SQLQuery3.sql - Z...ARSH SHARMA (81))*

```sql
/*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(e.student_id) AS student_count
    FROM Enrollments e
    GROUP BY e.course_id
) AS course_enrollments;
```

148 %

Results | Messages

	avg_students_per_course
1	1

SQLQuery4.sql - Z...ARSH SHARMA (51))* | SQLQuery3.sql - Z...ARSH SHARMA (81))*

```sql
/*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, p.amount
FROM Payments p
JOIN Students s ON p.student_id = s.student_id
WHERE p.amount = (SELECT MAX(amount) FROM Payments);
```

148 %

Results | Messages

	first_name	last_name	amount
1	sittu	gittu	700.00

SQLQuery7.sql - Z...ARSH SHARMA (52))*

```sql
/*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_id, C.course_name, COUNT(E.student_id) AS enrollment_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.student_id) = (
    SELECT MAX(enrollment_count)
    FROM (SELECT COUNT(student_id) AS enrollment_count FROM Enrollments GROUP BY course_id) AS MaxEnrollments);
```

111 %

Results | Messages

	course_id	course_name	enrollment_count
1	1	Mathematics	2

```sql
-- 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
        INNER JOIN Enrollments E ON P.student_id = E.student_id
        INNER JOIN Courses C ON E.course_id = C.course_id
        WHERE C.teacher_id = T.teacher_id
       ) AS total_payments FROM Teacher T;
```

111 %

Results | Messages

	teacher_id	first_name	last_name	total_payments
1	1	vibha	bhatore	1300.00
2	2	sandhya	ggete	1350.00
3	3	sudhir	sharma	800.00
4	4	smriti	irani	400.00
5	5	gaduda	hariye	NULL
6	6	vandana	sohani	NULL
7	7	mukesh	prajapati	NULL
8	8	shailendra	khede	NULL
9	9	sarthak	sharma	NULL
10	10	gangadhar	sharma	NULL

SQLQuery9.sql - Z...ARSH SHARMA (56))* | SQLQuery8.sql - Z...ARSH SHARMA (65))* | SQLQuery7.sql - Z...ARSH

```
/* 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, c.course_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id;
```

101 %

Results | Messages

	first_name	last_name	course_name
1	vandana	sohani	Art
2	mohini	rathore	Computer Science
3	jessica	sharma	History
4	vaishali	sharma	Chemistry
5	khushbu	sharma	Biology
6	sunita	sharma	Mathematics
7	adarsh	sharma	English Literature
8	adarsh	sharma	Mathematics
9	khushbu	sharma	Physics

```sql
-- 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);
```

101 %

Results | Messages

	first_name	last_name
1	gaduda	hariye
2	vandana	sohani
3	mukesh	prajapati
4	shailendra	khede
5	sarthak	sharma
6	gangadhar	sharma

SQLQuery11.sql -...ARSH SHARMA (51))* | SQLQuery10.sql -...ARSH SHARMA (61))* | SQLQuery9.sql - Z...ARSH SHARMA (56))* | SQLQuery8.sql - Z...ARSH SHARMA (65))*

```sql
-- 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(age) AS average_age
FROM ( SELECT FLOOR(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS age FROM Students) AS AgeCalculation;
```

101 %

Results | Messages

	average_age
1	21

SQLQuery12.sql -...ARSH SHARMA (72))* | SQLQuery11.sql -...ARSH SHARMA (51))* | SQLQuery10.sql -...ARSH SHARMA (61))*

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

101 %

Results | Messages

	course_name
1	Music
2	Economics

SQLQuery13.sql -...ARSH SHARMA (62))* | SQLQuery12.sql -...ARSH SHARMA (72))* | SQLQuery11.sql -...ARSH SHARMA (51))* | SQLQuery10.sql -...ARSH SHARMA (61))*

```sql
-- 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, C.course_name,
       (SELECT SUM(P.amount)
        FROM Payments P
        INNER JOIN Enrollments E2 ON P.student_id = E2.student_id
        WHERE E2.course_id = C.course_id AND E2.student_id = S.student_id) AS total_payments
FROM Students S
INNER JOIN Enrollments E ON S.student_id = E.student_id
INNER JOIN Courses C ON E.course_id = C.course_id
ORDER BY S.student_id, C.course_name;
```

101 %

Results | Messages

	student_id	first_name	last_name	course_name	total_payments
1	1	adarsh	sharma	English Literature	500.00
2	1	adarsh	sharma	Mathematics	500.00
3	2	sunita	sharma	Mathematics	300.00
4	3	khushbu	sharma	Biology	400.00
5	3	khushbu	sharma	Physics	400.00
6	4	vaishali	sharma	Chemistry	200.00
7	6	jessica	sharma	History	450.00
8	7	mohini	rathore	Computer Science	600.00
9	8	vandana	sohani	Art	500.00

SQLQuery14.sql -...ARSH SHARMA (54))* | SQLQuery13.sql -...ARSH SHARMA (62))* | SQL

```sql
/*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 t.first_name, t.last_name
FROM Teacher t
LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
WHERE c.course_id IS NULL;
```

101 %

Results | Messages

	first_name	last_name
1	gaduda	hariye
2	vandana	sohani
3	mukesh	prajapati
4	shailendra	khede
5	sarthak	sharma
6	gangadhar	sharma

SQLQuery15.sql -...ARSH SHARMA (59))* | SQLQuery13.sql -...ARSH SHARMA (62))* | SQLQuery14.sql -...ARSH SHARMA (54))* | SQLQuery12.sql -...ARSH

```sql
/*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,
(SELECT SUM(P.amount) FROM Payments P WHERE P.student_id = S.student_id) AS total_payments
FROM Students S;
```

101 %

Results | Messages

	student_id	first_name	last_name	total_payments
1	1	adarsh	sharma	500.00
2	2	sunita	sharma	300.00
3	3	khushbu	sharma	400.00
4	4	vaishali	sharma	200.00
5	5	arpit	gavshinde	350.00
6	6	jessica	sharma	450.00
7	7	mohini	rathore	600.00
8	8	vandana	sohani	500.00
9	9	sittu	gittu	700.00
10	10	sonali	rajguru	550.00
11	11	John	Doe	NULL

SQLQuery16.sql -...ARSH SHARMA (77))* | SQLQuery15.sql -...ARSH SHARMA (59))* | SQLQuery13.sql -...ARSH SHARMA (62))* | SQLQuery14.sql -...ARSH SHARMA (54))*

```sql
/*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, (SELECT COUNT(E.student_id) FROM Enrollments E WHERE E.course_id = C.course_id) AS student_count
FROM Courses C;
```

122 %

Results | Messages

	course_name	student_count
1	Mathematics	2
2	English Literature	1
3	Biology	1
4	Chemistry	1
5	Physics	1
6	History	1
7	Computer Science	1
8	Art	1
9	Music	0
10	Economics	0

SQLQuery17.sql -...ARSH SHARMA (78))* | SQLQuery16.sql -...ARSH SHARMA (77))* | SQLQuery15.sql -...ARSH SHARMA (59))* | SQLQuery13.sql -...ARSH SHARMA (62))*

```sql
/*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, S.first_name, S.last_name, AVG(P.amount) AS average_payment
FROM Students S
JOIN Payments P ON S.student_id = P.student_id
GROUP BY S.student_id, S.first_name, S.last_name;
```

122 %

Results | Messages

	student_id	first_name	last_name	average_payment
1	1	adarsh	sharma	500.000000
2	2	sunita	sharma	300.000000
3	3	khushbu	sharma	400.000000
4	4	vaishali	sharma	200.000000
5	5	arpit	gavshinde	350.000000
6	6	jessica	sharma	450.000000
7	7	mohini	rathore	600.000000
8	8	vandana	sohani	500.000000
9	9	sittu	gittu	700.000000
10	10	sonali	rajguru	550.000000