TASK-1

Create database SISDB;

CREATE TABLE students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(255),

last\_name VARCHAR(255),

date\_of\_birth VARCHAR(20),

email VARCHAR(255),

phone\_number VARCHAR(20)

);

CREATE TABLE teachers (

teacher\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(255) NOT NULL,

credits INT,

teacher\_id INT,

FOREIGN KEY (teacher\_id) REFERENCES teachers(teacher\_id)

);

CREATE TABLE enrollments (

enrollment\_id INT PRIMARY KEY,

student\_id INT,

course\_id INT,

enrollment\_date VARCHAR(20),

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (course\_id) REFERENCES courses(course\_id)

);

CREATE TABLE payments (

payment\_id INT PRIMARY KEY,

student\_id INT,

amount DECIMAL(10, 2) NOT NULL,

payment\_date VARCHAR(20),

FOREIGN KEY (student\_id) REFERENCES students(student\_id)

);

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

VALUES

(1, 'David', 'Robinson', '1998-08-12', 'david.robinson@example.com', '777-888-9999'),

(2, 'Alice', 'Smith', '1998-02-22', 'alice.smith@example.com', '9876543210'),

(3, 'Bob', 'Johnson', '1997-09-10', 'bob.johnson@example.com', '5551234567'),

(4, 'Eva', 'Williams', '1996-12-03', 'eva.williams@example.com', '3338884444'),

(5, 'Michael', 'Brown', '1999-07-18', 'michael.brown@example.com', '7772221111'),

(6, 'Sophia', 'Taylor', '1994-03-25', 'sophia.taylor@example.com', '1113339999'),

(7, 'Daniel', 'Miller', '1998-11-08', 'daniel.miller@example.com', '9997775555'),

(8, 'Olivia', 'Davis', '1995-06-30', 'olivia.davis@example.com', '4446662222'),

(9, 'Henry', 'Moore', '1997-04-12', 'henry.moore@example.com', '6664448888'),

(10, 'Grace', 'Anderson', '1996-01-05', 'grace.anderson@example.com', '2225553333');

INSERT INTO teacher (teacher\_id, first\_name, last\_name, email)

VALUES

(1, 'Emily', 'Smith', 'emily.smith@example.com'),

(2, 'Daniel', 'Johnson', 'daniel.johnson@example.com'),

(3, 'Sophie', 'Williams', 'sophie.williams@example.com'),

(4, 'Michael', 'Brown', 'michael.brown@example.com'),

(5, 'Olivia', 'Jones', 'olivia.jones@example.com'),

(6, 'William', 'Miller', 'william.miller@example.com'),

(7, 'Emma', 'Davis', 'emma.davis@example.com'),

(8, 'Alexander', 'Moore', 'alexander.moore@example.com'),

(9, 'Grace', 'Anderson', 'grace.anderson@example.com'),

(10, 'Ethan', 'Taylor', 'ethan.taylor@example.com');

INSERT INTO courses (course\_id, course\_name, credits, teacher\_id)

VALUES

(1, 'Mathematics 101', 3, 1),

(2, 'Computer Science Fundamentals', 4, 2),

(3, 'English Literature', 3, 3),

(4, 'Physics for Beginners', 4, 4),

(5, 'History of Art', 3, 5),

(6, 'Chemistry Basics', 4, 6),

(7, 'Introduction to Psychology', 3, 7),

(8, 'Environmental Science', 4, 8),

(9, 'Music Theory', 3, 9),

(10, 'Spanish Language', 4, 10);

INSERT INTO enrollments (enrollment\_id, student\_id, course\_id, enrollment\_date)

VALUES

(1, 1, 1, '2023-01-15'),

(2, 2, 2, '2023-02-22'),

(3, 3, 3, '2023-03-10'),

(4, 4, 4, '2023-04-03'),

(5, 5, 5, '2023-05-18'),

(6, 6, 6, '2023-06-25'),

(7, 7, 7, '2023-07-08'),

(8, 8, 8, '2023-08-30'),

(9, 9, 9, '2023-09-12'),

(10, 10, 10, '2023-10-05');

INSERT INTO payments (payment\_id, student\_id, amount, payment\_date)

VALUES

(1, 1, 500.00, '2023-01-15'),

(2, 2, 600.00, '2023-02-22'),

(3, 3, 450.00, '2023-03-10'),

(4, 4, 700.00, '2023-04-03'),

(5, 5, 550.00, '2023-05-18'),

(6, 6, 800.00, '2023-06-25'),

(7, 7, 400.00, '2023-07-08'),

(8, 8, 750.00, '2023-08-30'),

(9, 9, 600.00, '2023-09-12'),

(10, 10, 550.00, '2023-10-05');

TASK-2

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

(11, 'John', 'Doe', '1995-05-15', 'john.doe@example.com', '1234567890');

INSERT INTO enrollments ( enrollment\_id, student\_id, course\_id, enrollment\_date)

VALUES (11, 1, 1, '2024-01-09');

UPDATE ‘teacher’ SET ‘email’ = ‘moore.alex@example.com’ WHERE (‘teacher\_id’ = ‘8’);

DELETE FROM enrollments WHERE student\_id = 2 AND course\_id = 2;

UPDATE courses SET teacher\_id = 3 WHERE course\_id = 4;

DELETE FROM enrollments WHERE student\_id = 9;

DELETE FROM payments WHERE student\_id = 9;

DELETE FROM students WHERE student\_id = 9;

UPDATE payments SET amount = '900.00' WHERE (payment\_id = 7);

TASK-3

SELECT

s.student\_id,

s.first\_name,

s.last\_name,

SUM(p.amount) AS total\_payments

FROM

students s

JOIN

payments p ON s.student\_id = p.student\_id

WHERE

s.student\_id = 5 ;

SELECT

c.course\_id,

c.course\_name,

COUNT(e.student\_id) AS enrolled\_students\_count

FROM

courses c

INNER JOIN

enrollments e ON c.course\_id = e.course\_id

GROUP BY

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

SELECT

s.student\_id,

s.first\_name,

s.last\_name

FROM

students s

LEFT JOIN

enrollments e ON s.student\_id = e.student\_id

WHERE

e.student\_id IS NULL;

SELECT

s.first\_name,

s.last\_name,

c.course\_name

FROM

students s

JOIN

enrollments e ON s.student\_id = e.student\_id

JOIN

courses c ON e.course\_id = c.course\_id;

SELECT

t.first\_name AS teacher\_first\_name,

t.last\_name AS teacher\_last\_name,

c.course\_name

FROM

teacher t

JOIN

courses c ON t.teacher\_id = c.teacher\_id;

SELECT

Students.student\_id,

Students.first\_name as student\_first\_name,

students.last\_name as student\_last\_name,

Enrollments.enrollment\_date

FROM

Students

JOIN

Enrollments ON Students.student\_id = Enrollments.student\_id

JOIN

Courses ON Enrollments.course\_id = Courses.course\_id

WHERE

courses.course\_id = 6;

SELECT

Students.student\_id,

Students.first\_name as student\_first\_name,

students.last\_name as student\_last\_name

FROM

Students

LEFT JOIN

Payments ON Students.student\_id = Payments.student\_id

WHERE

Payments.payment\_id IS NULL;

SELECT

courses.course\_id,

courses.course\_name

FROM

courses

LEFT JOIN

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

WHERE

Enrollments.enrollment\_id IS NULL;

SELECT

E1.student\_id,

S.first\_name as Student\_first\_name,

S.last\_name as Students\_last\_name,

COUNT(DISTINCT E1.course\_id) AS enrolled\_courses\_count

FROM

Enrollments E1

JOIN

Students S ON E1.student\_id = S.student\_id

GROUP BY

E1.student\_id, S.first\_name, S.last\_name

HAVING

COUNT(DISTINCT E1.course\_id) > 1;

SELECT

T.teacher\_id,

T.first\_name,

T.last\_name,

c.course\_name

FROM

Teacher T

LEFT JOIN

Courses C ON T.teacher\_id = C.teacher\_id

WHERE

C.course\_id IS NULL;

TASK-4

1)

SELECT

course\_id,

AVG(num\_students) AS average\_students\_enrolled

FROM (

SELECT

course\_id,

COUNT(student\_id) AS num\_students

FROM

enrollments

GROUP BY

course\_id, student\_id

) AS subquery

GROUP BY

course\_id;

2)

SELECT

student\_id,

amount as highest\_payment

FROM

payments

WHERE

amount = (

SELECT

MAX(amount)

FROM

payments

);

3)

SELECT

c.course\_id,

c.course\_name,

enrollment\_count

FROM

courses c

JOIN

(

SELECT

course\_id,

COUNT(DISTINCT student\_id) AS enrollment\_count

FROM

enrollments

GROUP BY

course\_id

) AS subquery

ON

c.course\_id = subquery.course\_id

WHERE

enrollment\_count = (

SELECT

MAX(enrollment\_count)

FROM

(

SELECT

COUNT(DISTINCT student\_id) AS enrollment\_count

FROM

enrollments

GROUP BY

course\_id

) AS max\_enrollment

);

4)

SELECT t.teacher\_id, t.first\_name AS teacher\_first\_name,

t.last\_name AS teacher\_last\_name,

COALESCE(SUM(p.amount), 0) AS total\_payments

FROM teacher t

LEFT JOIN Courses c ON t.teacher\_id = c.teacher\_id

LEFT JOIN Enrollments e ON c.course\_id = e.course\_id

LEFT JOIN Payments p ON e.student\_id = p.student\_id

GROUP BY t.teacher\_id, t.first\_name ;

5)\_

SELECT s.student\_id , s.first\_name , s.last\_name FROM students s

WHERE(SELECT COUNT(DISTINCT course\_id) FROM courses)

=(SELECT COUNT(DISTINCT course\_id) FROM enrollments e

WHERE e.student\_id =s.student\_id);

6)

SELECT t.teacher\_id,t.first\_name,t.last\_name

FROM teacher t

WHERE NOT EXISTS (SELECT 1 FROM courses c

WHERE c.teacher\_id =t.teacher\_id);

7)

SELECT AVG(age) AS average\_age FROM

(SELECT TIMESTAMPDIFF(YEAR, date\_of\_birth, CURDATE()) AS age

FROM students) AS subquery;

8)

SELECT c.course\_id,c.course\_name

FROM courses c WHERE

NOT EXISTS (SELECT 1 FROM enrollments e

WHERE e.course\_id=c.course\_id);

9)

SELECT e.student\_id, e.course\_id, COALESCE(SUM(p.amount), 0) AS total\_payments

FROM Enrollments e

LEFT JOIN Payments p ON e.student\_id = p.student\_id

GROUP BY e.student\_id, e.course\_id;

10)

SELECT s.student\_id,s.first\_name,s.last\_name

FROM students s JOIN payments p ON s.student\_id = p.student\_id

GROUP BY s.student\_id, s.first\_name, s.last\_name

HAVING COUNT(p.payment\_id)>1;

11)

SELECT s.student\_id,s.first\_name,s.last\_name,

COALESCE(SUM(p.amount), 0) AS total\_payments

FROM students s

LEFT JOIN payments p ON s.student\_id = p.student\_id

GROUP BY s.student\_id, s.first\_name;

12)

SELECT

c.course\_id,

c.course\_name,

COUNT(e.student\_id) AS enrolled\_students

FROM

courses c

LEFT JOIN

enrollments e ON c.course\_id = e.course\_id

GROUP BY

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

13)

SELECT s.student\_id,s.first\_name,s.last\_name,

AVG(p.amount) AS average\_payment\_amount

FROM students s LEFT JOIN

payments p ON s.student\_id = p.student\_id

GROUP BY s.student\_id, s.first\_name, s.last\_name;