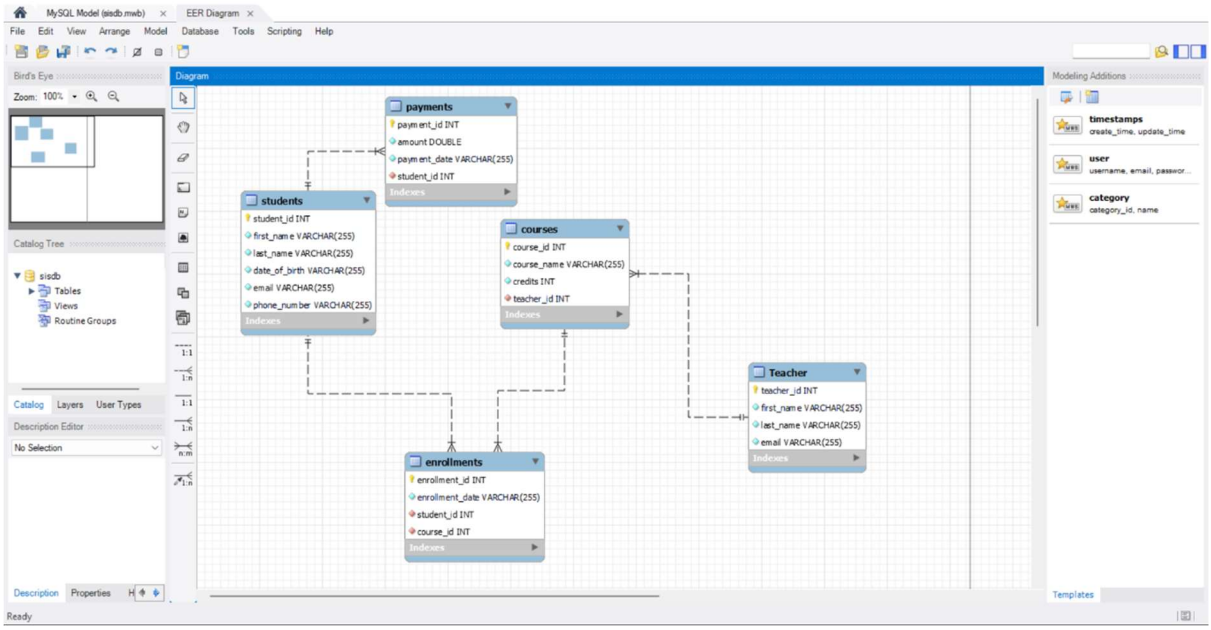
ASSIGNMENT-3

NAME	YALINI SHREE P V
SERIAL NO.	31
TOPIC	STUDENT MANAGEMENT SYSTEM

ER Diagram for reference



SQL code with outputs

```
/*
-- mysql workbench forward engineering
-- -----------------------------------------------------
-- schema sisdb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- schema sisdb
-- -----------------------------------------------------
create schema if not exists `sisdb` default character set utf8 ;
use `sisdb` ;
```

```
-- -----------------------------------------------------
-- table `sisdb`.`students`
-- -----------------------------------------------------
create table if not exists `sisdb`.`students` (
  `student_id` int not null auto_increment,
  `first_name` varchar(255) not null,
  `last_name` varchar(255) not null,
  `date_of_birth` varchar(255) not null,
  `email` varchar(255) not null,
  `phone_number` varchar(255) not null,
  primary key (`student_id`))
engine = innodb;
-- -----------------------------------------------------
-- table `sisdb`.`teacher`
-- -----------------------------------------------------
create table if not exists `sisdb`.`teacher` (
  `teacher_id` int not null auto_increment,
  `first_name` varchar(255) not null,
  `last_name` varchar(255) not null,
  `email` varchar(255) not null,
  primary key (`teacher_id`))
engine = innodb;
-- -----------------------------------------------------
-- table `sisdb`.`courses`
-- -----------------------------------------------------
create table if not exists `sisdb`.`courses` (
  `course_id` int not null auto_increment,
  `course_name` varchar(255) not null,
  `credits` int not null,
  `teacher_id` int not null,
  primary key (`course_id`),
```

```
  index `fk_courses_teacher_idx` (`teacher_id` asc) ,
  constraint `fk_courses_teacher`
    foreign key (`teacher_id`)
    references `sisdb`.`teacher` (`teacher_id`)
    on delete no action
    on update no action)
engine = innodb;
-- -----------------------------------------------------
-- table `sisdb`.`enrollments`
-- -----------------------------------------------------
create table if not exists `sisdb`.`enrollments` (
  `enrollment_id` int not null auto_increment,
  `enrollment_date` varchar(255) not null,
  `student_id` int not null,
  `course_id` int not null,
  primary key (`enrollment_id`),
  index `fk_enrollments_students1_idx` (`student_id` asc) ,
  index `fk_enrollments_courses1_idx` (`course_id` asc) ,
  constraint `fk_enrollments_students1`
    foreign key (`student_id`)
    references `sisdb`.`students` (`student_id`)
    on delete no action
    on update no action,
  constraint `fk_enrollments_courses1`
    foreign key (`course_id`)
    references `sisdb`.`courses` (`course_id`)
    on delete no action
    on update no action)
engine = innodb;
```

```
-- -----------------------------------------------------
-- table `sisdb`.`payments`
-- -----------------------------------------------------
create table if not exists `sisdb`.`payments` (
  `payment_id` int not null auto_increment,
  `amount` double not null,
  `payment_date` varchar(255) not null,
  `student_id` int not null,
  primary key (`payment_id`),
  index `fk_payments_students1_idx` (`student_id` asc) ,
  constraint `fk_payments_students1`
    foreign key (`student_id`)
    references `sisdb`.`students` (`student_id`)
    on delete no action
    on update no action)
engine = innodb;

use sisdb;
show tables;

insert into students (first_name,last_name,date_of_birth,email,phone_number)
values
('yalini shree','vetrivelan','2024-02-22','yalinishreevetrivelan@gmail.com','9486898136'),
('geetha','vetrivelan','1969-09-21','geethavetrivelan@gmail.com','9976211555'),
('vetrivelan','palanisamy','1968-10-12','pvetrivelan2014@gmail.com','9245408136'),
('poorrnima','vetrivelan','1996-03-09','poorrnimavetrivelan@gmail.com','6380408143');

insert into teacher (first_name,last_name,email)
values
('sheela','venkatraman','sheelavenkat@gmail.com'),
('pavithra','rangarajan','pavithraraj@gmail.com'),
```

```
('rajasekar','paneerselvam','rajselvam@gmail.com'),
('kavin','karthikeyan','kavinkarthi@gmail.com');

insert into payments (amount,payment_date,student_id)
values
(86000,'2024-01-10',1),
(74000,'2024-01-18',4),
(68000,'2024-01-22',2),
(91000,'2024-01-28',3);

insert into courses (course_name,credits,teacher_id)
values
('digital electronics',3,2),
('analog circuits',4,4),
('signals and system',5,1),
('electronic devices',1,3);

insert into enrollments (enrollment_date,student_id,course_id)
values
('2023-12-11',3,3),
('2023-12-17',1,2),
('2023-12-07',2,2),
('2023-12-01',4,1),
('2023-12-21',2,3),
('2023-12-29',1,4);
```

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 students (first_name,last_name,date_of_birth,email,phone_number)
values
('john','doe','1995-08-15','john.doe@example.com','1234567890');
select* from students;
```

1	yalini shree	vetrivelan	2024-02-22	yalinishreevetrivelan@gmail.com	9486898136
2	geetha	vetrivelan	1969-09-21	geethavetrivelan@gmail.com	9976211555
3	vetrivelan	palanisamy	1968-10-12	pvetrivelan2014@gmail.com	9245408136
4	poorrnima	vetrivelan	1996-03-09	poorrnimavetrivelan@gmail.com	6380408143
5	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 (enrollment_date,student_id,course_id)
values
('2024-01-18',5,2);
select* from enrollments;
```

1	2023-12-11	3	3
2	2023-12-17	1	2
3	2023-12-07	2	2
4	2023-12-01	4	1
5	2023-12-21	2	3
6	2023-12-29	1	4
7	2024-01-18	5	2

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 = 'kavinkarthikeyan@gmail.com'

where teacher_id=4;

select* from teacher;
```

1	sheela	venkatraman	sheelavenkat@gmail.com
2	pavithra	rangarajan	pavithraraj@gmail.com
3	rajasekar	paneerselvam	rajselvam@gmail.com
4	kavin	karthikeyan	kavinkarthikeyan@gmail.com

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

enrollment_id=6;

select* from enrollments;
```

1	2023-12-11	3	3
2	2023-12-17	1	2
3	2023-12-07	2	2
4	2023-12-01	4	1
5	2023-12-21	2	3
7	2024-01-18	5	2

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_name='analog circuits';

select* from courses;
```

1	digital electronics	3	2
2	analog circuits	4	2
3	signals and system	5	1

4 electronic devices 1 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=5;

select* from enrollments;

1 2023-12-11 3 3

2 2023-12-17 1 2

3 2023-12-07 2 2

4 2023-12-01 4 1

5 2023-12-21 2 3

delete from students

where first_name='john';

select* from students;

1 yalini shree vetrivelan 2024-02-22 yalinishreevetrivelan@gmail.com 9486898136

2 geetha vetrivelan 1969-09-21 geethavetrivelan@gmail.com 9976211555

3 vetrivelan palanisamy 1968-10-12 pvetrivelan2014@gmail.com 9245408136

4 poorrnima vetrivelan 1996-03-09 poorrnimavetrivelan@gmail.com 6380408143

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=98000

where payment_id=2;

select* from payments;

1 86000 2024-01-10 1

2 98000 2024-01-18 4

3 68000 2024-01-22 2

4 91000 2024-01-28 3

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.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 = 1
group by s.student_id;
```

yalini shree vetrivelan 86000

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) as enrolled_students
from courses c
left join enrollments e on c.course_id = e.course_id
group by c.course_id;
```

digital electronics 1

analog circuits 2

signals and system 2

electronic devices 0

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.first_name, s.last_name
from students s
left join enrollments e on s.student_id = e.student_id
where e.enrollment_id is null;
```

nill

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
join enrollments e on s.student_id = e.student_id
join courses c on e.course_id = c.course_id;
```

vetrivelan palanisamy signals and system

yalini shree vetrivelan analog circuits

geetha vetrivelan analog circuits

poorrnima vetrivelan digital electronics

geetha vetrivelan signals and system

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, c.course_name
from teacher t
join courses c on t.teacher_id = c.teacher_id;
```

pavithra rangarajan digital electronics

pavithra rangarajan analog circuits

sheela venkatraman signals and system

rajasekar paneerselvam electronic devices

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 s.first_name, s.last_name, e.enrollment_date, 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
where c.course_name = 'specific_course_name';
```

nill

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 s.first_name, s.last_name
```

from students s

left join payments p on s.student_id = p.student_id

where p.payment_id is null;

nill

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_name

from courses c

left join enrollments e on c.course_id = e.course_id

where e.enrollment_id is null;

electronic devices

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.

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.

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.

2. identify the students) who made the highest payment. use a subquery to find the maximum payment amount and then retrieve the students) associated with that amount.

select s.first_name, s.last_name, p.amount

from students s

join payments p on s.student_id = p.student_id

where p.amount = (select max(amount) from payments);

poorrnima vetrivelan 98000

3. retrieve a list of courses with the highest number of enrollments. use subqueries to find the courses with the maximum enrollment count.

4. calculate the total payments made to courses taught by each teacher. use subqueries to sum payments for each teacher's courses.

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.first_name, s.last_name
from students s
where (select count(distinct course_id) from enrollments) =
    (select count(distinct course_id)
     from enrollments e
     where e.student_id = s.student_id);
```

nill

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.first_name, t.last_name
from teacher t
where t.teacher_id not in (select distinct teacher_id from courses);
```

kavin karthikeyan

7. calculate the average age of all students. use subqueries to calculate the age of each student based on their date of birth.

8. identify courses with no enrollments. use subqueries to find courses without enrollment records.

```
select c.course_name
from courses c
where c.course_id not in (select distinct course_id from enrollments);
```

electronic devices

9. calculate the total payments made by each student for each course they are enrolled in. use subqueries and aggregate functions to sum payments.

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 s.first_name, s.last_name

from students s

join payments p on s.student_id = p.student_id

group by s.student_id

having count(p.payment_id) > 1;
```

nill