ASSIGNMENT – 2: Student Information System (SIS)

Task 1. Database Design:

1. Create the database named "SISDB"

```
Assignment-2.sql...ASANTH\vamsh (66))*
1  --TASK 1
2
3  -- 1. Create the database named "SISDB"
4
5  create database SIS;
6  use SIS;
7
```

133 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-10T10:46:20.5825002+05:30
```

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

```
Assignment-2.sql...ASANTH\vamsh (66))*   Execute (F5)
8  /*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.
9  a. Students
10 b. Courses
11 c. Enrollments
12 d. Teacher
13 e. Payments*/
14
15 CREATE TABLE Students (
16     student_id INT PRIMARY KEY,
17     first_name VARCHAR(255),
18     last_name VARCHAR(255),
19     date_of_birth DATE,
20     email VARCHAR(255),
21     phone_number VARCHAR(15));
```

133 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-10T10:46:55.4067565+05:30
```

b. Courses

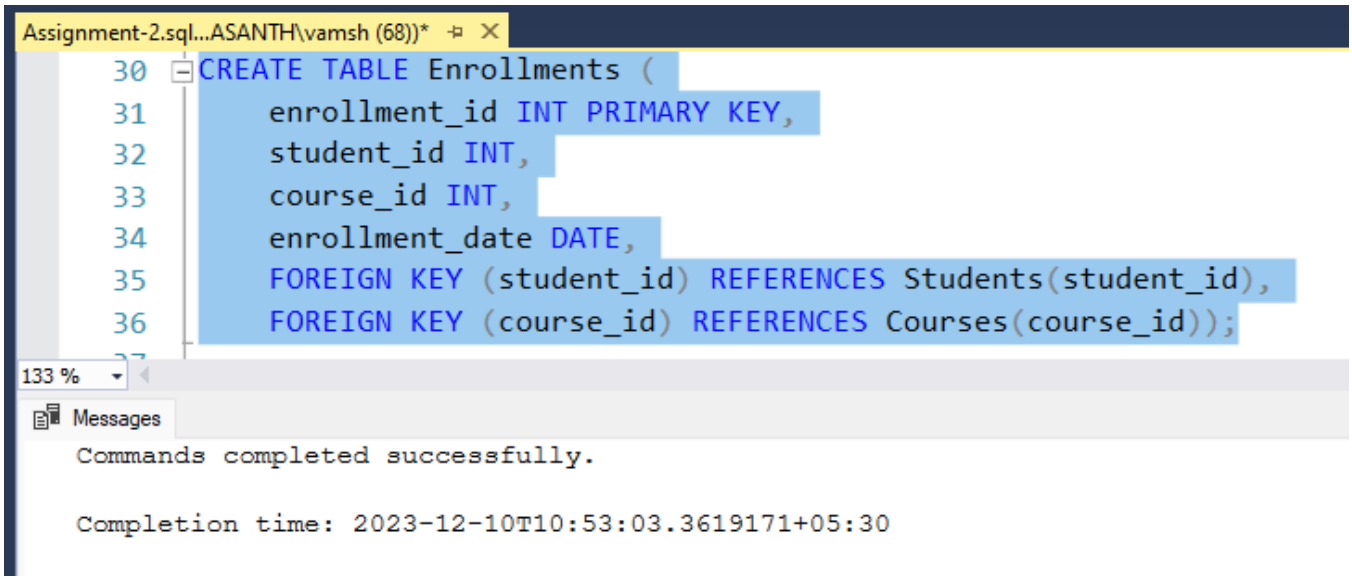
```
Assignment-2.sql...ASANTH\vamsh (68))
23 CREATE TABLE Courses (
24     course_id INT PRIMARY KEY,
25     course_name VARCHAR(255),
26     credits INT,
27     teacher_id INT,
28     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id));
```

133 %

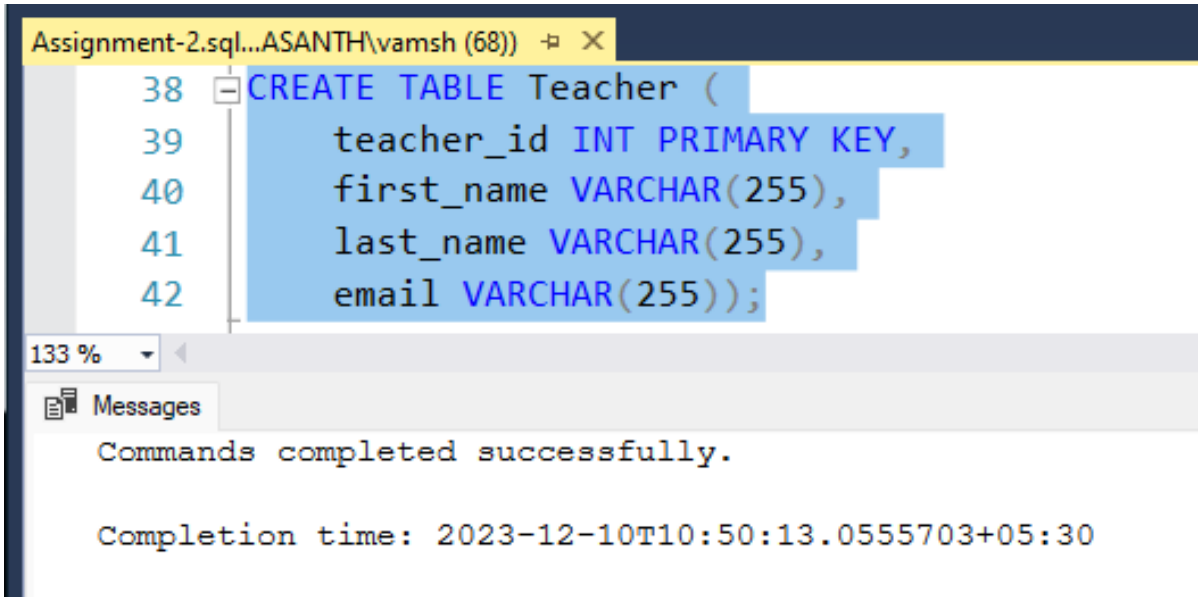
Messages

```
Commands completed successfully.

Completion time: 2023-12-10T10:50:54.3098168+05:30
```

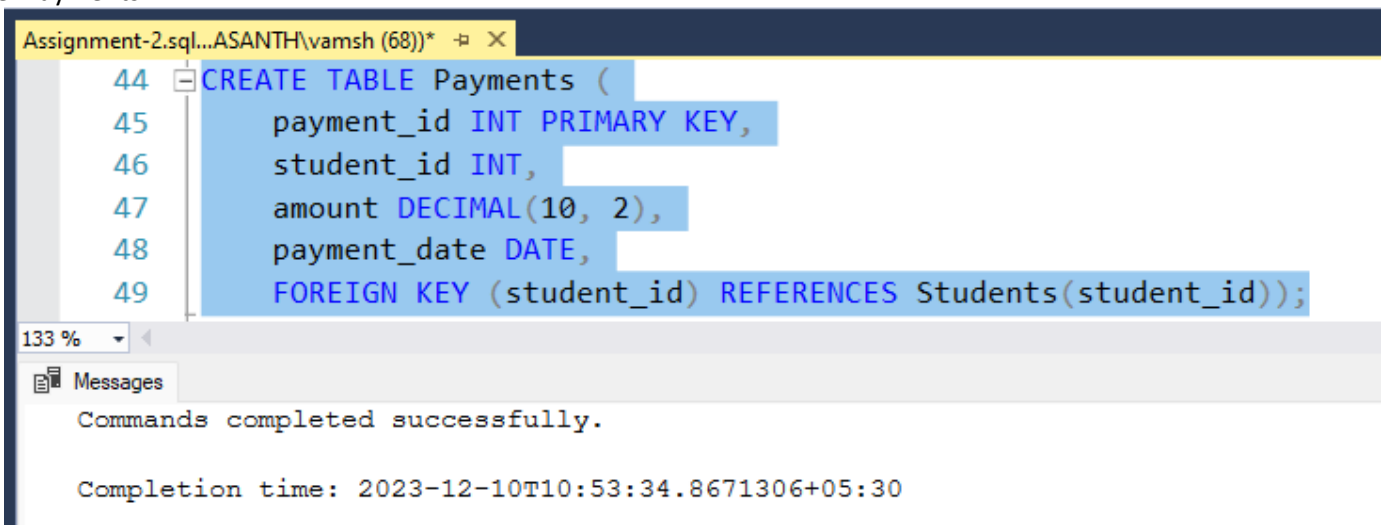
c. Enrollments



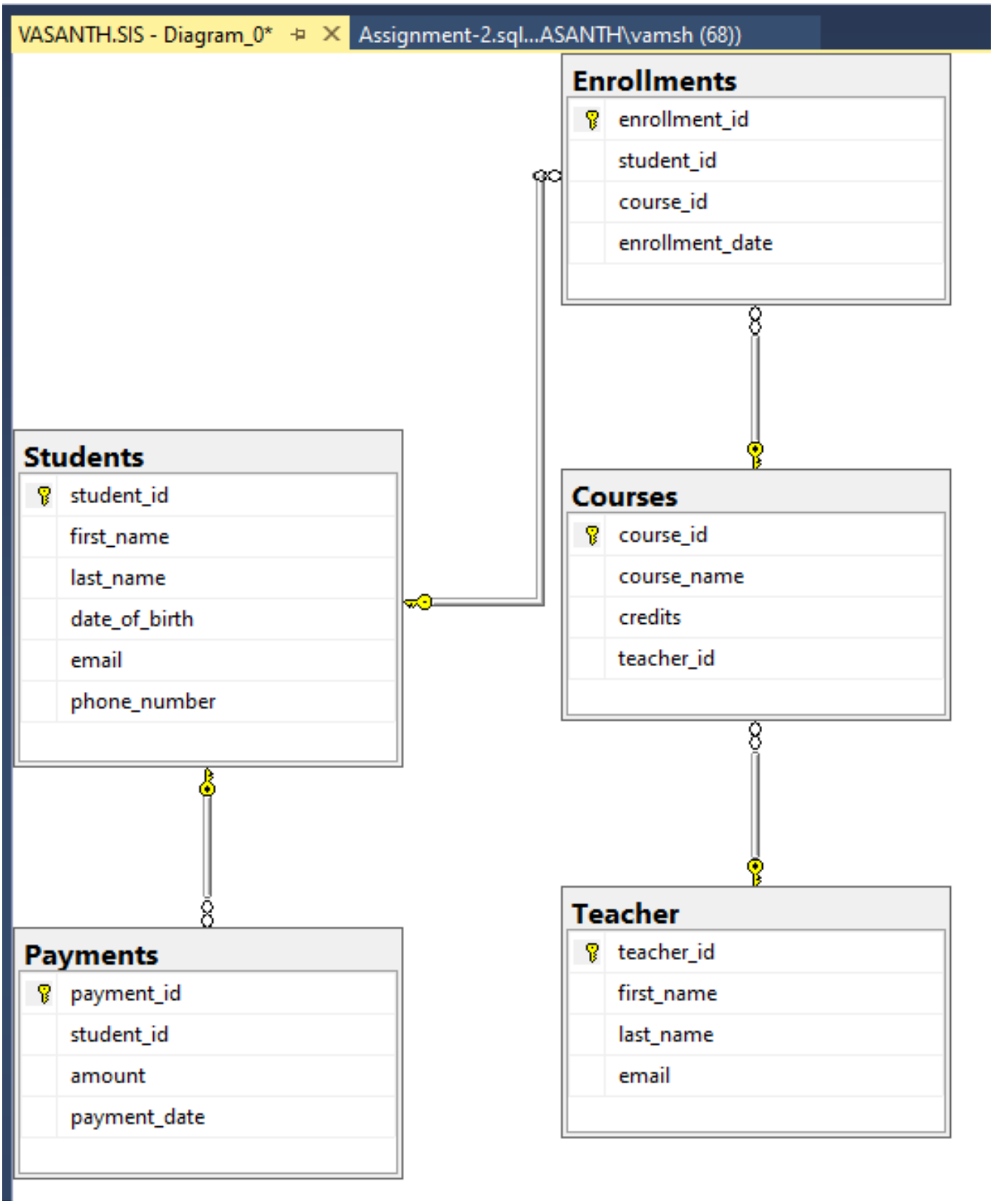
d. Teacher



e. Payments



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



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

Primary keys and foreign keys are already defined in the table creation statements.

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

i. Students

```sql
/*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 (student_id, first_name, last_name, date_of_birth, email,
  phone_number)
VALUES
(1, 'Rohan', 'jami', '1995-05-10', 'rohan.jami@email.com', '147-852-9630'),
(2, 'Jane', 'Smith', '1998-08-22', 'jane.smith@email.com', '987-654-3210'),
(3, 'Bob', 'Johnson', '1997-02-15', 'bob.johnson@email.com', '555-123-4567'),
(4, 'Alice', 'Williams', '1996-11-30', 'alice.williams@email.com', '222-333-4444'),
(5, 'Charlie', 'Brown', '1999-04-05', 'charlie.brown@email.com', '111-999-8888'),
(6, 'Emma', 'Davis', '1994-07-18', 'emma.davis@email.com', '777-888-9999'),
(7, 'Michael', 'Lee', '1993-09-25', 'michael.lee@email.com', '666-555-4444'),
(8, 'Sophia', 'Taylor', '2000-01-12', 'sophia.taylor@email.com', '444-222-1111'),
(9, 'David', 'Miller', '1992-12-03', 'david.miller@email.com', '123-987-6543'),
(10, 'Olivia', 'Moore', '1991-06-28', 'olivia.moore@email.com', '999-111-3333');
select * from Students;
```

Results Messages

	student_id	first_name	last_name	date_of_birth	email	phone_number
1	1	Rohan	jami	1995-05-10	rohan.jami@email.com	147-852-9630
2	2	Jane	Smith	1998-08-22	jane.smith@email.com	987-654-3210
3	3	Bob	Johnson	1997-02-15	bob.johnson@email.com	555-123-4567
4	4	Alice	Williams	1996-11-30	alice.williams@email.com	222-333-4444
5	5	Charlie	Brown	1999-04-05	charlie.brown@email.com	111-999-8888
6	6	Emma	Davis	1994-07-18	emma.davis@email.com	777-888-9999
7	7	Michael	Lee	1993-09-25	michael.lee@email.com	666-555-4444
8	8	Sophia	Taylor	2000-01-12	sophia.taylor@email.com	444-222-1111
9	9	David	Miller	1992-12-03	david.miller@email.com	123-987-6543
10	10	Olivia	Moore	1991-06-28	olivia.moore@email.com	999-111-3333

ii. Courses

```sql
INSERT INTO Courses (course_id, course_name, credits, teacher_id)
VALUES
(1, 'Computer Science Fundamentals', 4, 1),
(2, 'Indian History and Culture', 3, 3),
(3, 'Mathematics for Engineering', 5, 5),
(4, 'Environmental Science', 3, 2),
(5, 'English Literature', 4, 6),
(6, 'Business Management Principles', 3, 4),
(7, 'Artificial Intelligence and Machine Learning', 4, 1),
(8, 'Economics of India', 3, 7),
(9, 'Civil Engineering Design', 5, 9),
(10, 'Indian Classical Music', 2, 8);
select * from Courses;
```

Results Messages

	course_id	course_name	credits	teacher_id
1	1	Computer Science Fundamentals	4	1
2	2	Indian History and Culture	3	3
3	3	Mathematics for Engineering	5	5
4	4	Environmental Science	3	2
5	5	English Literature	4	6
6	6	Business Management Principles	3	4
7	7	Artificial Intelligence and Machine Learning	4	1
8	8	Economics of India	3	7
9	9	Civil Engineering Design	5	9
10	10	Indian Classical Music	2	8

iii. Enrollments

```sql
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES
(1, 1, 7, '2023-01-15'),
(2, 2, 3, '2023-02-20'),
(3, 3, 8, '2023-03-10'),
(4, 4, 1, '2023-04-05'),
(5, 5, 6, '2023-05-12'),
(6, 6, 2, '2023-06-18'),
(7, 7, 9, '2023-07-25'),
(8, 8, 4, '2023-08-30'),
(9, 9, 10, '2023-09-08'),
(10, 10, 5, '2023-10-15');
select * from Enrollments;
```

Results Messages

	enrollment_id	student_id	course_id	enrollment_date
1	1	1	7	2023-01-15
2	2	2	3	2023-02-20
3	3	3	8	2023-03-10
4	4	4	1	2023-04-05
5	5	5	6	2023-05-12
6	6	6	2	2023-06-18
7	7	7	9	2023-07-25
8	8	8	4	2023-08-30
9	9	9	10	2023-09-08
10	10	10	5	2023-10-15

iv. Teacher

```sql
INSERT INTO Teacher (teacher_id, first_name, last_name, email)
VALUES
(1, 'Amit', 'Kumar', 'amit.kumar@email.com'),
(2, 'Priya', 'Sharma', 'priya.sharma@email.com'),
(3, 'Rahul', 'Verma', 'rahul.verma@email.com'),
(4, 'Ananya', 'Srivastava', 'ananya.srivastava@email.com'),
(5, 'Vikram', 'Singh', 'vikram.singh@email.com'),
(6, 'Kavita', 'Joshi', 'kavita.joshi@email.com'),
(7, 'Raj', 'Patel', 'raj.patel@email.com'),
(8, 'Neha', 'Mishra', 'neha.mishra@email.com'),
(9, 'Ravi', 'Yadav', 'ravi.yadav@email.com'),
(10, 'Sonia', 'Gupta', 'sonia.gupta@email.com');
select * from Teacher;
```

Results Messages

	teacher_id	first_name	last_name	email
1	1	Amit	Kumar	amit.kumar@email.com
2	2	Priya	Sharma	priya.sharma@email.com
3	3	Rahul	Verma	rahul.verma@email.com
4	4	Ananya	Srivastava	ananya.srivastava@email.com
5	5	Vikram	Singh	vikram.singh@email.com
6	6	Kavita	Joshi	kavita.joshi@email.com
7	7	Raj	Patel	raj.patel@email.com
8	8	Neha	Mishra	neha.mishra@email.com
9	9	Ravi	Yadav	ravi.yadav@email.com
10	10	Sonia	Gupta	sonia.gupta@email.com

v. Payments

```sql
INSERT INTO Payments (payment_id, student_id, amount, payment_date)
VALUES
(1, 1, 500.00, '2023-01-05'),
(2, 2, 750.50, '2023-02-10'),
(3, 3, 600.25, '2023-03-15'),
(4, 4, 900.75, '2023-04-20'),
(5, 5, 350.00, '2023-05-25'),
(6, 6, 800.50, '2023-06-30'),
(7, 7, 450.25, '2023-07-05'),
(8, 8, 700.75, '2023-08-10'),
(9, 9, 550.00, '2023-09-15'),
(10, 10, 950.50, '2023-10-20');
select * from Payments;
```

Results

	payment_id	student_id	amount	payment_date
1	1	1	500.00	2023-01-05
2	2	2	750.50	2023-02-10
3	3	3	600.25	2023-03-15
4	4	4	900.75	2023-04-20
5	5	5	350.00	2023-05-25
6	6	6	800.50	2023-06-30
7	7	7	450.25	2023-07-05
8	8	8	700.75	2023-08-10
9	9	9	550.00	2023-09-15
10	10	10	950.50	2023-10-20

Tasks 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

```sql
-- Tasks 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
© Hexaware Technologies Limited. All rights www.hexaware.com
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 (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
select * from Students;
```

Results

	student_id	first_name	last_name	date_of_birth	email	phone_number
1	1	Rohan	jami	1995-05-10	rohan.jami@email.com	147-852-9630
2	2	Jane	Smith	1998-08-22	jane.smith@email.com	987-654-3210
3	3	Bob	Johnson	1997-02-15	bob.johnson@email.com	555-123-4567
4	4	Alice	Williams	1996-11-30	alice.williams@email.com	222-333-4444
5	5	Charlie	Brown	1999-04-05	charlie.brown@email.com	111-999-8888
6	6	Emma	Davis	1994-07-18	emma.davis@email.com	777-888-9999
7	7	Michael	Lee	1993-09-25	michael.lee@email.com	666-555-4444
8	8	Sophia	Taylor	2000-01-12	sophia.taylor@email.com	444-222-1111
9	9	David	Miller	1992-12-03	david.miller@email.com	123-987-6543
10	10	Olivia	Moore	1991-06-28	olivia.moore@email.com	999-111-3333
11	11	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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
142  --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.
143
144  INSERT INTO Enrollments (enrollment_id,student_id, course_id, enrollment_date)
145  VALUES (11,1, 2,'2023-12-08');
146  select * from Enrollments;
```

133 %

Results | Messages

	enrollment_id	student_id	course_id	enrollment_date
1	1	1	7	2023-01-15
2	2	2	3	2023-02-20
3	3	3	8	2023-03-10
4	4	4	1	2023-04-05
5	5	5	6	2023-05-12
6	6	6	2	2023-06-18
7	7	7	9	2023-07-25
8	8	8	4	2023-08-30
9	9	9	10	2023-09-08
10	10	10	5	2023-10-15
11	11	1	2	2023-12-08

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

```
Assignment-2.sql...ASANTH\vamsh (68))*
148  --3. Update the email address of a specific teacher in the "Teacher" table. Choose
       any teacher and modify their email address.
149
150  UPDATE Teacher
151  SET email = 'axar.patel@email.com'
152  WHERE teacher_id = 7;
153  select * from Teacher;
```

133 %

Results | Messages

	teacher_id	first_name	last_name	email
1	1	Amit	Kumar	amit.kumar@email.com
2	2	Priya	Sharma	priya.sharma@email.com
3	3	Rahul	Verma	rahul.verma@email.com
4	4	Ananya	Srivastava	ananya.srivastava@email.com
5	5	Vikram	Singh	vikram.singh@email.com
6	6	Kavita	Joshi	kavita.joshi@email.com
7	7	Raj	Patel	axar.patel@email.com
8	8	Neha	Mishra	neha.mishra@email.com
9	9	Ravi	Yadav	ravi.yadav@email.com
10	10	Sonia	Gupta	sonia.gupta@email.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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
155  --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.
156
157  DELETE FROM Enrollments
158  WHERE student_id = 5 AND course_id = 6;
159  select * from Enrollments;
```

133 %

Results | Messages

	enrollment_id	student_id	course_id	enrollment_date
1	1	1	7	2023-01-15
2	2	2	3	2023-02-20
3	3	3	8	2023-03-10
4	4	4	1	2023-04-05
5	6	6	2	2023-06-18
6	7	7	9	2023-07-25
7	8	8	4	2023-08-30
8	9	9	10	2023-09-08
9	10	10	5	2023-10-15
10	11	1	2	2023-12-08

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

Assignment-2.sql...ASANTH\vamsh (68))*

```
161  --5. Update the "Courses" table to assign a specific teacher to a course. Choose any
       course and teacher from the respective tables.
162
163  UPDATE Courses
164  SET teacher_id = 1
165  WHERE course_id = 3;
166  select * from Courses;
```

133 %

Results | Messages

	course_id	course_name	credits	teacher_id
1	1	Computer Science Fundamentals	4	1
2	2	Indian History and Culture	3	3
3	3	Mathematics for Engineering	5	1
4	4	Environmental Science	3	2
5	5	English Literature	4	6
6	6	Business Management Principles	3	4
7	7	Artificial Intelligence and Machine Learning	4	1
8	8	Economics of India	3	7
9	9	Civil Engineering Design	5	9
10	10	Indian Classical Music	2	8

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.

Assignment-2.sql...ASANTH\vamsh (68))*

```
168  --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.
169
170  DELETE FROM Payments
171  WHERE student_id = 1;
172  DELETE FROM Students
173  WHERE student_id = 1;
174  DELETE FROM Enrollments
175  WHERE student_id = 1;
176  select * from Students;
177  select * from Enrollments;
```

133 %

Results | Messages

	student_id	first_name	last_name	date_of_birth	email	phone_number
1	2	Jane	Smith	1998-08-22	jane.smith@email.com	987-654-3210
2	3	Bob	Johnson	1997-02-15	bob.johnson@email.com	555-123-4567
3	4	Alice	Williams	1996-11-30	alice.williams@email.com	222-333-4444
4	5	Charlie	Brown	1999-04-05	charlie.brown@email.com	111-999-8888
5	6	Emma	Davis	1994-07-18	emma.davis@email.com	777-888-9999
6	7	Michael	Lee	1993-09-25	michael.lee@email.com	666-555-4444
7	8	Sophia	Taylor	2000-01-12	sophia.taylor@email.com	444-222-1111
8	9	David	Miller	1992-12-03	david.miller@email.com	123-987-6543
9	10	Olivia	Moore	1991-06-28	olivia.moore@email.com	999-111-3333
10	11	John	Doe	1995-08-15	john.doe@example.com	1234567890

	enrollment_id	student_id	course_id	enrollment_date
1	2	2	3	2023-02-20
2	3	3	8	2023-03-10
3	4	4	1	2023-04-05
4	6	6	2	2023-06-18
5	7	7	9	2023-07-25
6	8	8	4	2023-08-30
7	9	9	10	2023-09-08
8	10	10	5	2023-10-15

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

```
Assignment-2.sql...ASANTH\vamsh (68))*
179  --7. Update the payment amount for a specific payment record in the "Payments" table.
        Choose any payment record and modify the payment amount.
180
181  UPDATE Payments
182  SET amount = 1200.50
183  WHERE payment_id = 3;
184  select * from Payments;
```

133 %

Results | Messages

	payment_id	student_id	amount	payment_date
1	2	2	750.50	2023-02-10
2	3	3	1200.50	2023-03-15
3	4	4	900.75	2023-04-20
4	5	5	350.00	2023-05-25
5	6	6	800.50	2023-06-30
6	7	7	450.25	2023-07-05
7	8	8	700.75	2023-08-10
8	9	9	550.00	2023-09-15
9	10	10	950.50	2023-10-20

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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
186  --Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:
187
188  --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.
189
190  SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments
191  FROM Students s
192  JOIN Payments p ON s.student_id = p.student_id
193  WHERE s.student_id = 5
194  GROUP BY s.student_id, s.first_name, s.last_name;
```

133 %

Results | Messages

	student_id	first_name	last_name	total_payments
1	5	Charlie	Brown	350.00

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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
196  --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.
197
198  SELECT
199      c.course_id,
200      c.course_name,
201      COUNT(e.student_id) AS enrolled_students_count
202  FROM Courses c
203  LEFT JOIN Enrollments e ON c.course_id = e.course_id
204  GROUP BY c.course_id, c.course_name;
```

133 %

Results | Messages

	course_id	course_name	enrolled_students_count
1	1	Computer Science Fundamentals	1
2	2	Indian History and Culture	1
3	3	Mathematics for Engineering	1
4	4	Environmental Science	1
5	5	English Literature	1
6	6	Business Management Principles	0
7	7	Artificial Intelligence and Machine Learning	0
8	8	Economics of India	1
9	9	Civil Engineering Design	1
10	10	Indian Classical Music	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.

```sql
--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.enrollment_id IS NULL;
```

	student_id	first_name	last_name
1	5	Charlie	Brown
2	11	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.

```sql
--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.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id;
```

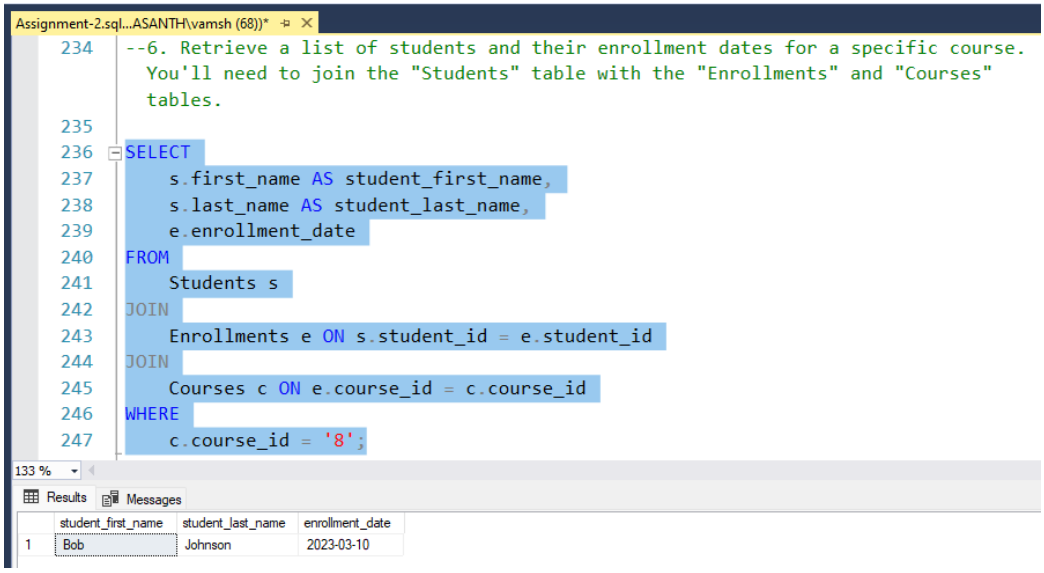
	first_name	last_name	course_name
1	Jane	Smith	Mathematics for Engineering
2	Bob	Johnson	Economics of India
3	Alice	Williams	Computer Science Fundamentals
4	Emma	Davis	Indian History and Culture
5	Michael	Lee	Civil Engineering Design
6	Sophia	Taylor	Environmental Science
7	David	Miller	Indian Classical Music
8	Olivia	Moore	English Literature

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.

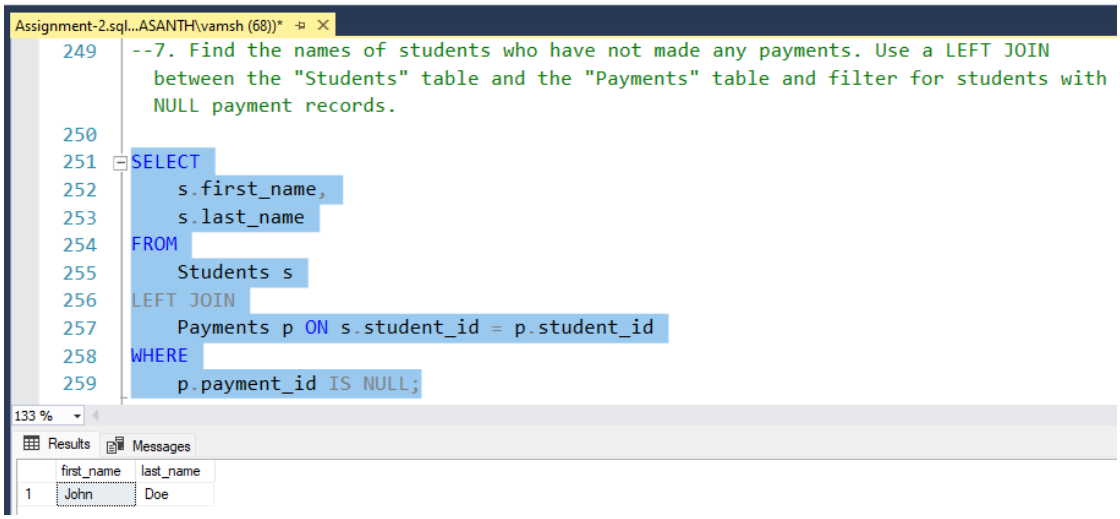
```sql
--5. Create a query to list the names of teachers and the courses they are assigned
  to. Join the "Teacher" table with the "Courses" table.

SELECT
    t.first_name AS teacher_first_name,
    t.last_name AS teacher_last_name,
    c.course_name
FROM
    Teacher t
JOIN
    Courses c ON t.teacher_id = c.teacher_id;
```

	teacher_first_name	teacher_last_name	course_name
1	Amit	Kumar	Computer Science Fundamentals
2	Rahul	Verma	Indian History and Culture
3	Amit	Kumar	Mathematics for Engineering
4	Priya	Sharma	Environmental Science
5	Kavita	Joshi	English Literature
6	Ananya	Srivastava	Business Management Principles
7	Amit	Kumar	Artificial Intelligence and Machine Learning
8	Raj	Patel	Economics of India
9	Ravi	Yadav	Civil Engineering Design
10	Neha	Mishra	Indian Classical Music

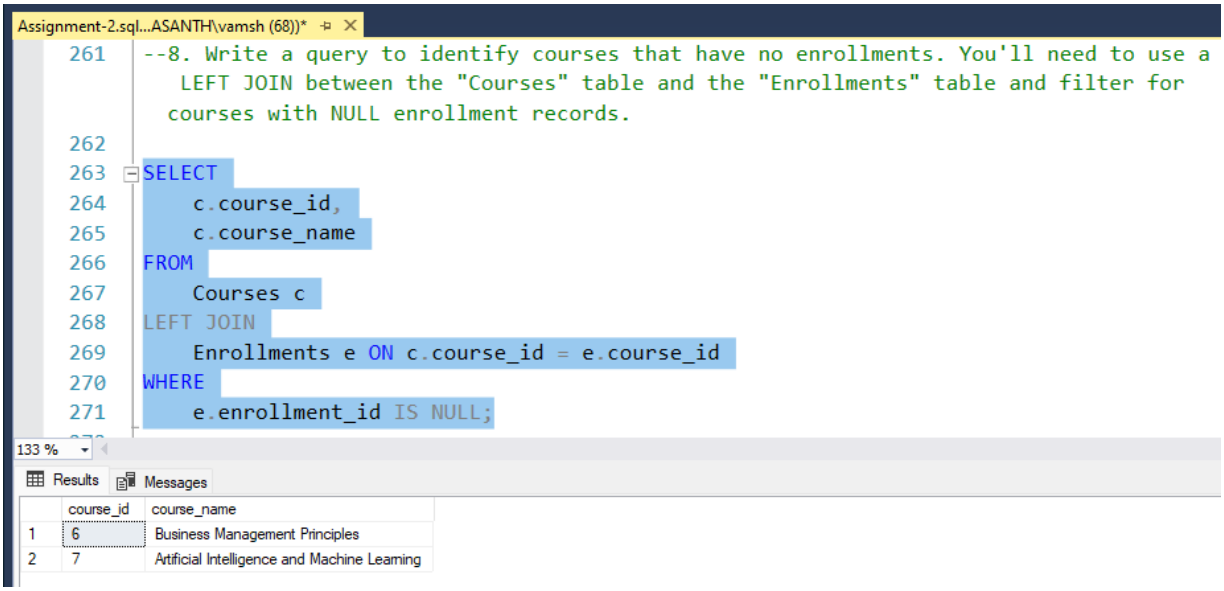
6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.



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.



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.



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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
273  --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.
274
275  SELECT DISTINCT
276      e1.student_id,
277      s.first_name,
278      s.last_name
279  FROM
280      Enrollments e1
281  JOIN
282      Enrollments e2 ON e1.student_id = e2.student_id AND e1.enrollment_id <>
         e2.enrollment_id
283  JOIN
284      Students s ON e1.student_id = s.student_id;
```

133 %

Results | Messages

student_id	first_name	last_name

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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
286  --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.
287
288  SELECT
289      t.teacher_id,
290      t.first_name,
291      t.last_name
292  FROM
293      Teacher t
294  LEFT JOIN
295      Courses c ON t.teacher_id = c.teacher_id
296  WHERE
297      c.teacher_id IS NULL;
```

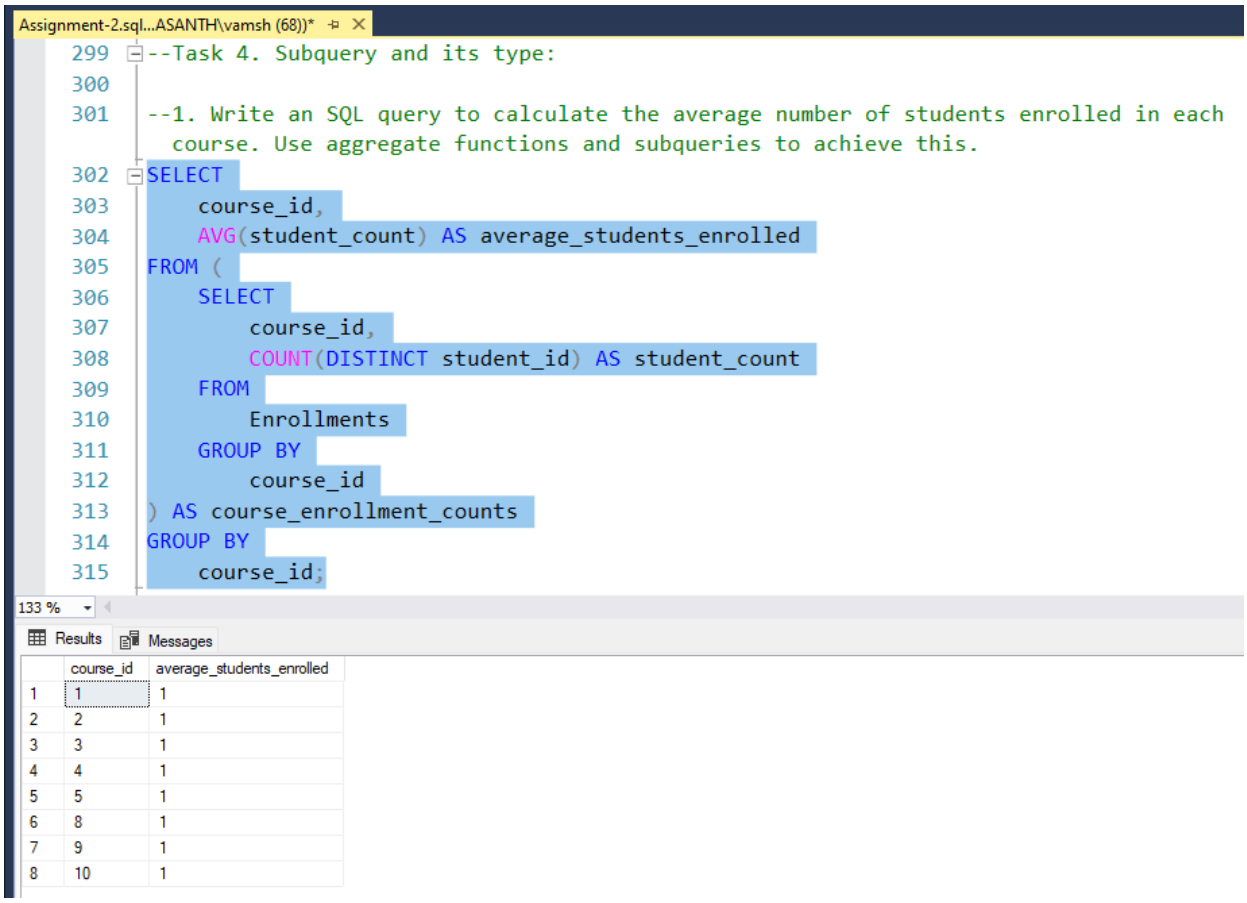
133 %

Results | Messages

	teacher_id	first_name	last_name
1	5	Vikram	Singh
2	10	Sonia	Gupta

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.



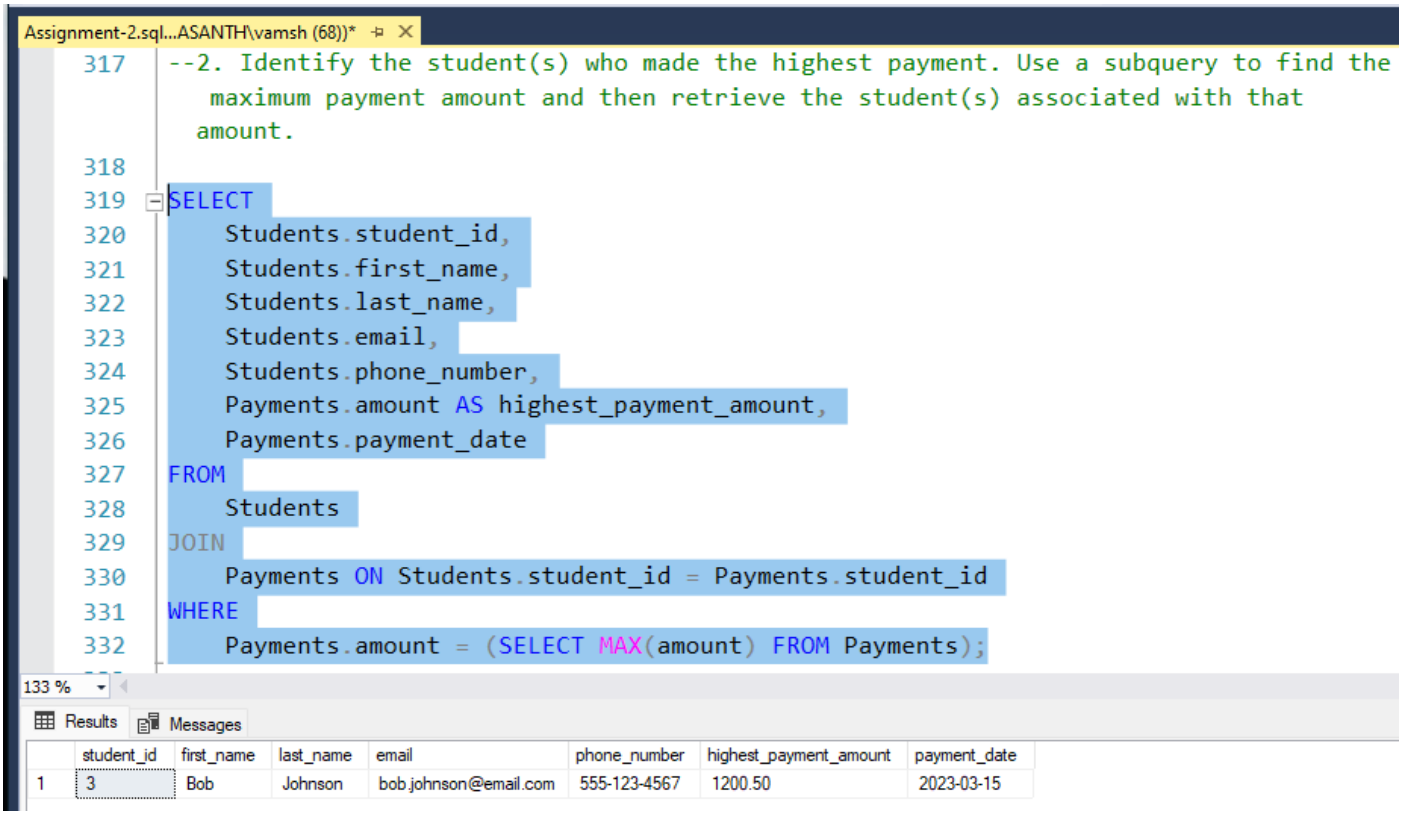
```
Assignment-2.sql...ASANTH\vamsh (68))*
299  --Task 4. Subquery and its type:
300
301  --1. Write an SQL query to calculate the average number of students enrolled in each
       course. Use aggregate functions and subqueries to achieve this.
302  SELECT
303      course_id,
304      AVG(student_count) AS average_students_enrolled
305  FROM (
306      SELECT
307          course_id,
308          COUNT(DISTINCT student_id) AS student_count
309      FROM
310          Enrollments
311      GROUP BY
312          course_id
313  ) AS course_enrollment_counts
314  GROUP BY
315      course_id;
```

133 %

Results | Messages

	course_id	average_students_enrolled
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	8	1
7	9	1
8	10	1

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.



```
Assignment-2.sql...ASANTH\vamsh (68))*
317  --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.
318
319  SELECT
320      Students.student_id,
321      Students.first_name,
322      Students.last_name,
323      Students.email,
324      Students.phone_number,
325      Payments.amount AS highest_payment_amount,
326      Payments.payment_date
327  FROM
328      Students
329  JOIN
330      Payments ON Students.student_id = Payments.student_id
331  WHERE
332      Payments.amount = (SELECT MAX(amount) FROM Payments);
```

133 %

Results | Messages

	student_id	first_name	last_name	email	phone_number	highest_payment_amount	payment_date
1	3	Bob	Johnson	bob.johnson@email.com	555-123-4567	1200.50	2023-03-15

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

```
--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
    Courses.course_id,
    Courses.course_name,
    COUNT(Enrollments.enrollment_id) AS enrollment_count FROM Courses
LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
GROUP BY  Courses.course_id, Courses.course_name
HAVING COUNT(Enrollments.enrollment_id) = (
        SELECT MAX(enrollment_count) FROM
        (SELECT course_id, COUNT(enrollment_id) AS enrollment_count
FROM Enrollments
            GROUP BY course_id) AS max_enrollments);
```

Results | Messages

	course_id	course_name	enrollment_count
1	1	Computer Science Fundamentals	1
2	2	Indian History and Culture	1
3	3	Mathematics for Engineering	1
4	4	Environmental Science	1
5	5	English Literature	1
6	8	Economics of India	1
7	9	Civil Engineering Design	1
8	10	Indian Classical Music	1

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

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

SELECT
    Teacher.teacher_id,
    Teacher.first_name,
    Teacher.last_name,
    sum(payments.amount) as totalpayments
FROM
    Teacher
LEFT JOIN
    Courses ON Teacher.teacher_id = Courses.teacher_id
LEFT JOIN
    Enrollments ON Courses.course_id = Enrollments.course_id
LEFT JOIN
    Payments ON Enrollments.student_id = Payments.student_id
GROUP BY
    Teacher.teacher_id, Teacher.first_name, Teacher.last_name;
```

Results | Messages

	teacher_id	first_name	last_name	totalpayments
1	1	Amit	Kumar	1651.25
2	2	Priya	Sharma	700.75
3	3	Rahul	Verma	800.50
4	4	Ananya	Srivastava	NULL
5	5	Vikram	Singh	NULL
6	6	Kavita	Joshi	950.50
7	7	Raj	Patel	1200.50
8	8	Neha	Mishra	550.00
9	9	Ravi	Yadav	450.25
10	10	Sonia	Gupta	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.

```
366  --5. Identify students who are enrolled in all available courses. Use subqueries to
       compare a student's enrollments with the total number of courses.
367
368  SELECT
369      s.student_id,
370      s.first_name,
371      s.last_name
372  FROM
373      Students s
374  WHERE
375      (SELECT COUNT(DISTINCT e.course_id) FROM  Enrollments e
376       WHERE  e.student_id = s.student_id ) = (
377       SELECT COUNT(DISTINCT c.course_id) FROM Courses c);
```

133 %

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.

```
379  --6. Retrieve the names of teachers who have not been assigned to any courses. Use
       subqueries to find teachers with no course assignments.
380
381  SELECT
382      t.first_name,
383      t.last_name
384  FROM Teacher t
385  WHERE  t.teacher_id NOT IN (SELECT DISTINCT c.teacher_id
386          FROM Courses c
387  WHERE c.teacher_id IS NOT NULL);
```

133 %

Results | Messages

	first_name	last_name
1	Vikram	Singh
2	Sonia	Gupta

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

```
389  --7. Calculate the average age of all students. Use subqueries to calculate the age
       of each student based on their date of birth.
390
391  SELECT
392      AVG(student_age) AS average_age
393  FROM ( SELECT student_id,DATEDIFF(YEAR, date_of_birth, GETDATE()) AS student_age
394      FROM  Students ) AS average_age;
```

133 %

Results | Messages

	average_age
1	27

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

```
Assignment-2.sql...ASANTH\vamsh (68))*
396  --8. Identify courses with no enrollments. Use subqueries to find courses without
       enrollment records.
397
398  SELECT c.course_id, c.course_name FROM Courses c
399  WHERE c.course_id NOT IN (
400     SELECT  DISTINCT e.course_id FROM
401     Enrollments e);
```

133 %

Results | Messages

	course_id	course_name
1	6	Business Management Principles
2	7	Artificial Intelligence and Machine Learning

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

```
Assignment-2.sql...ASANTH\vamsh (68))*
403  --9. Calculate the total payments made by each student for each course they are
       enrolled in. Use subqueries and aggregate functions to sum payments.
404
405  SELECT
406      s.student_id, s.first_name,s.last_name,c.course_id,c.course_name,SUM(p.amount)
407      AS total_payments FROM  Students s
408  JOIN Enrollments e ON s.student_id = e.student_id
409  JOIN Courses c ON e.course_id = c.course_id
410  LEFT JOIN  Payments p ON s.student_id = p.student_id AND c.course_id = e.course_id
411  GROUP BY s.student_id, s.first_name, s.last_name, c.course_id, c.course_name;
```

133 %

Results | Messages

	student_id	first_name	last_name	course_id	course_name	total_payments
1	4	Alice	Williams	1	Computer Science Fundamentals	900.75
2	6	Emma	Davis	2	Indian History and Culture	800.50
3	2	Jane	Smith	3	Mathematics for Engineering	750.50
4	8	Sophia	Taylor	4	Environmental Science	700.75
5	10	Olivia	Moore	5	English Literature	950.50
6	3	Bob	Johnson	8	Economics of India	1200.50
7	7	Michael	Lee	9	Civil Engineering Design	450.25
8	9	David	Miller	10	Indian Classical Music	550.00

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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
413  --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.
414
415  SELECT s.student_id,s.first_name,s.last_name
416     FROM Students s
417  JOIN (SELECT student_id FROM Payments
418  GROUP BY student_id
419  HAVING COUNT(*) > 1)
420   AS payment_counts ON s.student_id = payment_counts.student_id;
```

133 %

Results | Messages

student_id	first_name	last_name

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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
422  --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.
423
424  SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount)
425  AS total_payments
426  FROM Students s
427  LEFT JOIN Payments p ON s.student_id = p.student_id
428  GROUP BY s.student_id, s.first_name, s.last_name;
```

133 %

Results | Messages

	student_id	first_name	last_name	total_payments
1	2	Jane	Smith	750.50
2	3	Bob	Johnson	1200.50
3	4	Alice	Williams	900.75
4	5	Charlie	Brown	350.00
5	6	Emma	Davis	800.50
6	7	Michael	Lee	450.25
7	8	Sophia	Taylor	700.75
8	9	David	Miller	550.00
9	10	Olivia	Moore	950.50
10	11	John	Doe	NULL

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.

```
Assignment-2.sql...ASANTH\vamsh (68))*
430  --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.
431
432  SELECT c.course_id, c.course_name, COUNT(e.student_id)
433  AS enrolled_students_count
434  FROM Courses c
435  LEFT JOIN Enrollments e ON c.course_id = e.course_id
436  GROUP BY c.course_id, c.course_name
```

133 %

Results | Messages

	course_id	course_name	enrolled_students_count
1	1	Computer Science Fundamentals	1
2	2	Indian History and Culture	1
3	3	Mathematics for Engineering	1
4	4	Environmental Science	1
5	5	English Literature	1
6	6	Business Management Principles	0
7	7	Artificial Intelligence and Machine Learning	0
8	8	Economics of India	1
9	9	Civil Engineering Design	1
10	10	Indian Classical Music	1

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.

Assignment-2.sql...ASANTH\vamsh (68))*

```
438  --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.
439
440  SELECT s.student_id, s.first_name, s.last_name, AVG(p.amount)
441  AS average_payment_amount
442  FROM Students s
443  JOIN Payments p ON s.student_id = p.student_id
444  GROUP BY s.student_id, s.first_name, s.last_name;
```

133 %

Results | Messages

	student_id	first_name	last_name	average_payment_amount
1	2	Jane	Smith	750.500000
2	3	Bob	Johnson	1200.500000
3	4	Alice	Williams	900.750000
4	5	Charlie	Brown	350.000000
5	6	Emma	Davis	800.500000
6	7	Michael	Lee	450.250000
7	8	Sophia	Taylor	700.750000
8	9	David	Miller	550.000000
9	10	Olivia	Moore	950.500000