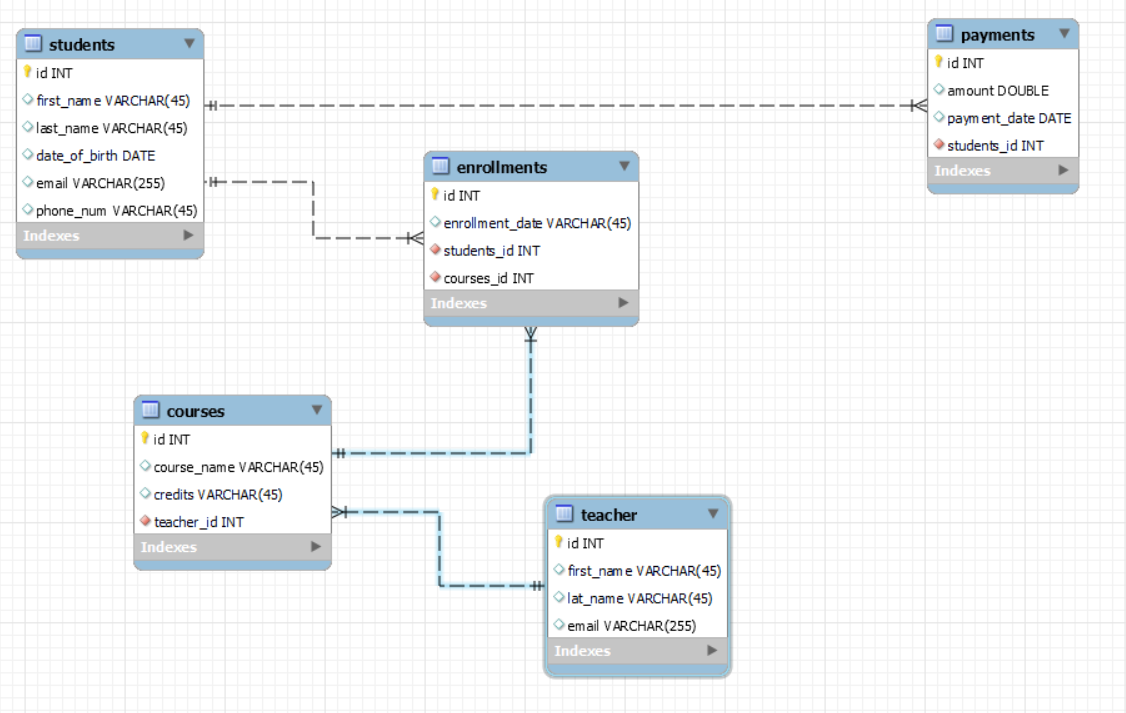
Student Information System



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

-- Schema student\_information\_system

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

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

-- Schema student\_information\_system

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

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

USE `student\_information\_system` ;

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

-- Table `student\_information\_system`.`students`

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

CREATE TABLE IF NOT EXISTS `student\_information\_system`.`students` (

`id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NULL,

`last\_name` VARCHAR(45) NULL,

`date\_of\_birth` DATE NULL,

`email` VARCHAR(255) NULL,

`phone\_num` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

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

-- Table `student\_information\_system`.`teacher`

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

CREATE TABLE IF NOT EXISTS `student\_information\_system`.`teacher` (

`id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NULL,

`lat\_name` VARCHAR(45) NULL,

`email` VARCHAR(255) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

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

-- Table `student\_information\_system`.`courses`

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

CREATE TABLE IF NOT EXISTS `student\_information\_system`.`courses` (

`id` INT NOT NULL AUTO\_INCREMENT,

`course\_name` VARCHAR(45) NULL,

`credits` VARCHAR(45) NULL,

`teacher\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_courses\_teacher\_idx` (`teacher\_id` ASC) ,

CONSTRAINT `fk\_courses\_teacher`

FOREIGN KEY (`teacher\_id`)

REFERENCES `student\_information\_system`.`teacher` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `student\_information\_system`.`enrollments`

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

CREATE TABLE IF NOT EXISTS `student\_information\_system`.`enrollments` (

`id` INT NOT NULL AUTO\_INCREMENT,

`enrollment\_date` VARCHAR(45) NULL,

`students\_id` INT NOT NULL,

`courses\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_enrollments\_students1\_idx` (`students\_id` ASC) ,

INDEX `fk\_enrollments\_courses1\_idx` (`courses\_id` ASC) ,

CONSTRAINT `fk\_enrollments\_students1`

FOREIGN KEY (`students\_id`)

REFERENCES `student\_information\_system`.`students` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_enrollments\_courses1`

FOREIGN KEY (`courses\_id`)

REFERENCES `student\_information\_system`.`courses` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `student\_information\_system`.`payments`

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

CREATE TABLE IF NOT EXISTS `student\_information\_system`.`payments` (

`id` INT NOT NULL AUTO\_INCREMENT,

`amount` DOUBLE NULL,

`payment\_date` DATE NULL,

`students\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_payments\_students1\_idx` (`students\_id` ASC) ,

CONSTRAINT `fk\_payments\_students1`

FOREIGN KEY (`students\_id`)

REFERENCES `student\_information\_system`.`students` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

--Insertions

insert into students(first\_name,last\_name,date\_of\_birth,email,phone\_num) values

('nivetha','R','12.11.2002','nive@gmail.com','98765'),

('nirupama','S','10.1.2002','niru@gmail.com','12345'),

('hari','A','7.8.2002','hari@gmail.com','10002'),

('anu','S','3.12.2002','anu@gmail.com','13579'),

('nithi','B','25.5.2003','nithi@gmail.com','24680'),

('sangeetha','S','15.11.2002','sangee@gmail.com','50000');

select \* from students;

+----+------------+-----------+---------------+----------------------+------------+

| id | first\_name | last\_name | date\_of\_birth | email | phone\_num |

+----+------------+-----------+---------------+----------------------+------------+

| 1 | nivetha | R | 2012-11-20 | nive@gmail.com | 98765 |

| 2 | nirupama | S | 2010-01-20 | niru@gmail.com | 12345 |

| 3 | hari | A | 2007-08-20 | hari@gmail.com | 10002 |

| 4 | anu | S | 2003-12-20 | anu@gmail.com | 13579 |

| 5 | nithi | B | 2025-05-20 | nithi@gmail.com | 24680 |

| 6 | sangeetha | S | 2015-11-20 | sangee@gmail.com | 50000 |

insert into teacher(first\_name,lat\_name,email) values

('jayanthi','S','jainthi@gmail.com'),

('shanthi','M','shanthi@gmail.com'),

('janai','R','janani@gmail.com'),

('sajeetha','B','sajee@gmail.com'),

('priya','E','priya@gmail.com');

Select \* from teacher;

+----+------------+----------+-------------------+

| id | first\_name | lat\_name | email |

+----+------------+----------+-------------------+

| 1 | jayanthi | S | jainthi@gmail.com |

| 2 | shanthi | M | shanthi@gmail.com |

| 3 | janai | R | janani@gmail.com |

| 4 | sajeetha | B | sajee@gmail.com |

| 5 | priya | E | priya@gmail.com |

+----+------------+----------+-------------------+

insert into courses(course\_name,credits,teacher\_id) values

('java','A','1'),

('python','B','2'),

('c#','C','3'),

('jva script','D','4'),

('SQL','E','5');

Select \* from courses;

+----+-------------+---------+------------+

| id | course\_name | credits | teacher\_id |

+----+-------------+---------+------------+

| 1 | java | A | 1 |

| 2 | python | B | 2 |

| 3 | c# | C | 3 |

| 4 | jva script | D | 4 |

| 5 | SQL | E | 5 |

+----+-------------+---------+------------+

INSERT INTO enrollments(enrollment\_date, students\_id, courses\_id)

VALUES

('2024-01-01', 1, 1),

('2024-01-02', 2, 2),

('2024-01-03', 3, 3),

('2024-01-04', 4, 4),

('2024-01-05', 5, 5);

Select \* from enrollments;

+----+-----------------+-------------+------------+

| id | enrollment\_date | students\_id | courses\_id |

+----+-----------------+-------------+------------+

| 1 | 2024-01-01 | 1 | 1 |

| 2 | 2024-01-02 | 2 | 2 |

| 3 | 2024-01-03 | 3 | 3 |

| 4 | 2024-01-04 | 4 | 4 |

| 5 | 2024-01-05 | 5 | 5 |

+----+-----------------+-------------+------------+

INSERT INTO payments(id, amount, payment\_date, students\_id)

VALUES

(1, 1000, '2024-01-01', 1),

(2, 1500, '2024-01-02', 2),

(3, 800, '2024-01-03', 3),

(4, 1200, '2024-01-04', 4),

(5, 1600, '2024-01-05', 5);

Select \* from payments;

+----+--------+--------------+-------------+

| id | amount | payment\_date | students\_id |

+----+--------+--------------+-------------+

| 1 | 1000 | 2024-01-01 | 1 |

| 2 | 1500 | 2024-01-02 | 2 |

| 3 | 800 | 2024-01-03 | 3 |

| 4 | 1200 | 2024-01-04 | 4 |

| 5 | 1600 | 2024-01-05 | 5 |

+----+--------+--------------+-------------+

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: 123456789

insert into students (first\_name, last\_name, date\_of\_birth, email, phone\_num)

values ('john', 'doe', '1995-08-15', 'john.doe@example.com', '1234567890');

+----+------------+-----------+---------------+----------------------+------------+

| id | first\_name | last\_name | date\_of\_birth | email | phone\_num |

+----+------------+-----------+---------------+----------------------+------------+

| 1 | nivetha | r | 2012-11-20 | nive@gmail.com | 98765 |

| 2 | nirupama | s | 2010-01-20 | niru@gmail.com | 12345 |

| 3 | hari | a | 2007-08-20 | hari@gmail.com | 10002 |

| 4 | anu | s | 2003-12-20 | anu@gmail.com | 13579 |

| 5 | nithi | b | 2025-05-20 | nithi@gmail.com | 24680 |

| 6 | sangeetha | s | 2015-11-20 | sangee@gmail.com | 50000 |

| 7 | 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, students\_id, courses\_id)

values ('2024-03-06', 3, 3);

select \* from enrollments;

+----+-----------------+-------------+------------+

| id | enrollment\_date | students\_id | courses\_id |

+----+-----------------+-------------+------------+

| 1 | 2024-01-01 | 1 | 1 |

| 2 | 2024-01-02 | 2 | 2 |

| 3 | 2024-01-03 | 3 | 3 |

| 4 | 2024-01-04 | 4 | 4 |

| 5 | 2024-01-05 | 5 | 5 |

| 6 | 2024-03-06 | 3 | 3 |

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 = 'teacher@gmail.com'

where id = 4;

select \* from teacher;

+----+------------+----------+-------------------+

| id | first\_name | lat\_name | email |

+----+------------+----------+-------------------+

| 1 | jayanthi | s | jainthi@gmail.com |

| 2 | shanthi | m | shanthi@gmail.com |

| 3 | janai | r | janani@gmail.com |

| 4 | sajeetha | b | teacher@gmail.com |

| 5 | priya | e | priya@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 students\_id = 1 and courses\_id = 1;

+----+-----------------+-------------+------------+

| id | enrollment\_date | students\_id | courses\_id |

+----+-----------------+-------------+------------+

| 2 | 2024-01-02 | 2 | 2 |

| 3 | 2024-01-03 | 3 | 3 |

| 4 | 2024-01-04 | 4 | 4 |

| 5 | 2024-01-05 | 5 | 5 |

| 6 | 2024-03-06 | 3 | 3 |

+------------------+---------------+-----+---------------+

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 = 1

where id = 2;

+----+-------------+---------+------------+

| id | course\_name | credits | teacher\_id |

+----+-------------+---------+------------+

| 1 | java | a | 1 |

| 2 | python | b | 1 |

| 3 | c# | c | 3 |

| 4 | jva script | d | 4 |

| 5 | sql | e | 5 |

+----+-------------+---------+------------+

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.

set student\_id = 1;

delete from enrollments where students\_id = 1;

delete from students where students\_id=1;

+----+-----------------+-------------+------------+

| id | enrollment\_date | students\_id | courses\_id |

+----+-----------------+-------------+------------+

| 4 | 2024-01-04 | 4 | 4 |

| 5 | 2024-01-05 | 5 | 5 |

+----+-----------------+-------------+------------+

+----+------------+-----------+---------------+----------------------+------------+

| id | first\_name | last\_name | date\_of\_birth | email | phone\_num |

+----+------------+-----------+---------------+----------------------+------------+

| 2 | nirupama | s | 2010-01-20 | niru@gmail.com | 12345 |

| 3 | hari | a | 2007-08-20 | hari@gmail.com | 10002 |

| 4 | anu | s | 2003-12-20 | anu@gmail.com | 13579 |

| 5 | nithi | b | 2025-05-20 | nithi@gmail.com | 24680 |

| 6 | sangeetha | s | 2015-11-20 | sangee@gmail.com | 50000 |

| 7 | john | doe | 1995-08-15 | john.doe@example.com | 1234567890 |

+----+------------+-----------+---------------+----------------------+------------+

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 = 2000

where id = 1;

+----+--------+--------------+-------------+

| id | amount | payment\_date | students\_id |

+----+--------+--------------+-------------+

| 1 | 2000 | 2024-01-01 | 1 |

| 2 | 1500 | 2024-01-02 | 2 |

| 3 | 800 | 2024-01-03 | 3 |

| 4 | 1200 | 2024-01-04 | 4 |

| 5 | 1600 | 2024-01-05 | 5 |

+----+--------+--------------+-------------+

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.id = p.students\_id

where s.id = 3;

+------------+-----------+----------------+

| first\_name | last\_name | total\_payments |

+------------+-----------+----------------+

| hari | a | 800 |

+------------+-----------+----------------+

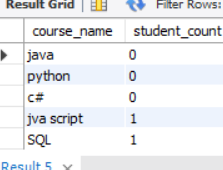
1. write an sql query to retrieve a list of courses along with the count of students enrolled in each course. use a join operation between the "courses" table and the "enrollments" table.

select c.course\_name, count(e.students\_id) as student\_count

from courses c

left join enrollments e on c.id = e.courses\_id

group by c.id;



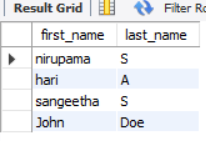
1. write an sql query to find the names of students who have not enrolled in any course. use a left join between the "students" table and the "enrollments" table to identify students without enrollments.

select s.first\_name, s.last\_name

from students s

left join enrollments e on s.id = e.students\_id

where e.students\_id is null;



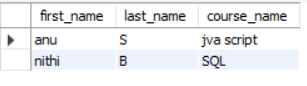
1. write an sql query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. use join operations between the "students" table and the "enrollments" and "courses" tables

select s.first\_name, s.last\_name, c.course\_name

from students s

join enrollments e on s.id = e.students\_id

join courses c on e.courses\_id = c.id;

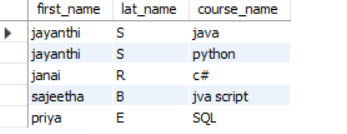


1. create a query to list the names of teachers and the courses they are assigned to. join the "teacher" table with the "courses" table.

select t.first\_name, t.lat\_name, c.course\_name

from teacher t

join courses c on t.id = c.teacher\_id



1. 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" table

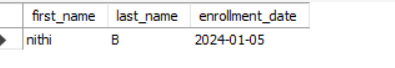
select s.first\_name, s.last\_name, e.enrollment\_date

from students s

join enrollments e on s.id = e.students\_id

join courses c on e.courses\_id = c.id

where c.course\_name = 'sql';



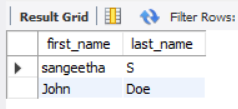
1. find the names of students who have not made any payments. use a left join between the "students" table and the "payments" table and filter for students with null payment records

select s.first\_name, s.last\_name

from students s

left join payments p on s.id = p.students\_id

where p.students\_id is null;



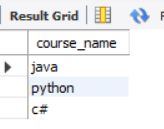
1. write a query to identify courses that have no enrollments. you'll need to use a left join between the "courses" table and the "enrollments" table and filter for courses with null enrollment records

select c.course\_name

from courses c

left join enrollments e on c.id = e.courses\_id

where e.courses\_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 s.first\_name, s.last\_name, count(e.courses\_id) as course\_count

from students s

join enrollments e on s.id = e.students\_id

group by s.id

having count(e.courses\_id) > 1;



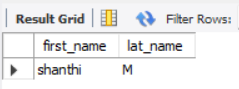
1. find teachers who are not assigned to any courses. use a left join between the "teacher" table and the "courses" table and filter for teachers with null course assignments.

select t.first\_name, t.lat\_name

from teacher t

left join courses c 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 avg(student\_count) as avg\_students\_per\_course

from (

select count(\*) as student\_count

from enrollments

group by courses\_id

) as course\_counts;

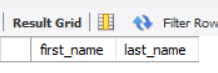


1. identify the student(s) who made the highest payment. use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount

select s.first\_name, s.last\_name

from students s

where s.id in (select students\_id from payments where amount = (select max(amount) from payments));



1. retrieve a list of courses with the highest number of enrollments. use subqueries to find the course(s) with the maximum enrollment count

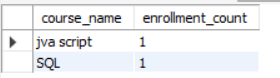
select c.course\_name, count(e.students\_id) as enrollment\_count

from courses c

left join enrollments e on c.id = e.courses\_id

group by c.id

having count(e.students\_id) = (select max(enrollment\_count) from (select count(\*) as enrollment\_count from enrollments group by courses\_id) as counts);



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

select t.first\_name, t.lat\_name, sum(p.amount) as total\_payments

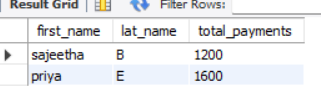
from teacher t

join courses c on t.id = c.teacher\_id

join enrollments e on c.id = e.courses\_id

join payments p on e.students\_id = p.students\_id

group by t.id;



1. 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 courses\_id) from enrollments where students\_id = s.id) = (select count(\*) from courses);

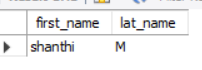


1. retrieve the names of teachers who have not been assigned to any courses. use subqueries to find teachers with no course assignments

select t.first\_name, t.last\_name

from teacher t

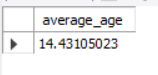
where t.id not in (select distinct teacher\_id from courses);



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

select avg(datediff(curdate(), date\_of\_birth) / 365) as average\_age

from students;

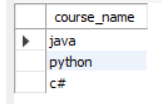


1. identify courses with no enrollments. use subqueries to find courses without enrollment records.

select c.course\_name

from courses c

where c.id not in (select distinct courses\_id from enrollments);



1. 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.course\_name, sum(p.amount) as total\_payments

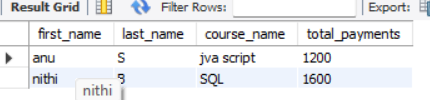
from students s

join enrollments e on s.id = e.students\_id

join courses c on e.courses\_id = c.id

join payments p on e.students\_id = p.students\_id

group by s.id, c.id;



1. 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

where s.id in (select students\_id from payments group by students\_id having count(\*) > 1);



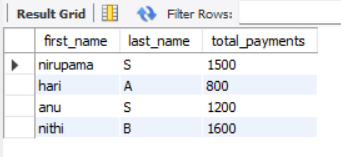
1. write an sql query to calculate the total payments made by each student. join the "students" table with the "payments" table and use group by to calculate the sum of payments for each student.

select s.first\_name, s.last\_name, sum(p.amount) as total\_payments

from students s

join payments p on s.id = p.students\_id

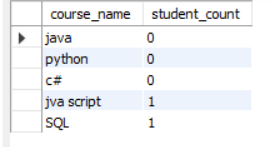
group by s.id;



1. 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.course\_name, (select count(\*) from enrollments where courses\_id = c.id) as student\_count

from courses c;



1. 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 avg(amount) as average\_payment\_amount

from payments;

