ASSIGNMENT

Name: Chandana Priya Reddy Veeramreddy

Assignment: Student Information System

Task -1:

1. Create the database named "SISDB”

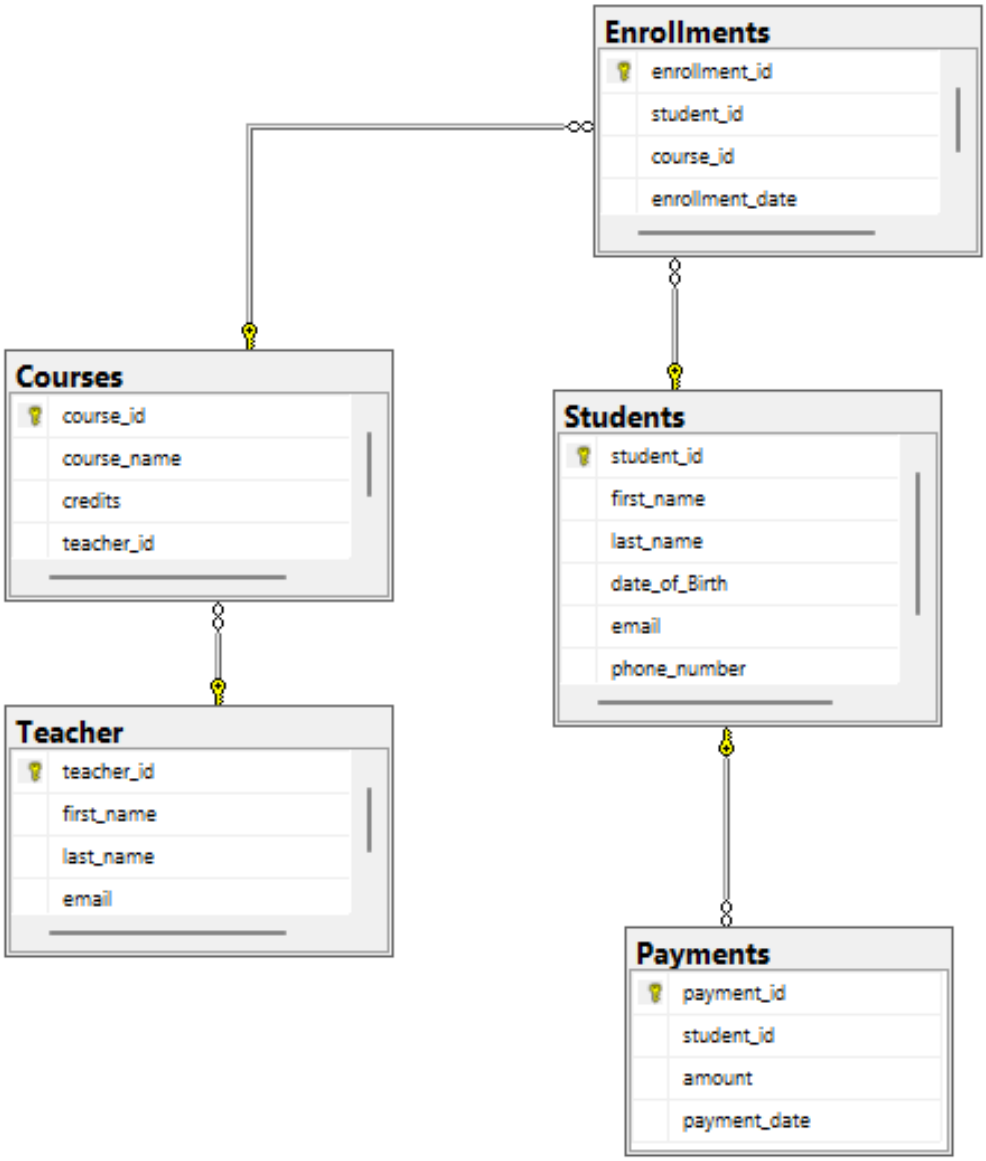
```
create database SISDB;
use SISDB;
```

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.

```
/*Creating Tables*/

/*1.Students
• student_id (Primary Key)
• first_name
• last_name
• date_of_birth
• email
• phone_number*/

create table Students(
student_id varchar(5) primary key not null,
first_name varchar(30),last_name  varchar(50),
date_of_Birth date,email varchar(100) not null,
phone_number BIGINT);


/*2.Teacher
• teacher_id (Primary Key)
• first_name
• last_name
• email*/
```

```sql
create table Teacher(
teacher_id varchar(5) primary key not null, first_name varchar(50),
last_name varchar(50),email varchar(100));


/*3. Courses
• course_id (Primary Key)
• course_name
• credits
• teacher_id (Foreign Key)*/

create table Courses(
course_id varchar(5) primary key not null,
course_name varchar(10),
credits int, teacher_id varchar(5),
Foreign key(teacher_id) REFERENCES Teacher(teacher_id));

/*4. Enrollments
• enrollment_id (Primary Key)
• student_id (Foreign Key)
• course_id (Foreign Key)
• enrollment_date*/

create table Enrollments(
enrollment_id varchar(5) not null primary key,student_id varchar(5),
foreign key(student_id) references Students(student_id),
course_id varchar(5), foreign key(course_id) references Courses(course_id),
enrollment_date date);

/*5 Payments
• payment_id (Primary Key)
• student_id (Foreign Key)
• amount
• payment_date*/

create table Payments(
payment_id varchar(5) not null primary key, student_id varchar(5),
foreign key(student_id) references Students(student_id),
amount int,payment_date date);
```

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



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

```
alter table Enrollments
add constraint Enrollments_studentid_fk
FOREIGN KEY (student_id) REFERENCES Students(student_id) on delete cascade;

alter table payments
add constraint payments_studentid_fk
FOREIGN KEY (student_id) REFERENCES Students(student_id) on delete cascade;
```

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

i. Students

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

```
INSERT INTO Students values
('S1', 'Emily', 'Chen', '1999-02-15', 'emilyc@gmail.com', 1234567890),
('S2', 'Ryan', 'Thompson', '2001-10-22', 'ryan.t@gmail.com', 9876543210),
('S3','Priya','Singh','2000-08-12','psn@gmail.com',4567894321),
('S4', 'Michael', 'Lee', '1998-05-10', 'mike.lee@yahoo.com', 7418529630),
('S5', 'Sophia', 'Patel', '2002-01-05', 'sophia.patel@hotmail.com', 3692581470),
('S6', 'David', 'Kim', '1997-09-20', 'david.kim@outlook.com', 8529637410),
('S7', 'Olivia', 'Brown', '2000-04-15', 'olivia.brown@gmail.com', 1357924680),
('S8', 'Ethan', 'Hall', '1999-11-25', 'ethan.hall@yahoo.com', 2468013579),
('S9', 'Ava', 'Martin', '2001-06-01', 'ava.martin@hotmail.com', 9638527410),
('S10', 'Liam', 'Davis', '1998-03-10', 'liam.davis@outlook.com', 5792468103);

insert into Teacher values
('T1','Ram','Patel','ram@edu.in'),
('T2','Curie','Joe','joe@school.in'),
('T3','Sita','Singh','s@gmail.com'),
('T4','Rama','Martin','rama@edu.org'),
('T5','Raghu','Reddy','rr@email.com'),
('T6','Vishwa','Sen','sen@school.edu'),
('T7','Pavani','Rose','prose@school.org'),
('T8','Anitha','Rathode','anitha@gmail.com'),
('T9','Teju','Lee','lee@edu.in'),
('T10','Nani','Saheb','nani@org.in');

insert into Courses values
('C1','English',3,'T2'),
('C2','Maths',2,'T5'),
('C3','Science',1,'T6'),
('C4','C',3,'T1'),
('C5','Python',5,'T7'),
('C6','C++',4,'T8'),
('C7','Java',6,'T10'),
```

```
('C8','SQL',4,'T9'),
('C9','C#',2,'T3'),
('C10','Skills',1,'T4');

insert into Enrollments values

  ('E1', 'S1', 'C1', '2022-01-01'),
  ('E2', 'S1', 'C2', '2022-01-15'),
  ('E3', 'S2', 'C3', '2022-02-01'),
  ('E4', 'S3', 'C1', '2022-03-01'),
  ('E5', 'S4', 'C4', '2022-04-01'),
  ('E6', 'S5','C6', '2022-05-01'),
  ('E7', 'S1', 'C3', '2022-06-01'),
  ('E8', 'S2', 'C10', '2022-07-01'),
  ('E9', 'S3', 'C2', '2022-08-01'),
  ('E10', 'S4', 'C5', '2022-09-01');


insert into Payments values
  ('P1', 'S1', 100, '2022-01-05'),
  ('P2', 'S1', 500, '2022-01-20'),
  ('P3', 'S2', 200, '2022-02-10'),
  ('P4', 'S3', 800, '2022-03-15'),
  ('P5', 'S4', 150, '2022-04-20'),
  ('P6', 'S5', 250, '2022-05-25'),
  ('P7', 'S1', 300, '2022-06-15'),
  ('P8', 'S2', 400, '2022-07-10'),
  ('P9', 'S3', 900, '2022-08-15'),
  ('P10', 'S4', 200, '2022-09-10');


   -----displaying the data

select * from Students;
select * from Courses;
select * from Teacher;
select * from Payments;
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(student_id,first_name,last_name,date_of_Birth,email,phone_number)
values('S11','John','Doe','1995-08-15','john.doe@example.com',1234567890);

select * from Students;
```

Output:

Results | Messages

	student_id	first_name	last_name	date_of_Birth	email	phone_number
1	S1	Emily	Chen	1999-02-15	emilyc@gmail.com	1234567890
2	S10	Liam	Davis	1998-03-10	liam.davis@outlook.com	5792468103
3	S11	John	Doe	1995-08-15	john.doe@example.com	1234567890
4	S2	Ryan	Thompson	2001-10-22	ryan.t@gmail.com	9876543210
5	S3	Priya	Singh	2000-08-12	psn@gmail.com	4567894321
6	S4	Michael	Lee	1998-05-10	mike.lee@yahoo.com	7418529630
7	S5	Sophia	Patel	2002-01-05	sophia.patel@hotmail.com	3692581470
8	S6	David	Kim	1997-09-20	david.kim@outlook.com	8529637410
9	S7	Olivia	Brown	2000-04-15	olivia.brown@gmail.com	1357924680
10	S8	Ethan	Hall	1999-11-25	ethan.hall@yahoo.com	2468013579
11	S9	Ava	Martin	2001-06-01	ava.martin@hotmail.com	9638527410

```
/*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_id,student_id,course_id,enrollment_date)
values('E11','S7','C6','2022-10-10');

select * from Enrollments;
```

Output:

Results Messages

	enrollment_id	student_id	course_id	enrollment_date
1	E1	S1	C1	2022-01-01
2	E10	S4	C5	2022-09-01
3	E11	S7	C6	2022-10-10
4	E2	S1	C2	2022-01-15
5	E3	S2	C3	2022-02-01
6	E4	S3	C1	2022-03-01
7	E5	S4	C4	2022-04-01
8	E6	S5	C6	2022-05-01
9	E7	S1	C3	2022-06-01
10	E8	S2	C10	2022-07-01
11	E9	S3	C2	2022-08-01

```
/* 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='raghu@gmail.com' where first_name='raghu';
select * from Teacher;
```

Output:

Results Messages

	teacher_id	first_name	last_name	email
1	T1	Ram	Patel	ram@edu.in
2	T10	Nani	Saheb	nani@org.in
3	T2	Curie	Joe	joe@school.in
4	T3	Sita	Singh	s@gmail.com
5	T4	Rama	Martin	rama@edu.org
6	T5	Raghu	Reddy	raghu@gmail.com
7	T6	Vishwa	Sen	sen@school.edu
8	T7	Pavani	Rose	prose@school.org
9	T8	Anitha	Rathode	anitha@gmail.com
10	T9	Teju	Lee	lee@edu.in

```
/*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 student_id='S5' and course_id='C6';

select * from Enrollments;
```

Output:

Results Messages

	enrollment_id	student_id	course_id	enrollment_date
1	E1	S1	C1	2022-01-01
2	E10	S4	C5	2022-09-01
3	E11	S7	C6	2022-10-10
4	E2	S1	C2	2022-01-15
5	E3	S2	C3	2022-02-01
6	E4	S3	C1	2022-03-01
7	E5	S4	C4	2022-04-01
8	E7	S1	C3	2022-06-01
9	E8	S2	C10	2022-07-01
10	E9	S3	C2	2022-08-01

```
/* 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 course_id='C12' where teacher_id='T10';

select * from Courses;
```

Output:

Results Messages

	course_id	course_name	credits	teacher_id
1	C1	English	3	T2
2	C10	Skills	1	T4
3	C12	Java	6	T10
4	C2	Maths	2	T5
5	C3	Science	1	T6
6	C4	C	3	T1
7	C5	Python	5	T7
8	C6	C++	4	T8
9	C8	SQL	4	T9
10	C9	C#	2	T3

```
/*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 Enrollments where student_id=4;

delete from Students where student_id=4;
```

Output:

Results Messages

	enrollment_id	student_id	course_id	enrollment_date
1	E1	S1	C1	2022-01-01
2	E11	S7	C6	2022-10-10
3	E2	S1	C2	2022-01-15
4	E3	S2	C3	2022-02-01
5	E4	S3	C1	2022-03-01
6	E7	S1	C3	2022-06-01
7	E8	S2	C10	2022-07-01
8	E9	S3	C2	2022-08-01

Results Messages

	student_id	first_name	last_name	date_of_Birth	email	phone_number
1	S1	Emily	Chen	1999-02-15	emilyc@gmail.com	1234567890
2	S10	Liam	Davis	1998-03-10	liam.davis@outlook.com	5792468103
3	S11	John	Doe	1995-08-15	john.doe@example.com	1234567890
4	S2	Ryan	Thompson	2001-10-22	ryan.t@gmail.com	9876543210
5	S3	Priya	Singh	2000-08-12	psn@gmail.com	4567894321
6	S5	Sophia	Patel	2002-01-05	sophia.patel@hotmail.com	3692581470
7	S6	David	Kim	1997-09-20	david.kim@outlook.com	8529637410
8	S7	Olivia	Brown	2000-04-15	olivia.brown@gmail.com	1357924680
9	S8	Ethan	Hall	1999-11-25	ethan.hall@yahoo.com	2468013579
10	S9	Ava	Martin	2001-06-01	ava.martin@hotmail.com	9638527410

```
/*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=1000 where payment_id='P8';

select * from Payments;
```

Output:

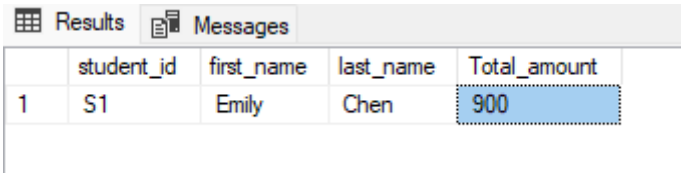
Results Messages

	payment_id	student_id	amount	payment_date
1	P1	S1	100	2022-01-05
2	P2	S1	500	2022-01-20
3	P3	S2	200	2022-02-10
4	P4	S3	800	2022-03-15
5	P6	S5	250	2022-05-25
6	P7	S1	300	2022-06-15
7	P8	S2	1000	2022-07-10
8	P9	S3	900	2022-08-15

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.student_id,s.first_name,s.
last_name,Sum(p.amount) as Total_amount
from Students s JOIN Payments p on s.student_id=p.student_id
where s.student_id='S1'
group by s.student_id,s.first_name,s.last_name;
```

Output:

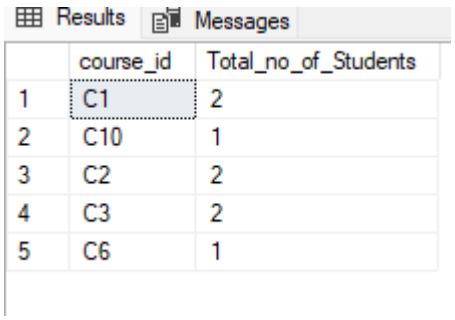


Results | Messages

	student_id	first_name	last_name	Total_amount
1	S1	Emily	Chen	900

```
/*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_id , COUNT(e.student_id) as Total_no_of_Students
from Courses c Join Enrollments e
on c.course_id = e.course_id group by c.course_id;
```

Output:

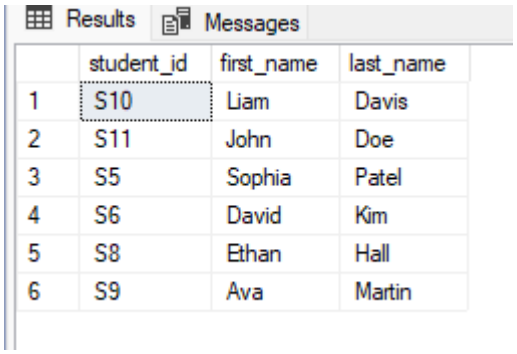


Results | Messages

	course_id	Total_no_of_Students
1	C1	2
2	C10	1
3	C2	2
4	C3	2
5	C6	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.student_id,s.first_name,
s.last_name from Students s
Left join Enrollments e on s.student_id=e.student_id
where e.student_id is Null;
```

Output:

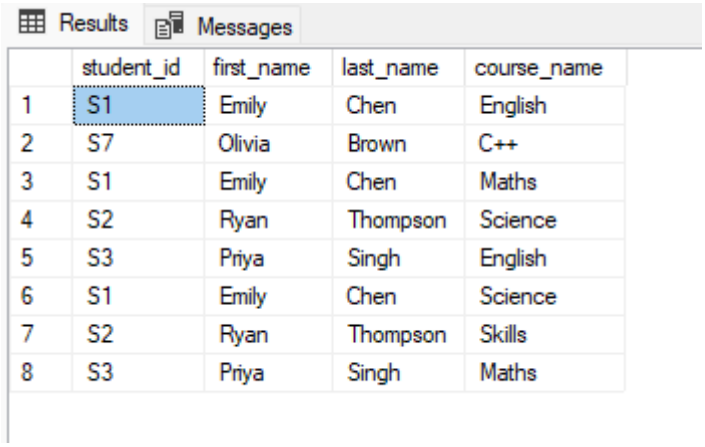


Results | Messages

	student_id	first_name	last_name
1	S10	Liam	Davis
2	S11	John	Doe
3	S5	Sophia	Patel
4	S6	David	Kim
5	S8	Ethan	Hall
6	S9	Ava	Martin

```
/*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.student_id,s.first_name,s.last_name,
c.course_name from Students s
JOIN Enrollments e on e.student_id=s.student_id
JOIN Courses c on c.course_id = e.course_id;
```

Output:



Results | Messages

	student_id	first_name	last_name	course_name
1	S1	Emily	Chen	English
2	S7	Olivia	Brown	C++
3	S1	Emily	Chen	Maths
4	S2	Ryan	Thompson	Science
5	S3	Priya	Singh	English
6	S1	Emily	Chen	Science
7	S2	Ryan	Thompson	Skills
8	S3	Priya	Singh	Maths

```
/*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(t.first_name,' ',t.last_name) as Teacher_name,c.course_name from Teacher
t join Courses c  on  c.teacher_id=t.teacher_id;
```

Output:

Results | Messages

	Teacher_name	course_name
1	Curie Joe	English
2	Rama Martin	Skills
3	Nani Saheb	Java
4	Raghu Reddy	Maths
5	Vishwa Sen	Science
6	Ram Patel	C
7	Pavani Rose	Python
8	Anitha Rathode	C++
9	Teju Lee	SQL
10	Sita Singh	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.student_id,s.first_name,s.last_name ,
e.enrollment_date from Students s JOIN Enrollments e
on e.student_id=s.student_id JOIN Courses c
on e.course_id=c.course_id where c.course_name='Maths';
```

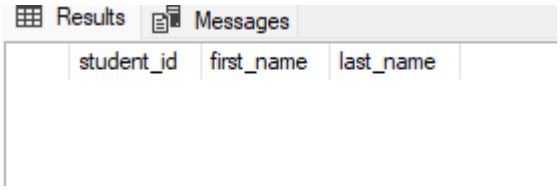
Output:

Results | Messages

	student_id	first_name	last_name	enrollment_date
1	S1	Emily	Chen	2022-01-15
2	S3	Priya	Singh	2022-08-01

```
/*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.student_id,s.first_name,s.last_name from Students s left join Payments
p on p.student_id=s.student_id where s.student_id=NULL;
```

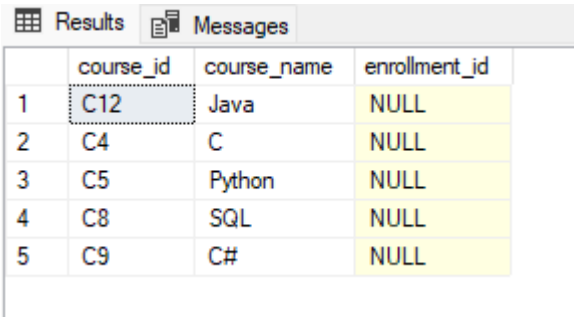
Output:



student_id	first_name	last_name

```
/*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_id,c.course_name,e.enrollment_id from Courses c
left join Enrollments e on e.course_id=c.course_id
where e.enrollment_id IS NULL;
```

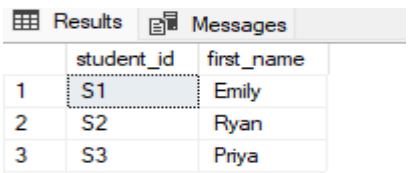
Output:



	course_id	course_name	enrollment_id
1	C12	Java	NULL
2	C4	C	NULL
3	C5	Python	NULL
4	C8	SQL	NULL
5	C9	C#	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 distinct e1.student_id,s.first_name from Enrollments e1
JOIN Enrollments e2 on e1.student_id=e2.student_id
and e1.course_id!=e2.course_id JOIN Students s
on e1.student_id=s.student_id;
```

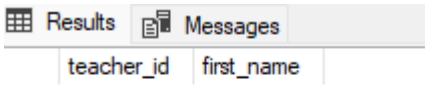
Output:



	student_id	first_name
1	S1	Emily
2	S2	Ryan
3	S3	Priya

```
/*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 t.teacher_id,t.first_name from Teacher t
Left join Courses c on t.teacher_id=c.teacher_id
where c.teacher_id IS Null;
```

Output:



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 c.course_id, c.course_name,AVG(e.student_count)
as Average_student_Enrollments from Courses c
left join (select course_id,COUNT(student_id) as student_count
from Enrollments Group by course_id) as e
on c.course_id=e.course_id group by c.course_id,c.course_name;
```

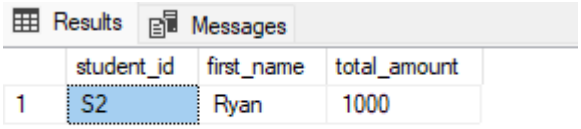
Output:

Results Messages

	course_id	course_name	Average_student_Enrollments
1	C1	English	2
2	C10	Skills	1
3	C12	Java	NULL
4	C2	Maths	2
5	C3	Science	2
6	C4	C	NULL
7	C5	Python	NULL
8	C6	C++	1
9	C8	SQL	NULL
10	C9	C#	NULL

```
/*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.student_id,s.first_name, p.amount as total_amount
from Students s
Left Join Payments p on student_id=p.student_id
where p.amount=(select Max(amount) from Payments);
```

Output:

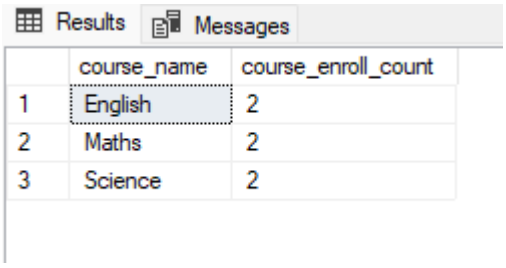


Results | Messages

	student_id	first_name	total_amount
1	S2	Ryan	1000

```
/*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, COUNT(e.enrollment_id) AS course_enroll_count
FROM Courses c
INNER JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name
HAVING COUNT(e.enrollment_id) = (
  SELECT MAX(student_count)
  FROM (
    SELECT course_id, COUNT(*) AS student_count
    FROM Enrollments
    GROUP BY course_id
  ) AS subquery
);
```

Output:

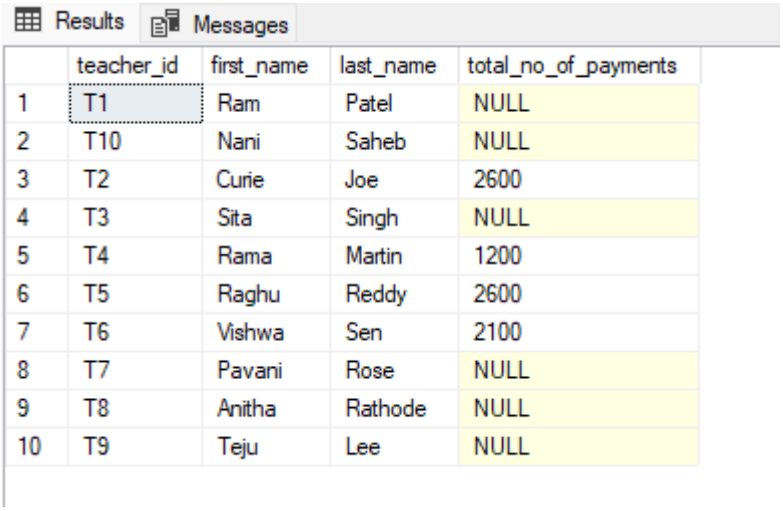


Results | Messages

	course_name	course_enroll_count
1	English	2
2	Maths	2
3	Science	2

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

select t.teacher_id,t.first_name,t.last_name,
(select SUM(p.amount) from Payments p join
Enrollments e on p.student_id=e.student_id
join Courses c on e.course_id=c.course_id
where c.teacher_id=t.teacher_id) as total_no_of_payments from Teacher t;
```

Output:

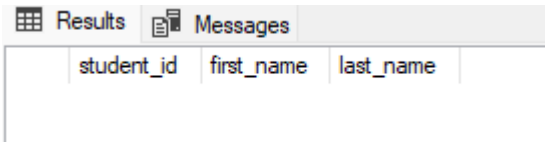


Results | Messages

	teacher_id	first_name	last_name	total_no_of_payments
1	T1	Ram	Patel	NULL
2	T10	Nani	Saheb	NULL
3	T2	Curie	Joe	2600
4	T3	Sita	Singh	NULL
5	T4	Rama	Martin	1200
6	T5	Raghu	Reddy	2600
7	T6	Vishwa	Sen	2100
8	T7	Pavani	Rose	NULL
9	T8	Anitha	Rathode	NULL
10	T9	Teju	Lee	NULL

```
/*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.student_id,s.first_name,s.last_name from Students s
where (select COUNT(distinct course_id) from courses)=(select COUNT(distinct course_id)
from Enrollments e where s.student_id=e.student_id);
```

Output:

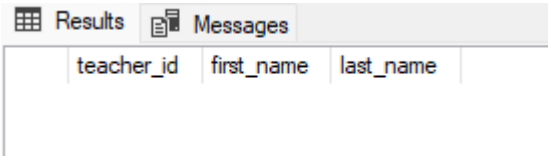


Results | Messages

	student_id	first_name	last_name

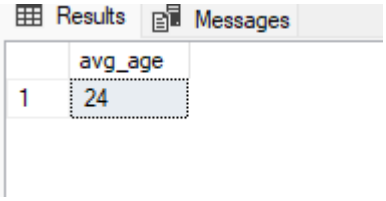
```
/*6. Retrieve the names of teachers who have not been assigned to any courses. Use
subqueries to find teachers with no course assignments.*/

select teacher_id, first_name, last_name from Teacher where teacher_id not in (select
distinct teacher_id from Courses);
```

Output:



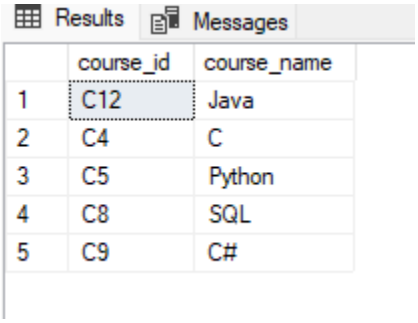

```
/*7. Calculate the average age of all students. Use subqueries to calculate the age of
each student based on their date of birth*/

select AVG(A.age) as avg_age from
(select DATEDIFF(YEAR,s.date_of_birth, GETDATE()) as age from Students s) as A;
```

Output:




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

select distinct c.course_id ,c.course_name from  Courses c where c.course_id Not in
(select distinct course_id from Enrollments);
```

Output:




```
/*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.student_id,s.first_name,s.last_name, SUM(p.amount) as total_payments
```

```
from Students s Join Payments p
on s.student_id =p.student_id
Group by s.student_id,s.first_name,s.last_name;
```

Output:

Results Messages

	student_id	first_name	last_name	total_payments
1	S1	Emily	Chen	900
2	S2	Ryan	Thompson	1200
3	S3	Priya	Singh	1700
4	S5	Sophia	Patel	250

```
/*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 s.student_id,s.first_name,s.last_name from Students s
where (select COUNT(*) from Payments p where p.student_id=s.student_id)>1;
```

Output:

Results Messages

	student_id	first_name	last_name
1	S1	Emily	Chen
2	S2	Ryan	Thompson
3	S3	Priya	Singh

```
/*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.student_id,s.first_name,s.last_name, SUM(p.amount) as total_payments
from Students s  Join Payments p on s.student_id =p.student_id
Group by s.student_id,s.first_name,s.last_name;
```

Output:

Results Messages

	student_id	first_name	last_name	total_payments
1	S1	Emily	Chen	900
2	S2	Ryan	Thompson	1200
3	S3	Priya	Singh	1700
4	S5	Sophia	Patel	250

```
/*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.student_id) as enrollment_count from Courses c
left Join Enrollments e on c.course_id=e.course_id
Group by c.course_name;
```

Output:

Results Messages

	course_name	enrollment_count
1	C	0
2	C#	0
3	C++	1
4	English	2
5	Java	0
6	Maths	2
7	Python	0
8	Science	2
9	Skills	1
10	SQL	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*/

select AVG(p.amount) as Average_payments from Students s
JOIN Payments p on s.student_id=p.student_id;
```

Output:

Results Messages

	Average_payments
1	506