Task 1

1.Create the database named "SISDB"

create database SISDB;

2.Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

* 1. Students
  2. Courses
  3. Enrollments
  4. Teacher
  5. Payments

a)create table Students (

studentid int primary key,

firstname varchar(255),

lastname varchar(255),

dateofbirth date,

email varchar(255),

phone\_number int(20)

);

b)Courses table

create table Courses (

courseid int primary key,

coursename varchar(255),

credits int,

teacherid int,

foreign key (teacherid) references Teacher(teacherid)

);

c)create table Enrollments (

enrollmentid int primary key,

studentid int,

courseid int,

enrollmentdate date,

foreign key (studentid) references Students(studentid),

foreign key (courseid) references Courses(courseid)

);

d)create table Teacher (

teacherid int primary key,

firstname varchar(255),

lastname varchar(255),

email varchar(255)

);

e)create table Payments (

paymentid int primary key,

studentid int,

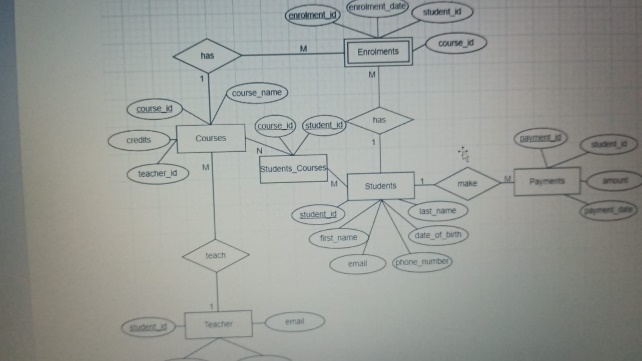
amount decimal(10, 2),

paymentdate date,

foreign key (studentid) references Students(studentid)

);

3)Create an ERD (Entity Relationship Diagram) for the database.



5)Insert at least 10 sample records into each of the following tables.

* 1. Students
  2. Courses
  3. Enrollments

iv. Teacher

v. Payments

i)insert into Students (student\_id, first\_name, last\_name, date\_of\_birth, email, phone\_number)

values

(1, 'ramesh', 'kumar', '2000-04-04', rameshkumar@email.com', 987654321'),

(2, 'kartheek', 'naidu', '2003-05-07', 'kartheeknaidu@email.com', 789654324'),

(3, 'pradeep', 'raju', '2004-04-09', pradeeprajun@email.com', 983647397'),

(4, 'nayani', 'kumari', '2004-05-06', nayanikumari@email.com',’832937383 '),

(5, 'eesha', 'priya', '2000-05-05', 'eeshapriya@email.com', ‘376478479'),

(6, 'krishna', 'veni', '2000-06-06', 'krishnaveni@email.com', 987847849'),

(7, 'rahul', 'nair', '2000-07-07', 'rahulnair@email.com', 982748939'),

(8,’bindhu’,’jyotsna’,’2008-05-06,’bindhujyotsna@email.com,’983748937’),

(9, 'venu', 'naidu', '2000-08-08', 'venunaidu@email.com', ‘837847839 '),

(10, 'narayana', ‘varma', '2000-09-09', 'narayanavarma@email.com', ‘349734783'),

ii)insert into Courses (course\_id, course\_name, credits, teacher\_id)

values

(1, 'cse', 12, 987),

(2, 'ece', 3, 2),

(3, 'eee', 3, 3),

(4, 'mech', 4, 4),

(5, 'cse\_ds', 4, 5),

(6, 'civil', 2, 6),

(7, 'aids', 2, 7),

(8, 'it', 2, 8),

(9, 'cse\_ai', 3, 9),

(10, 'aiml', 3, 10);

iii) insert into Enrollments (enrollment\_id, student\_id, course\_id, enrollment\_date)

values

(1, 1, 1, '2022-09-01'),

(2, 1, 2, '2022-09-01'),

(3, 2, 1, '2022-09-01'),

(4, 2, 3, '2022-09-01'),

(5, 3, 1, '2022-09-01'),

(6, 3, 2, '2022-09-01'),

(7, 4, 3, '2022-09-01'),

(8, 4, 4, '2022-09-01'),

(9, 5, 5, '2022-09-01'),

(10, 5, 6, '2022-09-01');

iv)insert into Teacher (teacher\_id, first\_name, last\_name, email)

values

(1, 'Professor', 'Smriti', 'prof.smrithi@example.com'),

(2, 'Professor', 'Sowmiya', 'prof.sowmiya@example.com'),

(3, 'Professor', 'Karpagavalli', 'prof.karpagavalli@example.com'),

(4, 'Professor', 'Barghav', 'prof.barghav@example.com'),

(5, 'Professor', 'Danush', 'prof.dansuh@example.com'),

(6, 'Professor', 'Mahendra', 'prof.mahendra@example.com'),

(7, 'Professor', 'Ganesh', 'prof.ganesh@example.com'),

(8, 'Professor', 'Ranjith', 'prof.ranjith@example.com'),

(9, 'Professor', 'Lakshmi', 'prof.lakshmi@example.com'),

(10, 'Professor', 'Ranganath', 'prof.ranganath@example.com');

v)insert into Payments (payment\_id, student\_id, amount, payment\_date)

values

(1, 1, 100.00, '2022-09-01'),

(2, 1, 150.00, '2022-10-01'),

(3, 2, 120.00, '2022-09-15'),

(4, 2, 130.00, '2022-10-15'),

(5, 3, 110.00, '2022-09-01'),

(6, 3, 120.00, '2022-10-01'),

(7, 4, 140.00, '2022-09-01'),

(8, 4, 150.00, '2022-10-01'),

(9, 5, 130.00, '2022-09-15'),

(10, 5, 140.00, '2022-10-15');

select \* from Students;

select \* from Courses;

select \* from Enrollments;

select \* from Teacher;

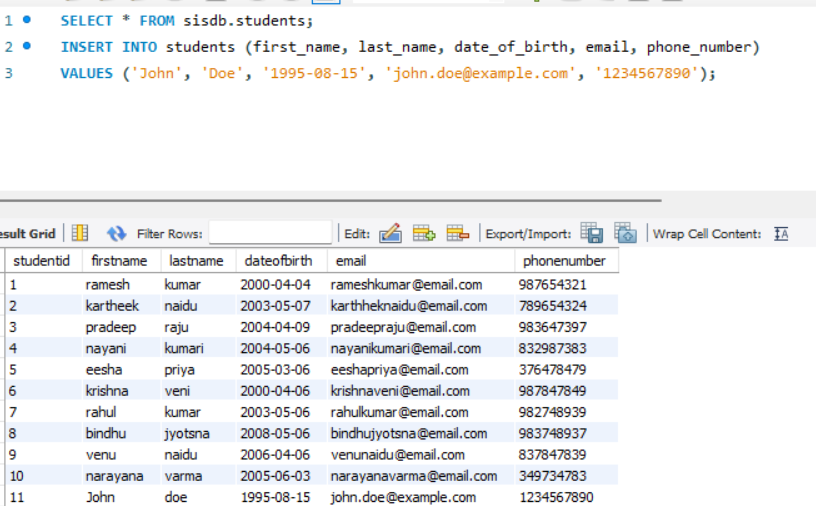
select \* from Payments;

**Tasks 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
   1. Last Name: Doe
   2. Date of Birth: 1995-08-15
   3. Email: john.doe@example.com
   4. Phone Number: 1234567890

SELECT \* FROM sisdb.students;

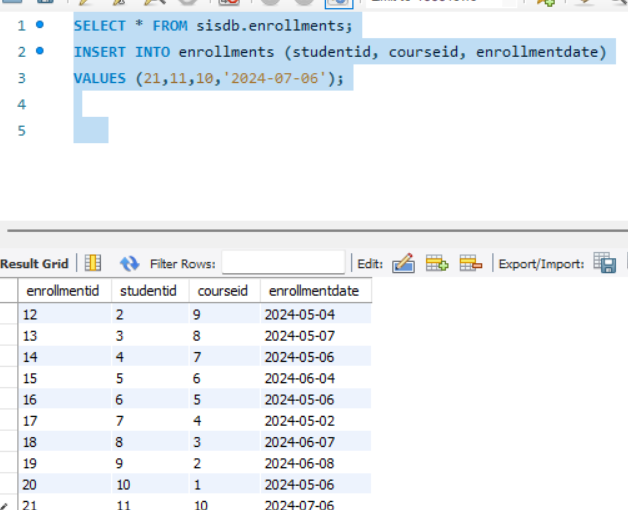
INSERT INTO students (first\_name, last\_name, date\_of\_birth, email, phone\_number)

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 (studentid, courseid, enrollmentdate)

VALUES (21,11,’2024-07-06,);



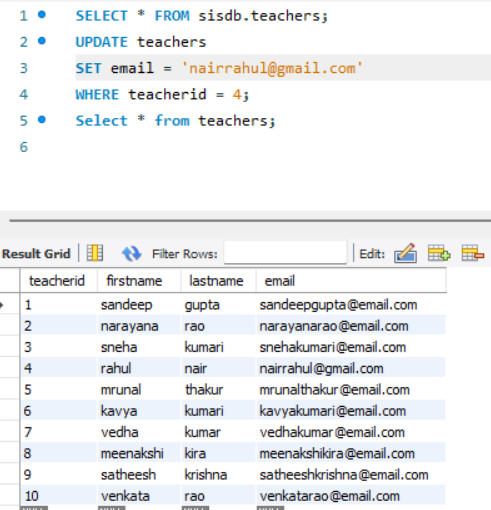
3) Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.

UPDATE teachers

SET email = ‘nairrahul@gmail.com,

WHERE teacherid = (4);

Select \* from teachers;



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 studentid = (SELECT id FROM students WHERE firstname = 'Rahul' AND lastname = 'nair')

AND courseid = (SELECT id FROM courses WHERE coursename = 'YourCourseName');

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

UPDATE courses

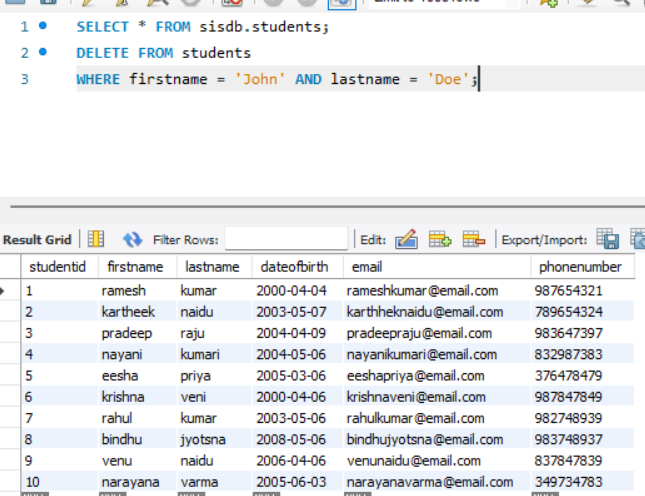
SET teacherid = (SELECT id FROM teachers WHERE name = 'TeacherName')

WHERE courseid = (SELECT id FROM courses WHERE coursename = 'CourseName');

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 students

WHERE firstname = 'John' AND lastname = 'Doe';



TASK 3

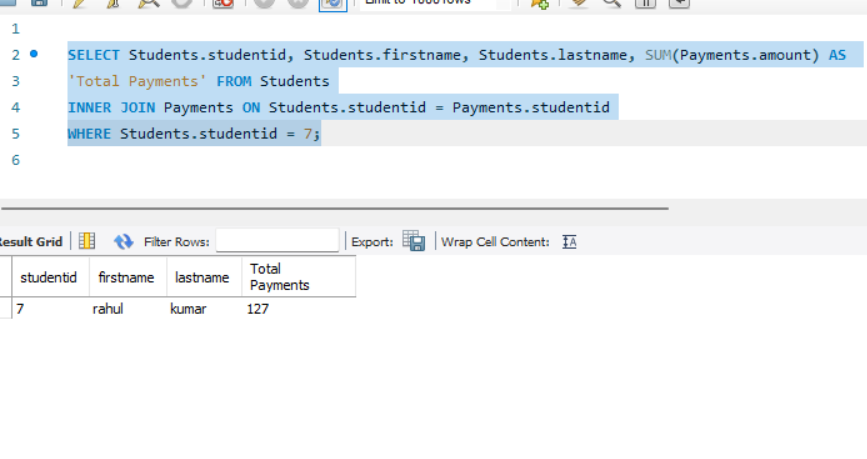
1) Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.

SELECT Students.student\_id, Students.firstname, Students.lastname, SUM(Payments.amount) AS

'Total Payments' FROM Students

INNER JOIN Payments ON Students.studentid = Payments.studentid

WHERE Students.studentid = 7;



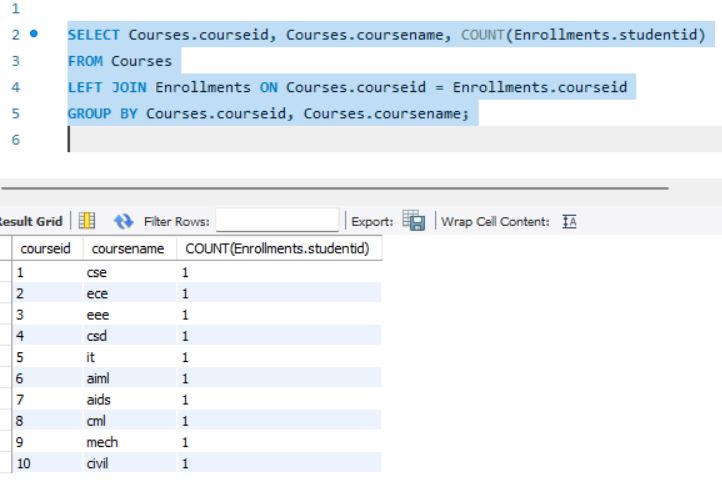
2) Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

SELECT Courses.courseid, Courses.coursename, COUNT(Enrollments.studentid)

FROM Courses

LEFT JOIN Enrollments ON Courses.courseid = Enrolments.courseid

GROUP BY Courses.courseid, Courses.coursename;



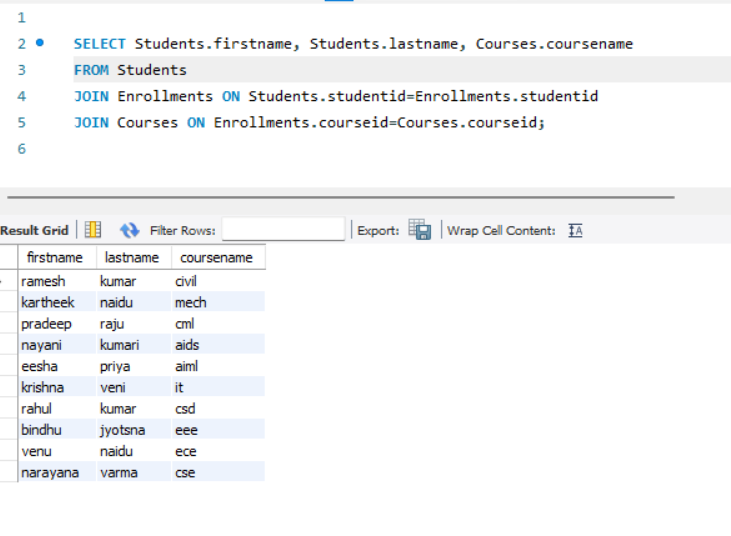
4) Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.

SELECT Students.firstname, Students.lastname, Courses.coursename

FROM Students

JOIN Enrollments ON Students.studentid=Enrollments.studentid

JOIN Courses ON Enrollments.courseid=Courses.courseid;



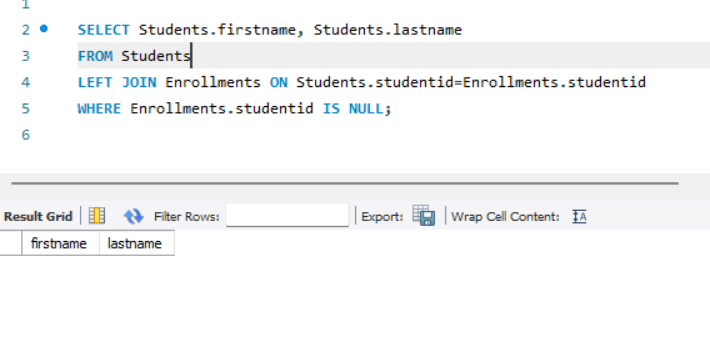
3) Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.

SELECT Students.firstname, Students.lastname

FROM Students

LEFT JOIN Enrollments ON Students.studentid=Enrollments.studentid

WHERE Enrollments.studentid IS NULL;



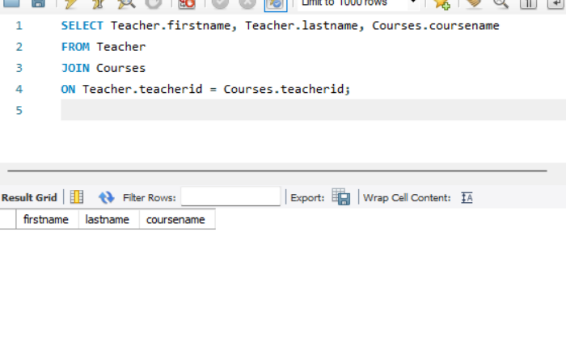
5) Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.

SELECT Teacher.firstname, Teacher.lastname, Courses.coursename

FROM Teacher

JOIN Courses

ON Teacher.teacherid = Courses.teacherid;



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.

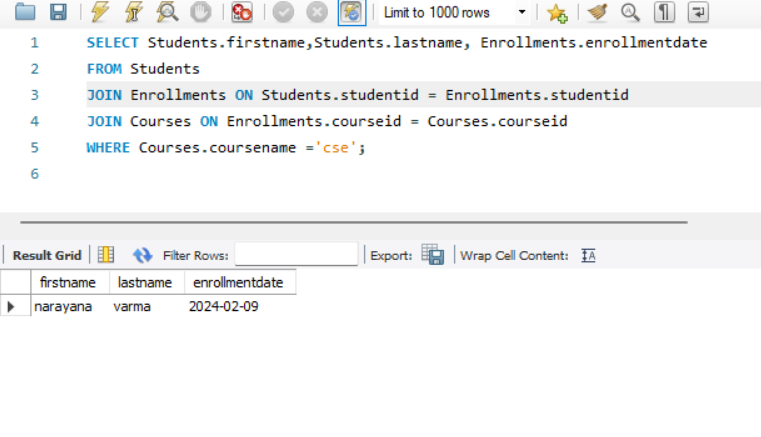
SELECT Students.firstname,Students.lastname, Enrollments.enrollmentdate

FROM Students

JOIN Enrollments ON Students.studentid = Enrollments.studentid

JOIN Courses ON Enrollments.courseid = Courses.courseid

WHERE Courses.coursename ='cse';



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.

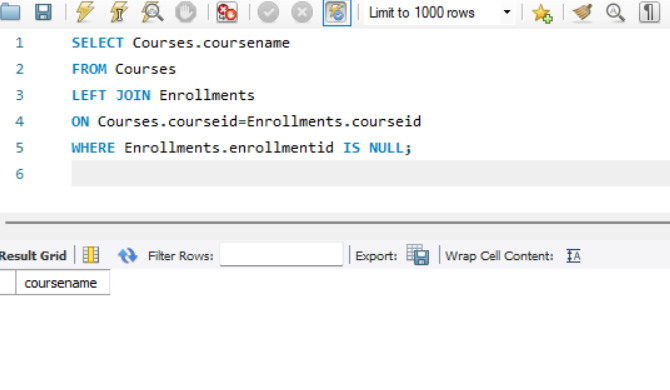
SELECT Courses.coursename

FROM Courses

LEFT JOIN Enrollments

ON Courses.courseid=Enrollments.courseid

WHERE Enrollments.enrollmentid 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.

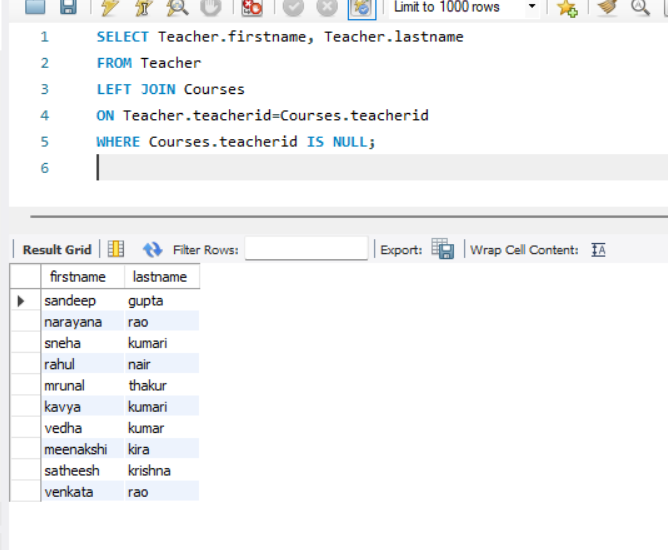
SELECT Teacher.firstname, Teacher.lastname

FROM Teacher

LEFT JOIN Courses

ON Teacher.teacherid=Courses.teacherid

WHERE Courses.teacherid 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 \* FROM sisdb.courses;

SELECT courseid, AVG(studentcount)

FROM (

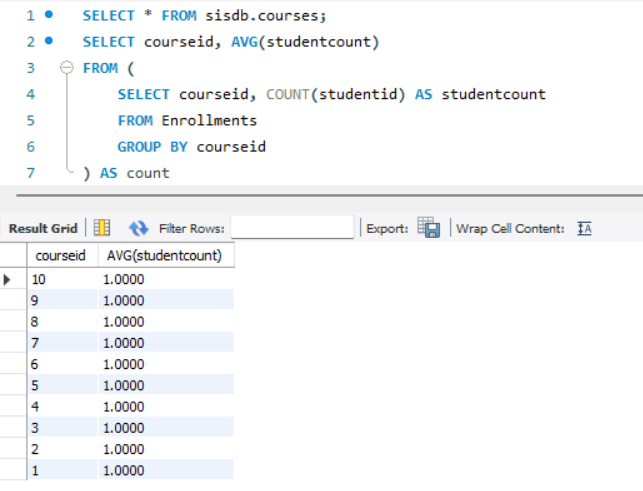
SELECT courseid, COUNT(studentid) AS studentcount

FROM Enrollments

GROUP BY courseid

) AS count

GROUP BY courseid;



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.

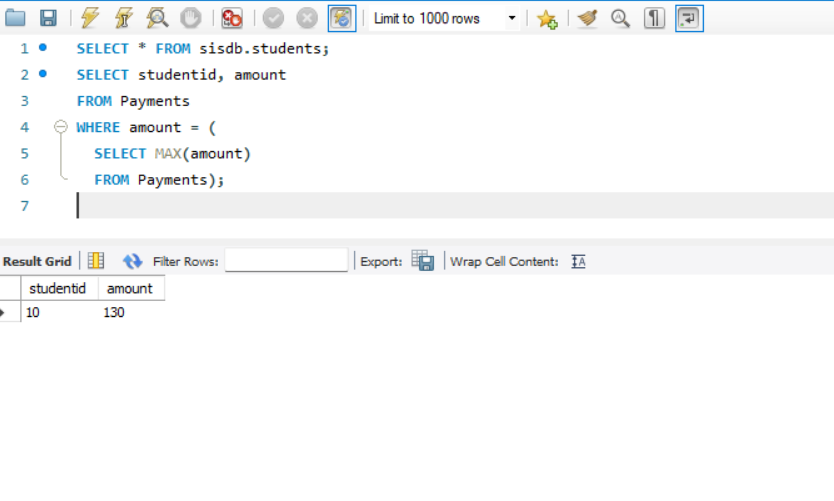
SELECT studentid, amount

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 courseid, enrollmentcount

FROM (

SELECT courseid, COUNT(studentid) AS enrollmentcount

FROM enrollments

GROUP BY courseid

) AS courseenrollments

WHERE enrollmentcount = (

SELECT MAX(enrollmentcount)

FROM (

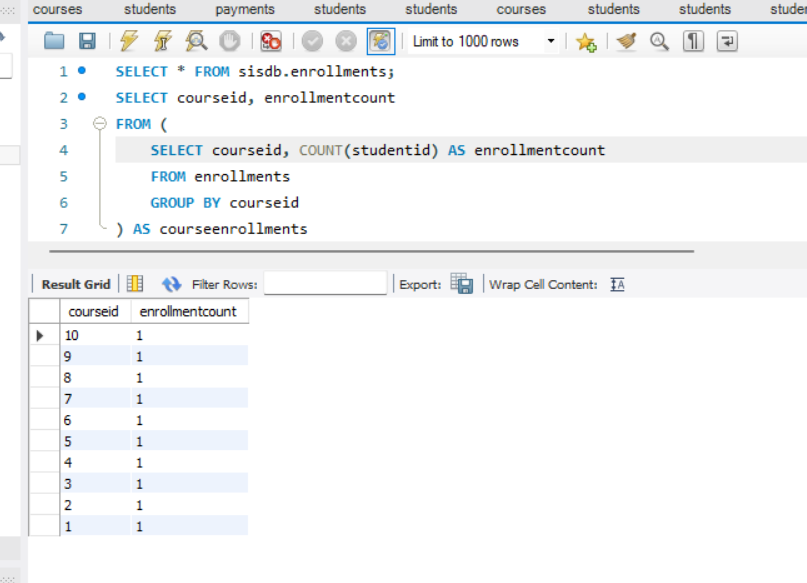
SELECT COUNT(studentid) AS enrollmentcount

FROM enrollments

GROUP BY courseid

) AS maxenrollment

);



5) SELECT \* FROM sisdb.students;

SELECT studentid, COUNT(courseid) AS enroll

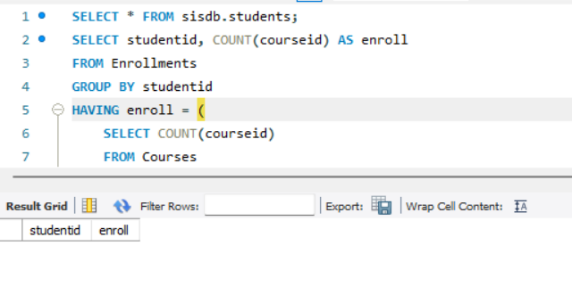
FROM Enrollments

GROUP BY studentid

HAVING enroll = (

SELECT COUNT(courseid)

FROM Courses

);

6) SELECT \* FROM sisdb.teacher;

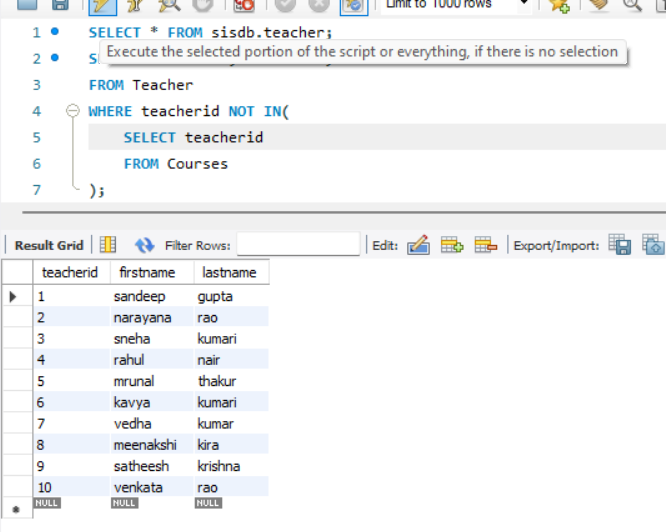
SELECT teacherid, firstname, lastname

FROM Teacher

WHERE teacherid NOT IN(

SELECT teacherid

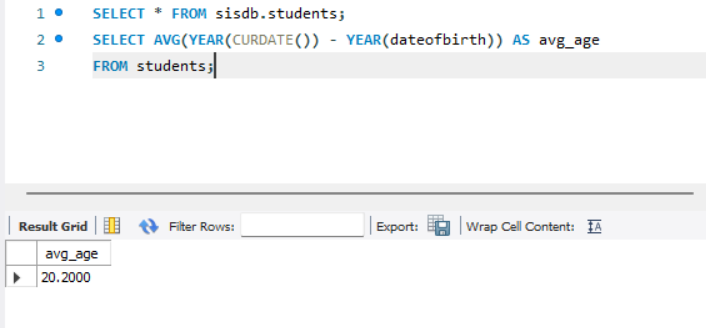
FROM Courses);



7) SELECT \* FROM sisdb.students;

SELECT AVG(YEAR(CURDATE()) - YEAR(dateofbirth)) AS avg\_age

FROM students;

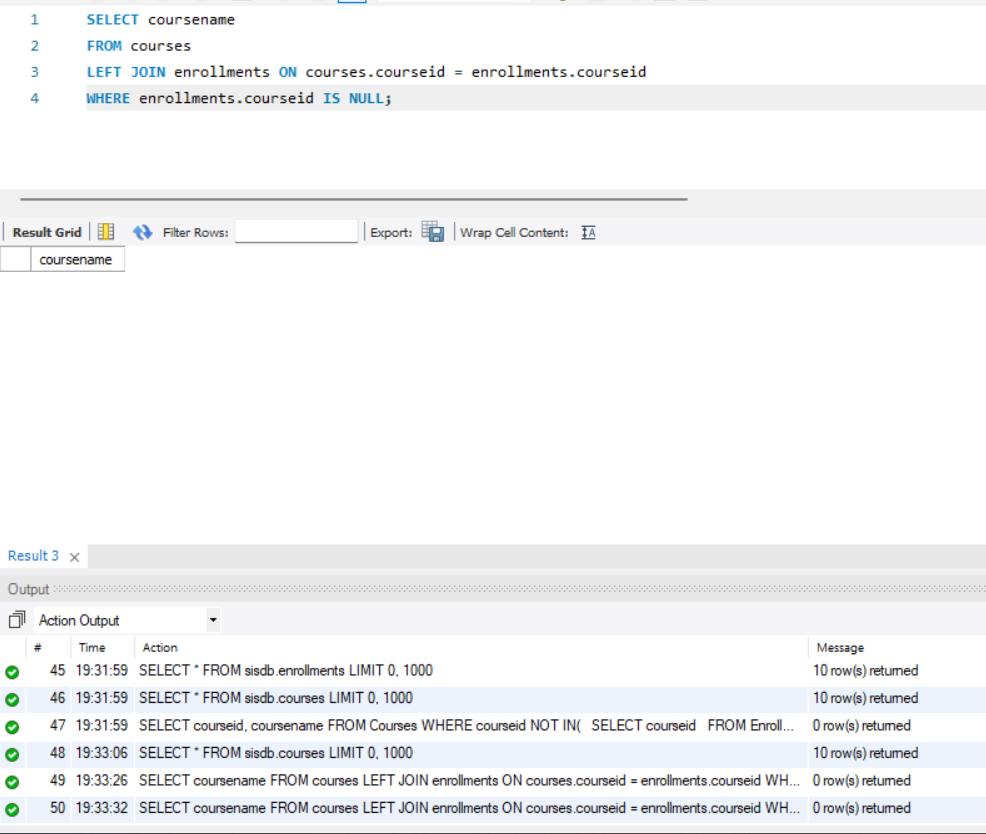


8) SELECT course\_name

FROM courses

LEFT JOIN enrollments ON courses.course\_id = enrollments.course\_id

WHERE enrollments.course\_id IS NULL;



9) SELECT studentid, courseid, SUM(paymentamount) AS totalpayments

FROM payments

WHERE studentid IN (

SELECT DISTINCT studentid

FROM enrollments

)

GROUP BY studentid, courseid;

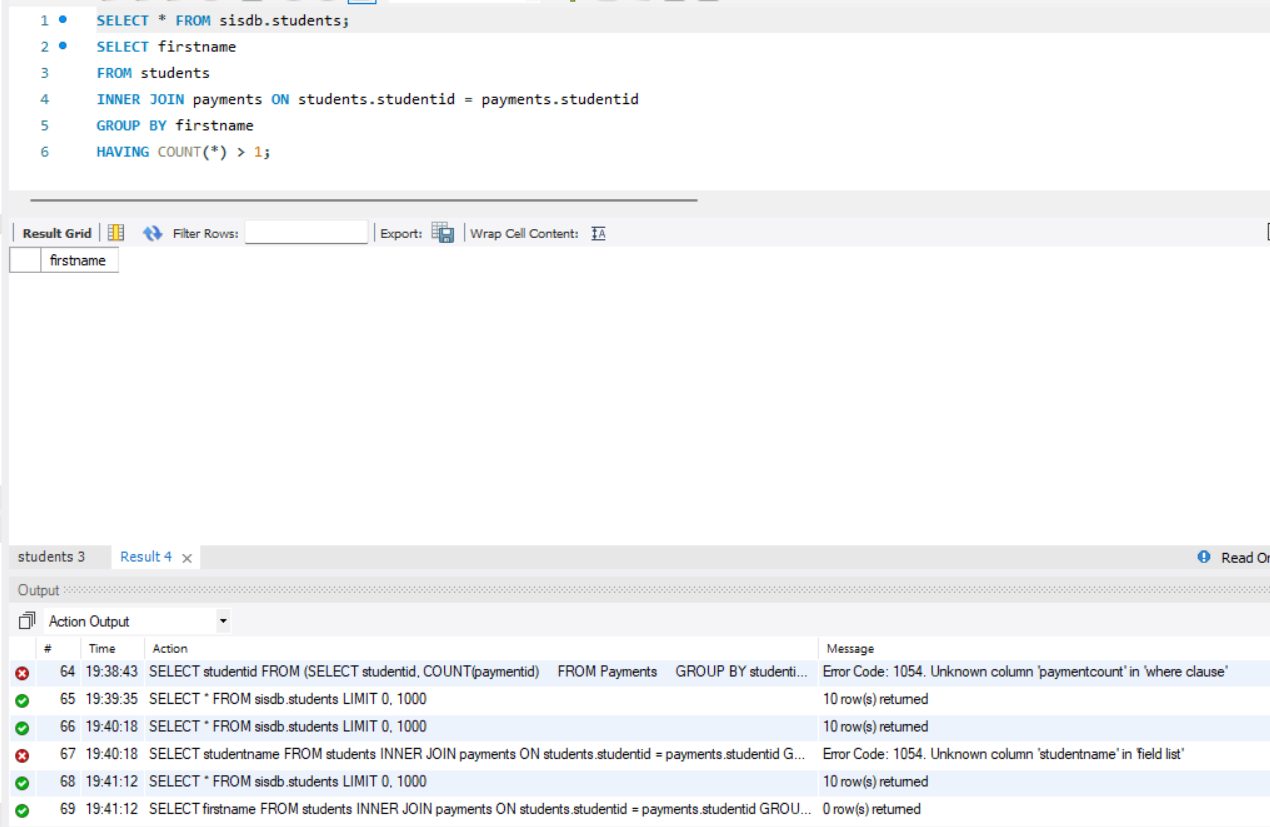
10) SELECT \* FROM sisdb.students;

SELECT firstname

FROM students

INNER JOIN payments ON students.studentid = payments.studentid

GROUP BY firstname HAVING COUNT(\*) > 1;



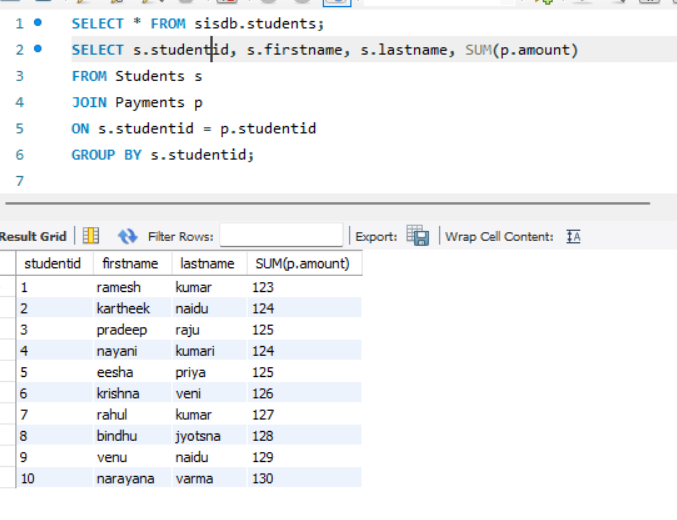
11) SELECT s.studentid, s.firstname, s.lastname, SUM(p.amount)

FROM Students s

JOIN Payments p

ON s.studentid = p.studentid

GROUP BY s.studentid;

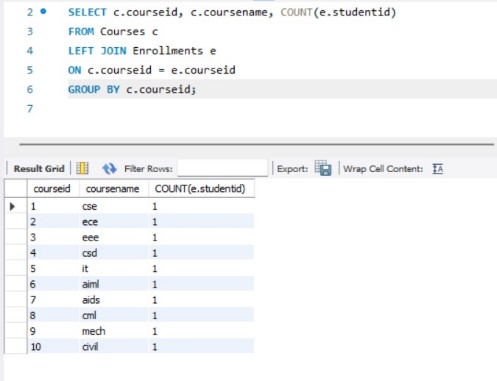


12)SELECT c.courseid, c.coursename, COUNT(e.studentid)

FROM Courses c

LEFT JOIN Enrollments e

ON c.courseid = e.courseid GROUP BY c.courseid;



13) SELECT s.studentid, s.firstname, s.lastname, AVG(p.amount)

FROM Students s

JOIN Payments p

ON s.studentid = p.studentid

GROUP BY s.studentid;

