**ASSIGNMENT - Student Information System (SIS)**

**J502 Abhishek.A.G**

**CREATION OF DATABASE AND TABLES**

create database SISDB;

use SISDB;

create table Students(student\_id int primary key auto\_increment,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

date\_of\_birth date,

email varchar(100) unique,phone\_number varchar(15));

create table Teacher(

teacher\_id int primary key auto\_increment,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

email varchar(100) unique);

create table Courses(

course\_id int primary key auto\_increment,

course\_name varchar(100) not null,

credits int check (credits>0),

teacher\_id int,

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

create table Enrollments(

enrollment\_id int primary key auto\_increment,

student\_id int,

course\_id int,

enrollment\_date date,

foreign key (student\_id) references Students(student\_id),

foreign key(course\_id) references Courses(course\_id));

create table Payments(

payment\_id int primary key auto\_increment,

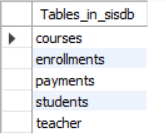
student\_id int,

amount decimal(10,2),

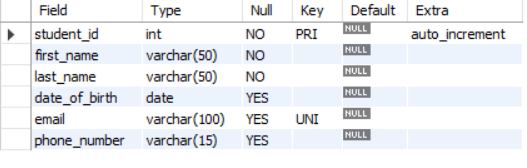
payment\_date date,

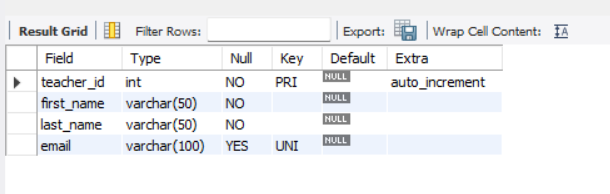
foreign key (student\_id) references Students(student\_id));

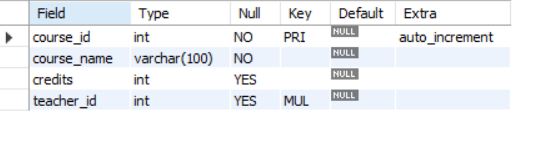
**DATABASE DESCRIPTION :**

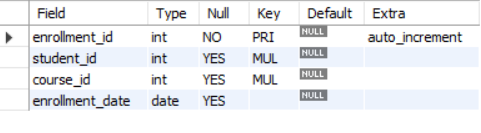
****

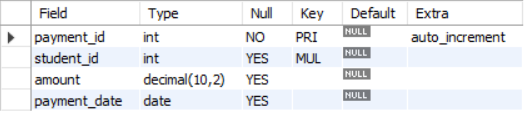
**TABLE DESCRIPTION :**

****









**Insertion Of Data**

INSERT INTO Students (first\_name, last\_name, date\_of\_birth, email, phone\_number) VALUES

('Amit', 'Sharma', '2000-05-14', 'amit.sharma@gmail.com', '9876543210'),

('Priya', 'Verma', '1999-08-22', 'priya.verma@yahoo.com', '9823456789'),

('Rahul', 'Patel', '2001-03-10', 'rahul.patel@outlook.com', '7894561230'),

('Sneha', 'Reddy', '2002-12-05', 'sneha.reddy@gmail.com', '9123456780'),

('Vikram', 'Singh', '1998-09-18', 'vikram.singh@gmail.com', '9012345678'),

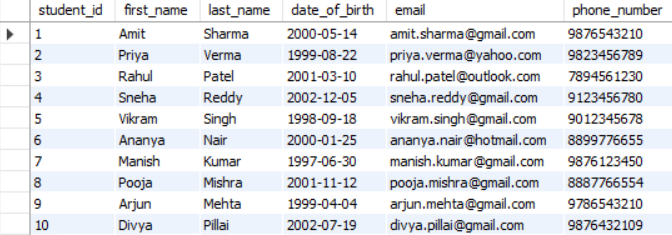
('Ananya', 'Nair', '2000-01-25', 'ananya.nair@hotmail.com', '8899776655'),

('Manish', 'Kumar', '1997-06-30', 'manish.kumar@gmail.com', '9876123450'),

('Pooja', 'Mishra', '2001-11-12', 'pooja.mishra@gmail.com', '8887766554'),

('Arjun', 'Mehta', '1999-04-04', 'arjun.mehta@gmail.com', '9786543210'),

('Divya', 'Pillai', '2002-07-19', 'divya.pillai@gmail.com', '9876432109');



INSERT INTO Teacher (first\_name, last\_name, email) VALUES

('Suresh', 'Iyer', 'suresh.iyer@college.edu'),

('Lakshmi', 'Menon', 'lakshmi.menon@college.edu'),

('Rajesh', 'Deshmukh', 'rajesh.deshmukh@college.edu'),

('Neha', 'Kapoor', 'neha.kapoor@college.edu'),

('Arvind', 'Chatterjee', 'arvind.chatterjee@college.edu'),

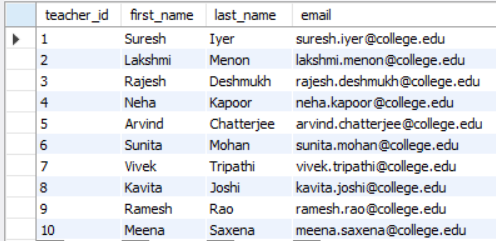
('Sunita', 'Mohan', 'sunita.mohan@college.edu'),

('Vivek', 'Tripathi', 'vivek.tripathi@college.edu'),

('Kavita', 'Joshi', 'kavita.joshi@college.edu'),

('Ramesh', 'Rao', 'ramesh.rao@college.edu'),

('Meena', 'Saxena', 'meena.saxena@college.edu');



INSERT INTO Courses (course\_name, credits, teacher\_id) VALUES

('Data Structures', 4, 1),

('Database Management Systems', 3, 2),

('Operating Systems', 4, 3),

('Computer Networks', 3, 4),

('Software Engineering', 3, 5),

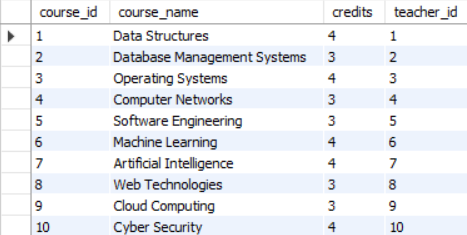
('Machine Learning', 4, 6),

('Artificial Intelligence', 4, 7),

('Web Technologies', 3, 8),

('Cloud Computing', 3, 9),

('Cyber Security', 4, 10);



INSERT INTO Enrollments (student\_id, course\_id, enrollment\_date) VALUES

(1, 1, '2024-01-15'),

(2, 2, '2024-01-16'),

(3, 3, '2024-01-17'),

(4, 4, '2024-01-18'),

(5, 5, '2024-01-19'),

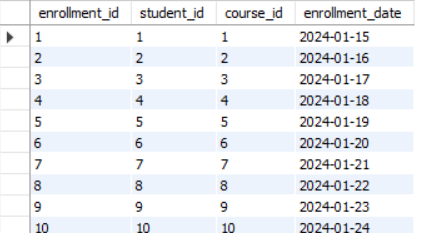
(6, 6, '2024-01-20'),

(7, 7, '2024-01-21'),

(8, 8, '2024-01-22'),

(9, 9, '2024-01-23'),

(10, 10, '2024-01-24');



INSERT INTO Payments (student\_id, amount, payment\_date) VALUES

(1, 15000.00, '2024-01-25'),

(2, 12000.00, '2024-01-26'),

(3, 13000.00, '2024-01-27'),

(4, 14000.00, '2024-01-28'),

(5, 11000.00, '2024-01-29'),

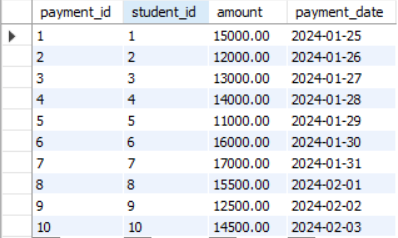
(6, 16000.00, '2024-01-30'),

(7, 17000.00, '2024-01-31'),

(8, 15500.00, '2024-02-01'),

(9, 12500.00, '2024-02-02'),

(10, 14500.00, '2024-02-03');



**TASK 2 : Select, Where, Between, AND, LIKE**

**1. Write an SQL query to insert a new student into the "Students" table with the following details:**

**a. First Name: John**

**b. Last Name: Doe**

**c. Date of Birth: 1995-08-15**

**d. Email: john.doe@example.com**

**e. Phone Number: 1234567890**

insert into Students (first\_name,last\_name,date\_of\_birth,email,phone\_number) values('John','Doe','1995-08-15','john.doe@example.com','1234567890');

**2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.**

insert into Enrollments(student\_id,course\_id,enrollment\_date) values(2,5,'2025-03-18');

**3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address**

update Teacher set email='thisisupdateemail@gmail.com' where teacher\_id=4;

**4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.**

delete from Enrollments where student\_id=6 and course\_id=6;

**5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.**

update Courses set teacher\_id=2 where course\_id=3;

**6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.**

delete from Enrollments where student\_id=7;

delete FROM Payments WHERE student\_id = 7;

delete from Students where student\_id=7;

**7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.**

update Payments set amount=20000 where payment\_id=6;

**TASK 3 : Aggregate functions, Having, Order By, GroupBy and Joins**

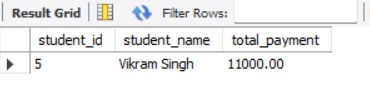
1. **Write an SQL query to calculate the total payments made by a specific student.**

**You will need to join the "Payments" table with the "Students" table based on the student's ID.**

select s.student\_id,concat(first\_name,' ',last\_name) as student\_name,sum(p.amount) as total\_payment

from Students s inner join Payments p on s.student\_id=p.student\_id

where s.student\_id=5 group by s.student\_id,student\_name;



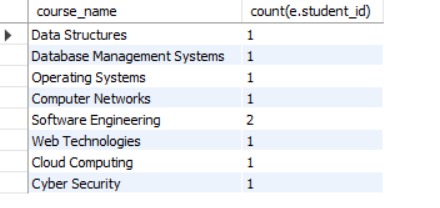
**2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each**

**course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.**

select c.course\_name,count(e.student\_id)

from Courses c inner join Enrollments e on c.course\_id=e.course\_id

group by c.course\_id,c.course\_name;



3. **Write an SQL query to find the names of students who have not enrolled in any course. Use a**

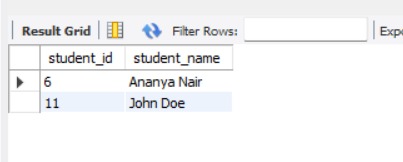
**LEFT JOIN between the "Students" table and the "Enrollments" table to identify students**

**without enrollments.**

select s.student\_id,concat(s.first\_name,' ',s.last\_name) as student\_name

from Students s left join Enrollments e on s.student\_id=e.student\_id

where e.student\_id is NULL;



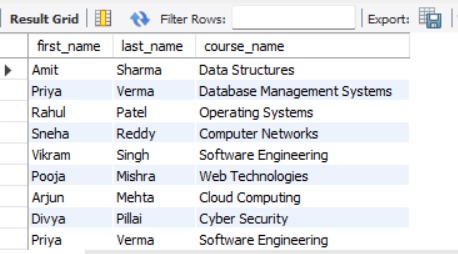
**4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.**

select s.first\_name,s.last\_name,c.course\_name

from Students s inner join Enrollments e on s.student\_id=e.student\_id

inner join Courses c on e.course\_id=c.course\_id

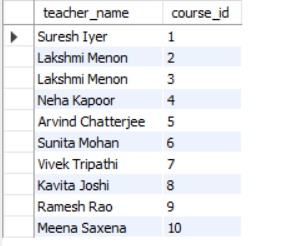
group by s.student\_id,c.course\_name;



**5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.**

select t.first\_name,t.last\_name,course\_id

from Teacher t inner join Courses c on t.teacher\_id=c.teacher\_id;



**6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.**

select concat(s.first\_name,' ',s.last\_name) as student\_name,e.enrollment\_date,c.course\_name

from Students s inner join Enrollments e on s.student\_id=e.student\_id

inner join Courses c on c.course\_id=e.course\_id where c.course\_name='Data Structures';

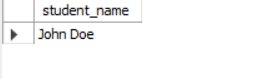


**7. Find the names of students who have not made any payments. Use a LEFT JOIN between the**

**"Students" table and the "Payments" table and filter for students with NULL payment records.**

select concat(s.first\_name,' ',s.last\_name) as student\_name

from Students s left join Payments p on s.student\_id=p.student\_id where p.payment\_id is null;

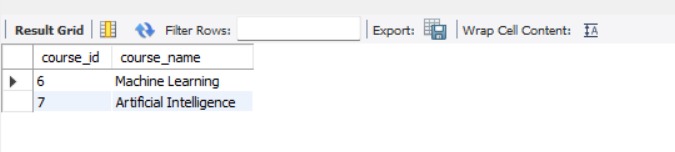


8. **Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN**

**between the "Courses" table and the "Enrollments" table and filter for courses with NULL**

**enrollment records.**

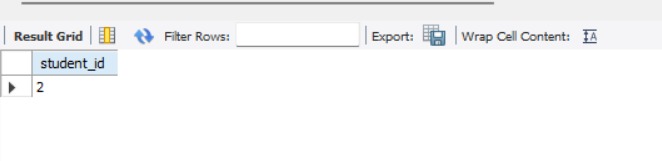
select c.course\_id,c.course\_name from Courses c left join Enrollments e on c.course\_id=e.course\_id where e.enrollment\_id is null;



**9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments"**

**table to find students with multiple enrollment records.**

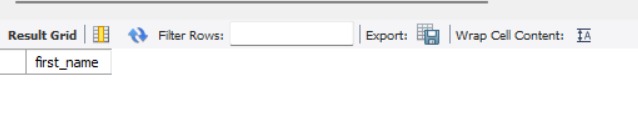
SELECT DISTINCT e1.student\_id FROM Enrollments e1 JOIN Enrollments e2 ON e1.student\_id = e2.student\_id AND e1.course\_id <> e2.course\_id;



**10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher"**

**table and the "Courses" table and filter for teachers with NULL course assignments.**

select t.first\_name from Teacher t left join Courses c on t.teacher\_id=c.teacher\_id where c.course\_id is null;



**TASK 4 : Subquery and its type**

**1. Write an SQL query to calculate the average number of students enrolled in each course. Use**

**aggregate functions and subqueries to achieve this.**

select avg(student\_count) from (

select course\_id,count(student\_id) as student\_count from Enrollments group by course\_id

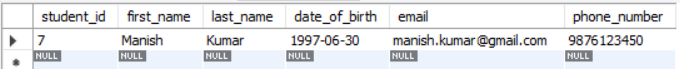
) as course\_count;



2. **Identify the student(s) who made the highest payment. Use a subquery to find the maximum**

**payment amount and then retrieve the student(s) associated with that amount.**

select \* from Students where student\_id in(select student\_id from Payments where amount=(select max(amount) from Payments));



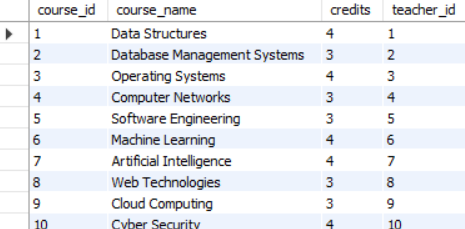
3. **Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the**

**course(s) with the maximum enrollment count.**

select \* from Courses where course\_id in(

select course\_id from(select course\_id,count(\*) as enrollment\_count from Enrollments group by course\_id) as CourseCount

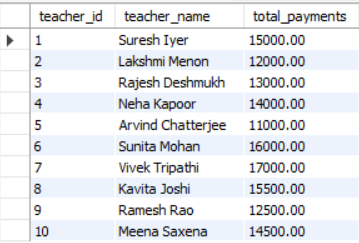
where enrollment\_count=(select max(enrollment\_count) from (select count(\*) as enrollment\_count from Enrollments group by course\_id) as EnrollmentCount));



**4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum**

**payments for each teacher's courses.**

select t.teacher\_id,concat(t.first\_name,' ',t.last\_name),(select max(p.amount) from Payments p where p.student\_id in (select e.student\_id from Enrollments e where e.course\_id in(select c.course\_id from Courses c where c.teacher\_id=t.teacher\_id) ) as total\_payment from Teacher t;



**5. Identify students who are enrolled in all available courses. Use subqueries to compare a**

**student's enrollments with the total number of courses.**

select s.student\_id,s.first\_name,s.last\_name from Students s where (select count(\*) from Enrollments e where e.student\_id=s.student\_id)=(select count(\*) from Courses);



**6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to**

**find teachers with no course assignments.**

select t.teacher\_id,t.first\_name,t.last\_name from Teacher t where t.teacher\_id not in (select distinct teacher\_id from Courses where teacher\_id is not null);



**7. Calculate the average age of all students. Use subqueries to calculate the age of each student**

**based on their date of birth.**

select avg(student\_age) from(select student\_id,timestampdiff(year,date\_of\_birth,CURDATE()) as student\_age from Students)as AGE;



**8. Identify courses with no enrollments. Use subqueries to find courses without enrollment**

**records.**

select c.course\_id,c.course\_name from Courses c where c.course\_id not in (select distinct course\_id from Enrollments where course\_id is not null);



**9. Calculate the total payments made by each student for each course they are enrolled in. Use**

**subqueries and aggregate functions to sum payments.**

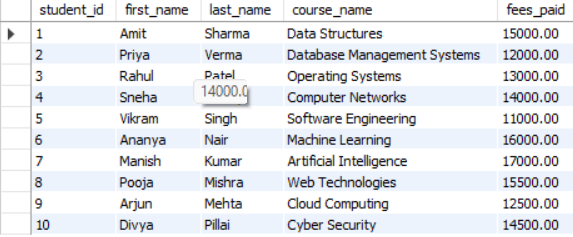
select student\_id,

(select first\_name from Students s where s.student\_id=e.student\_id) as first\_name,

(select last\_name from Students s where s.student\_id=e.student\_id) as last\_name,

(select course\_name from Courses c where c.course\_id=e.course\_id) as course\_name,

(select sum(amount) from Payments p where p.student\_id=e.student\_id) as fees\_paid from Enrollments e;



**10. Identify students who have made more than one payment. Use subqueries and aggregate**

**functions to count payments per student and filter for those with counts greater than one.**

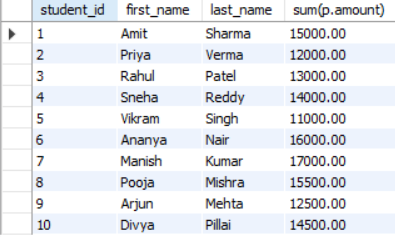
select student\_id,first\_name,last\_name from Students where student\_id in(select student\_id from Payments group by student\_id having count(payment\_id)>1);



**11. Write an SQL query to calculate the total payments made by each student. Join the "Students"**

**table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.**

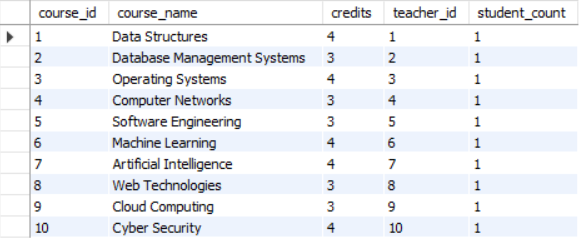
select s.student\_id,s.first\_name,s.last\_name,sum(p.amount) from Students s join Payments p on s.student\_id=p.student\_id group by s.student\_id;



**12. Retrieve a list of course names along with the count of students enrolled in each course. Use**

**JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.**

select c.\*,count(e.student\_id) as student\_count from Courses c left join Enrollments e on c.course\_id=e.course\_id group by c.course\_id;



**13. Calculate the average payment amount made by students. Use JOIN operations between the**

**"Students" table and the "Payments" table and GROUP BY to calculate the average.**

select s.first\_name,s.last\_name,s.student\_id,avg(p.amount) from Students s left join Payments p on s.student\_id=p.student\_id group by s.student\_id;

