var1 FROM `dbms`.`employee` WHERE EmpID<var1;
END
- SET @M = '105';
CALL get_details(@M);
SELECT @M as age ;



	age
	25

Result

The query was executed and the output was successfully obtained.

TRIGGER

Experiment No.: 13

20-06-2023

Aim

Familiarization of PL/SQL Program- Trigger

CO

CO2: Apply PL/SQL for processing databases.

Procedure

Create the following tables and demonstrate the use of triggers.

employee (emp_no, emp_name, DOB, address, doj, designation, mobile_no, dept_no, salary).

salary(empno, old_sal, new_sal, rev_date)

trainees_data()

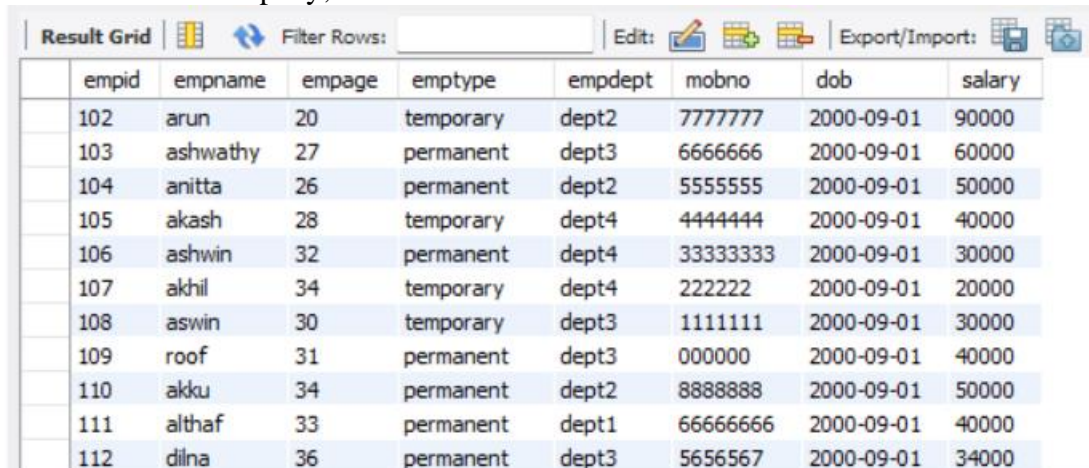
personal_updatons()

past_employees(emp_no, name, last_designation, dept_no)

1. Create a new database/schema named “company” and create the required tables.

Query and Output:

- create database company;



The screenshot shows a database application interface with a 'Result Grid' tab selected. It displays a table with 9 columns: empid, empname, empage, emptype, empdept, mobno, dob, and salary. The table contains 12 rows of employee data.

empid	empname	empage	emptype	empdept	mobno	dob	salary
102	arun	20	temporary	dept2	7777777	2000-09-01	90000
103	ashwathy	27	permanent	dept3	6666666	2000-09-01	60000
104	anitta	26	permanent	dept2	5555555	2000-09-01	50000
105	akash	28	temporary	dept4	4444444	2000-09-01	40000
106	ashwin	32	permanent	dept4	3333333	2000-09-01	30000
107	akhil	34	temporary	dept4	222222	2000-09-01	20000
108	aswin	30	temporary	dept3	1111111	2000-09-01	30000
109	roof	31	permanent	dept3	000000	2000-09-01	40000
110	akku	34	permanent	dept2	8888888	2000-09-01	50000
111	althaf	33	permanent	dept1	6666666	2000-09-01	40000
112	dilna	36	permanent	dept3	5656567	2000-09-01	34000

2. Create a Trigger for the employee table that will store the old and new salary into another table SALARY while updating salary.

Query and Output:

- CREATE DEFINER=`root`@`localhost` TRIGGER `employee_BEFORE_UPDATE` BEFORE UPDATE ON `employee` FOR EACH ROW BEGIN

INSERT INTO `company`.`salary` values(old.emp_no,old.salary,new.salary, curdate());

END

	empid	oldsalary	newsalary
▶	113	8000	23000
*	NULL	NULL	NULL

3. Create a Trigger to populate the trainees_data table when a new person joins the company as a trainee. Save the employee number, name and the department.

Query and Output:

- CREATE DEFINER=`root`@`localhost` TRIGGER `employee_AFTER_INSERT`
AFTER INSERT ON `employee` FOR EACH ROW BEGIN

IF NEW.designation='trainee'

then

INSERT INTO `company`.`trainees_data` values(new.emp_no ,new.emp_name,new
.dept_no);

end if;

END

	empid	empname	dept
▶	113	ardra	dept4

4. Create a Trigger for the employee table that will store the updated mobile number into another table personal_updates while updating mobile number.

Query and Output:

- CREATE DEFINER=`root`@`localhost` TRIGGER `employee_AFTER_UPDATE`
AFTER UPDATE ON `employee` FOR EACH ROW BEGIN

INSERT INTO `company`.`personal_updates` values(old.emp_name,new.mobile_no);

END

	empname	mobno
▶	arun	777777

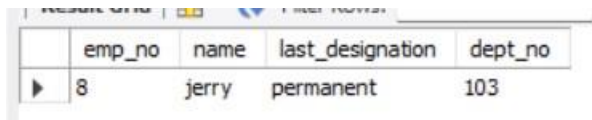
5. Create a Trigger to save the details of the employee to past_employees table upon removing that employee from database.

Query and Output:

- CREATE DEFINER=`root`@`localhost` TRIGGER `employee_BEFORE_DELETE` BEFORE DELETE ON `employee` FOR EACH ROW BEGIN

INSERT INTO `company`.`past_employees` value(old.emp_no, old.emp_name, old.designation ,old.dept_no);

END



	emp_no	name	last_designation	dept_no
▶	8	jerry	permanent	103

Result

The query was executed and the output was successfully obtained.

MONGO DB

Experiment No.: 14

27-06-2023

Aim

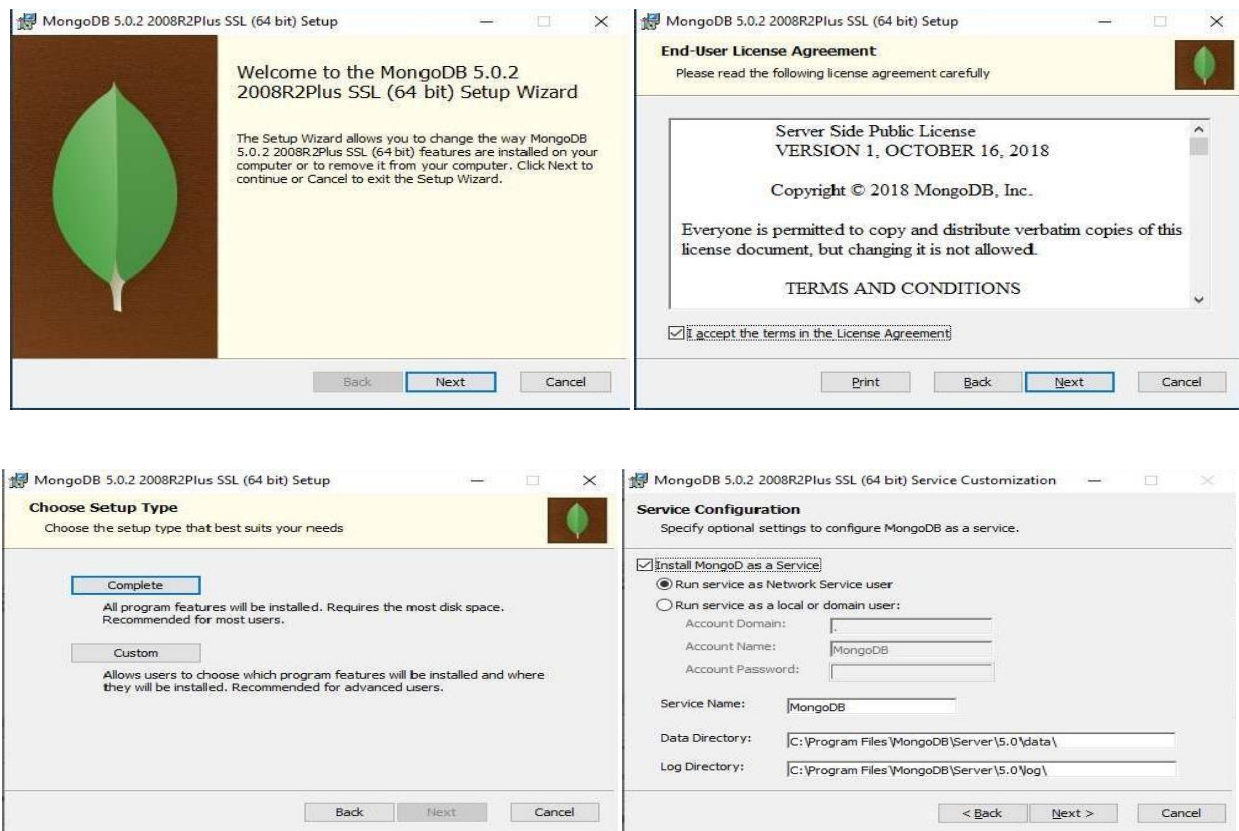
Installation and configuration of NoSQL Database - MongoDB

CO

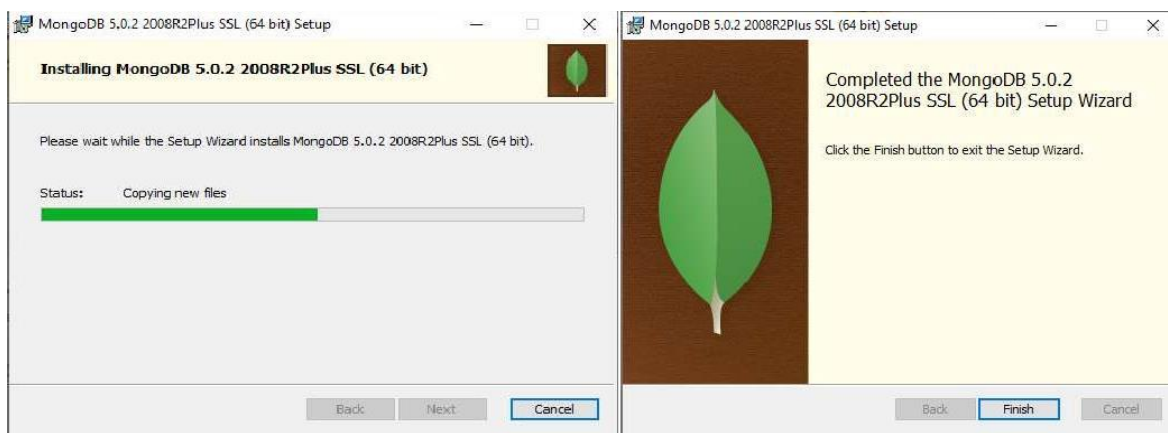
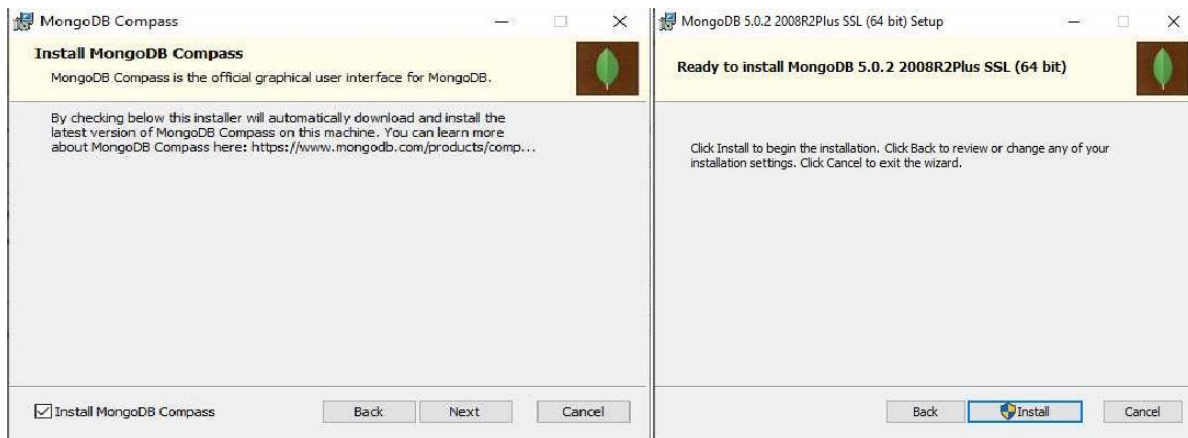
CO3: Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

Procedure

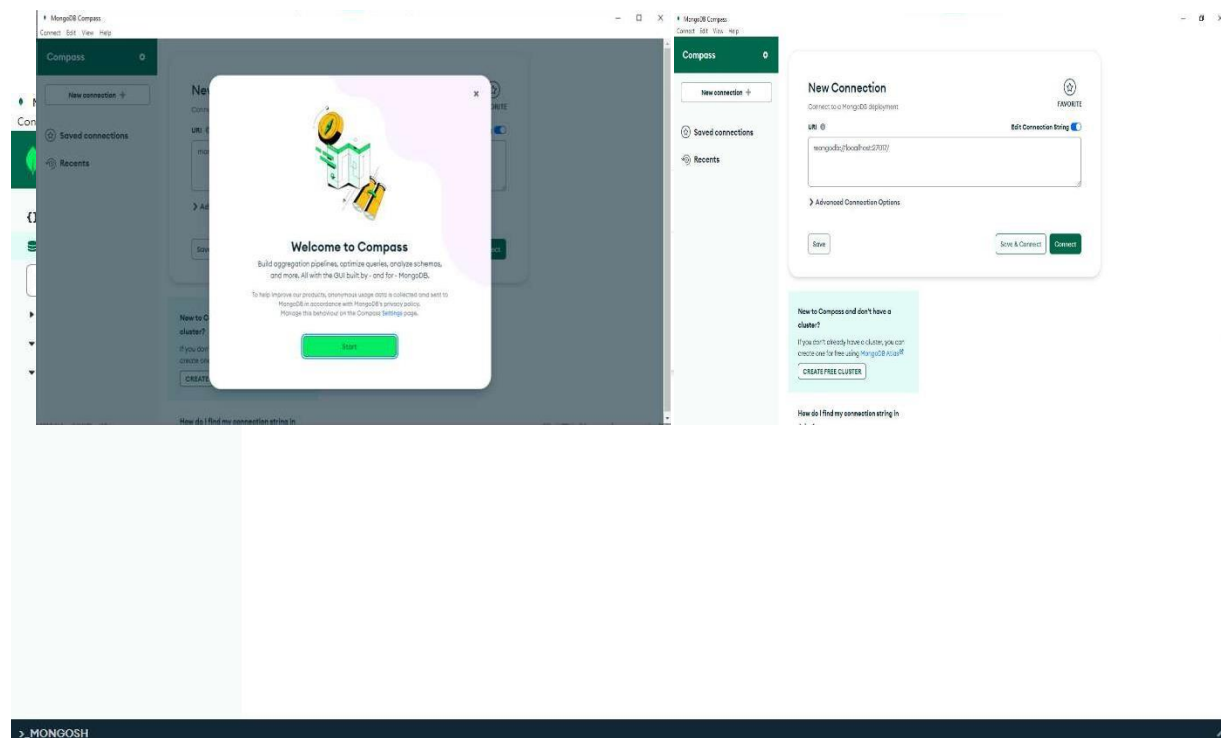
1. Setup Wizard



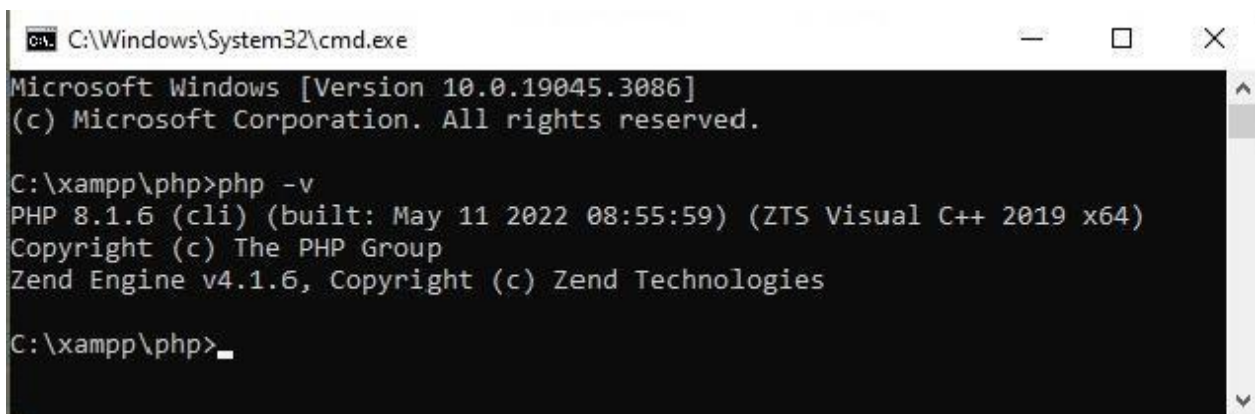
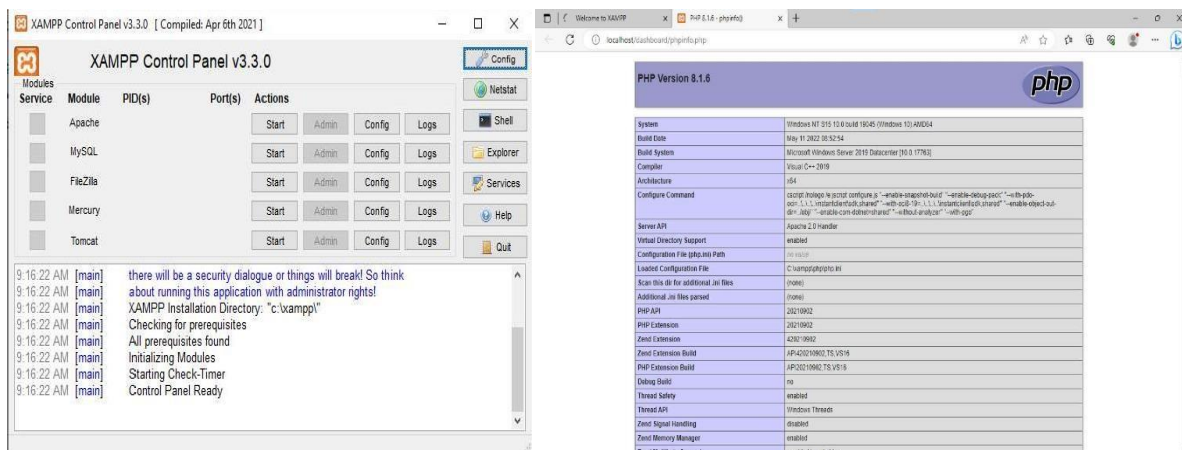
2. Installing MongoDB Compass



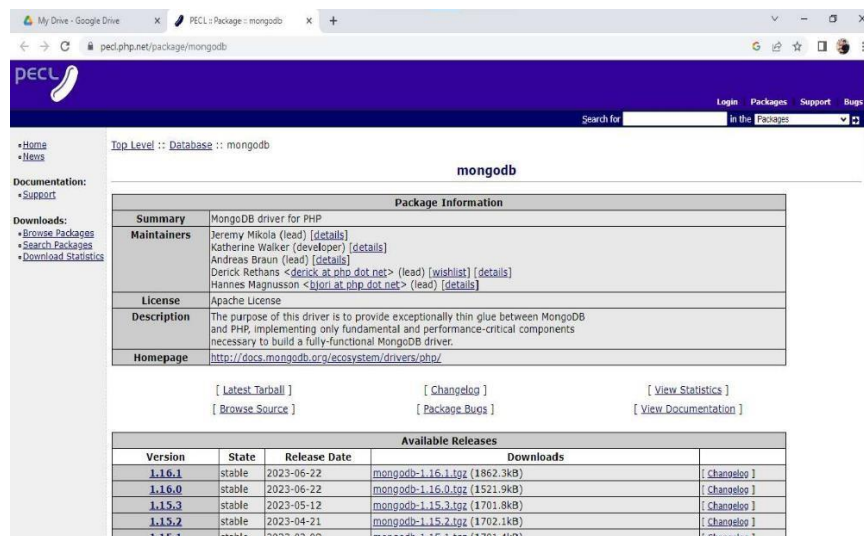
3. MongoDB Compass Home Page



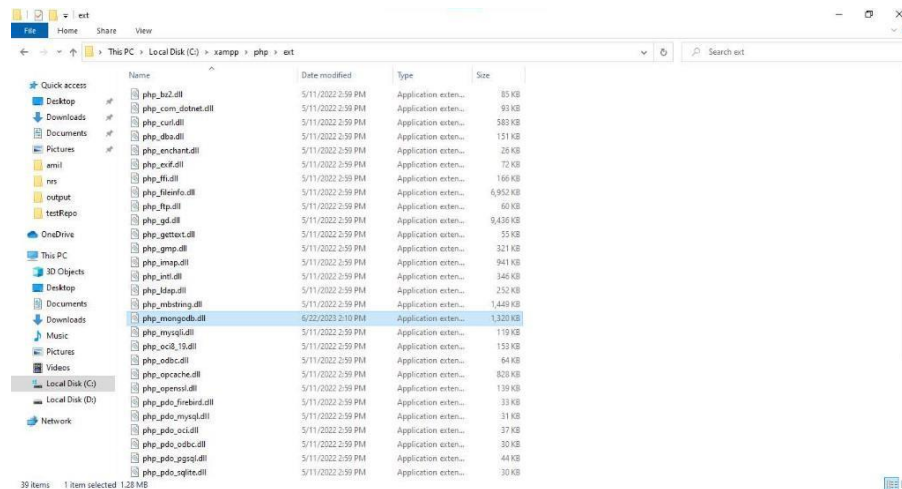
4. Configuring MongoDB to Connect to PHP



5. Installing MongoDB Package for PHP



6. Configuring Apache



Apache (httpd.conf)

Apache (httpd-ssl.conf)

Apache (httpd-xampp.conf)

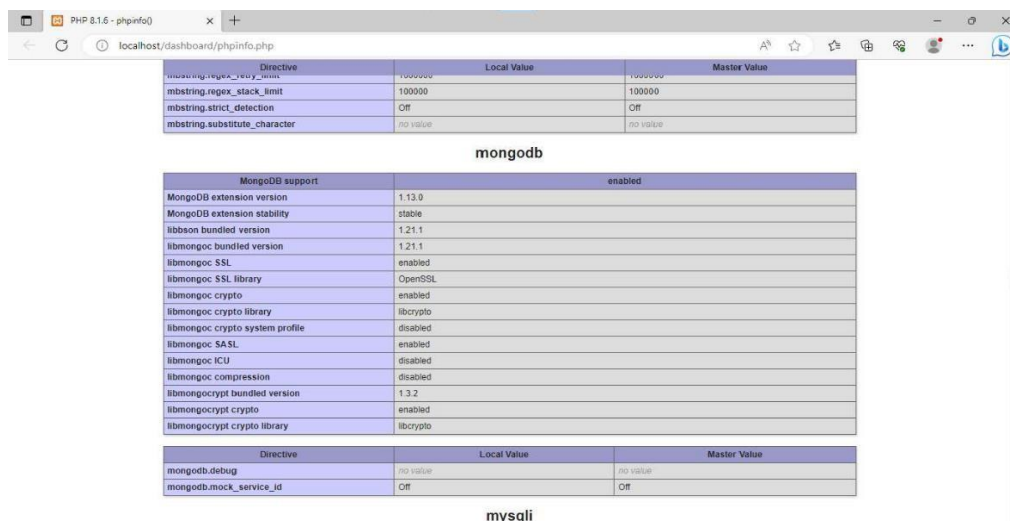
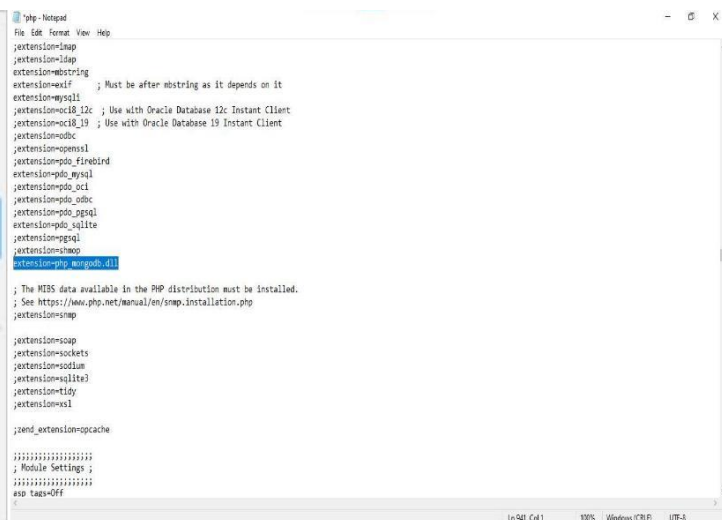
PHP (php.ini)

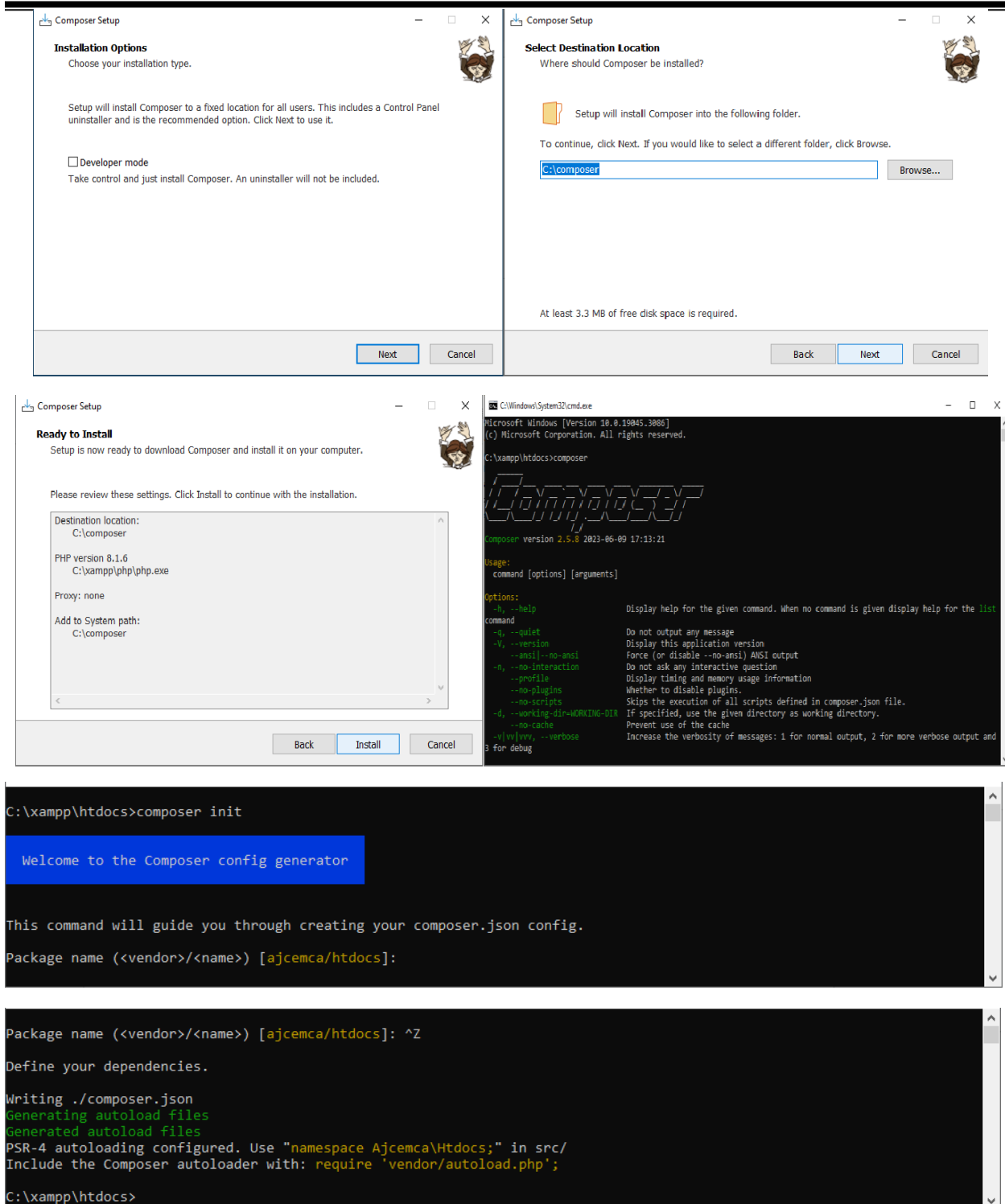
phpMyAdmin (config.inc.php)

<Browse> [Apache]

<Browse> [PHP]

<Browse> [phpMyAdmin]





The image displays the Composer Setup process in four stages:

- Installation Options:** The window shows the 'Developer mode' checkbox is unchecked. The text states: '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.' The 'Next' button is highlighted.
- Select Destination Location:** The window asks 'Where should Composer be installed?'. The text says: 'Setup will install Composer into the following folder. To continue, click Next. If you would like to select a different folder, click Browse.' The text box contains 'C:\composer' and the 'Next' button is highlighted.
- Ready to Install:** The window shows a summary of settings: Destination location: C:\composer, PHP version 8.1.6, Proxy: none, Add to System path: C:\composer. The 'Install' button is highlighted.
- Command Prompt:** The terminal shows the command 'C:\xampp\htdocs>composer init' and the output of the 'composer init' command, which generates a 'composer.json' file. The output includes: 'Welcome to the Composer config generator', 'This command will guide you through creating your composer.json config.', 'Package name (<vendor>/<name>) [ajcemca/htdocs]:', '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";', and '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 query was executed and the output was successfully obtained.

CRUD OPERATIONS

Experiment No.: 15

10-07-2023

Aim

NoSQL Operations - Build sample collections/documents to perform query operations

CO

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

Procedure

1. Create/Use a database

> use expmongo



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

2. Display current database

>db



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

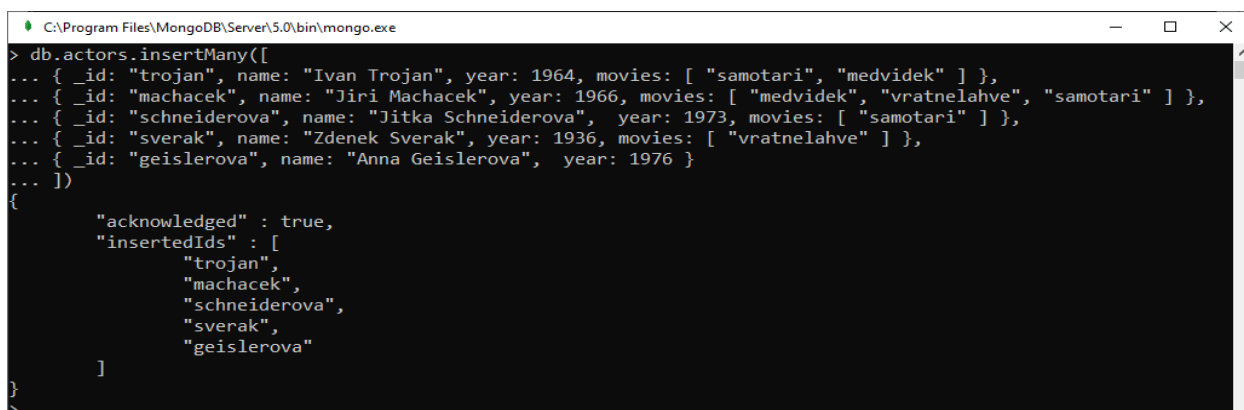
3. Create a collection

>db.createCollection("actors")



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

4. Insert data into the collection



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

5. Display documents in collection

>db.actors.find()

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

6. Display documents in collection

>db.actors.find({ })

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

7. Display

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

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

8. Display

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

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

9. Display

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

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

10. Display

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

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

11. Display

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


```

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

```

12. Display

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

```

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

```

13. Display

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

```

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

```

14. Display

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

```

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

```

15. Display

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

```

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

```

16. Display

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

```

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

```

17. Display

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

```

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

```


18. Display

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

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

19. Display

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

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

20. Display

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

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

21. Display

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

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

Result

The query was executed and the output was successfully obtained.

SHELL COMMANDS

Experiment No.: 16

11-07-2023

Aim

Build sample collections / documents to perform the shell commands.

CO

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

Procedure

1. Create and Use a Database

```
> use indexdemo;
switched to db indexdemo
> _
```

2. Show Databases

```
> 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

```
> 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

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

5. Create descending index on a field

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

6. Create index with the index name

```
> 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>,

```
> 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

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

9. Insert data

```
> db.scores.insertMany([
... { "_id" : ObjectId("523b6e32fb408eea0eec2647"), "userid" : "newbie" },
... { "_id" : ObjectId("523b6e61fb408eea0eec2648"), "userid" : "abby", "score" : 82 },
... { "_id" : ObjectId("523b6e6fffb408eea0eec2649"), "userid" : "nina", "score" : 90 } ]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("523b6e32fb408eea0eec2647"),
    ObjectId("523b6e61fb408eea0eec2648"),
    ObjectId("523b6e6fffb408eea0eec2649")
  ]
}
>
```

10. Create a sparse index

```
> db.scores.createIndex( { score: 1 }, { sparse: true } );
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
>
```

11. Use Sparse index on the score field

```
> db.scores.find( { score: { $lt: 90 } } );
{ "_id" : ObjectId("523b6e61fb408eea0eec2648"), "userid" : "abby", "score" : 82 }
>
```

12. Drop all indexes but _id index

```
> db.collection.dropIndexes();
{
  "nIndexesWas" : 3,
  "msg" : "non-_id indexes dropped for collection",
  "ok" : 1
}
>
```

Result

The query was executed and the output was successfully obtained.

APPLICATION

Experiment No.: 17

24-07-2023

Aim

Develop sample applications using any of the front-end tools and NoSQL

CO

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

Procedure

Index.php

```
<?php
require_once __DIR__ . '/vendor/autoload.php';
$client = new MongoDB\Client('mongodb://localhost:27017');
echo $client;

$db = $client->cars;

// $collection = $db->createCollection("rentalcustomer");
$collection = $db->rentalcustomer;

if (isset($_POST['is_submit'])) {
    try {
        $insertOneResult = $collection->insertOne([

            'id' => $_POST['id'],
            'name' => $_POST['name'],
            'mobno' => $_POST['mobno'],
            'address' => $_POST['address'],
            'cartype' => $_POST['cartype'],
            'price' => $_POST['price']
        ]);
    }
```

```
        if ($insertOneResult->getInsertedCount() > 0) {  
            echo '<script>alert("Inserted Successfully!");</script>';  
  
        }  
  
    } catch (Exception $e) {  
        echo 'Error: ' . $e->getMessage();  
    }  
}  
?>  
  
<html>  
<head>  
  
<link rel="stylesheet" href="style.css">  
  
<title>SignIn</title>  
</head>  
<body>  
  
<form method="post" action="#">  
    <div id="main" style="padding-left:5px;padding-right:5px;padding-bottom:5px;border-radius:3px" >  
        <div class="h-tag">  
            <center><h2 style="margin-top:150px;color:green;padding-top:10px">VROOM  
VROOM!!</h2></center>  
  
        </div>  
  
        <div class="login">  
            <table cellpadding="2" cellspacing="2" align="center" border="1">  
                <tr>
```

```

<td align="right">Id:</td>
<td><input type="text" class="tb" name="id"/></td>
</tr>
<tr>
<td align="right">Customer Name :</td>
<td><input type="text" class="tb" name="name"/></td>
</tr>
<tr>
<td align="right">Mob No :</td>
<td><input type="number" placeholder="+91 0000000000" class="tb" name="mobno"/></td>
</tr>
<tr>
<td align="right">Address :</td>
<td><input type="text" class="tb" name="address"/></td>
</tr>
<TR align="RIGHT">

<TD align="RIGHT">
<label for="cartype"> CarType: </label>
<select name ="cartype">
<option value="Select">Select</option>
<option value="BUGATTI">Bugatti</option>
<option value="BMW X5">Bmw X5</option>
<option value="GOLF GTI">GOLF GTI</option>
</TD>
</TR>
</select>
</tr>
<tr>
<td align="right">Price :</td>

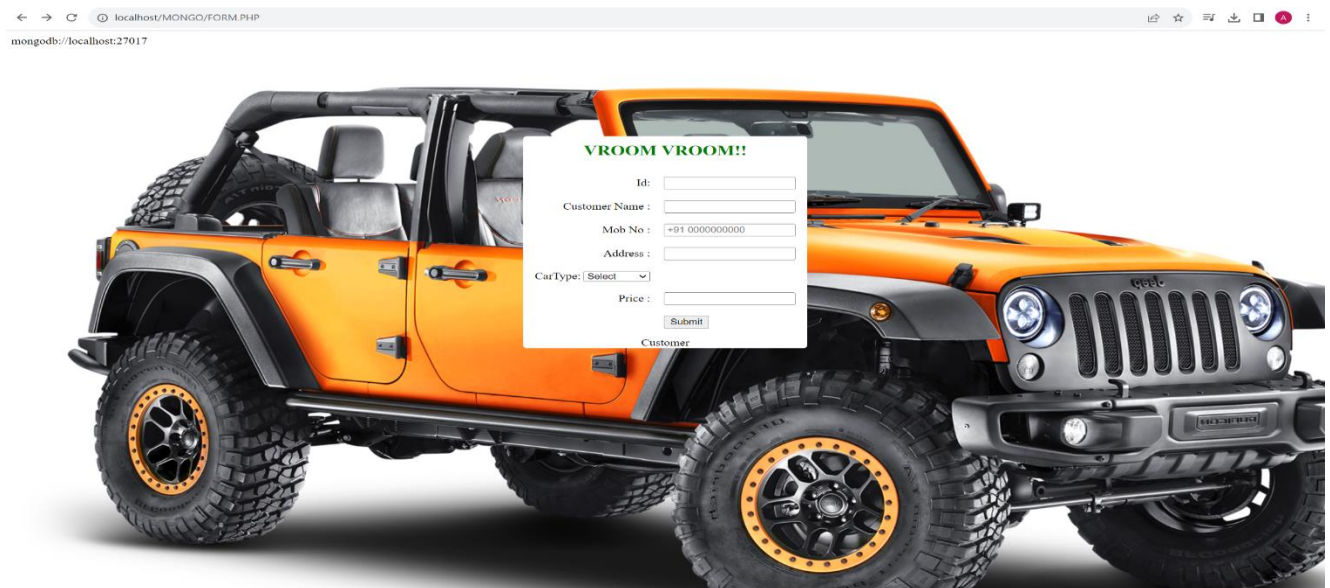
```

```

<td><input type="number" class="tb" name="price"/></td>
</tr>
<tr>
<td></td>
<td>
<input type="submit" value="Submit" class="btn" name="is_submit" /></td>
</tr>
</table>
</div>
</div>
<center><a href="view.php" style="text-decoration:none;color:black">Customer</Details></a></center>

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

```



View.php

```
<?php
require_once __DIR__ . '/vendor/autoload.php';

$client = new MongoDB\Client('mongodb://localhost:27017');
$db = $client->cars;
$collection = $db->rentalcustomer;

?>

<?php
$rentalcustomers = $collection->find();

?>

<html>
<head>
<title>View Customer</title>
<style>
    body {
        background-image:
            url("bg1.png");
    }
    table {
        border: 1px solid black;

        border-collapse: collapse;
        margin: auto;
    }
```



```
th, td {  
    border: 1px solid black;  
    padding: 8px;  
    text-align: center;  
}  
  
th {  
    background-color: #f2f2f2;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div style="text-align: center;">
```

```
<h2>Customer Details</h2>
```

```
<table>
```

```
<tr>
```

```
<th>Name</th>
```

```
<th>Mob No</th>
```

```
<th>Address</th>
```

```
<th>cartype</th>
```

```
<th>Price</th>
```

```
<th>Update</th>
```

```
<th>Delete</th>
```

```
</tr>
```

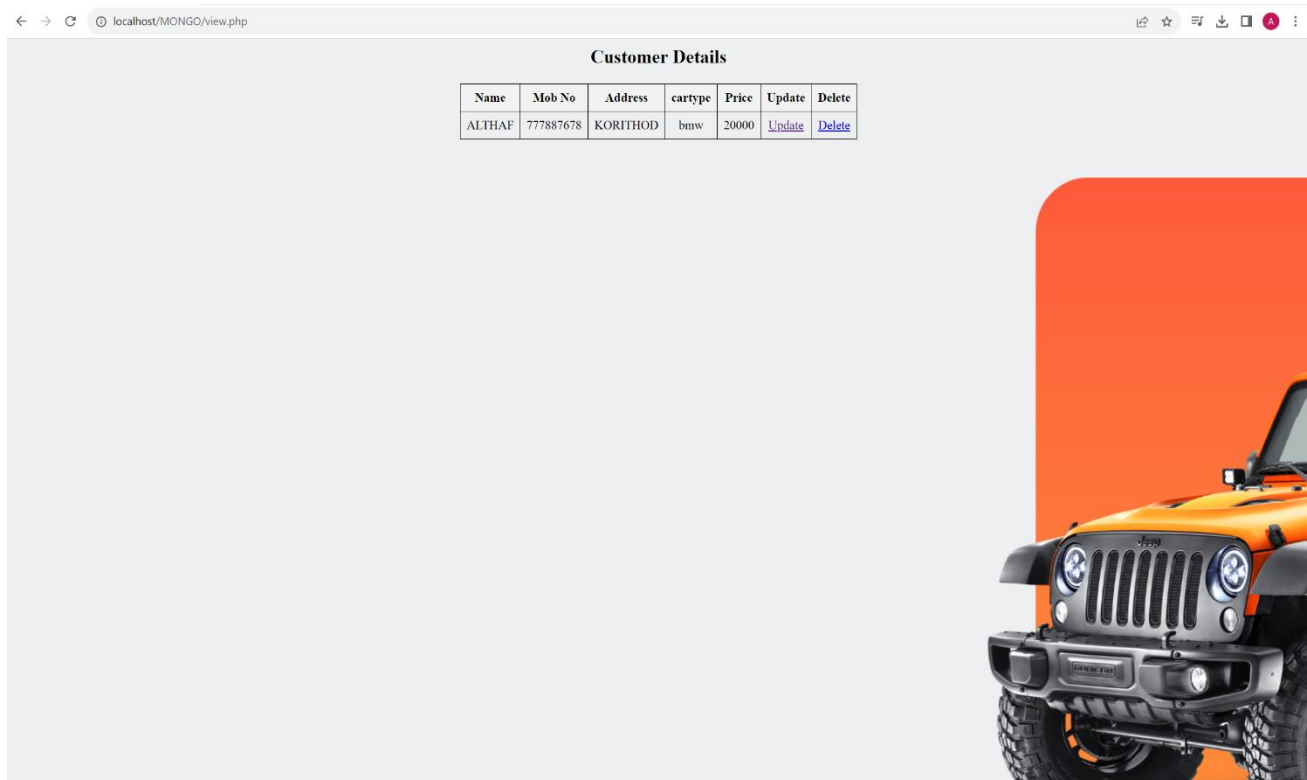
```
<?php foreach ($rentalcustomers as $rentalcustomer) : ?>
```

```

<tr>
    <td><?php echo $rentalcustomer['name']; ?></td>
    <td><?php echo $rentalcustomer['mobno']; ?></td>
    <td><?php echo $rentalcustomer['address']; ?></td>
    <td><?php echo $rentalcustomer['cartype']; ?></td>
    <td><?php echo $rentalcustomer['price']; ?></td>
    <td><a href="update.php?id=<?php echo $rentalcustomer['_id']; ?>">Update</a></td>
    <td><a href="delete.php?id=<?php echo $rentalcustomer['_id']; ?>">Delete</a></td>
</tr>

<?php endforeach; ?>
</table>
</div>
</body>
</html>

```



Update.php

```
<?php
require_once __DIR__ . '/vendor/autoload.php';

$client = new MongoDB\Client('mongodb://localhost:27017');
$db = $client->cars;
$collection = $db->rentalcustomer;

// Check if the form is submitted
if (isset($_POST['is_submit'])) {
    try {
        // Get the patient ID from the URL parameter
        $rentalcustomerId = $_GET['id'];

        // Update the patient's information in the database
        $updateResult = $collection->updateOne(
            ['_id' => new MongoDB\BSON\ObjectID($rentalcustomerId)],
            ['$set' => [
                'name' => $_POST['name'],
                'mobno' => $_POST['mobno'],
                'address' => $_POST['address'],
                'cartype' => $_POST['cartype'],
                'price' => $_POST['price']
            ]]
        );

        if ($updateResult->getModifiedCount() > 0) {
            echo '<script>alert("Customer information updated successfully!");</script>';
        } else {
```

```

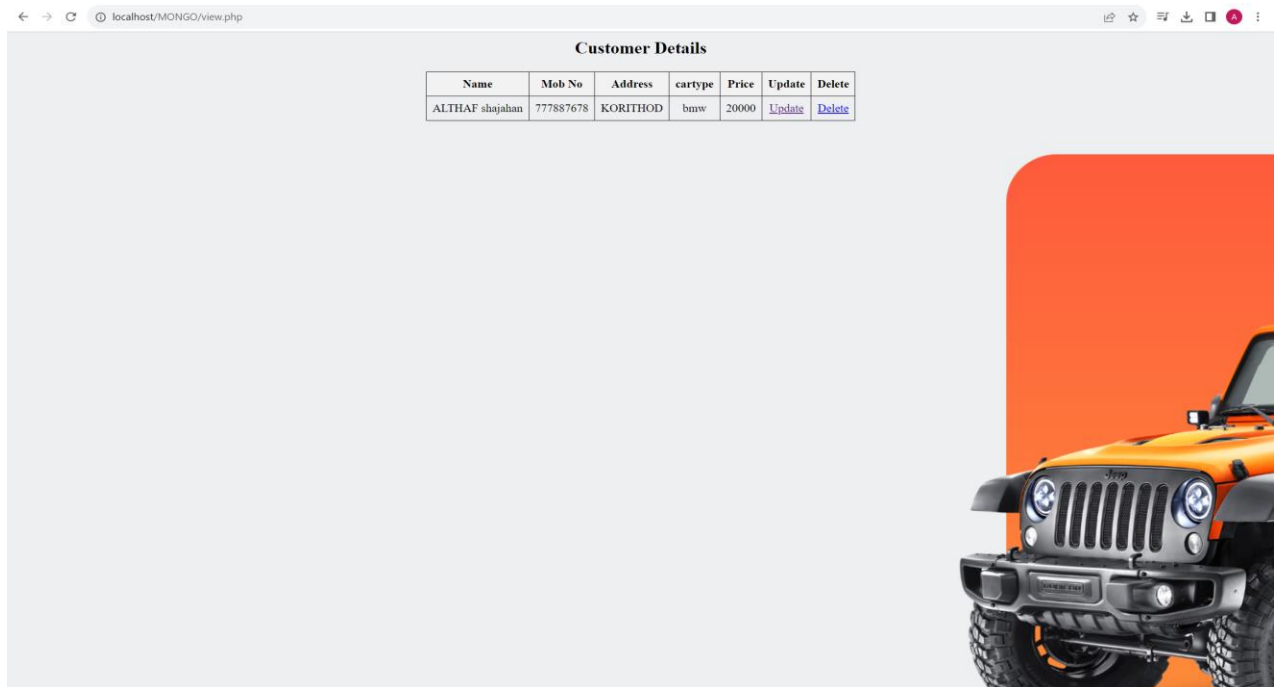
        echo '<script>alert("No changes made to Customer information.");</script>';
    }
} catch (Exception $e) {
    echo 'Error: ' . $e->getMessage();
}
}

// Retrieve the patient's information
$rentalcustomerId = $_GET['id'];
$rentalcustomer = $collection->findOne(['_id' => new MongoDB\BSON\ObjectID($rentalcustomerId)]);
?>

<html>
<head>
<title>Update Customer</title>
</head>
<body>
<h2>Update Customer Information</h2>
<form method="post">
    <table>
        <tr>
            <td>Name:</td>
            <td><input type="text" name="name" value="<?php echo $rentalcustomer['name']; ?>"></td>
        </tr>
        <tr>
            <td>Mob No:</td>
            <td><input type="text" name="mobno" value="<?php echo $rentalcustomer['mobno']; ?>"></td>
        </tr>
    </table>

```

```
<tr>
<td>Address:</td>
<td><input type="text" name="address" value="<?php echo $rentalcustomer['address']; ?>"></td>
</tr>
<tr>
<td>Cartype:</td>
<td><input type="text" name="cartype" value="<?php echo $rentalcustomer['cartype']; ?>"></td>
</tr>
<tr>
<td>Price:</td>
<td><input type="text" name="price" value="<?php echo $rentalcustomer['price']; ?>"></td>
</tr>
<tr>
<td></td>
<td><input type="submit" value="Update" name="is_submit"></td>
</tr>
</table>
</form>
<a href="view.php">Back</a>
</body>
</html>
```



Delete.php

```
<?php
```

```
require_once __DIR__ . '/vendor/autoload.php';
```

```
$client = new MongoDB\Client('mongodb://localhost:27017');
```

```
$db = $client->cars;
```

```
$collection = $db->rentalcustomer;
```

```
// Check if the patient ID is provided in the URL
```

```
if (isset($_GET['id'])) {
```

```
    try {
```

```
        // Get the patient ID from the URL parameter
```

```
        $rentalcustomerId = $_GET['id'];
```

```
        // Delete the patient's information from the database
```

```
$deleteResult = $collection->deleteOne(['_id' => new
MongoDB\BSON\ObjectID($rentalcustomerId)]);
if ($deleteResult->getDeletedCount() > 0) {
    echo '<script>alert("customer information deleted successfully!");</script>';
} else {
    echo '<script>alert("customer not found.");</script>';
}
} catch (Exception $e) {
    echo 'Error: ' . $e->getMessage();
}
}

// Redirect back to the view page after deletion
header('Location: view.php');

exit;

?>
```

