Assignment - 2 :

Task-1:

1. Create the database named "SISDB".

A. create database SISDB;

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

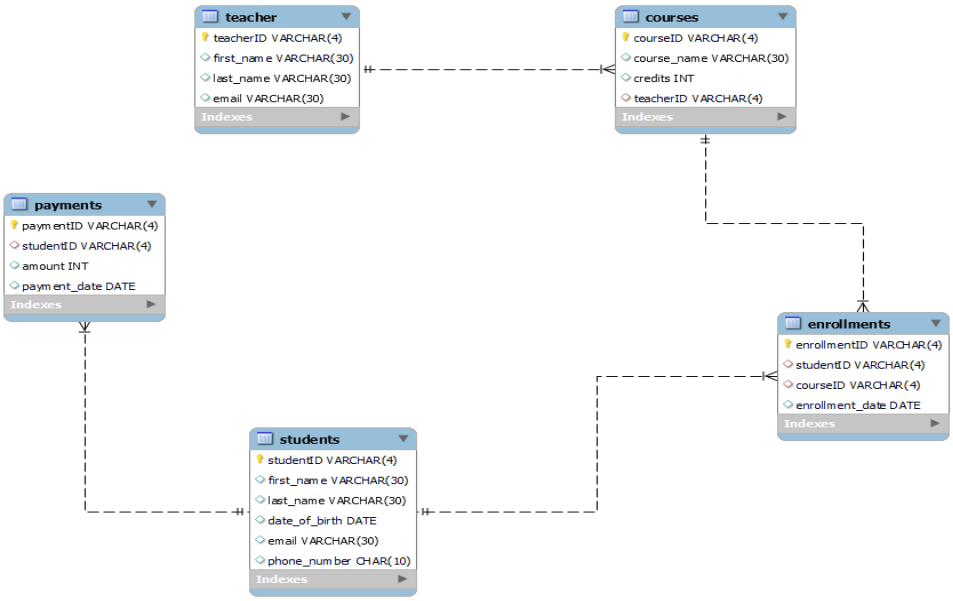
A.

```
create table Students(
    studentID varchar(4) primary key,
    first_name varchar(30),
    last_name varchar(30),
    date_of_birth date,
    email varchar(30),
    phone_number char(10)
);

create table Courses(
    courseID varchar(4) primary key,
    course_name varchar(30),
    credits int,
    teacherID varchar(4),
    foreign key(teacherID) references Teacher(teacherID)
);

create table Enrollments(
    enrollmentID varchar(4) primary key,
    studentID varchar(4),
    courseID varchar(4),
    enrollment_date date,
    foreign key(studentID) references Students(studentID),
    foreign key(courseID) references Courses(courseID)
);

create table Teacher(
    teacherID varchar(4) primary key,
    first_name varchar(30),
    last_name varchar(30),
    email varchar(30)
);
```

```
create table Payments(
    paymentID varchar(4) primary key,
    studentID varchar(4),
    amount int,
    payment_date date,
    foreign key(studentID) references Students(studentID)
);
```

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

A.



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

A.

```
-- Students table
INSERT INTO Students VALUES
('S001', 'John', 'Doe', '1998-05-15', 'john.doe@email.com', '1234567890'),
('S002', 'Alice', 'Smith', '1999-03-22', 'alice.smith@email.com', '9876543210'),
('S003', 'Bob', 'Johnson', '2000-08-10', 'bob.johnson@email.com', '5551112222'),
('S004', 'Emma', 'Williams', '1997-12-05', 'emma.williams@email.com', '6663339999'),
('S005', 'Michael', 'Brown', '1996-06-30', 'michael.brown@email.com', '4447778888'),
('S006', 'Olivia', 'Miller', '1998-11-18', 'olivia.miller@email.com', '2224446666'),
('S007', 'Daniel', 'Taylor', '2001-02-25', 'daniel.taylor@email.com', '9990001111'),
('S008', 'Sophia', 'Johnson', '1999-07-12', 'sophia.johnson@email.com', '8885552222'),
('S009', 'Ethan', 'Wilson', '1997-09-08', 'ethan.wilson@email.com', '7772224444'),
('S010', 'Ava', 'Davis', '2000-04-14', 'ava.davis@email.com', '1119998888');

-- Teacher table
INSERT INTO Teacher VALUES
('T001', 'Professor', 'Smith', 'prof.smith@email.com'),
('T002', 'Dr.', 'Jones', 'dr.jones@email.com'),
('T003', 'Ms.', 'Williams', 'ms.williams@email.com'),
('T004', 'Mr.', 'Brown', 'mr.brown@email.com'),
('T005', 'Professor', 'Miller', 'prof.miller@email.com'),
('T006', 'Dr.', 'Taylor', 'dr.taylor@email.com'),
('T007', 'Ms.', 'Davis', 'ms.davis@email.com'),
('T008', 'Mr.', 'Wilson', 'mr.wilson@email.com'),
('T009', 'Professor', 'Anderson', 'prof.anderson@email.com'),
('T010', 'Dr.', 'Moore', 'dr.moore@email.com');
```

```
-- Courses table
INSERT INTO Courses VALUES
('C001', 'Introduction to Programming', 3, 'T001'),
('C002', 'Database Management', 4, 'T002'),
('C003', 'Data Structures', 3, 'T003'),
('C004', 'Web Development', 4, 'T004'),
('C005', 'Machine Learning', 5, 'T005'),
('C006', 'Software Engineering', 4, 'T006'),
('C007', 'Computer Networks', 3, 'T007'),
('C008', 'Artificial Intelligence', 5, 'T008'),
('C009', 'Cybersecurity', 4, 'T009'),
('C010', 'Mobile App Development', 3, 'T010');

-- Enrollments table
INSERT INTO Enrollments VALUES
('E001', 'S001', 'C001', '2023-01-15'),
('E002', 'S002', 'C003', '2023-02-20'),
('E003', 'S003', 'C005', '2023-03-25'),
('E004', 'S004', 'C008', '2023-04-10'),
('E005', 'S005', 'C002', '2023-05-15'),
('E006', 'S006', 'C006', '2023-06-20'),
('E007', 'S007', 'C004', '2023-07-25'),
('E008', 'S008', 'C009', '2023-08-10'),
('E009', 'S009', 'C007', '2023-09-15'),
('E010', 'S010', 'C010', '2023-10-20');
```

```sql
-- Payments table
INSERT INTO Payments VALUES
('P001', 'S001', 500, '2023-01-25'),
('P002', 'S002', 600, '2023-02-28'),
('P003', 'S003', 750, '2023-03-31'),
('P004', 'S004', 900, '2023-04-30'),
('P005', 'S005', 550, '2023-05-31'),
('P006', 'S006', 700, '2023-06-30'),
('P007', 'S007', 800, '2023-07-31'),
('P008', 'S008', 950, '2023-08-31'),
('P009', 'S009', 600, '2023-09-30'),
('P010', 'S010', 700, '2023-10-31');
```

```
6  19:00:12  INSERT INTO Students VALUES ('S001', 'John', 'Doe', '1998-05-15', 'john.doe@email.com', '1234567890'), ('S00...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0  0.000 sec
7  19:00:12  INSERT INTO Teacher VALUES ('T001', 'Professor', 'Smith', 'prof.smith@email.com'), ('T002', 'Dr.', 'Jones', 'dr.jon...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0  0.000 sec
8  19:00:12  INSERT INTO Courses VALUES ('C001', 'Introduction to Programming', 3, 'T001'), ('C002', 'Database Manageme...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0  0.000 sec
9  19:00:12  INSERT INTO Enrollments VALUES ('E001', 'S001', 'C001', '2023-01-15'), ('E002', 'S002', 'C003', '2023-02-20'), ('...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0  0.000 sec
10  19:00:12  INSERT INTO Payments VALUES ('P001', 'S001', 500, '2023-01-25'), ('P002', 'S002', 600, '2023-02-28'), ('P003', '...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0  0.000 sec
```

Task-2:

1.

```sql
/*
Write an SQL query to insert a new student into the "Students" table with the following details:
a. First Name: John
b. Last Name: Harbour
c. Date of Birth: 1995-08-15
d. Email: john.doe@example.com
e. Phone Number: 1234567890
*/
insert into students
values('S011', 'John', 'Harbour', '1998-05-15', 'john.hr@email.com', '9811267890')
```

```
12  19:05:44  insert into students values('S011', 'John', 'Harbour', '1998-05-15', 'john.hr@email.com', '9811267890')  1 row(s) affected  0.000 sec
```

2.

```sql
-- 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.
select * from enrollments;
Insert into enrollments
values('E011', 'S004', 'C007', '2023-11-11');
```

```
1  19:10:24  Insert into enrollments values('E011', 'S004', 'C007', '2023-11-11')  1 row(s) affected  0.000 sec
```

3.

```sql
-- Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.
Update Teacher
Set email = 'gregbrown@gmail.com'
where teacherID = 'T004';
```

```
2  19:14:15  Update Teacher  Set email = 'gregbrown@gmail.com'  where teacherID = 'T004'  1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0  0.000 sec
```

4.

```sql
-- Write an SQL query to delete a specific enrollment record from the "Enrollments" table.
-- Select an enrollment record based on the student and course.
 select * from enrollments;
 delete from enrollments
 where studentID = 'S004' and courseID = 'C008';
```

```
7  19:17:22  delete from enrollments  where studentID = 'S004' and courseID = 'C008'   1 row(s) affected   0.000 sec
```

5.

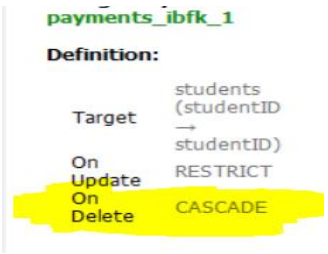
```sql
-- Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.
 select * from courses;
 Update courses
 set teacherID = 'T003'
 where courseID = 'C005';
```

```
10  19:26:21  Update courses  set teacherID = 'T003'  where courseID = 'C005'   1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0   0.000 sec
```

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.

```sql
-- To maintain the referential integrity, we can add ON_DELETE_CASCADE to the foreign key constraints on the children tables.
 select * from payments;
 delete from students
 where studentID = 'S002';
```

```
19  19:42:30  delete from students where studentID = 'S002'   1 row(s) affected   0.016 sec
```



7.

```sql
-- Update the payment amount for a specific payment record in the "Payments" table.
-- Choose any payment record and modify the payment amount.
select * from payments;
Update payments
set amount = 650
where paymentID = 'P003';
```

```
23  19:49:13  Update payments set amount = 650 where paymentID = 'P003'   1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0   0.000 sec
```

Task-3:

1.

```sql
-- 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 concat(s.first_Name,' ', s.last_Name) as `Student Name`, count(p.studentID) as `Number of Payments`
from students s
join payments p on s.studentID = p.studentID
where p.studentID = 'S001'
group by p.studentID;
```

	Student Name	Number of Payments
▶	John Doe	1

2.

```sql
-- 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.courseID, c.course_Name, count(e.courseID) as `Total Enrollments`
from courses c
join enrollments e on c.courseID = e.courseID
group by e.courseID;
```

	courseID	course_Name	Total Enrollments
▶	C001	Introduction to Programming	1
	C002	Database Management	1
	C004	Web Development	1
	C005	Machine Learning	1
	C006	Software Engineering	1
	C007	Computer Networks	2
	C009	Cybersecurity	1
	C010	Mobile App Development	1

3.

```sql
-- 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.*
from students s
left join enrollments e on s.studentID = e.studentID
where e.studentID IS NULL;
```

	studentID	first_name	last_name	date_of_birth	email	phone_number
▶	S011	John	Harbour	1998-05-15	john.hr@email.com	9811267890
	S012	Mary	Jane	2000-04-15	mjane@email.com	9847292018

4.

```sql
-- 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 as `First Name`, s.last_name as `Last Name`, c.course_name as `Course Name`
from students s
join enrollments e on s.studentId = e.studentID
join courses c on e.courseId = c.courseID;
```

First Name	Last Name	Course Name
John	Doe	Introduction to Programming
Bob	Johnson	Machine Learning
Michael	Brown	Database Management
Olivia	Miller	Software Engineering
Daniel	Taylor	Web Development
Sophia	Johnson	Cybersecurity
Ethan	Wilson	Computer Networks
Ava	Davis	Mobile App Development
Emma	Williams	Computer Networks

5.

```sql
-- 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 concat(t.first_Name,' ', t.last_Name) as `Teacher Name`, c.course_name as `Assigned Courses`
from teacher t
join courses c on t.teacherID = c.teacherID;
```

Teacher Name	Assigned Courses
Professor Smith	Introduction to Programming
Dr. Jones	Database Management
Ms. Williams	Data Structures
Ms. Williams	Machine Learning
Mr. Brown	Web Development
Dr. Taylor	Software Engineering
Ms. Davis	Computer Networks
Mr. Wilson	Artificial Intelligence
Professor Anderson	Cybersecurity
Dr. Moore	Mobile App Development

6.

```sql
-- 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 s.*
from students s
join enrollments e on s.studentId = e.studentID
where e.enrollment_date = '2023-05-15';
```

studentID	first_name	last_name	date_of_birth	email	phone_number
S005	Michael	Brown	1996-06-30	michael.brown@email.com	4447778888

7.

```sql
-- 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 s.*
from students s
left join payments p on s.studentID = p.studentID
where p.studentID is null;
```

studentID	first_name	last_name	date_of_birth	email	phone_number
S011	John	Harbour	1998-05-15	john.hr@email.com	9811267890
S012	Mary	Jane	2000-04-15	mjane@email.com	9847292018

8.

```sql
-- 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.*
from courses c
left join enrollments e on c.courseID = e.courseID
where e.courseID is null;
```

courseID	course_name	credits	teacherID
C003	Data Structures	3	T003
C008	Artificial Intelligence	5	T008

9.

```sql
-- 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.StudentID
from Enrollments e1
join Enrollments e2 on e1.StudentID = e2.StudentID
                    and e1.CourseID <> e2.CourseID;
```

StudentID

10.

```sql
-- 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.*
from teacher t
left join courses c on c.teacherID = t.teacherID
where c.teacherID is null;
```

teacherID	first_name	last_name	email
T005	Professor	Miller	prof.miller@email.com

Task-4:

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 courseID, avg(NumStudents)
from (select CourseID, count(distinct StudentID) AS NumStudents
from Enrollments
group by CourseID) cte
group by courseID;
```

courseID	avg(NumStudents)
C001	1.0000
C002	1.0000
C004	1.0000
C005	1.0000
C006	1.0000
C007	2.0000
C009	1.0000
C010	1.0000

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 s.*
from payments p
join students s on s.studentID = p.studentID
group by studentID
having sum(amount) = (select max(a_payment)
                      from (select studentID, sum(amount) as a_payment
                          from payments
                          group by studentID) a);
```

studentID	first_name	last_name	date_of_birth	email	phone_number
S006	Olivia	Miller	1998-11-18	olivia.miller@email.com	2224446666

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, count(courseID)
from enrollments
group by courseID
having count(courseID) = (
             select max(enrollment_count)
             from (select count(courseID) as enrollment_count
                 from enrollments
                 group by courseID) enroll_count) ;
```

courseID	count(courseID)
C007	2

4.

```
-- Calculate the total payments made to courses taught by each teacher.
-- Use subqueries to sum payments for each teacher's courses.
select c.teacherID, concat(t.first_name, ' ', t.last_name) as `Name`, sum(p.amount)
from teacher t
join courses c on c.teacherID = t.teacherID
join enrollments e on e.courseID = c.courseID
join payments p on e.studentID = p.studentID
group by c.teacherID;
```

teacherID	Name	sum(p.amount)
T001	Professor Smith	500
T003	Ms. Williams	650
T002	Dr. Jones	550
T006	Dr. Taylor	700
T004	Mr. Brown	800
T009	Professor Anderson	950
T007	Ms. Davis	1500
T010	Dr. Moore	700

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 studentID
from enrollments
group by studentID
having count(distinct courseID) =(select count(courseID)
                        from courses);
```

studentID

6.

```
-- Retrieve the names of teachers who have not been assigned to any courses.
-- Use subqueries to find teachers with no course assignments.
select teacherID, concat(first_name, ' ', last_name) as `Name`
from teacher
where teacherID not in (select teacherID
                    from courses
                    group by teacherID);
```

teacherID	Name
T005	Professor Miller

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(age)
from (select
    (year(curdate()) - year(date_of_birth) - (month(curdate())<month(date_of_birth))) as age
    from students) age;
```

	avg(age)
▶	24.4545

8.

```
-- Identify courses with no enrollments. Use subqueries to find courses without enrollment records.
select distinct courseID
from courses
where courseID not in (select distinct(courseID)
                       from enrollments);
```

	courseID
▶	C003
	C008
*	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 e.studentID, e.courseID, sum(amount)
from payments p
join (select studentID, courseID
    from enrollments
    group by studentID, courseID) e on e.studentID = p.studentID
group by e.studentID, e.courseID;
```

	studentID	courseID	sum(amount)
▶	S001	C001	500
	S003	C005	650
	S005	C002	550
	S006	C006	700
	S007	C004	800
	S008	C009	950
	S009	C007	600
	S010	C010	700
	S004	C007	900

10.

```sql
-- 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 st.*
from
    (select studentID, count(studentID) as counts
    from payments
    group by studentID) s
join students st on st.studentID = s.studentID
where s.counts>1;
```

studentID	first_name	last_name	date_of_birth	email	phone_number
S006	Olivia	Miller	1998-11-18	olivia.miller@email.com	2224446666

11.

```sql
-- 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.*, sum(amount) as `Total Amount Paid`
from payments p
join students s on s.studentID = p.studentID
group by p.studentID;
```

studentID	first_name	last_name	date_of_birth	email	phone_number	Total Amount Paid
S001	John	Doe	1998-05-15	john.doe@email.com	1234567890	500
S003	Bob	Johnson	2000-08-10	bob.johnson@email.com	5551112222	650
S004	Emma	Williams	1997-12-05	emma.williams@email.com	6663339999	900
S005	Michael	Brown	1996-06-30	michael.brown@email.com	4447778888	550
S006	Olivia	Miller	1998-11-18	olivia.miller@email.com	2224446666	1200
S007	Daniel	Taylor	2001-02-25	daniel.taylor@email.com	9990001111	800
S008	Sophia	Johnson	1999-07-12	sophia.johnson@email.com	8885552222	950
S009	Ethan	Wilson	1997-09-08	ethan.wilson@email.com	7772224444	600
S010	Ava	Davis	2000-04-14	ava.davis@email.com	1119998888	700

12.

```sql
-- 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 e.courseID, c.course_name, count(e.studentID)
from enrollments e
join courses c on e.courseID = c.courseID
group by e.courseID
order by e.courseID;
```

courseID	course_name	count(e.studentID)
C001	Introduction to Programming	1
C002	Database Management	1
C004	Web Development	1
C005	Machine Learning	1
C006	Software Engineering	1
C007	Computer Networks	2
C009	Cybersecurity	1
C010	Mobile App Development	1

13.

```sql
-- 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.*, round(avg(amount), 2) as `Average Amount`
from payments p
join students s on s.studentID = p.studentID
group by p.studentID;
```

studentID	first_name	last_name	date_of_birth	email	phone_number	Average Amount
S001	John	Doe	1998-05-15	john.doe@email.com	1234567890	500.00
S003	Bob	Johnson	2000-08-10	bob.johnson@email.com	5551112222	650.00
S004	Emma	Williams	1997-12-05	emma.williams@email.com	6663339999	900.00
S005	Michael	Brown	1996-06-30	michael.brown@email.com	4447778888	550.00
S006	Olivia	Miller	1998-11-18	olivia.miller@email.com	2224446666	600.00
S007	Daniel	Taylor	2001-02-25	daniel.taylor@email.com	9990001111	800.00
S008	Sophia	Johnson	1999-07-12	sophia.johnson@email.com	8885552222	950.00
S009	Ethan	Wilson	1997-09-08	ethan.wilson@email.com	7772224444	600.00
S010	Ava	Davis	2000-04-14	ava.davis@email.com	1119998888	700.00