Muthoot Institute of Technology and Science

Varikoli P.O, Puthencruz,

Ernakulam – 682308

Master of Computer Applications

LAB RECORD

Course Name: 20MCA134 ADVANCED DBMS LAB

NAME.....

REG. NO..... SEMESTER.....

MONTH & YEAR.....

Muthoot
Institute of Technology & Science

Certificate

Certified that this is the Bonafide Record of Practical work done in the …… Lab of Muthoot Institute of Technology and Science by ***Name:*** *……. **Reg.No:** ……. for the partial fulfillment of the requirement for the award of the degree of Master of Computer Applications during the year……..*

Head of the Department Faculty in Charge

University Exam Reg. No……. of.....202….

Date of Examination…….

Internal Examiner External Examiner

Muthoot
Institute of Technology & Science

Vision of Institute

To be a centre of excellence for learning and research in engineering and technology, producing intellectually well-equipped and socially committed citizens possessing an ethical value system.

Mission of Institute

- Offer well-balanced programme of instruction, practical exercise and opportunities in technology.
- Foster innovation and ideation of technological solutions on sustainable basis.
- Nurture a value system in students and engender in them a spirit of inquiry.

INDEX

13	Compare relational and non-relational databases.	19-03-2024	46
14	To build a sample collections/documents to perform query operations. Create database college and collection students & insert student details into it.	26-03-2024	51
15	Create a Database ‘Student’ with the fields SRN, SName, degree, semester, CGPA and create a collection ‘Students’.	02-04-2024	53
16	Create an employee database with the fields: {eid, ename, dept, desig, salary, yoj, address {dno,street,locality,city}}	05-04-2024	61
17	Create a database named college and then create a collection named students. Insert some values into it.	09-04-2024	73
18	Create database ‘candidate’ and collection ‘details’.	12-04-2024	77
19	Create a database named college and then create a collection named studlist. Insert some values into it .	12-04-2024	80
20	Create a database in MongoDB named "mcadb" with collections named "course" and "students" and performaggregate functions, and regular expressions	16-04-2024	89
CO4	**Understand the basic storage architecture of distributed file systems**		
21	Build collections mcaDB documents students, courseand perform shell commands to create replica set, indexing etc	19-04-2024	93
CO5	**Design and deployment of NoSQL databases withreal time requirements.**		
22	Develop students’ marks calculation applicationsusing Python and MongoDB.	19-04-2024	94
23	Microproject	03-05-2024	97

Course Outcome 1

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

PROGRAM 1

AIM: Creation a database, S2MCA and Two tables, Department and Employee using DDL commands and perform insertion , selection using where condition , logical operation using DML commands.

1.Creates a new database named S1MCA.

QUERY

mysql>create database s1mca; Query
OK, 1 row affected (0.001 sec) mysql>
show databases;
mysql> show databases;

OUTPUT

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cdcol              |
| college            |
| college1           |
| db                 |
| insuretech         |
| library            |
| mysql              |
| performance_schema |
| phpmyadmin         |
| s1mca              |
| s2mca              |
| school             |
| test               |
| voting_system      |
| voting_systemnew   |
| webauth            |
+--------------------+
 17 rows in set (0.00 sec)
```

2.Selects a specific database to perform operations within.

QUERY

MariaDB [s1mca]> use s1mca;

OUTPUT

Database changed

3.Create a table to store employee information with primary key and foreign key constraints.

QUERY

mysql> create table employee(employee_id integer primary key,first_name varchar(20),last_name varchar(20),department_id integer,salary integer,foreign key(department_id) references department(department_id));
Query OK, 0 rows affected (0.28 sec)

mysql> desc employee;

OUTPUT

```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| employee_id   | int(11)     | NO   | PRI | NULL    |       |
| first_name    | varchar(20) | YES  |     | NULL    |       |
| last_name     | varchar(20) | YES  |     | NULL    |       |
| department_id | int(11)     | YES  | MUL | NULL    |       |
| salary        | int(11)     | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

5 rows in set (0.12 sec)

4. Create a table to store department information with primary key .

QUERY

mysql> create table department(department_id integer primary key,department_name varchar(20));
Query OK, 0 rows affected (0.41 sec) mysql>
desc department;

OUTPUT

```
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| department_id   | int(11)     | NO   | PRI | NULL    |       |
| department_name | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

2 rows in set (0.12 sec)

5. Insert data into the department table.

QUERY

```
mysql>  insert into department
values(1,'hr'),(2,'finance'),(3,'it'),(4,'marketing'),(5,'operations'),(6,'operat
ion'),(7,'research'),(8,'engineering'),(9,'customer
service'),(10,'administration'),(11,'logistics'),(12,'quality
control'),(13,'production'),(14,'distribution'),(15,'legal'),(16,'purchasing'),(1 7,'public
relations'),(18,'advertising'),(19,'human resources'),(20,'information technology');
Query OK, 20 rows affected, 1 warning (0.11 sec)
Records: 20  Duplicates: 0  Warnings: 1 mysql>
select *from department;
```

OUTPUT

```
+---------------+----------------------+
| department_id | department_name      |
+---------------+----------------------+
|             1 | hr                   |
|             2 | finance              |
|             3 | it                   |
|             4 | marketing            |
|             5 | operations           |
|             6 | operat ion           |
|             7 | research             |
|             8 | engineering          |
|             9 | customer service     |
|            10 | administration       |
|            11 | logistics            |
|            12 | quality control      |
|            13 | production           |
|            14 | distribution         |
|            15 | legal                |
|            16 | purchasing           |
|            17 | public relations     |
|            18 | advertising          |
|            19 | human resources      |
|            20 | information technolo |
+---------------+----------------------+
```

20 rows in set (0.00 sec)

6. Insert data into the employee table.

QUERY

```
mysql> insert into employee
values(1,'john','doe',1,50000),(2,'jane','smith',2,60000),(3,'alice','johnson',3,70000),(4,'bob','williams'
,1,55000),(5,'sarah','lee',4,62000),(6,'michael','brown',3,72000),(7,'lisa','taylor',2,65000),(8,'kevin','cl
ark',1,58000),(9,'amanda','martinez',4,60000),(10,'eic','anderson',3,75000),(11,'emily','wilson',2,5800
0),(12,'ryan','garcia',3,67000),(13,'samantha','martinez',1,56000),(14,'david','lee',4,64000),(15,'jessic
a','brown',3,69000),(16,'andrew','johnson',2,62000),(17,'lauren','white',1,57000),(18,'christopher','lop
ez',4,61000),(19,'kimberly','young',3,73000),(20,'matthew','hall',2,64000);
Query OK, 20 rows affected (0.20 sec)
Records: 20  Duplicates: 0  Warnings: 0
mysql> select *from employee;
```

OUTPUT

employee_id	first_name	last_name	department_id	salary
1	john	doe	1	50000
2	jane	smith	2	60000
3	alice	johnson	3	70000
4	bob	williams	1	55000
5	sarah	lee	4	62000
6	michael	brown	3	72000
7	lisa	taylor	2	65000
8	kevin	clark	1	58000
9	amanda	martinez	4	60000
10	eic	anderson	3	75000
11	emily	wilson	2	58000
12	ryan	garcia	3	67000
13	samantha	martinez	1	56000
14	david	lee	4	64000
15	jessica	brown	3	69000
16	andrew	johnson	2	62000
17	lauren	white	1	57000
18	christopher	lopez	4	61000
19	kimberly	young	3	73000
20	matthew	hall	2	64000

20 rows in set (0.00 sec)

7. Use DISTINCT to select unique department names.

QUERY

```
mysql> select distinct department_name from department;
```

OUTPUT

department_name
hr
finance
it
marketing
operations
operation
research
engineering
customer service
administration
logistics
quality control
production
distribution
legal
purchasing
public relations
advertising
human resources
information technolo

20 rows in set (0.06 sec)

8. Select all columns from the employee table.

QUERY

mysql> select *from employee;

OUTPUT

employee_id	first_name	last_name	department_id	salary
1	john	doe	1	50000
2	jane	smith	2	60000
3	alice	johnson	3	70000
4	bob	williams	1	55000
5	sarah	lee	4	62000
6	michael	brown	3	72000
7	lisa	taylor	2	65000
8	kevin	clark	1	58000
9	amanda	martinez	4	60000
10	eic	anderson	3	75000
11	emily	wilson	2	58000
12	ryan	garcia	3	67000
13	samantha	martinez	1	56000
14	david	lee	4	64000
15	jessica	brown	3	69000
16	andrew	johnson	2	62000
17	lauren	white	1	57000
18	christopher	lopez	4	61000
19	kimberly	young	3	73000
20	matthew	hall	2	64000

20 rows in set (0.00 sec)

9. Select specific columns (first_name, last_name) from the employee table.

QUERY

mysql> select first_name,last_name from employee;

OUTPUT

```
+-------------+-----------+
| first_name  | last_name |
+-------------+-----------+
| john        | doe       |
| jane        | smith     |
| alice       | johnson   |
| bob         | williams  |
| sarah       | lee       |
| michael     | brown     |
| lisa        | taylor    |
| kevin       | clark     |
| amanda      | martinez  |
| eic         | anderson  |
| emily       | wilson    |
| ryan        | garcia    |
| samantha    | martinez  |
| david       | lee       |
| jessica     | brown     |
| andrew      | johnson   |
| lauren      | white     |
| christopher | lopez     |
| kimberly    | young     |
| matthew     | hall      |
+-------------+-----------+
```

20 rows in set (0.01 sec)

10. Select employees earning more than $60,000.

QUERY

mysql> select *from employee where salary>60000;

OUTPUT

```
+-------------+-------------+-----------+---------------+--------+
| employee_id | first_name  | last_name | department_id | salary |
+-------------+-------------+-----------+---------------+--------+
|           3 | alice       | johnson   |             3 |  70000 |
|           5 | sarah       | lee       |             4 |  62000 |
|           6 | michael     | brown     |             3 |  72000 |
|           7 | lisa        | taylor    |             2 |  65000 |
|          10 | eic         | anderson  |             3 |  75000 |
|          12 | ryan        | garcia    |             3 |  67000 |
|          14 | david       | lee       |             4 |  64000 |
|          15 | jessica     | brown     |             3 |  69000 |
|          16 | andrew      | johnson   |             2 |  62000 |
|          18 | christopher | lopez     |             4 |  61000 |
|          19 | kimberly    | young     |             3 |  73000 |
|          20 | matthew     | hall      |             2 |  64000 |
+-------------+-------------+-----------+---------------+--------+
```

12 rows in set (0.05 sec)

11. Select employees in the HR department(dept_id =1)

QUERY

```
mysql> select *from employee where department_id=1;
```

OUTPUT

```
+-------------+------------+-----------+---------------+--------+
| employee_id | first_name | last_name | department_id | salary |
+-------------+------------+-----------+---------------+--------+
|           1 | john       | doe       |             1 |  50000 |
|           4 | bob        | williams  |             1 |  55000 |
|           8 | kevin      | clark     |             1 |  58000 |
|          13 | samantha   | martinez  |             1 |  56000 |
|          17 | lauren     | white     |             1 |  57000 |
+-------------+------------+-----------+---------------+--------+
```

5 rows in set (0.02 sec)

12. Add a new column (hired) to the employee table and insert values.

QUERY

```
mysql> alter table employee add column hired date;
```

OUTPUT

```
Query OK, 0 rows affected (0.83 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc employee;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| employee_id   | int(11)     | NO   | PRI | NULL    |       |
| first_name    | varchar(20) | YES  |     | NULL    |       |
| last_name     | varchar(20) | YES  |     | NULL    |       |
| department_id | int(11)     | YES  | MUL | NULL    |       |
| salary        | int(11)     | YES  |     | NULL    |       |
| hired         | date        | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

6 rows in set (0.02 sec)

QUERY

```
mysql> update employee set hired='23-7-4' where employee_id <=5;
mysql> update employee set hired='23-7-4' where employee_id >5 and employee_id<15;
mysql> update employee set hired='22-7-4' where employee_id >14 and employee_id<21;
```

mysql> select *from employee;

OUTPUT

employee_id	first_name	last_name	department_id	salary	hired
1	john	doe	1	50000	2023-07-04
2	jane	smith	2	60000	2023-07-04
3	alice	johnson	3	70000	2023-07-04
4	bob	williams	1	55000	2023-07-04
5	sarah	lee	4	62000	2023-07-04
6	michael	brown	3	72000	2023-07-04
7	lisa	taylor	2	65000	2023-07-04
8	kevin	clark	1	58000	2023-07-04
9	amanda	martinez	4	60000	2023-07-04
10	eic	anderson	3	75000	2023-07-04
11	emily	wilson	2	58000	2023-07-04
12	ryan	garcia	3	67000	2023-07-04
13	samantha	martinez	1	56000	2023-07-04
14	david	lee	4	64000	2023-07-04
15	jessica	brown	3	69000	2022-07-04
16	andrew	johnson	2	62000	2022-07-04
17	lauren	white	1	57000	2022-07-04
18	christopher	lopez	4	61000	2022-07-04
19	kimberly	young	3	73000	2022-07-04
20	matthew	hall	2	64000	2022-07-04

20 rows in set (0.00 sec)

13. Select employees hired after January 1, 2023.

QUERY

mysql> select *from employee where hired > '2023-01-01';

OUTPUT

employee_id	first_name	last_name	department_id	salary	hired
1	john	doe	1	50000	2023-07-04
2	jane	smith	2	60000	2023-07-04
3	alice	johnson	3	70000	2023-07-04
4	bob	williams	1	55000	2023-07-04
5	sarah	lee	4	62000	2023-07-04
6	michael	brown	3	72000	2023-07-04
7	lisa	taylor	2	65000	2023-07-04
8	kevin	clark	1	58000	2023-07-04
9	amanda	martinez	4	60000	2023-07-04
10	eic	anderson	3	75000	2023-07-04
11	emily	wilson	2	58000	2023-07-04
12	ryan	garcia	3	67000	2023-07-04
13	samantha	martinez	1	56000	2023-07-04
14	david	lee	4	64000	2023-07-04

14 rows in set (0.00 sec)

14. Write a query to retrieve all employees with a salary greater than $50,000.

QUERY

```
mysql> select * from employee where salary>50000;
```

OUTPUT

employee_id	first_name	last_name	department_id	salary	hired
2	jane	smith	2	60000	2023-07-04
3	alice	johnson	3	70000	2023-07-04
4	bob	williams	1	55000	2023-07-04
5	sarah	lee	4	62000	2023-07-04
6	michael	brown	3	72000	2023-07-04
7	lisa	taylor	2	65000	2023-07-04
8	kevin	clark	1	58000	2023-07-04
9	amanda	martinez	4	60000	2023-07-04
10	eic	anderson	3	75000	2023-07-04
11	emily	wilson	2	58000	2023-07-04
12	ryan	garcia	3	67000	2023-07-04
13	samantha	martinez	1	56000	2023-07-04
14	david	lee	4	64000	2023-07-04
15	jessica	brown	3	69000	2022-07-04
16	andrew	johnson	2	62000	2022-07-04
17	lauren	white	1	57000	2022-07-04
18	christopher	lopez	4	61000	2022-07-04
19	kimberly	young	3	73000	2022-07-04
20	matthew	hall	2	64000	2022-07-04

19 rows in set (0.00 sec)

15.Find all employees whose department ID is not equal to 3.

QUERY

```
mysql> select * from employee where department_id!=3;
```

OUTPUT

employee_id	first_name	last_name	department_id	salary	hired
1	john	doe	1	50000	2023-07-04
2	jane	smith	2	60000	2023-07-04
4	bob	williams	1	55000	2023-07-04
5	sarah	lee	4	62000	2023-07-04
7	lisa	taylor	2	65000	2023-07-04
8	kevin	clark	1	58000	2023-07-04
9	amanda	martinez	4	60000	2023-07-04
11	emily	wilson	2	58000	2023-07-04
13	samantha	martinez	1	56000	2023-07-04
14	david	lee	4	64000	2023-07-04
16	andrew	johnson	2	62000	2022-07-04
17	lauren	white	1	57000	2022-07-04
18	christopher	lopez	4	61000	2022-07-04
20	matthew	hall	2	64000	2022-07-04

14 rows in set (0.00 sec)

16.Retrieve employees with a salary greater than $50,000 and who belong to department ID 2.

QUERY

mysql> select * from employee where salary>50000 and department_id=2;

OUTPUT

```
+-------------+------------+-----------+---------------+--------+------------+
| employee_id | first_name | last_name | department_id | salary | hired      |
+-------------+------------+-----------+---------------+--------+------------+
|           2 | jane       | smith     |             2 |  60000 | 2023-07-04 |
|           7 | lisa       | taylor    |             2 |  65000 | 2023-07-04 |
|          11 | emily      | wilson    |             2 |  58000 | 2023-07-04 |
|          16 | andrew     | johnson   |             2 |  62000 | 2022-07-04 |
|          20 | matthew    | hall      |             2 |  64000 | 2022-07-04 |
+-------------+------------+-----------+---------------+--------+------------+
```

5 rows in set (0.00 sec)

17. Select all employees who belong to department ID 1 or 2.

QUERY

mysql> select * from employee where department_id in (1,2);

OUTPUT

```
+-------------+------------+-----------+---------------+--------+------------+
| employee_id | first_name | last_name | department_id | salary | hired      |
+-------------+------------+-----------+---------------+--------+------------+
|           1 | john       | doe       |             1 |  50000 | 2023-07-04 |
|           2 | jane       | smith     |             2 |  60000 | 2023-07-04 |
|           4 | bob        | williams  |             1 |  55000 | 2023-07-04 |
|           7 | lisa       | taylor    |             2 |  65000 | 2023-07-04 |
|           8 | kevin      | clark     |             1 |  58000 | 2023-07-04 |
|          11 | emily      | wilson    |             2 |  58000 | 2023-07-04 |
|          13 | samantha   | martinez  |             1 |  56000 | 2023-07-04 |
|          16 | andrew     | johnson   |             2 |  62000 | 2022-07-04 |
|          17 | lauren     | white     |             1 |  57000 | 2022-07-04 |
|          20 | matthew    | hall      |             2 |  64000 | 2022-07-04 |
+-------------+------------+-----------+---------------+--------+------------+
```

10 rows in set (0.09 sec)

18. Write a query to fetch the first name, last name, and department name of employees from the “employees” and “departments” tables, joining them based on the department ID.

QUERY

mysql> SELECT e.first_name, e.last_name, d.department_name FROM employee e INNER JOIN department d ON e.department_id = d.department_id;

OUTPUT

```
+-------------+-----------+-----------------+
| first_name  | last_name | department_name |
+-------------+-----------+-----------------+
| john        | doe       | hr              |
| jane        | smith     | finance         |
| alice       | johnson   | it              |
| bob         | williams  | hr              |
| sarah       | lee       | marketing       |
| michael     | brown     | it              |
| lisa        | taylor    | finance         |
| kevin       | clark     | hr              |
| amanda      | martinez  | marketing       |
| eic         | anderson  | it              |
| emily       | wilson    | finance         |
| ryan        | garcia    | it              |
| samantha    | martinez  | hr              |
| david       | lee       | marketing       |
| jessica     | brown     | it              |
| andrew      | johnson   | finance         |
| lauren      | white     | hr              |
| christopher | lopez     | marketing       |
| kimberly    | young     | it              |
| matthew     | hall      | finance         |
+-------------+-----------+-----------------+
```

20 rows in set (0.11 sec)

19.Write a query to retrieve the first name, last name, and department name of all employees who belong to the "IT" department

QUERY

mysql> SELECT e.first_name, e.last_name, d.department_name FROM employee e INNER JOIN department d ON e.department_id = d.department_id WHERE d.department_name = 'IT';

OUTPUT

```
+------------+-----------+-----------------+
| first_name | last_name | department_name |
+------------+-----------+-----------------+
| alice      | johnson   | it              |
| michael    | brown     | it              |
| eic        | anderson  | it              |
| ryan       | garcia    | it              |
| jessica    | brown     | it              |
| kimberly   | young     | it              |
+------------+-----------+-----------------+
```

6 rows in set (0.00 sec)

20.Find employees who have the same salary.

QUERY

mysql> SELECT e1.first_name, e1.last_name, e1.salary FROM employee e1 INNER JOIN employee e2 ON e1.salary = e2.salary WHERE e1.employee_id <> e2.employee_id ORDER BY e1.salary;

OUTPUT

```
+------------+-----------+--------+
| first_name | last_name | salary |
+------------+-----------+--------+
| emily      | wilson    |  58000 |
| kevin      | clark     |  58000 |
| jane       | smith     |  60000 |
| amanda     | martinez  |  60000 |
| sarah      | lee       |  62000 |
| andrew     | johnson   |  62000 |
| david      | lee       |  64000 |
| matthew    | hall      |  64000 |
+------------+-----------+--------+
```

8 rows in set (0.00 sec)

21. Drop the salary column from the employee table.

QUERY

mysql> ALTER TABLE employee DROP COLUMN salary;
Query OK, 0 rows affected (0.54 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> SELECT * FROM employee;

OUTPUT

```
+--------+------------+-----------+---------+------------+
| emp_id | first_name | last_name | dept_id | hired      |
+--------+------------+-----------+---------+------------+
|      1 | John       | Doe       |       1 | 2014-03-16 |
|      2 | Jane       | Smith     |       2 | 2022-10-11 |
|      3 | Alice      | Johnson   |       3 | 2023-05-16 |
|      4 | Bob        | Williams  |       1 | 2014-03-16 |
|      5 | Sarah      | Lee       |       4 | 2023-09-12 |
|      6 | Michael    | Brown     |       3 | 2023-05-16 |
|      7 | Lisa       | Taylor    |       2 | 2022-10-11 |
|      8 | Kevin      | Clark     |       1 | 2014-03-16 |
|      9 | Amanda     | Martinez  |       4 | 2023-09-12 |
|     10 | Eric       | Anderson  |       3 | 2023-05-16 |
|     11 | Emily      | Wilson    |       2 | 2022-10-11 |
|     12 | Ryan       | Garcia    |       3 | 2023-05-16 |
|     13 | Samantha   | Martinez  |       1 | 2014-03-16 |
|     14 | David      | Lee       |       4 | 2023-09-12 |
|     15 | Jessica    | Brown     |       3 | 2023-05-16 |
|     16 | Andrew     | Johnson   |       2 | 2022-10-11 |
|     17 | Lauren     | White     |       1 | 2014-03-16 |
|     18 | Christopher| Lopez     |       4 | 2023-09-12 |
|     19 | Kimberly   | Young     |       3 | 2023-05-16 |
|     20 | Matthew    | Hall      |       2 | 2022-10-11 |
+--------+------------+-----------+---------+------------+
```

22. Delete a row from employee where department-id -4.

QUERY

```
MariaDB [s1mca]> delete from employee where dept_id=4;
Query OK, 4 rows affected (0.01 sec)
MariaDB [s1mca]> select*from employee;
```

OUTPUT

```
+--------+------------+-----------+---------+------------+
| emp_id | first_name | last_name | dept_id | hired      |
+--------+------------+-----------+---------+------------+
|      1 | John       | Doe       |       1 | 2014-03-16 |
|      2 | Jane       | Smith     |       2 | 2022-10-11 |
|      3 | Alice      | Johnson   |       3 | 2023-05-16 |
|      4 | Bob        | Williams  |       1 | 2014-03-16 |
|      6 | Michael    | Brown     |       3 | 2023-05-16 |
|      7 | Lisa       | Taylor    |       2 | 2022-10-11 |
|      8 | Kevin      | Clark     |       1 | 2014-03-16 |
|     10 | Eric       | Anderson  |       3 | 2023-05-16 |
|     11 | Emily      | Wilson    |       2 | 2022-10-11 |
|     12 | Ryan       | Garcia    |       3 | 2023-05-16 |
|     13 | Samantha   | Martinez  |       1 | 2014-03-16 |
|     15 | Jessica    | Brown     |       3 | 2023-05-16 |
|     16 | Andrew     | Johnson   |       2 | 2022-10-11 |
|     17 | Lauren     | White     |       1 | 2014-03-16 |
|     19 | Kimberly   | Young     |       3 | 2023-05-16 |
|     20 | Matthew    | Hall      |       2 | 2022-10-11 |
+--------+------------+-----------+---------+------------+
```

23. Delete employee table from the database

QUERY

```
MariaDB [s1mca]> drop table employee;
MariaDB [s1mca]> select*from employee;
```

OUTPUT

```
ERROR 1146 (42S02): Table 's1mca.employee' doesn't exist
```

24. Deletes an existing database (S2MCA).

QUERY

```
MariaDB [s1mca]> drop database s1mca;
Query OK, 1 row affected (0.01 sec)
MariaDB [s1mca]> show databases;
```

OUTPUT

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| book               |
| college_db         |
| coloryourfarm      |
| db                 |
| mysql              |
| performance_schema |
| school             |
| test               |
+--------------------+
```

PROGRAM 2

AIM: Perform Alter commands, Join operations, Update commands, order by clause, Like operator using DML commands

1. Create tables named Employee (with Emp_ID as primary key and Dept_ID as foreign key) and Department (with Dept_ID as primary key).

QUERY

CREATE TABLE employees (EmployeeID INT PRIMARY KEY,FirstName VARCHAR(50),LastName VARCHAR(50),Email VARCHAR(100),DepartmentID INT,FOREIGN KEY (DepartmentID) REFERENCES department(DepartmentID));

CREATE TABLE department (DepartmentID INT PRIMARY KEY,DepartmentName VARCHAR(50));

mysql> INSERT INTO department values(1,'HR'),(2,'Finance'),(3,'IT'),(4,'Marketing'),(5,'Sales'),(6,'Operations'),(7,'Research'),(8,'Development'),(9, 'Customer Service'),(10, 'Administration');

Query OK, 10 rows affected (0.05 sec)

Records: 10 Duplicates: 0 Warnings: 0

mysql> select *from department;

OUTPUT

```
+--------------+------------------+
| DepartmentID | DepartmentName   |
+--------------+------------------+
|            1 | HR               |
|            2 | Finance          |
|            3 | IT               |
|            4 | Marketing        |
|            5 | Sales            |
|            6 | Operations       |
|            7 | Research         |
|            8 | Development      |
|            9 | Customer Service |
|           10 | Administration   |
+--------------+------------------+
```

10 rows in set (0.01 sec)

QUERY

mysql> INSERT INTO employees values (101, 'John', 'Doe', 'john.doe@email.com', 1),(102, 'Jane', 'Smith', 'jane.smith@email.com', 2),(103, 'Robert', 'Johnson', 'robert.johnson@email.com', 1),(104, 'Mary', 'Jones', 'mary.jones@email.com', 3),(105, 'Michael', 'Brown', 'michael.brown@email.com', 4),(106, 'Jennifer', 'Davis', 'jennifer.davis@email.com', 5),(107, 'David', 'Martinez', 'david.martinez@email.com', 6),(108, 'Lisa', 'Rodriguez', 'lisa.rodriguez@email.com', 7),(109, 'William', 'Taylor', 'william.taylor@email.com', 8),(110, 'Sarah', 'Thomas', 'sarah.thomas@email.com', 9);
Query OK, 10 rows affected (0.22 sec)
Records: 10 Duplicates: 0 Warnings: 0

mysql> select *from employees;

OUTPUT

EmployeeID	FirstName	LastName	Email	DepartmentID
101	John	Doe	john.doe@email.com	1
102	Jane	Smith	jane.smith@email.com	2
103	Robert	Johnson	robert.johnson@email.com	1
104	Mary	Jones	mary.jones@email.com	3
105	Michael	Brown	michael.brown@email.com	4
106	Jennifer	Davis	jennifer.davis@email.com	5
107	David	Martinez	david.martinez@email.com	6
108	Lisa	Rodriguez	lisa.rodriguez@email.com	7
109	William	Taylor	william.taylor@email.com	8
110	Sarah	Thomas	sarah.thomas@email.com	9

10 rows in set (0.00 sec)

2. Add a new column named "Salary" to the Employee table with the data type DECIMAL (10,2).

QUERY

mysql> ALTER TABLE employees ADD COLUMN Salary DECIMAL(10,2);

Query OK, 0 rows affected (0.56 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> UPDATE employees SET Salary = 48000.00 WHERE DepartmentID BETWEEN 1 AND 4 OR DepartmentID BETWEEN 7 AND 9;

Query OK, 8 rows affected (0.06 sec)

Rows matched: 8 Changed: 8 Warnings: 0

mysql> UPDATE employees SET Salary = 50000.00 WHERE EmployeeID IN (106, 107);

Query OK, 2 rows affected (0.13 sec) Rows

matched: 2 Changed: 2 Warnings: 0

mysql> select *from employees;

OUTPUT

EmployeeID	FirstName	LastName	Email	DepartmentID	Salary
101	John	Doe	john.doe@email.com	1	48000.00
102	Jane	Smith	jane.smith@email.com	2	48000.00
103	Robert	Johnson	robert.johnson@email.com	1	48000.00
104	Mary	Jones	mary.jones@email.com	3	48000.00
105	Michael	Brown	michael.brown@email.com	4	48000.00
106	Jennifer	Davis	jennifer.davis@email.com	5	50000.00
107	David	Martinez	david.martinez@email.com	6	50000.00
108	Lisa	Rodriguez	lisa.rodriguez@email.com	7	48000.00
109	William	Taylor	william.taylor@email.com	8	48000.00
110	Sarah	Thomas	sarah.thomas@email.com	9	48000.00

10 rows in set (0.00 sec)

3.Alter the Department table to rename the column "DepartmentName" to "DeptName".

QUERY

mysql> ALTER TABLE department CHANGE DepartmentName DeptName varchar(50);

OUTPUT

Query OK, 0 rows affected (0.17 sec)
Records: 0 Duplicates: 0 Warnings: 0

4.Update the email of the employee with EmployeeID 102 to "jane.smith@example.com".

QUERY

mysql> UPDATE employees SET Email = 'jane.smith@example.com' WHERE EmployeeID = 102;

Query OK, 1 row affected (0.05 sec) Rows matched: 1 Changed: 1 Warnings: 0

mysql> select *from employees;

OUTPUT

EmployeeID	FirstName	LastName	Email	DepartmentID	Salary
101	John	Doe	john.doe@email.com	1	48000.00
102	Jane	Smith	jane.smith@example.com	2	48000.00
103	Robert	Johnson	robert.johnson@email.com	1	48000.00
104	Mary	Jones	mary.jones@email.com	3	48000.00
105	Michael	Brown	michael.brown@email.com	4	48000.00
106	Jennifer	Davis	jennifer.davis@email.com	5	50000.00
107	David	Martinez	david.martinez@email.com	6	50000.00
108	Lisa	Rodriguez	lisa.rodriguez@email.com	7	48000.00
109	William	Taylor	william.taylor@email.com	8	48000.00
110	Sarah	Thomas	sarah.thomas@email.com	9	48000.00

10 rows in set (0.00 sec)

5. Provide an example of an INNER JOIN between the Employee table and the Department table on the common column "DepartmentID".

QUERY

MariaDB [s1mca]> select e.Emp_ID, e.FirstName, e.LastName, e.Email, d.DeptName from Employee e inner join Department d on e.Dept_ID = d.Dept_ID;

OUTPUT

Emp_ID	FirstName	LastName	Email	DeptName
101	John	Doe	john.doe@email.com	HR
102	Jane	Smith	jane.smith@example.com	Finanace
103	Robert	Johnson	robert.johnson@email.com	HR
104	Mary	Jones	mary.jones@email.com	IT
105	Michael	Brown	michael.brown@email.com	Marketing
106	Jennifer	Davis	jennifer.davis@email.com	Sales
107	David	Martinez	davidmartiniz@gmail.com	operations
108	Lisa	Rodriguez	lisa.rodriguez@email.com	Reasearch
109	William	Taylor	william.taylor@email.com	Development
110	Sarah	Thomas	sarah.thomas@email.com	Customerservice

10 rows in set (0.01 sec)

6. Perform a LEFT JOIN between the Employee table and the Department table to display all employees regardless of whether they are assigned to a department or not.

QUERY

mysql> SELECT e.EmployeeID, e.FirstName, e.LastName, e.Email, d.DeptName, e.Salary, e.DepartmentID FROM employees e INNER JOIN department d ON e.DepartmentID = d.DepartmentID;

OUTPUT

EmployeeID	FirstName	LastName	Email	DeptName	Salary	DepartmentID
101	John	Doe	john.doe@email.com	HR	48000.00	1
102	Jane	Smith	jane.smith@example.com	Finance	48000.00	2
103	Robert	Johnson	robert.johnson@email.com	HR	48000.00	1
104	Mary	Jones	mary.jones@email.com	IT	48000.00	3
105	Michael	Brown	michael.brown@email.com	Marketing	48000.00	4
106	Jennifer	Davis	jennifer.davis@email.com	Sales	50000.00	5
107	David	Martinez	david.martinez@email.com	Operations	50000.00	6
108	Lisa	Rodriguez	lisa.rodriguez@email.com	Research	48000.00	7
109	William	Taylor	william.taylor@email.com	Development	48000.00	8
110	Sarah	Thomas	sarah.thomas@email.com	Customer Service	48000.00	9

10 rows in set (0.00 sec)

7.Write a SQL query to calculate the total number of employees in each department.

QUERY

mysql> SELECT d.DepartmentID, d.DeptName, COUNT(e.EmployeeID) AS TotalEmployees FROM department d LEFT JOIN employees e ON d.DepartmentID = e.DepartmentID GROUP BY d.DepartmentID, d.DeptName;

OUTPUT

DepartmentID	DeptName	TotalEmployees
1	HR	2
2	Finance	1
3	IT	1
4	Marketing	1
5	Sales	1
6	Operations	1
7	Research	1
8	Development	1
9	Customer Service	1
10	Administration	0

10 rows in set (0.01 sec)

8. Retrieve the concatenation of the FirstName and LastName columns for all employees in the Employee table.

QUERY

mysql> SELECT CONCAT(FirstName, ' ', LastName) AS FullName FROM employees;

OUTPUT

FullName
John Doe
Jane Smith
Robert Johnson
Mary Jones
Michael Brown
Jennifer Davis
David Martinez
Lisa Rodriguez
William Taylor
Sarah Thomas

10 rows in set (0.13 sec)

9. Write a query that uses the WHERE clause to select all employees whose FirstName is "Jennifer".

QUERY

mysql> SELECT * FROM employees WHERE FirstName = 'Jennifer';

OUTPUT

```
+------------+-----------+----------+-------------------------+--------------+----------+
| EmployeeID | FirstName | LastName | Email                   | DepartmentID | Salary   |
+------------+-----------+----------+-------------------------+--------------+----------+
|        106 | Jennifer  | Davis    | jennifer.davis@email.com |            5 | 50000.00 |
+------------+-----------+----------+-------------------------+--------------+----------+
```

1 row in set (0.01 sec)

10. Use the ORDER BY clause to sort the records in the Employee table based on the LastName column in ascending order.

QUERY

mysql> SELECT * FROM employees ORDER BY LastName ASC;

OUTPUT

```
+------------+-----------+-----------+-------------------------+--------------+----------+
| EmployeeID | FirstName | LastName  | Email                   | DepartmentID | Salary   |
+------------+-----------+-----------+-------------------------+--------------+----------+
|        105 | Michael   | Brown     | michael.brown@email.com |            4 | 48000.00 |
|        106 | Jennifer  | Davis     | jennifer.davis@email.com |            5 | 50000.00 |
|        101 | John      | Doe       | john.doe@email.com      |            1 | 48000.00 |
|        103 | Robert    | Johnson   | robert.johnson@email.com |            1 | 48000.00 |
|        104 | Mary      | Jones     | mary.jones@email.com    |            3 | 48000.00 |
|        107 | David     | Martinez  | david.martinez@email.com |            6 | 50000.00 |
|        108 | Lisa      | Rodriguez | lisa.rodriguez@email.com |            7 | 48000.00 |
|        102 | Jane      | Smith     | jane.smith@example.com  |            2 | 48000.00 |
|        109 | William   | Taylor    | william.taylor@email.com |            8 | 48000.00 |
|        110 | Sarah     | Thomas    | sarah.thomas@email.com  |            9 | 48000.00 |
+------------+-----------+-----------+-------------------------+--------------+----------+
```

10 rows in set (0.00 sec)

11. Provide an example of using the LIKE operator to select all employees whose last names start with the letter "S".

QUERY

mysql> SELECT * FROM employees WHERE LastName LIKE 'S%';

OUTPUT

```
+------------+-----------+----------+-----------------------+--------------+----------+
| EmployeeID | FirstName | LastName | Email                 | DepartmentID | Salary   |
+------------+-----------+----------+-----------------------+--------------+----------+
|        102 | Jane      | Smith    | jane.smith@example.com |            2 | 48000.00 |
+------------+-----------+----------+-----------------------+--------------+----------+
```

1 row in set (0.00 sec)

12. Write a query using the IN operator to select all employees whose DepartmentID is either 7, 8, or 9.

QUERY

mysql> SELECT * FROM employees WHERE DepartmentID IN (7, 8, 9);

OUTPUT

```
+------------+-----------+-----------+-------------------------+--------------+----------+
| EmployeeID | FirstName | LastName  | Email                   | DepartmentID | Salary   |
+------------+-----------+-----------+-------------------------+--------------+----------+
|        108 | Lisa      | Rodriguez | lisa.rodriguez@email.com |            7 | 48000.00 |
|        109 | William   | Taylor    | william.taylor@email.com |            8 | 48000.00 |
|        110 | Sarah     | Thomas    | sarah.thomas@email.com   |            9 | 48000.00 |
+------------+-----------+-----------+-------------------------+--------------+----------+
```

3 rows in set (0.00 sec)

PROGRAM 3

AIM: Creation of a database COMPANY, and tables using DDL commands including integrity constraints. Populate the tables with DML commands.

DDL COMMANDS

1.Create an employee table with the attributes specified above (set E_ID as the primary key).

QUERY

```
create table employee( E_id varchar(20) primary key,E_name char(20),E_gender char(10),E_salary numeric(8,2),E_branch varchar(10));
```

OUTPUT

```
Query OK, 0 rows affected (0.016 sec)
```

2.Insert the given values into the attributes.

QUERY

```
mysql> insert into employee
    -> values('E1','A','M','12500','B1'),('E2','B','F','15000','B4')
    -> ,('E3','C','M','36574','B3'),('E4','D','F','35674','B2'),('E5
    '> ','E','F','46572','B3'),('E6','F','M','43564','B4'),('E7','G'
    -> ,'M','65431','B2');
Query OK, 7 rows affected (0.14 sec)
Records: 7  Duplicates: 0  Warnings: 0
mysql> select *from employee;
```

OUTPUT

E_id	E_name	E_gender	E_salary	E_branch
E1	A	M	12500.00	B1
E2	B	F	15000.00	B4
E3	C	M	36574.00	B3
E4	D	F	35674.00	B2
E5	E	F	46572.00	B3
E6	F	M	43564.00	B4
E7	G	M	65431.00	B2

7 rows in set (0.00 sec)

3. Delete all rows from the table and free the space containing the table (TRUNCATE). Redo question 2.

QUERY

mysql> TRUNCATE table employee;
Query OK, 0 rows affected (0.47 sec) mysql>

select *from employee;

OUTPUT

Empty set (0.00 sec)

DML COMMANDS

1. Select all attributes of the EMP table.

QUERY

mysql> select * from employee;

OUTPUT

```
+------+--------+----------+----------+----------+
| E_id | E_name | E_gender | E_salary | E_branch |
+------+--------+----------+----------+----------+
| E1   | A      | M        | 12500.00 | B1       |
| E2   | B      | F        | 15000.00 | B4       |
| E3   | C      | M        | 36574.00 | B3       |
| E4   | D      | F        | 35674.00 | B2       |
| E5   | E      | F        | 46572.00 | B3       |
| E6   | F      | M        | 43564.00 | B4       |
| E7   | G      | M        | 65431.00 | B2       |
+------+--------+----------+----------+----------+
```

7 rows in set (0.02 sec)

2. Retrieve the average salary of the employees.

QUERY

mysql> select avg(E_salary) from employee;

OUTPUT

```
+---------------+
| avg(E_salary) |
+---------------+
|  36473.571429 |
+---------------+
```

1 row in set (0.09 sec)

3. Retrieve the name of the employee with the minimum salary.

QUERY

mysql> select E_name from employee where E_salary=(select MIN(E_salary) from employee);

OUTPUT

```
+--------+
| E_name |
+--------+
| A      |
+--------+
```

1 row in set (0.11 sec)

4. Retrieve the name of the employee with the maximum salary.

QUERY

mysql> select E_name from employee where

-> E_salary=(select MAX(E_salary) from employee);

OUTPUT

```
+--------+
| E_name |
+--------+
| G      |
+--------+
```

1 row in set (0.00 sec)

5. Find the total number of employees.

QUERY

mysql> select count(*) from employee;

OUTPUT

```
+----------+
| count(*) |
+----------+
|        7 |
+----------+
```

1 row in set (0.00 sec)

6. Calculate the total amount of salary.

QUERY

mysql> select SUM(E_salary) from employee;

OUTPUT

```
+---------------+
| SUM(E_salary) |
+---------------+
|     255315.00 |
+---------------+
```

1 row in set (0.03 sec)

7. Find the total number of employees segregated based on branches.

QUERY

mysql> select E_branch,count(*) from employee group by E_branch;

OUTPUT

```
+----------+----------+
| E_branch | count(*) |
+----------+----------+
| B1       |        1 |
| B2       |        2 |
| B3       |        2 |
| B4       |        2 |
+----------+----------+
```

4 rows in set (0.00 sec)

8. Find out the name of the employee having a salary > 15,000.

QUERY

mysql> select E_name from employee where E_salary>15000;

OUTPUT

```
+--------+
| E_name |
+--------+
| C      |
| D      |
| E      |
| F      |
| G      |
+--------+
```

5 rows in set (0.01 sec)

9. Display the names of the employees in ascending order.

QUERY

mysql> select E_name from employee order by E_name ->

ASC;

OUTPUT

```
+--------+
| E_name |
+--------+
| A      |
| B      |
| C      |
| D      |
| E      |
| F      |
| G      |
+--------+
```

7 rows in set (0.00 sec)

10. Display the names of the employees in descending order.

QUERY

mysql> select E_name from employee order by E_name DESC;

OUTPUT

```
+--------+
| E_name |
+--------+
| G      |
| F      |
| E      |
| D      |
| C      |
| B      |
| A      |
+--------+
```

7 rows in set (0.00 sec)

10. Find names of the employees belonging to the same branch. (Hint: GROUP BY, HAVING).

QUERY

mysql> select E_branch,group_concat(E_name) as E_name from employee group by E_branch having count(*)>1;

OUTPUT

```
+----------+--------+
| E_branch | E_name |
+----------+--------+
| B2       | D,G    |
| B3       | C,E    |
| B4       | B,F    |
+----------+--------+
```

3 rows in set (0.06 sec)

11. List names and salaries of the employees whose salary is more than the average salary of the employees.

QUERY

mysql> select E_name,E_salary from employee where -

> E_salary > (select avg(E_salary)from employee);

OUTPUT

```
+--------+----------+
| E_name | E_salary |
+--------+----------+
| C      | 36574.00 |
| E      | 46572.00 |
| F      | 43564.00 |
| G      | 65431.00 |
+--------+----------+
```

4 rows in set (0.00 sec)

12. Create a view named name of employee whose salary is greater than the average salary.

QUERY mysql> create view Emp_name as select * from employee where

-> E_salary > (select avg(E_salary)from employee);

Query OK, 0 rows affected (0.08 sec) mysql> select

*from Emp_name;

OUTPUT

```
+------+--------+----------+----------+----------+
| E_id | E_name | E_gender | E_salary | E_branch |
+------+--------+----------+----------+----------+
| E3   | C      | M        | 36574.00 | B3       |
| E5   | E      | F        | 46572.00 | B3       |
| E6   | F      | M        | 43564.00 | B4       |
| E7   | G      | M        | 65431.00 | B2       |
+------+--------+----------+----------+----------+
```

4 rows in set (0.00 sec)

13. The name of the employee who is in branch B2 or male.

QUERY

mysql> select E_name from employee where ->

E_branch='B2' || E_gender='M';

OUTPUT

```
+--------+
| E_name |
+--------+
| A      |
| C      |
| D      |
| F      |
| G      |
+--------+
```

5 rows in set (0.00 sec)

14. The name of the employee who is in branch B3 and female.

QUERY

mysql> select E_name from employee where E_branch='B3' || E_gender='F';

OUTPUT

```
+--------+
| E_name |
+--------+
| B      |
| C      |
| D      |
| E      |
+--------+
```

4 rows in set (0.00 sec)

15. The name of the employee who is in branch B2 but not male.

QUERY

mysql> select E_name from employee where E_branch='B2' && E_gender='F';

OUTPUT

```
+--------+
| E_name |
+--------+
| D      |
+--------+
```

1 row in set (0.00 sec)

PROGRAM 4

AIM: Application of views and joins for query optimization.

1. Create a view named emp whose salary is greater than the average salary.

QUERY

```
create view Emp_List as select * from Emp where
E_Salary>(select avg(E_Salary) from Emp);
```

OUTPUT

```
Query OK, 0 rows affected (0.065 sec)
```

QUERY

```
select * from Emp_List;
```

OUTPUT

```
+------+--------+----------+----------+----------+
| E_ID | E_Name | E_Gender | E_Salary | E_Branch |
+------+--------+----------+----------+----------+
| E3   | C      | M        | 36574.00 | B3       |
| E5   | E      | F        | 46572.00 | B3       |
| E6   | F      | M        | 43564.00 | B4       |
| E7   | G      | M        | 65431.00 | B2       |
+------+--------+----------+----------+----------+
```

2. The name of the employee who is in branch B2 or male.

QUERY

```
select E_Name from Emp where E_Gender='M' or E_Branch='B2';
```

OUTPUT

```
+--------+
| E_Name |
+--------+
| A      |
| C      |
| D      |
| F      |
| G      |
+--------+
```

3. The name of the employee who is in branch B3 and female.

QUERY

selectE_Name from Emp where E_Gender='F' and E_Branch='B3';

OUTPUT

```
+--------+
| E_name |
+--------+
| E      |
+--------+
```

4. The name of the employee who is in branch B2 but not male.

QUERY

selectE_Name from Emp where E_Gender='F' and E_Branch='B2';

OUTPUT

```
+--------+
| E_Name |
+--------+
| D      |
+--------+
```

PROGRAM 5

AIM: Apply DCL and TCL commands to impose restrictions on database.

DCL COMMANDS

1. Grant user access privileges to a database. (GRANT).

QUERY

```
GRANT ALL PRIVILEGES ON *.* TO `MITS`@`%`;
```

OUTPUT

Query OK, 0 rows affected (0.002 sec)

2. The employees table stores the data of employees.

QUERY

```
REVOKE ALL ON *.* FROM 'MITS'@'%';
```

OUTPUT

Query OK, 0 rows affected (0.002 sec)

TCL COMMANDS

1. Save all transactions to the database. (COMMIT).

QUERY

```
GRANT ALL PRIVILEGES ON *.* TO `MITS`@`%`;
```

OUTPUT

Query OK, 0 rows affected (0.002 sec)

2. Undo transactions that have not already been saved to the database. (Rollback).

QUERY

```
REVOKE ALL ON *.* FROM 'MITS'@'%';
```

OUTPUT

Query OK, 0 rows affected (0.002 sec)

Course Outcome 2

Apply PL/SQL for processing databases.

PROGRAM 6

AIM: Basics of PL/SQL

1. Write a program to add two numbers

PROGRAM

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
A INTEGER;
B INTEGER;
   BEGIN
A := &A;
B := &B;
   DBMS_OUTPUT.PUT_LINE('ENTERED VALUE:'||(A+B));
   END;
   /
```

OUTPUT

```
Enter value for a: 5
old   5: A := &A; new
5: A := 5; Enter value
for b: 6 old   6: B :=
&B; new   6: B := 6;
ENTERED VALUE:11

PL/SQL procedure successfully completed.
```

2. Write a program to find largest of 3 numbers

PROGRAM

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
A INTEGER;
B INTEGER;
C INTEGER;
   BEGIN
A :=&A;
B :=&B;
C :=&C;
```

```
  IF (A > B AND A > C)

 THEN
  DBMS_OUTPUT.PUT_LINE('THE GREATEST NUMBER IS '||A);
  ELSIF (B > C)
  THEN
  DBMS_OUTPUT.PUT_LINE('THE GREATEST NUMBER IS '||B);   ELSE
  DBMS_OUTPUT.PUT_LINE('THE GREATEST NUMBER IS '||C);
  END IF;
  END;
  /
```

OUTPUT

```
Enter value for a: 8
old   6: A :=&A;
new   6: A :=8; Enter
value for b: 9 old
7: B :=&B; new   7:
B :=9; Enter value
for c: 21 old   8: C
:=&C;
new   8: C :=21;
THE GREATEST NUMBER IS 21

PL/SQL procedure successfully completed.
```

3. Write a program to check whether given number is even or odd

PROGRAM

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
    A INTEGER;
    BEGIN
    A :=&A;
    IF (MOD(A,2)=0)
    THEN
    DBMS_OUTPUT.PUT_LINE(A||' IS AN EVEN NUMBER');
    ELSE
    DBMS_OUTPUT.PUT_LINE(A||' IS AN ODD NUMBER');
   END IF;
   END;
   /
```

OUTPUT

```
Enter value for a: 5
old      4:  A  :=&A;
new   4: A :=5;

5 IS AN ODD NUMBER

PL/SQL procedure successfully completed.
```

PROGRAM 7

AIM: Create a function to print annual salary of the employees in HR department.

PROGRAM

```
create table empl(emp_id number,name varchar(20),dept varchar(20),sal number);
insert into empl values(101,'Bobby','HR',25000);
insert into empl values(102,'George','HR',32000);
insert into empl values(103,'James','Hardware',55000);
insert into empl values(104,'David','Hardware',65000);
insert into empl values(105,'Sona','Marketing',20000);
insert into empl values(106,'Saira','HR',21000); insert
into empl values(107,'Fawaz','Software',50000);


SQL> CREATE OR REPLACE FUNCTION CALC_TOT_SAL RETURN NUMBER IS
       S NUMBER := 0;
    BEGIN
    FOR A IN (SELECT SAL FROM EMPL WHERE DEPT = 'HR') LOOP
      S := S + A.SAL;
     END LOOP;
     DBMS_OUTPUT.PUT_LINE('ANNUAL SALARY: ' || (S * 12));
     RETURN S;
     END;
     /
Function created.

SQL> SELECT CALC_TOT_SAL FROM DUAL;

CALC_TOT_SAL
------------
78000
```

OUTPUT

```
ANNUAL SALARY: 936000
```

PROGRAM 8

AIM: Create a cursor to print employee name of employees whose salary is greater than 50000.

PROGRAM

```
SQL> DECLARE
   CURSOR curs_work IS
   SELECT name
    FROM worker
WHERE sal > 50000;
   v_ename worker.name%TYPE; -- Variable to hold employee name
BEGIN

   OPEN curs_work;
   LOOP
   FETCH curs_work INTO v_ename;
   EXIT WHEN curs_work%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_ename);
  END LOOP;
  CLOSE curs_work;
  END;
  /
```

OUTPUT

```
Employee Name: James
Employee Name: David

PL/SQL procedure successfully completed.
```

PROGRAM 9

AIM: Construct a Trigger code for a table in database

PROGRAM

```
create table worker as select *from empl;
CREATE OR REPLACE TRIGGER salary_update_trigger
BEFORE UPDATE OF sal ON worker
FOR EACH ROW DECLARE
    old_salary worker.sal%TYPE;
    new_salary worker.sal%TYPE;
BEGIN
    old_salary := :OLD.sal;
    new_salary := :NEW.sal;

    DBMS_OUTPUT.PUT_LINE('Salary difference: ' || (new_salary - old_salary));
END;
/

SQL> update worker set sal=50000 where EMP_ID=102;
Salary difference: 18000

1 row updated.

SQL> SELECT * FROM worker;
```

OUTPUT

```
    EMP_ID NAME                 DEPT                          SAL
---------- -------------------- -------------------- ----------
101 Bobby               HR                        25000
102 George              HR                        50000
103 James               Hardware                  55000
104 David               Hardware                  65000
105 Sona                Marketing                 20000
106 Saira               HR                        21000
107 Fawaz               Software                  50000

7 rows selected.
```

PROGRAM 10

AIM: Write a PL/SQL Procedure to list all even and odd number between 1 and 20

PROGRAM

```
SQL> create or replace procedure eve_odd is
i number;
   BEGIN
         FOR   i   IN   1..20   LOOP
IF MOD(i, 2) = 0 THEN
               dbms_output.put_line(i||'      IS      EVEN');
ELSE
               dbms_output.put_line(i||' IS ODD');
            END IF;
         END LOOP;
  END;
  /
Procedure created.
```

OUTPUT

```
SQL> execute eve_odd;
1   IS ODD
2   IS EVEN
3   IS ODD
4   IS EVEN
5   IS ODD
6   IS EVEN
7   IS ODD
8   IS EVEN
9   IS ODD
10 IS EVEN
11 IS ODD
12 IS EVEN
13 IS ODD
14 IS EVEN
15 IS ODD
16 IS EVEN
17 IS ODD
18 IS EVEN
19 IS ODD
20 IS EVEN
```

PL/SQL procedure successfully complete.

PROGRAM 11

AIM: Write A PL/SQL Procedure to find factorial of a number.

PROGRAM

```
SQL> CREATE OR REPLACE PROCEDURE fact(num number) is
res NUMBER;
   BEGIN
          res := 1;
          FOR i IN 1..num LOOP
             res    :=    res    *    i;
END                          LOOP;
dbms_output.put_line(res);
    END;
   /
Procedure created.
```

OUTPUT

```
SQL> execute fact(5);
120

PL/SQL procedure successfully completed.
```

Course Outcome 3

Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases. Apply CRUD operations and retrieve data in NoSQL environment.

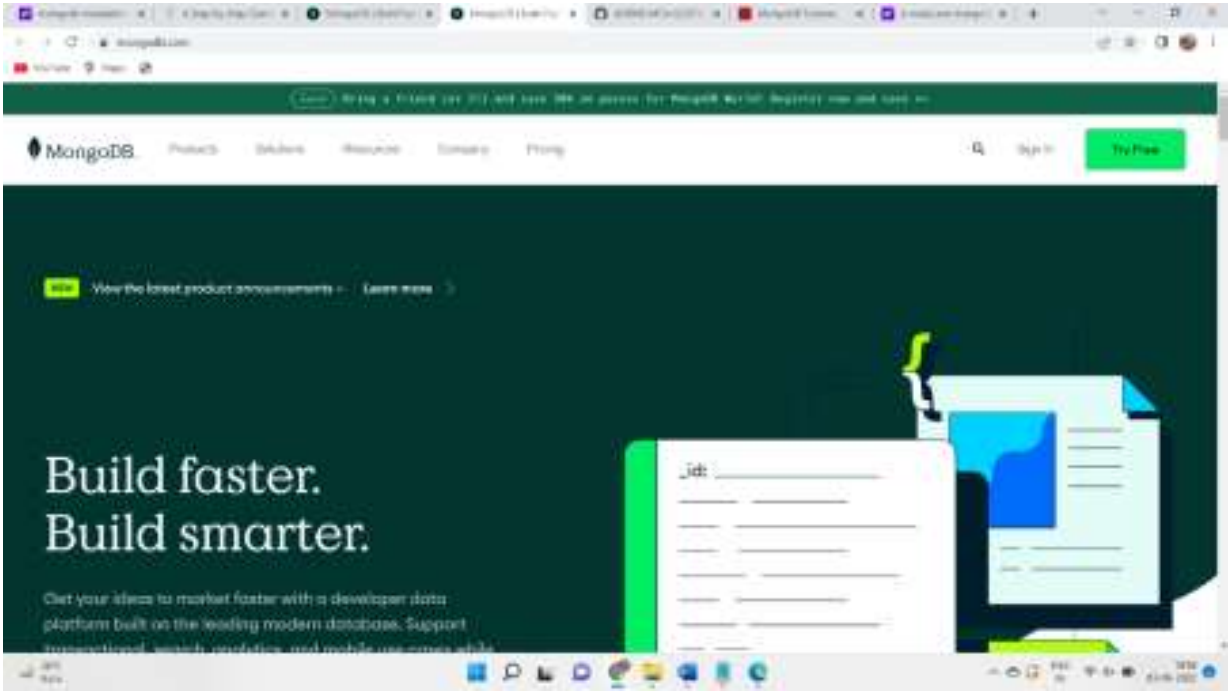
PROGRAM 12

AIM: Installation and configuration of NoSQL database- MongoDB

MongoDB is a cross-platform, document oriented NoSql database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

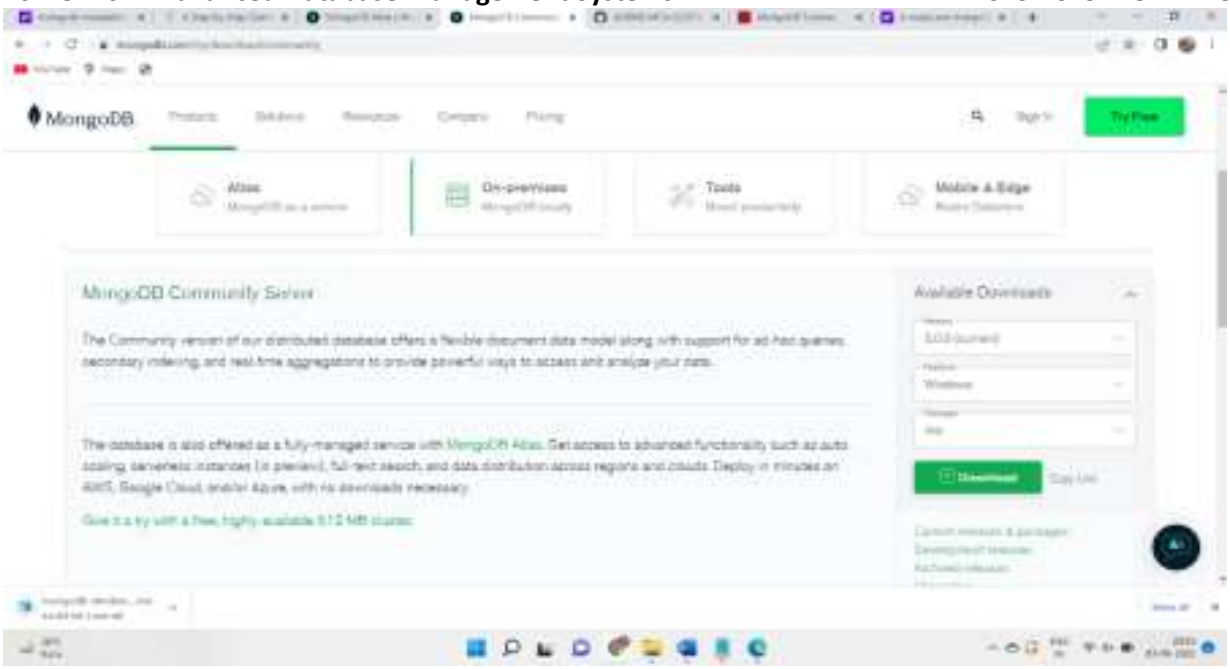
STEP 1:

Navigate to the official MongoDB website.



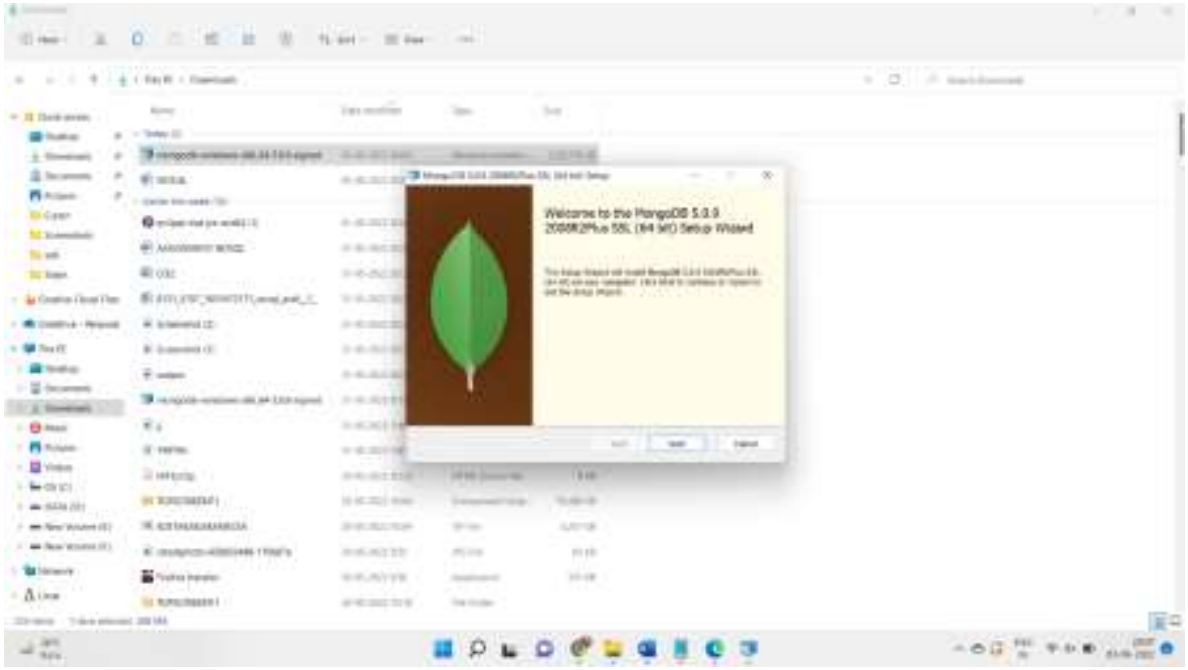
STEP 2:

Under the products section, click on the Community server version. Make sure that the specifications to the right of the screen are correct. At the time of writing, the latest version is 4.4.5. Ensure that the platform is Windows, and the package is MSI. Go ahead and click on download.



STEP 3:

You can find the downloaded file in the downloads directory. Install the software step by step.



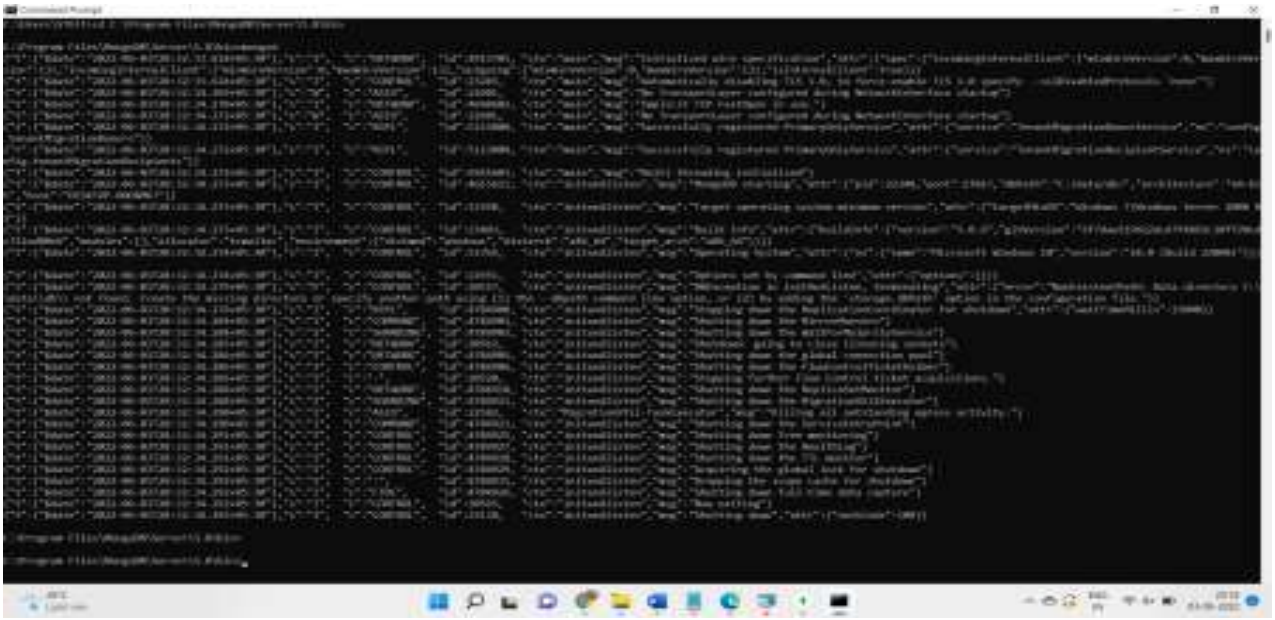
STEP:4

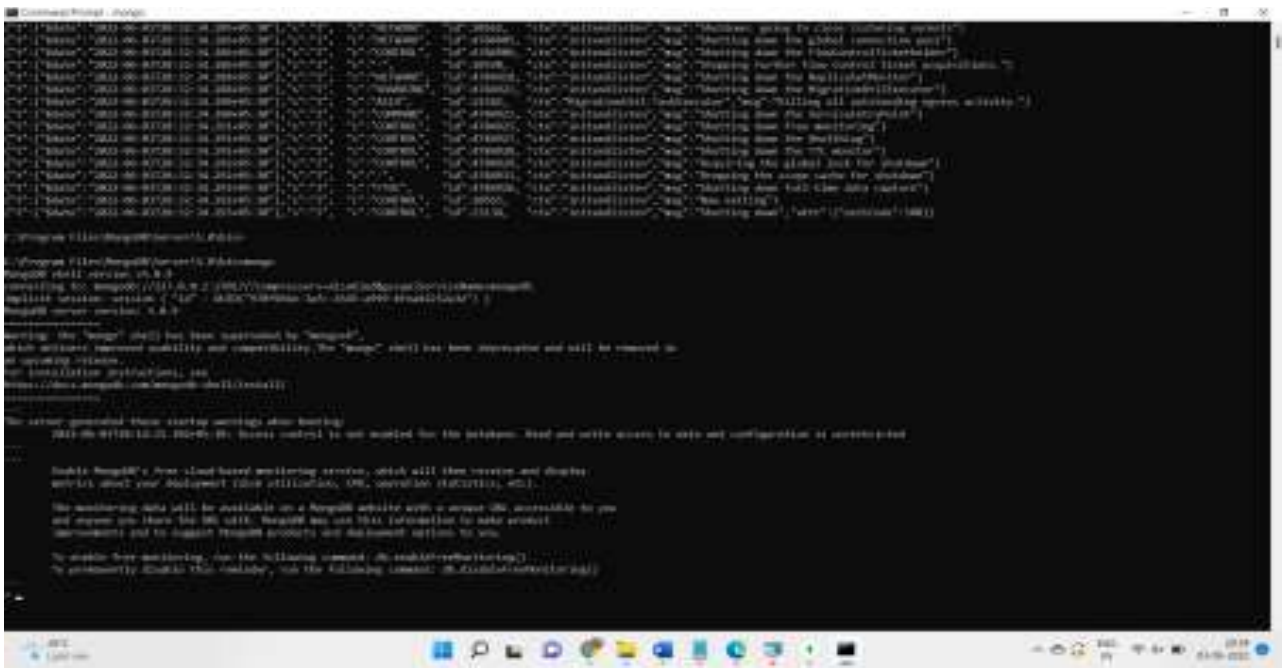
create an environment variable for the executable file so that we don't have to change the directory structure every time we want to execute the file.

STEP:5

After creating an environment path, download mongosh and install. we can open the command prompt and type mongod. An instance of mongodb server is started. Now take

another terminal and type mongosh. This creates a client instance of mongodb in your local system.





Step:6

You can start creating new databases and use them.

PROGRAM 13

AIM:Compare relational and non-relational databases.

1.Answer the following questions in SQL (relational database) and MongoDB (nonrelational database).

a. Create a student table/collection with columns/fields for name, age, and city.

MONGODB

```
db.createCollection("student")
```

SQL

```
MariaDB [college]> create table student(name varchar(50),age int,city varchar(50)); Query
OK, 0 rows affected (0.03 sec)
```

b. Insert a new row/document into the students table/collection with the following information: Name: John Doe, Age: 25, City:New York.

MONGODB

```
college> db.student.insertOne({name:"John Doe",age:25,city:"New york"}) {
 acknowledged: true,  insertedId:
 ObjectId('66028648848386b2f0d14a0e')
}

college> db.student.find()
[
  {
   _id: ObjectId('66028648848386b2f0d14a0e'),
   name:  'John  Doe',
   age:   25,      city:
   'New york'
  }  ]
```

SQL

```
mysql> insert into student values("Jhon Doe",25,"New York"); Query
OK, 1 row affected (0.01 sec)  mysql>
select * from student;
+----------+------+----------+
| name     | age  | city     |
+----------+------+----------+
 | Jhon Doe | 25   | New York |
+----------+------+----------+
1 row in set (0.00 sec)
```

c. Retrieve all data from the students table/collection.

MONGODB

```
college> db.student.insertMany([{name:"Roy",age:27,city:"New
Delhi"},{name:"Alan",age:25,city:"Kochi"}])
{
 acknowledged: true,  insertedIds:
  {
   '0': ObjectId('6602879b848386b2f0d14a0f'),
   '1': ObjectId('6602879b848386b2f0d14a10')
  }
}

college> db.student.find()
[
  {
   _id: ObjectId('66028648848386b2f0d14a0e'),
   name:  'John  Doe',
   age:    25,       city:
   'New york'
  },
  {
   _id: ObjectId('6602879b848386b2f0d14a0f'),  name:
   'Roy',
   age: 27,  city:
   'New Delhi'
  },
  {
   _id:   ObjectId('6602879b848386b2f0d14a10'),
   name: 'Alan',  age: 25,  city: 'Kochi'
  } ]
```

SQL

```
mysql> insert into student values("Roy",27,"New Delhi"),("Alan",25,"Kochi"); Query
OK, 2 rows affected (0.01 sec)  Records:
2 Duplicates: 0 Warnings: 0  mysql>
select * from student;
+----------+------+-----------+
| name     | age| city        |
+----------+------+-----------+
| Jhon Doe | 25 | New York    |
| Roy      | 27 | New Delhi   |
| Alan     | 25 | Kochi       |
+----------+------+-----------+
```

d. Update the age of a student named John Doe to 30 in the students table/collection, assuming there's already record/document for him

MONGODB

```
college> db.student.updateOne({name:"John Doe"},{$set:{age:30}}) {
  acknowledged: true,
  insertedId: null,  matchedCount:
  1,  modifiedCount: 1,
  upsertedCount: 0
} college>
db.student.find()
[
  {
    _id: ObjectId('66028648848386b2f0d14a0e'), name:
    'John Doe',  age:
    30,  city: 'New
    york'
  },
    {
    _id: ObjectId('6602879b848386b2f0d14a0f'), name:
    'Roy',  age: 27,
    city: 'New Delhi'
  },  {
    _id: ObjectId('6602879b848386b2f0d14a10'), name:
    'Alan',  age:
    25,  city:
    'Kochi'
  } ]
```

SQL

```
mysql> update student set age=30 where name="Jhon Doe"; Query
OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0  mysql>
select * from student;
+----------+------+-----------+  |
name      | age| city      |
+----------+------+-----------+
| Jhon Doe | 30 | New York  |
| Roy      | 27 | New Delhi |
| Alan     | 25 | Kochi     |
+----------+------+-----------+
3 rows in set (0.00 sec)
```

e. Get all details of students whose age is older than 25.

MONGODB

```
college> db.student.find({age:{$gt:25}})
[
  {
   _id: ObjectId('66028648848386b2f0d14a0e'),
   name:  'John  Doe',
   age:    30,       city:
   'New york'
  },
   {
   _id: ObjectId('6602879b848386b2f0d14a0f'),
   name: 'Roy',
   age: 27,  city:
   'New Delhi'
  } ]
```

SQL mysql> select * from student where age>25;

```
+----------+------+-----------+
| name     | age  | city      |
+----------+------+-----------+
| Jhon Doe | 30   | New York  |
| Roy      | 27   | New Delhi |
+----------+------+-----------+
```

2 rows in set (0.00 sec)

f. Get the name of students whose names begin with the letter 'V'.

MONGODB

```
college> db.student.insertOne({name:"Varun Sharma",age:23,city:"UP"}) {
 acknowledged: true,  insertedId:
 ObjectId('660290ad848386b2f0d14a11')
} college>
db.student.find()
[
  {
   _id: ObjectId('66028648848386b2f0d14a0e'), name:
   'John Doe',  age:
   30,
   city: 'New york'
  },  {
   _id: ObjectId('6602879b848386b2f0d14a0f'), name:
   'Roy',
```

```
    age: 32,
    city: 'New Delhi'
  },  {
    _id: ObjectId('6602879b848386b2f0d14a10'), name:
    'Alan',  age:
    50,  city:
    'Kochi'
  },  {
    _id: ObjectId('660290ad848386b2f0d14a11'), name:
    'Varun   Sharma',
    age:   23,     city:
    'UP'
  }
]

college> db.student.find({name: /^V/}) [   {
    _id: ObjectId('660290ad848386b2f0d14a11'), name:
    'Varun   Sharma',
    age:   23,     city:
    'UP'
  }  ]
```

SQL mysql> insert into student values("Varun Sharma",23,"UP");
Query OK, 1 row affected (0.01 sec) mysql> select name from student where name LIKE 'V%';

```
+--------------+
| name         |
+--------------+
| Varun Sharma |
+--------------+
```

1 row in set (0.00 sec)

PROGRAM 14

AIM: To build a sample collections/documents to perform query operations. Create database college and collection students and insert student details into it.

```
test> use college;
switched to db college
college> db.createCollection("students");
{ ok: 1 }
college> db.students.insertMany([
{id: 1,name: "Arjun",age: 20,gender: "Male",department: "Computer Science",gpa: 3.8},
{id: 2,name: "Akhila",age: 22,gender: "Female",department: "Electrical   Engineering",gpa: 3.6},
{id: 3,name: "Akash",age: 19,gender: "Male",department: "Civil Engineering",gpa: 3.9},
{id: 4,name: "Anu",age: 21,gender: "Female",department: "Computer Science",gpa: 3.7},
{id: 5,name: "Binu",age: 23,gender: "Female",department: "Mechanical Engineering",gpa: 3.5}
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660eccb0f96a2845bc9f990a'),
    '1': ObjectId('660eccb0f96a2845bc9f990b'),
    '2': ObjectId('660eccb0f96a2845bc9f990c'),
    '3': ObjectId('660eccb0f96a2845bc9f990d'),
    '4': ObjectId('660eccb0f96a2845bc9f990e')
  }
}
college> db.students.find()
[
  {
    _id: ObjectId('660eccb0f96a2845bc9f990a'),
    id: 1,
    name: 'Arjun',
    age: 20,
    gender: 'Male',
    department: 'Computer Science',
    gpa: 3.8
  },
  {
    _id: ObjectId('660eccb0f96a2845bc9f990b'),
    id: 2,
    name: 'Akhila',
    age: 22,
    gender: 'Female',
    department: 'Electrical Engineering',
    gpa: 3.6
  },
  {
    _id: ObjectId('660eccb0f96a2845bc9f990c'),
    id: 3,
```

```
    name: 'Akash',
    age: 19,
    gender: 'Male',
    department: 'Civil Engineering',
    gpa: 3.9
  },
  {
    _id: ObjectId('660eccb0f96a2845bc9f990d'),
    id: 4,
    name: 'Anu',
    age: 21,
    gender: 'Female',
    department: 'Computer Science',
    gpa: 3.7
  },
  {
    _id: ObjectId('660eccb0f96a2845bc9f990e'),
    id: 5,
    name: 'Binu',
    age: 23,
    gender: 'Female',
    department: 'Mechanical Engineering',
    gpa: 3.5
  }
]
```

PROGRAM 15

AIM: Create a Database ‘Student’ with the fields SRN, SName, degree, semester, CGPA and create a collection ‘Students’.

1. Display all the documents.

```
test> use Student
switched to db Student
Student> db.createCollection("Students")
{ ok: 1 }
Student> db.Students.insertMany([
{ SRN: 1, SName: "Anu", degree: "BCA", semester: 1, CGPA: 7.5 },
{ SRN: 2, SName: "Binu", degree: "BCA", semester: 2, CGPA: 8.1 },
{ SRN: 3, SName: "Akhila", degree: "BCA", semester: 3, CGPA: 6.8 },
{ SRN: 4, SName: "Dona", degree: "BCA", semester: 4, CGPA: 7.2 },
{ SRN: 5, SName: "Poja", degree: "BCA", semester: 5, CGPA: 6.9 },
{ SRN: 6, SName: "Jency", degree: "BCA", semester: 6, CGPA: 8.0 },
{ SRN: 7, SName: "Meenakshi", degree: "BCA", semester: 7, CGPA: 7.5 },
{ SRN: 8, SName: "Rachel", degree: "MCA", semester: 1, CGPA: 8.5 },
{ SRN: 9, SName: "Sherin", degree: "MCA", semester: 2, CGPA: 8.2 },
{ SRN: 10, SName: "Anjali", degree: "MCA", semester: 3, CGPA: 7.8 }
])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660ed51467c9d77d059f990a'),
    '1': ObjectId('660ed51467c9d77d059f990b'),
    '2': ObjectId('660ed51467c9d77d059f990c'),
    '3': ObjectId('660ed51467c9d77d059f990d'),
    '4': ObjectId('660ed51467c9d77d059f990e'),
    '5': ObjectId('660ed51467c9d77d059f990f'),
    '6': ObjectId('660ed51467c9d77d059f9910'),
    '7': ObjectId('660ed51467c9d77d059f9911'),
    '8': ObjectId('660ed51467c9d77d059f9912'),
    '9': ObjectId('660ed51467c9d77d059f9913')
  }
}
```

2. Display all the students in BCA.

```
Student> db.Students.find({ degree: "BCA" })
[
  {
    _id: ObjectId('660ed51467c9d77d059f990a'),
```

```
  SRN: 1,
  SName: 'Anu',
  degree: 'BCA',
  semester: 1,
  CGPA: 7.5
},
{
  _id: ObjectId('660ed51467c9d77d059f990b'),
  SRN: 2,
  SName: 'Binu',
  degree: 'BCA',
  semester: 2,
  CGPA: 8.1
},
{
  _id: ObjectId('660ed51467c9d77d059f990c'),
  SRN: 3,
  SName: 'Akhila',
  degree: 'BCA',
  semester: 3,
  CGPA: 6.8
},
{
  _id: ObjectId('660ed51467c9d77d059f990d'),
  SRN: 4,
  SName: 'Dona',
  degree: 'BCA',
  semester: 4,
  CGPA: 7.2
},
{
  _id: ObjectId('660ed51467c9d77d059f990e'),
  SRN: 5,
  SName: 'Poja',
  degree: 'BCA',
  semester: 5,
  CGPA: 6.9
},
{
  _id: ObjectId('660ed51467c9d77d059f990f'),
  SRN: 6,
  SName: 'Jency',
  degree: 'BCA',
  semester: 6,
  CGPA: 8
},
{
  _id: ObjectId('660ed51467c9d77d059f9910'),
  SRN: 7,
```

```
    SName: 'Meenakshi',
    degree: 'BCA',
    semester: 7,
    CGPA: 7.5
  }
]
```

3. Display all the students in ascending order

```
Student> db.Students.find().sort({SRN: 1})
[
  {
    _id: ObjectId('660ed51467c9d77d059f990a'),
    SRN: 1,
    SName: 'Anu',
    degree: 'BCA',
    semester: 1,
    CGPA: 7.5
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990b'),
    SRN: 2,
    SName: 'Binu',
    degree: 'BCA',
    semester: 2,
    CGPA: 8.1
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990c'),
    SRN: 3,
    SName: 'Akhila',
    degree: 'BCA',
    semester: 3,
    CGPA: 6.8
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990d'),
    SRN: 4,
    SName: 'Dona',
    degree: 'BCA',
    semester: 4,
    CGPA: 7.2
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990e'),
    SRN: 5,
    SName: 'Poja',
    degree: 'BCA',
```

```
    semester: 5,
    CGPA: 6.9
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990f'),
    SRN: 6,
    SName: 'Jency',
    degree: 'BCA',
    semester: 6,
    CGPA: 8
  },
  {
    _id: ObjectId('660ed51467c9d77d059f9910'),
    SRN: 7,
    SName: 'Meenakshi',
    degree: 'BCA',
    semester: 7,
    CGPA: 7.5
  },
  {
    _id: ObjectId('660ed51467c9d77d059f9911'),
    SRN: 8,
    SName: 'Rachel',
    degree: 'MCA',
    semester: 1,
    CGPA: 8.5
  },
  {
    _id: ObjectId('660ed51467c9d77d059f9912'),
    SRN: 9,
    SName: 'Sherin',
    degree: 'MCA',
    semester: 2,
    CGPA: 8.2
  },
  {
    _id: ObjectId('660ed51467c9d77d059f9913'),
    SRN: 10,
    SName: 'Anjali',
    degree: 'MCA',
    semester: 3,
    CGPA: 7.8
  }
]
```

4. Display all the first five students.

```
Student> db.Students.find().limit(5)
[
  {
    _id: ObjectId('660ed51467c9d77d059f990a'),
    SRN: 1,
    SName: 'Anu',
    degree: 'BCA',
    semester: 1,
    CGPA: 7.5
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990b'),
    SRN: 2,
    SName: 'Binu',
    degree: 'BCA',
    semester: 2,
    CGPA: 8.1
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990c'),
    SRN: 3,
    SName: 'Akhila',
    degree: 'BCA',
    semester: 3,
    CGPA: 6.8
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990d'),
    SRN: 4,
    SName: 'Dona',
    degree: 'BCA',
    semester: 4,
    CGPA: 7.2
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990e'),
    SRN: 5,
    SName: 'Poja',
    degree: 'BCA',
    semester: 5,
    CGPA: 6.9
  }
]
```

5. Display students 5,6,7

```
Student> db.Students.find({SRN:{$gte:5,$lte:7}})
[
  {
    _id: ObjectId('660ed51467c9d77d059f990e'),
    SRN: 5,
    SName: 'Poja',
    degree: 'BCA',
    semester: 5,
    CGPA: 6.9
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990f'),
    SRN: 6,
    SName: 'Jency',
    degree: 'BCA',
    semester: 6,
    CGPA: 8
  },
  {
    _id: ObjectId('660ed51467c9d77d059f9910'),
    SRN: 7,
    SName: 'Meenakshi',
    degree: 'BCA',
    semester: 7,
    CGPA: 7.5
  }
]
```

6. Display the degree of student ‘Anu’.

```
Student> db.Students.findOne({ SName:"Anu"}).degree
BCA
```

7. Display student details of 5,6,7 in descending order of percentage.

```
Student> db.Students.find({SRN:{$gte:5,$lte:7}}).sort({CGPA:-1})
[
  {
    _id: ObjectId('660ed51467c9d77d059f990f'),
    SRN: 6,
    SName: 'Jency',
    degree: 'BCA',
    semester: 6,
    CGPA: 8
```

```
  },
  {
    _id: ObjectId('660ed51467c9d77d059f9910'),
    SRN: 7,
    SName: 'Meenakshi',
    degree: 'BCA',
    semester: 7,
    CGPA: 7.5
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990e'),
    SRN: 5,
    SName: 'Poja',
    degree: 'BCA',
    semester: 5,
    CGPA: 6.9
  }
]
```

8. Display the number of students in BCA

```
Student> db.Students.count({degree:"BCA"})
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or
estimatedDocumentCount.
7
```

9. Display all the degrees without "_id"

```
Student> db.Students.find({},{_id:0,degree:1})
[
  { degree: 'BCA' },
  { degree: 'BCA' },
  { degree: 'BCA' },
  { degree: 'BCA' },
  { degree: 'BCA' },
  { degree: 'BCA' },
  { degree: 'BCA' },
  { degree: 'MCA' },
  { degree: 'MCA' },
  { degree: 'MCA' }
]
```

10. Display the distinct degrees.

```
Student> db.Students.distinct("degree")
[ 'BCA', 'MCA' ]
```

11. Display all the BCA students with CGPA>6 but less than 7.1

```
Student> db.Students.find({degree:"BCA",CGPA:{$gt:6,$lt:7.1}})
[
  {
    _id: ObjectId('660ed51467c9d77d059f990c'),
    SRN: 3,
    SName: 'Akhila',
    degree: 'BCA',
    semester: 3,
    CGPA: 6.8
  },
  {
    _id: ObjectId('660ed51467c9d77d059f990e'),
    SRN: 5,
    SName: 'Poja',
    degree: 'BCA',
    semester: 5,
    CGPA: 6.9
  }
]
```

12. Display all the BCA students and in 2nd sem.

```
Student> db.Students.find({degree:"BCA",semester:2})
[
  {
    _id: ObjectId('660ed51467c9d77d059f990b'),
    SRN: 2,
    SName: 'Binu',
    degree: 'BCA',
    semester: 2,
    CGPA: 8.1
  }
]
```

PROGRAM 16

AIM: Create an employee database with the fields: {eid, ename, dept, desig, salary, yoj, address {dno,street,locality,city}}

1. Display all the employees with salary in the range(50000,75000).

```
test> use employee;
switched to db employee
employee> db.createCollection("employees");
{ ok: 1 }
employee> db.employees.insertMany([
{id: 1,ename: "Rahul",dept: "IT",desig: "Developer",salary: 60000,yoj: 2010,address: {dno:
123,street: "Tech Park",locality: "Silicon Valley",city: "San Jose"}},
{id: 2,ename: "Anu",dept: "HR",desig: "Manager",salary: 80000,yoj: 2005,address: {dno:
456,street: "Corporate Park",locality: "Do
wntown",city: "San Francisco"}},
{id: 3,ename: "Binu",dept: "IT",desig: "Tester",salary: 55000,yoj: 2012,address: {dno:
789,street: "Tech Plaza",locality: "Silicon Oasis",city: "San Jose"}}
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660ee074ef0f76ffb19f990a'),
    '1': ObjectId('660ee074ef0f76ffb19f990b'),
    '2': ObjectId('660ee074ef0f76ffb19f990c')
  }
}
    employee> db.employees.find({salary:{$gte:50000,$lt:75000}});
    [
      {
        _id: ObjectId('660ee074ef0f76ffb19f990a'),
        id: 1,
        ename: 'Rahul',
        dept: 'IT',
        desig: 'Developer',
        salary: 60000,
        yoj:  2010,
        address:
        { dno: 123,
          street: 'Tech Park',
          locality: 'Silicon Valley',
          city: 'San Jose'
        }
      },
      {
        _id: ObjectId('660ee074ef0f76ffb19f990c'),
```

```
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
    }
  }
]
```

2. Display all the employees with design developer

```
employee> db.employees.find({desig:"Developer"});
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 60000,
    yoj:  2010,
    address:
    { dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    }
  }
]
```

3. Display the salary of Rahul

```
employee> db.employees.findOne({ename:"Rahul"},{salary:1});
{ _id: ObjectId('660ee074ef0f76ffb19f990a'), salary: 60000 }
```

4. Display the city of employee.

```
employee> db.employees.find({},{"address.city":1});
```

```
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    address: { city: 'San Jose' }
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    address: { city: 'San Francisco' }
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    address: { city: 'San Jose' }
  },
]
```

5. Update the salary of developers by 5000.

```
employee> db.employees.updateMany({desig:"Developer"},{$inc:{salary:5000}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
employee> db.employees.find()
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    yoj:  2010,
    address:
    { dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    }
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    id: 2,
    ename: 'Anu',
    dept: 'HR',
```

```
    desig: 'Manager',
    salary: 80000,
    yoj:  2005,
    address:
    { dno: 456,
      street: 'Corporate Park',
      locality: 'Downtown',
      city: 'San Francisco'
    }
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
    }
  }
]
```

6. Add field age to employee.

```
employee> db.employees.updateMany({},{$set:{age:30}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
employee> db.employees.find()
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    yoj: 2010,
    address: {
```

```
      dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    },
    age: 30
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    id: 2,
    ename: 'Anu',
    dept: 'HR',  desig:
    'Manager', salary:
    80000,
    yoj:  2005,
    address:
    { dno: 456,
      street: 'Corporate Park',
      locality: 'Downtown',
      city: 'San Francisco'
    },
    age: 30
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
    },
    age: 30
  }
]
```

7. Remove yoj from Rahul

```
employee> db.employees.updateOne({ename:"Rahul"},{$unset:{yoj:""}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
```

```
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.employees.find()
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    address: {
      dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    },
    age: 30
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    id: 2,
    ename: 'Anu',
    dept: 'HR',  desig:
    'Manager', salary:
    80000,
    yoj:  2005,
    address:
    { dno: 456,
      street: 'Corporate Park',
      locality: 'Downtown',
      city: 'San Francisco'
    },
    age: 30
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
```

```
    },
    age: 30
  }
]
```

8. Add an array field project to Rahul.

```
employee> db.employees.updateOne({ename:"Rahul"},{$push:{projects:"p1"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.employees.find()
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    address: {
      dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    },
    age: 30,
    projects: [ 'p1' ]
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    id: 2,
    ename: 'Anu',
    dept: 'HR',  desig:
    'Manager', salary:
    80000,
    yoj:  2005,
    address:
    { dno: 456,
      street: 'Corporate Park',
      locality: 'Downtown',
      city: 'San Francisco'
    },
    age: 30
```

```
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
    },
    age: 30
  }
]
```

9. Add p2 and p3 project to Rahul

```
employee> db.employees.updateOne({ename:"Rahul"},{$push:{projects:{$each:["p2","p3"]}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.employees.find()
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    address: {
      dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    },
    age: 30,
    projects: [ 'p1', 'p2', 'p3' ]
  },
  {
```

```
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    id: 2,
    ename: 'Anu',
    dept: 'HR',  desig:
    'Manager', salary:
    80000,
    yoj:  2005,
    address:
    { dno: 456,
      street: 'Corporate Park',
      locality: 'Downtown',
      city: 'San Francisco'
    },
    age: 30
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
    },
    age: 30
  }
]
```

10. Remove p3 from Rahul.

```
employee> db.employees.updateOne({ename:"Rahul"},{$pull:{projects:"p3"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.employees.find()
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
```

```
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    address: {
      dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    },
    age: 30,
    projects: [ 'p1', 'p2' ]
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    id: 2,
    ename: 'Anu',
    dept: 'HR',  desig:
    'Manager', salary:
    80000,
    yoj:  2005,
    address:
    { dno: 456,
      street: 'Corporate Park',
      locality: 'Downtown',
      city: 'San Francisco'
    },
    age: 30
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
    },
    age: 30
  }
]
```

11. Add a new embedded object “contacts” with “email” and “phone” as array objects to Rahul.

```
employee>
db.employees.updateOne({ename:"Rahul"},{$set:{contacts:{email:["rahul@gmail.com"],p
hone:["04829-262234","04829-225678"]}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.employees.find()
[
  {
    _id: ObjectId('660ee074ef0f76ffb19f990a'),
    id: 1,
    ename: 'Rahul',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    address: {
      dno: 123,
      street: 'Tech Park',
      locality: 'Silicon Valley',
      city: 'San Jose'
    },
    age: 30,
    projects: [ 'p1', 'p2' ],
    contacts: {
      email: [ 'rahul@gmail.com' ],
      phone: [ '04829-262234', '04829-225678' ]
    }
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990b'),
    id: 2,
    ename: 'Anu',
    dept: 'HR',  desig:
    'Manager', salary:
    80000,
    yoj:  2005,
    address:
    { dno: 456,
      street: 'Corporate Park',
      locality: 'Downtown',
      city: 'San Francisco'
    },
    age: 30
```

```
  },
  {
    _id: ObjectId('660ee074ef0f76ffb19f990c'),
    id: 3,
    ename: 'Binu',
    dept: 'IT',
    desig: 'Tester',
    salary: 55000,
    yoj:  2012,
    address:
    { dno: 789,
      street: 'Tech Plaza',
      locality: 'Silicon Oasis',
      city: 'San Jose'
    },
    age: 30
  }
]
```

PROGRAM 17

AIM: Create a database named college and then create a collection named students. Insert some values into it. Write a MongoDB Query to:

1. Display details of students who have their name starting with the letter 'C' using $regex operator

```
test> use college;
switched to db college
colleg> db.createCollection("students");
{ ok: 1 }
college> db.students.insertMany([
{id: 1, name: "Chris", dept: "CS", age: 21, gender: "Male" },
{id: 2, name: "Doanl", dept: "EE", age: 22, gender: "Male" },
{id: 3, name: "Anu", dept: "CS", age: 23, gender: "Female" },
{id: 4, name: "Karthika", dept: "ME", age: 24, gender: "Female" },
{id: 5, name: "Jency", dept: "EC", age: 25, gender: "Female" },
{id: 6, name: "Ryan", dept: "CS", age: 26, gender: "Male" },
{id: 7, name: "Ameer", dept: "EC", age: 27, gender: "Male" }
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660ee84b4181dffc519f990a'),
    '1': ObjectId('660ee84b4181dffc519f990b'),
    '2': ObjectId('660ee84b4181dffc519f990c'),
    '3': ObjectId('660ee84b4181dffc519f990d'),
    '4': ObjectId('660ee84b4181dffc519f990e'),
    '5': ObjectId('660ee84b4181dffc519f990f'),
    '6': ObjectId('660ee84b4181dffc519f9910')
  }
}
college> db.students.find()
[
  {
    _id: ObjectId('660ee84b4181dffc519f990a'),
    id: 1,
    name: 'Chris',
    dept: 'CS',
    age: 21,
    gender: 'Male'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990b'),
    id: 2,
    name: 'Doanl',
    dept: 'EE',
```

```
    age: 22,
    gender: 'Male'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990c'),
    id: 3,
    name: 'Anu',
    dept: 'CS',
    age: 23,
    gender: 'Female'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990d'),
    id: 4,
    name: 'Karthika',
    dept: 'ME',
    age: 24,
    gender: 'Female'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990e'),
    id: 5,
    name: 'Jency',
    dept: 'EC',
    age: 25,
    gender: 'Female'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990f'),
    id: 6,
    name: 'Ryan',
    dept: 'CS',
    age: 26,
    gender: 'Male'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f9910'),
    id: 7,
    name: 'Ameer',
    dept: 'EC',
    age: 27,
    gender: 'Male'
  }
]

college> db.students.find({name:{$regex:/^C/i}});
[
  {
    _id: ObjectId('660ee84b4181dffc519f990a'),
```

```
    id: 1,
    name: 'Chris',
    dept: 'CS',
    age: 21,
    gender: 'Male'
  }
]
```

2. Display details of students who have their name ending with the letter ‘r’ using $regex Operator

```
college> db.students.find({name:{$regex:/r$/i}});
[
  {
    _id: ObjectId('660ee84b4181dffc519f9910'),
    id: 7,
    name: 'Ameer',
    dept: 'EC',
    age: 27,
    gender: 'Male'
  }
]
```

3. Display details of students who are having ‘CS’ as their department using $regex operator

```
college> db.students.find({dept:{$regex:/CS/i}});
[
  {
    _id: ObjectId('660ee84b4181dffc519f990a'),
    id: 1,
    name: 'Chris',
    dept: 'CS',
    age: 21,
    gender: 'Male'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990c'),
    id: 3,
    name: 'Anu',
    dept: 'CS',
    age: 23,
    gender: 'Female'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990f'),
    id: 6,
    name: 'Ryan',
    dept: 'CS',
    age: 26,
```

```
    gender: 'Male'
  }
]
```

4. Remove details of student who are having ‘EC’ as their department

```
college> db.students.deleteMany({dept:{$regex:/EC/i}});
{ acknowledged: true, deletedCount: 2 }
college> db.students.find()
[
  {
    _id: ObjectId('660ee84b4181dffc519f990a'),
    id: 1,
    name: 'Chris',
    dept: 'CS',
    age: 21,
    gender: 'Male'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990b'),
    id: 2,
    name: 'Doanl',
    dept: 'EE',
    age: 22,
    gender: 'Male'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990c'),
    id: 3,
    name: 'Anu',
    dept: 'CS',
    age: 23,
    gender: 'Female'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990d'),
    id: 4,
    name: 'Karthika',
    dept: 'ME',
    age: 24,
    gender: 'Female'
  },
  {
    _id: ObjectId('660ee84b4181dffc519f990f'),
    id: 6,
    name: 'Ryan',
    dept: 'CS',
    age: 26,
    gender: 'Male'
  }
```

PROGRAM 18

AIM: Create database ‘candidate’ and collection ‘details’.

```
test> use candidate
switched to db candidate
candidate> db.createCollection("Details")
{ ok: 1 }
candidate>db.details.insert({"name":"Anu","age":21,"gender":"female","amount":7000});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f6da955ffdfc4748bf202') }
}
candidate>db.details.insert({"name":"Akhila","age":22,"gender":"female","amount":6000});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f6da955ffdfc4748bf203') }
}
candidate>db.details.insert({"name":"Arjun","age":32,"gender":"male","amount":20000});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f6da955ffdfc4748bf204') }
}
candidate>db.details.insert({"name":"Amal","age":45,"gender":"male","amount":40000});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f6da955ffdfc4748bf205') }
}
candidate>db.details.insert({"name":"Akash","age":53,"gender":"male","amount":50000});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f6da955ffdfc4748bf206') }
}
candidate> db.details.find()
[
  {cknowledged: true,
    _id: ObjectId('660f6da955ffdfc4748bf202'),c4748bf206') }
    name: 'Anu',
    age: 21,
    gender: 'female',
    amount: 7000
  },
  {
    _id: ObjectId('660f6da955ffdfc4748bf203'),
    name: 'Akhila',
    age: 22,
```

```
    gender: 'female',
    amount: 6000
  },
  {
    _id: ObjectId('660f6da955ffdfc4748bf204'),
    name: 'Arjun',
    age: 32,
    gender: 'male',
    amount: 20000
  },
  {
    _id: ObjectId('660f6da955ffdfc4748bf205'),
    name: 'Amal',
    age: 45,
    gender: 'male',
    amount: 40000
  },
  {
    _id: ObjectId('660f6dac55ffdfc4748bf206'),
    name: 'Akash',
    age: 53,
    gender: 'male',
    amount: 50000
  }
]
```

1. Query customer who are either male or younger than 25?

```
candidate> db.details.find({$or:[{'gender':'male'},{'age':{$lt:25}}]})
[
  {
    _id: ObjectId('660f6da955ffdfc4748bf202'),
    name: 'Anu',
    age: 21,
    gender: 'female',
    amount: 7000
  },
  {
    _id: ObjectId('660f6da955ffdfc4748bf203'),
    name: 'Akhila',
    age: 22,
    gender: 'female',
    amount: 6000
  },
  {
    _id: ObjectId('660f6da955ffdfc4748bf204'),
    name: 'Arjun',
    age: 32,
    gender: 'male',
```

```
    amount: 20000
  },
  {
    _id: ObjectId('660f6da955ffdfc4748bf205'),
    name: 'Amal',
    age: 45,
    gender: 'male',
    amount: 40000
  },
  {
    _id: ObjectId('660f6dac55ffdfc4748bf206'),
    name: 'Akash',
    age: 53,
    gender: 'male',
    amount: 50000
  }
]
```

2. Calculate total purchase amount for males and females using aggregate method

```
candidate> db.details.find({$or:[{'gender':'male'},{'age':{$lt:25}}]})
[
{ _id: 'male', 'total amount': 110000 },
  { _id: 'female', 'total amount': 13000 }
]
```

3. Select customers who are older than 25 and calculate the average purchase amount for males and females

```
candidate>
db.details.aggregate([{$match:{"age":{$gt:25}}},{$group:{_id:"$gender",'totalamount':{$avg:'
$amount'}}}])
[ { _id: 'male', totalamount: 36666.666666666664 } ]
```

4. sort the data based on average amount.

```
candidate>
db.details.aggregate([{$match:{"age":{$gt:25}}},{$group:{_id:"$gender",'totalamount':{$avg:'$am
ount'}}},{$sor$sort:{avg:1}}])
[ { _id: 'male', totalamount: 36666.666666666664 } ]
```

PROGRAM 19

AIM: Create a database named college and then create a collection named studlist. Insert some values into it .Write a MongoDB Query to:

```
test> use college;
switched to db college
college> db.createCollection("details");
{ ok: 1 }

college>
db.details.insertMany([{"fname":"Akhila","lname":"T","mark":"95","gender":"F","dept":"MCA","g
rade":"A+","contact":"9947558569","loc":"kollam"}])
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f7448401dc347898bf203') }
}
college>
db.details.insertMany([{"fname":"Arjun","lname":"P","mark":"82","gender":"M","dept":"mech","gr
ade":"A","contact":"9947558569","loc":"kannur"}])
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f7448401dc347898bf204') }
}
college>
db.details.insertMany([{"fname":"Anu","lname":"S","mark":"85","gender":"F","dept":"mech","grad
e":"A","contact":"9947558569","loc":"tvm"}])
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f7448401dc347898bf205') }
}
college>
db.details.insertMany([{"fname":"Pooja","lname":"m","mark":"85","gender":"F","dept":"mech","gr
ade":"A","contact":"9947558569","loc":"tvm"}])
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f7448401dc347898bf206') }
}
college>
db.details.insertMany([{"fname":"Adarsh","lname":"h","mark":"85","gender":"M","dept":"mech","
grade":"A","contact":"9947558569","loc":"pkd"}])
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f7448401dc347898bf207') }
}
```

```
college>
db.details.insertMany([{"fname":"Amal","lname":"h","mark":"78","gender":"M","dept":"MCA","gr
ade":"B","contact":"9947558569","loc":"pkd"}])
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f744a401dc347898bf208') }
}
college> db.details.find()
[
  {
    _id: ObjectId('660f7413401dc347898bf202'),
    fname: 'Arjun',
    lname: 'P',
    mark: '82',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'kannur'
  },
  {
    _id: ObjectId('660f7448401dc347898bf203'),
    fname: 'Akhila',
    lname: 'T',
    mark: '95',
    gender: 'F',
    dept: 'MCA',
    grade: 'A+',
    contact: '9947558569',
    loc: 'kollam'
  },
  {
    _id: ObjectId('660f7448401dc347898bf204'),
    fname: 'Arjun',
    lname: 'P',
    mark: '82',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'kannur'
  },
  {
    _id: ObjectId('660f7448401dc347898bf205'),
    fname: 'Anu',
    lname: 'S',
    mark: '85',
    gender: 'F',
    dept: 'mech',
```

```
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  },
  {
    _id: ObjectId('660f7448401dc347898bf206'),
    fname: 'Pooja',
    lname: 'm',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  },
  {
    _id: ObjectId('660f7448401dc347898bf207'),
    fname: 'Adarsh',
    lname: 'h',
    mark: '85',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'pkd'
  },
  {
    _id: ObjectId('660f744a401dc347898bf208'),
    fname: 'Amal',
    lname: 'h',
    mark: '78',
    gender: 'M',
    dept: 'MCA',
    grade: 'B',
    contact: '9947558569',
    loc: 'pkd'
  }
]
```

1. Display name (both fname and lname) and mark of all female students in MCA department.

```
college> db.details.find({dept:"MCA",gender:"F"}).pretty()
[
  {
    _id: ObjectId('660f7448401dc347898bf203'),
    fname: 'Akhila',
    lname: 'T',
    mark: '95',
    gender: 'F',
```

```
    dept: 'MCA',
    grade: 'A+',
    contact: '9947558569',
    loc: 'kollam'
  }
]
```

2. Display the details of student who secured highest mark in the course MCA

```
college> db.details.find({dept:"MCA"},{_id:0}).sort({mark:-1}).limit(1)
[
  {
    fname: 'Akhila',
    lname: 'T',
    mark: '95',
    gender: 'F',
    dept: 'MCA',
    grade: 'A+',
    contact: '9947558569',
    loc: 'kollam'
  }
]
```

3. Display all male students who secured A+ grade.

```
college>
db.details.find({grade:"A+",gender:"F"},{fname:1,lname:1,gender:1,dept:1,loc:1,grade:1,mark:1})
[
  {
    _id: ObjectId('660f7448401dc347898bf203'),
    fname: 'Akhila',
    lname: 'T',
    mark: '95',
    gender: 'F',
    dept: 'MCA',
    grade: 'A+',
    loc: 'kollam'
  }
]
```

4. Display the names of the top three students in Mechanical department.

```
college> db.details.find({dept:"mech"}).sort({mark:-1}).limit(3)
[
  {
    _id: ObjectId('660f7448401dc347898bf207'),
    fname: 'Adarsh',
    lname: 'h',
    mark: '85',
```

```
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'pkd'
  },
  {
    _id: ObjectId('660f7448401dc347898bf206'),
    fname: 'Pooja',
    lname: 'm',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  },
  {
    _id: ObjectId('660f7448401dc347898bf205'),
    fname: 'Anu',
    lname: 'S',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  }
]
```

5. Display the details of female students [fname,lname,grade,mark,contact] who achieved a mark more than 90.

```
college> db.details.find({mark:{$gt:90},gender:'F'},{fname:1,lname:1,mark:1,contact:1,grade:1})
[
  {
    _id: ObjectId('660f7448401dc347898bf203'),
    fname: 'Akhila',
    lname: 'T',
    mark: '95',
    gender: 'F',
    dept: 'MCA',
    grade: 'A+',
    loc: 'kollam'
  }
]
```

6. Display the details of students who secured mark, more than 80 but less than 90.

```
college>
db.details.find({$and:[{mark:{$gt:80}},{mark:{$lt:90}}]},{fname:1,lname:1,mark:1,contact:1,grad
e:1})
[
  {
    _id: ObjectId('660f7413401dc347898bf202'),
    fname: 'Arjun',
    lname: 'P',
    mark: '82',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'kannur'
  },
  {
    _id: ObjectId('660f7448401dc347898bf203'),
    fname: 'Akhila',
    lname: 'T',
    mark: '95',
    gender: 'F',
    dept: 'MCA',
    grade: 'A+',
    contact: '9947558569',
    loc: 'kollam'
  },
  {
    _id: ObjectId('660f7448401dc347898bf204'),
    fname: 'Arjun',
    lname: 'P',
    mark: '82',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'kannur'
  },
  {
    _id: ObjectId('660f7448401dc347898bf205'),
    fname: 'Anu',
    lname: 'S',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
```

```
  },
  {
    _id: ObjectId('660f7448401dc347898bf206'),
    fname: 'Pooja',
    lname: 'm',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  },
  {
    _id: ObjectId('660f7448401dc347898bf207'),
    fname: 'Adarsh',
    lname: 'h',
    mark: '85',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'pkd'
  }
 ]
```

7. Display the details of students whose name starts with ‘P’

```
college> db.details.find({fname:{$regex:'^P'}},{})
[
  {
    _id: ObjectId('660f7448401dc347898bf206'),
    fname: 'Pooja',
    lname: 'm',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  }
]
```

8. Display all students from Kollam

```
college> db.details.find({loc:"kollam"},{})
[
  {
    _id: ObjectId('660f7448401dc347898bf203'),
    fname: 'Akhila',
```

```
    lname: 'T',
    mark: '95',
    gender: 'F',
    dept: 'MCA',
    grade: 'A+',
    contact: '9947558569',
    loc: 'kollam'
  }
]
```

9. Display all students who does not belong to neither Kollam nor Thiruvananthapuram

```
college> db.details.find({$nor:[{loc:"kollam"},{loc:"tvm"}]},{})
[
  {
    _id: ObjectId('660f7413401dc347898bf202'),
    fname: 'Arjun',
    lname: 'P',
    mark: '82',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'kannur'
  },
  {
    _id: ObjectId('660f7448401dc347898bf204'),
    fname: 'Arjun',
    lname: 'P',
    mark: '82',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'kannur'
  },
  {
    _id: ObjectId('660f7448401dc347898bf207'),
    fname: 'Adarsh',
    lname: 'h',
    mark: '85',
    gender: 'M',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'pkd'
  },
  {
    _id: ObjectId('660f744a401dc347898bf208'),
```

```
    fname: 'Amal',
    lname: 'h',
    mark: '78',
    gender: 'M',
    dept: 'MCA',
    grade: 'B',
    contact: '9947558569',
    loc: 'pkd'
  }
]
```

10. Display all female students who belong to either Kollam or Thiruvananthapuram

```
college> db.details.find({$or:[{loc:"kollam"},{loc:"tvm"}],gender:"F"},{})
[
  {
    _id: ObjectId('660f7448401dc347898bf203'),
    fname: 'Akhila',
    lname: 'T',
    mark: '95',
    gender: 'F',
    dept: 'MCA',
    grade: 'A+',
    contact: '9947558569',
    loc: 'kollam'
  },
  {
    _id: ObjectId('660f7448401dc347898bf205'),
    fname: 'Anu',
    lname: 'S',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  },
  {
    _id: ObjectId('660f7448401dc347898bf206'),
    fname: 'Pooja',
    lname: 'm',
    mark: '85',
    gender: 'F',
    dept: 'mech',
    grade: 'A',
    contact: '9947558569',
    loc: 'tvm'
  }
]
```

PROGRAM 20

AIM: Create a database in MongoDB named "mcadb" with collections named "course" and "students" and perform aggregate functions, and regular expressions on it.

```
test> use mcadb
switched to db mcadb
mcadb> db.createCollection("course")
{ ok: 1 }
mcadb> db.createCollection("students")
{ ok: 1 }
mcadb> db.students.insertMany([
{ name: "Arjun", age: 22, gender: "Male" },
{ name: "Anu", age: 25, gender: "Female" },
{ name: "Jishnu", age: 28, gender: "Male" },
{ name: "Akhila", age: 30, gender: "Female" },
{ name: "Arun", age: 35, gender: "Male" }
])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('661428cbdfca1e307f9f990a'),
    '1': ObjectId('661428cbdfca1e307f9f990b'),
    '2': ObjectId('661428cbdfca1e307f9f990c'),
    '3': ObjectId('661428cbdfca1e307f9f990d'),
    '4': ObjectId('661428cbdfca1e307f9f990e')
  }
}
mcadb> db.students.find()
[
  {
    _id: ObjectId('6614d9adb0081094f2d14a0e'),
    name: 'Arjun',
    age: 22,
    gender: 'Male'
  },
  {
    _id: ObjectId('6614d9adb0081094f2d14a0f'),
    name: 'Anu',
    age: 25,
    gender: 'Female'
  },
  {
    _id: ObjectId('6614d9adb0081094f2d14a10'),
    name: 'Jishnu',
    age: 28,
    gender: 'Male'
  },
```

```
  {
    _id: ObjectId('6614d9adb0081094f2d14a11'),
    name: 'Akhila',
    age: 30,
    gender: 'Female'
  },
  {
    _id: ObjectId('6614d9adb0081094f2d14a12'),
    name: 'Arun',
    age: 35,
    gender: 'Male'
  }
]
mcadb> db.course.insertMany([
{ code: "ENG101", name: "Introduction to English", credits: 3 },
{ code: "MTH101", name: "Discrete Mathematics", credits: 4 },
{ code: "CSC101", name: "Introduction to Computer Science", credits: 4 },
{ code: "ENG201", name: "Advanced English", credits: 4 },
{ code: "ADB201", name: "ADBMS", credits: 4 },
{ code: "CSC201", name: "Data Structures", credits: 4 }
])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6614293edfca1e307f9f990f'),
    '1': ObjectId('6614293edfca1e307f9f9910'),
    '2': ObjectId('6614293edfca1e307f9f9911'),
    '3': ObjectId('6614293edfca1e307f9f9912'),
    '4': ObjectId('6614293edfca1e307f9f9913'),
    '5': ObjectId('6614293edfca1e307f9f9914')
  }
}
mcadb> db.course.find()
[
  {
    _id: ObjectId('6614d9d9b0081094f2d14a13'),
    code: 'ENG101',
    name: 'Introduction to English',
    credits: 3
  },
  {
    _id: ObjectId('6614d9d9b0081094f2d14a14'),
    code: 'MTH101',
    name: 'Discrete Mathematics',
    credits: 4
  },
  {
    _id: ObjectId('6614d9d9b0081094f2d14a15'),
    code: 'CSC101',
```

```
    name: 'Introduction to Computer Science',
    credits: 4
  },
  {
    _id: ObjectId('6614d9d9b0081094f2d14a16'),
    code: 'ENG201',
    name: 'Advanced English',
    credits: 4
  },
  {
    _id: ObjectId('6614d9d9b0081094f2d14a17'),
    code: 'ADB201',
    name: 'ADBMS',
    credits: 4
  },
  {
    _id: ObjectId('6614d9d9b0081094f2d14a18'),
    code: 'CSC201',
    name: 'Data Structures',
    credits: 4
  }
]
```

1. Calculate the average age of students

```
mcadb> db.students.aggregate([{$group:{_id:null,avgAge:{$avg:"$age"}}}])
[ { _id: null, avgAge: 28 } ]
```

2. Count the number of male and female students

```
mcadb> db.students.aggregate([{$group:{_id:"$gender",count:{$sum:1}}}])
[ { _id: 'Male', count: 3 }, { _id: 'Female', count: 2 } ]
```

3. Find the courses with the highest number of credits

```
mcadb> db.course.aggregate([{ $sort: { credits: -1 } },{ $limit: 1 }])
[
  {
    _id: ObjectId('6614d9d9b0081094f2d14a14'),
    code: 'MTH101',
    name: 'Discrete Mathematics',
    credits: 4
  }
]
```

4. Find students whose names start with "J"

```
mcadb> db.students.find({ name:{$regex:/^J/i}})
[
  {
```

```
    _id: ObjectId('661428cbdfca1e307f9f990c'),
    name: 'Jishnu',
    age: 28,
    gender: 'Male'
  }
]
```

5. Find courses with codes containing "ENG"

```
mcadb> db.course.find({code:{$regex:/ENG/i}})
[
  {
    _id: ObjectId('6614293edfca1e307f9f990f'),
    code: 'ENG101',
    name: 'Introduction to English',
    credits: 3
  },
  {
    _id: ObjectId('6614293edfca1e307f9f9912'),
    code: 'ENG201',
    name: 'Advanced English',
    credits: 4
  }
]
```

Course Outcome 4

Understand the basic storage architecture of distributed file systems

PROGRAM 21

AIM: Build collections mcaDB documents students,course and perform shell commands to create replicaset, indexing etc

```
test> use mca
switched to db mca
mca> db.createCollection("students")
{ ok: 1 }
mca> db.createCollection("course")
{ ok: 1 }
mca> db.students.insert({ "name": "John", "age": 25, "course": "MCA" })
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('663526abd2a4a954b39f990a') }
}
mca> db.course.insert({ "courseName": "Database Systems", "credits": 3 })
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('663526abd2a4a954b39f990b') }
}
mca> db.students.createIndex({ "name": 1 })
name_1
mca> db.students.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { rollNo: 1 }, name: 'rollNo_1' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]
```

Course Outcome 5

Design and deployment of NoSQL databases with real time requirements.

PROGRAM 22

AIM: Develop students' marks calculation applications using Python and MongoDB

```
import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["student_marks"]
collection = db["students"]

def enter_student_data():
    name = input("Enter student name: ")
    dbms = float(input("Enter DBMS marks: "))
    oops = float(input("Enter OOPS marks: "))
    networks = float(input("Enter Networks marks: "))

    student_data =
    { "name": name,
    "dbms": dbms,

    "oops": oops,
    "networks": networks
    }
    collection.insert_one(student_data)
    print("Student data entered successfully!")

def calculate_student_marks():
    name = input("Enter student name to calculate marks: ")
    student = collection.find_one({"name": name})

    if student:
        total_marks = student["dbms"] + student["oops"] + student["networks"]
        print(f"Total marks for {name}: {total_marks}")
    else:
        print(f"Student '{name}' not found!")

def view_students():
```

```
    print("List of Students:")
    for student in collection.find():
        print(f"Name: {student['name']}, DBMS: {student['dbms']}, OOPS:{student['oops']},
Networks: {student['networks']}")
def main():

    while True:
        print("\nStudent Marks Calculation Application") print("1.
        Enter student data")
        print("2. Calculate student marks")
        print("3. View students")
        print("4. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            enter_student_data()
        elif choice == "2":
            calculate_student_marks()
        elif choice == "3":
            view_students()
        elif choice == "4":
            print("Exiting the application.")
            break
        else:
            print("Invalid choice. Please try again.")
if _name___== "_main_":
    main()
```

Output

```
Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice: 1
Enter student name: AKHILA
Enter DBMS marks: 56
Enter OOPS marks: 78
Enter Networks marks: 79
Student data entered successfully!
```

Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice: 2
Enter student name to calculate marks: AKHILA
Total marks for AKHILA: 213.0

Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice: 3
List of Students:
Name: AKHILA, DBMS: 56.0, OOPS:78.0, Networks: 79.0

Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice:4

Muthoot Institute of Technology and Science

Varikoli P.O, Puthencruz,

Ernakulam – 682308

Master of Computer Applications

Micro Project Report

Submitted by

AMAL TOM PARAKKADEN

ANJALI SUNIL KUMAR

ARCHANA AJIT

DIVYA.D

SUHANA BAIJAN P A

SAFAL A

Acknowledgement

We would like to express our sincere gratitude to all those who supported and guided us throughout the development of the EMPLOYEE LEAVE MANAGEMENT web application. This project would not have been possible without their invaluable contributions and assistance.

We are deeply thankful to our mentor Mrs Jiss Kuruvilla, for the expertise, guidance, and encouragement. We are grateful to our peers and friends who provided valuable insights, engaged in meaningful discussions, and offered their assistance whenever needed. Their diverse perspectives enriched the project's outcome.

We would like to acknowledge the open-source community for developing and maintaining the libraries, frameworks, and tools that we utilized in this project. Their contributions have greatly facilitated the implementation process.

INDEX

1. Introduction

This microproject is dedicated to the refinement and enhancement of leave management within the broader context of Human Resource Management (HRM). Recognizing the pivotal role of effective leave management in fostering organizational productivity and employee satisfaction, the project focuses on developing a sophisticated Leave Management System tailored to the unique needs of modern workplaces. Through specialized modules catering to Employees, Managers, and the CEO, the system aims to centralize and streamline the leave approval process, thereby ensuring seamless coordination and transparency across all hierarchical levels. By integrating advanced features such as automated notifications and comprehensive leave analytics, the system empowers HR professionals to efficiently handle leave requests, optimize workforce planning, and make data-driven decisions to drive organizational success.

At its core, this microproject represents a concerted effort to elevate the standards of leave management practices within HRM. By leveraging technology to automate and enhance traditional leave administration processes, the system not only reduces administrative burden but also facilitates a more agile and responsive approach to managing employee leave. Through its emphasis on transparency, accountability, and efficiency, the Leave Management System aims to serve as a cornerstone of HRM, empowering organizations to cultivate a positive work environment, maintain high levels of employee satisfaction, and ultimately achieve their strategic objectives with greater efficacy.

2. Key Features

Employee Module:

Submit leave requests specifying type and duration.

View the status of submitted requests.

Receive notifications on request approval/rejection.

Manager Module:

Receive leave requests from employees.

Review and approve requests or forward to the CEO.

Access leave history and reports for team members.

CEO Module:

Receive forwarded leave requests from managers.

Review and approve leave requests.

Access comprehensive leave analytics and reports.

Customizable Leave Policies:

- Define and configure various types of leave such as annual leave, sick leave, maternity/paternity leave, etc.
- Set up rules for accrual rates, carryover limits, and eligibility criteria based on organizational policies.

Automated Approval Workflow:

- Design flexible approval workflows based on organizational hierarchy and departmental structures.
- Route leave requests to the appropriate managers for approval, with options for delegation and escalation.

Real-Time Notifications:

- Send automatic notifications to employees upon submission, approval, rejection, or modification of leave requests.
- Notify managers of pending leave requests and upcoming leave schedules for better planning.

Leave Calendar:

- Provide a centralized calendar view to visualize employee leave schedules and identify potential conflicts.
- Allow managers to make informed decisions about leave approvals based on team availability.

Compliance Management:

- Ensure compliance with company policies, labor laws, and regulatory requirements.
- Automatically enforce leave policies and track usage to prevent unauthorized leaves and mitigate compliance risks.

Integration with HRIS and Payroll Systems:

- Seamlessly integrate with existing HRIS (Human Resource Information System) and payroll systems to synchronize employee data and leave balances.
- Streamline payroll processing by automatically updating leave balances and deductions.

Reporting and Analytics:

- Generate customizable reports on leave utilization, trends, and patterns to gain insights into workforce management.
- Monitor attendance, absenteeism, and productivity metrics to identify areas for improvement.

Employee Engagement Tools:

- Provide self-service features for employees to manage their profiles, preferences, and notifications.
- Offer feedback mechanisms and surveys to gather employee input and enhance user experience.

3.Benefits

Streamlined Booking Process:

- Effortless Booking: Customers can conveniently book train tickets from anywhere, reducing the need to visit physical ticket counters or stations.
- Time-saving: The online booking system saves time for both customers and railway staff by automating the ticketing process, thereby reducing long queues and waiting times.

Convenient Management:

- Centralized System: The system provides administrators with a centralized platform to manage train schedules, seat availability, and bookings, streamlining operations and reducing administrative overhead.
- Real-time Updates: Admins can quickly update train details, such as schedules or seat availability, ensuring that customers have access to the latest information

Improved User Experience:

- User-friendly Interface: The intuitive and easy-to-navigate interface enhances the overall user experience, making it simple for customers to search for trains, book tickets, and manage their bookings.
- Customization: The system allows customers to personalize their booking preferences, such as selecting preferred seats or specifying meal options, enhancing their satisfaction and loyalty.
- Prompt Notifications: Automated notifications regarding booking confirmations, seat assignments, and any changes in train schedules keep customers informed and engaged throughout the booking process.

Efficiency Improvement: An automated leave management system streamlines the entire process, reducing the time and effort spent on manual tasks such as leave request submission, approval routing, and tracking. This efficiency improvement allows HR personnel to focus on more strategic initiatives.

Enhanced Accuracy: Automation reduces the likelihood of errors associated with manual data entry and calculations. Accurate tracking of leave balances, entitlements, and usage ensures that employees receive the correct benefits and prevents disputes over discrepancies.

Compliance Assurance: Leave management systems can be configured to enforce company policies and legal regulations, ensuring that leave requests are processed consistently and in accordance with applicable laws. This reduces the risk of non-compliance penalties and litigation.

Transparency and Visibility: Both employees and managers have access to real-time information regarding leave balances, request statuses, and approvals. This transparency fosters trust and communication within the organization, as employees know exactly where they stand regarding their leave entitlements and managers can make informed decisions about staffing levels.

Employee Empowerment: Self-service portals empower employees to manage their leave requests autonomously, without the need for constant intervention from HR personnel. This independence promotes a sense of ownership and accountability among employees, leading to higher satisfaction and engagement.

Cost Savings: By streamlining processes, reducing errors, and improving efficiency, an employee leave management system can result in significant cost savings for the organization. These savings come from reduced administrative overhead, increased productivity, and better resource allocation.

Strategic Insights: Leave management systems often come equipped with reporting and analytics capabilities that provide valuable insights into leave trends, patterns, and utilization rates. HR administrators can use this data to identify areas for improvement, optimize staffing levels, and make data-driven decisions to better manage workforce resources.

Flexibility and Adaptability: Modern leave management systems are flexible and can be customized to meet the unique needs of different organizations and industries. They can accommodate various types of leave, including vacation, sick leave, maternity/paternity leave, and unpaid leave, as well as complex leave policies and accrual rules.

4.Abstract

The Leave Management System aims to automate and streamline the process of handling employee leave requests within the organization. By incorporating separate modules for Employees, Managers, and the CEO, the system ensures a structured and transparent leave approval workflow, enhancing organizational efficiency and employee satisfaction

Managing employee leave efficiently is crucial for organizational productivity and employee satisfaction. Traditional leave management systems often rely on manual processes, leading to inefficiencies and errors. To address these challenges, this project introduces StreamlineHR, a modern employee leave management system. StreamlineHR offers a comprehensive platform that automates leave requests, approval workflows, and tracking, while providing real-time updates and customizable reporting tools. By streamlining the leave management process, StreamlineHR enhances efficiency, improves compliance, and enhances the employee experience. This abstract provides an overview of the features and benefits of StreamlineHR, highlighting its potential to revolutionize leave management and optimize workforce management practices.

5.Existing System Study

1. Overview of the Current System:

- Brief description of the existing leave management system.
- How leave requests are currently submitted, processed, and tracked.
- Any manual processes involved in managing leave requests.
- Challenges faced by employees and HR personnel with the current system.

2. Process Flow Analysis:

- Step-by-step analysis of the leave management process.
- Identification of key stakeholders involved in leave request submission and approval.
- Mapping out the flow of information and approval hierarchy.
- Analysis of the time taken to process leave requests and any bottlenecks in the system.

3. Technology Infrastructure:

- Overview of the technology stack used in the existing system.
- Description of any software applications, databases, or spreadsheets used for leave management.
- Integration with other HR systems or payroll software.

- Assessment of the scalability and flexibility of the current technology infrastructure.

4. User Experience Evaluation:

- Feedback from employees regarding the usability of the current leave management system.
- Common pain points or usability issues reported by users.
- Employee satisfaction levels with the existing system.
- Comparison of user experience across different departments or teams.

5. Compliance and Policy Adherence:

- Analysis of how leave policies are enforced and communicated within the organization.
- Compliance with legal regulations and labor laws related to leave entitlements, accruals, and usage.
- Documentation of any instances of policy violations or inconsistencies.

6. Data Management and Reporting:

- Examination of how leave data is stored, managed, and accessed.
- Reporting capabilities of the existing system (e.g., leave balances, usage reports).
- Accuracy and reliability of leave data for payroll and compliance purposes.

- Identification of any gaps in reporting or analytics functionality.

7. Cost and Resource Analysis:

- Assessment of the time and resources allocated to managing leave requests.
- Calculation of the cost associated with manual processes, administrative overhead, and errors.
- Comparison of the total cost of ownership of the existing system versus potential alternatives.

6. Modules

❖ **Admin Panel:**

This module provides functionalities for system administrators to manage various aspects of the railway reservation system.

- **Login**: Secure authentication for administrators to access the admin panel. **Admin management** : Allows admins to add, edit, and delete leave details, including the employee name, leave reject/approve..
- View employee leave Management: View and manage leave type including the ability to cancel leave if necessary.
- User Management: Manage user accounts, including adding new users and modifying existing ones.
- Reports: Generate reports on employee schedules, leave , and other system data for analysis and decision-making.

❖ **Customer Interface:**

This module provides functionalities for customers to search for leavetype, apply for leave, and manage their leave.

- User Authentication: Secure login for customers to access the booking functionalities.
- Leave type Search: Enables customers to search for leave type based on starting date, ending date, .
- Apply leave : Allows customers to select their desired leave , specify user details, and confirm their leave.

7. Hardware & Software Requirements

Hardware Requirements:

Server:

- A dedicated or virtual server is recommended to host the employee leave management
- Minimum hardware specifications include:
  - Processor: Dual-core or higher processor.
  - RAM: 2 GB or more for smooth performance.
  - Storage: Sufficient disk space to store system files and database backups.
- Network Infrastructure:
- Stable internet connectivity is essential to ensure uninterrupted access to the system.
- Adequate network bandwidth to handle concurrent user requests during peak hours.

Software Requirements:

Operating System:

- The server should run a stable and secure operating system capable of hosting web applications. Recommended choices include:
  - Linux distributions (e.g., Ubuntu Server, CentOS)
  - Windows Server

Web Server:

- Apache HTTP Server or compatible web server software is required to host the PHP-based web application.
- Configuration of the web server to handle PHP scripts is necessary for proper functionality.

Database Management System (DBMS):

- MySQL or compatible relational database management system is needed to store and manage data related to trains, users, bookings, and passengers.
- Ensure compatibility with the chosen web server and PHP version.

Server-Side Scripting Language:

- PHP (Hypertext Preprocessor) is the primary scripting language used for server-side processing in the railway reservation system.
- The server should have PHP installed and configured to execute PHP scripts.

Client:

- Any modern web browser with JavaScript enabled is compatible with the employe leave management
- Compatibility with popular browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge,

8. Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a visual representation that depicts the flow of data within a system or process. It's a graphical tool used to model the interactions and transformations of data as it moves through various components of a system.

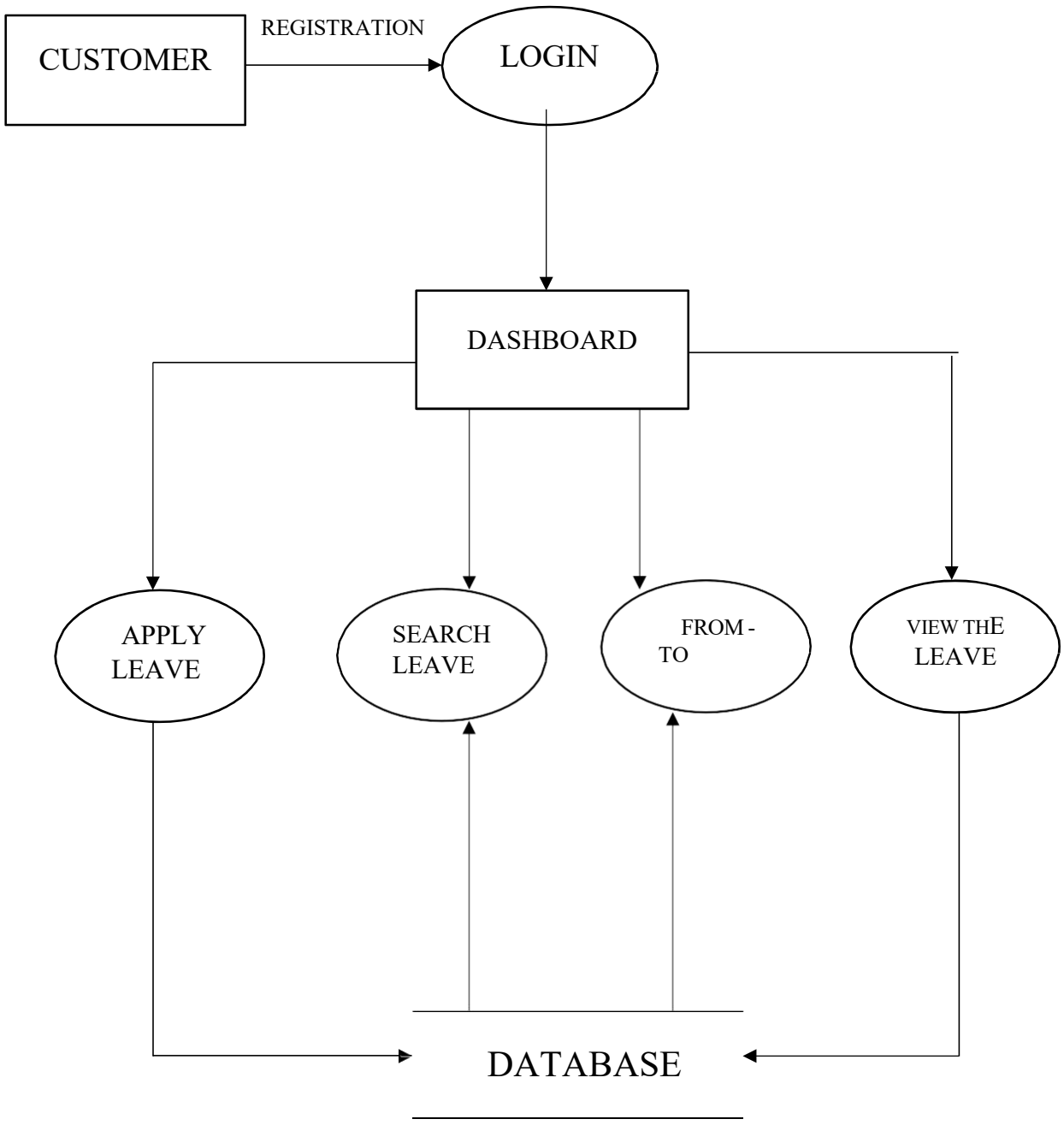
- **Zero th Level Data Flow Diagram**

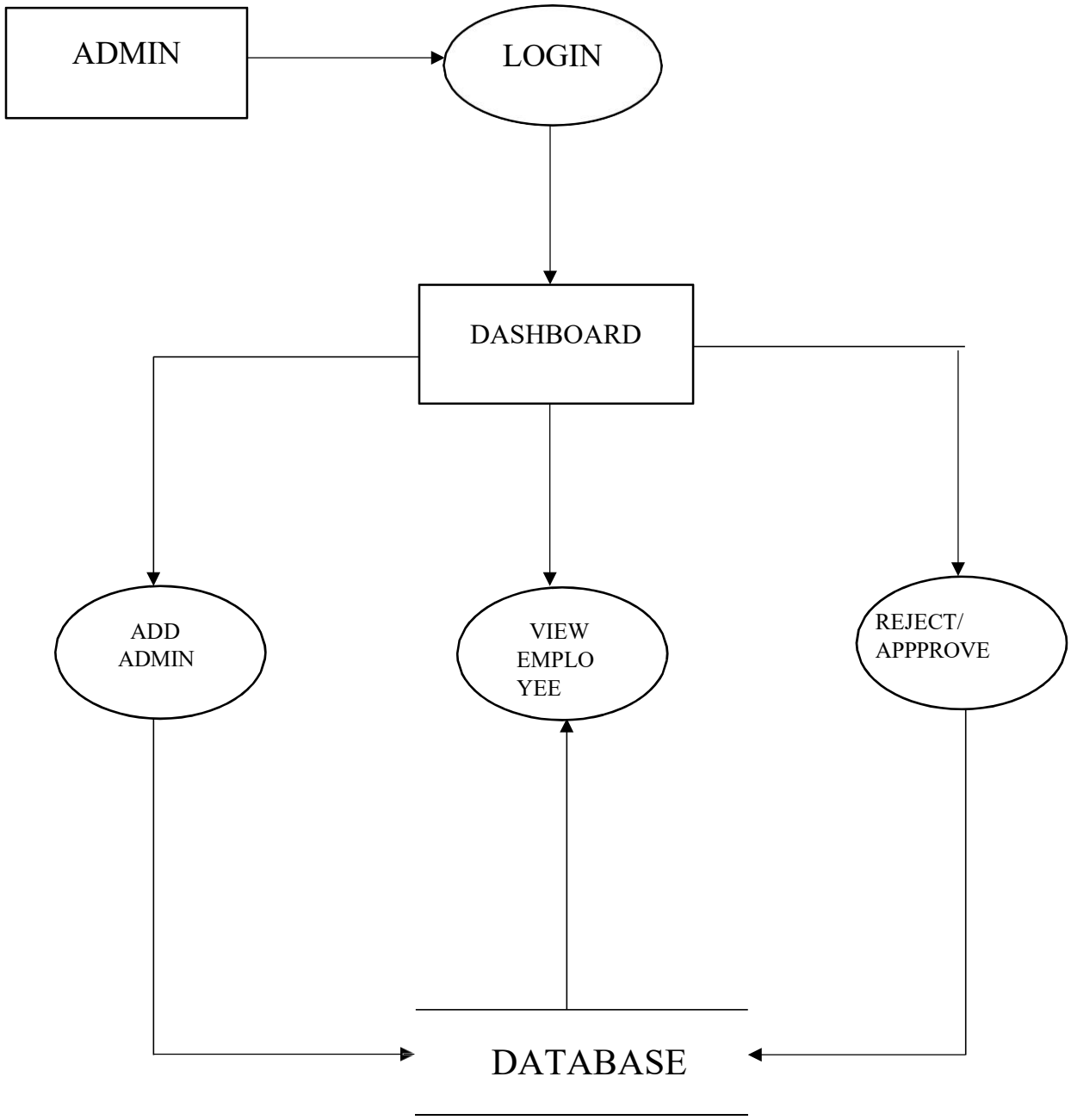
Provides an overview of the entire system or process at a high level, showing the interactions between system and user. It's a simplified diagram that presents a top-level perspective without going into the details of internal processes.

- **First Level Data Flow Diagram**

A First-Level Data Flow Diagram (DFD) is a visual representation of the most essential processes and data flows within a system or process. It provides a high-level overview of how data moves between major components of the system without delving into intricate details. First-Level DFDs are often used as an initial step in the process of system analysis and design.

FIRST LEVEL DFD

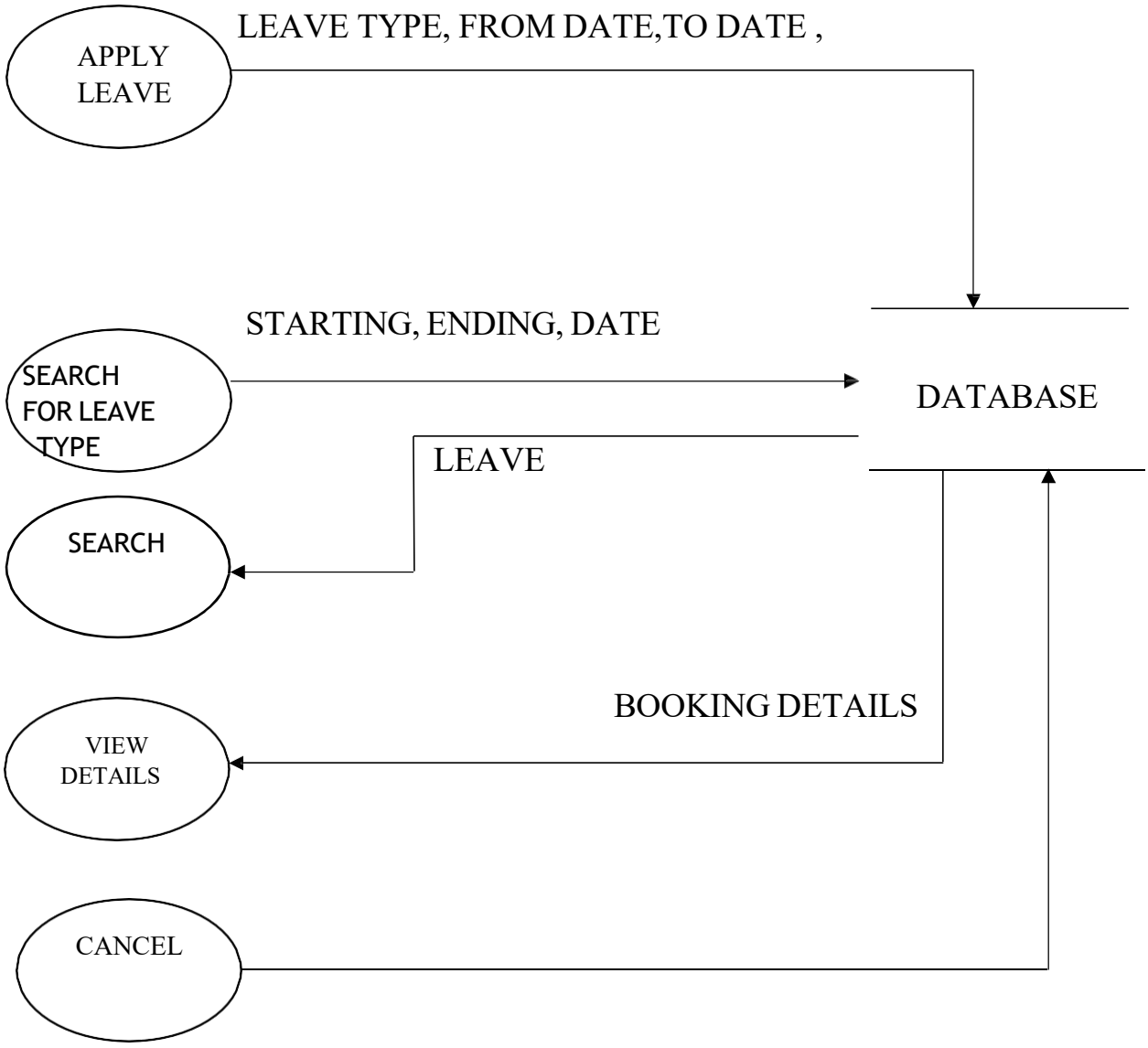



ADMIN
LOGIN
DASHBOARD
ADD ADMIN
VIEW EMPLO YEE
REJECT/ APPPROVE
DATABASE

- **Second Level Data Flow Diagram**

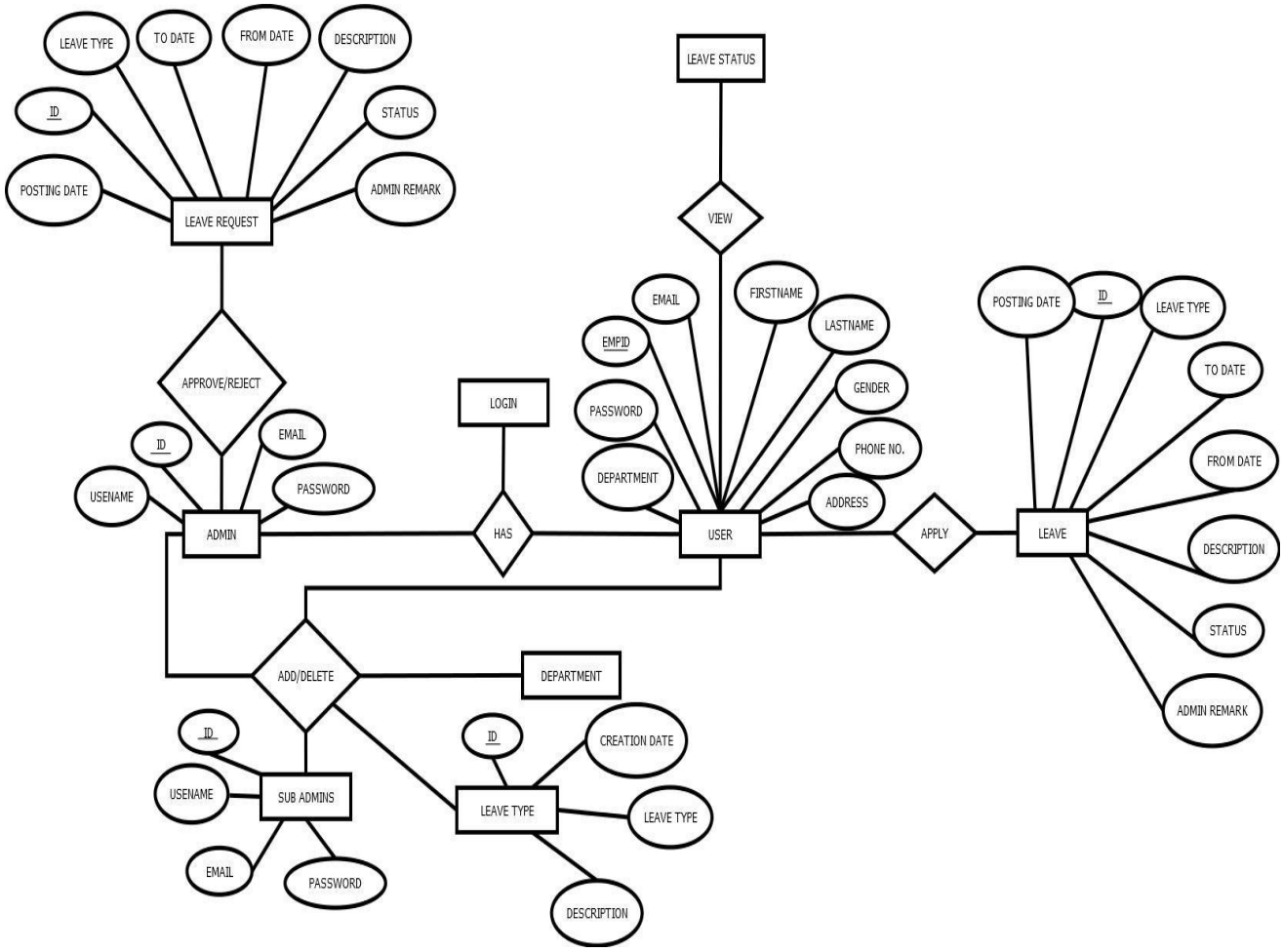
A Second-Level Data Flow Diagram (DFD) is a more detailed representation of specific processes and data flows within a system that were initially outlined in a First-Level DFD. It provides a deeper insight into the system's functionality by breaking down major processes from the first-level diagram into their sub-processes and interactions.

SECOND LEVEL DFD



9. Entity Relationship Diagram

An Entity-Relationship Diagram (ERD) is a visual representation of the relationships between different entities (objects or concepts) within a system or database. ERDs are commonly used to model the structure of databases and the relationships between the data entities.



10. Table Designs

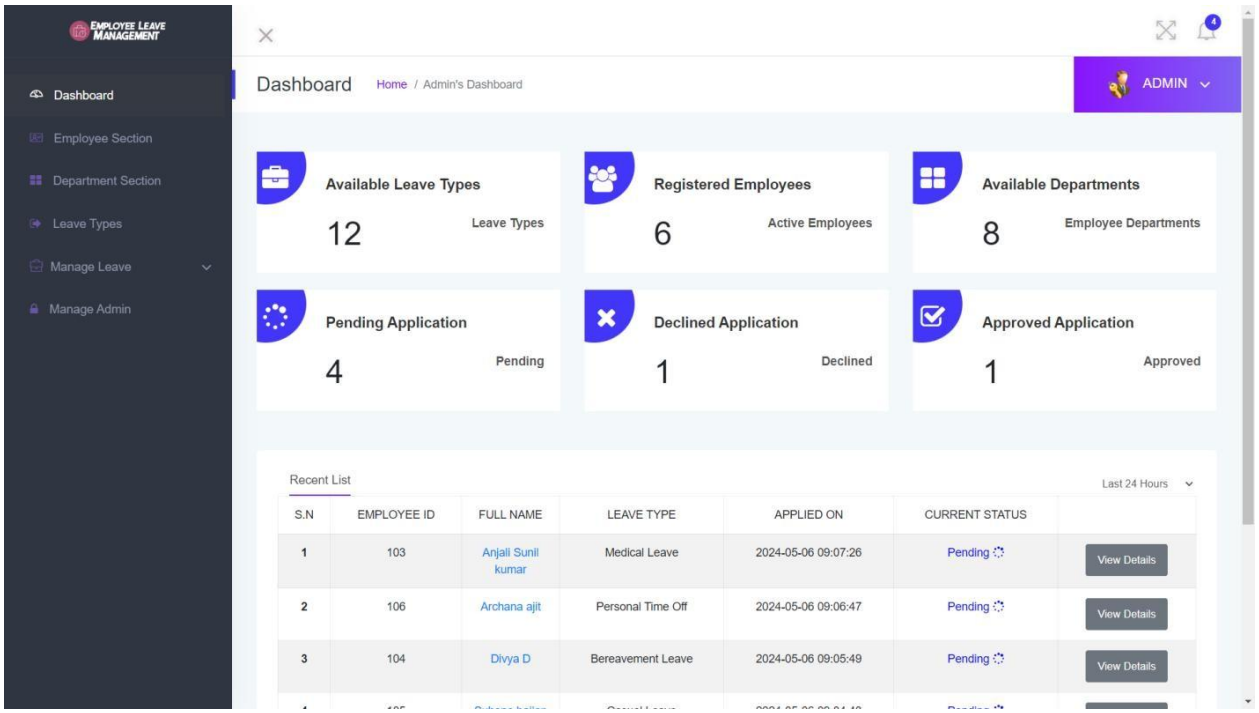
Admin Table

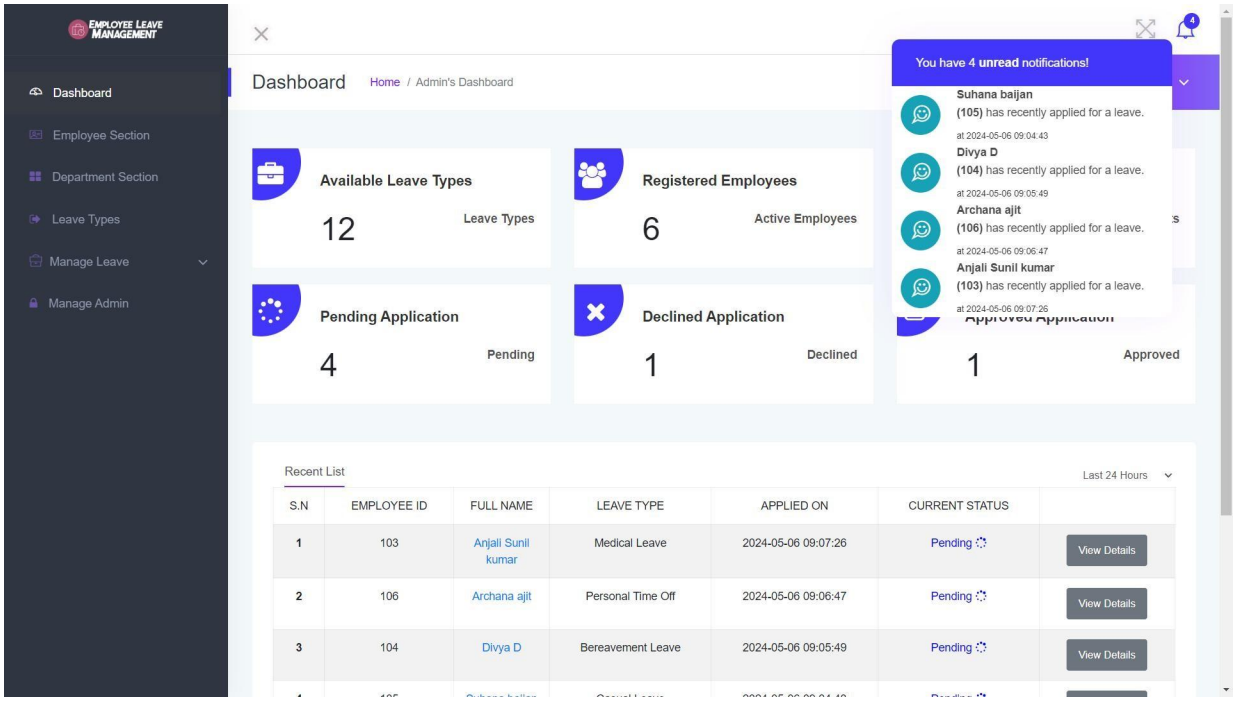
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	UserName	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
3	Password	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
4	fullname	varchar(255)	latin1_swedish_ci		No	None			Change Drop More
5	email	varchar(55)	latin1_swedish_ci		No	None			Change Drop More
6	updationDate	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change Drop More

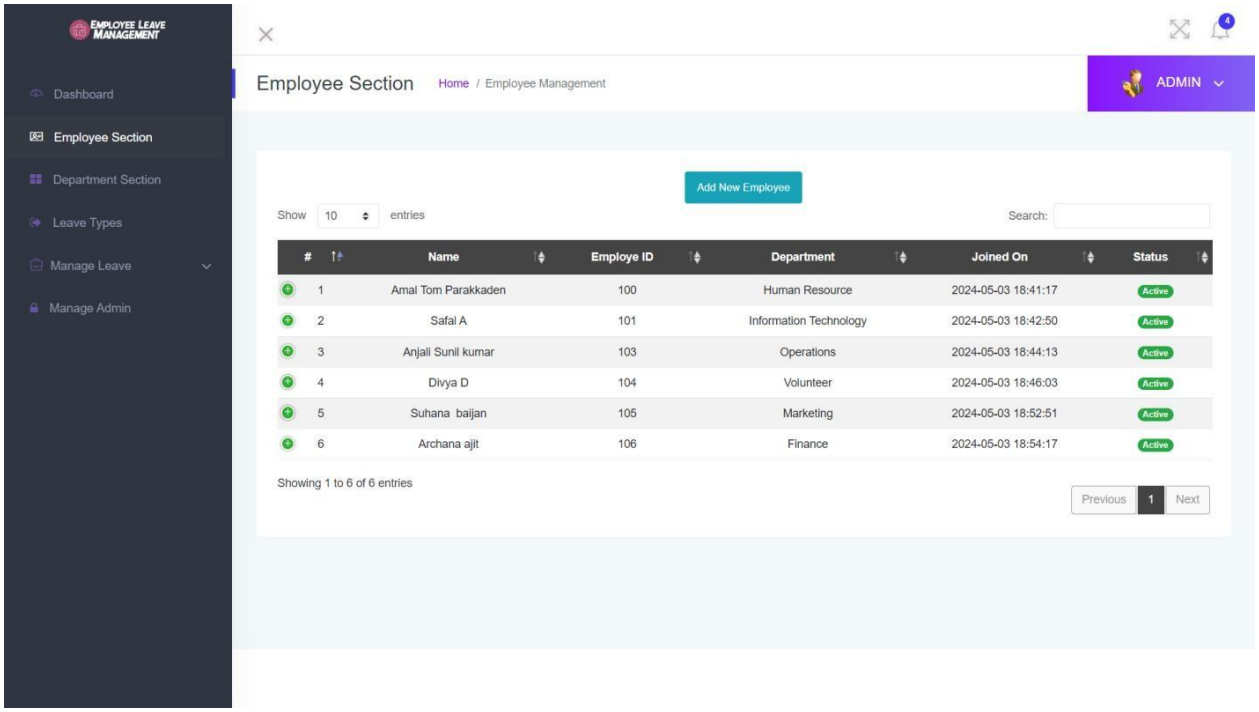
Employee Table

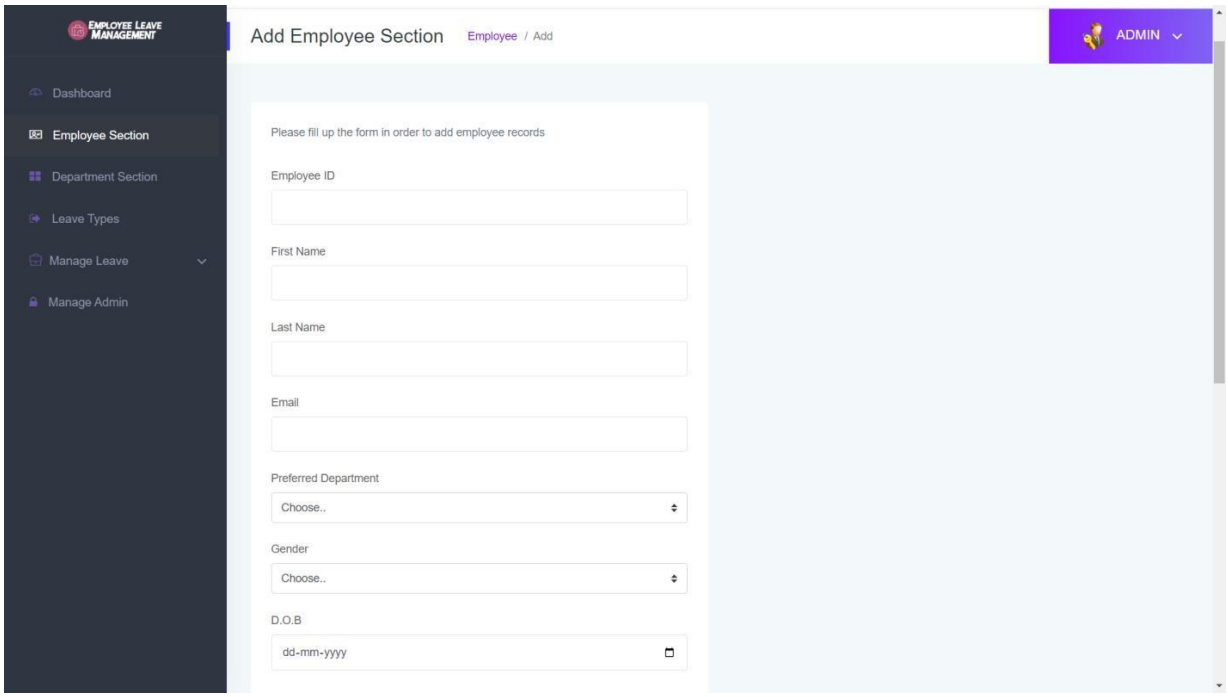
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	EmpId	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
3	FirstName	varchar(150)	latin1_swedish_ci		No	None			Change Drop More
4	LastName	varchar(150)	latin1_swedish_ci		No	None			Change Drop More
5	EmailId	varchar(200)	latin1_swedish_ci		No	None			Change Drop More
6	Password	varchar(180)	latin1_swedish_ci		No	None			Change Drop More
7	Gender	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
8	Dob	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
9	Department	varchar(255)	latin1_swedish_ci		No	None			Change Drop More
10	Address	varchar(255)	latin1_swedish_ci		No	None			Change Drop More
11	City	varchar(200)	latin1_swedish_ci		No	None			Change Drop More
12	Country	varchar(150)	latin1_swedish_ci		No	None			Change Drop More
13	Phonenumber	char(11)	latin1_swedish_ci		No	None			Change Drop More
14	Status	int(1)			No	None			Change Drop More
15	RegDate	timestamp			No	current_timestamp()			Change Drop More

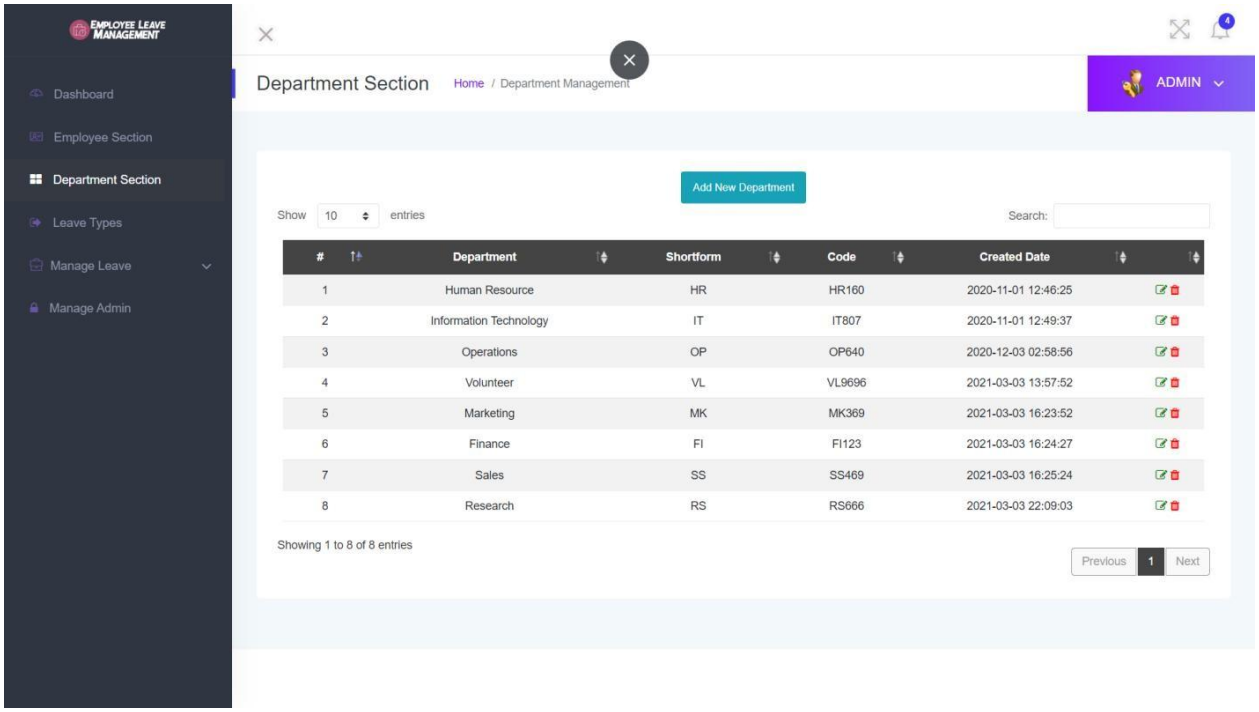
11. Screenshots

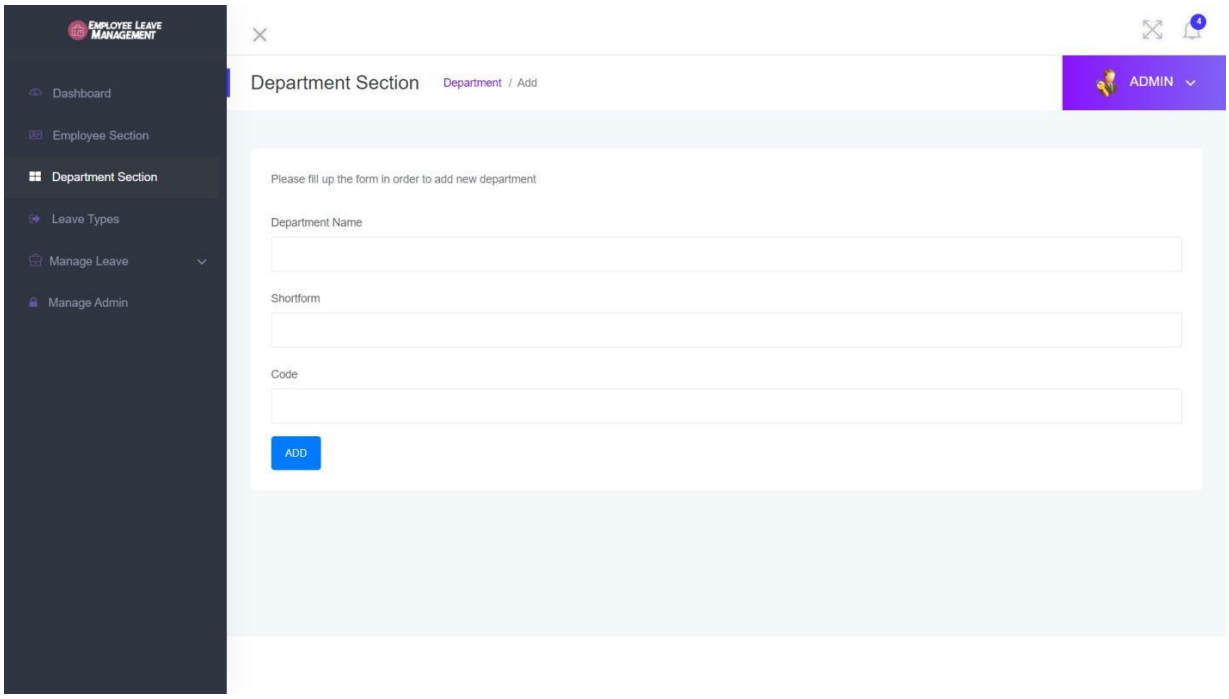


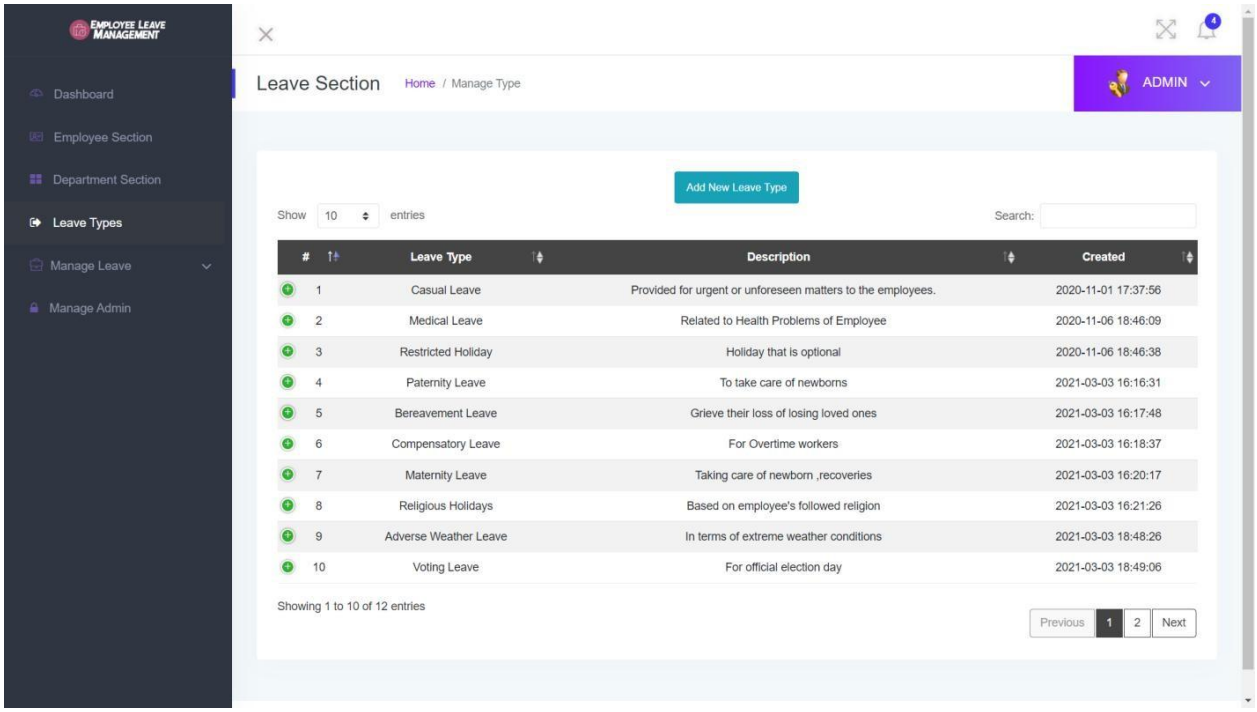


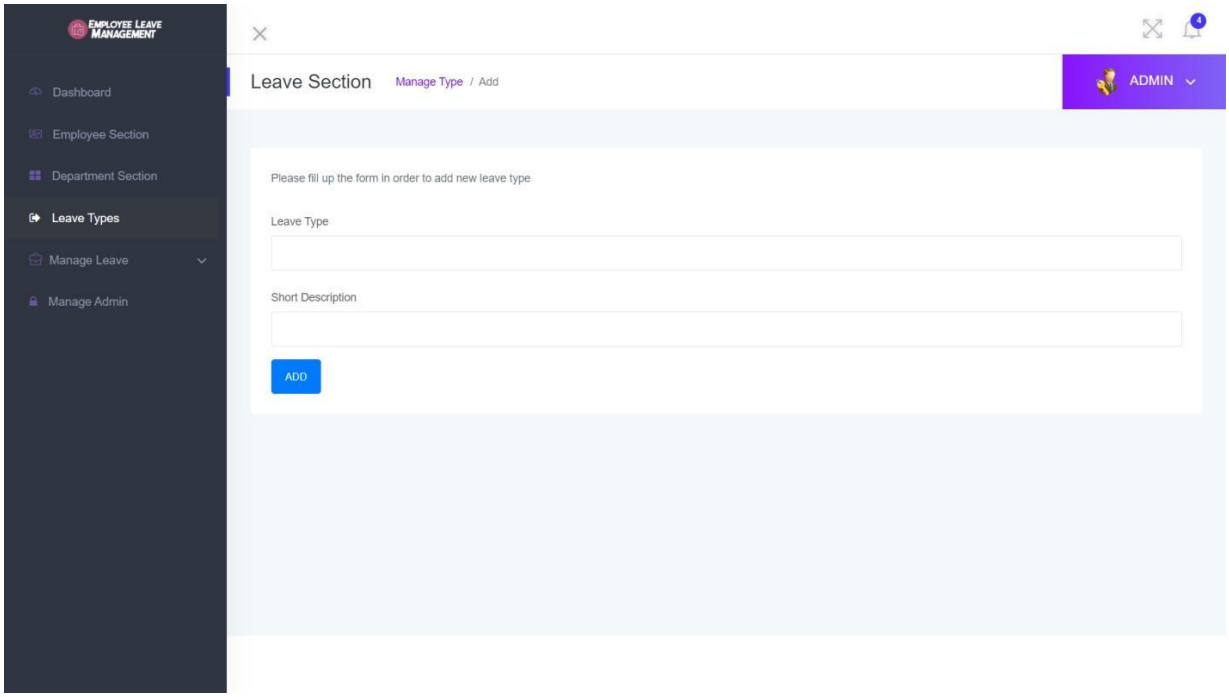

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Employee Section
Home / Employee Management
ADMIN
Add New Employee
Show 10 entries
Search:
Name Employe ID Department Joined On Status
1 Amal Tom Parakkaden 100 Human Resource 2024-05-03 18:41:17 Active
2 Safal A 101 Information Technology 2024-05-03 18:42:50 Active
3 Anjali Sunil kumar 103 Operations 2024-05-03 18:44:13 Active
4 Divya D 104 Volunteer 2024-05-03 18:46:03 Active
5 Suhana baijan 105 Marketing 2024-05-03 18:52:51 Active
6 Archana ajit 106 Finance 2024-05-03 18:54:17 Active
Showing 1 to 6 of 6 entries
Previous 1 Next

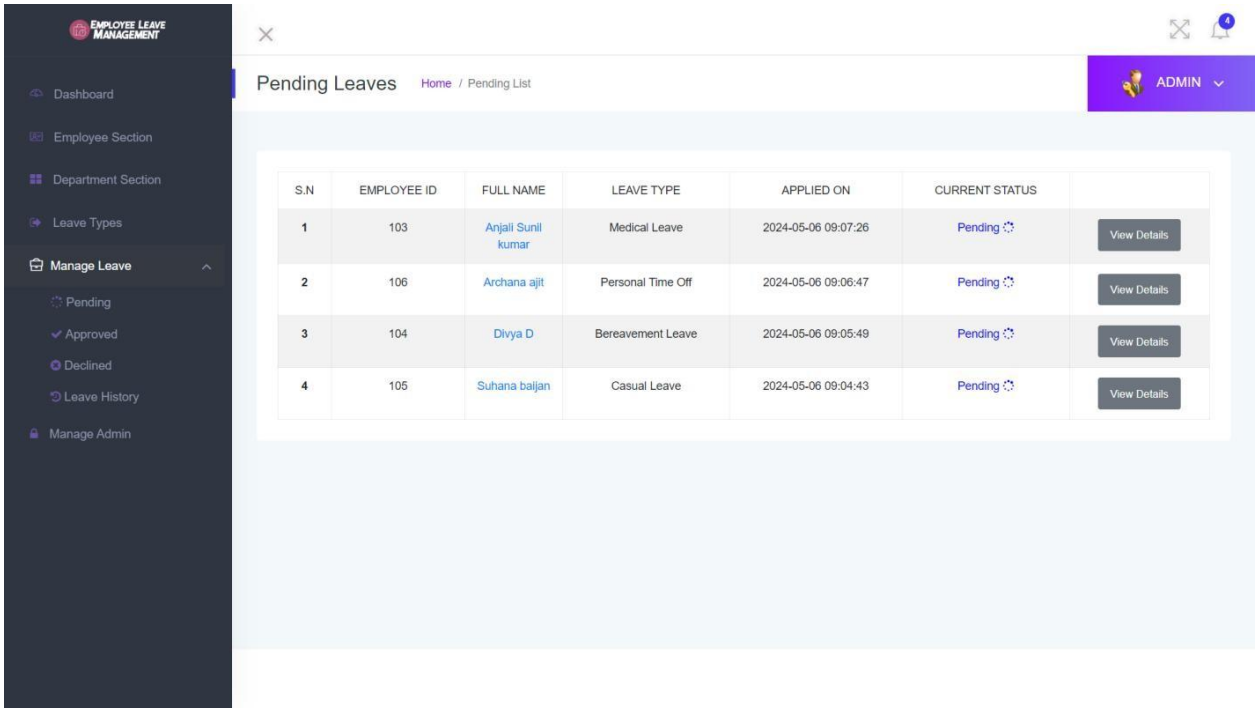

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Add Employee Section
Employee / Add
ADMIN
Please fill up the form in order to add employee records
Employee ID
First Name
Last Name
Email
Preferred Department
Choose..
Gender
Choose..
D.O.B
dd-mm-yyyy

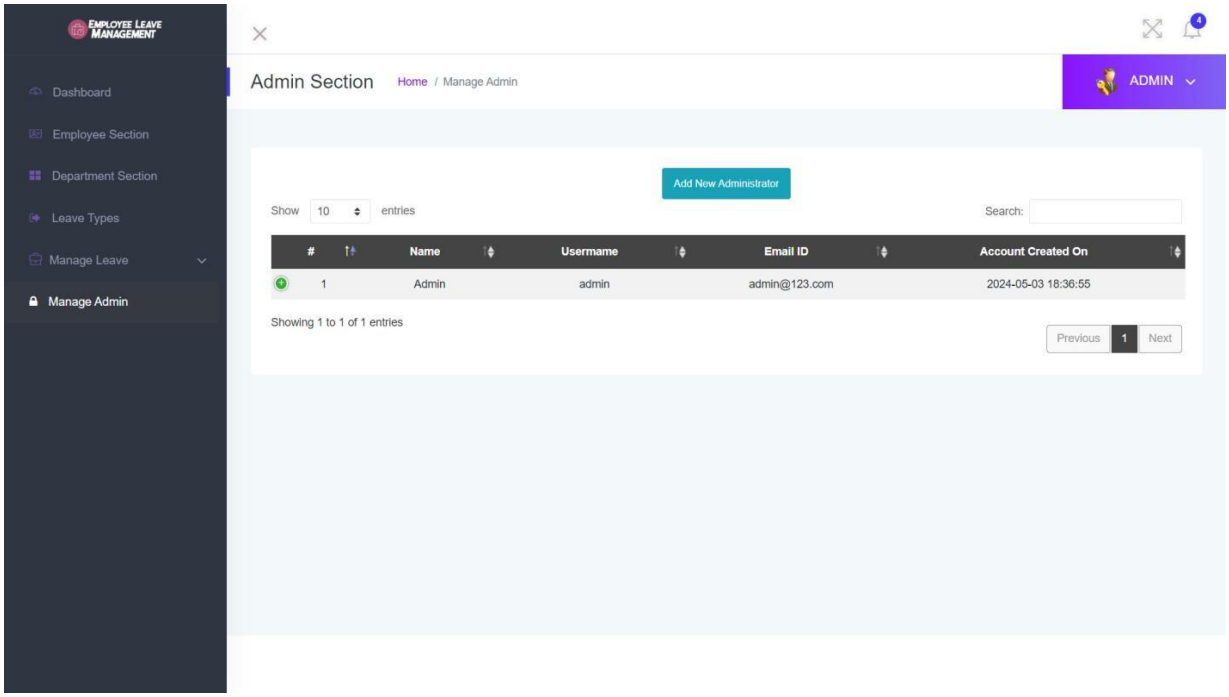

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Department Section
Home / Department Management
ADMIN
Add New Department
Show 10 entries
Search:
Department Shortform Code Created Date
1 Human Resource HR HR160 2020-11-01 12:46:25
2 Information Technology IT IT807 2020-11-01 12:49:37
3 Operations OP OP640 2020-12-03 02:58:56
4 Volunteer VL VL9696 2021-03-03 13:57:52
5 Marketing MK MK369 2021-03-03 16:23:52
6 Finance FI FI123 2021-03-03 16:24:27
7 Sales SS SS469 2021-03-03 16:25:24
8 Research RS RS666 2021-03-03 22:09:03
Showing 1 to 8 of 8 entries
Previous 1 Next

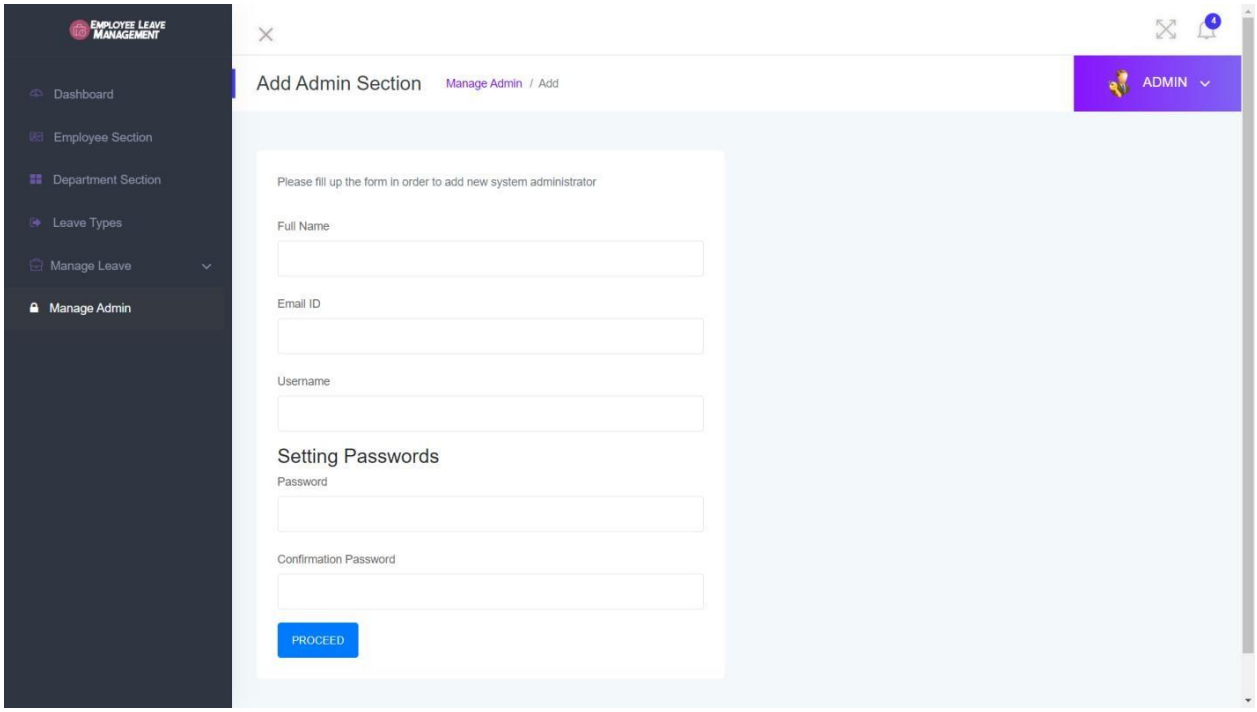

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Department Section
Department / Add
ADMIN
Please fill up the form in order to add new department
Department Name
Shortform
Code
ADD

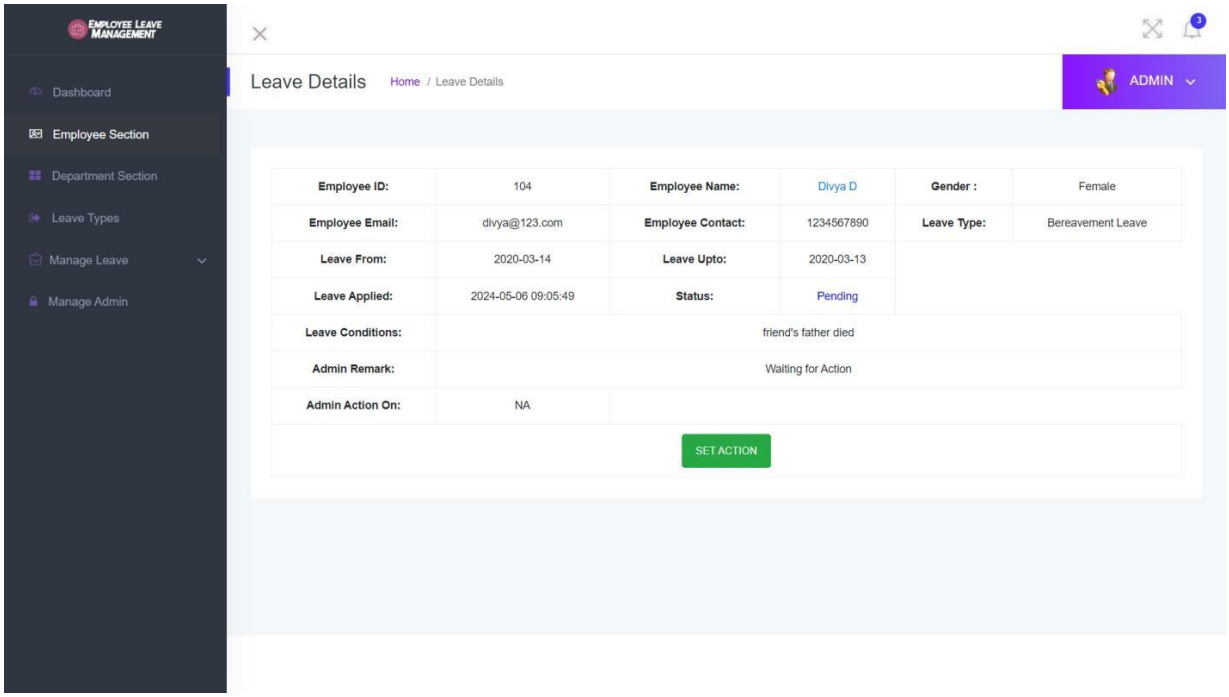

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Leave Section
Home / Manage Type
ADMIN
Add New Leave Type
Show 10 entries
Search:
Leave Type Description Created
1 Casual Leave Provided for urgent or unforeseen matters to the employees. 2020-11-01 17:37:56
2 Medical Leave Related to Health Problems of Employee 2020-11-06 18:46:09
3 Restricted Holiday Holiday that is optional 2020-11-06 18:46:38
4 Paternity Leave To take care of newborns 2021-03-03 16:16:31
5 Bereavement Leave Grieve their loss of losing loved ones 2021-03-03 16:17:48
6 Compensatory Leave For Overtime workers 2021-03-03 16:18:37
7 Maternity Leave Taking care of newborn ,recoveries 2021-03-03 16:20:17
8 Religious Holidays Based on employee's followed religion 2021-03-03 16:21:26
9 Adverse Weather Leave In terms of extreme weather conditions 2021-03-03 18:48:26
10 Voting Leave For official election day 2021-03-03 18:49:06
Showing 1 to 10 of 12 entries
Previous 1 2 Next

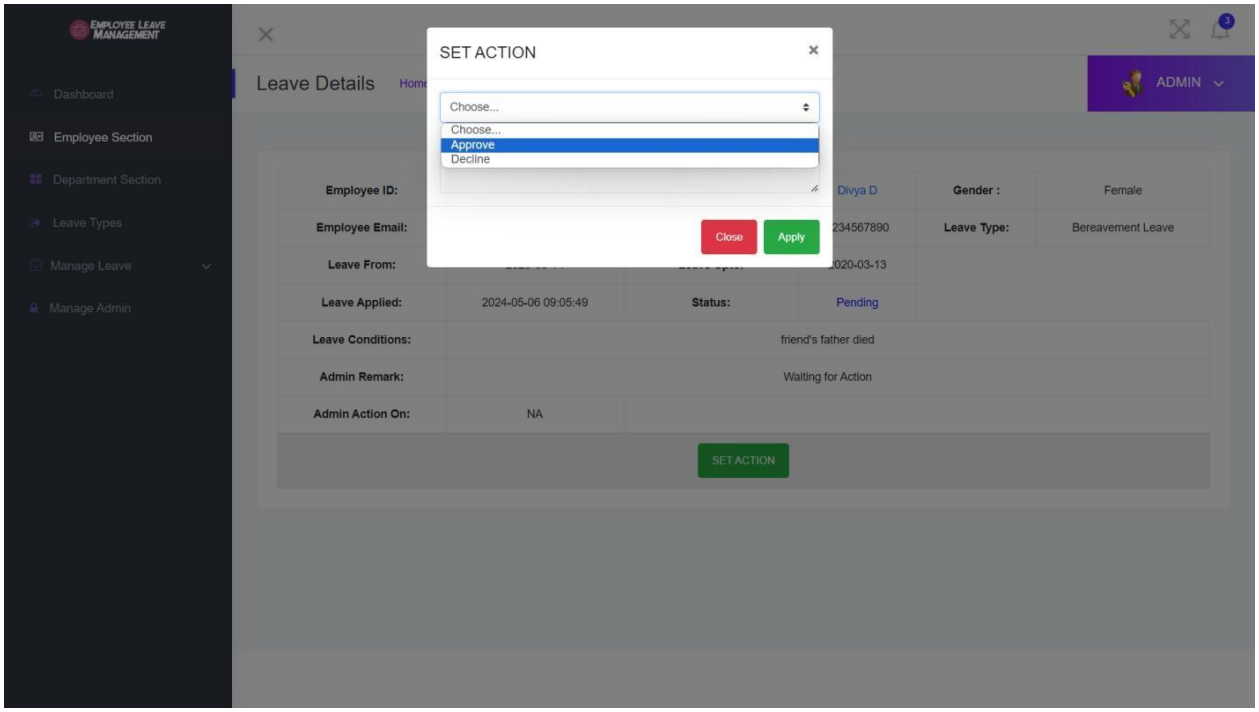

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Leave Section
Manage Type / Add
ADMIN
Please fill up the form in order to add new leave type
Leave Type
Short Description
ADD

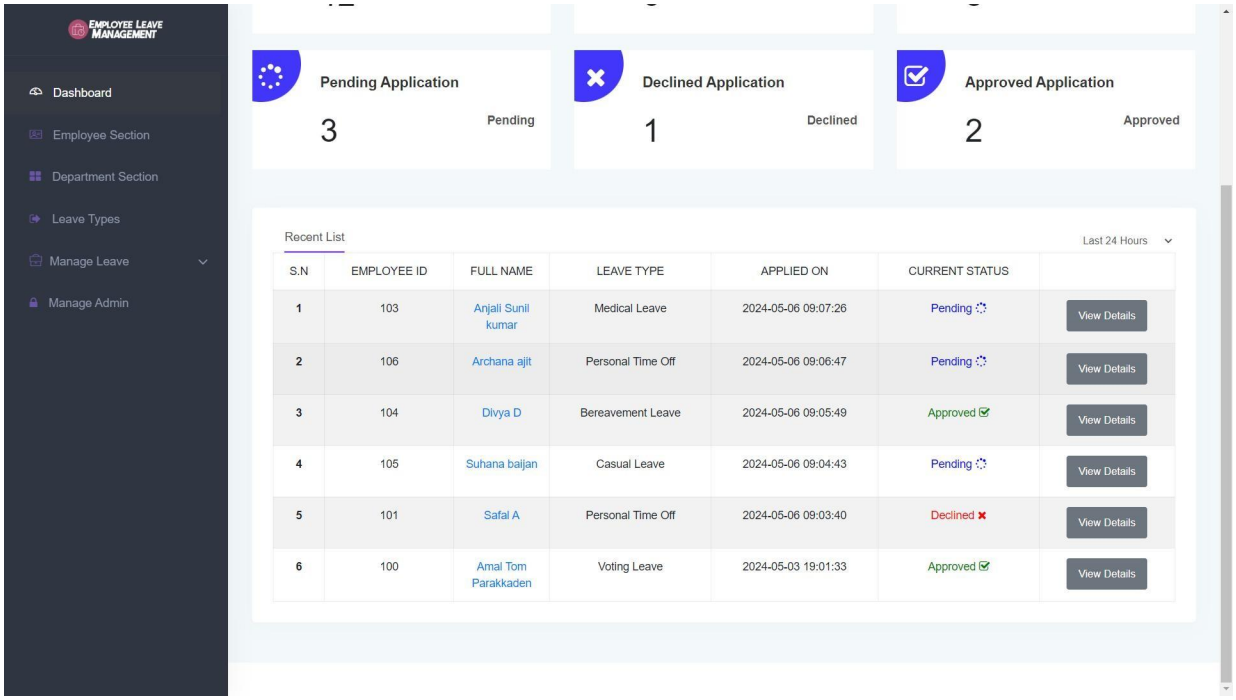

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Pending
Approved
Declined
Leave History
Manage Admin
Pending Leaves
Home / Pending List
ADMIN
S.N
EMPLOYEE ID
FULL NAME
LEAVE TYPE
APPLIED ON
CURRENT STATUS
1
103
Anjali Sunil kumar
Medical Leave
2024-05-06 09:07:26
Pending
View Details
2
106
Archana ajit
Personal Time Off
2024-05-06 09:06:47
Pending
View Details
3
104
Divya D
Bereavement Leave
2024-05-06 09:05:49
Pending
View Details
4
105
Suhana baljan
Casual Leave
2024-05-06 09:04:43
Pending
View Details

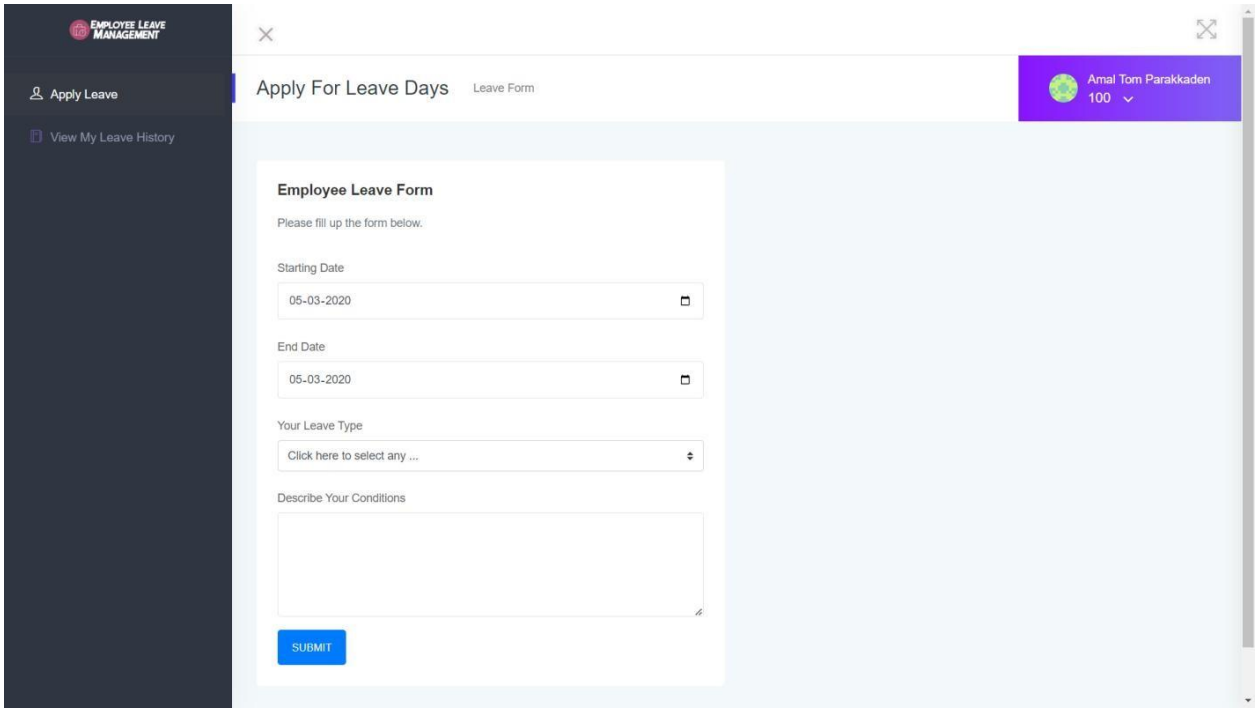

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Admin Section
Home / Manage Admin
ADMIN
Add New Administrator
Show 10 entries
Search:
Name Username Email ID Account Created On
1 Admin admin admin@123.com 2024-05-03 18:36:55
Showing 1 to 1 of 1 entries
Previous 1 Next

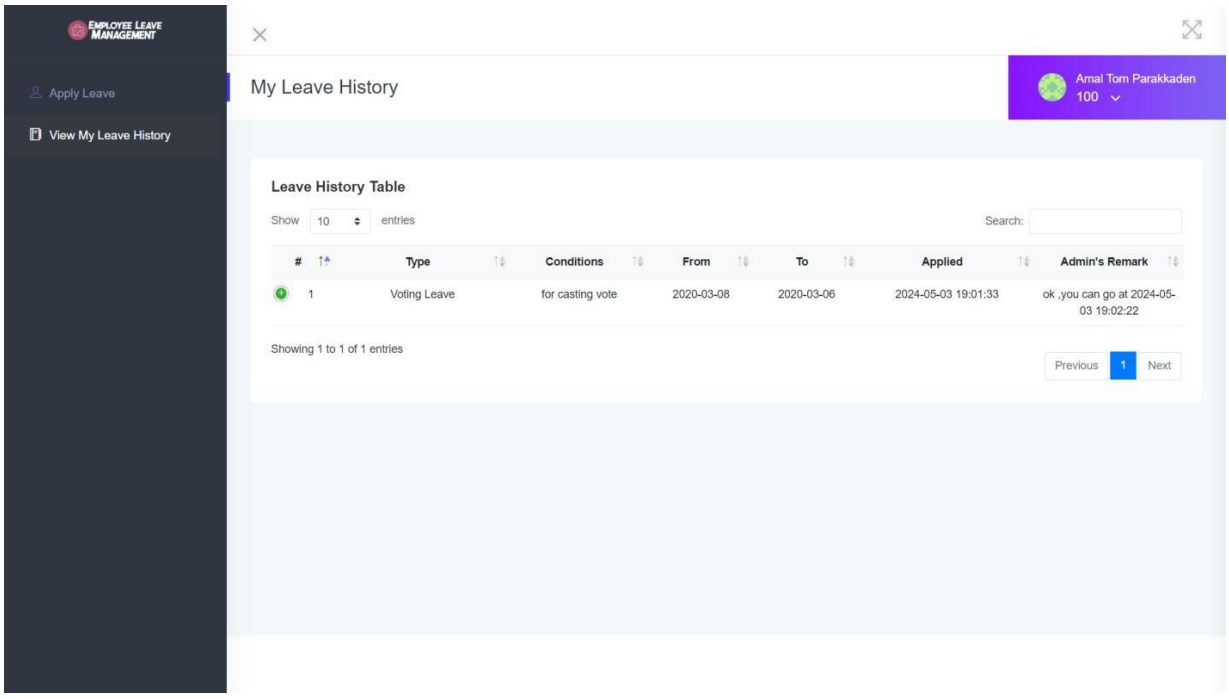

EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Add Admin Section
Manage Admin / Add
ADMIN
Please fill up the form in order to add new system administrator
Full Name
Email ID
Username
Setting Passwords
Password
Confirmation Password
PROCEED


EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Leave Details
Home / Leave Details
ADMIN
Employee ID: 104
Employee Name: Divya D
Gender : Female
Employee Email: divya@123.com
Employee Contact: 1234567890
Leave Type: Bereavement Leave
Leave From: 2020-03-14
Leave Upto: 2020-03-13
Leave Applied: 2024-05-06 09:05:49
Status: Pending
Leave Conditions: friend's father died
Admin Remark: Waiting for Action
Admin Action On: NA
SET ACTION


EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Leave Details
ADMIN
SET ACTION
Choose...
Choose...
Approve
Decline
Close
Apply
Employee ID:
Employee Email:
Leave From:
Leave Applied: 2024-05-06 09:05:49
Status: Pending
Leave Conditions: friend's father died
Admin Remark: Waiting for Action
Admin Action On: NA
Divya D
Gender : Female
Leave Type: Bereavement Leave
SET ACTION


EMPLOYEE LEAVE MANAGEMENT
Dashboard
Employee Section
Department Section
Leave Types
Manage Leave
Manage Admin
Pending Application
3
Pending
Declined Application
1
Declined
Approved Application
2
Approved
Recent List
Last 24 Hours
S.N
EMPLOYEE ID
FULL NAME
LEAVE TYPE
APPLIED ON
CURRENT STATUS
1 103 Anjali Sunil kumar Medical Leave 2024-05-06 09:07:26 Pending View Details
2 106 Archana ajit Personal Time Off 2024-05-06 09:06:47 Pending View Details
3 104 Divya D Bereavement Leave 2024-05-06 09:05:49 Approved View Details
4 105 Suhana baijan Casual Leave 2024-05-06 09:04:43 Pending View Details
5 101 Safal A Personal Time Off 2024-05-06 09:03:40 Declined View Details
6 100 Amal Tom Parakkaden Voting Leave 2024-05-03 19:01:33 Approved View Details


EMPLOYEE LEAVE MANAGEMENT
Apply Leave
View My Leave History
Apply For Leave Days
Leave Form
Amal Tom Parakkaden
100
Employee Leave Form
Please fill up the form below.
Starting Date
05-03-2020
End Date
05-03-2020
Your Leave Type
Click here to select any ...
Describe Your Conditions
SUBMIT


EMPLOYEE LEAVE MANAGEMENT
Apply Leave
View My Leave History
My Leave History
Amal Tom Parakkaden
100
Leave History Table
Show 10 entries
Search:
Type Conditions From To Applied Admin's Remark
1 Voting Leave for casting vote 2020-03-08 2020-03-06 2024-05-03 19:01:33 ok ,you can go at 2024-05-03 19:02:22
Showing 1 to 1 of 1 entries
Previous 1 Next

12. Conclusion

In conclusion, the implementation of an efficient employee leave management system is crucial for organizations striving to enhance productivity, employee satisfaction, and operational excellence. Through our project, we have developed a robust solution that addresses the challenges inherent in traditional leave management processes.

By leveraging advanced technology and automation, our system significantly reduces the administrative burden associated with managing leave requests. Employees benefit from a user-friendly interface that allows them to submit requests conveniently and track their leave balances in real-time. Managers are empowered with streamlined approval workflows and comprehensive reporting tools, enabling them to make informed decisions and maintain compliance with organizational policies and regulations.

Moreover, our system promotes transparency and communication by providing instant notifications and updates throughout the leave approval process. This fosters trust between employees and management, leading to a more positive work environment and higher levels of engagement.

Ultimately, our employee leave management project aims to revolutionize how organizations handle leave-related tasks, saving time, reducing errors, and enhancing overall efficiency. By investing in modern solutions like ours, companies can unlock the full potential of their workforce and achieve greater success in today's competitive business landscape