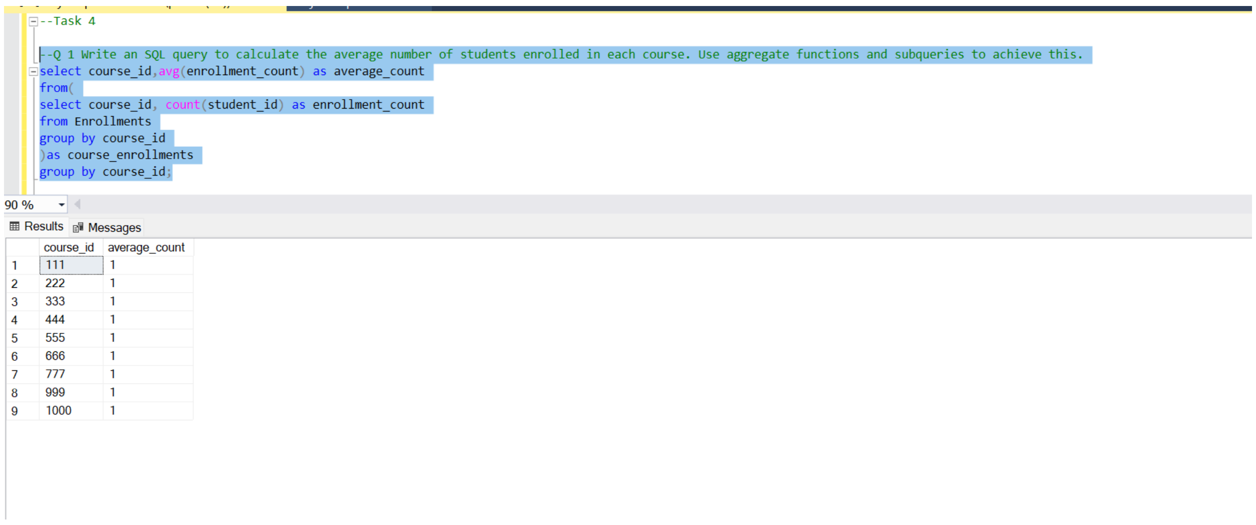
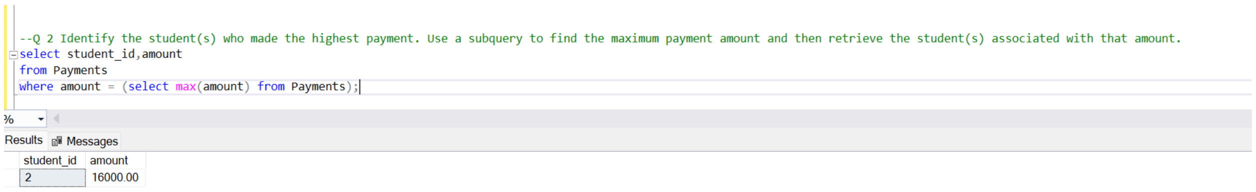


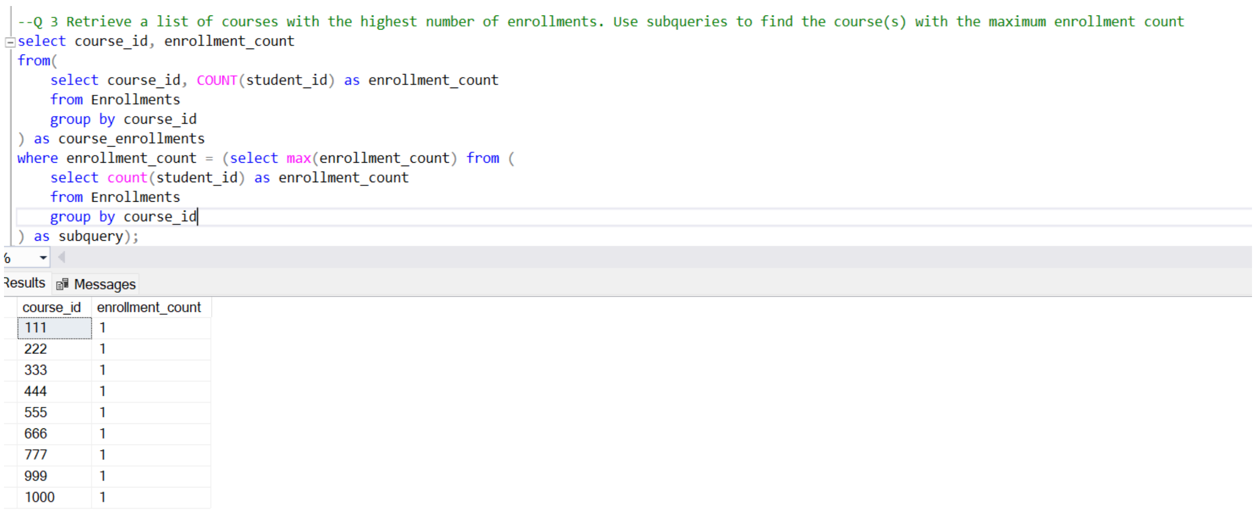
```sql
--Task 4

--Q 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 course_id,avg(enrollment_count) as average_count
from(
select course_id, count(student_id) as enrollment_count
from Enrollments
group by course_id
)as course_enrollments
group by course_id;
```

	course_id	average_count
1	111	1
2	222	1
3	333	1
4	444	1
5	555	1
6	666	1
7	777	1
8	999	1
9	1000	1



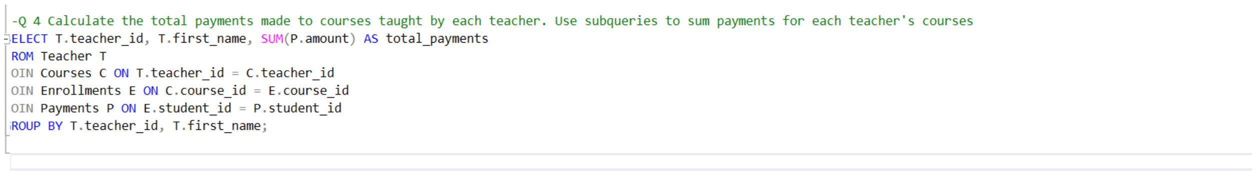
```sql
--Q 2 Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.
select student_id,amount
from Payments
where amount = (select max(amount) from Payments);
```

student_id	amount
2	16000.00

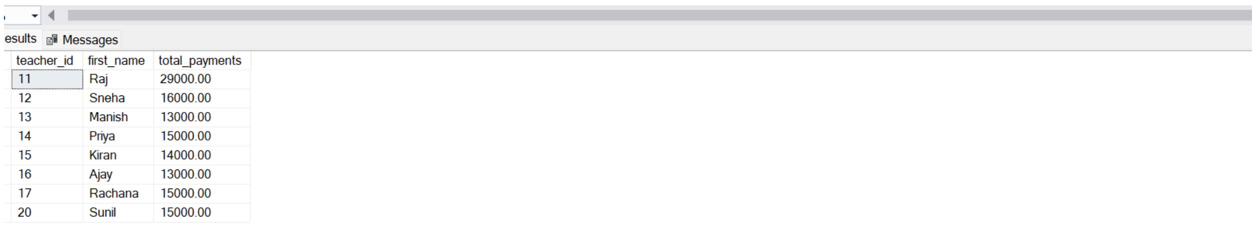


```sql
--Q 3 Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count
select course_id, enrollment_count
from(
    select course_id, COUNT(student_id) as enrollment_count
    from Enrollments
    group by course_id
) as course_enrollments
where enrollment_count = (select max(enrollment_count) from (
    select count(student_id) as enrollment_count
    from Enrollments
    group by course_id
) as subquery);
```

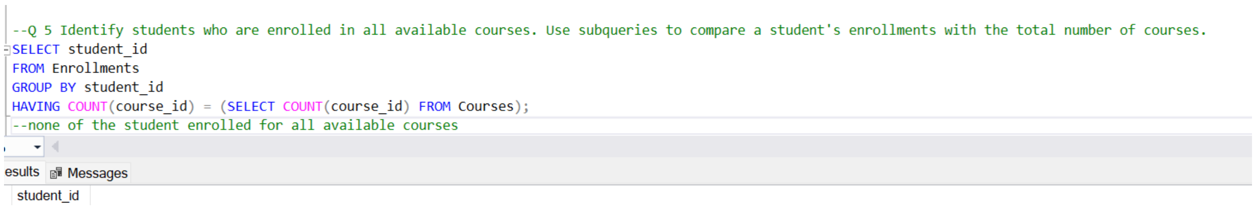
course_id	enrollment_count
111	1
222	1
333	1
444	1
555	1
666	1
777	1
999	1
1000	1



```
-Q 4 Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses
ELECT T.teacher_id, T.first_name, SUM(P.amount) AS total_payments
ROM Teacher T
OIN Courses C ON T.teacher_id = C.teacher_id
OIN Enrollments E ON C.course_id = E.course_id
OIN Payments P ON E.student_id = P.student_id
ROUP BY T.teacher_id, T.first_name;
```

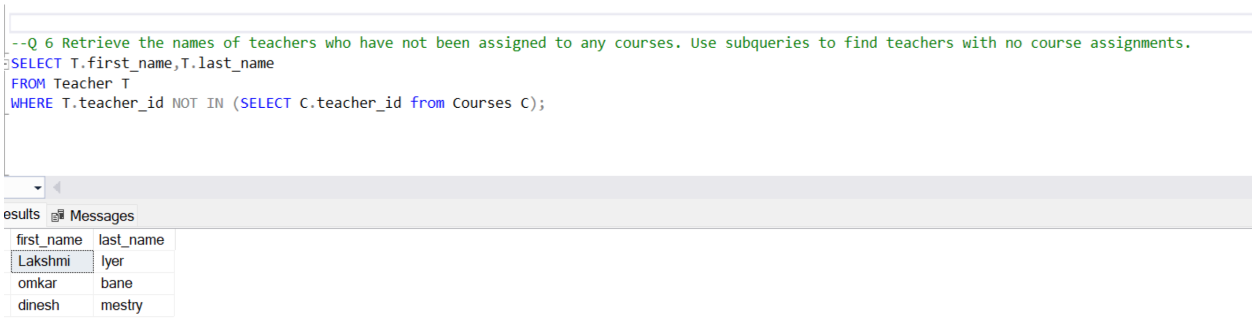


teacher_id	first_name	total_payments
11	Raj	29000.00
12	Sneha	16000.00
13	Manish	13000.00
14	Priya	15000.00
15	Kiran	14000.00
16	Ajay	13000.00
17	Rachana	15000.00
20	Sunil	15000.00



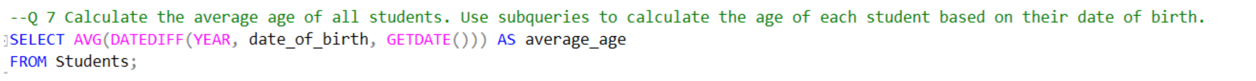
```
--Q 5 Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.
SELECT student_id
FROM Enrollments
GROUP BY student_id
HAVING COUNT(course_id) = (SELECT COUNT(course_id) FROM Courses);
--none of the student enrolled for all available courses
```

student_id

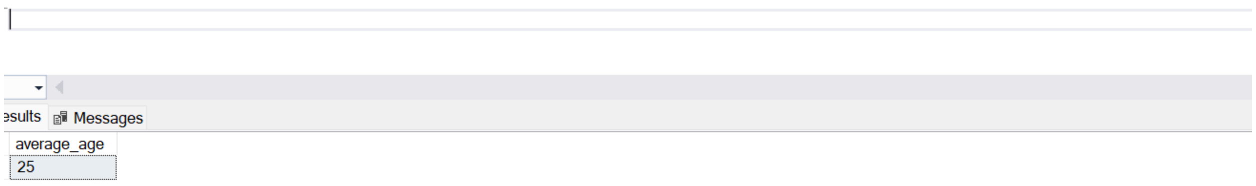


```
--Q 6 Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.
SELECT T.first_name,T.last_name
FROM Teacher T
WHERE T.teacher_id NOT IN (SELECT C.teacher_id from Courses C);
```

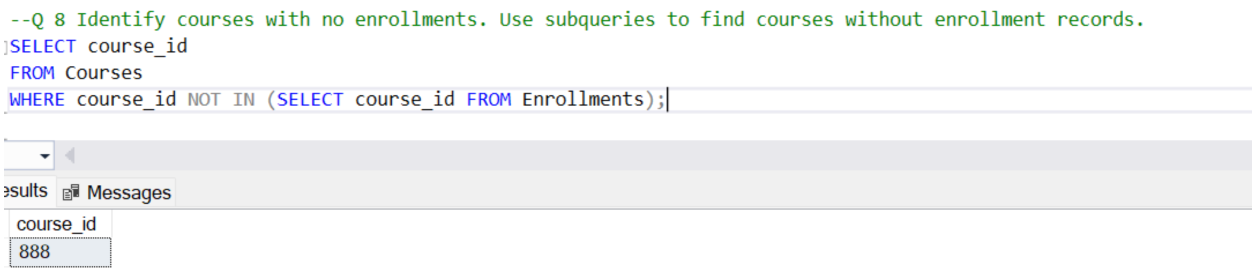
first_name	last_name
Lakshmi	Iyer
omkar	bane
dinesh	mestry



```
--Q 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(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age
FROM Students;
```



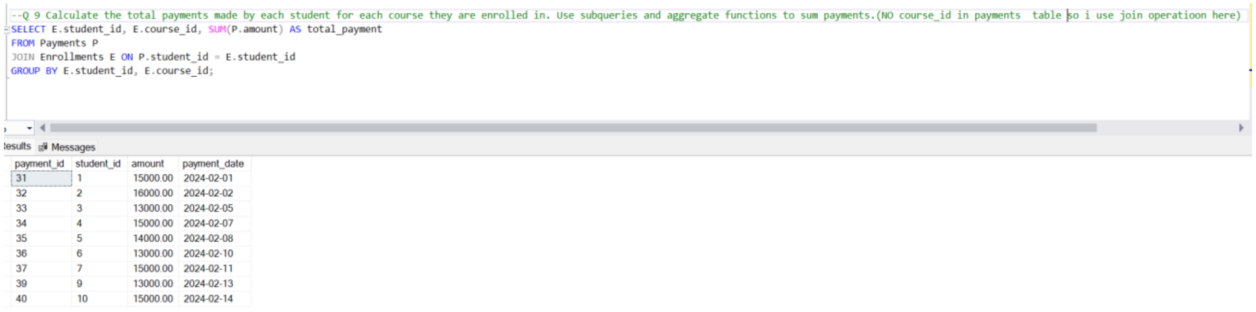
average_age
25



```sql
--Q 8 Identify courses with no enrollments. Use subqueries to find courses without enrollment records.
SELECT course_id
FROM Courses
WHERE course_id NOT IN (SELECT course_id FROM Enrollments);
```

esults   Messages

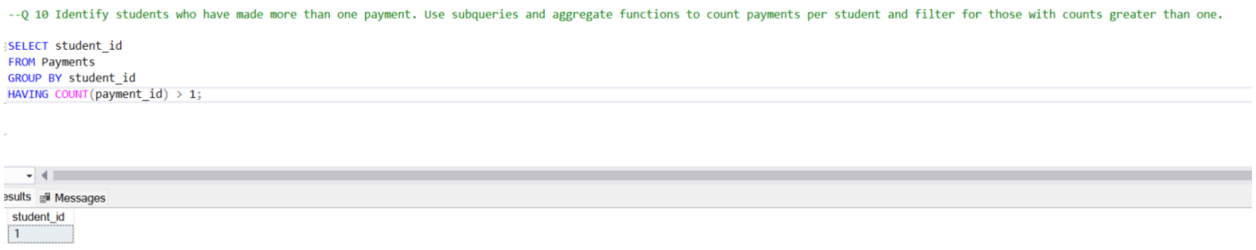
```
course_id
888
```



```sql
--Q 9 Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.(NO course_id in payments  table so i use join operatioon here)
SELECT E.student_id, E.course_id, SUM(P.amount) AS total_payment
FROM Payments P
JOIN Enrollments E ON P.student_id = E.student_id
GROUP BY E.student_id, E.course_id;
```

esults   Messages

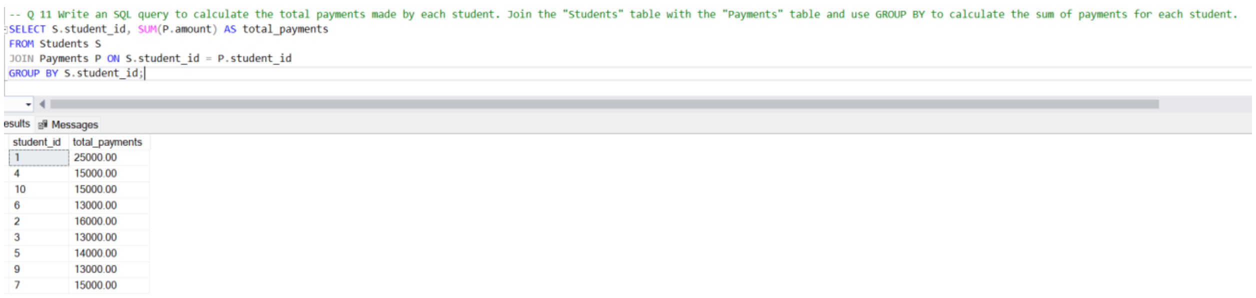
```
payment_id  student_id  amount    payment_date
31          1           15000.00  2024-02-01
32          2           16000.00  2024-02-02
33          3           13000.00  2024-02-05
34          4           15000.00  2024-02-07
35          5           14000.00  2024-02-08
36          6           13000.00  2024-02-10
37          7           15000.00  2024-02-11
39          9           13000.00  2024-02-13
40          10          15000.00  2024-02-14
```



```sql
--Q 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 student_id
FROM Payments
GROUP BY student_id
HAVING COUNT(payment_id) > 1;
```

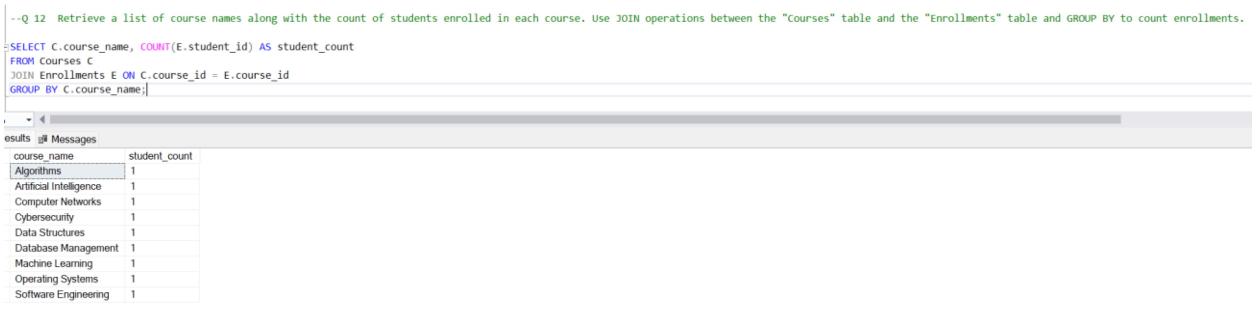
esults   Messages

```
student_id
1
```



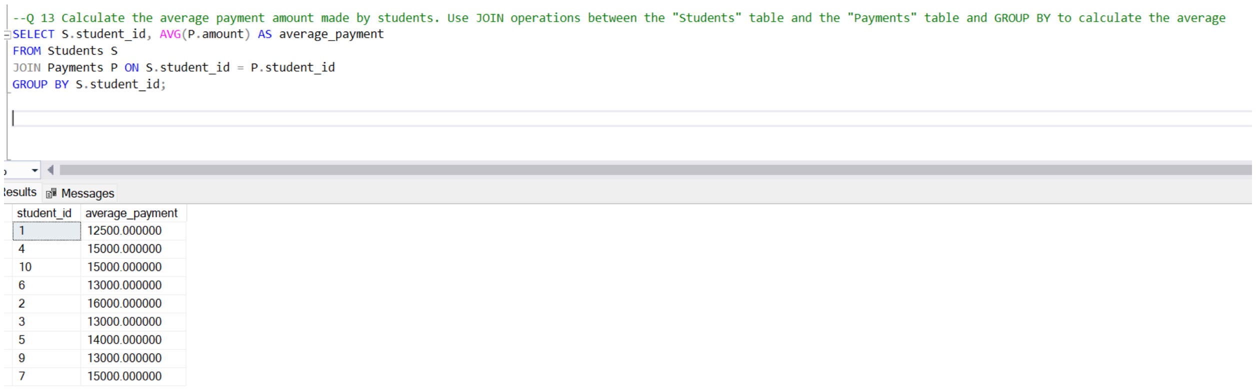
```sql
-- Q 11 Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.
SELECT S.student_id, SUM(P.amount) AS total_payments
FROM Students S
JOIN Payments P ON S.student_id = P.student_id
GROUP BY S.student_id;
```

student_id	total_payments
1	25000.00
4	15000.00
10	15000.00
6	13000.00
2	16000.00
3	13000.00
5	14000.00
9	13000.00
7	15000.00



```sql
--Q 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 student_count
FROM Courses C
JOIN Enrollments E ON C.course_id = E.course_id
GROUP BY C.course_name;
```

course_name	student_count
Algorithms	1
Artificial Intelligence	1
Computer Networks	1
Cybersecurity	1
Data Structures	1
Database Management	1
Machine Learning	1
Operating Systems	1
Software Engineering	1



```sql
--Q 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.student_id, AVG(P.amount) AS average_payment
FROM Students S
JOIN Payments P ON S.student_id = P.student_id
GROUP BY S.student_id;
```

student_id	average_payment
1	12500.000000
4	15000.000000
10	15000.000000
6	13000.000000
2	16000.000000
3	13000.000000
5	14000.000000
9	13000.000000
7	15000.000000