**Module-4 Lab Exercise**

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. Insert five records into the students table and retrieve all records using the SELECT statement.

Query:- create database School\_db;

use School\_db;

create table students(students\_id integer primary key auto\_increment,

student\_name varchar(20),

age integer,

class varchar(10),

address varchar(30));

insert into students(Student\_name,age,class,address)values

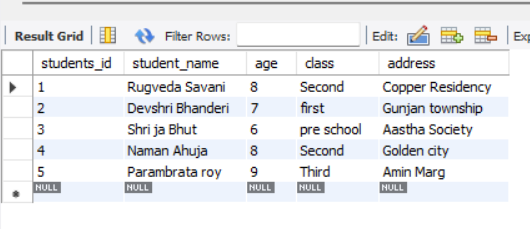
('Rugveda Savani',8,'Second','Copper Residency'),

('Devshri Bhanderi',7,'first','Gunjan township'),

('Shri ja Bhut',6,'pre school','Aastha Society'),

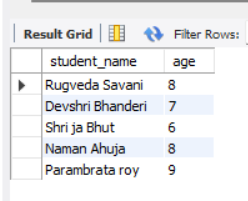
('Naman Ahuja',8,'Second','Golden city'),

('Parambrata roy',9,'Third','Amin Marg');

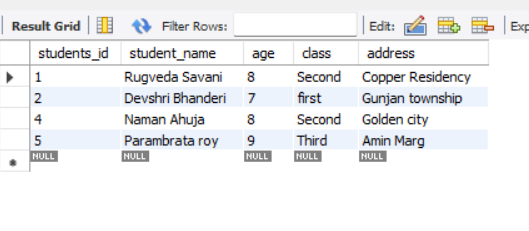
select \*from students;

1. Write SQL queries to retrieve specific columns (student\_name and age) from the student’s table. Write SQL queries to retrieve all students whose age is greater than 6.

Query: - select student\_name, age from students;



Query: - select \*from students where age>6;



1. Create a table teachers with the following columns: teacher\_id (Primary Key), teacher\_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).Implement a FOREIGN KEY constraint to relate the teacher\_id from the teachers table with the students table.

Query: - create table teachers(teacher\_id integer primary key auto\_increment,

teacher\_name varchar(20) not null,

subject varchar(20) not null,

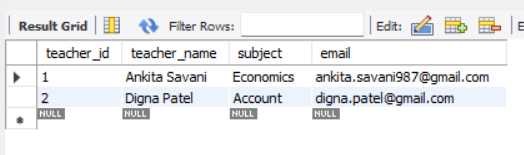
email varchar(50) unique,

foreign key (teacher\_id) references students(students\_id));

Query: - insert into teachers(teacher\_name,subject,email)values

('Ankita Savani',"Economics","ankita.savani987@gmail.com"),

('Digna Patel',"Account","digna.patel@gmail.com");



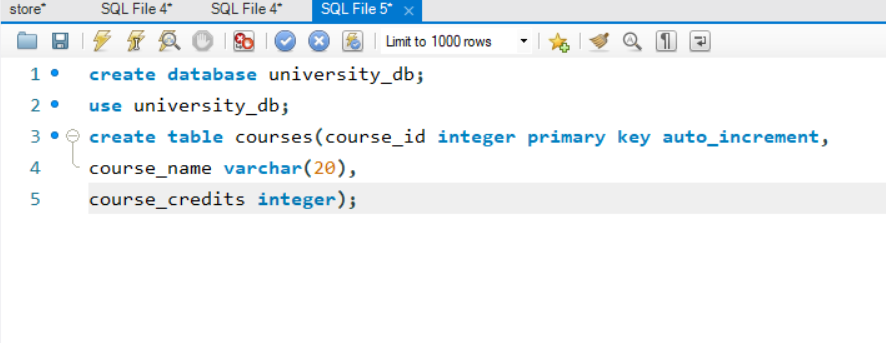
1. Use the CREATE command to create a database university\_db. Create a table courses with columns: course\_id, course\_name, and course\_credits. Set the course\_id as the primary key.

Query:- create database university\_db;

use university\_db;

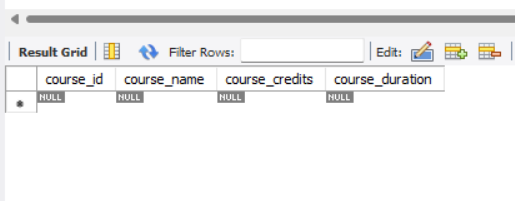
create table courses(course\_id integer primary key auto\_increment,

course\_name varchar(20),

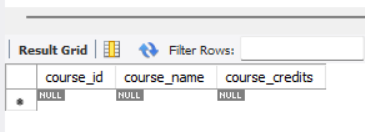
course\_credits integer);

1. Modify the courses table by adding a column course\_duration using the ALTER command. Drop the course\_credits column from the courses table.

Query: - alter table courses add column course\_duration text;



Query: - alter table courses drop column course\_duration;



1. Drop the teachers table from the school\_db database.Drop the students table from the school\_db database and verify that the table has been removed.

Query: - drop table teachers;

Query: - drop table students;

1. Insert three records into the courses table using the INSERT command. Update the course duration of a specific course using the UPDATE command. Delete a course with a specific course\_id from the courses table using the DELETE command.

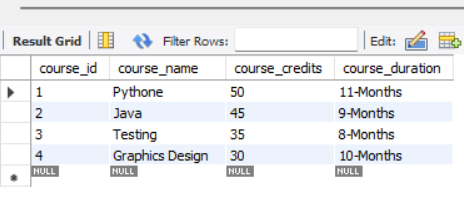
Query:- insert into courses(course\_name,course\_credits,course\_duration)values

("Pythone",50,"11-Months"),

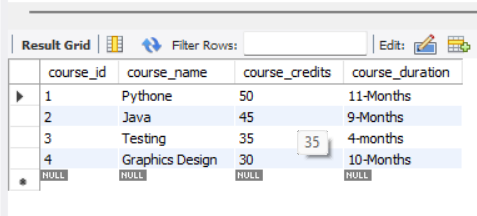
("Java",45,"9-Months"),

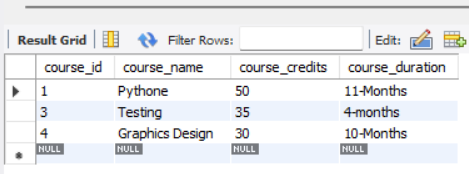
("Testing",35,"8-Months"),

("Graphics Design",30,"10-Months");



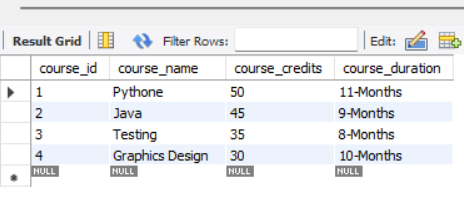
Query: - update courses set course\_duration='4-months' where course\_id=3;

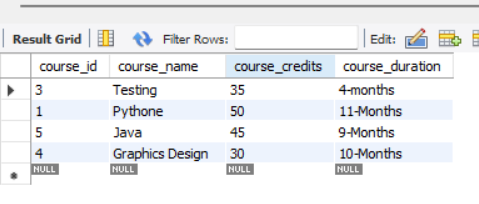


 Query: - delete from courses where course\_id=2;

1. Retrieve all courses from the courses table using the SELECT statement. Sort the courses based on course\_name in descending order using ORDER BY. Limit the results of the SELECT query to show only the top two courses using LIMIT.

Query: - select \*from courses;



 Query: - select \*from courses order by course\_name desc;

1. Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table. Revoke the INSERT permission from user1 and give it to user2.
2. Insert a few rows into the courses table and use COMMIT to save the changes. Insert additional rows, then use ROLLBACK to undo the last insert operation. Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

Query: - insert into courses(course\_name,course\_credits,course\_duration)values

("marketing",80,"2-Years"),

("Developing",75,"18-Months"),

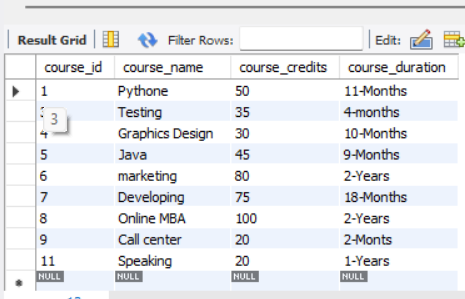
("Online MBA",100,"2-Years"),

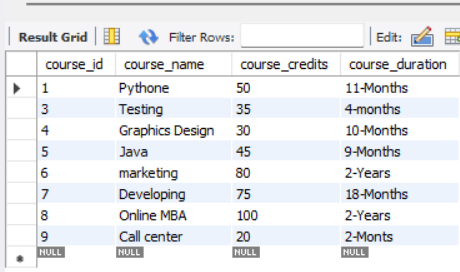
("Call center",20,"2-Monts");

Commit ;

insert into courses(course\_name,course\_credits,course\_duration)values

("Speaking",20,"1-Years");

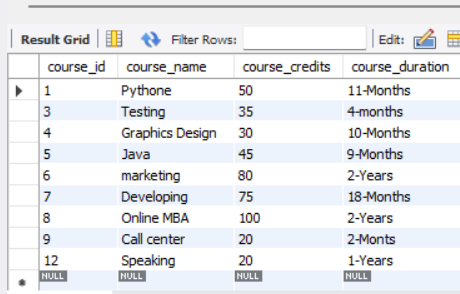
Rollback ;

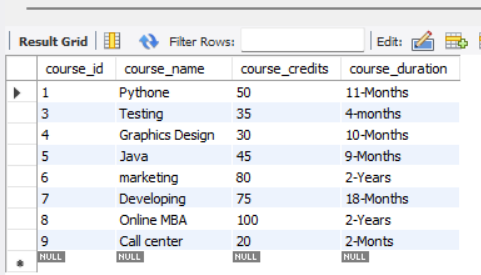


Query: - savepoint sp1;

insert into courses(course\_name,course\_credits,course\_duration)values

("Speaking",20,"1-Years");



Query: -rollback to savepoint sp1;

Query:-select \*from Departments join Employees on Departments.Department=Employees.Department;

1. Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments. Use a LEFT JOIN to show all departments, even those without employees.

Query: -

create database corporate;

use corporate;

create table Departments(id integer primary key auto\_increment,

Department text);

insert into Departments(Department)values

('Renewable Energy'),

('Farm Machinery & PE'),

('Soil & Water Conservation'),

('Food & Agro Engineering');

select \* from Departments;

create table Employees(Eid integer primary key auto\_increment,

Employ\_name text,

Department text,

Salary Real);

insert into Employees(Employ\_name,Department,Salary)values

('Mr.Jethva',"Food & Agro Engineering",75000),

('Mr.Sravan',"Farm Machinery & PE",85000),

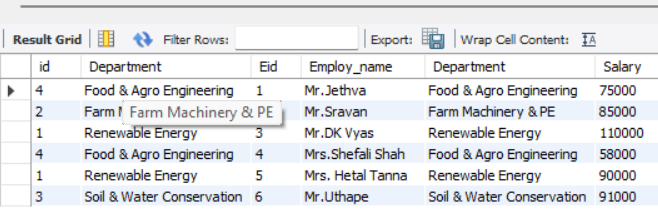
('Mr.DK Vyas',"Renewable Energy",110000),

('Mrs.Shefali Shah',"Food & Agro Engineering",58000),

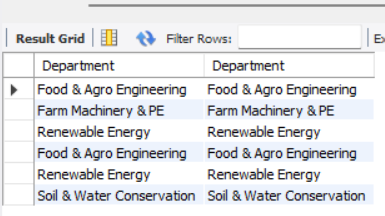
('Mrs. Hetal Tanna',"Renewable Energy",90000),

('Mr.Uthape',"Soil & Water Conservation",91000);

select \* from Employees;

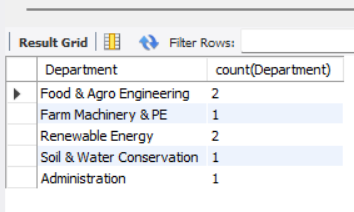
**Query:-select \*from Departments join Employees on Departments.Department=Employees.Department;**

**Query:- select Departments.Department,Employees.Department from Departments join Employees on Departments.Department=Employees.Department;**

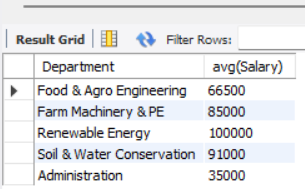
****

1. Group employees by department and count the number of employees in each department using GROUP BY. Use the AVG aggregate function to find the average salary of employees in each department.

Query: -select Department,count(Department) from Employees group by Department;



Query:- select Department,avg(Salary) from Employees group by Department;



1. Write a stored procedure to retrieve all employees from the employees table based on department. Write a stored procedure that accepts course\_id as input and returns the course details.

Query: -delimiter //

create procedure getdata()

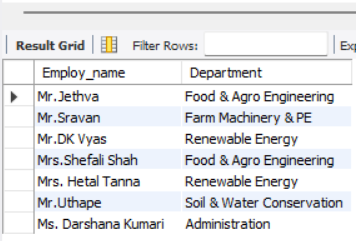
begin

select Employ\_name,Department from Employees;

end //

delimiter ;

call getdata();



Query:- delimiter //

create procedure Sdata(in s\_id int)

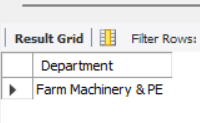
begin

select Department from Employees where (s\_id=Eid);

end //

delimiter ;

call sdata(1);



1. Create a trigger to automatically log changes to the employees table when a new employee is added. Create a trigger to update the last\_modified timestamp whenever an employee record is updated.

Query: -delimiter //

create trigger insert\_log

after insert

on Products

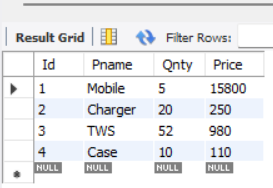
for each row

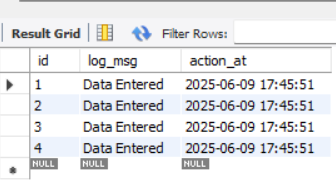
begin

insert into insert\_log(log\_msg,action\_at)values("Data Entered",now());

end //

delimiter ;





Query:- delimiter //

create trigger insert\_log1

after Update

on Products

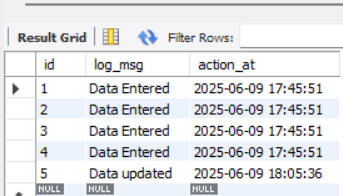
for each row

begin

insert into insert\_log(log\_msg,action\_at)values("Data updated",now());

end //

delimiter ;

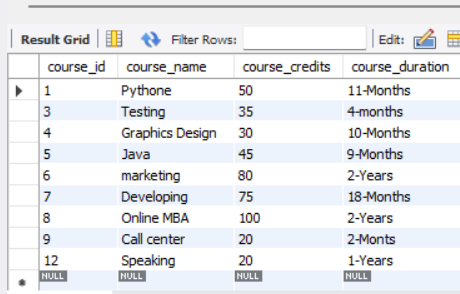


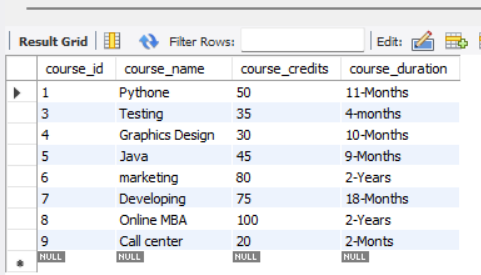
1. Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint. Commit part of a transaction after using a savepoint and then rollback the remaining changes.

Query:- Query: - savepoint sp1;

insert into courses(course\_name,course\_credits,course\_duration)values

("Speaking",20,"1-Years");



Query: -rollback to savepoint sp1;

**----End----**