**EXPERIMENT 1**

**AIM:Execute DDL statements**

1)create a table student with fields roll number ,name,age,course,year.

Program code:

mysql> create database education ;

Query OK, 1 row affected (0.03 sec)

mysql> use education;

Database changed

mysql> create table student(roll\_no int primary key,name varchar(200),age int,course varchar(100),year int);

Query OK, 0 rows affected (0.10 sec)

mysql> desc student;

+---------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+--------------+------+-----+---------+-------+

| roll\_no | int | NO | PRI | NULL | |

| name | varchar(200) | YES | | NULL | |

| age | int | YES | | NULL | |

| course | varchar(100) | YES | | NULL | |

| year | int | YES | | NULL | |

+---------+--------------+------+-----+---------+-------+

5 rows in set (0.01 sec)

mysql> insert into student values(1,"vibil",22,"mca",2015);

Query OK, 1 row affected (0.01 sec)

mysql> insert into student values(2,"Rahul",21,"mba",2019);

Query OK, 1 row affected (0.01 sec)

mysql> insert into student values(3,"joel",21,"msw",2009);

Query OK, 1 row affected (0.01 sec)

mysql> insert into student values(4,"anandhan",24,"mca",2024);

Query OK, 1 row affected (0.02 sec)

mysql> insert into student values(5,"sivaji",21,"msw",2004);

Query OK, 1 row affected (0.02 sec)

mysql> select \* from student;

+---------+-------------+------+--------+------+

| roll\_no | name | age | course | year |

+---------+-------------+------+--------+------+

| 1 | vibil | 22 | mca | 2015 |

| 2 | Rahul | 21 | mba | 2019 |

| 3 | joel | 21 | msw | 2009 |

| 4 | anandhan | 24 | mca | 2024 |

| 5 | sivaji | 21 | msw | 2004 |

+---------+-------------+------+--------+------+

5 rows in set (0.00 sec)

2)alter table

Program code:

mysql> alter table student add mark int;

Query OK, 0 rows affected (0.08 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> select \* from student;

+---------+-------------+------+--------+------+------+

| roll\_no | name | age | course | year | mark |

+---------+-------------+------+--------+------+------+

| 1 | vibil | 22 | mca | 2015 | NULL |

| 2 | Rahul | 21 | mba | 2019 | NULL |

| 3 | joel | 21 | msw | 2009 | NULL |

| 4 | anandhan | 24 | mca | 2024 | NULL |

| 5 | sivaji | 21 | msw | 2004 | NULL |

+---------+-------------+------+--------+------+------+

5 rows in set (0.00 sec)

3)truncate table

Program code:

mysql> truncate student;

Query OK, 0 rows affected (0.16 sec)

mysql> select \* from student;

Empty set (0.00 sec)

mysql> desc student;

+---------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+--------------+------+-----+---------+-------+

| roll\_no | int | NO | PRI | NULL | |

| name | varchar(200) | YES | | NULL | |

| age | int | YES | | NULL | |

| course | varchar(100) | YES | | NULL | |

| year | int | YES | | NULL | |

| mark | int | YES | | NULL | |

+---------+--------------+------+-----+---------+-------+

6 rows in set (0.00 sec)

4)drop table

mysql> drop table student;

Query OK, 0 rows affected (0.07 sec)

mysql> select \* from student;

ERROR 1146 (42S02): Table 'education.student' doesn't exist

**EXPERIMENT 2**

AIM:Execute DML STATEMENTS (select,insert,update,delete)

a)create table employee(emp\_id,emp\_name,dept\_id,salary) create another table department(Dep\_id,dep\_name,dept\_head )

program code:

mysql> create table employee(emp\_id int primary key,emp\_name varchar(20),dept\_id int ,salary int);

Query OK, 0 rows affected (0.10 sec)

mysql> create table department(dept\_id int primary key,dept\_name varchar(30),dept\_head varchar(20));

Query OK, 0 rows affected (0.11 sec)

mysql> desc employee;

+----------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+-------------+------+-----+---------+-------+

| emp\_id | int | NO | PRI | NULL | |

| emp\_name | varchar(20) | YES | | NULL | |

| dept\_id | int | YES | | NULL | |

| salary | int | YES | | NULL | |

+----------+-------------+------+-----+---------+-------+

4 rows in set (0.01 sec)

mysql> desc department;

+-----------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-----------+-------------+------+-----+---------+-------+

| dept\_id | int | NO | PRI | NULL | |

| dept\_name | varchar(30) | YES | | NULL | |

| dept\_head | varchar(20) | YES | | NULL | |

+-----------+-------------+------+-----+---------+-------+

3 rows in set (0.00 sec)

b)insert minimum 4 rows

mysql> insert into employee values(12,’vibil’,102,25000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into employee values(13,'rahul',103,5000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into employee values(14,'anandhu',104,50000);

Query OK, 1 row affected (0.01 sec)

mysql> insert into employee values(15,'joel',105,30000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into department values(102,'computer','prof.binil');

Query OK, 1 row affected (0.02 sec)

mysql> insert into department values(103,'biology','prof.sinoj');

Query OK, 1 row affected (0.01 sec)

mysql> insert into department values(104,'microbiology','prof.Ebin');

Query OK, 1 row affected (0.02 sec)

mysql> insert into department values(105,'maths','prof.babu');

Query OK, 1 row affected (0.02 sec)

c)set primary key and foreign key consraints

program code:

mysql> alter table employee add constraint fk\_dept\_id foreign key(dept\_id)references department(dept\_id) ;

Query OK, 4 rows affected (0.30 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> desc employee;

+----------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+-------------+------+-----+---------+-------+

| emp\_id | int | NO | PRI | NULL | |

| emp\_name | varchar(20) | YES | | NULL | |

| dept\_id | int | YES | MUL | NULL | |

| salary | int | YES | | NULL | |

+----------+-------------+------+-----+---------+-------+

4 rows in set (0.00 sec)

d)display the record

program code:

mysql> select\*from employee;

+--------+----------+---------+--------+

| emp\_id | emp\_name | dept\_id | salary |

+--------+----------+---------+--------+

| 12 | vibil | 102 | 25000 |

| 13 | rahul | 103 | 5000 |

| 14 | anandhan | 104 | 50000 |

| 15 | joel | 105 | 30000 |

+--------+----------+---------+--------+

4 rows in set (0.00 sec)

mysql> select\*from department;

+---------+--------------+--------------+

| dept\_id | dept\_name | dept\_head |

+---------+--------------+--------------+

| 102 | computer | prof.binil |

| 103 | biology | prof.sinoj |

| 104 | microbiology | prof.Ebin |

| 105 | maths | prof.babu |

+---------+--------------+--------------+

4 rows in set (0.01 sec)

e)update a record

e)mysql> update department set dept\_head='denna' where dept\_name='maths';

Query OK, 1 row affected (0.02 sec)

Rows matched: 1 Changed: 1 Warnings: 0

f)delete a record

program code:

mysql> delete from employee where dept\_id=102;

Query OK, 1 row affected (0.02 sec)

mysql>select\*from employee;

+--------+----------+---------+--------+

| emp\_id | emp\_name | dept\_id | salary |

+--------+----------+---------+--------+

| 13 | rahul | 103 | 5000 |

| 14 | anandhan | 104 | 50000 |

| 15 | joel | 105 | 30000 |

+--------+----------+---------+--------+

3 rows in set (0.00 sec)

**EXPERIMENT 3**

AIM:create a table and execute DCL statements (grant and revoke)

program code:

mysql> create user 'vibil'@'localhost' identified by 'a123';

Query OK, 0 rows affected (0.02 sec)

mysql> grant all privileges on employee to 'vibil'@'localhost';

Query OK, 0 rows affected (0.01 sec)

mysql> show grants for 'vibil'@'localhost';

+------------------------------------------------------------------+

| Grants for vibil@localhost |

+------------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `vibil`@`localhost` |

| GRANT ALL PRIVILEGES ON `work`.`employee` TO `vibil`@`localhost` |

+------------------------------------------------------------------+

2 rows in set (0.00 sec)

mysql> quit;

Bye

student@student-Veriton-Series:~$ mysql -u 'vibil' -p;

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 13

Server version: 8.0.41-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use work;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> select\*from employee;

+--------+----------+---------+--------+

| emp\_id | emp\_name | dept\_id | salary |

+--------+----------+---------+--------+

| 13 | rahul | 103 | 5000 |

| 14 | anandhan | 104 | 50000 |

| 15 | joel | 105 | 30000 |

+--------+----------+---------+--------+

3 rows in set (0.00 sec)

mysql> quit;

Bye

student@student-Veriton-Series:~$ sudo mysql;

[sudo] password for student:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 14

Server version: 8.0.41-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use work;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> revoke all privileges on employee from 'vibil'@'localhost';

Query OK, 0 rows affected (0.02 sec)

mysql> quit;

Bye

tudent@student-Veriton-Series:~$ mysql -u ‘vibil’ -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 15

Server version: 8.0.41-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select\*from employee;

ERROR 1046 (3D000): No database selected

mysql> use work;

ERROR 1044 (42000): Access denied for user ‘vibil’@'localhost' to database 'work'

mysql>

**EXPERIMENT 4**

AIM:a)create a table and execute tcl statements(commit,rollback,savepoint)

mysql> create table student1 (rollno int,name varchar(30),course varchar(10));

Query OK, 0 rows affected (0.12 sec)

Mysql> insert into student1 values(2,”vishnu”,”bca”);

Query OK, 1 row affected (0.00 sec)

Mysql> insert into student1 values(1,”priya”,”mca”);

Query OK, 1 row affected (0.00 sec)

Mysql> start transaction;

Query OK, 0 rows affected (0.02 sec)

Mysql> savepoint a;

Query OK, 0 rows affected (0.00 sec)

Mysql> select\*from student1;

+--------+--------+--------+

| rollno | name | course |

+--------+--------+--------+

| 2 | vishnu | bca |

| 1 | priya | mca |

+--------+--------+--------+

2 rows in set (0.00 sec)

Mysql> insert into student1 values(6,”vinu”, “bsc “);

Query OK, 1 row affected (0.01 sec)

Mysql> insert into student1 values(7,” vimala”,”bsc”);

Query OK, 1 row affected (0.00 sec)

Mysql> select\*from student1;

+--------+--------+--------+

| rollno | name | course |

+--------+--------+--------+

| 2 | vishnu | bca |

| 1 | priya | mca |

| 6 | vinu | bsc |

| 7 | vimala | bsc |

+--------+--------+--------+

4 rows in set (0.00 sec)

Mysql>rollback to a;

Query OK, 0 rows affected (0.01 sec)

Mysql> select\*from student1;

+--------+--------+--------+

| rollno | name | course |

+--------+--------+--------+

| 2 | vishnu | bca |

| 1 | priya | mca |

+--------+--------+--------+

2 rows in set (0.00 sec)

Mysql>commit;

Query OK, 0 rows affected (0.00 sec)

Mysql> insert into student1 values(10,”nivi”, “btech”);

Query OK, 1 row affected (0.02 sec)

Mysql> select\*from student1;

+--------+--------+--------+

| rollno | name | course |

+--------+--------+--------+

| 2 | vishnu | bca |

| 1 | priya | mca |

| 10 | nivi | btech |

+--------+--------+--------+

3 rows in set (0.00 sec)

Mysql>rollback;

Query OK, 0 rows affected (0.00 sec)

Mysql> select\*from student1;

+--------+--------+--------+

| rollno | name | course |

+--------+--------+--------+

| 2 | vishnu | bca |

| 1 | priya | mca |

| 10 | nivi | btech |

+--------+--------+--------+

3 rows in set (0.00 sec)

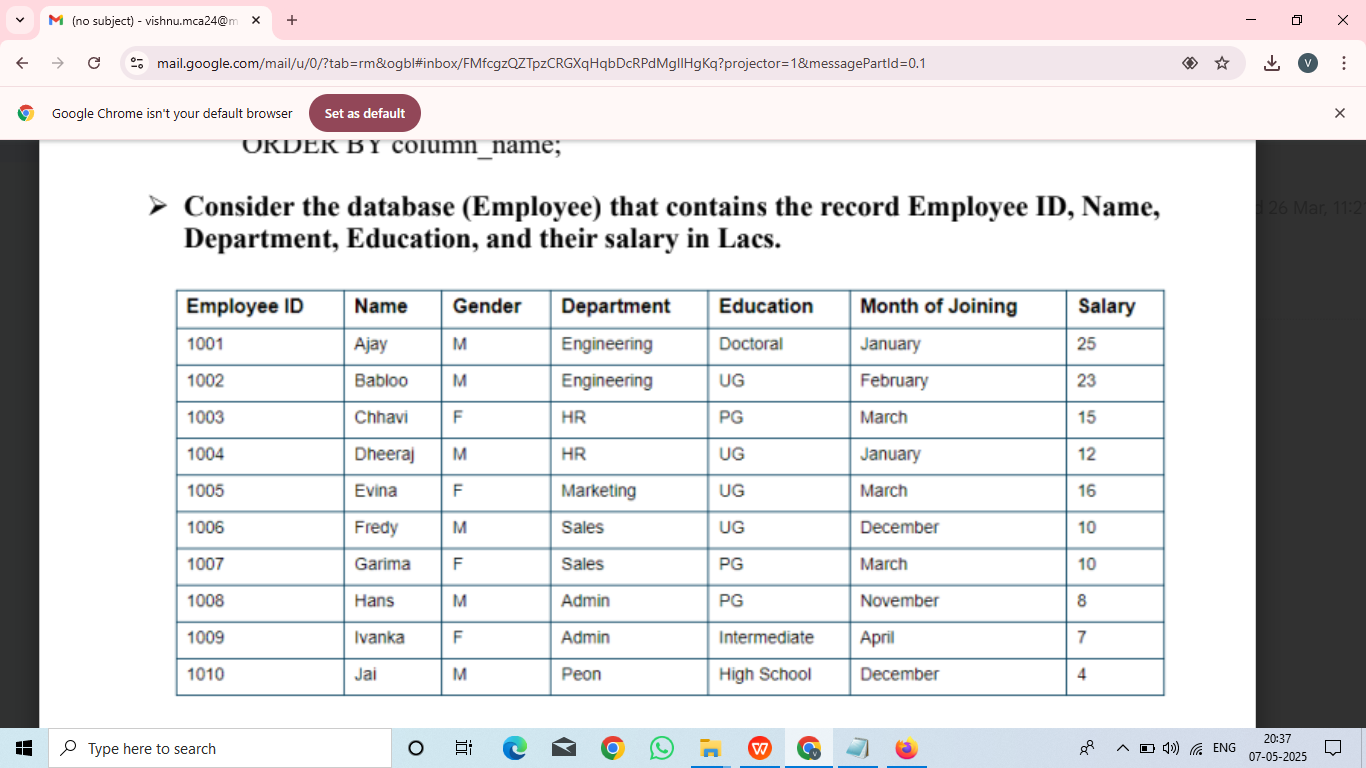
**EXPERIMENT 5**

AIM:Accessing database (SELECT, Filtering using WHERE, HAVING, GROUP BY,

ORDER BY Clauses)

Consider the database (Employee) that contains the record Employee ID, Name,

Department, Education, and their salary in Lacs.



1. (i): Calculate the sum of salaries of each department.

Program code:

mysql> select department,sum(salary) from employee group by department;

+-------------+-------------+

| department | sum(salary) |

+-------------+-------------+

| engineering | 48 |

| hr | 27 |

| marketing | 16 |

| sales | 20 |

| admin | 15 |

| peon | 4 |

+-------------+-------------+

6 rows in set (0.00 sec)

(ii): Find the departments in which the SUM of the salaries is greater than or equal

to 20 lacs

Program code

mysql> select department,sum(salary) from employee group by department having sum(salary)>=20;

+-------------+-------------+

| department | sum(salary) |

+-------------+-------------+

| engineering | 48 |

| hr | 27 |f

| sales | 20 |

+-------------+-------------+

3 rows in set (0.00 sec)

(iii) Display distinct department of Employee.

mysql> select distinct department from employee;

+-------------+

| department |

+-------------+

| engineering |

| hr |

| marketing |

| sales |

| admin |

| peon |

+-------------+

6 rows in set (0.00 sec)

(iv) Display total number of Salary of Employees.

Program code:

mysql> select count(salary) from employee;

+---------------+

| count(salary) |

+---------------+

| 10 |

+---------------+

1 row in set (0.00 sec)

(v) Display details of Employees from Employee tables in which Department of the

employee is Engineering and Education is Doctorial.

Program code:

mysql> select \* from employee where department='engineering' or education='doctorial';

+--------+--------+--------+-------------+-----------+------------+--------+

| emp\_id | name | gender | department | education | join\_month | salary |

+--------+--------+--------+-------------+-----------+------------+--------+

| 1001 | ajay | m | engineering | doctoral | january | 25 |

| 1002 | babloo | m | engineering | ug | february | 23 |

+--------+--------+--------+-------------+-----------+------------+--------+

2 rows in set (0.00 sec)

(vi) Display details of Employees from Employee tables in which Department of

the employee is Engineering or Education is Doctorial.

Program code:

mysql> select \* from employee where department='engineering' and education='doctoral';

+--------+------+--------+-------------+-----------+------------+--------+

| emp\_id | name | gender | department | education | join\_month | salary |

+--------+------+--------+-------------+-----------+------------+--------+

| 1001 | ajay | m | engineering | doctoral | january | 25 |

+--------+------+--------+-------------+-----------+------------+--------+

1 row in set (0.00 sec)

(vii) Rename the columns "Name" and "Education" to "First\_Name" and

"Qualification", respectively.

Program code:

mysql> select name as first\_name ,education as qualification from employee;

+------------+---------------+

| first\_name | qualification |

+------------+---------------+

| ajay | doctoral |

| babloo | ug |

| chhavi | pg |

| dheeraj | ug |

| evina | ug |

| fredy | ug |

| garima | pg |

| hans | pg |

| ivanka | intermediate |

| jai | high school |

+------------+---------------+

10 rows in set (0.00 sec)

(viii)List records of Employees whose names start with “A”.

Program code:

mysql> select \* from employee where name like 'A%';

+--------+------+--------+-------------+-----------+------------+--------+

| emp\_id | name | gender | department | education | join\_month | salary |

+--------+------+--------+-------------+-----------+------------+--------+

| 1001 | ajay | m | engineering | doctoral | january | 25 |

+--------+------+--------+-------------+-----------+------------+--------+

1 row in set (0.00 sec)

2) Find the department in which SUM salary is greater than or equal to 20 lacs, but

the education of employees is not UG.

Program code:

mysql> select department,sum(salary) from employee where education !='ug' group by department having sum(salary)>=20;

+-------------+-------------+

| department | sum(salary) |

+-------------+-------------+

| engineering | 25 |

+-------------+-------------+

1 row in set (0.00 sec)

3) Find the departments in which the SUM of the salaries is greater than or equal to

15 lacs and arrange the Salary in descending order.

Program code:

mysql> select department ,sum(salary) from employee group by department having s

um(salary)>=15 order by sum(salary) desc;

+-------------+-------------+

| department | sum(salary) |

+-------------+-------------+

| engineering | 48 |

| hr | 27 |

| sales | 20 |

| marketing | 16 |

| admin | 15 |

+-------------+-------------+

5 rows in set (0.00 sec)

**EXPERIMENT 6**

AIM:Creating views

(i) From the following table, create a view for those sales people who belong to

the city of New York.

Table: salesman

+------------+------------+---------+-----------+

| salesmanid | name | city | comission |

+------------+------------+---------+-----------+

| 5001 | james hoog | newyork | 0.15 |

| 5002 | nail knife | paris | 0.13 |

| 5003 | lauson hen | sanjose | 0.13 |

| 5005 | ml lyon | paris | 0.14 |

| 5006 | pit alex | london | 0.11 |

| 5007 | paul adam | rome | 0.13 |

+------------+------------+---------+-----------+

Program code:

mysql> create table salesman (salesman\_id int primary key,name varchar(200),city varchar(200),comission decimal(3,3));

Query OK, 0 rows affected (0.10 sec)

mysql> insert into salesman values(5003,'lauson hen','san jose',0.12);

Query OK, 1 row affected (0.02 sec)

mysql> insert into salesman values(5007,'paul adam','rome',0.13);

Query OK, 1 row affected (0.02 sec)

mysql> insert into salesman values(5006,'mc lyon','paris',0.14);

Query OK, 1 row affected (0.02 sec)

mysql> insert into salesman values(5005,'pit alex','london',0.11);

Query OK, 1 row affected (0.01 sec)

mysql> insert into salesman values(5002,'nail knite','paris',0.13);

Query OK, 1 row affected (0.01 sec)

mysql> insert into salesman values(5001,'james hoog','new york',0.15);

Query OK, 1 row affected (0.02 sec)

mysql> select \* from salesman;

+-------------+------------+----------+-----------+

| salesman\_id | name | city | comission |

+-------------+------------+----------+-----------+

| 5001 | james hoog | new york | 0.150 |

| 5002 | nail knite | paris | 0.130 |

| 5003 | lauson hen | san jose | 0.120 |

| 5005 | pit alex | london | 0.110 |

| 5006 | mc lyon | paris | 0.140 |

| 5007 | paul adam | rome | 0.130 |

+-------------+------------+----------+-----------+

6 rows in set (0.00 sec)

i)mysql> create view newyorkstaff as select \* from salesman where city='new york';

Query OK, 0 rows affected (0.04 sec)

mysql> select \* from newyorkstaff;

+-------------+------------+----------+-----------+

| salesman\_id | name | city | comission |

+-------------+------------+----------+-----------+

| 5001 | james hoog | new york | 0.150 |

+-------------+------------+----------+-----------+

1 row in set (0.00 sec)

ii) From the following table, create a view that counts the number of customers

in each grade .

Table: customer

+-------------+----------------+------------+-------+------------+

| customer\_id | cust\_name | city | grade | salesmanid |

+-------------+----------------+------------+-------+------------+

| 3001 | Brad Guzan | London | 100 | 5005 |

| 3002 | Nick Rimando | New York | 100 | 5001 |

| 3003 | Jozy Altidor | Moscow | 200 | 5007 |

| 3004 | Fabian Johnson | Paris | 300 | 5006 |

| 3005 | Graham Zusi | California | 200 | 5002 |

| 3007 | Brad Davis | New York | 200 | 5001 |

| 3008 | Julian Green | London | 300 | 5002 |

| 3009 | Geoff Cameron | Berlin | 100 | 5003 |

+-------------+----------------+------------+-------

Program code:

ii)mysql> CREATE TABLE customer (

-> customer\_id INT PRIMARY KEY,

-> cust\_name VARCHAR(100) NOT NULL,

-> city VARCHAR(100) NOT NULL,

-> grade INT,

-> salesman\_id INT,

-> FOREIGN KEY (salesman\_id) REFERENCES salesman(salesman\_id)

-> );

Query OK, 0 rows affected (0.12 sec)

mysql> desc customer;

+-------------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------------+--------------+------+-----+---------+-------+

| customer\_id | int | NO | PRI | NULL | |

| cust\_name | varchar(100) | NO | | NULL | |

| city | varchar(100) | NO | | NULL | |

| grade | int | YES | | NULL | |

| salesman\_id | int | YES | MUL | NULL | |

+-------------+--------------+------+-----+---------+-------+

5 rows in set (0.01 sec)

mysql> INSERT INTO customer (customer\_id, cust\_name, city, grade, salesman\_id)

-> VALUES

-> (3002, 'Nick Rimando', 'New York', 100, 5001),

-> (3007, 'Brad Davis', 'New York', 200, 5001),

-> (3005, 'Graham Zusi', 'California', 200, 5001),

-> (3008, 'Julian Green', 'London', 300, 5002),

-> (3004, 'Fabian Johnson', 'Paris', 300, 5006),

-> (3009, 'Geoff Cameron', 'Berlin', 100, 5003),

-> (3003, 'Jozy Altidore', 'Moscow', 200, 5007),

-> (3001, 'Brad Guzan', 'London', 100, 5005);

Query OK, 8 rows affected (0.01 sec)

Records: 8 Duplicates: 0 Warnings: 0

mysql> select \* from customer;

+-------------+----------------+------------+-------+-------------+

| customer\_id | cust\_name | city | grade | salesman\_id |

+-------------+----------------+------------+-------+-------------+

| 3001 | Brad Guzan | London | 100 | 5005 |

| 3002 | Nick Rimando | New York | 100 | 5001 |

| 3003 | Jozy Altidore | Moscow | 200 | 5007 |

| 3004 | Fabian Johnson | Paris | 300 | 5006 |

| 3005 | Graham Zusi | California | 200 | 5001 |

| 3007 | Brad Davis | New York | 200 | 5001 |

| 3008 | Julian Green | London | 300 | 5002 |

| 3009 | Geoff Cameron | Berlin | 100 | 5003 |

+-------------+----------------+------------+-------+-------------+

8 rows in set (0.00 sec)

mysql> create view gradecount(grade,number) as select grade,count(\*) from customer group by grade;

Query OK, 0 rows affected (0.03 sec)

mysql> select \* from gradecount;

+-------+--------+

| grade | number |

+-------+--------+

| 100 | 3 |

| 200 | 3 |

| 300 | 2 |

+-------+--------+

3 rows in set (0.00 sec)

iii)

From the following table, create a view to count the number of unique

customers, compute the average and the total purchase amount of customer

orders by each date.

+--------+-----------+------------+-------------+------------+

| ord\_no | purch\_amt | ord\_date | customer\_id | salesmanid |

+--------+-----------+------------+-------------+------------+

| 70001 | 150.50 | 2012-10-05 | 3005 | 5002 |

| 70002 | 65.26 | 2012-10-05 | 3002 | 5001 |

| 70003 | 2480.40 | 2012-10-10 | 3009 | 5003 |

| 70004 | 110.50 | 2012-08-17 | 3009 | 5003 |

| 70005 | 2400.60 | 2012-07-27 | 3007 | 5001 |

| 70007 | 948.50 | 2012-09-10 | 3005 | 5002 |

| 70008 | 5760.00 | 2012-09-10 | 3002 | 5001 |

| 70009 | 270.65 | 2012-09-10 | 3001 | 5005 |

| 70010 | 1983.43 | 2012-10-10 | 3004 | 5006 |

| 70011 | 75.29 | 2012-08-17 | 3003 | 5007 |

| 70012 | 250.45 | 2012-06-27 | 3008 | 5002 |

| 70013 | 3045.60 | 2012-04-25 | 3002 | 5001 |

+--------+-----------+------------+-------------+------------+

Program code:

mysql> CREATE TABLE orders (

-> ord\_no INT PRIMARY KEY,

-> purch\_amt DECIMAL(10, 2),

-> ord\_date DATE,

-> customer\_id INT,

-> salesman\_id INT,

-> FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id),

-> FOREIGN KEY (salesman\_id) REFERENCES salesman(salesman\_id)

-> );

Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO orders (ord\_no, purch\_amt, ord\_date, customer\_id, salesman\_id)

-> VALUES

-> (70001, 150.5, '2012-10-05', 3005, 5002),

-> (70009, 270.65, '2012-09-10', 3001, 5005),

-> (70002, 65.26, '2012-10-05', 3002, 5001),

-> (70004, 110.5, '2012-08-17', 3009, 5003),

-> (70007, 948.5, '2012-09-10', 3005, 5002),

-> (70005, 2400.6, '2012-07-27', 3007, 5001),

-> (70008, 5760, '2012-09-10', 3002, 5001),

-> (70010, 1983.43, '2012-10-10', 3004, 5006),

-> (70003, 2480.4, '2012-10-10', 3009, 5003),

-> (70012, 250.45, '2012-06-27', 3008, 5002),

-> (70011, 75.29, '2012-08-17', 3003, 5007),

-> (70013, 3045.6, '2012-04-25', 3002, 5001);

Query OK, 12 rows affected (0.01 sec)

Records: 12 Duplicates: 0 Warnings: 0

mysql> select \* from orders

-> ;

+--------+-----------+------------+-------------+-------------+

| ord\_no | purch\_amt | ord\_date | customer\_id | salesman\_id |

+--------+-----------+------------+-------------+-------------+

| 70001 | 150.50 | 2012-10-05 | 3005 | 5002 |

| 70002 | 65.26 | 2012-10-05 | 3002 | 5001 |

| 70003 | 2480.40 | 2012-10-10 | 3009 | 5003 |

| 70004 | 110.50 | 2012-08-17 | 3009 | 5003 |

| 70005 | 2400.60 | 2012-07-27 | 3007 | 5001 |

| 70007 | 948.50 | 2012-09-10 | 3005 | 5002 |

| 70008 | 5760.00 | 2012-09-10 | 3002 | 5001 |

| 70009 | 270.65 | 2012-09-10 | 3001 | 5005 |

| 70010 | 1983.43 | 2012-10-10 | 3004 | 5006 |

| 70011 | 75.29 | 2012-08-17 | 3003 | 5007 |

| 70012 | 250.45 | 2012-06-27 | 3008 | 5002 |

| 70013 | 3045.60 | 2012-04-25 | 3002 | 5001 |

+--------+-----------+------------+-------------+-------------+

12 rows in set (0.00 sec)

mysql> create view totalforday as select ord\_date,count(distinct customer\_id),avg(purch\_amt),sum(purch\_amt) fromQuery OK, 0 rows affected (0.03 sec)

mysql> select \* from totalforday;

+------------+-----------------------------+----------------+----------------+

| ord\_date | count(distinct customer\_id) | avg(purch\_amt) | sum(purch\_amt) |

+------------+-----------------------------+----------------+----------------+

| 2012-04-25 | 1 | 3045.600000 | 3045.60 |

| 2012-06-27 | 1 | 250.450000 | 250.45 |

| 2012-07-27 | 1 | 2400.600000 | 2400.60 |

| 2012-08-17 | 2 | 92.895000 | 185.79 |

| 2012-09-10 | 3 | 2326.383333 | 6979.15 |

| 2012-10-05 | 2 | 107.880000 | 215.76 |

| 2012-10-10 | 2 | 2231.915000 | 4463.83 |

+------------+-----------------------------+----------------+----------------+

7 rows in set (0.00 sec)

(iv) From the order table, create a view to find the salespersons who issued

orders on either August 17th, 2012 or October 10th, 2012. Return

salesperson ID, order number and customer ID.

Program code:

mysql> create view sorder as select salesman\_id,ord\_no,customer\_id from orders where ord\_date in('2012-08-17','2012-10-10');

Query OK, 0 rows affected (0.02 sec)

mysql> select \* from sorder;

+-------------+--------+-------------+

| salesman\_id | ord\_no | customer\_id |

+-------------+--------+-------------+

| 5003 | 70003 | 3009 |

| 5003 | 70004 | 3009 |

| 5006 | 70010 | 3004 |

| 5007 | 70011 | 3003 |

+-------------+--------+-------------+

4 rows in set (0.01 sec)

(v) From the salesman and order tables, create a view to find the salesperson

who handles a customer who makes the highest order of the day. Return order date, salesperson ID, name.

Program code:

mysql> create view elitsalesman as select b.ord\_date,a.salesman\_id,a.name from salesman a,orders b where a.salesman\_id=b.salesman\_id and b.purch\_amt=(select max(purch\_amt) from orders c where c.ord\_date=b.ord\_date);

Query OK, 0 rows affected (0.03 sec)

mysql> select \* from elitsalesman;

+------------+-------------+------------+

| ord\_date | salesman\_id | name |

+------------+-------------+------------+

| 2012-07-27 | 5001 | james hoog |

| 2012-09-10 | 5001 | james hoog |

| 2012-04-25 | 5001 | james hoog |

| 2012-10-05 | 5002 | nail knite |

| 2012-06-27 | 5002 | nail knite |

| 2012-10-10 | 5003 | lauson hen |

| 2012-08-17 | 5003 | lauson hen |

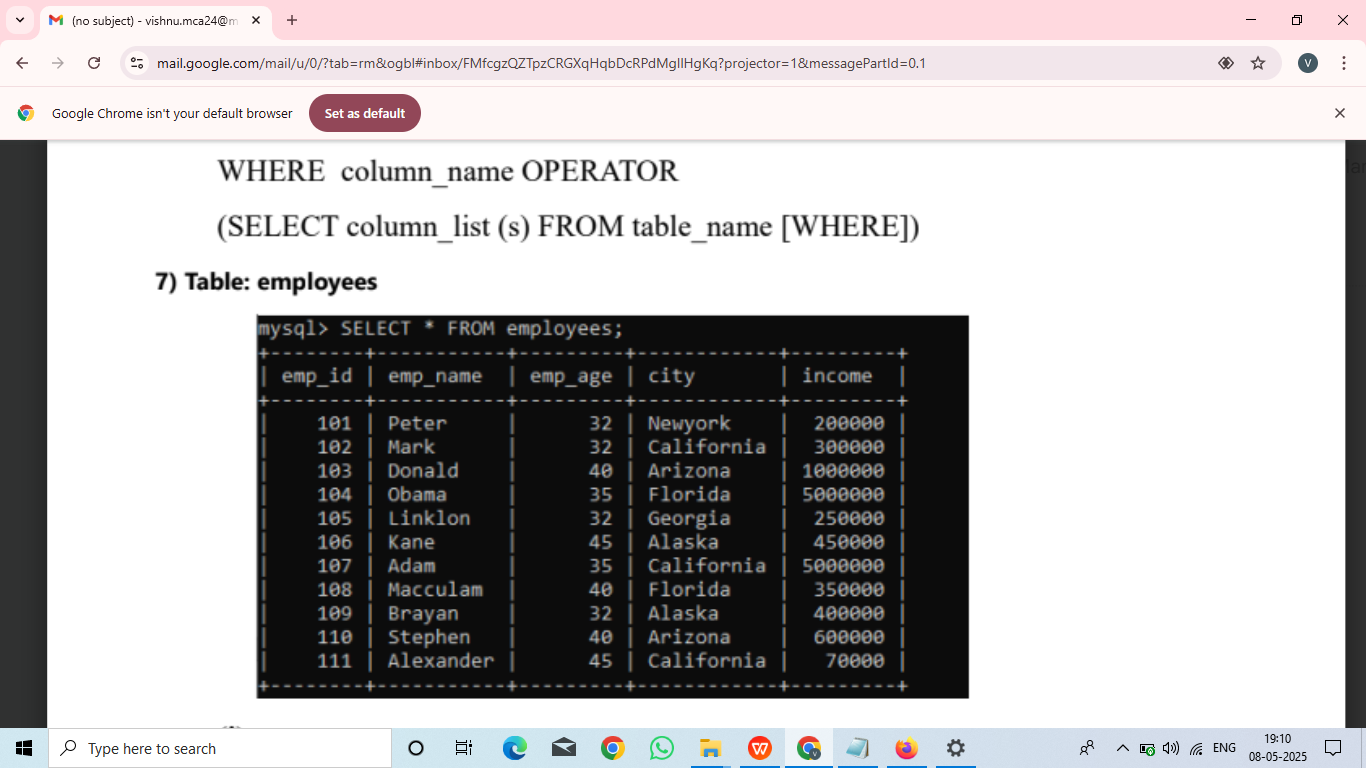
+------------+-------------+------------+

7 rows in set (0.00 sec)

**EXPERIMENT 7**

AIM:

Table: employees



mysql> create table employees(emp\_id int primary key,emp\_name varchar(100),emp\_age int,city varchar(100),income int);

Query OK, 0 rows affected (0.10 sec)

mysql> desc employees;

+----------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+--------------+------+-----+---------+-------+

| emp\_id | int | NO | PRI | NULL | |

| emp\_name | varchar(100) | YES | | NULL | |

| emp\_age | int | YES | | NULL | |

| city | varchar(100) | YES | | NULL | |

| income | int | YES | | NULL | |

+----------+--------------+------+-----+---------+-------+

5 rows in set (0.01 sec)

mysql> insert into employees values(101,'peter',32,'newyork',200000);

Query OK, 1 row affected (0.03 sec)

mysql> insert into employees values(102,'mark',32,'california',300000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into employees values(103,'Donald',40,'arizona',1000000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into employees values(104,'obama',35,'florida',5000000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into employees values(105,'linklon',32,'georgia',250000);

Query OK, 1 row affected (0.01 sec)

mysql> insert into employees values(106,'kane',45,'alaska',450000);

Query OK, 1 row affected (0.01 sec)

mysql> insert into employees values(107,'adam',35,'california',5000000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into employees values(108,'macculam',40,'florida',350000);

Query OK, 1 row affected (0.01 sec)

mysql> insert into employees values(109,'brayan',32,'alaska',400000);

Query OK, 1 row affected (0.02 sec)

mysql> insert into employees values(110,'stephen',40,'arisona',600000);

Query OK, 1 row affected (0.01 sec)

mysql> insert into employees values(111,'alexander',45,'california',70000);

Query OK, 1 row affected (0.01 sec)

mysql> select \* from employees;

+--------+-----------+---------+------------+---------+

| emp\_id | emp\_name | emp\_age | city | income |

+--------+-----------+---------+------------+---------+

| 101 | peter | 32 | newyork | 200000 |

| 102 | mark | 32 | california | 300000 |

| 103 | Donald | 40 | arizona | 1000000 |

| 104 | obama | 35 | florida | 5000000 |

| 105 | linklon | 32 | georgia | 250000 |

| 106 | kane | 45 | alaska | 450000 |

| 107 | adam | 35 | california | 5000000 |

| 108 | macculam | 40 | florida | 350000 |

| 109 | brayan | 32 | alaska | 400000 |

| 110 | stephen | 40 | arisona | 600000 |

| 111 | alexander | 45 | california | 70000 |

+--------+-----------+---------+------------+---------+

11 rows in set (0.00 sec)

1. Find employee detail whose id matches in a subquery:

Program code:

mysql> select emp\_name,city,income from employees where emp\_id in(select emp\_id

from employees);

+-----------+------------+---------+

| emp\_name | city | income |

+-----------+------------+---------+

| peter | newyork | 200000 |

| mark | california | 300000 |

| Donald | arizona | 1000000 |

| obama | florida | 5000000 |

| linklon | georgia | 250000 |

| kane | alaska | 450000 |

| adam | california | 5000000 |

| macculam | florida | 350000 |

| brayan | alaska | 400000 |

| stephen | arisona | 600000 |

| alexander | california | 70000 |

+-----------+------------+---------+

11 rows in set (0.00 sec)

(ii) Find employee detail whose income is more than 350000 with the help of subquery:

mysql> select \* from employees where emp\_id in (select emp\_id from employees where income>350000);

+--------+----------+---------+------------+---------+

| emp\_id | emp\_name | emp\_age | city | income |

+--------+----------+---------+------------+---------+

| 103 | Donald | 40 | arizona | 1000000 |

| 104 | obama | 35 | florida | 5000000 |

| 106 | kane | 45 | alaska | 450000 |

| 107 | adam | 35 | california | 5000000 |

| 109 | brayan | 32 | alaska | 400000 |

| 110 | stephen | 40 | arisona | 600000 |

+--------+----------+---------+------------+---------+

6 rows in set

(iii) Find employee details with maximum income using a subquery.

mysql> select emp\_name,city,income from employees where income=(select max(income) from employees);

+----------+------------+---------+

| emp\_name | city | income |

+----------+------------+---------+

| obama | florida | 5000000 |

| adam | california | 5000000 |

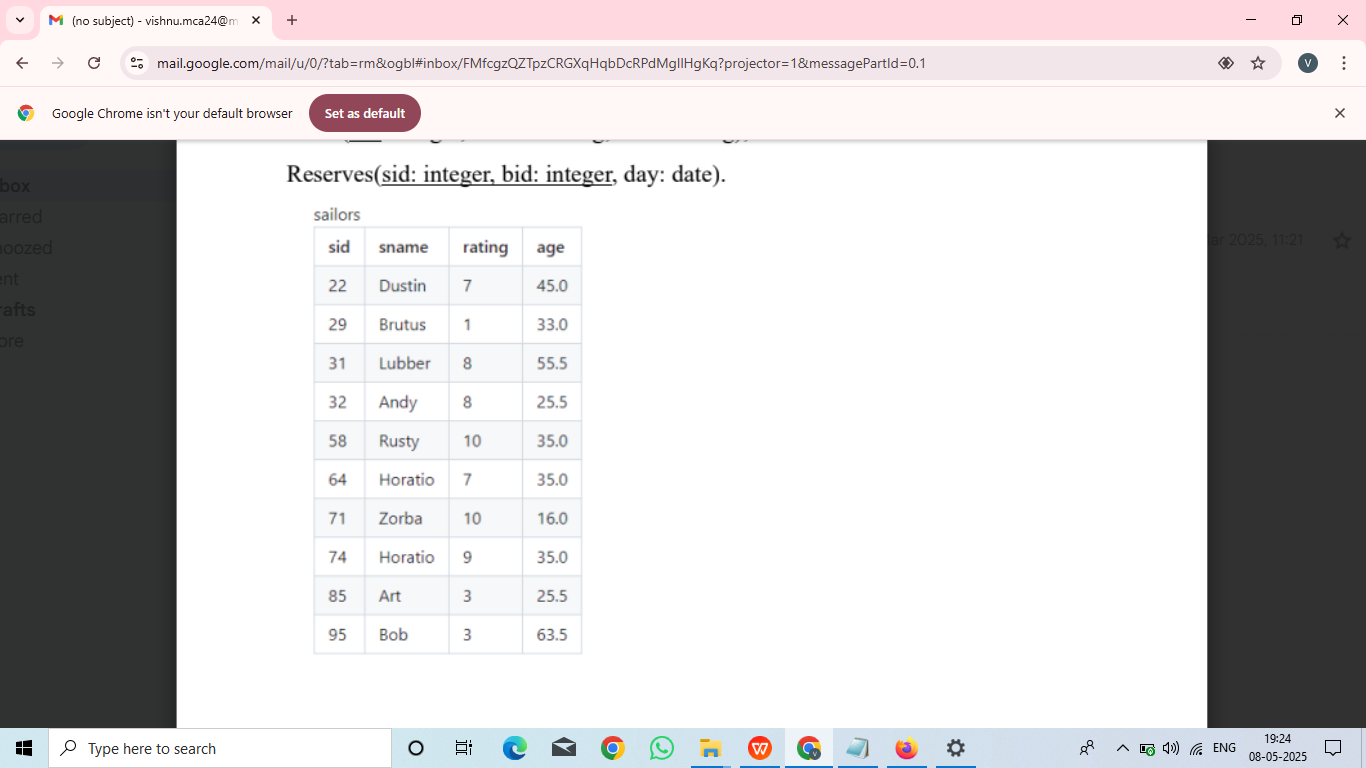
+----------+------------+---------+

2 rows in set (0.01 sec)

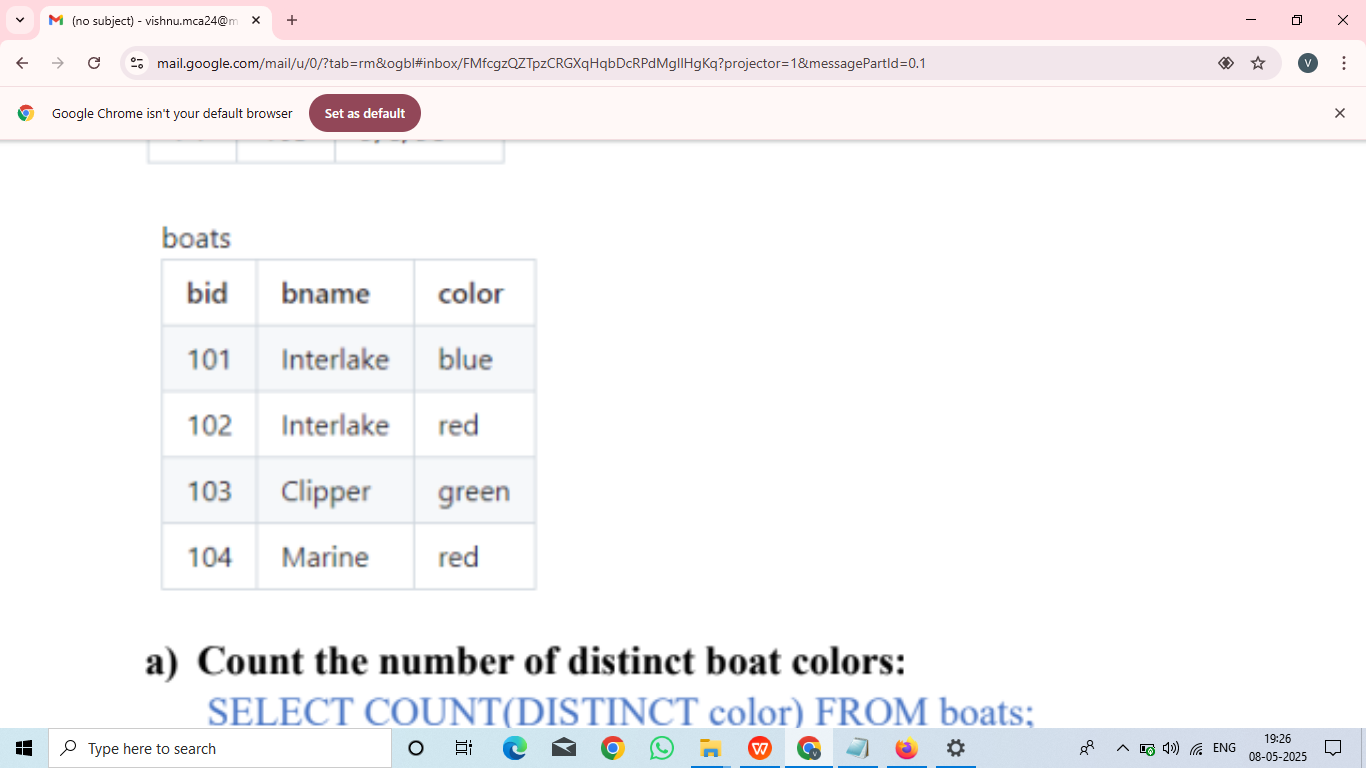
**EXPERIMENT 8**

AIM:

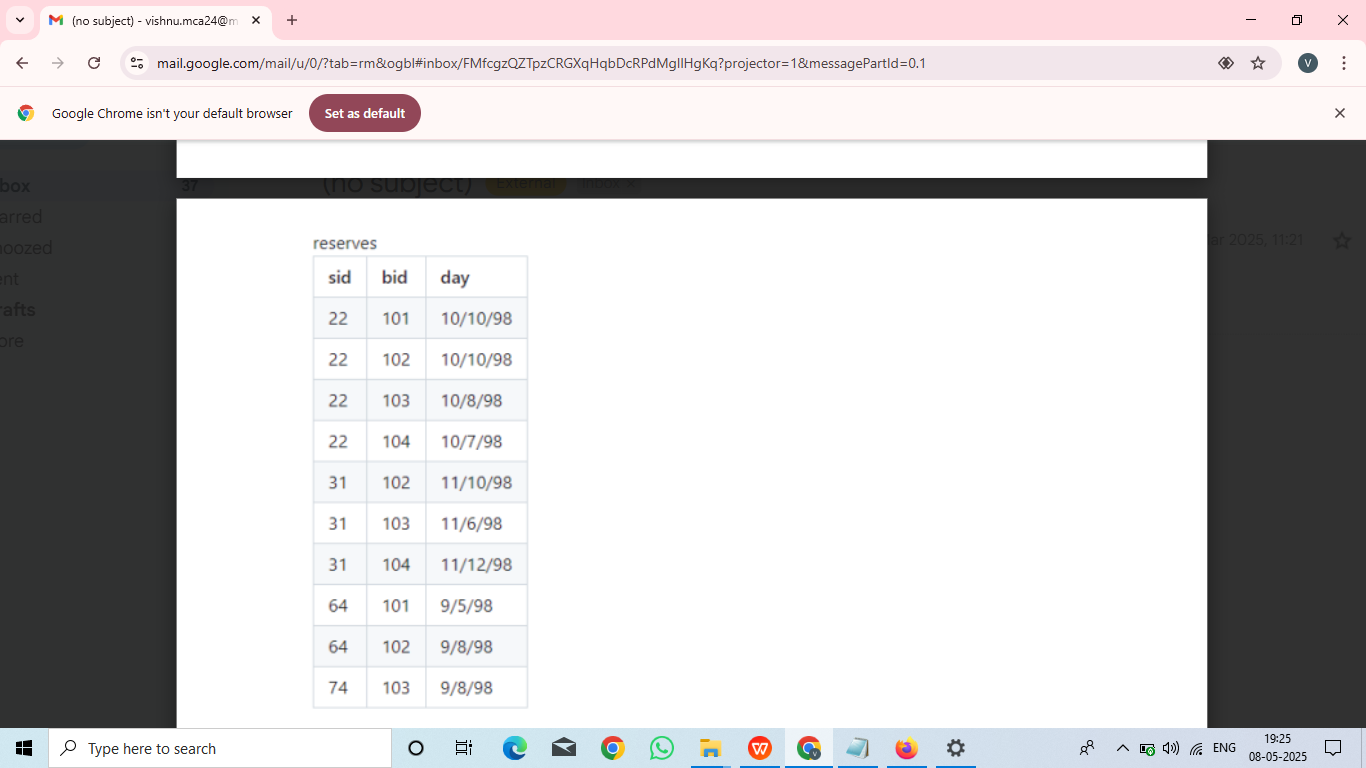
Sailors(sid: integer, sname: string, rating: integer, age: real)



Boats(bid: integer, bname: string, color: string);



Reserves(sid: integer, bid: integer, day: date).



1. Count the number of distinct boat colors:

Program code:

SELECT COUNT(DISTINCT color) FROM boats;

+-----------------------+

| COUNT(DISTINCT color) |

+-----------------------+

| 3 |

+-----------------------+

b) Find all information of sailors who have reserved boat number 101.

Program code:

select \* from sailors where sid in (select sid from reserves where bid=101);

+-----+---------+--------+------+

| sid | bname | rating | age |

+-----+---------+--------+------+

| 22 | dustbin | 7 | 45 |

| 64 | horatio | 7 | 35 |

+-----+---------+--------+------+

2 rows in set (0.01 sec)

c)Find names of sailors who have reserved at least one boat.

Program code:

SELECT sname FROM sailors S, Reserves R WHERE S.sid = R.sid;

+---------+

| sname |

+---------+

| dustbin |

| dustbin |

| dustbin |

| dustbin |

| lubber |

| lubber |

| lubber |

| horatio |

| horatio |

| horatio |

+---------+

d) Find names of sailors who have reserved a red boat and list in the order of

their age.

Program code:

select sname,age from sailors where sid in (select sid from reserves,boats where

reserves.bid=boats.bid and color='red') order by age;

+---------+------+

| sname | age |

+---------+------+

| horatio | 35 |

| dustbin | 45 |

| lubber | 55.5 |

+---------+------+

3 rows in set (0.00 sec)

e) Display boat names and the names of sailors who have sailed on them:

Program code:

mysql> SELECT b.bname, s.sname

-> FROM boats b

-> INNER JOIN reserves r ON b.bid = r.bid

-> INNER JOIN sailors s ON s.sid = r.sid;

+-----------+---------+

| bname | sname |

+-----------+---------+

| interlake | dustbin |

| interlake | horatio |

| interlake | dustbin |

| interlake | lubber |

| interlake | horatio |

| clipper | dustbin |

| clipper | lubber |

| clipper | horatio |

| marine | dustbin |

| marine | lubber |

+-----------+---------+

10 rows in set (0.00 sec)

f) Find the ids and names of sailors who have reserved two different boats on the

same day.

Program code:

mysql> SELECT DISTINCT S.sid, S.sname

-> FROM sailors S, reserves R1, reserves R2

-> WHERE S.sid = R1.sid AND R1.day = R2.day

-> AND R1.bid <> R2.bid;

+-----+---------+

| sid | sname |

+-----+---------+

| 22 | dustbin |

| 64 | horatio |

| 74 | horatio |

+-----+---------+

3 rows in set (0.00 sec)

g) Find the ids of sailors who have reserved a red boat or a green boat.

Program code:

mysql> SELECT R.sid

-> FROM boats B, reserves R

-> WHERE R.bid = B.bid AND B.color = 'red'

-> UNION

-> SELECT R2.sid

-> FROM boats B2, reserves R2

-> WHERE R2.bid = B2.bid AND B2.color = 'green';

+------+

| sid |

+------+

| 22 |

| 31 |

| 64 |

| 74 |

+------+

4 rows in set (0.00 sec)

h) Find the names of sailors who have reserved all boats.

Program code:

mysql> SELECT S.sname FROM sailors S WHERE NOT EXISTS (SELECT B.bid FROM boats B WHERE NOT EXISTS(SELECT R.bid FROM reserves R WHERE R.bid = B.bid AND R.sid = S.sid));

+---------+

| sname |

+---------+

| dustbin |

+---------+

1 row in set (0.00 sec)

**EXPERIMENT 9**

AIM: Write a PL/SQL procedure to display all fields from a table .

Program code:

mysql> delimiter //

mysql> create procedure display()

-> begin

-> select \* from sailors

-> end

-> //

mysql> delimiter ;

mysql> call display();

+-----+---------+--------+------+

| sid | sname | rating | age |

+-----+---------+--------+------+

| 22 | dustbin | 7 | 45 |

| 29 | brutus | 1 | 33 |

| 31 | lubber | 8 | 55.5 |

| 32 | andy | 8 | 25.5 |

| 58 | rusty | 10 | 35 |

| 64 | horatio | 7 | 35 |

| 71 | zorba | 10 | 16 |

| 74 | horatio | 9 | 35 |

| 85 | art | 3 | 25.5 |

| 95 | bob | 3 | 63.5 |

+-----+---------+--------+------+

10 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**EXPERIMENT 10**

AIM: Write a PL/SQL procedure to add two numbers.

Program code:

mysql> create procedure `sum` (IN a INT,IN b INT) begin declare c INT; set c=a+b; select concat('sum of two numbers=',c)as result; end///

Query OK, 0 rows affected (0.02 sec)

mysql> delimiter ;

mysql> call `sum`(5,10);

+-----------------------+

| result |

+-----------------------+

| sum of two numbers=15 |

+-----------------------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**EXPERIMENT 11**

AIM: Write a PL/SQL procedure to check whether a number is odd or even.

Program code:

mysql> DELIMITER //

mysql> CREATE PROCEDURE CheckOddOrEven(IN input\_number INT)

-> BEGIN

-> IF MOD(input\_number, 2) = 0 THEN

-> SELECT 'Even' AS result;

-> ELSE

-> SELECT 'Odd' AS result;

-> END IF;

-> END //

Query OK, 0 rows affected (0.02 sec)

mysql> delimiter ;

mysql> call CheckOddOrEven(2);

+--------+

| result |

+--------+

| Even |

+--------+

1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

**EXPERIMENT 12**

AIM: Write a PL/SQL procedure to find Factorial of a number

Program code:

mysql> delimiter //

mysql> create procedure fact(in a int)

-> begin

-> declare f int default 1;

-> while a>0 do

-> set f=f\*a;

-> set a=a-1;

-> end while;

-> select concat('fact=',f)as result;

-> end //

Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;

mysql> call fact(6);

+----------+

| result |

+----------+

| fact=720 |

+----------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**EXPERIMENT 13**

AIM:.Write a PL/SQL procedure to find maximum of three values.

Program code:

mysql>delimiter //

mysql> CREATE PROCEDURE find\_max\_of\_three(IN a INT, IN b INT, IN c INT)

->BEGIN

->DECLARE max\_value INT;

->IF a >= b AND a >= c THEN

->SET max\_value = a;

->ELSEIF b >= a AND b >= c THEN

->SET max\_value = b;

->ELSE

->SET max\_value = c;

->END IF;

->SELECT CONCAT('Max value = ', max\_value) AS result;

-> END//

Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;

mysql> call find\_max\_of\_three(6,8,2);

+---------------+

| result |

+---------------+

| Max value = 8 |

+---------------+

1 row in set (0.00 sec)

**EXPERIMENT 14**

AIM:Write a PL/SQL procedure to find the sum of digits

Program code:

mysql> delimiter //

mysql> create procedure sumof(in a int)

-> begin

-> declare sum int default 0;

-> declare digit int;

-> while a>0 do

-> set digit=mod(a,10);

-> set sum=sum+digit;

-> set a=a div 10;

-> end while;

-> select concat('sum of digits=',sum)as result;

-> end //

Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;

mysql> call sumof(123);

+-----------------+

| result |

+-----------------+

| sum of digits=6 |

+-----------------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**EXPERIMENT 15**

SQL Trigger

AIM:.Given an employees table that stores employee details and an audit\_log table that

records changes made to the employees table. A trigger has been implemented to

automatically log insertions into the audit\_log table whenever a new employee is

added.

Program code:

mysql>create table employees(employee\_id int primary key,first\_name varchar(50),last\_name varchar(50),salary decimal(10,2));

Query OK, 0 rows affected (0.09 sec)

mysql> create table audit\_log(log\_id int primary key auto\_increment,action varchar(255),timestamp timestamp default current\_timestamp);

Query OK, 0 rows affected (0.11 sec)

mysql> insert into employees(employee\_id,first\_name,last\_name,salary)values(1,'jhon','doe',55000.00);

Query OK, 3 rows affected (0.02 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> insert into audit\_log(action,timestamp)values('new employees',current\_timestamp);

Query OK, 1 row affected (0.02 sec)

mysql> select \* from employees;

+-------------+------------+-----------+----------+

| employee\_id | first\_name | last\_name | salary |

+-------------+------------+-----------+----------+

| 1 | jhon | doe | 55000.00 |

| 2 | jane | smith | 60000.00 |

| 3 | michael | jhonson | 55000.00 |

+-------------+------------+-----------+----------+

3 rows in set (0.00 sec)

mysql> select \*from audit\_log;;

+--------+---------------------+---------------------+

| log\_id | action | timestamp |

+--------+---------------------+---------------------+

| 1 | new employees added | 2025-03-18 14:26:51 |

+--------+---------------------+---------------------+

1 row in set (0.00 sec)

mysql> delimiter //

mysql> create trigger after\_employee\_insert after insert on employees

-> for each row

-> begin

-> insert into audit\_log(action,timestamp)

-> values('new employee inserted ',now())

-> end//

mysql> DELIMITER ;

mysql> INSERT INTO employees (employee\_id, first\_name, last\_name, salary)

-> VALUES (4, 'Minnu', 'Joseph', 65000.00);

Query OK, 1 row affected (0.02 sec)

mysql> select \*from employees;

+-------------+------------+-----------+----------+

| employee\_id | first\_name | last\_name | salary |

+-------------+------------+-----------+----------+

| 1 | jhon | doe | 55000.00 |

| 2 | jane | smith | 60000.00 |

| 3 | michael | jhonson | 55000.00 |

| 4 | minnu | joseph | 65000.00 |

+-------------+------------+-----------+----------+

4 rows in set (0.00 sec)

mysql> select \* from audit\_log;

+--------+------------------------+---------------------+

| log\_id | action | timestamp |

+--------+------------------------+---------------------+

| 1 | new employees added | 2025-03-18 14:26:51 |

| 2 | new employee inserted | 2025-03-18 14:39:34 |

+--------+------------------------+---------------------+

2 rows in set (0.00 sec)

**EXPERIMENT 16**

AIM: MySQL Trigger : Example BEFORE UPDATE

Program code:

mysql> create table std1(studentid int primary key,name varchar(20),sub1 int,sub2 int,sub3 int,sub4 int,sub5 int,total int,per\_marks int,grade varchar(20));

Query OK, 0 rows affected (0.17 sec)

mysql> insert into std1(studentid,name,sub1,sub2,sub3,sub4,sub5)values(1,'student1',0,0,0,0,0),

-> (2,'student2',0,0,0,0,0);

Query OK, 2 rows affected (0.02 sec)

Records: 2 Duplicates: 0 Warnings: 0

mysql> select \* from std1;

+-----------+----------+------+------+------+------+------+-------+-----------+-------+

| studentid | name | sub1 | sub2 | sub3 | sub4 | sub5 | total | per\_marks | grade |

+-----------+----------+------+------+------+------+------+-------+-----------+-------+

| 1 | student1 | 0 | 0 | 0 | 0 | 0 | NULL | NULL | NULL |

| 2 | student2 | 0 | 0 | 0 | 0 | 0 | NULL | NULL | NULL |

+-----------+----------+------+------+------+------+------+-------+-----------+-------+

2 rows in set (0.01 sec)

mysql> delimiter //

mysql> CREATE TRIGGER std\_before\_update

-> BEFORE UPDATE ON std1

-> FOR EACH ROW

-> BEGIN

-> DECLARE total\_marks INT;

-> DECLARE per\_marks INT;

-> DECLARE grade VARCHAR(20);

-> SET total\_marks = NEW.sub1 + NEW.sub2 + NEW.sub3 + NEW.sub4 + NEW.sub5;

-> SET per\_marks = total\_marks / 5;

-> IF per\_marks >= 90 THEN

-> SET grade = 'excellent';

-> ELSEIF per\_marks >= 75 AND per\_marks < 90 THEN

-> SET grade = 'very good';

-> ELSEIF per\_marks >= 60 AND per\_marks < 75 THEN

-> SET grade = 'good';

-> ELSEIF per\_marks >= 40 AND per\_marks < 60 THEN

-> SET grade = 'average';

-> ELSE

-> SET grade = 'not promoted';

-> END IF;

-> SET NEW.total = total\_marks;

-> SET NEW.per\_marks = per\_marks;

-> SET NEW.grade = grade;

-> END;

-> //

Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;

mysql> UPDATE std1 SET sub1 = 90, sub2 = 75, sub3 = 90, sub4 = 95, sub5 = 85 WHERE studentID = 1;

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \*from std1;

+-----------+----------+------+------+------+------+------+-------+-----------+-----------+

| studentid | name | sub1 | sub2 | sub3 | sub4 | sub5 | total | per\_marks | grade |

+-----------+----------+------+------+------+------+------+-------+-----------+-----------+

| 1 | student1 | 90 | 75 | 90 | 95 | 85 | 435 | 87 | very good |

| 2 | student2 | 0 | 0 | 0 | 0 | 0 | NULL | NULL | NULL |

+-----------+----------+------+------+------+------+------+-------+-----------+-----------+

2 rows in set (0.00 sec)

**EXPERIMENT 17**

AIM: AFTER UPDATE Trigger Example

Program code:

mysql> create table students(id int not null auto\_increment,name varchar(45) not null,class int not null,email\_id varchar(65) not null,primary key(id));

Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO students(name, class, email\_id)

-> VALUES ('Stephen', 6, 'stephen@gmail.com'),

-> ('Bob', 7, 'bob@gmail.com'),

-> ('Steven', 8, 'steven@ gmail.com'),

-> ('Alexandar', 7, 'alexandar@ gmail.com');

Query OK, 4 rows affected (0.02 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> create table student\_log(user varchar(45) not null,descriptions varchar(65) not null);

Query OK, 0 rows affected (0.11 sec)

mysql> select\* from students;

+----+-----------+-------+---------------------+

| id | name | class | email\_id |

+----+-----------+-------+---------------------+

| 1 | stephen | 6 | stephen@gmail.com |

| 2 | bob | 7 | bob@gmail.com |

| 3 | steven | 8 | steven@gmail.com |

| 4 | alexandra | 7 | alexandra@gmail.com |

+----+-----------+-------+---------------------+

4 rows in set (0.00 sec)

mysql> delimiter //

mysql> create trigger ater\_update\_stdnts after update

-> on students

-> for each row

-> begin

-> insert into student\_log values(user(),

-> concat("update student record",old.name,"previous class:",

-> old.class,"present class",new.class));

-> end //

Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;

mysql> select \* from student\_log;

Empty set (0.01 sec)

mysql> update students set class=class+1;

Query OK, 4 rows affected (0.02 sec)

Rows matched: 4 Changed: 4 Warnings: 0

mysql> select \*from student\_log;

+----------------+--------------------------------------------------------------+

| user | descriptions |

+----------------+--------------------------------------------------------------+

| root@localhost | update student recordstephenprevious class:6present class7 |

| root@localhost | update student recordbobprevious class:7present class8 |

| root@localhost | update student recordstevenprevious class:8present class9 |

| root@localhost | update student recordalexandraprevious class:7present class8 |

+----------------+--------------------------------------------------------------+

4 rows in set (0.00 sec)

mysql> select \*from students;

+----+-----------+-------+---------------------+

| id | name | class | email\_id |

+----+-----------+-------+---------------------+

| 1 | stephen | 7 | stephen@gmail.com |

| 2 | bob | 8 | bob@gmail.com |

| 3 | steven | 9 | steven@gmail.com |

| 4 | alexandra | 8 | alexandra@gmail.com |

+----+-----------+-------+---------------------+

4 rows in set (0.00 sec)

mysql> DELIMITER //

mysql>

mysql> CREATE PROCEDURE calculate\_av\_salary()

-> BEGIN

-> DECLARE cur\_salary INT;

-> DECLARE total\_salary INT DEFAULT 0;

-> DECLARE num\_rows INT DEFAULT 0;

-> DECLARE avg\_salary INT DEFAULT 0;

-> DECLARE done BOOLEAN DEFAULT FALSE;

-> DECLARE salary\_cursor CURSOR FOR

-> SELECT salary FROM employees;

-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

-> OPEN salary\_cursor;

-> fetch\_loop: LOOP

-> FETCH salary\_cursor INTO cur\_salary;

-> IF done THEN

-> LEAVE fetch\_loop;

-> END IF;

-> IF cur\_salary IS NOT NULL THEN

-> SET total\_salary = total\_salary + cur\_salary;

-> SET num\_rows = num\_rows + 1;

-> END IF;

-> END LOOP fetch\_loop;

-> CLOSE salary\_cursor;

-> IF num\_rows > 0 THEN

-> SET avg\_salary = total\_salary / num\_rows;

-> END IF;

-> SELECT avg\_salary;

-> END //

Query OK, 0 rows affected (0.03 sec)

**EXPERIMENT 18**

Aim:Execution of cursor

Write a program in PL/SQL to find average salary using cursor

A cursor in MySQL is used to iterate over multiple rows from a query result set one row

at a time. In this stored procedure, the cursor helps retrieve and process each salary value

individually from the employees table.

Program code:

mysql>

DELIMITER //

CREATE PROCEDURE calculate\_av\_salary()

BEGIN

DECLARE cur\_salary INT;

DECLARE total\_salary INT DEFAULT 0;

DECLARE num\_rows INT DEFAULT 0;

DECLARE avg\_salary INT DEFAULT 0;

DECLARE done BOOLEAN DEFAULT FALSE;

DECLARE salary\_cursor CURSOR FOR

SELECT salary FROM employees;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN salary\_cursor;

fetch\_loop: LOOP

FETCH salary\_cursor INTO cur\_salary;

IF done THEN

LEAVE fetch\_loop;

END IF;

IF cur\_salary IS NOT NULL THEN

SET total\_salary = total\_salary + cur\_salary;

SET num\_rows = num\_rows + 1;

END IF;

END LOOP fetch\_loop;

CLOSE salary\_cursor;

IF num\_rows > 0 THEN

SET avg\_salary = total\_salary / num\_rows;

END IF;

SELECT avg\_salary;

END //

mysql> DELIMITER ;

mysql> select \* from employees;

+-------------+------------+-----------+----------+

| employee\_id | first\_name | last\_name | salary |

+-------------+------------+-----------+----------+

| 1 | jhon | doe | 55000.00 |

| 2 | jane | smith | 60000.00 |

| 3 | michael | jhonson | 55000.00 |

| 4 | minnu | joseph | 65000.00 |

+-------------+------------+-----------+----------+

4 rows in set (0.00 sec)

mysql> call calculate\_av\_salary();

+------------+

| avg\_salary |

+------------+

| 58750 |

+------------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**EXPERIMENT 19**

AIM:Write a program in PL/SQL to list the name of students using cursor

Program code:

mysql> delimiter $$

mysql> create procedure list\_names(inout name\_list varchar(4000))

-> begin

-> declare is\_done integer default 0;

-> declare s\_name varchar(100)demysql> delimiter $$

mysql> create procedure list\_names(inout name\_list varchar(4000))

-> begin

-> declare is\_done integer default 0;

-> declare s\_name varchar(100)default "";

-> declare stud\_cursor cursor for select Name from student;

-> declare continue handler for not found set is\_done=1;

-> open stud\_cursor;

-> get\_list: LOOP

-> fetch stud\_cursor into s\_name;

-> if is\_done = 1 then

-> leave get\_list;

-> end if;

-> set name\_list= concat(s\_name,";",name\_list);

-> end loop get\_list;

-> close stud\_cursor;

-> end $$

Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;

mysql> set@name\_list="";

Query OK, 0 rows affected (0.00 sec)

mysql> call list\_names(@name\_list);

Query OK, 0 rows affected (0.00 sec)

mysql> select @name\_list;

+----------------------------------------------------+

| @name\_list |

+----------------------------------------------------+

| vishnu;priya;vinu;vimala;jisma;vimal;vishnu;priya; |

+----------------------------------------------------+

1 row in set (0.00 sec)

**.**

**EXPERIMENT 20**

AIM: MongoDB

Program code:

use text

switched to db text

db.createCollection("student")

{ "ok" : 1 }

> db.student.insert({rollno:101,name:"alex",branch:"mca"})

WriteResult({ "nInserted" : 1 })

> db.student.insert({rollno:102,name:"maya",branch:"mba"})

WriteResult({ "nInserted" : 1 })

> db.student.insert({rollno:103,name:"niya",branch:"bca"})

WriteResult({ "nInserted" : 1 })

> db.student.find().pretty()

{

"\_id" : ObjectId("67ecce0b63a77874b24b6c16"),

"rollno" : 101,

"name" : "alex",

"branch" : "mca"

}

{

"\_id" : ObjectId("67ecce2163a77874b24b6c17"),

"rollno" : 102,

"name" : "maya",

"branch" : "mba"

}

{

"\_id" : ObjectId("67ecce3363a77874b24b6c18"),

"rollno" : 103,

"name" : "niya",

"branch" : "bca"

}

**EXPERIMENT 21**

AIM:Designing NoSQL Database - Employee Management

 Create a NoSQL database named "Employee".

 Create a collection named "EMPL" with fields: "Empno", "Name", "Salary", and

"Role".

 Insert 10 records into the "EMPL" collection.

Display the data from the "EMPL" collection in a proper format.

Retrieve employees from the "EMPL" collection based on their roles.

Update the salary of an employee in the "EMPL" collection.

Program code:

use Employee1

switched to db Employee1

> db.createCollection("EMPL")

{ "ok" : 1 }

>db.EMPL.insertMany([{"empno":1,"name":"john doe","salary":6000,"rodb.EMPL.insertMany([{"empno":1,"name":"john doe","salary":6000,"rodb.EMPL.insdb.EMPL.insertMany([{"empno":1,"name":"john doe","salary":6000,"role":"manager"},{ "Empno": 2, "Name": "Alice Smith", "Salary": 50000, "Role": "Developer"}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("67f35c3008c4e276144362af"),

ObjectId("67f35c3008c4e276144362b0")

]

}

> db.EMPL.find().pretty()

{

"\_id" : ObjectId("67f35c3008c4e276144362af"),

"empno" : 1,

"name" : "john doe",

"salary" : 6000,

"role" : "manager"

}

{

"\_id" : ObjectId("67f35c3008c4e276144362b0"),

"Empno" : 2,

"Name" : "Alice Smith",

"Salary" : 50000,

"Role" : "Developer"

}

> db.EMPL.find({"role":"manager"}).pretty()

{

"\_id" : ObjectId("67f35c3008c4e276144362af"),

"empno" : 1,

"name" : "john doe",

"salary" : 6000,

"role" : "manager"

}

> db.EMPL.updateOne( {"name":"alice smith"}, {$set:{"salary":5500}})

{ "acknowledged" : true, "matchedCount" : 0, "modifiedCount" : 0 }

**EXPERIMENT 22**

AIM: Performing CRUD Operations - Product Catalog

 Create a NoSQL database named "ProductCatalog".

 Create a collection named "Products" with fields: "ProductID", "ProductName",

"Price", and "Quantity".

 Insert several records into the "Products" collection.

 Display the data from the "Products" collection.

 Update the details of a specific product. (For example, increase the quantity of

laptops by 10)

 Delete a product from the collection. (For example, remove the smartphone

from the catalog.)

Program code:

use productcatalog1

switched to db productcatalog1

> db.createCollection("products")

{ "ok" : 1 }

> db.products.insertMany([

... {"productid":1,"productname":"laptop","price":1000,"quantity":20},{"productid":2,"productname":"smartphone","price":500,"quantity":30},{"productid":3,"productname":"joystick","price":1500,"quantity":25},{"productid":4,"productname":"earpod","price":10000,"quantity":5},{"productid":5,"productname":"hardisk","price":4500,"quantity":8}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("67f5fed41234568b935fa81f"),

ObjectId("67f5fed41234568b935fa820"),

ObjectId("67f5fed41234568b935fa821"),

ObjectId("67f5fed41234568b935fa822"),

ObjectId("67f5fed41234568b935fa823")

]

}

> db.products.find().pretty()

"\_id" : ObjectId("67f5fed41234568b935fa81f"),

"productid" : 1,

"productname" : "laptop",

"price" : 1000,

"quantity" : 20

}

{

"\_id" : ObjectId("67f5fed41234568b935fa820"),

"productid" : 2,

"productname" : "smartphone",

"price" : 500,

"quantity" : 30

}

{

"\_id" : ObjectId("67f5fed41234568b935fa821"),

"productid" : 3,

"productname" : "joystick",

"price" : 1500,

"quantity" : 25

}

{

"\_id" : ObjectId("67f5fed41234568b935fa822"),

"productid" : 4,

"productname" : "earpod",

"price" : 10000,

"quantity" : 5

}

{

"\_id" : ObjectId("67f5fed41234568b935fa823"),

"productid" : 5,

"productname" : "hardisk",

"price" : 4500,

"quantity" : 8

}

> db.products.updateOne( {"productname":"laptop"},{$inc:{"quantity":10}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

>db.products.deleteOne({ "productname": "smartphone" } )

{ "acknowledged" : true, "deletedCount" : 1 }

> db.products.find().pretty()

{

"\_id" : ObjectId("67f5fed41234568b935fa81f"),

"productid" : 1,

"productname" : "laptop",

"price" : 1000,

"quantity" : 20

}

{

"\_id" : ObjectId("67f5fed41234568b935fa821"),

"productid" : 3,

"productname" : "joystick",

"price" : 1500,

"quantity" : 25

}

{

"\_id" : ObjectId("67f5fed41234568b935fa822"),

"productid" : 4,

"productname" : "earpod",

"price" : 10000,

"quantity" : 5

}

{

"\_id" : ObjectId("67f5fed41234568b935fa823"),

"productid" : 5,

"productname" : "hardisk",

"price" : 4500,

"quantity" : 8

}

**EXPERIMENT 23**

AIM:Usage of aggregate function

Program code:

> use students

switched to db students

> db.students.insert({name:"Tony",age:17,id:1,sec:"A",subject:["physics","maths"]})

WriteResult({ "nInserted" : 1 })

> db.students.insert({name:"Bony",age:18,id:2,sec:"B",subject:["physics","chemistry"]})

WriteResult({ "nInserted" : 1 })

> db.students.insert({name:"swetha",age:28,id:3,sec:"B",subject:["biology","chemistry"]})

WriteResult({ "nInserted" : 1 })

> db.students.insert({name:"meenashi",age:18,id:4,sec:"A",subject:["biology","english"]})

WriteResult({ "nInserted" : 1 })

> db.students.insert({name:"rahul",age:21,id:5,sec:"A",subject:["biology","english"]})

WriteResult({ "nInserted" : 1 })

> db.students.find().pretty()

{

"\_id" : ObjectId("67f5fed53c0f794bb1e3b54d"),

"name" : "Tony",

"age" : 17,

"id" : 1,

"sec" : "A",

"subject" : [

"physics",

"maths"

]

}

{

"\_id" : ObjectId("67f5fef83c0f794bb1e3b54e"),

"name" : "Bony",

"age" : 18,

"id" : 2,

"sec" : "B",

"subject" : [

"physics",

"chemistry"

]

}

{

"\_id" : ObjectId("67f5ff1c3c0f794bb1e3b54f"),

"name" : "swetha",

"age" : 28,

"id" : 3,

"sec" : "B",

"subject" : [

"biology",

"chemistry"

]

}

{

"\_id" : ObjectId("67f5ff3f3c0f794bb1e3b550"),

"name" : "meenashi",

"age" : 18,

"id" : 4,

"sec" : "A",

"subject" : [

"biology",

"english"

]

}

{

"\_id" : ObjectId("67f5ff523c0f794bb1e3b551"),

"name" : "rahul",

"age" : 21,

"id" : 5,

"sec" : "A",

"subject" : [

"biology",

"english"

]

}

>

i)Displaying the total number of students in one section only

> db.students.aggregate([{$match:{sec:"B"}},{$count:"Total student in sec:B"}])

{ "Total student in sec:B" : 2 }

ii) Displaying the total number of students in both the sections and maximum age

from both section

> db.students.aggregate([{$match:{sec:"B"}},{$count:"Total student in sec:B"}])

{ "Total student in sec:B" : 4 }

> db.students.aggregate([{$group: {\_id:"$sec",total\_st: {$sum:1},max\_age:{$max:"$age"}}}])

{ "\_id" : "A", "total\_st" : 3, "max\_age" : 21 }

{ "\_id" : "B", "total\_st" : 2, "max\_age" : 28 }

>

iii)Displaying details of students whose age is greater than 20 using match stage

> db.students.aggregate([{$match:{age:{$gt:20}}}]).pretty()

{

"\_id" : ObjectId("67f60315f69c2fa90c4e6905"),

"name" : "swetha",

"age" : 28,

"id" : 3,

"sec" : "B",

"subject" : [

"biology",

"chemistry"

]

}

{

"\_id" : ObjectId("67f60323f69c2fa90c4e6907"),

"name" : "rahul",

"age" : 21,

"id" : 5,

"sec" : "A",

"subject" : [

"biology",

"english"

]

}

iv)Sorting the students on the basis of age

db.students.aggregate([{"$sort": {"age":1}}])

{ "\_id" : ObjectId("67f5fed53c0f794bb1e3b54d"), "name" : "Tony", "age" : 17, "id" : 1, "sec" : "A", "subject" : [ "physics", "maths" ] }

{ "\_id" : ObjectId("67f5ff3f3c0f794bb1e3b550"), "name" : "meenashi", "age" : 18, "id" : 4, "sec" : "A", "subject" : [ "biology", "english" ] }

{ "\_id" : ObjectId("67f6030ff69c2fa90c4e6904"), "name" : "Bony", "age" : 18, "id" : 2, "sec" : "B", "subject" : [ "physics", "chemistry" ] }

{ "\_id" : ObjectId("67f5ff523c0f794bb1e3b551"), "name" : "rahul", "age" : 21, "id" : 5, "sec" : "A", "subject" : [ "biology", "english" ] }" ] }

{ "\_id" : ObjectId("67f60315f69c2fa90c4e6905"), "name" : "swetha", "age" : 28, "id" : 3, "sec" : "B", "subject" : [ "biology", "chemistry" ] }

v)Counting the total numbers of documents

> db.students.count()

5

vi))Displaying distinct names and ages (non-repeating)

> db.students.distinct("name")

[ "Bony", "Tony", "meenashi", "rahul", "swetha" ]

> db.students.distinct("age")

[ 17, 18, 21, 28 ]

**EXPERIMENT 24**

AIM:create users and roles in the employee database

Program code:

> use employee

switched to db employee

> db.createCollection("EMPL")

{ "ok" : 1 }

> db.EMPL.insertMany([

... {"empno":1,"name":"john doe","salary":60000,"role":"manager"},

... {"empno":2,"name":"alice smith","salary":5000,"role":"developer"},])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("67f60df5d579642e45880d74"),

ObjectId("67f60df5d579642e45880d75")

]

}

> db.EMPL.find().pretty()

{

"\_id" : ObjectId("67f60df5d579642e45880d74"),

"empno" : 1,

"name" : "john doe",

"salary" : 60000,

"role" : "manager"

}

{

"\_id" : ObjectId("67f60df5d579642e45880d75"),

"empno" : 2,

"name" : "alice smith",

"salary" : 5000,

"role" : "developer"

}

>