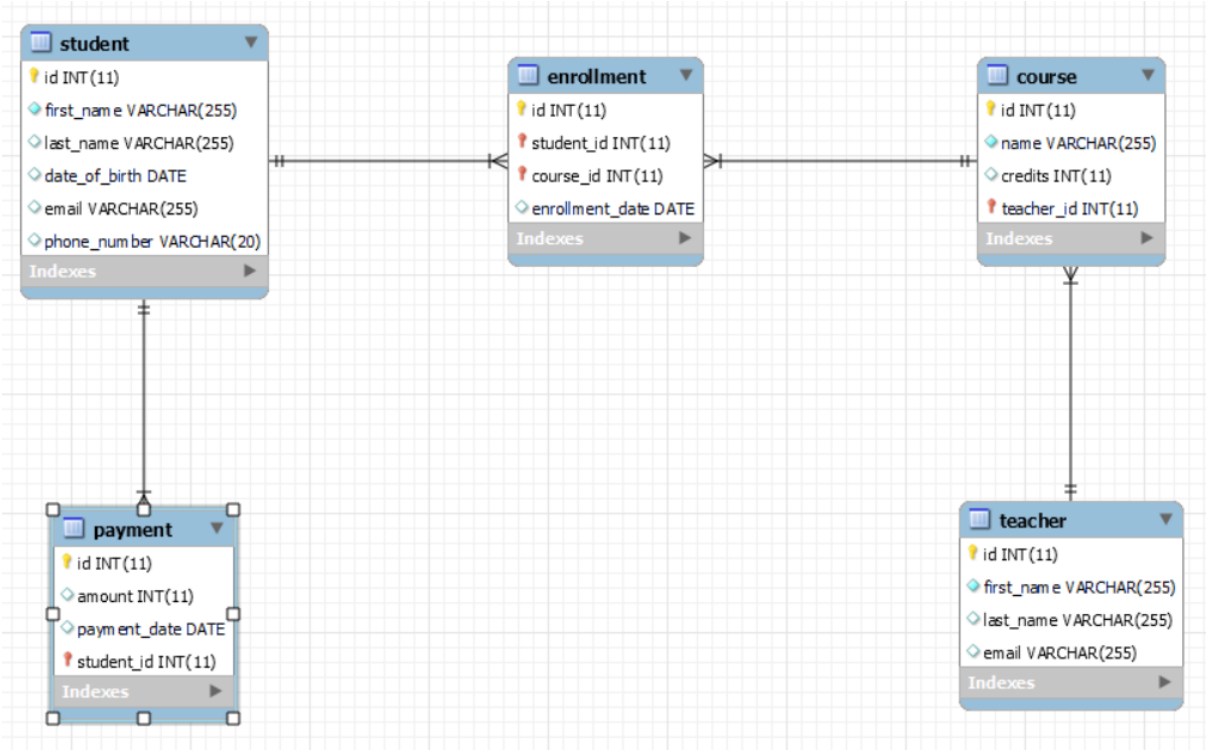
ASSIGNMENT NO : 2

Student Information System (SIS)

ER DIAGRAM:



Task:1. Database Design:

-- MySQL Workbench Forward Engineering

-- ---

-- Schema sisdb

-- ---

-- ---

-- Schema sisdb

-- ---

```
CREATE SCHEMA IF NOT EXISTS sisdb DEFAULT CHARACTER SET utf8 ;
USE sisdb ;

-- -----------------------------------------------------
-- Table sisdb.student
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS sisdb.student (
  id INT NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(255) NOT NULL,
  last_name VARCHAR(255) NULL,
  date_of_birth DATE NULL,
  email VARCHAR(255) NULL,
  phone_number VARCHAR(20) NULL,
  PRIMARY KEY (id))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table sisdb.teacher
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS sisdb.teacher (
  id INT NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(255) NOT NULL,
  last_name VARCHAR(255) NULL,
  email VARCHAR(255) NULL,
  PRIMARY KEY (id))
ENGINE = InnoDB;


-- -----------------------------------------------------
```

```
-- Table sisdb.course
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS sisdb.course (
  id INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  credits INT NULL,
  teacher_id INT NOT NULL,
  PRIMARY KEY (id, teacher_id),
  INDEX fk_course_teacher_idx (teacher_id ASC) ,
  CONSTRAINT fk_course_teacher
    FOREIGN KEY (teacher_id)
    REFERENCES sisdb.teacher (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table sisb.payment
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS sisdb.payment (
  id INT NOT NULL AUTO_INCREMENT,
  amount INT NULL,
  payment_date DATE NULL,
  student_id INT NOT NULL,
  PRIMARY KEY (id, student_id),
  INDEX fk_payment_student1_idx (student_id ASC) ,
  CONSTRAINT fk_payment_student1
    FOREIGN KEY (student_id)
    REFERENCES sisdb.student (id)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table sisdb.enrollment
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS sisdb.enrollment (
  id INT NOT NULL AUTO_INCREMENT,
  student_id INT NOT NULL,
  course_id INT NOT NULL,
  enrollment_date DATE NULL,
  PRIMARY KEY (id, student_id, course_id),
  INDEX fk_student_has_course_course1_idx (course_id ASC) ,
  INDEX fk_student_has_course_student1_idx (student_id ASC) ,
  CONSTRAINT fk_student_has_course_student1
    FOREIGN KEY (student_id)
    REFERENCES sisdb.student (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT fk_student_has_course_course1
    FOREIGN KEY (course_id)
    REFERENCES sisdb.course (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

use sisdb;

INSERTIONS:

-- student table

insert into student(first_name,last_name,date_of_birth,email,phone_number)

values

('Ram','Prasad','2002-03-30','ram@gmail.com','9024554745'),

('Sandiya','Vishwanath','2002-08-25','sandiya@gmail.com','9174543526'),

('Jayanthi','Selvam','2002-08-25','selvam@gmail.com','9082707895'),

('Swetha','Seetharaman','2005-04-11','swetha@gmail.com','7098645321'),

('Divya','Dharshini','2004-06-14','divya@gmail.com','9123765480'),

('Nisha','Vaithiyanathan','2000-07-14','nisha@gmail.com','9865432178'),

('Darshini','Balamurali','2001-07-15','darshnini@gmail.com','709834521'),

('Agalya','Shanmugam','2002-12-07','agalya@gmail.com','8143256790'),

('Harini','Murugavel','2002-12-16','harini@gmail.com','9024554745'),

('Selva','Ramaiah','1998-08-12','selva@gmail.com','9156473420');

```
mysql> select *from student;
+----+------------+---------------+--------------+----------------------+--------------+
| id | first_name | last_name     | date_of_birth | email                | phone_number |
+----+------------+---------------+--------------+----------------------+--------------+
|  1 | Ram        | Prasad        | 2002-03-30    | ram@gmail.com        | 9024554745   |
|  2 | Sandiya    | Vishwanath    | 2002-08-25    | sandiya@gmail.com    | 9174543526   |
|  3 | Jayanthi   | Selvam        | 2002-08-25    | selvam@gmail.com     | 9082707895   |
|  4 | Swetha     | Seetharaman   | 2005-04-11    | swetha@gmail.com     | 7098645321   |
|  5 | Divya      | Dharshini     | 2004-06-14    | divya@gmail.com      | 9123765480   |
|  6 | Nisha      | Vaithiyanathan | 2000-07-14    | nisha@gmail.com      | 9865432178   |
|  7 | Darshini   | Balamurali    | 2001-07-15    | darshnini@gmail.com  | 709834521    |
|  8 | Agalya     | Shanmugam     | 2002-12-07    | agalya@gmail.com     | 8143256790   |
|  9 | Harini     | Murugavel     | 2002-12-16    | harini@gmail.com     | 9024554745   |
| 10 | Selva      | Ramaiah       | 1998-08-12    | selva@gmail.com      | 9156473420   |
```

-- teacher table :

insert into teacher(first_name,last_name,email)values

('saraswathi','sri','saras@gmail.com'),

('anandha','valli','anandha@gmail.com'),

('ashwini','shanmugam','ashwini@gmail.com'),

('hema','malini','hema@gmail.com'),

('lenida','rose','rose@gmail.com'),

('karpaga','valli','karpagam@gmail.com'),

('kavitha','dharshini','kavitha@gmail.com'),

('lourd','mary','mary@gmail.com'),

('subha','dharshini','subhadharshini@gmail.com'),

('madhu','bala','madhu@gmail.com');

```
mysql> select*from teacher;
+----+------------+-----------+--------------------------+
| id | first_name | last_name | email                    |
+----+------------+-----------+--------------------------+
|  1 | saraswathi | sri       | saras@gmail.com          |
|  2 | anandha    | valli     | anandha@gmail.com        |
|  3 | ashwini    | shanmugam | ashwini@gmail.com        |
|  4 | hema       | malini    | hema@gmail.com           |
|  5 | lenida     | rose      | rose@gmail.com           |
|  6 | karpaga    | valli     | karpagam@gmail.com       |
|  7 | kavitha    | dharshini | kavitha@gmail.com        |
|  8 | lourd      | mary      | mary@gmail.com           |
|  9 | subha      | dharshini | subhadharshini@gmail.com |
| 10 | madhu      | bala      | madhu@gmail.com          |
+----+------------+-----------+--------------------------+
```

-- course table :

insert into course(name,credits,teacher_id)values

('java',75,1),

('C#',50,3),

('python',50,2),

('css',10,7),

('html',10,8),

('java fst',85,4),

('python fst',65,6),

('nodejs',68,5),

('javascript',45,9),

('reactjs',70,10);

```
mysql> select*from course;
+----+------------+---------+------------+
| id | name       | credits | teacher_id |
+----+------------+---------+------------+
| 11 | java       |      75 |          1 |
| 12 | C#         |      50 |          3 |
| 13 | python     |      50 |          2 |
| 14 | css        |      10 |          7 |
| 15 | html       |      10 |          8 |
| 16 | java fst   |      85 |          4 |
| 17 | python fst |      65 |          6 |
| 18 | nodejs     |      68 |          5 |
| 19 | javascript |      45 |          9 |
| 20 | reactjs    |      70 |         10 |
+----+------------+---------+------------+
```

-- enrollment table :

insert into 7nrolment(student_id,course_id,7nrolment_date)values

(1,12,’2023-08-03’),

(2,13,’2023-03-30’),

(3,12,’2023-03-30’),

(1,18,’2023-11-27’),

(4,14,’2023-08-03’),

(5,15,’2022-12-11’),

(6,19,’2024-01-31’),

(7,17,’2024-05-10’),

(8,20,’2023-08-03’),

(9,20,’2024-03-12’);

```
mysql> select * from enrollment;
+----+------------+-----------+-----------------+
| id | student_id | course_id | enrollment_date |
+----+------------+-----------+-----------------+
|  1 |          1 |        12 | 2023-08-03      |
|  2 |          2 |        13 | 2023-03-30      |
|  3 |          3 |        12 | 2023-03-30      |
|  4 |          1 |        18 | 2023-11-27      |
|  5 |          4 |        14 | 2023-08-03      |
|  6 |          5 |        15 | 2022-12-11      |
|  7 |          6 |        19 | 2024-01-31      |
|  8 |          7 |        17 | 2024-05-10      |
|  9 |          8 |        20 | 2023-08-03      |
| 10 |          9 |        20 | 2024-03-12      |
+----+------------+-----------+-----------------+
```

-- payment table :

```
insert into payment(amount,payment_date,student_id)values
(10000,'2023-08-03',1),
(1500,'2023-03-30',2),
(2000,'2023-03-30',3),
(1000,'2023-11-27',4),
(1000,'2023-08-03',1),
(10000,'2022-12-11',6),
(4000,'2024-01-31',7),
(1600,'2024-05-10',8),
(1600,'2023-08-03',9),
(1800,'2024-03-12',10);
```

```
mysql> select*from payment;
+----+--------+--------------+------------+
| id | amount | payment_date | student_id |
+----+--------+--------------+------------+
|  1 |  10000 | 2023-08-03   |          1 |
|  2 |   1500 | 2023-03-30   |          2 |
|  3 |   2000 | 2023-03-30   |          3 |
|  4 |   1000 | 2023-11-27   |          4 |
|  5 |   1000 | 2023-08-03   |          1 |
|  6 |  10000 | 2022-12-11   |          6 |
|  7 |   4000 | 2024-01-31   |          7 |
|  8 |   1600 | 2024-05-10   |          8 |
|  9 |   1600 | 2023-08-03   |          9 |
| 10 |   1800 | 2024-03-12   |         10 |
+----+--------+--------------+------------+
```

-- 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

b. Last Name: Doe

c. Date of Birth: 1995-08-15

d. Email: john.doe@example.com

e. Phone Number: 1234567890 */

```
insert into student(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 enrollment(student_id,course_id,enrollment_date)values
(11,16,'2020-05-23');
```

/* 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='lenidarose@gmail.com' where id=5;
```

/* 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 enrollment
where student_id=9 and course_id=20;
```

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

```
update course
set teacher_id=5  where name='python';
```

/* 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 s
from enrollment e,student s
where s.id=e.student_id and student_id = 9;
```

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

```
update payment
```

```
set amount=2000 where id=10;
```

-- 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.id,concat(s.first_name,s.last_name) as name,sum(p.amount) as total_payment
 from payment p join student s on s.id=p.student_id
 group by s.id;

 /*  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.name,count(e.id)as students_enrolled
from course c join enrollment e on c.id=e.course_id
group by c.id;

/* 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 concat(s.first_name,' ',last_name) as names
 from student s left join enrollment e on s.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.name
from student s join enrollment e on s.id=e.student_id join
course c on c.id=e.course_id
group by e.id;

/*  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 concat(t.first_name,' ',last_name) as names,c.name
from teacher t join course c on t.id=c.teacher_id
group by t.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 name,e.enrollment_date
from student s join enrollment e on s.id=e.student_id join course c on c.id=e.course_id
where c.name='java fst';

/* 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 name
from student s left join payment p on s.id=p.student_id
where p.amount 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.name
from course c left join enrollment e on c.id=e.course_id
where e.course_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 concat(s.first_name,' ',s.last_name) as name
from student s join enrollment e on s.id=e.student_id
having count(e.student_id)>1;
```

/* 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 concat(t.first_name,' ',t.last_name)as name
from course c left join teacher t  on t.id=c.teacher_id
where c.teacher_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 course_id,avg(student_id) as avg_student
from enrollment
where exists(
        select name
        from course )
group by course_id;
```

/* 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 concat (first_name, ' ', last_name) as name,(
 select max(amount)
  from payment
  where amount = (
                    select max(amount)
                    from payment
                    group by student_id limit 1))as max_amount
from student
limit 1;
```

/* 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 c.name ,(select count(e.student_id)
from enrollment e where e.course_id=c.id)as count_enrolled
from course c group by c.id
order by count_enrolled desc
limit 0,1;
```

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

```
select concat(t.first_name," ",t.last_name) as teacher_name, c.name,p.amount
from teacher t join course c on t.id=c.teacher_id join enrollment e on e.course_id=c.id join payment p on p.student_id=e.student_id
group by t.id;
```

/* 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 concat(s.first_name," ",s.last_name) as name
from student s join enrollment e
on e.student_id=s.id
group by s.id;
```

/* 6. Retrieve the names of teachers who have not been assigned to
any courses. Use subqueries to find teachers with no course
assignments. */

```
select concat(first_name," ",last_name) as teacher_name
from teacher
where id not in(select teacher_id from course);
```

/* 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(timestampdiff(year,date_of_birth,curdate()))
as average_age
from student;
```

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

```
select name from course
where id not in (select course_id from enrollment);
```

/* 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 s.first_name, s.last_name, c.name AS course_name,SUM(p.amount) AS
total_payments
```

```
from student s join enrollment e on s.id = e.student_id
join course c on e.course_id = c.id
join payment p on s.id = p.student_id
group by s.id, c.id;

/* 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 concat(first_name," ",last_name) as name from student
where id in(select student_id from payment
group by student_id
having count(*)>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 CONCAT(s.first_name, " ", s.last_name) AS student_name,SUM(p.amount) AS
total_payments
from student s
join payment p on s.id = p.student_id
group by s.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.name, count(e.student_id) as count_of_students
from course c join enrollment e
on c.id=e.course_id
group by c.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 concat(s.first_name," ",s.last_name) as
name,avg(p.amount) as avg_payment
from student s join payment p
on p.student_id=s.id
group by s.id;
```