SQL ASSIGNMENT
2
NAME : VENKATA SAI SUKHESH

Creating database:

```
mysql> CREATE DATABASE SISDB2;
Query OK, 1 row affected (0.01 sec)

mysql> USE SISDB2;
Database changed
```

Creating Tables:

```
mysql> CREATE TABLE Students (
    ->     student_id INT PRIMARY KEY,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     date_of_birth DATE,
    ->     email VARCHAR(100),
    ->     phone_number VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Teacher (
    ->     teacher_id INT PRIMARY KEY,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Courses (
    ->     course_id INT PRIMARY KEY,
    ->     course_name VARCHAR(50),
    ->     credits INT,
    ->     teacher_id INT,
    ->     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Enrollments (
    ->     enrollment_id INT PRIMARY KEY,
    ->     student_id INT,
    ->     course_id INT,
    ->     enrollment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id),
    ->     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Payments (
    ->     payment_id INT PRIMARY KEY,
    ->     student_id INT,
    ->     amount DECIMAL(10, 2),
    ->     payment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id)
    -> );
```

Tash 1:

Inserting Values into Tables:

```
mysql> INSERT INTO Students VALUES
    -> (1, 'John', 'Doe', '1990-01-01', 'john.doe@email.com', '123-456-7890'),
    -> (2, 'Arun', 'Kumar', '1992-05-15', 'arun.kumar@email.com', '987-654-3210'),
    -> (3, 'Priya', 'Menon', '1991-08-20', 'priya.menon@email.com', '876-543-2109'),
    -> (4, 'Rajesh', 'Nair', '1993-03-10', 'rajesh.nair@email.com', '765-432-1098'),
    -> (5, 'Deepa', 'Sundar', '1994-12-05', 'deepa.sundar@email.com', '654-321-0987'),
    -> (6, 'Vijay', 'Chandran', '1990-11-25', 'vijay.chandran@email.com', '543-210-9876
'),
    -> (7, 'Lakshmi', 'Krishnan', '1992-07-08', 'lakshmi.krishnan@email.com', '432-109-
8765'),
    -> (8, 'Suresh', 'Rao', '1995-02-18', 'suresh.rao@email.com', '321-098-7654'),
    -> (9, 'Anjali', 'Pillai', '1993-09-30', 'anjali.pillai@email.com', '210-987-6543')
,
    -> (10, 'Karthik', 'Menon', '1991-04-12', 'karthik.menon@email.com', '109-876-5432'
);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Teacher VALUES
    -> (1, 'Jane', 'Smith', 'jane.smith@email.com'),
    -> (2, 'Aishwarya', 'Menon', 'aishwarya.menon@email.com'),
    -> (3, 'Prakash', 'Nair', 'prakash.nair@email.com'),
    -> (4, 'Divya', 'Sundar', 'divya.sundar@email.com'),
    -> (5, 'Manoj', 'Chandran', 'manoj.chandran@email.com'),
    -> (6, 'Sarita', 'Krishnan', 'sarita.krishnan@email.com'),
    -> (7, 'Ramesh', 'Rao', 'ramesh.rao@email.com'),
    -> (8, 'Ananya', 'Pillai', 'ananya.pillai@email.com'),
    -> (9, 'Vikram', 'Menon', 'vikram.menon@email.com'),
    -> (10, 'Shalini', 'Menon', 'shalini.menon@email.com');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Courses VALUES
    -> (1, 'Computer Science', 3, 1),
    -> (2, 'Electrical Engineering', 4, 2),
    -> (3, 'Mechanical Engineering', 3, 3),
    -> (4, 'Civil Engineering', 3, 4),
    -> (5, 'Chemical Engineering', 4, 5),
    -> (6, 'Information Technology', 3, 6),
    -> (7, 'Electronics and Communication', 4, 7),
    -> (8, 'Aerospace Engineering', 4, 8),
    -> (9, 'Biotechnology', 3, 9),
    -> (10, 'Computer Science', 3, 10);
Query OK, 10 rows affected (0.01 sec)
```

```
mysql> INSERT INTO Enrollments VALUES
    -> (1, 1, 1, '2023-01-01'),
    -> (2, 2, 2, '2023-02-01'),
    -> (3, 3, 3, '2023-02-15'),
    -> (4, 4, 4, '2023-03-01'),
    -> (5, 5, 5, '2023-03-15'),
    -> (6, 6, 6, '2023-04-01'),
    -> (7, 7, 7, '2023-04-15'),
    -> (8, 8, 8, '2023-05-01'),
    -> (9, 9, 9, '2023-05-15');
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Payments VALUES
    -> (1, 1, 500.00, '2023-01-15'),
    -> (2, 2, 600.00, '2023-02-10'),
    -> (3, 3, 750.00, '2023-02-25'),
    -> (4, 4, 900.00, '2023-03-10'),
    -> (5, 5, 550.00, '2023-03-25'),
    -> (6, 6, 700.00, '2023-04-10'),
    -> (7, 7, 800.00, '2023-04-25'),
    -> (8, 8, 950.00, '2023-05-10'),
    -> (9, 9, 500.00, '2023-05-25'),
    -> (10, 10, 850.00, '2023-06-10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Task 2:

1. Write an SQL query to insert a new student into the "Students" table with the following details:

```
mysql> INSERT INTO Students (Student_id,first_name, last_name, date_of_birth, email, ph
one_number)
    -> VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.01 sec)
```

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.

```
mysql> UPDATE Teacher
    -> SET email = 'new_email@example.com'
    -> WHERE teacher_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

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

```
mysql> DELETE FROM Enrollments
    -> WHERE enrollment_id = 1;
Query OK, 1 row affected (0.01 sec)
```

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.

```
mysql> UPDATE Courses
    -> SET teacher_id = 2
    -> WHERE course_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

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

```
mysql> DELETE FROM Students
    -> WHERE student_id = 11;
Query OK, 1 row affected (0.01 sec)
```

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.

```
mysql> UPDATE Payments
    -> SET amount = 700.00
    -> WHERE payment_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Task 3:

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.

```
mysql> SELECT Students.first_name, Students.last_name, SUM(Payments.amount) as total_pa
yments
    -> FROM Students
    -> JOIN Payments ON Students.student_id = Payments.student_id
    -> WHERE Students.student_id = 1;
+------------+-----------+---------------+
| first_name | last_name | total_payments |
+------------+-----------+---------------+
| John       | Doe       |         700.00 |
+------------+-----------+---------------+
1 row in set (0.00 sec)
```

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.

```
mysql> SELECT Courses.course_name, COUNT(Enrollments.student_id) as enrolled_students
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> GROUP BY Courses.course_name;
+-----------------------------+-------------------+
| course_name                 | enrolled_students |
+-----------------------------+-------------------+
| Computer Science            |                 1 |
| Electrical Engineering      |                 1 |
| Mechanical Engineering      |                 1 |
| Civil Engineering           |                 1 |
| Chemical Engineering        |                 1 |
| Information Technology      |                 1 |
| Electronics and Communication |                 1 |
| Aerospace Engineering       |                 1 |
| Biotechnology               |                 1 |
+-----------------------------+-------------------+
9 rows in set (0.00 sec)
```

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.

```
mysql> SELECT Students.first_name, Students.last_name
    -> FROM Students
    -> LEFT JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> WHERE Enrollments.student_id IS NULL;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Karthik    | Menon     |
+------------+-----------+
1 row in set (0.00 sec)
```

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.

```
mysql> SELECT Teacher.first_name, Teacher.last_name, Courses.course_name
    -> FROM Teacher
    -> JOIN Courses ON Teacher.teacher_id = Courses.teacher_id;
+------------+-----------+-------------------------------+
| first_name | last_name | course_name                   |
+------------+-----------+-------------------------------+
| Aishwarya  | Menon     | Computer Science              |
| Aishwarya  | Menon     | Electrical Engineering        |
| Prakash    | Nair      | Mechanical Engineering        |
| Divya      | Sundar    | Civil Engineering             |
| Manoj      | Chandran  | Chemical Engineering          |
| Sarita     | Krishnan  | Information Technology        |
| Ramesh     | Rao       | Electronics and Communication |
| Ananya     | Pillai    | Aerospace Engineering         |
| Vikram     | Menon     | Biotechnology                 |
| Shalini    | Menon     | Computer Science              |
+------------+-----------+-------------------------------+
10 rows in set (0.00 sec)
```

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.

```
mysql> SELECT Students.first_name, Students.last_name, Enrollments.enrollment_date
    -> FROM Students
    -> JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> WHERE Enrollments.course_id = 1;
+------------+-----------+-----------------+
| first_name | last_name | enrollment_date |
+------------+-----------+-----------------+
| John       | Doe       | 2023-07-01      |
+------------+-----------+-----------------+
1 row in set (0.00 sec)
```

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.

```
mysql> SELECT Students.first_name, Students.last_name
    -> FROM Students
    -> LEFT JOIN Payments ON Students.student_id = Payments.student_id
    -> WHERE Payments.student_id IS NULL;
Empty set (0.00 sec)
```

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.

```
mysql> SELECT Courses.course_name
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> WHERE Enrollments.course_id IS NULL;
+------------------+
| course_name      |
+------------------+
| Computer Science |
+------------------+
1 row in set (0.00 sec)
```

Task 4:

1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.

```
mysql> SELECT customer_id, MAX(account_balance) as highest_balance
    -> FROM accounts
    -> GROUP BY customer_id
    -> ORDER BY highest_balance DESC
    -> LIMIT 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.

```
mysql> SELECT course_id, AVG(enrollment_count) as avg_students_enrolled
    -> FROM (
    ->     SELECT course_id, COUNT(student_id) as enrollment_count
    ->     FROM Enrollments
    ->     GROUP BY course_id
    -> ) as enrollments_per_course
    -> GROUP BY course_id;
+-----------+----------------------+
| course_id | avg_students_enrolled |
+-----------+----------------------+
|         1 |               1.0000 |
|         2 |               1.0000 |
|         3 |               1.0000 |
|         4 |               1.0000 |
|         5 |               1.0000 |
|         6 |               1.0000 |
|         7 |               1.0000 |
|         8 |               1.0000 |
|         9 |               1.0000 |
+-----------+----------------------+
9 rows in set (0.00 sec)
```

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

```
mysql> SELECT student_id, first_name, last_name, amount
    -> FROM Students
    -> JOIN Payments ON Students.student_id = Payments.student_id
    -> WHERE amount = (SELECT MAX(amount) FROM Payments);
```

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

```
mysql> SELECT course_id, course_name, enrollment_count
    -> FROM (
    ->     SELECT course_id, course_name, COUNT(student_id) as enrollment_count
    ->     FROM Courses
    ->     JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    ->     GROUP BY course_id, course_name
    -> ) as enrollments_per_course
    -> WHERE enrollment_count = (SELECT MAX(enrollment_count) FROM enrollments_per_cour
se);
```

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

```
mysql> SELECT student_id, first_name, last_name
    -> FROM Students
    -> WHERE (SELECT COUNT(DISTINCT course_id) FROM Courses) = (
    ->     SELECT COUNT(DISTINCT course_id)
    ->     FROM Enrollments
    ->     WHERE Students.student_id = Enrollments.student_id
    -> );
Empty set (0.00 sec)
```

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

```
mysql> SELECT teacher_id, first_name, last_name
    -> FROM Teacher
    -> WHERE teacher_id NOT IN (SELECT DISTINCT teacher_id FROM Courses);
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          1 | Jane       | Smith     |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT AVG(age) as average_age
    -> FROM (
    ->     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) as age
    ->     FROM Students
    -> ) as student_ages;
+-------------+
| average_age |
+-------------+
|     30.9000 |
+-------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT course_id, course_name
    -> FROM Courses
    -> WHERE course_id NOT IN (SELECT DISTINCT course_id FROM Enrollments);
+-----------+------------------+
| course_id | course_name      |
+-----------+------------------+
|        10 | Computer Science |
+-----------+------------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT Students.student_id, first_name, last_name, Courses.course_id, course_nam
e, SUM(amount) as total_payments
    -> FROM Students
    -> JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> JOIN Courses ON Enrollments.course_id = Courses.course_id
    -> JOIN Payments ON Students.student_id = Payments.student_id
    -> GROUP BY Students.student_id, first_name, last_name, Courses.course_id, course_n
ame;
+------------+------------+-----------+-----------+-------------------------------+----
------------+
| student_id | first_name | last_name | course_id | course_name                   | tot
al_payments |
+------------+------------+-----------+-----------+-------------------------------+----
------------+
|          2 | Arun       | Kumar     |         2 | Electrical Engineering        |
     600.00 |
|          3 | Priya      | Menon     |         3 | Mechanical Engineering        |
     750.00 |
|          4 | Rajesh     | Nair      |         4 | Civil Engineering             |
     900.00 |
|          5 | Deepa      | Sundar    |         5 | Chemical Engineering          |
     550.00 |
|          6 | Vijay      | Chandran  |         6 | Information Technology        |
     700.00 |
|          7 | Lakshmi    | Krishnan  |         7 | Electronics and Communication |
     800.00 |
|          8 | Suresh     | Rao       |         8 | Aerospace Engineering         |
     950.00 |
|          9 | Anjali     | Pillai    |         9 | Biotechnology                 |
     500.00 |
|          1 | John       | Doe       |         1 | Computer Science              |
     700.00 |
+------------+------------+-----------+-----------+-------------------------------+----
------------+
9 rows in set (0.00 sec)
```

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.

```
mysql> SELECT Students.student_id, first_name, last_name, SUM(amount) as total_payments

    -> FROM Students
    -> JOIN Payments ON Students.student_id = Payments.student_id
    -> GROUP BY Students.student_id, first_name, last_name;
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|          1 | John       | Doe       |         700.00 |
|          2 | Arun       | Kumar     |         600.00 |
|          3 | Priya      | Menon     |         750.00 |
|          4 | Rajesh     | Nair      |         900.00 |
|          5 | Deepa      | Sundar    |         550.00 |
|          6 | Vijay      | Chandran  |         700.00 |
|          7 | Lakshmi    | Krishnan  |         800.00 |
|          8 | Suresh     | Rao       |         950.00 |
|          9 | Anjali     | Pillai    |         500.00 |
|         10 | Karthik    | Menon     |         850.00 |
+------------+------------+-----------+----------------+
10 rows in set (0.00 sec)
```

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.

```
mysql> SELECT student_id, first_name, last_name
    -> FROM Students
    -> JOIN Payments ON Students.student_id = Payments.student_id
    -> GROUP BY Students.student_id, first_name, last_name
    -> HAVING COUNT(Payments.payment_id) > 1;
```

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.

```
mysql> SELECT Courses.course_id, course_name, COUNT(Enrollments.student_id) as enrolled
_students
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> GROUP BY Courses.course_id, course_name;
+-----------+-----------------------------+------------------+
| course_id | course_name                 | enrolled_students |
+-----------+-----------------------------+------------------+
|         1 | Computer Science            |                1 |
|         2 | Electrical Engineering      |                1 |
|         3 | Mechanical Engineering      |                1 |
|         4 | Civil Engineering           |                1 |
|         5 | Chemical Engineering        |                1 |
|         6 | Information Technology      |                1 |
|         7 | Electronics and Communication |                1 |
|         8 | Aerospace Engineering       |                1 |
|         9 | Biotechnology               |                1 |
|        10 | Computer Science            |                0 |
+-----------+-----------------------------+------------------+
10 rows in set (0.00 sec)
```

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.

```
mysql> SELECT AVG(amount) as average_payment_amount
    -> FROM Payments;
+------------------------+
| average_payment_amount |
+------------------------+
|             730.000000 |
+------------------------+
1 row in set (0.00 sec)
```

CREATING AN ERD FOR THE ABOVE DATABASE :