ASSIGNMENT- STUDENT MANAGEMENT SYSTEM

Task1

-use student_management_system;

- show tables;

- describe course;

- describe enrollments;

- describe payments;

- describe students;

-describe teacher;

- insert into students (first_name,last_name,dob,email,phone_no) values

  ('harry','potter','2002-04-04','harry@hexa.com','7654562'),

  ('ron','weasley','2004-06-02','ronnie@hexa.com','7659263'),

  ('draco','malfoy','2003-05-04','draco@hexa.com','7629345'),

  ('stefan','salvatore','2002-01-01','salvatore@hexa.com','7125434'),

  ('enzo','john','2000-08-18','enzo@hexa.com','6007663');

-insert into teacher (first_name,last_name,email) values

 ('elena','gilbert','elena@hexa.com'),

 ('bonnie','bennet','bonnie@hexa.com'),

 ('klaus','mikelson','klaus@hexa.com'),

 ('caroline','forbes','care@hexa.com');

-insert into course (course_name,credits,teacher_id) values

 ('os',3,2),

 ('sql',2,1),

 ('python',4,2),

 ('cloud',2,3),

 ('java',1,3),

 ('c++',3,4);

-select* from course;

-insert into payments(amount,payment_date,students_id) values

 (10000,'2024-02-02',2),

 (20000,'2024-10-04',4),

 (15000,'2024-12-23',3),

```
 (22000,'2024-01-05',1 );

-insert into enrollments (enrollment_date,students_id,course_id) values

 ('2022-04-05',1,13),

 ('2020-05-05',2,14),

 ('2022-04-05',3,15),

 ('2023-02-05',5,15),

 ('2024-03-05',4,13),

 ('2022-06-05',2,16),

 ('2020-05-05',1,17);
```

Task-2

```
-update teacher set email='caroline@hex.com' where id=4;

-delete from enrollments where students_id=4 and course_id=11;

-update course set teacher_id=1 where course_name='java';

-delete from enrollments where students_id=5;

-delete from students where first_name='enzo';

-update payments set amount='12000' where id=1;
```

Task-3

-- 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 = p.students_id

GROUP BY s.id;

/*

harry   potter   22000

ron     weasley 12000

draco   malfoy  15000

stefan  salvatore          20000

*/
```

-- 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.students_id) AS enrolled_students

FROM course c
```

```
LEFT JOIN enrollments e ON c.id = e.course_id

GROUP BY c.id;

/*

os      2

sql     1

python  1

cloud   1

java    1

c++     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

JOIN enrollments e ON s.id = e.students_id

WHERE e.students_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.course_name

FROM students s

JOIN enrollments e ON s.id = e.students_id

JOIN course c ON e.course_id = c.id;

 /*

 harry   potter    os

harry    potter    java

ron      weasleysql

ron      weasleycloud

draco    malfoy  python

stefan   salvatore         os

 */
```

 -- 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 course c ON t.id = c.teacher_id;

/*

elena	gilbert	sql

elena	gilbert	java

bonnie	bennet	os

bonnie	bennet	python

klaus	mikelson		cloud

caroline	forbes	c++

*/
```

-- 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 course c ON e.course_id = c.id

WHERE c.course_name = 'java';

/*

harry	potter	2020-05-05

*/
```

-- 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

JOIN payments p ON s.id = p.students_id

WHERE p.students_id 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.course_name

FROM course c

JOIN enrollments 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 s.first_name, s.last_name

FROM students s

JOIN (

   SELECT students_id

   FROM enrollments

   GROUP BY students_id

   HAVING COUNT(course_id) > 1

) AS multi_enroll ON s.id = multi_enroll.students_id;

/*

harry     potter

ron       weasley

*/
```

-- 10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher"

```
SELECT t.first_name, t.last_name

FROM teacher t

JOIN course c ON t.id = c.teacher_id

WHERE c.teacher_id IS NULL;
```

-- TASK-4

-- 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(students_id) AS student_count

   FROM enrollments

   GROUP BY course_id

) AS course_student_counts;

/*

1.2000

*/
```

-- 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 s.first_name, s.last_name

FROM students s

JOIN payments p ON s.id = p.students_id

WHERE p.amount = (SELECT MAX(amount) FROM payments);

/*

harry    potter

*/
```

-- 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.course_name

FROM course c

JOIN (

   SELECT course_id, COUNT(students_id) AS enrollment_count

   FROM enrollments

   GROUP BY course_id

   ORDER BY enrollment_count DESC

   LIMIT 1

) AS max_enrollment ON c.id = max_enrollment.course_id;

/*

os

*/
```

-- 4. 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.last_name, SUM(p.amount) AS total_payments

FROM teacher t

JOIN course c ON t.id = c.teacher_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 s.first_name, s.last_name

FROM students s

WHERE (SELECT COUNT(DISTINCT course_id) FROM enrollments) = (

   SELECT COUNT(DISTINCT course_id) FROM enrollments WHERE students_id = 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 t.first_name, t.last_name

FROM teacher t

LEFT JOIN course c ON t.id = c.teacher_id

WHERE c.teacher_id IS NULL;

-- 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(YEAR(CURDATE()) - YEAR(dob)) AS avg_age

FROM students;

/*

21.2500

*/

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

SELECT c.course_name

FROM course c

LEFT JOIN enrollments e ON c.id = e.course_id

WHERE e.course_id IS NULL;

/*

c++

*/

-- 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.course_name, SUM(p.amount) AS total_payments

FROM students s

JOIN enrollments e ON s.id = e.students_id

JOIN course c ON e.course_id = c.id

JOIN payments p ON s.id = p.students_id AND e.course_id = p.course_id

GROUP BY s.id, c.id;


-- 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.first_name, s.last_name
FROM students s
JOIN (
    SELECT students_id
    FROM payments
    GROUP BY students_id
    HAVING COUNT(id) > 1
) AS multi_payment ON s.id = multi_payment.students_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 s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM students s
LEFT JOIN payments p ON s.id = p.students_id
GROUP BY s.id;
/*
potter    harry    22000
weasleyron         12000
malfoy  draco      15000
salvatore          stefan   20000
*/
```

/*

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 c.course_name, COUNT(e.students_id) AS enrolled_students
FROM course c
LEFT JOIN enrollments e ON c.id = e.course_id
GROUP BY c.id;
/*
os      2
sql     1
python  1
```

```
cloud    1
java     1
c++      0
*/
```