Student Course Management Database

The Student Course Management project involves the creation of a database to manage student enrollments in various courses. The database is designed with four primary tables: Students, Courses, Instructors, and Enrollments. The project also includes inserting sample data, running queries to retrieve information, and implementing stored procedures and functions.

Database Creation

```
SQL statement:
CREATE DATABASE StudentCourseManagement;
USE StudentCourseManagement;
```

`Students` Table

```
CREATE TABLE Students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    date_of_birth DATE
);
```

`Courses` Table

```
CREATE TABLE Courses (
    course_id INT PRIMARY KEY AUTO_INCREMENT,
    course_name VARCHAR(100),
    course_description TEXT
);
```

`Instructors` Table

```
CREATE TABLE Instructors (
    instructor_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100)
);
```

`Enrollments` Table

```
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    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));
```

3. Data Insertion

```
-- Inserting data into Students table

INSERT INTO Students (student_id, first_name, last_name, email,
date_of_birth)
VALUES
(10123, 'Ali', 'Ali', 'ali.ali@gmail.com', '2000-01-15'),
(10234, 'Fatima', 'Ahmad', 'fatima.ahmad@gmail.com', '1999-05-22'),
(10345, 'Mariam', 'Barakat', 'mariam.barakat@gmail.com', '2001-03-11'),
(10456, 'Sara', 'Hosseini', 'sara.hosseini@gmail.com', '1998-07-09'),
(10567, 'Youssef', 'Omari', 'youssef.omari@gmail.com', '2002-11-30'),
(10678, 'Saeed', 'Nasser', 'saeed.alnasser@gmail.com', '2000-02-18'),
(10789, 'Jamila', 'Zaid', 'jamila.alzaid@gmail.com', '1999-08-14'),
(10890, 'Ibrahim', 'Fatih', 'ibrahim.alfatih@gmail.com', '2001-12-05'),
(10901, 'Hala', 'Sadiq', 'hala.alsadiq@gmail.com', '2003-04-28'),
(11012, 'Rashid', 'Madani', 'rashid.almadani@gmail.com', '2000-06-17');

-- Inserting data into Courses table
INSERT INTO Courses (course_id, course_name, course_description)
VALUES
(30512, 'Mathematics', 'Study of numbers and shapes.'),
(40987, 'Physics', 'Study of matter and energy.'),
(21874, 'Chemistry', 'Study of substances and their interactions.'),
(52301, 'Biology', 'Study of living organisms.'),
(63429, 'Computer Science', 'Study of computers and computational systems.');

-- Inserting data into Instructors table
INSERT INTO Instructors (instructor_id, first_name, last_name, email)
VALUES
(41234, 'Ahmed', 'Hassan', 'ahmed.alhassan@gmail.com'),
(52345, 'Laila', 'Mansoor', 'fatima.almansoor@gmail.com'),
(63456, 'Omar', 'Sayed', 'omar.alsayed@gmail.com');

-- Inserting data into Enrollments table
INSERT INTO Enrollments (enrollment_id, student_id, course_id,
enrollment_date)
VALUES
(43821, 10123, 30512, '2024-01-10'),
(59234, 10234, 40987, '2024-01-15'),
(48375, 10345, 21874, '2024-01-20'),
(15792, 10456, 52301, '2024-02-01'),
(67289, 10567, 63429, '2024-02-05'),
(32817, 10678, 30512, '2024-02-10'),
(84920, 10789, 40987, '2024-02-15'),
(71524, 10890, 21874, '2024-03-01'),
(29573, 10901, 52301, '2024-03-05'),
(64831, 11012, 63429, '2024-03-10'),
(43928, 10123, 40987, '2024-04-01'),
(58273, 10234, 21874, '2024-04-05'),
(72839, 10345, 52301, '2024-04-10'),
(16492, 10456, 63429, '2024-05-01'),
(73948, 10567, 30512, '2024-05-05');
```

SQL File 4* | SQLAdditions

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	21:08:22	CREATE DATABASE StudentCourseManagement	1 row(s) affected	0.031 sec
2	21:08:22	USE StudentCourseManagement	0 row(s) affected	0.000 sec
3	21:08:22	CREATE TABLE Students (student_id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR(...	0 row(s) affected	0.016 sec
4	21:08:22	CREATE TABLE Courses (course_id INT PRIMARY KEY AUTO_INCREMENT, course_name VARCHA...	0 row(s) affected	0.031 sec
5	21:08:22	CREATE TABLE Instructors (instructor_id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCH...	0 row(s) affected	0.016 sec
6	21:08:22	CREATE TABLE Enrollments (enrollment_id INT PRIMARY KEY AUTO_INCREMENT, student_id INT, ...	0 row(s) affected	0.047 sec
7	21:08:22	INSERT INTO Students (student_id, first_name, last_name, email, date_of_birth) VALUES (10123, 'Ali', 'Ali', 'ali...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.015 sec
8	21:08:22	INSERT INTO Courses (course_id, course_name, course_description) VALUES (30512, 'Mathematics', 'Study ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
9	21:08:22	INSERT INTO Instructors (instructor_id, first_name, last_name, email) VALUES (41234, 'Ahmed', 'Hassan', 'ah...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
10	21:08:22	INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES (43821, 10123, 3...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.016 sec
11	21:08:22	SELECT * FROM Students LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
12	21:08:22	SELECT * FROM Courses LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
13	21:08:22	SELECT E.enrollment_id, S.first_name, S.last_name, C.course_name FROM Enrollments E JOIN Students S O...	15 row(s) returned	0.000 sec / 0.000 sec
14	21:08:22	SELECT S.* FROM Students S JOIN Enrollments E ON S.student_id = E.student_id WHERE E.course_id = 4...	3 row(s) returned	0.000 sec / 0.000 sec
15	21:08:22	SELECT C.course_name FROM Courses C JOIN Enrollments E ON C.course_id = E.course_id GROUP BY C.c...	5 row(s) returned	0.000 sec / 0.000 sec
16	21:08:22	UPDATE Students SET email = 'myemail@gmail.com' WHERE student_id = 10345	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
17	21:08:22	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.000 sec
18	21:08:22	DELETE FROM Courses WHERE course_id NOT IN (SELECT DISTINCT course_id FROM Enrollments)	0 row(s) affected	0.000 sec
19	21:08:22	SELECT AVG(YEAR(CURDATE()) - YEAR(date_of_birth)) AS avg_age FROM Students LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
20	21:08:22	SELECT C.course_name FROM Courses C JOIN Enrollments E ON C.course_id = E.course_id GROUP BY C.c...	1 row(s) returned	0.000 sec / 0.000 sec
21	21:08:22	SELECT C.course_name, COUNT(E.student_id) AS num_students FROM Courses C LEFT JOIN Enrollments E...	5 row(s) returned	0.000 sec / 0.000 sec
22	21:08:22	SELECT S.first_name, S.last_name, C.course_name FROM Students S JOIN Enrollments E ON S.student_id = ...	15 row(s) returned	0.000 sec / 0.000 sec
23	21:08:22	SELECT S.first_name, S.last_name FROM Students S LEFT JOIN Enrollments E ON S.student_id = E.student_...	0 row(s) returned	
24	21:08:22	SELECT S.* FROM Students S WHERE S.student_id IN (SELECT E.student_id FROM Enrollments E ...	5 row(s) returned	
25	21:08:22	SELECT S.first_name, S.last_name, COUNT(E.course_id) AS num_enrollments FROM Students S JOIN Enroll...	3 row(s) returned	
26	21:08:22	CREATE PROCEDURE AddStudent(IN fname VARCHAR(50), IN lname VARCHAR(50), IN email VA...	0 row(s) affected	
27	21:08:22	CREATE FUNCTION CalculateAge(dob DATE) RETURNS INT DETERMINISTIC BEGIN RETURN YEAR(...	0 row(s) affected	
28	21:08:22	SELECT COUNT(*) AS total_students FROM Students LIMIT 0, 1000	1 row(s) returned	

Battery saver

Battery saver is on
Consider plugging in your device.

4. Querying the Database

a. Selecting All Students

```
SELECT * FROM Students;

SELECT * FROM Courses;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

student_id	first_name	last_name	email	date_of_birth
10001	John	Doe	john.doe@example.com	2000-01-15
10002	Jane	Smith	jane.smith@example.com	1999-05-22
10003	Michael	Brown	michael.brown@example.com	2001-03-11
10004	Emily	Davis	emily.davis@example.com	1998-07-09
10005	Sarah	Wilson	sarah.wilson@example.com	2002-11-30
10006	David	Lee	david.lee@example.com	2000-02-18
10007	Sophia	Martinez	sophia.martinez@example.com	1999-08-14
10008	James	Taylor	james.taylor@example.com	2001-12-05
10009	Olivia	Anderson	olivia.anderson@example.com	2003-04-28
10010	William	Thomas	william.thomas@example.com	2000-06-17
NULL	NULL	NULL	NULL	NULL

b. Selecting All Courses

```
SELECT * FROM Courses;

SELECT E.enrollment_id, S.first_name, S.last_name, C.course_name
FROM Enrollments E
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

course_id	course_name	course_description
20001	Mathematics	Study of numbers and shapes.
20002	Physics	Study of matter and energy.
20003	Chemistry	Study of substances and their interactions.
20004	Biology	Study of living organisms.
20005	Computer Science	Study of computers and computational systems.
NULL	NULL	NULL

c. Selecting all enrollments with student first and last names and course name

```
SELECT E.enrollment_id, S.first_name, S.last_name, C.course_name
FROM Enrollments E
JOIN Students S ON E.student_id = S.student_id
JOIN Courses C ON E.course_id = C.course_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

enrollment_id	first_name	last_name	course_name
40001	John	Doe	Mathematics
40006	David	Lee	Mathematics
40015	Sarah	Wilson	Mathematics
40002	Jane	Smith	Physics
40007	Sophia	Martinez	Physics
40011	John	Doe	Physics
40003	Michael	Brown	Chemistry
40008	James	Taylor	Chemistry
40012	Jane	Smith	Chemistry
40004	Emily	Davis	Biology
40009	Olivia	Anderson	Biology
40013	Michael	Brown	Biology
40005	Sarah	Wilson	Computer Sc...
40010	William	Thomas	Computer Sc...

Result 3

d. Finding Students Enrolled in a Specific Course

```
SELECT S.*
FROM Students S
JOIN Enrollments E ON S.student_id = E.student_id
WHERE E.course_id = 40987;

SELECT C.course_name
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	email	date_of_birth
10123	Ali	Ali	ali.ali@gmail.com	2000-01-15
10234	Fatima	Ahmad	fatima.ahmad@gmail.com	1999-05-22
10789	Jamila	Zaid	jamila.alzaid@gmail.com	1999-08-14

e. Counting Courses with More Than Two Enrollments

```
SELECT C.course_name
FROM Courses C
JOIN Enrollments E ON C.course_id = E.course_id
GROUP BY C.course_name
HAVING COUNT(E.student_id) > 2;

UPDATE Students
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_name
Chemistry
Mathematics
Physics
Biology
Computer Science

f. Updating Student Email

```
UPDATE Students

SET email = 'myemail@gmail.com'

WHERE student_id = 10345;

SET SQL_SAFE_UPDATES = 0;
```

g. Deleting Courses with No Enrollments

```
DELETE FROM Courses

WHERE course_id NOT IN (

    SELECT DISTINCT course_id

    FROM Enrollments);
```

h. Calculating Average Age of Students

```
SELECT AVG(YEAR(CURDATE()) - YEAR(date_of_birth)) AS avg_age
FROM Students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg_age
23.7000

i. Finding the Course with the Most Enrollments

```
SELECT C.course_name
FROM Courses C
JOIN Enrollments E ON C.course_id = E.course_id
GROUP BY C.course_name
ORDER BY COUNT(E.student_id) DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

course_name
Chemistry

j. List courses along with the number of students enrolled

```sql
SELECT C.course_name, COUNT(E.student_id) AS num_students
FROM Courses C
LEFT JOIN Enrollments E ON C.course_id = E.course_id
GROUP BY C.course_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_name	num_students
Mathematics	3
Physics	3
Chemistry	3
Biology	3
Computer Science	3

Select all students with their enrolled course

```sql
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;

SELECT I.first_name, I.last_name, C.course_name
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	course_name
John	Doe	Mathematics
David	Lee	Mathematics
Sarah	Wilson	Mathematics
Jane	Smith	Physics
Sophia	Martinez	Physics
John	Doe	Physics
Michael	Brown	Chemistry
James	Taylor	Chemistry
Jane	Smith	Chemistry
Emily	Davis	Biology
Olivia	Anderson	Biology
Michael	Brown	Biology
Sarah	Wilson	Computer Sc...
William	Thomas	Computer Sc

k. Find Students who are not enrolled in any course

```sql
SELECT 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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name

l. Finding Students Enrolled in More Than One Course

```sql
FROM Students S
WHERE S.student_id IN (
    SELECT E.student_id
    FROM Enrollments E
    GROUP BY E.student_id
    HAVING COUNT(E.course_id) > 1
);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

student_id	first_name	last_name	email	date_of_birth
10001	John	Doe	john.doe@example.com	2000-01-15
10002	Jane	Smith	jane.smith@example.com	1999-05-22
10003	Michael	Brown	michael.brown@example.com	2001-03-11
10004	Emily	Davis	emily.davis@example.com	1998-07-09
10005	Sarah	Wilson	sarah.wilson@example.com	2002-11-30
NULL	NULL	NULL	NULL	NULL

m. Finding Top 3 Students with the Most Enrollments

```sql
SELECT S.first_name, S.last_name, COUNT(E.course_id) AS num_enrollments
FROM Students S
JOIN Enrollments E ON S.student_id = E.student_id
GROUP BY S.student_id
ORDER BY num_enrollments DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

first_name	last_name	num_enrollments
John	Doe	2
Jane	Smith	2
Michael	Brown	2

5. Stored Procedures and Functions

a. Add a Student

```
DELIMITER $$
CREATE PROCEDURE AddStudent(
    IN fname VARCHAR(50),
    IN lname VARCHAR(50),
    IN email VARCHAR(100),
    IN dob DATE
)
BEGIN
    INSERT INTO Students (first_name, last_name, email, date_of_birth)
    VALUES (fname, lname, email, dob);
END $$
DELIMITER ;
```

b.Calculate Age

```
DELIMITER $$
CREATE FUNCTION CalculateAge(dob DATE)
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN YEAR(CURDATE()) - YEAR(dob);
END $$
DELIMITER ;
```

6. Additional Queries

a. Counting Total Students

```
171
172 •   SELECT COUNT(*) AS total_students FROM Students;
173
174 •   SELECT AVG(num_students), MIN(num_students), MAX(num_students)
175     FROM (
176         SELECT COUNT(E.student_id) AS num_students
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_students
10

b. Calculating Enrollment Statistics

```sql
SELECT AVG(num_students), MIN(num_students), MAX(num_students)
FROM (
    SELECT COUNT(E.student_id) AS num_students
    FROM Enrollments E
    GROUP BY E.course_id
) AS EnrollmentStats;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVG(num_students)	MIN(num_students)	MAX(num_students)
3.0000	3	3

c. Categorizing Students by Age

```sql
SELECT first_name, last_name,
CASE
    WHEN CalculateAge(date_of_birth) < 20 THEN 'Teenager'
    WHEN CalculateAge(date_of_birth) BETWEEN 20 AND 30 THEN 'Young Adult'
    ELSE 'Adult'
END AS age_category
FROM Students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	age_category
Ali	Ali	Young Adult
Fatima	Ahmad	Young Adult
Mariam	Barakat	Young Adult
Sara	Hosseini	Young Adult
Youssef	Omari	Young Adult
Saeed	Nasser	Young Adult
Jamila	Zaid	Young Adult
Ibrahim	Fatih	Young Adult
Hala	Sadiq	Young Adult
Rashid	Madani	Young Adult

d. Checking for Course Enrollments

```sql
SELECT course_name
FROM Courses C
WHERE EXISTS (
    SELECT 1
    FROM Enrollments E
    WHERE E.course_id = C.course_id
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_name
Mathematics
Physics
Chemistry
Biology
Computer Science