SMS

```
CREATE SCHEMA IF NOT EXISTS `sms` DEFAULT CHARACTER SET utf8 ;
USE `sms` ;
create database sms;
use sms;
-- -----------------------------------------------------
-- Table `sms`.`students`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sms`.`students` (
 `id` INT NOT NULL AUTO_INCREMENT,
 `first_name` VARCHAR(255) NULL,
 `last_name` VARCHAR(255) NULL,
 `date_of_birth` DATE NULL,
 `email` VARCHAR(255) NULL,
 `phone_number` BIGINT NULL,
 PRIMARY KEY (`id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `sms`.`teacher`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sms`.`teacher` (
 `id` INT NOT NULL AUTO_INCREMENT,
 `first_name` VARCHAR(255) NULL,
 `last_name` VARCHAR(255) NULL,
 `email` VARCHAR(255) NULL,
 PRIMARY KEY (`id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `sms`.`courses`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sms`.`courses` (
 `id` INT NOT NULL AUTO_INCREMENT,
 `course_name` VARCHAR(255) NULL,
 `credits` INT NULL,
 `teacher_id` INT NOT NULL,
 PRIMARY KEY (`id`, `teacher_id`),
 INDEX `fk_courses_teacher_idx` (`teacher_id` ASC) ,
 CONSTRAINT `fk_courses_teacher`
 FOREIGN KEY (`teacher_id`)
 REFERENCES `sms`.`teacher` (`id`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `sms`.`enrollments`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sms`.`enrollments` (
 `id` INT NOT NULL AUTO_INCREMENT,
 `enrollment_date` DATE NULL,
 `students_id` INT NOT NULL,
 `courses_id` INT NOT NULL,
 PRIMARY KEY (`id`, `students_id`, `courses_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 `sms`.`students` (`id`)
 ON DELETE NO ACTION
```

```
 ON UPDATE NO ACTION,
 CONSTRAINT `fk_enrollments_courses1`
 FOREIGN KEY (`courses_id`)
 REFERENCES `sms`.`courses` (`id`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `sms`.`payments`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sms`.`payments` (
 `id` INT NOT NULL AUTO_INCREMENT,
 `amount` DECIMAL(10,2) NULL,
 `students_id` INT NOT NULL,
 `payment_date` DATE NULL,
 PRIMARY KEY (`id`, `students_id`),
 INDEX `fk_payments_students1_idx` (`students_id` ASC) ,
 CONSTRAINT `fk_payments_students1`
 FOREIGN KEY (`students_id`)
 REFERENCES `sms`.`students` (`id`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION)
ENGINE = InnoDB;
show tables;
describe students;
-- Inserting data into Students table without spaces in phone numbers
INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number) VALUES
('John', 'Doe', '2000-05-15', 'john.doe@example.com', '1234567890'),
('Jane', 'Smith', '1999-10-20', 'jane.smith@example.com', '4567890123'),
```

```
('Michael', 'Johnson', '2001-03-25', 'michael.johnson@example.com', '7890123456'),
('Emily', 'Brown', '2000-08-10', 'emily.brown@example.com', '0123456789'),
('David', 'Martinez', '1998-12-05', 'david.martinez@example.com', '3456789012'),
('Sarah', 'Williams', '2002-02-18', 'sarah.williams@example.com', '6789012345');
describe teacher;
-- Inserting data into Teacher table without specifying teacher_id
INSERT INTO Teacher (first_name, last_name, email) VALUES
('John', 'Smith', 'john.smith@example.com'),
('Jane', 'Doe', 'jane.doe@example.com'),
('Michael', 'Johnson', 'michael.johnson@example.com'),
('Emily', 'Brown', 'emily.brown@example.com'),
('David', 'Martinez', 'david.martinez@example.com'),
('Sarah', 'Williams', 'sarah.williams@example.com');
-- Inserting data into Payments table without specifying payment_id
describe payments;
INSERT INTO Payments (students_id, amount, payment_date) VALUES
(1, 100.00, '2024-04-01'),
(2, 150.00, '2024-04-02'),
(3, 200.00, '2024-04-03'),
(4, 175.00, '2024-04-04'),
(5, 120.00, '2024-04-05'),
(1, 90.00, '2024-04-06');
-- Inserting data into Courses table without specifying course_id
INSERT INTO Courses (course_name, credits, teacher_id) VALUES
('Mathematics', 4, 1),
('Physics', 3, 2),
('Literature', 3, 3),
('History', 3, 4),
('Biology', 4, 5),
```

```
('Computer Science', 4, 6);
-- Inserting data into Enrollments table without specifying enrollment_id
describe enrollments;
INSERT INTO Enrollments (students_id, courses_id, enrollment_date) VALUES
(1, 1, '2024-04-01'),
(2, 2, '2024-04-02'),
(3, 3, '2024-04-03'),
(4, 4, '2024-04-04'),
(5, 5, '2024-04-05'),
(1, 6, '2024-04-06');
select * from enrollments;
-- task 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');
-- UPDATED
-- 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.
-- Enroll a student in a course
insert into Enrollments (students_id, courses_id, enrollment_date)
VALUES (1, 1, '2024-04-09');
-- UPDATED
-- 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='sarah.will@example.com' where id=6;

-- updated

-- 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 a specific enrollment record

DELETE FROM Enrollments

WHERE students_id = 1

AND courses_id = 1;

-- deleted

-- 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=3 where id=2;

-- updated

-- 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 students where id=2;

-- UPDATED

-- 7. 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='145.00' where id=4;

-- UPDATED

-- 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 sum(p.amount) as total_payment from students s join payments p on s.id=p.students_id where

s.id=3;

/*200.00*/

-- 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.id, c.course_name,count(e.students_id) as count_of_students

from courses c join enrollments e on c.id=e.courses_id group by c.course_name,c.id;

/*6 Computer Science 1

2 Physics 1

3 Literature 1

4 History 1

5 Biology 1*/

-- 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 where s.id not in(select students_id from enrollments);

-- alternate solution

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;

/*SarahWilliams

John Doe

John Doe*/

-- 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.id = e.students_id

join courses c on c.id=e.courses_id;

/*John Doe Computer Science

MichaelJohnson Literature
```

Emily Brown History

David Martinez Biology*/

-- 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(first_name,' ',last_name) as teacher_name ,c.course_name from teacher t

join courses c on t.id=c.teacher_id;

/*John Smith Mathematics

Michael Johnson Physics

Michael Johnson Literature

Emily Brown History

David Martinez Biology

Sarah Williams Computer Science*/

-- 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

from Students s

join Enrollments e on s.id = e.students_id

join courses c on c.id=e.courses_id

where c.course_name='mathematics';

-- 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 distinct concat(first_name,' ',last_name) as student_name

from students s left join payments p on s.id=p.students_id where p.id is null;

/*Sarah Williams

John Doe*/

-- 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

```
select c.course_name from courses c join enrollments e on c.id=e.courses_id where c.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.id,s.first_name, s.last_name

from Students s

join Enrollments e ON s.id = e.students_id

group by s.id, s.first_name, s.last_name

having count(e.courses_id)>1;

-- alternate solution

select distinct e1.students_id, s.first_name, s.last_name

from Enrollments e1

join Enrollments e2 on e1.students_id = e2.students_id and e1.courses_id <> e2.courses_id

join Students s on e1.students_id = s.id;

-- 10 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(first_name,' ',last_name) as teacher_name from teacher t

join courses c on t.id=c.teacher_id where c.id is null;

/*

Task 4. Subquery and its type:*/

-- 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 students_id,amount

from Payments

where amount = (select MAX(amount)from Payments);

/*3 200.00*/

-- 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 id,course_name
from Courses c
where c.id = (
 select courses_id
 from Enrollments
 group by courses_id
 order by count(*) desc
 limit 1);
 /*2 Physics*/
-- 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 id, first_name, last_name
from Students
where (select count(distinct courses_id)from Enrollments)
= (select count(distinct id)from Courses);
-- 6 Retrieve the names of teachers who have not been assigned to any courses.
-- Use subqueries to find teachers with no course assignments.
select  first_name, last_name
from Teacher
where id not in(select distinct teacher_id from Courses);
/*Jane Doe*/
-- 7 Calculate the average age of all students.
-- Use subqueries to calculate the age of each student based on their date of birth.;
-- not able to solve
-- 8 Identify courses with no enrollments. Use subqueries to find courses without enrollment records.
select id, course_name
from Courses
```

```
where id not in (select distinct courses_id from Enrollments
);
/*1 Mathematics*/
-- 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.students_id, e.courses_id, SUM(p.amount) AS total_payments
from Enrollments e
left join Payments p on e.students_id = p.students_id
group by e.students_id, e.courses_id;
/*1 6 190.00
2 2 150.00
3 3 200.00
4 4 145.00
5 5 120.00*/
-- 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 students_id from payments group by students_id having count(*)>1;
/*1*/
-- 11 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 s.id,sum(amount) as total_payment from students s join payments p on s.id=p.students_id group by s.id ;
/*1 190.00
3 200.00
4 145.00
5 120.00*/
-- 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.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.course_name;
/*Mathematics 0
Physics 1
Literature 1
History 1
Biology 1
Computer Science 1*/
-- 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 s.id,avg(amount)as avg_payment_amount from students s join payments p on s.id=p.students_id
group by s.id;
/*1 95.000000
3 200.000000
4 145.000000
5 120.000000*/
```