

# ASSIGNMENT 2:

P Manideep

## Task 1. Database Design:

1. Create the database named "SISDB"

The screenshot shows a SQL query window titled "Assignment-2.sql...ASANTH\vamsh (66)\*". The code is as follows:

```
1 --TASK 1
2
3 -- 1. Create the database named "SISDB"
4
5 create database SIS;
6 use SIS;
```

The "Messages" pane at the bottom shows the output: "Commands completed successfully." and the completion time: "Completion time: 2023-12-10T10:46:20.5825002+05:30".

2. Define the schema for the Students, Courses, Enrolments, 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

The screenshot shows a SQL query window titled "Assignment-2.sql...ASANTH\vamsh (66)\*". The code is as follows:

```
8 /*2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments
 9  tables based on the provided schema. Write SQL scripts to create the mentioned
10  tables with appropriate data types, constraints, and relationships.
11
12 a. Students
13 b. Courses
14 c. Enrollments
15 d. Teacher
16 e. Payments*/
17
18 CREATE TABLE Students (
19     student_id INT PRIMARY KEY,
20     first_name VARCHAR(255),
21     last_name VARCHAR(255),
22     date_of_birth DATE,
23     email VARCHAR(255),
24     phone_number VARCHAR(15));
```

The "Messages" pane at the bottom shows the output: "Commands completed successfully." and the completion time: "Completion time: 2023-12-10T10:46:55.4067565+05:30".

b. Courses

The screenshot shows a SQL query window titled "Assignment-2.sql...ASANTH\vamsh (68)\*". The code is as follows:

```
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));
```

The "Messages" pane at the bottom shows the output: "Commands completed successfully." and the completion time: "Completion time: 2023-12-10T10:50:54.3098168+05:30".

c. Enrolments:

The screenshot shows a SQL editor window titled "Assignment-2.sql...ASANTH\varmsh (68)\*". The code being run is:

```
30 CREATE TABLE Enrollments (
31     enrollment_id INT PRIMARY KEY,
32     student_id INT,
33     course_id INT,
34     enrollment_date DATE,
35     FOREIGN KEY (student_id) REFERENCES Students(student_id),
36     FOREIGN KEY (course_id) REFERENCES Courses(course_id));
```

The status bar at the bottom indicates "Commands completed successfully." and a completion time of "Completion time: 2023-12-10T10:53:03.3619171+05:30".

d. Teacher

The screenshot shows a SQL editor window titled "Assignment-2.sql...ASANTH\varmsh (68)\*". The code being run is:

```
38 CREATE TABLE Teacher (
39     teacher_id INT PRIMARY KEY,
40     first_name VARCHAR(255),
41     last_name VARCHAR(255),
42     email VARCHAR(255));
```

The status bar at the bottom indicates "Commands completed successfully." and a completion time of "Completion time: 2023-12-10T10:50:13.0555703+05:30".

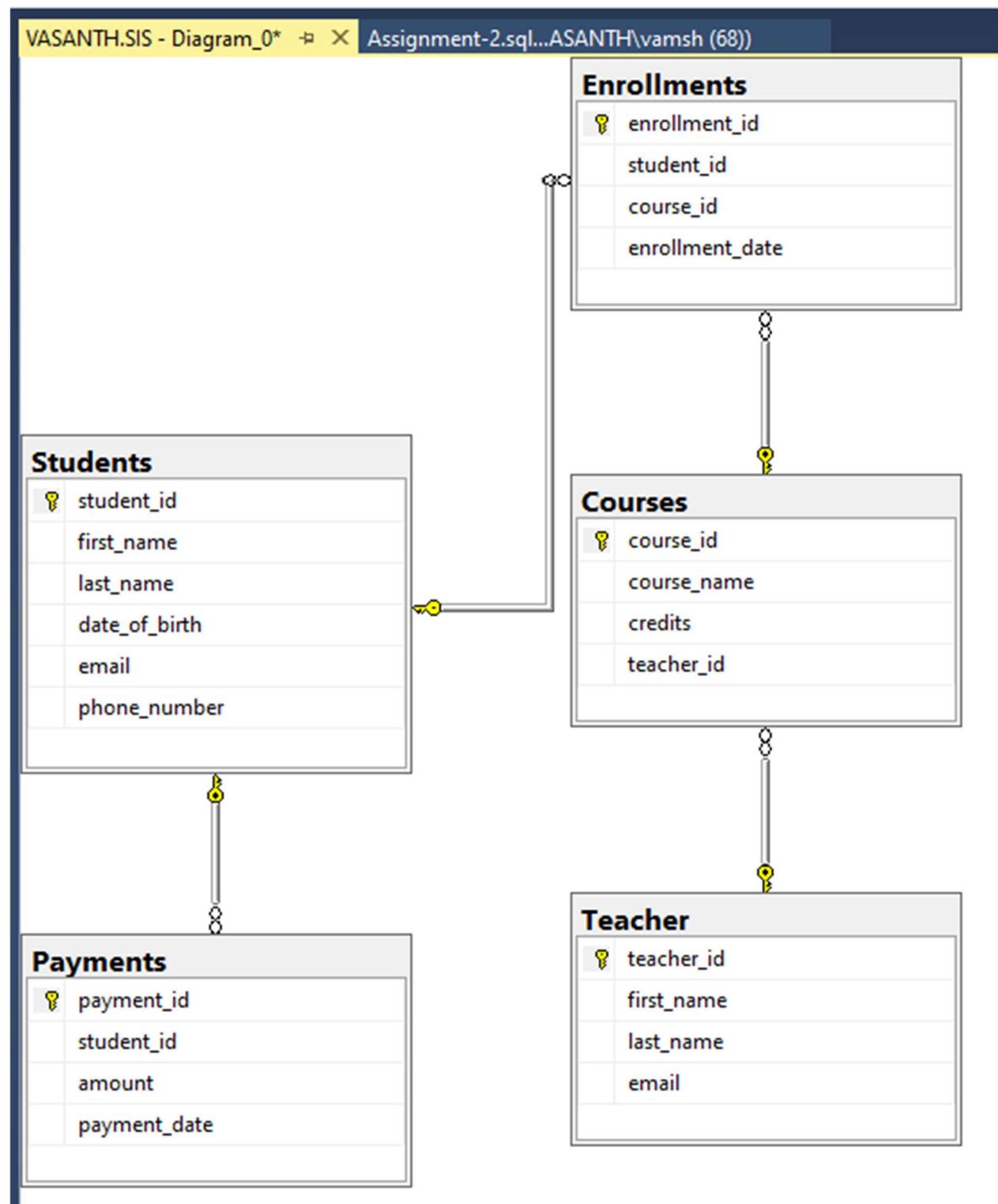
e. Payments

The screenshot shows a SQL editor window titled "Assignment-2.sql...ASANTH\varmsh (68)\*". The code being run is:

```
44 CREATE TABLE Payments (
45     payment_id INT PRIMARY KEY,
46     student_id INT,
47     amount DECIMAL(10, 2),
48     payment_date DATE,
49     FOREIGN KEY (student_id) REFERENCES Students(student_id));
```

The status bar at the bottom indicates "Commands completed successfully." and a completion time of "Completion time: 2023-12-10T10:53:34.8671306+05:30".

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

```
Assignment-2.sql...ASANTH\avamsh (68)* ✎ X
51 /*5. Insert at least 10 sample records into each of the following tables.
52 i. Students
53 ii. Courses
54 iii. Enrollments
55 iv. Teacher
56 v. Payments*/
57
58 INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email,
59   phone_number)
60 VALUES
61 (1, 'Rohan', 'jami', '1995-05-10', 'rohan.jami@email.com', '147-852-9630'),
62 (2, 'Jane', 'Smith', '1998-08-22', 'jane.smith@email.com', '987-654-3210'),
63 (3, 'Bob', 'Johnson', '1997-02-15', 'bob.johnson@email.com', '555-123-4567'),
64 (4, 'Alice', 'Williams', '1996-11-30', 'alice.williams@email.com', '222-333-4444'),
65 (5, 'Charlie', 'Brown', '1999-04-05', 'charlie.brown@email.com', '111-999-8888'),
66 (6, 'Emma', 'Davis', '1994-07-18', 'emma.davis@email.com', '777-888-9999'),
67 (7, 'Michael', 'Lee', '1993-09-25', 'michael.lee@email.com', '666-555-4444'),
68 (8, 'Sophia', 'Taylor', '2000-01-12', 'sophia.taylor@email.com', '444-222-1111'),
69 (9, 'David', 'Miller', '1992-12-03', 'david.miller@email.com', '123-987-6543'),
70 (10, 'Olivia', 'Moore', '1991-06-28', 'olivia.moore@email.com', '999-111-3333);
71 select * from Students;
```

Results Messages

student_id	first_name	last_name	date_of_birth	email	phone_number
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

ii. Courses

```
Assignment-2.sql...ASANTH\avamsh (68)* ✎ X
86 INSERT INTO Courses (course_id, course_name, credits, teacher_id)
87 VALUES
88 (1, 'Computer Science Fundamentals', 4, 1),
89 (2, 'Indian History and Culture', 3, 3),
90 (3, 'Mathematics for Engineering', 5, 5),
91 (4, 'Environmental Science', 3, 2),
92 (5, 'English Literature', 4, 6),
93 (6, 'Business Management Principles', 3, 4),
94 (7, 'Artificial Intelligence and Machine Learning', 4, 1),
95 (8, 'Economics of India', 3, 7),
96 (9, 'Civil Engineering Design', 5, 9),
97 (10, 'Indian Classical Music', 2, 8);
98 select * from Courses;
```

Results Messages

course_id	course_name	credits	teacher_id
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

### iii. Enrolments

```
Assignment-2.sql...ASANTH\vatmsh (68)* ↵ X
100 INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
101 VALUES
102 (1, 1, 7, '2023-01-15'),
103 (2, 2, 3, '2023-02-20'),
104 (3, 3, 8, '2023-03-10'),
105 (4, 4, 1, '2023-04-05'),
106 (5, 5, 6, '2023-05-12'),
107 (6, 6, 2, '2023-06-18'),
108 (7, 7, 9, '2023-07-25'),
109 (8, 8, 4, '2023-08-30'),
110 (9, 9, 10, '2023-09-08'),
111 (10, 10, 5, '2023-10-15');
112 select * from Enrollments;
```

133 %

	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

```
Assignment-2.sql...ASANTH\vatmsh (68)* ↵ X
72 INSERT INTO Teacher (teacher_id, first_name, last_name, email)
73 VALUES
74 (1, 'Amit', 'Kumar', 'amit.kumar@email.com'),
75 (2, 'Priya', 'Sharma', 'priya.sharma@email.com'),
76 (3, 'Rahul', 'Verma', 'rahul.verma@email.com'),
77 (4, 'Ananya', 'Srivastava', 'ananya.srivastava@email.com'),
78 (5, 'Vikram', 'Singh', 'vikram.singh@email.com'),
79 (6, 'Kavita', 'Joshi', 'kavita.joshi@email.com'),
80 (7, 'Raj', 'Patel', 'raj.patel@email.com'),
81 (8, 'Neha', 'Mishra', 'neha.mishra@email.com'),
82 (9, 'Ravi', 'Yadav', 'ravi.yadav@email.com'),
83 (10, 'Sonia', 'Gupta', 'sonia.gupta@email.com');
84 select * from Teacher;
```

133 %

	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

```

Assignment-2.sql...ASANTH\varmsh (68)* ✘ X
114 INSERT INTO Payments (payment_id, student_id, amount, payment_date)
115 VALUES
116 (1, 1, 500.00, '2023-01-05'),
117 (2, 2, 750.50, '2023-02-10'),
118 (3, 3, 600.25, '2023-03-15'),
119 (4, 4, 900.75, '2023-04-20'),
120 (5, 5, 350.00, '2023-05-25'),
121 (6, 6, 800.50, '2023-06-30'),
122 (7, 7, 450.25, '2023-07-05'),
123 (8, 8, 700.75, '2023-08-10'),
124 (9, 9, 550.00, '2023-09-15'),
125 (10, 10, 950.50, '2023-10-20');
126 select * from Payments;
  
```

The screenshot shows an SQL query being run in a query editor. The query inserts 10 rows into the 'Payments' table with columns: payment\_id, student\_id, amount, and payment\_date. The results are displayed in a table below the query.

	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

```

Assignment-2.sql...ASANTH\varmsh (68)* ✘ X
128 -- Tasks 2: Select, Where, Between, AND, LIKE:
129
130 /*1. Write an SQL query to insert a new student into the "Students" table with the
   following details:
131 a. First Name: John
132 © Hexaware Technologies Limited. All rights www.hexaware.com
133 b. Last Name: Doe
134 c. Date of Birth: 1995-08-15
135 d. Email: john.doe@example.com
136 e. Phone Number: 1234567890*/
137
138 INSERT INTO Students (student_id,first_name, last_name, date_of_birth, email,
139   phone_number)
140 VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
141 select * from Students;
  
```

The screenshot shows an SQL query being run in a query editor. The query inserts a new row into the 'Students' table with columns: student\_id, first\_name, last\_name, date\_of\_birth, email, and phone\_number. The results are displayed in a table below the query.

	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 enrol a student in a course. Choose an existing student and course and insert a record into the "Enrolments" table with the enrolment date.

```
Assignment-2.sql...ASANTH\yamsh (68)* ↵ X
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;
```

Results

	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\yamsh (68)* ↵ X
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;
```

Results

	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	Vema	rahul.vema@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 enrolment record from the "Enrolments" table. Select an enrolment record based on the student and course.

```
Assignment-2.sql...ASANTH\yamsh (68)* ↵ X
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;
```

Results

	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\varunsh (68)* ↵ X
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;
```

Results

course_id	course_name	credits	teacher_id
1	Computer Science Fundamentals	4	1
2	Indian History and Culture	3	3
3	Mathematics for Engineering	5	1
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

6. Delete a specific student from the "Students" table and remove all their enrolment records from the "Enrolments" table. Be sure to maintain referential integrity.

```
Assignment-2.sql...ASANTH\varunsh (68)* ↵ X
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;
```

Results

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

Results

enrollment_id	student_id	course_id	enrollment_date
1	2	3	2023-02-20
2	3	8	2023-03-10
3	4	1	2023-04-05
4	6	2	2023-06-18
5	7	9	2023-07-25
6	8	4	2023-08-30
7	9	10	2023-09-08
8	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\yamsh (68)* -> X
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;
```

Results

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

### Task 3. Aggregate functions, Having, Order By, Group by 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\yamsh (68)* -> X
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;
```

Results

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\yamsh (68)* -> X
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;
```

Results

course_id	course_name	enrolled_students_count
1	Computer Science Fundamentals	1
2	Indian History and Culture	1
3	Mathematics for Engineering	1
4	Environmental Science	1
5	English Literature	1
6	Business Management Principles	0
7	Artificial Intelligence and Machine Learning	0
8	Economics of India	1
9	Civil Engineering Design	1
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 "Enrolments" table to identify students without enrolments.

The screenshot shows an SQL query being run in a query editor. The code is as follows:

```
206 --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 "Enrolments" table to identify students without enrolments.
207
208 SELECT s.student_id, s.first_name, s.last_name
209 FROM Students s
210 LEFT JOIN Enrolments e ON s.student_id = e.student_id
211 WHERE e.enrollment_id IS NULL;
```

The results pane shows a table with two rows:

student_id	first_name	last_name
5	Charlie	Brown
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 "Enrolments" and "Courses" tables.

The screenshot shows an SQL query being run in a query editor. The code is as follows:

```
213 --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 "Enrolments" and "Courses" tables.
214
215 SELECT
216     s.first_name,
217     s.last_name,
218     c.course_name
219 FROM Students s
220 JOIN Enrolments e ON s.student_id = e.student_id
221 JOIN Courses c ON e.course_id = c.course_id;
```

The results pane shows a table with 8 rows:

first_name	last_name	course_name
Jane	Smith	Mathematics for Engineering
Bob	Johnson	Economics of India
Alice	Williams	Computer Science Fundamentals
Emma	Davis	Indian History and Culture
Michael	Lee	Civil Engineering Design
Sophia	Taylor	Environmental Science
David	Miller	Indian Classical Music
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.

The screenshot shows an SQL query being run in a query editor. The code is as follows:

```
223 --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.
224
225 SELECT
226     t.first_name AS teacher_first_name,
227     t.last_name AS teacher_last_name,
228     c.course_name
229 FROM
230     Teacher t
231 JOIN
232     Courses c ON t.teacher_id = c.teacher_id;
```

The results pane shows a table with 10 rows:

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

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

The screenshot shows a SQL query in the 'Assignment-2.sql' file. The code is as follows:

```
--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.first_name AS student_first_name,  
    s.last_name AS student_last_name,  
    e.enrollment_date  
FROM  
    Students s  
JOIN  
    Enrollments e ON s.student_id = e.student_id  
JOIN  
    Courses c ON e.course_id = c.course_id  
WHERE  
    c.course_id = '8';
```

The results pane shows a single row of data:

	student_first_name	student_last_name	enrollment_date
1	Bob	Johnson	2023-03-10

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.

The screenshot shows a SQL query in the 'Assignment-2.sql' file. The code is as follows:

```
--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.first_name,  
    s.last_name  
FROM  
    Students s  
LEFT JOIN  
    Payments p ON s.student_id = p.student_id  
WHERE  
    p.payment_id IS NULL;
```

The results pane shows a single row of data:

	first_name	last_name
1	John	Doe

8. Write a query to identify courses that have no enrolments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrolments" table and filter for courses with NULL enrolment records.

The screenshot shows a SQL query in the 'Assignment-2.sql' file. The code is as follows:

```
--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  
FROM  
    Courses c  
LEFT JOIN  
    Enrollments e ON c.course_id = e.course_id  
WHERE  
    e.enrollment_id IS NULL;
```

The results pane shows two rows of data:

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

9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrolments" table to find students with multiple enrolment records.

```
Assignment-2.sql...ASANTH\vatmash (68)* ➔ X
273 --9. Identify students who are enrolled in more than one course. Use a self-join on
274   the "Enrolments" table to find students with multiple enrollment records.
275
276 SELECT DISTINCT
277   e1.student_id,
278   s.first_name,
279   s.last_name
280
281 FROM
282   Enrolments e1
283 JOIN
284   Enrolments e2 ON e1.student_id = e2.student_id AND e1.enrollment_id <>
285   e2.enrollment_id
286 JOIN
287   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\vatmash (68)* ➔ X
286 --10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the
287   "Teacher" table and the "Courses" table and filter for teachers with NULL course
288   assignments.
289
290 SELECT
291   t.teacher_id,
292   t.first_name,
293   t.last_name
294
295 FROM
296   Teacher t
297 LEFT JOIN
298   Courses c ON t.teacher_id = c.teacher_id
299 WHERE
300   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.

The screenshot shows an SQL query in the 'Assignment-2.sql...ASANTH\yamsh (68)\*' window. The code uses a subquery to calculate the count of distinct student IDs for each course, which is then used in an outer query to calculate the average number of students enrolled per course. The results are displayed in a 'Results' grid.

```
Assignment-2.sql...ASANTH\yamsh (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 course_id, AVG(student_count) AS average_students_enrolled 303 FROM ( 304     SELECT course_id, COUNT(DISTINCT student_id) AS student_count 305     FROM Enrollments 306     GROUP BY course_id 307 ) AS course_enrollment_counts 308     GROUP BY course_id; 309 310 311 312 313 314 315
```

course_id	average_students_enrolled
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	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.

The screenshot shows an SQL query in the 'Assignment-2.sql...ASANTH\yamsh (68)\*' window. The code uses a subquery to find the maximum payment amount and then joins it with the 'Students' and 'Payments' tables to retrieve the student details. The results are displayed in a 'Results' grid.

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

student_id	first_name	last_name	email	phone_number	highest_payment_amount	payment_date
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 enrolments. Use subqueries to find the course(s) with the maximum enrolment count.

```
Assignment-2.sql...ASANTH\avamsh (68)* ↵ X
334 --3. Retrieve a list of courses with the highest number of enrollments. Use
335     subqueries to find the course(s) with the maximum enrollment count.
336     SELECT
337         Courses.course_id,
338         Courses.course_name,
339         COUNT(Enrollments.enrollment_id) AS enrollment_count FROM Courses
340     LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
341     GROUP BY Courses.course_id, Courses.course_name
342     HAVING COUNT(Enrollments.enrollment_id) = (
343             SELECT MAX(enrollment_count) FROM
344             (SELECT course_id, COUNT(enrollment_id) AS enrollment_count
345             FROM Enrollments
346             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.

```
Assignment-2.sql...ASANTH\avamsh (68)* ↵ X
348 --4. Calculate the total payments made to courses taught by each teacher. Use
349     subqueries to sum payments for each teacher's courses.
350     SELECT
351         Teacher.teacher_id,
352         Teacher.first_name,
353         Teacher.last_name,
354         sum(payments.amount) as totalpayments
355     FROM
356         Teacher
357     LEFT JOIN
358         Courses ON Teacher.teacher_id = Courses.teacher_id
359     LEFT JOIN
360         Enrollments ON Courses.course_id = Enrollments.course_id
361     LEFT JOIN
362         Payments ON Enrollments.student_id = Payments.student_id
363     GROUP BY
364         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 enrolments with the total number of courses.

```
Assignment-2.sql...ASANTH\yamsh (68)* ↵ X
366 --5. Identify students who are enrolled in all available courses. Use subqueries to
367   compare a student's enrolments with the total number of courses.
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.

```
Assignment-2.sql...ASANTH\yamsh (68)* ↵ X
379 --6. Retrieve the names of teachers who have not been assigned to any courses. Use
380   subqueries to find teachers with no course assignments.
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.

```
Assignment-2.sql...ASANTH\yamsh (68)* ↵ X
389 --7. Calculate the average age of all students. Use subqueries to calculate the age
390   of each student based on their date of birth.
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 enrolments. Use subqueries to find courses without enrollment records.

```
Assignment-2.sql...ASANTH\yamsh (68)* ↵ X
396 --8. Identify courses with no enrolments. 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\yamsh (68)* ↵ X
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\yamsh (68)* ↵ X
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\yamsh (68)* ↵ X
422 --11. Write an SQL query to calculate the total payments made by each student. Join
423   the "Students" table with the "Payments" table and use GROUP BY to calculate the
424   sum of payments for each student.
425
426 SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount)
427   AS total_payments
428 FROM Students s
429 LEFT JOIN Payments p ON s.student_id = p.student_id
430 GROUP BY s.student_id, s.first_name, s.last_name;
```

Results

	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\yamsh (68)* ↵ X
430 --12. Retrieve a list of course names along with the count of students enrolled in
431   each course. Use JOIN operations between the "Courses" table and the "Enrollments"
432   table and GROUP BY to count enrollments.
433
434 SELECT c.course_id, c.course_name, COUNT(e.student_id)
435   AS enrolled_students_count
436 FROM Courses c
437 LEFT JOIN Enrollments e ON c.course_id = e.course_id
438 GROUP BY c.course_id, c.course_name;
```

Results

	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\wamsh (68)*  ▾ X
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