Module-4: Introduction to DBMS

Lab exercise

❖ 1.INTRODUCTION TO SQL:-

→ Lab 1: Create a new database named school_db and a table called students with the following columns: student_id, student_name, age, class, and address.

Query:-

```
create database school_db//for database create

use school_db//use database
```

Query:-

```
create table student(student_id int primary key
auto_increment,student_name varchar(20), age int,

class varchar(20),address varchar(20))
```

→ Lab 2: Insert five records into the students table and retrieve all records using the SELECT statement.

Query:-

```
insert into student(student_name,age,class,address)
values(“Yogesh”,20,”graduate”,”Modasa,Gujarat”),

("Vanraj",20,"graduate","Megraj,Gujarat"),

("Ayan",20,"graduate","Ahmedabad,Gujarat"),

("Yash",23,"graduate","Rajasthan"),
```

("Jay",20,"graduate","Porbandor,Gujarat")//data inserted successfully

Query:-

Select * from student//retrive all inserted data

//output:-

student_id	student_name	age	class	address
1	Yogesh	20	graduate	Modasa,Gujarat
2	Vanraj	20	graduate	Megraj,Gujarat
3	Ayan	20	graduate	Ahmedabad,Gujarat
4	Yash	23	graduate	Rajasthan
5	Jay	20	graduate	Porbandor,Gujarat
NULL	NULL	NULL	NULL	NULL

❖ 2.SQL SYNTAX-

→Lab 1: Write SQL queries to retrieve specific columns (student_name and age) from the students table.

Query:-

select student_name,age from student//select specific data

//output

student_name	age
Yogesh	20
Vanraj	20
Ayan	20
Yash	23
Jay	20

→Lab 2: Write SQL queries to retrieve all students whose age is greater than 10.

Query:-

select * from student where age>10

//output:-

student_id	student_name	age	class	address
1	Yogesh	20	graduate	Modasa,Gujarat
2	Vanraj	20	graduate	Megraj,Gujarat
3	Ayan	20	graduate	Ahmedabad,Gujarat
4	Yash	23	graduate	Rajasthan
5	Jay	20	graduate	Porbandor,Gujarat
NULL	NULL	NULL	NULL	NULL

❖ Que-3. SQL CONSTRAINTS:-

→Lab 1: Create a table teachers with the following columns: teacher_id (Primary Key), teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

Query:-

create table teachers(teacher_id int primary key auto_increment,teacher_name varchar(20) not null,subject varchar(20) not null,email varchar(30) unique)

//output

teacher_id	teacher_name	subject	email
NULL	NULL	NULL	NULL

→Lab 2: Implement a FOREIGN KEY constraint to relate the teacher_id from the teachers table with the students table

→First add teacher_id field in student table:

Query:-

alter table student add column teacher_id int//add new column in student.

→foreign key add in column:-

alter table student add constraint foreign key(teacher_id) references teachers(teacher_id)

//output

student_id	student_name	age	class	address	teacher_id
1	Yogesh	20	graduate	Modasa,Gujarat	NULL
2	Vanraj	20	graduate	Megraj,Gujarat	NULL
3	Ayan	20	graduate	Ahmedabad,Gujarat	NULL
4	Yash	23	graduate	Rajasthan	NULL
5	Jay	20	graduate	Porbandor,Gujarat	NULL
NULL	NULL	NULL	NULL	NULL	NULL

❖4. Main SQL Commands and Sub-commands(DDL):-

→Lab 1: Create a table courses with columns: course_id, course_name, and course_credits. Set the course_id as the primary key.

Query:-

create table course(course_id int primary key auto_increment,course_name varchar(20),course_credits varchar(20))

//output

Field	Type	Null	Key	Default	Extra
course_id	int	NO	PRI	NULL	auto_increment
course_name	varchar(20)	YES		NULL	
course_credits	varchar(20)	YES		NULL	

→Lab 2: Use the CREATE command to create a database university_db.

Query:-

create database university_db//database created successfully.

❖ 5. ALTER COMMAND:-

→Lab 1: Modify the courses table by adding a column course_duration using the ALTER command.

Query:-

alter table course add course_duration varchar(20) //colum added

---

→Lab 2: Drop the course_credits column from the courses table.

Query:-

alter table course drop column course_credits//column dropped

---

//output:

Field	Type	Null	Key	Default	Extra
course_id	int	NO	PRI	NULL	auto_increment
course_name	varchar(20)	YES		NULL	
course_duration	varchar(20)	YES		NULL	

❖ 6. DROP COMMAND:-

→Lab 1: Drop the teachers table from the school_db database.

Query:-

drop table teachers;//manually

→Lab 2: Drop the students table from the school_db database and verify that the table has been removed.

Query:-

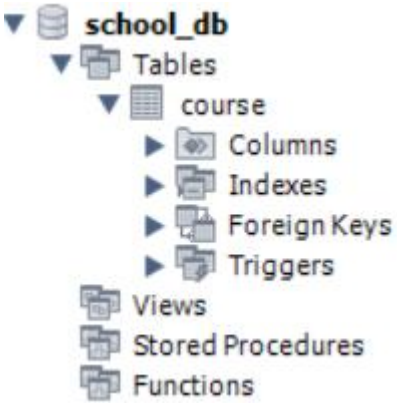
drop table student;//manually

Note :- this manually query can't work because we can use foreign key that connect teacher_id in both student and teacher table so these query use for it

Query:-

drop tables teacher,student

//verification |Output:-



//only course table there

❖7. Data Manipulation Language (DML):-

→Lab 1: Insert three records into the courses table using the INSERT command.

Query:-//bydefault value of course_id start with 101

insert into course(course_name,course_duration) values("Full Stack","12-15 Months"),("Graphicd Designer","10-14 Months"),("UI/Ux","7-11 Months")

//output

course_id	course_name	course_duration
101	Full Stack	12-15 Months
102	Graphicd Designer	10-14 Months
103	UI/Ux	7-11 Months
NULL	NULL	NULL

→ Lab 2: Update the course duration of a specific course using the UPDATE command.

Query:-

update course set course_duration="1 year" where course_id=101//for 101

update course set course_duration=" less than 1 year" where course_id=102// for 102

update course set course_duration="more than 1 year" where course_id=103//for 103

//output

course_id	course_name	course_duration
101	Full Stack	1 year
102	Graphicd Designer	less than 1 year
103	UI/Ux	more than 1 year
NULL	NULL	NULL

→Lab 3: Delete a course with a specific course_id from the courses table using the DELETE command.

Query:-

delete from course where course_id=102//delete 102 course

❖ 8. Data Query Language (DQL):-

→Lab 1: Retrieve all courses from the courses table using the SELECT statement.

Query:- select * from course;

→Lab 2: Sort the courses based on course_duration in descending order using ORDER BY.

Query:-

select * from course order by course_duration desc;

→Lab 3: Limit the results of the SELECT query to show only the top two courses using LIMIT.

Query:- select * from course order by course_duration desc limit 1;

❖9. Data Control Language (DCL):-

→Lab 1: Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.

Query:-

create user 'user1'@'localhost' identified by 'Mysql';

create user 'user2'@'localhost' identified by 'Mysql';

grant select on course to user1@localhost

→Lab 2: Revoke the INSERT permission from user1 and give it to user2

Query:-

revoke insert on course from user1@localhost

grant insert on course to user2@localhost

❖10. Transaction Control Language (TCL):-

→Lab 1: Insert a few rows into the courses table and use COMMIT to save the changes.

Query:-

```
insert into  course(course_name,course_duration) values
('C++ Programming', '1 month'),
('Java Fundamentals', '6 months'),
('Python for Beginners', '7 months'),
('Web Development', '2 months'),
('Database Management', '3 months')

Commit;
```

→Lab 2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

Query:-

```
start transaction;
Insert into course(course_name,course_duration) values
('Advance php','3 Months'),
('Advance java','3 Months');
rollback;
```

→Lab 3: Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

Query:-

start transaction;

savepoint sp2;

update course set course_duration='1 Month'

where course_id=103;

rollback to sp2;

❖ 11. SQL JOIN:-

→Lab 1: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

Query:-//create table

create table department

(dept_id int primary key auto_increment,

dept_name varchar(20));

create table employee

(emp_id int primary key auto_increment,

emp_name varchar(20),

```
dept_id int,
foreign key(dept_id) references department(dept_id))
```

Query : //insert data into table

```
INSERT INTO department VALUES (1, 'HR');
INSERT INTO department VALUES (2, 'IT');
INSERT INTO department VALUES (3, 'Finance');

-- Insert employees
INSERT INTO employee VALUES (1001, 'Yogesh', 2);
INSERT INTO employee VALUES (1002, 'Ram', 1);
INSERT INTO employee VALUES (1003, 'Rohit', 3);
INSERT INTO employee VALUES (1004, 'Sai', 2);
```

Query:-//Inner Join

```
Select emp_id,emp_name,department.dept_name
 from employee
Inner join department
on employee.dept_id=department.dept_id
order by emp_id;
```

→Lab 2: Use a LEFT JOIN to show all departments, even those without employees.

Query :- //first we can add new department

insert into department values(4,'Management');

Query:-//Left Join

Select emp_id,emp_name,department.dept_name

from department left join employee

on employee.dept_id=department.dept_id

order by emp_id;

❖12. SQL Group By :-

→Lab 1: Group employees by department and count the number of employees in each department using GROUP BY.

Query:-

select count(emp_id) as Total_Emp,dept_name

from employee

right join department

on employee.dept_id=department.dept_id

group by dept_name;

→Lab 2: Use the AVG aggregate function to find the average salary of employees in each department.

Query:-//alter the new row emp_sal

Alter table employee add emp_sal int ;

Query:-update all row

update employee set emp_sal=100000 where emp_id=1001;

update employee set emp_sal=70000 where emp_id=1002;

update employee set emp_sal=40000 where emp_id=1003;

update employee set emp_sal=80000 where emp_id=1004;

Query:- Search Average

select avg(emp_sal) as avg_sal,dept_name

from employee,department

where employee.dept_id=department.dept_id

group by dept_name;

Output:-

avg_sal	dept_name
90000.0000	IT
70000.0000	HR
40000.0000	Finance

❖ 13. SQL Stored Procedure:-

→Lab 1: Write a stored procedure to retrieve all employees from the employees table based on department.

Query:-//create procedure

```
delimiter //
create procedure getinfo(deptname varchar(20))
begin
select emp_id,emp_name,emp_sal,dept_name from employee
join department
on employee.dept_id=department.dept_id
where dept_name=deptname;
end//
```

Query:-//call procedure

```
call getinfo('IT');
```

→Lab 2: Write a stored procedure that accepts course_id as input and returns the course details

Query:-//create procedure

```
delimiter //
create procedure getcourseinfo(courseid int)
begin
select * from course where course_id=courseid;
```

end//

Query:- //Call procedure

call getcourseinfo(101);

❖ 14. SQL View:-

→Lab 1: Create a view to show all employees along with their department names.

Query:-//create view

create view f_view

as

select emp_id,emp_name,emp_sal,dept_name

from employee,department

where employee.dept_id=department.dept_id;

Query:-//select all data

Select * from f_view;

→Lab 2: Modify the view to exclude employees whose salaries are below $50,000.

Query:-//drop old view

Drop view if exists f_view;

Query:-// create modified view

create view f_view

as

select emp_id,emp_name,emp_sal,dept_name

from employee,department

where employee.dept_id=department.dept_id and emp_sal>50000;

Query:-

Select * from f_view;

❖15. SQL Triggers :-

→Lab 1: Create a trigger to automatically log changes to the employees table when a new employee is added.

Query:-

delimiter //

create trigger log

after insert on employee

for each row

begin

insert into employee(emp_name,emp_sal,dept_id) values(new.emp_name,new.emp_sal,new.dept_id);

end //

Query:-

insert into employee(emp_name,emp_sal,dept_id)
value('Rahul',85000,2);

Note:- Must Be Run on Online Editor .

→Lab 2: Create a trigger to update the last_modified timestamp whenever an employee record is updated.

Query:-//first we can add new column for time data added/or modified

alter table employee add column last_modified timestamp default current_timestamp ;

delimiter //

create trigger last_edited

before update on employee

for each row

begin

set new.last_modified=current_timestamp;

end//

delimiter ;

Query:-//update something to check trigger

update employee set emp_name='Rahul' where emp_id=1004;

❖ 16. Introduction to PL/SQL:-

→Lab 1: Write a PL/SQL block to print the total number of employees from the employees table.

Note :- cannot create pl/sql block derectly in mysql

```
delimiter //
create procedure get_total_emp()
begin
declare total int;
select count(emp_id) into total from employee;
select concat('total employee: ',total) as Result;
end//
delimiter ;

call get_total_emp();
```

→Lab 2: Create a PL/SQL block that calculates the total sales from an orders table.

Query:

```
delimiter //
create procedure cal_total_sale()
```

```
begin
declare total int;
select sum(sales) into total from orders;
select concat('total sales:- ',total) as result;
end//
delimiter ;
call cal_total_sale();
```

❖17. PL/SQL Control Structures:-

→Lab 1: Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

Query:-

```
delimiter //
 create procedure check_dept(employ_id int)
 begin
 declare emp_dept int;
 select dept_id into emp_dept from employee where emp_id=employ_id;
 if emp_dept= 1 then
 select concat('Employee',employ_id,'work in HR department') as message;
 elseif emp_dept=2 then
```

```
select concat('Employee',employ_id,'work in IT department') as message;

elseif emp_dept=3 then

select concat('Employee',employ_id,'work in Finance department') as message;

elseif emp_dept=4 then

select concat('Employee',employ_id,'work in Management department') as message;

end if;

end//

 delimiter ;

 call check_dept(1001);
```

→Lab 2: Use a FOR LOOP to iterate through employee records and display their names.

Query:-

```
delimiter //

create procedure loop_ex()

begin

declare done int default false;

declare empname varchar(20);
```

```
declare emp_cur cursor for
select emp_name from employee;

declare continue handler for not found set done=true;
open emp_cur;
emp_loop:loop
fetch emp_cur into empname;
if done then
leave emp_loop;
end if;
select empname as 'employee name ';
end loop;
close emp_cur;

end//
delimiter ;

call loop_ex();
```

❖ 18. SQL Cursors:-

→Lab 1: Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

```
delimiter //
create procedure cur()
begin
declare c_emp_id int;
declare c_emp_name varchar(20);
declare c_emp_sal int;
declare done int default false;

declare emp_cur cursor for
select emp_id,emp_name,emp_sal from employee;

declare continue handler for not found set done=true;
open emp_cur;

read_loop:loop
fetch emp_cur into c_emp_id,c_emp_name,c_emp_sal;
if done then
leave read_loop;
end if;
```

```
select concat('ID= ',c_emp_id,' Name= ',c_emp_name,' Salary= ',c_emp_sal)as emp_details;

end loop;

end//

delimiter ;


call cur();
```

→Lab 2: Create a cursor to retrieve all courses and display them one by one.

```
delimiter //

create procedure cur_2()

begin

declare done int default false;

declare c_course_id int;

declare c_course_name varchar(30);

declare c_course_duration varchar(20);


declare course_cur cursor for

select course_id,course_name,course_duration from course;


declare continue handler for not found set done=true;
```

```
create temporary table if not exists tmp_course(
cid int,c_name varchar(20),c_dura varchar(20));
truncate table tmp_course;
open course_cur;
read_loop:loop
fetch course_cur into
c_course_id,c_course_name,c_course_duration;
if done then
leave read_loop;
end if;
insert into tmp_course
values(c_course_id,c_course_name,c_course_duration);
end loop;
close course_cur;
select * from tmp_course;
end//
delimiter ;
call cur_2();
```

❖ 19. Rollback and Commit Savepoint :-

→Lab 1: Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

```
start transaction;


select * from  employee; -- check already exists data


insert into employee(emp_name,dept_id,emp_sal)
values('om',4,150000);

insert into employee(emp_name,dept_id,emp_sal)
values('Anuj',2,50000);


savepoint sp1;



insert into employee(emp_name,dept_id,emp_sal)
values('rudra',1,20000);

insert into employee(emp_name,dept_id,emp_sal)
values('prince',3,10000);


rollback to sp1;

commit;


select * from employee; -- check update data
```

→Lab 2: Commit part of a transaction after using a savepoint and then rollback the remaining changes

```
start transaction;

insert into employee(emp_name,dept_id,emp_sal)
values('rudra',1,20000);

insert into employee(emp_name,dept_id,emp_sal)
values('prince',3,10000);

savepoint sp2;

commit;

insert into employee(emp_name,dept_id,emp_sal)
values('rahul',2,20000);

insert into employee(emp_name,dept_id,emp_sal)
values('pandya',4,10000);

rollback;

select * from employee;
```
