

Student Information System

SQL

Task 1.Database Design

1. Create the database named "SISDB"

```
CREATE DATABASE sisdb;  
USE sisdb;
```

2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

a. Students table

```
CREATE TABLE `sisdb`.`students` (  
  `student_id` INT NOT NULL,  
  `first_name` VARCHAR(45) NULL,  
  `last_name` VARCHAR(45) NULL,  
  `date_of_birth` DATE NULL,  
  `email` VARCHAR(45) NULL,  
  `phone_number` VARCHAR(45) NULL,  
  PRIMARY KEY (`student_id`));
```

b. Courses table

```
CREATE TABLE courses (  
  course_id INT PRIMARY KEY,  
  course_name VARCHAR(100),  
  credits INT,  
  teacher_id INT,  
  FOREIGN KEY (teacher_id) REFERENCES teacher(teacher_id)  
);  
use sisdb;
```

c. Enrollments table

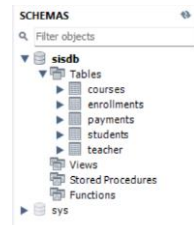
```
use sisdb;  
CREATE TABLE enrollments (  
  enrollment_id INT PRIMARY KEY,  
  student_id INT,  
  course_id INT,  
  enrollment_date DATE,  
  FOREIGN KEY (student_id) REFERENCES students(student_id),  
  FOREIGN KEY (course_id) REFERENCES courses(course_id)  
);
```

d. Teacher table

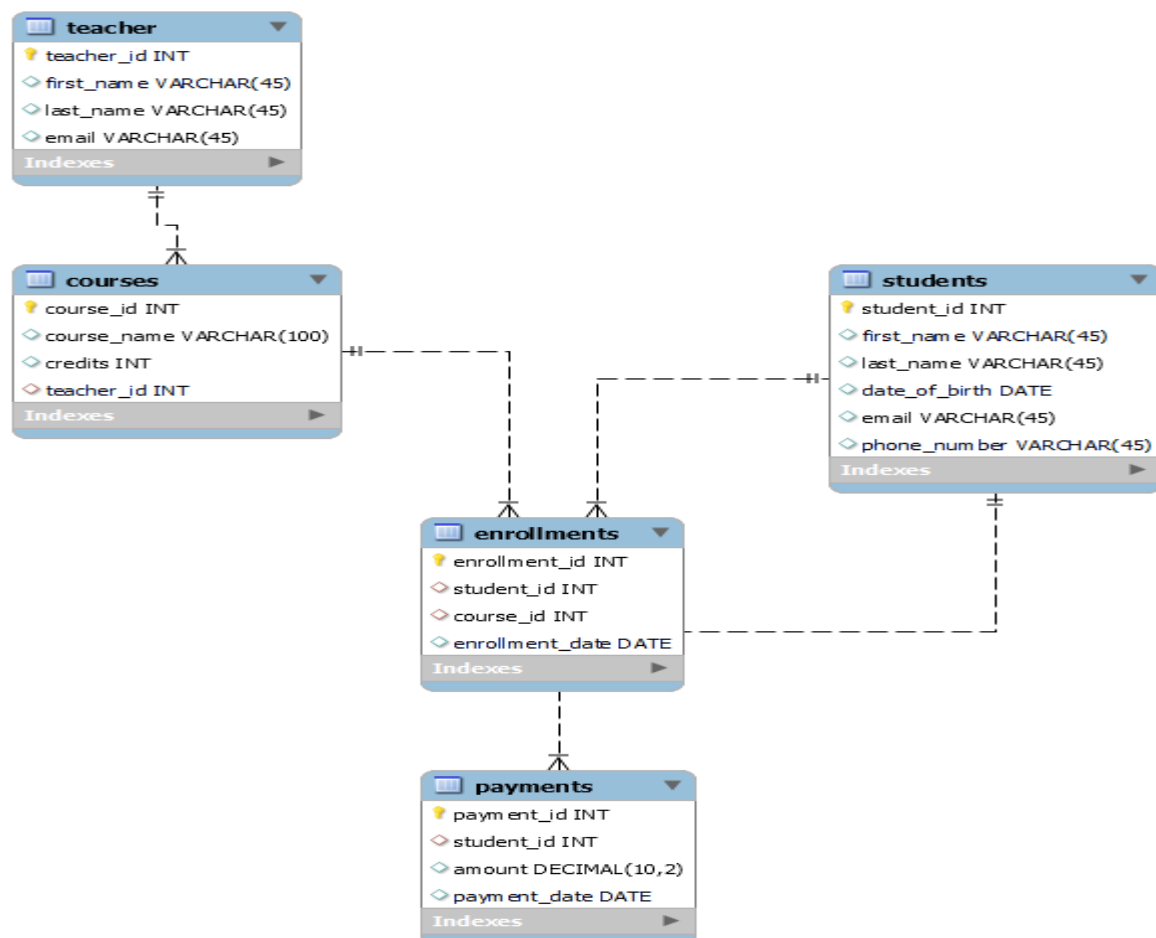
```
CREATE TABLE `sisdb`.`teacher` (  
  `teacher_id` INT NULL,  
  `first_name` VARCHAR(45) NULL,  
  `last_name` VARCHAR(45) NULL,  
  `email` VARCHAR(45) NULL,  
  PRIMARY KEY (`teacher_id`));
```

e. Payments table

```
use sisdb;
CREATE TABLE payments (
  payment_id INT PRIMARY KEY,
  student_id INT,
  amount DECIMAL(10, 2),
  payment_date DATE,
  FOREIGN KEY (student_id) REFERENCES students(student_id)
);
```



3. Create an ERD (Entity Relationship Diagram) for the database.



4. Primary key and Foreign key constraints are created in table

5. Insert at least 10 sample records into each of the following tables.

i Students

```
1 • SELECT * FROM sisdb.students;
```

student_id	first_name	last_name	date_of_birth	email	phone_number
102	divya	sri	2000-12-12	divya34@gmail.com	9765834215
103	naga	sri	2002-03-30	nagasri123@gmail.com	6754389076
104	navya	lakshmi	2001-02-23	navyag89@gmail.com	7893412654
105	sirisha	s	2001-08-21	sirisha23@gmail.com	6754390823
106	vaisnavi	reddy	2000-11-28	vaishu38@gmail.com	7653900231
107	supraja	gowd	2002-02-16	supraja09@gmail.com	6702245661
108	lavanya	konidela	2001-05-30	lavanya67@gmail.com	8765490123
109	harshita	mutta	2002-09-19	harshita81@gmail.com	6754344770
110	harsh	sri	2000-03-15	harsh45@gmail.com	8065614571

ii Courses

```
1  
2  
3 • select * from sisdb.courses;  
4
```

course_id	course_name	credits	teacher_id
11	Operating S...	3	2
12	DBMS	3	1
13	ML	4	3
14	AI	3	7
15	Python	3	6
16	Java	3	4
17	English	2	2
18	Maths	2	3
19	Computer G...	3	5

iii. Enrollments

```
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('201', '102', '11', '2023-08-30');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('202', '101', '13', '2023-08-29');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('203', '101', '13', '2023-07-18');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('204', '103', '12', '2023-07-19');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('205', '104', '12', '2023-09-24');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('206', '105', '16', '2023-09-30');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('207', '104', '18', '2023-08-28');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('208', '107', '14', '2023-08-23');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('209', '108', '19', '2023-09-23');  
INSERT INTO `sisdb`.`enrollments` (`enrollment_id`, `student_id`, `course_id`, `enrollment_date`) VALUES ('210', '106', '14', '2023-09-24');
```

1 • `SELECT * FROM sisdb.enrollments;`

enrollment_id	student_id	course_id	enrollment_date
201	102	11	2023-08-30
202	101	13	2023-08-29
203	101	13	2023-07-18
204	103	12	2023-07-19
205	104	12	2023-09-24
206	105	16	2023-09-30
207	104	18	2023-08-28
208	107	14	2023-08-23
209	108	19	2023-09-23
210	106	14	2023-08-24

enrollments 2 x

iv. Teacher

1 • `SELECT * FROM sisdb.teacher;`

teacher_id	first_name	last_name	email
1	shyamala	rao	shy...
2	sunitha	p	sun...
3	kusuma	sri	kus...
5	amrutha	k	amr...
4	raghav	reddy	rag...
3	vishnav	shetty	sett...
6	pavan	kalyan	kaly...
7	nagendra	rao	nag...
6	suresh	reddy	sure...

teacher 2 x

v. Payments

1 • `SELECT * FROM sisdb.payments;`

payment_id	student_id	amount	payment_date
31	102	10000...	2023-01-02
32	102	10000...	2023-01-02
33	104	20000...	2023-02-03
34	104	30000...	2023-09-16
35	105	15000...	2023-09-18
36	109	30000...	2023-08-19
37	108	15000...	2023-11-22
38	109	20000...	2023-09-16
39	108	20000...	2023-08-19
40	108	10000...	2023-08-19

teacher 2 payments 3 x

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:

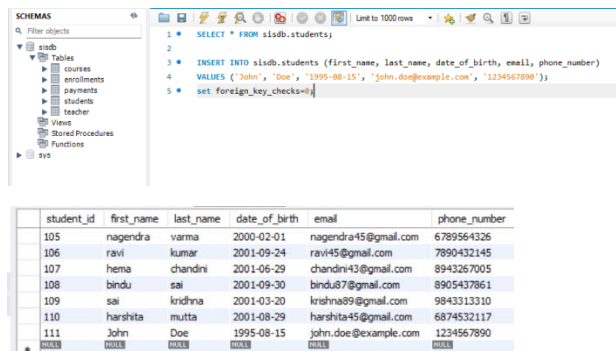
a. First Name: John

b. Last Name: Doe

c. Date of Birth: 1995-08-15

d. Email: john.doe@example.com

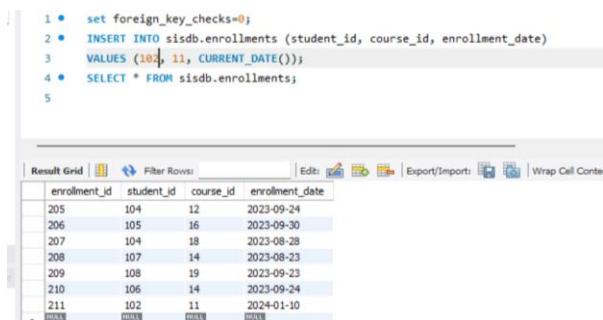
e. Phone Number: 1234567890



The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' pane lists tables: sisdb, courses, enrollments, payments, students, teacher, Views, Stored Procedures, Functions, and sys. The main area displays a SQL query with five lines: 1. SELECT * FROM sisdb.students; 2. 3. INSERT INTO sisdb.students (first_name, last_name, date_of_birth, email, phone_number) 4. VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890'); 5. set foreign_key_checks=0; Below the query, a table of results is shown with columns: student_id, first_name, last_name, date_of_birth, email, and phone_number. The table contains 11 rows, with the last row (111) being the newly inserted student John Doe.

student_id	first_name	last_name	date_of_birth	email	phone_number
105	nagendra	varma	2000-02-01	nagendra45@gmail.com	6789564326
106	ravi	kumar	2001-09-24	ravi45@gmail.com	7890432145
107	hema	chandini	2001-06-29	chendini43@gmail.com	8943267005
108	bindu	sa	2001-09-30	bindu87@gmail.com	8905437861
109	sai	krishna	2001-03-20	krishna89@gmail.com	9843313310
110	harshita	muttha	2001-08-29	harshita45@gmail.com	6874532117
111	John	Doe	1995-08-15	john.doe@example.com	1234567890

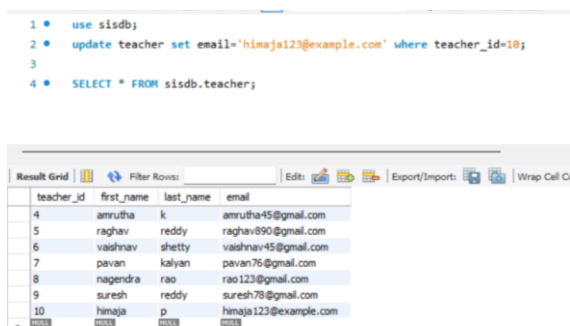
2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.



The screenshot shows a database management tool interface. The main area displays a SQL query with five lines: 1. set foreign_key_checks=0; 2. INSERT INTO sisdb.enrollments (student_id, course_id, enrollment_date) 3. VALUES (104, 11, CURRENT_DATE()); 4. SELECT * FROM sisdb.enrollments; 5. Below the query, a 'Result Grid' is shown with columns: enrollment_id, student_id, course_id, and enrollment_date. The grid contains 7 rows, with the last row (211) being the newly inserted enrollment record for student 104 in course 11 on 2024-01-10.

enrollment_id	student_id	course_id	enrollment_date
205	104	12	2023-09-24
206	105	16	2023-09-30
207	104	18	2023-08-28
208	107	14	2023-08-23
209	108	19	2023-09-23
210	106	14	2023-09-24
211	102	11	2024-01-10

3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.



The screenshot shows a database management tool interface. The main area displays a SQL query with four lines: 1. use sisdb; 2. update teacher set email='himaja123@example.com' where teacher_id=10; 3. 4. SELECT * FROM sisdb.teacher; Below the query, a 'Result Grid' is shown with columns: teacher_id, first_name, last_name, and email. The grid contains 10 rows, with the last row (10) being the updated teacher record for teacher_id 10, first_name 'himaja', last_name 'p', and email 'himaja123@example.com'.

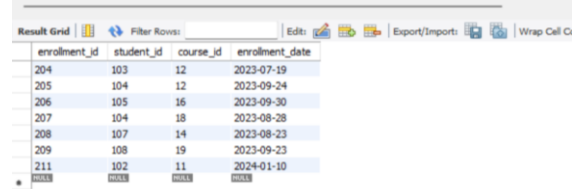
teacher_id	first_name	last_name	email
4	amrutha	k	amrutha45@gmail.com
5	raghav	reddy	raghav890@gmail.com
6	vaishnav	shetty	vaishnav45@gmail.com
7	pavan	kalyan	pavan76@gmail.com
8	nagendra	rao	rao123@gmail.com
9	suresh	reddy	suresh78@gmail.com
10	himaja	p	himaja123@example.com

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.

```

1
2 • DELETE FROM enrollments WHERE student_id = 106 AND course_id = 14;
3
4 • SELECT * FROM sisdb.enrollments;
5
6

```



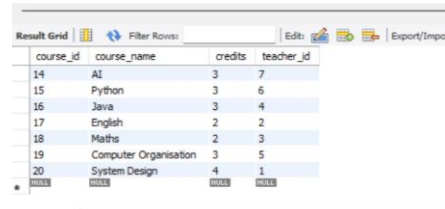
enrollment_id	student_id	course_id	enrollment_date
204	103	12	2023-07-19
205	104	12	2023-09-24
206	105	16	2023-09-30
207	104	18	2023-08-28
208	107	14	2023-08-23
209	108	19	2023-09-23
211	102	11	2024-01-10

5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.

```

1 • UPDATE courses SET teacher_id = 1 WHERE course_id = 20;
2
3 • SELECT * FROM sisdb.courses;

```



course_id	course_name	credits	teacher_id
14	AI	3	7
15	Python	3	6
16	Java	3	4
17	English	2	2
18	Maths	2	3
19	Computer Organisation	3	5
20	System Design	4	1

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

```

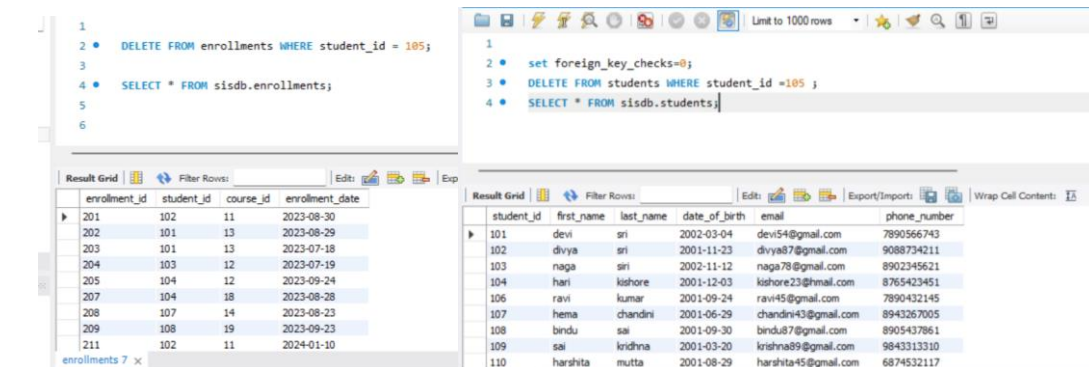
1
2 • DELETE FROM enrollments WHERE student_id = 105;
3
4 • SELECT * FROM sisdb.enrollments;
5
6

```

```

1
2 • set foreign_key_checks=0;
3 • DELETE FROM students WHERE student_id =105 ;
4 • SELECT * FROM sisdb.students;

```



enrollment_id	student_id	course_id	enrollment_date
201	102	11	2023-08-30
202	101	13	2023-08-29
203	101	13	2023-07-18
204	103	12	2023-07-19
205	104	12	2023-09-24
207	104	18	2023-08-28
208	107	14	2023-08-23
209	108	19	2023-09-23
211	102	11	2024-01-10

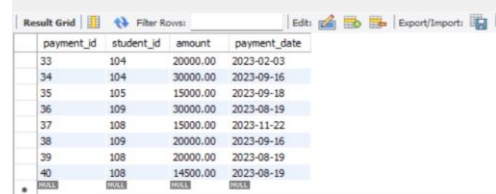
student_id	first_name	last_name	date_of_birth	email	phone_number
101	devi	sri	2002-03-04	devi54@gmail.com	7890566743
102	divya	sri	2001-11-23	divya87@gmail.com	9088734211
103	naga	sri	2002-11-12	naga78@gmail.com	8902345621
104	hari	kishore	2001-12-03	kishore23@gmail.com	8765423451
106	ravi	kumar	2001-09-24	ravi45@gmail.com	7890432145
107	hema	chandini	2001-06-29	chandini43@gmail.com	8943267005
108	bindu	sai	2001-09-30	bindu87@gmail.com	8905437861
109	sai	kridhna	2001-03-20	krishna89@gmail.com	9843313310
110	harshita	mutta	2001-08-29	harshita45@gmail.com	6874532117

7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.

```

2 • UPDATE payments SET amount = 14500 WHERE payment_id = 40;
3
4 • SELECT * FROM sisdb.payments;

```



payment_id	student_id	amount	payment_date
33	104	20000.00	2023-02-03
34	104	30000.00	2023-09-16
35	105	15000.00	2023-09-18
36	109	30000.00	2023-08-19
37	108	15000.00	2023-11-22
38	109	20000.00	2023-09-16
39	108	20000.00	2023-08-19
40	108	14500.00	2023-08-19

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.

```
1 • SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
2 FROM students s JOIN payments p ON s.student_id = p.student_id WHERE s.student_id = 104
3 GROUP BY s.student_id, s.first_name, s.last_name;
4
```

student_id	first_name	last_name	total_payments
104	hari	kishore	50000.00

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.

```
2 • SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrolled_students FROM courses c
3 LEFT JOIN enrollments e ON c.course_id = e.course_id GROUP BY c.course_id, c.course_name;
4
```

course_id	course_name	enrolled_students
11	Operating System	2
12	DBMS	2
13	ML	2
14	AI	1
15	Python	0
16	Java	0
17	English	0
18	Maths	1
19	Computer Organisation	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

```
2 • SELECT s.student_id, s.first_name, s.last_name FROM students s
3 LEFT JOIN enrollments e ON s.student_id = e.student_id
4 WHERE e.student_id IS NULL;
5
```

student_id	first_name	last_name
106	ravi	kumar
109	sai	kridhna
110	harshita	mutta
111	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.

```
2 • SELECT s.first_name, s.last_name, c.course_name FROM students s
3 JOIN enrollments e ON s.student_id = e.student_id
4 JOIN courses c ON e.course_id = c.course_id;
5
```

first_name	last_name	course_name
divya	sri	Operating System
devi	sri	ML
devi	sri	ML
naga	siri	DBMS
hari	kishore	DBMS
hari	kishore	Maths
hema	chandini	AI
bindu	sai	Computer Organisation
divya	sri	Operating 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.

```

1
2 * SELECT t.first_name AS teacher_first_name, t.last_name AS teacher_last_name,c.course_name
3 FROM teacher t
4 JOIN courses c ON t.teacher_id = c.teacher_id;
5
6

```

teacher_first_name	teacher_last_name	course_name
shyamala	rao	DBMS
shyamala	rao	System Design
sunitha	p	Operating System
sunitha	p	English
kusuma	sei	ML
kusuma	sei	Maths
amrutha	k	Java
raghav	reddy	Computer Organisation
vashnav	shetty	Python

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.

```

3 * SELECT s.first_name, s.last_name, e.enrollment_date FROM students s
4 JOIN enrollments e ON s.student_id = e.student_id
5 JOIN courses c ON e.course_id = c.course_id
6 WHERE c.course_id = 12;
7

```

first_name	last_name	enrollment_date
naga	sri	2023-07-19
hari	kishore	2023-09-24

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.

```

1 * SELECT s.first_name,s.last_name FROM students s
2 LEFT JOIN payments p ON s.student_id = p.student_id
3 WHERE p.student_id IS NULL;
4
5
6
7

```

first_name	last_name
devi	sri
naga	sri
ravi	kumar
hema	chandini
harshita	mutta
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 enrollment records.

```

1 * SELECT c.course_id,c.course_name FROM courses c
2 LEFT JOIN enrollments e ON c.course_id = e.course_id
3 WHERE e.course_id IS NULL;
4
5
6
7

```

course_id	course_name
15	Python
16	Java
17	English
20	System Design

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.

```

1 • SELECT e1.student_id,s.first_name,s.last_name FROM enrollments e1
2 JOIN enrollments e2 ON e1.student_id = e2.student_id AND e1.course_id != e2.course_id
3 JOIN students s ON e1.student_id = s.student_id
4 GROUP BY e1.student_id, s.first_name, s.last_name
5 HAVING COUNT(DISTINCT e1.course_id) > 1;
6
7

```

student_id	first_name	last_name
104	hari	kishore

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.

```

1 • SELECT t.teacher_id,t.first_name,t.last_name FROM teacher t
2 LEFT JOIN courses c ON t.teacher_id = c.teacher_id
3 WHERE c.course_id IS NULL;
4
5
6
7

```

teacher_id	first_name	last_name
8	nagendra	rao
9	suresh	reddy
10	himaja	p

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 • SELECT c.course_id,c.course_name,AVG(enrollment_count) AS average_students_enrolled
3 FROM courses c LEFT JOIN
4 (SELECT course_id,COUNT(DISTINCT student_id) AS enrollment_count
5 FROM enrollments GROUP BY course_id) AS subquery
6 ON c.course_id = subquery.course_id
7 GROUP BY c.course_id, c.course_name;

```

course_id	course_name	average_students_enrolled
11	Operating System	1.0000
12	DBMS	2.0000
13	ML	1.0000
14	AI	1.0000
15	Python	NULL
16	Java	NULL
17	English	NULL
18	Maths	1.0000

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.

```

2 FROM students s
3 JOIN payments p ON s.student_id = p.student_id
4 WHERE p.amount = (SELECT MAX(amount) FROM payments);
5
6
7
8

```

student_id	first_name	last_name	highest_payment
104	hari	kishore	30000.00
109	sai	krishna	30000.00

3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.

```

1 * SELECT c.course_id,c.course_name,enrollment_count
2 FROM courses c JOIN(
3     SELECT course_id,COUNT(DISTINCT student_id) AS enrollment_count
4     FROM enrollments GROUP BY course_id) AS subquery
5 ON c.course_id = subquery.course_id
6 WHERE enrollment_count = (SELECT MAX(enrollment_count)
7     FROM (SELECT COUNT(DISTINCT student_id) AS enrollment_count
8     FROM enrollments GROUP BY course_id) AS max_enrollment);
9
10

```

course_id	course_name	enrollment_count
12	DBMS	2

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

```

1 * SELECT t.teacher_id, t.first_name AS teacher_first_name, t.last_name AS teacher_last_name, COALESCE(SUM(p.amount), 0) AS total_payments
2 FROM Teacher t
3 LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
4 LEFT JOIN Enrollments e ON c.course_id = e.course_id
5 LEFT JOIN Payments p ON e.student_id = p.student_id
6 GROUP BY t.teacher_id, t.first_name, t.last_name;
7
8

```

teacher_id	teacher_first_name	teacher_last_name	total_payments
1	shyamala	rao	50000.00
2	sunitha	p	40000.00
3	kusuma	sei	50000.00
4	amrutha	k	0.00
5	raghav	reddy	49500.00
6	vashnav	shetty	0.00
7	pavan	kalyan	0.00
8	nagendra	rao	0.00
9	suresh	reddy	0.00
10	himaja	p	0.00

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

```

1 * SELECT s.student_id,s.first_name,s.last_name FROM students s
2 WHERE (SELECT COUNT(DISTINCT course_id) FROM courses)
3 =(SELECT COUNT(DISTINCT course_id) FROM enrollments e WHERE e.student_id = s.student_id);
4
5
6
7
8
9
10

```

student_id	first_name	last_name
112	surendra	kumar

6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

```

1 * SELECT t.teacher_id,t.first_name,t.last_name
2 FROM teacher t
3 WHERE NOT EXISTS (SELECT 1 FROM courses c
4     WHERE c.teacher_id = t.teacher_id);
5
6
7
8
9
10

```

teacher_id	first_name	last_name
8	nagendra	rao
9	suresh	reddy
10	himaja	p

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

```

1 • SELECT AVG(age) AS average_age FROM
2   (SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
3    FROM students) AS subquery;
4
5
6
7
8
9
10

```

average_age
22.4000

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

```

1 • SELECT c.course_id,c.course_name
2   FROM courses c WHERE
3   NOT EXISTS (SELECT 1 FROM enrollments e
4               WHERE e.course_id = c.course_id);
5
6
7
8
9
10

```

course_id	course_name
15	Python
16	Java
17	English
20	System Design

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

```

1 • SELECT s.student_id, s.first_name, s.last_name
2   FROM Students s
3  JOIN Payments p ON s.student_id = p.student_id
4  GROUP BY s.student_id, s.first_name, s.last_name
5  HAVING COUNT(p.payment_id) > 1;

```

student_id	first_name	last_name
102	divya	sri
104	hari	kishore
108	bindu	sai
109	sai	kridhna

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.

```

1 • SELECT s.student_id,s.first_name,s.last_name
2   FROM students s JOIN payments p ON s.student_id = p.student_id
3  GROUP BY s.student_id, s.first_name, s.last_name
4  HAVING COUNT(p.payment_id) > 1;
5
6
7
8
9
10

```

student_id	first_name	last_name
102	divya	sri
104	hari	kishore
108	bindu	sai
109	sai	kridhna

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.

```

1 • SELECT s.student_id,s.first_name,s.last_name,
2     COALESCE(SUM(p.amount), 0) AS total_payments
3 FROM students s
4 LEFT JOIN payments p ON s.student_id = p.student_id
5 GROUP BY s.student_id, s.first_name, s.last_name;
6
7
8
9
10
11
12

```

student_id	first_name	last_name	total_payments
101	devi	sri	0.00
102	divya	sri	20000.00
103	naga	sri	0.00
104	hari	kishore	50000.00
106	ravi	kumar	0.00
107	hema	chandni	0.00
108	bindu	sai	49500.00

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.

```

1 • SELECT c.course_id,c.course_name,
2     COUNT(e.student_id) AS enrolled_students
3 FROM courses c LEFT JOIN
4     enrollments e ON c.course_id = e.course_id
5 GROUP BY c.course_id, c.course_name;
6
7
8
9
10
11
12

```

course_id	course_name	enrolled_students
11	Operating System	2
12	DBMS	2
13	ML	2
14	AI	1
15	Python	0
16	Java	0
17	English	0

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.

```

1 • SELECT s.student_id,s.first_name,s.last_name,
2     AVG(p.amount) AS average_payment_amount
3 FROM students s LEFT JOIN
4     payments p ON s.student_id = p.student_id
5 GROUP BY s.student_id, s.first_name, s.last_name;
6
7
8
9
10
11
12

```

student_id	first_name	last_name	average_payment_amount
101	devi	sri	0.000000
102	divya	sri	10000.000000
103	naga	sri	0.000000
104	hari	kishore	25000.000000
106	ravi	kumar	0.000000
107	hema	chandni	0.000000
108	bindu	sai	16500.000000