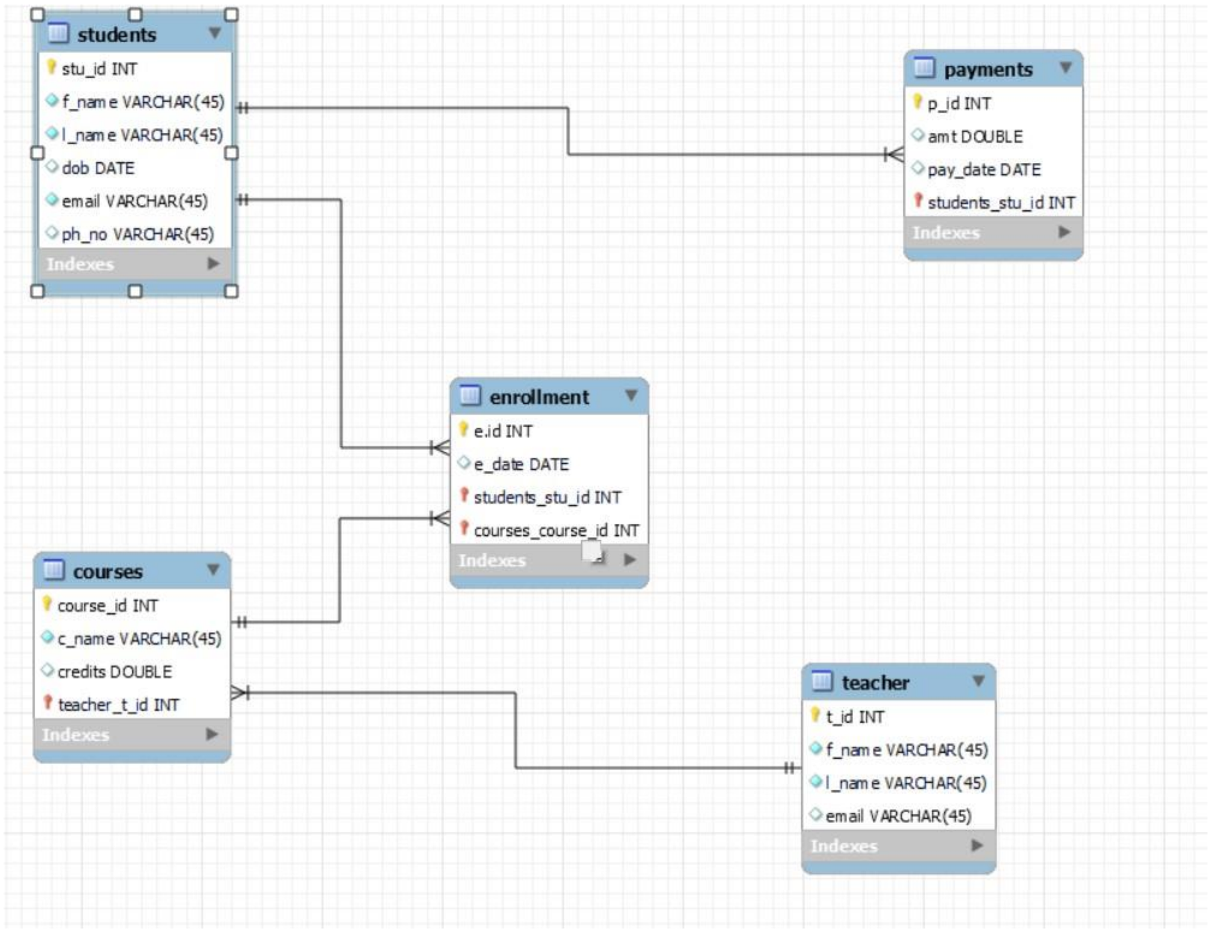
STUDENT

ER DIAGRAM:



TASK – 1

```
CREATE SCHEMA IF NOT EXISTS `student` DEFAULT CHARACTER SET utf8 ;

USE `student` ;

-- -----------------------------------------------------

-- Table `student`.`students`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `student`.`students` (
  `stu_id` INT NOT NULL AUTO_INCREMENT,
  `f_name` VARCHAR(45) NOT NULL,
  `l_name` VARCHAR(45) NOT NULL,
  `dob` DATE NULL,
  `email` VARCHAR(45) NOT NULL,
  `ph_no` VARCHAR(45) NULL,
  PRIMARY KEY (`stu_id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `student`.`teacher`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `student`.`teacher` (
  `t_id` INT NOT NULL AUTO_INCREMENT,
  `f_name` VARCHAR(45) NOT NULL,
  `l_name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NULL,
  PRIMARY KEY (`t_id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `student`.`courses`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `student`.`courses` (
  `course_id` INT NOT NULL AUTO_INCREMENT,
  `c_name` VARCHAR(45) NOT NULL,
  `credits` DOUBLE NULL,
  `teacher_t_id` INT NOT NULL,
  PRIMARY KEY (`course_id`, `teacher_t_id`),
  INDEX `fk_courses_teacher_idx` (`teacher_t_id` ASC) ,
  CONSTRAINT `fk_courses_teacher`
```

```
    FOREIGN KEY (`teacher_t_id`)
    REFERENCES `student`.`teacher` (`t_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `student`.`enrollment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `student`.`enrollment` (
  `e.id` INT NOT NULL AUTO_INCREMENT,
  `e_date` DATE NULL,
  `students_stu_id` INT NOT NULL,
  `courses_course_id` INT NOT NULL,
  PRIMARY KEY (`e.id`, `students_stu_id`, `courses_course_id`),
  INDEX `fk_enrollment_students1_idx` (`students_stu_id` ASC) ,
  INDEX `fk_enrollment_courses1_idx` (`courses_course_id` ASC) ,
  CONSTRAINT `fk_enrollment_students1`
    FOREIGN KEY (`students_stu_id`)
    REFERENCES `student`.`students` (`stu_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_enrollment_courses1`
    FOREIGN KEY (`courses_course_id`)
    REFERENCES `student`.`courses` (`course_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `student`.`payments`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `student`.`payments` (
  `p_id` INT NOT NULL AUTO_INCREMENT,
  `amt` DOUBLE NULL,
  `pay_date` DATE NULL,
  `students_stu_id` INT NOT NULL,
  PRIMARY KEY (`p_id`, `students_stu_id`),
  INDEX `fk_payments_students1_idx` (`students_stu_id` ASC) ,
  CONSTRAINT `fk_payments_students1`
    FOREIGN KEY (`students_stu_id`)
    REFERENCES `student`.`students` (`stu_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
use student;
 show tables;
 desc courses;
 desc enrollment;
 desc payments;
 desc students;
 desc teacher;
 insert into students(f_name,l_name,dob,email,ph_no) values('ram','chandran','2001-04-14','ram@gmail.com','78459658745'),
 ('rama','krishna','2002-03-04','rama@gmail.com','98459658745'),
 ('ravi','thakur','2002-06-14','ravi@gmail.com','78487658745'),
 ('rani','shekar','2001-12-12','rani@gmail.com','84459658745');
 delete from students where stu_id in(7,8,9);
  insert into students(f_name,l_name,dob,email,ph_no) values('raja','chandu','2001-05-14','raja@gmail.com','78459458745'),
  ('sarah','khan','2002-12-14','sarah@gmail.com','78459158745'),
 ('chandra','mauraya','2002-10-27','chandra@gmail.com','98459658745'),

 ('rakesh','gupta','2001-08-07','rakesh@gmail.com','88459658745'),

 ('vinoth','kumar','2003-01-14','vinoth@gmail.com','78459658425');

select * from students;

insert into teacher(f_name,l_name,email) values
('vani','bhojan','vani@gmail.com'),('sita','ram','sita@gmail.com'),

('narmitha','reddy','narmitha@gmail.com'),('helen','keller','helen@gmail.com');

delete from teacher where t_id in(5,6,7,8);

insert into teacher(f_name,l_name,email) values

('Olivia', 'Brown', 'olivia.brown@gmail.com'),

('James', 'Jones', 'james.jones@gmail.com'),

('Sophia', 'Miller', 'sophia.miller@gmail.com'),

('William', 'Davis', 'william.davis@gmail.com'),

('Ava', 'Garcia', 'ava.garcia@gmail.com');

select * from teacher;

insert into courses(c_name,credits,teacher_t_id)
values('c',4,2),('c++',3.5,1),('python',3,4),('java',4.5,3),

('javascript',2,1),('r',3,11),('problem_solving',4,12),('react',3,10),('junit',4,9);

delete from courses where course_id in (11,12,13,14,15);

delete from courses where course_id in (6,7,8,9,10);

select * from courses;

insert into payments(amt,pay_date,students_stu_id) values(20000,'2024-02-01',1),

(25000,'2024-02-11',3),(11000,'2024-03-01',1),(15000,'2024-01-21',2),(27000,'2024-01-
31',4);

insert into payments(amt,pay_date,students_stu_id) values(18000,'2024-02-
01',12),(18000,'2024-02-01',13),(11000,'2024-02-03',10),(15000,'2024-02-02',11);

select * from payments;

insert into enrollment(e_date,students_stu_id,courses_course_id) values('2024-04-
01',1,5),('2024-04-11',2,3),

('2024-03-27',4,2),('2024-04-10',3,4);
```

```
insert into enrollment(e_date,students_stu_id,courses_course_id) values('2024-04-01',5,17),('2024-04-01',10,19),('2024-04-01',12,18),('2024-04-01',14,1),('2024-04-01',13,5);
select * from enrollment;
```

TASK - 2

#1.insert a new student into student table

```
insert into students(f_name,l_name,dob,email,ph_no) values ("john","doe","1995-08-15","john.doe@exampl.com","1234567890");
```

#2.insert a record in ennrollment table with already existing student and course

```
insert into enrollment(e_date,students_stu_id,courses_course_id) values('2024-04-01',1,1);
```

#3.update email of specific teacher

```
update teacher
set email="narmi@gmail.com"
where t_id=3;
```

#4.delete a record from enrollment table

```
-- set foreign_key_checks=0;
delete from enrollment where courses_course_id=2;
-- set foreign_key_checks=1;
```

#5.update course table by assigning specific teacher

```
update courses
set teacher_t_id=2
where teacher_t_id=3;
```

#6.delete specific student from student table and remove their enrollments

```
-- set foreign_key_checks=0;
```

```
delete from students where f_name="ram";
-- set foreign_key_checks=1;
delete from enrollment where students_stu_id=1;
```

#7.modify payment amount

```
update payments
set amt=26500
where amt=27000;
```

TASK - 3

#1.total payments made by a specific student

```
select students_stu_id,sum(amt) as tot_amt
from payments
group by students_stu_id;
```

#2.list of courses along with no of students enrolled in a course

```
select c.c_name,count(e.courses_course_id)
from courses c left join enrollment e
on c.course_id=e.courses_course_id
group by c.course_id;
```

#3. courses that are not enrolled by students

```
select c.c_name
from courses c left join enrollment e
on c.course_id=e.courses_course_id
group by c.course_id
having count(e.courses_course_id)=0;
```

#4.to retrieve names of students and courses they are enrolled in

```
select s.f_name,c.c_name
from students s,enrollment e,courses c
where s.stu_id=e.students_stu_id and e.courses_course_id=c.course_id;
```

#5.names of teachers and the courses they are assigned

```
select t.f_name,c.c_name
from teacher t join courses c
on t.t_id=c.teacher_t_id;
```

#6.list of students and their enrollment dates for a specific course

```
select s.f_name,e.e_date
from students s join enrollment e on s.stu_id=e.students_stu_id
join courses c on e.courses_course_id=c.course_id
where c.c_name="java";
-- Alternative
select s.f_name,e.e_date
from students s,enrollment e,courses c
where s.stu_id=e.students_stu_id and e.courses_course_id=c.course_id and
c.c_name="java";
```

#7.name of students who have not made any payments

```
select f_name from students where stu_id not in (select p.students_stu_id
from students s  join payments p
on s.stu_id=p.students_stu_id
);
```

#8.students who are enrolled in more than 1 course

```
select s.f_name,count(e.e_date) as enrolled
from students s join enrollment e
on s.stu_id=e.students_stu_id
```

```
group by courses_course_id
having enrolled>1;
-- select * from enrollment
```

#9.teachers who are not assigned to any course

```
select f_name from teacher where t_id not in (select t.t_id
from courses c join teacher t
on t.t_id=c.teacher_t_id);
delete from payments where students_stu_id =1;
```

TASK - 4

#1.students who made highest payments

```
select f_name from students where stu_id in(select students_stu_id from payments where amt =
(select sum(amt) as tot from payments
group by students_stu_id order by tot desc limit 1)) ;
/*
select * from students;
select * from payments;
select sum(amt) as tot from payments
group by students_stu_id order by tot desc limit 1;
*/
```

#2.retrieve list of courses with highest number of enrollments

```
select * from enrollment;
select c_name from courses where course_id in(select courses_course_id from enrollment where enrolled=
(select count(students_stu_id) as enrolled from enrollment group by courses_course_id));
```

```
-- alternative
select count(students_stu_id) as enrolled from enrollment group by courses_course_id
having enrolled>10;
```

#3.courses with no enrollments

```
select course_id from courses where course_id not in(select courses_course_id from enrollment where students_stu_id
in (select stu_id from students));
select * from courses;
select * from enrollment;
select * from students;
```

#4.names of teacher not aasigned to any courses

```
select * from teacher;
select t_id from teacher where t_id not in(select teacher_t_id from courses);
```

#5.calculate total payments made by each student

```
select s.f_name,sum(p.amt) as pay
from students s join payments p
on s.stu_id = p.students_stu_id
group by s.f_name order by pay desc;
```

#6.courses name along with count of students

```
select c.c_name,count(s.stu_id)
from students s,enrollment e,courses c
where s.stu_id=e.students_stu_id
and e.courses_course_id=c.course_id
group by c.c_name;
```

#7.students who made more than 1 payment

```
select s.f_name,count(p.students_stu_id) as pay

from students s join payments p

on s.stu_id=p.students_stu_id

group by s.f_name

having pay>1;
```